Departamento de Ingeniería de Sistemas y Automática

UNIVERSIDAD DE MÁLAGA

Planificación de Trayectorias en Sistemas Multirrobot

Tesis doctoral presentada por Dª Ana M. Cruz Martín para optar al grado de Doctora Ingeniera en Informática

Dirigida por el Dr. D. Alfonso José García Cerezo, Catedrático de Universidad, y por el Dr. D. Víctor Fernando Muñoz Fernández, Profesor Titular de Universidad, ambos del Área de Ingeniería de Sistemas y Automática.

Málaga, Diciembre de 2004

UNIVERSIDAD DE MÁLAGA
DEPARTAMENTO DE INGENIERÍA DE
SISTEMAS Y AUTOMÁTICA

Dr. D. Alfonso J. García Cerezo, Catedrático de Universidad, y **Dr. D. Víctor F. Muñoz Martínez**, Profesor Titular de Universidad, adscritos ambos al Departamento de Ingeniería de Sistemas y Automática de la Universidad de Málaga,

CERTIFICAN

Que la memoria titulada "Planificación de Trayectorias en Sistemas Multirrobot" ha sido realizada por Dª Ana María Cruz Martín bajo su dirección en el Departamento de Ingeniería de Sistemas y Automática de la Universidad de Málaga. Esta memoria constituye la Tesis que Dª Ana María Cruz Martín presenta para optar al Grado de Doctora.

Málaga, a 21 de Julio de 2004

Dr. D. Alfonso J. García Cerezo Dr. D. Víctor F. Muñoz Martínez.

Vº Bº El Director del Departamento
Fdo. Dr. Alfonso J. García Cerezo

A mis padres y mi hermana, por lo que soy.

A Juan Antonio, que hace que todo valga la pena, siempre.

Agradecimientos

Afortunadamente, son muchas las personas a las que puedo dar las gracias por haber llegado hasta aquí. Creo que ése es el premio más valioso que puedo obtener de este trabajo.

En primer lugar se encuentran mis directores de tesis, los doctores D. Víctor F. Muñoz Martínez y D. Alfonso J. García Cerezo, puesto que han sido ellos los que me han iniciado en el mundo de la investigación, y han guiado mis primeros pasos, cuyo resultado aquí presento. Espero que su impronta haya quedado reflejada a lo largo de estas páginas. Sobre todo, quiero agradecerles la confianza y el apoyo que siempre me han mostrado, que son lo que más valoro.

También me gustaría dar las gracias a todos los compañeros del Departamento de Ingeniería de Sistemas y Automática de la Universidad de Málaga, cuyo trabajo conjunto ha dado lugar al hardware y al software sobre los que se ha desarrollado esta tesis. En este esfuerzo común también ha intervenido la Dr. Dª Raquel Fernández Ramos, cuya cooperación y ánimo han sido muy importantes para mí.

Mis padres, Adolfo y Ana, y mi hermana Elena, también tienen mucho que ver en este trabajo, ya que sin su entusiasmo en los momentos buenos, y su energía, apoyo incondicional y ejemplo en los momentos malos, mi tesis seguramente no habría visto la luz. Gracias por estar ahí siempre y para todo.

Todos mis amigos, tanto de La Línea como de Málaga y, en general, de cualquier parte, me han demostrado su amistad e interés continuamente a lo largo de los años que ha durado la realización de esta tesis. En ellos se incluyen el Departamento de Internet de Telenor Media España, primero bajo la dirección de Miguel Ángel García y Arturo de Luiz, y después bajo la dirección de Mundo Gutiérrez y Clara Medina, con los que durante año y medio compartí aprendizaje, trabajo y experiencias. Muchas gracias a todos, y ojalá no os pierda nunca.

Y, por supuesto, gracias a Juan Antonio. Tú más que nadie sabes todo el esfuerzo que hay detrás de esta tesis. Y quiero que sepas que tus inteligentes consejos, tu enorme e interesante experiencia, tu inacabable paciencia y tu cariño constante han sido los pilares en los que me he sostenido para llevarla a cabo. Espero poder seguir aprendiendo de ti, y contigo, durante muchos años.

Síntesis

En esta tesis se aborda el problema de la planificación de trayectorias en sistemas robóticos compuestos por múltiples elementos. Como punto de partida, se toma un algoritmo de planificación de velocidades para un único robot que genera trayectorias cuyos perfiles de velocidad tienen en cuenta las limitaciones de velocidad, tanto físicas como operacionales, que influyen de manera significativa en el movimiento del vehículo. Dicho algoritmo monorrobot se extiende al caso multirrobot, proporcionando una trayectoria segura a cada uno de los vehículos que integran el sistema. La planificación de trayectorias multirrobot se ha incorporado a la arquitectura de control del robot autónomo móvil Auriga-α.

La planificación de trayectorias multirrobot planteada admite diferentes soluciones, cuyo grado de bondad será también diferente. Por tanto, resolver el problema convenientemente no implica hallar cualquier solución, sino determinar la mejor solución según ciertos criterios que puedan establecerse para cada sistema multirrobot. Este trabajo analiza en profundidad qué tipo de herramienta sería la más apropiada para determinar la ordenación que proporcione la ordenación óptima, llegando a la conclusión de que los algoritmos genéticos resultan una buena elección, e implantando también un algoritmo genético que establece el orden de prioridades en el que se aplica el algoritmo de planificación de trayectorias multirrobot.

Para facilitar las tareas de experimentación y pruebas se ha desarrollado un entorno gráfico de simulación de sistemas multirrobot, lo que evita la necesidad de emplear un sistema multirrobot real, que habitualmente resulta muy costoso y complejo de organizar y manejar.

A todo ello se añade una recopilación pormenorizada de las diferentes arquitecturas de sistemas multirrobot existentes en la literatura hasta la fecha, así como el estudio, bastante amplio, y la posterior clasificación de las estrategias de navegación deliberativa que pueden encontrarse en la actualidad para este tipo de sistemas.

INDICE

CAPÍTULO 1. Introducción

1.1 Del robot a los sistemas multirrobot

El siglo XX fue testigo del nacimiento del término *robot*, que se define en el Diccionario de la Real Academia de la Lengua Española como la *máquina o ingenio electrónico programable capaz de manipular objetos y realizar operaciones antes reservadas sólo a las personas*. Asimismo, también se incluye una definición para *Robótica*, como la *técnica que aplica la informática al diseño y empleo de aparatos que, en sustitución de personas, realizan operaciones o trabajos, por lo general en instalaciones industriales*.

La Robótica experimentó una rápida evolución a lo largo del siglo pasado, y hoy en día puede considerarse una disciplina científica plenamente asentada, de la que están surgiendo nuevas áreas de conocimiento. Una de estas nuevas ramas se centra en el estudio de sistemas compuestos por varios robots, los denominados sistemas *multirrobot*[1]. A lo largo de este apartado se analizarán, con mayor profundidad, los aspectos más característicos y destacables de este tipo de sistemas.

1.1.1. Sistemas multirrobot

Un sistema multirrobot puede definirse como un conjunto de robots situados en un entorno común. Esta definición, tan genérica, engloba una gran diversidad de sistemas, que pueden presentar importantes diferencias entre sí. Existen algunas taxonomías que organizan los sistemas multirrobot en función de ciertas características o ejes principales; aunque estas clasificaciones se estudiarán más detalladamente en el segundo capítulo, a continuación se exponen algunos rasgos de este tipo de sistemas que pueden dar una idea aproximada de las variaciones que son posibles en su diseño y construcción:

1. El vocablo inglés *multirobot* o *multi-robot* se ha traducido al castellano como *multirrobot*, tras consulta a la Real Academia de la Lengua Española, respetando las reglas ortográficas y de formación de palabras del castellano.

- *Tipos de robot.* Dependiendo de los tipos de robots que los forman, los sistemas multirrobot podrían clasificarse en tres grandes grupos. Por una parte, los sistemas denominados *estáticos*, compuestos por elementos como manipuladores industriales, brazos mecánicos, manos provistas de varios dedos, etc. Otro tipo de sistemas multirrobot estaría formado por *robots móviles*, ya sean terrestres (para entornos cerrados o para entornos exteriores), aéreos o submarinos. El tercer grupo sería el correspondiente a los sistemas multirrobot *híbridos*, en los que sus elementos pertenecen a distintos tipos, como por ejemplo, un robot móvil que incluya un brazo manipulador, como el presentado en (Martínez Sánchez, 2001). También serían posibles, dentro de los sistemas multirrobot móviles, tipos híbridos compuestos, por ejemplo, por varios robots terrestres coordinados con vehículos aéreos. Todo ello se refleja en la Figura 1.

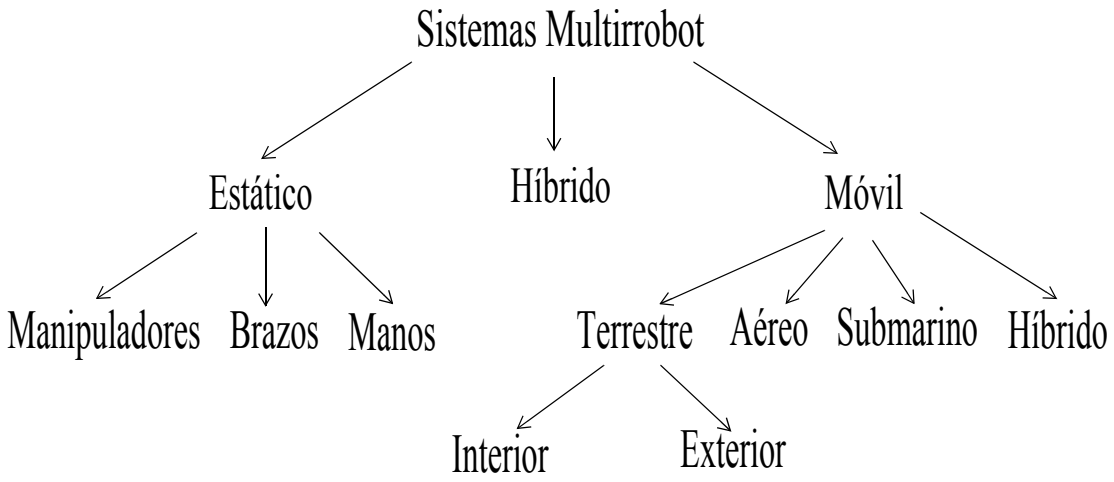


Figura 1.1. Clasificación de sistemas multirrobot según los tipos de robot que participen.

Por otra parte, además de pertenecer a un cierto tipo, los robots de un sistema pueden ser *homogéneos*, si sus capacidades son las mismas, o *heterogéneos*, en caso contrario.

Los sistemas de robots móviles terrestres[2] y, en menor medida, los formados por manipuladores, son los sistemas multirrobot más estudiados; los primeros se denominan a veces enjambres (*swarms*), colonias (*colonies*), o colectivos (*collectives*), lo que indica que la Biología y la Etología han sido la base inspiradora de muchos de estos desarrollos.

- *Interacción surgida entre los elementos del sistema.* El hecho de compartir un entorno provoca la aparición de relaciones entre los componentes del sistema, es decir, surge un comportamiento colectivo (Cao *et al*., 1997). Dicho comportamiento puede tomar varias formas:
    - Apatía o indiferencia, es decir, la tarea de cada robot es independiente de la de los demás, y puede ser llevada a cabo sin relacionarse con el resto de integrantes, y sin que una posible colaboración aporte beneficios.
    - Cooperación. De manera general, la cooperación se define como la asociación de sujetos para lograr un fin común, aunque en la bibliografía relativa a sistemas multirrobot aparecen trabajos que aportan diferentes matices a esta idea básica. Ejemplos de comportamientos colectivos cooperativos son la recolección (*foraging*), en la que los robots buscan y recogen objetos del entorno, o el apacentamiento, cuyo objetivo se centra en que los caminos de los vehículos cubran todo el entorno (*grazing)*.
    - Competición o antagonismo. Si en los comportamientos colectivos indiferentes y cooperativos las metas de los robots son armonizables, en la competición los objetivos de los elementos del sistema son incompatibles entre sí. Aquí entrarían las técnicas de persecución-evasión, o juegos competitivos como el fútbol robótico o *soccer.*

2. Habitualmente, se identifican los robots móviles terrestres con el término más genérico de robots móviles. A lo largo de esta tesis se utilizarán ambas nomenclaturas, indicándose cuando sea necesario que el vehículo móvil no es terrestre.

Sin embargo, a veces los límites entre cooperación y competición son un tanto difusos (Balch, 1998a); por ejemplo, la recolección es una tarea típicamente cooperativa, pero en la que los elementos del sistema están compitiendo por los recursos del mismo, que en este caso son los elementos a recoger; por otra parte, el fútbol es una tarea competitiva, pero si los equipos de fútbol están formados por más de un integrante, deben cooperar entre ellos para vencer al contrario.

Las ventajas de utilizar un sistema multirrobot dependerán del tipo de tarea que se lleve a cabo y del entorno en el que se desarrolle; ésto implica que no siempre es útil emplear sistemas multirrobot, y puede ocurrir que su utilización no aporte beneficios. Normalmente, un sistema multirrobot resulta provechoso en situaciones como las que siguen:

- Tareas con un ámbito geográfico extenso que requieran algún tipo de sincronización, ya que un único robot no podría realizarlas. Por ejemplo, activar simultáneamente cuatro interruptores situados en las esquinas de una habitación grande.
- División del trabajo en porciones que se asignen a los elementos del sistema, de manera que se concluya antes y, por tanto, aumente el rendimiento global.
- Un grupo de robots simples pueden ser una mejor opción, en términos de coste, flexibilidad, facilidad de manejo y tolerancia a fallos, que un único robot más capaz y costoso.
- Posibilidad de analizar con un sistema multirrobot modelos sociales, biológicos, o de computación distribuida, que permitan profundizar en dichas ciencias.
- En entornos industriales, el hecho de contar con varios manipuladores en una misma celda de fabricación permite ampliar las posibilidades del manufacturado, así como incrementar la eficiencia (Costa i Castelló *et al*., 1995).

Junto con el apacentamiento y la recolección mencionados con anterioridad, algunas tareas típicas en las que tradicionalmente se han empleado sistemas multirrobot son: control de tráfico, manipulación cooperativa/*box-pushing*, lucha contra incendios, fútbol robótico, problemas de transporte, exploración y reconocimiento de entornos, sistemas industriales con varios manipuladores, mantenimiento de formación, o

aplicaciones militares como la desactivación de minas. Las posibilidades de diseño y construcción de un sistema multirrobot son muy amplias, incluso dentro de un dominio concreto; en el segundo capítulo se introducirán algunos de los sistemas multirrobot existentes más relevantes, así como taxonomías que ayudan a su clasificación.

Los sistemas multirrobot están afianzándose poco a poco, tanto en el mundo científico como en el industrial. Un factor importante para este avance ha sido el desarrollo tecnológico, aunque aún existen ciertas restricciones que complican un tanto su expansión, como limitaciones de percepción, o la dificultad de mantener operativas flotas de robots (Cao *et al*., 1997). Para conocer cómo se encuentra el estado del arte relativo a los sistemas multirrobot, pueden consultarse las referencias (Arai *et al.*, 2002) y (Parker, 2000).

1.1.2. Relación entre sistemas multirrobot y sistemas multiagente

Los sistemas multirrobot también aparecen mencionados en la literatura como sistemas robóticos multiagente o robótica multiagente, es decir, se tiende a identificar o asemejar los términos "multirrobot" y "multiagente", si bien esto no es siempre correcto. Los Sistemas Multi-Agente son una rama de la Inteligencia Artificial Distribuida (*Distributed Artificial Intelligence, DAI*), que a su vez es un área de la Inteligencia Artificial surgida hace tres décadas. La DAI, que puede definirse como el estudio de la concurrencia en Inteligencia Artificial, se divide tradicionalmente en dos campos: la Resolución de Problemas Distribuida (*Distributed Problem Solving, DSP*) y los Sistemas Multiagente (*Multiagent Systems, MAS*). Ambas se describen más detalladamente a continuación:

- La investigación en DSP se centra en establecer la solución de un problema usando varios agentes; aspectos importantes a examinar serían la descomposición del problema inicial en tareas, la asignación de tareas a los distintos agentes, o la síntesis de la solución. En DSP se supone que cada agente desea colaborar para obtener el resultado final, es decir, están predispuestos a la cooperación.

- Los MAS también intentan resolver un problema mediante varios agentes. Sin embargo, frente a la aproximación de DSP, los elementos de un MAS no están inicialmente resueltos a cooperar, sino que poseen sus propios intereses y objetivos. Como introducción a los MAS pueden consultarse los trabajos expuestos en (Stone y Veloso, 2000; Cao *et al*., 1997 y Visser, 2002).

Entonces, si los MAS son una disciplina conocida, ¿por qué se utilizan indistintamente "multirrobot" y "multiagente"? El motivo radica en que no existe una definición formal de agente globalmente aceptada; como ejemplo, sirvan éstas encontradas en la literatura:

- Para (Wooldrigde y Ciancarini, 2001), un agente es un sistema que disfruta de las siguientes propiedades:
    - Autonomía: los agentes encapsulan algún estado (no accesible a otros agentes), y toman decisiones sobre qué hacer basándose en dicho estado, sin la intervención directa de humanos u otros agentes.
    - Capacidad de reacción (*reactivity*): los agentes están situados en un entorno, (que puede ser el mundo real, un usuario a través de un interfaz gráfico de usuario, una colección de otros agentes, Internet, o puede que una combinación de los anteriores), son capaces de percibir este entorno (usando sensores potencialmente imperfectos), y son capaces de responder de manera oportuna a los cambios que ocurren en él.
    - Pro-actividad (*pro-activeness*): los agentes no actúan simplemente en respuesta a su entorno, sino que pueden exhibir un comportamiento dirigido a objetivos tomando la iniciativa.
    - Sociabilidad: los agentes interaccionan con otros agentes (y posiblemente con humanos) mediante algún tipo de lenguaje de comunicación de agentes, y típicamente poseen la habilidad de comprometerse en actividades sociales (como resolución cooperativa de problemas o negociación) para lograr sus objetivos.
- Tras analizar múltiples definiciones de agentes, (Franklin y Graesser, 1996) definen un agente autónomo como un sistema situado dentro de un entorno, como parte integrante de él y con capacidad de percibirlo y actuar sobre el mismo, a lo largo del tiempo, buscando su propia agenda, para afectar a lo que perciba en el futuro.

Así, a menudo se asume que un robot es un agente, y esto no siempre es cierto. Podría ocurrir que careciera de alguno de los aspectos esenciales del concepto "agente" (objetivos, autonomía, interacción con el entorno), y por tanto no podría ser considerado como tal. Por ejemplo, supongamos un sistema compuesto por varios robots móviles con capacidades sensoriales, que envían la información percibida a un ordenador central,

encargado de analizarla y tomar las decisiones oportunas, que serán enviadas de vuelta a los vehículos. Si bien estaríamos ante un sistema multirrobot, no podríamos denominarlo multi-agente, ya que salvo el computador maestro, ninguno de sus integrantes es autónomo ni tiene sus propios objetivos.

1.2 Planificación de trayectorias en robots móviles

Los primeros avances de la Robótica se dieron en el ámbito industrial -la idea de robot se asoció en su origen al trabajo y la producción- como confluencia de tres tecnologías: el control de máquinas herramientas, los manipuladores teleoperados, y los ordenadores (Ollero, 1991). Durante los años sesenta, la introducción de los robots manipuladores en la industria se convierte en un hecho, a la par que se profundiza en su estudio y mejora. Debido a que el proceso de producción cada vez requiere mayor flexibilidad, a los manipuladores industriales se les unen los vehículos móviles. Aunque inicialmente las posibilidades de tales robots estaban bastante restringidas, paulatinamente fueron incrementándose hasta llegar al concepto actual de robot móvil: robots con capacidad de movimiento sobre entornos no estructurados, de los que posee un conocimiento incierto, mediante la interpretación de la información suministrada a través de sus sensores y del estado actual del vehículo (Muñoz, 1995). El siguiente paso, en el que la tecnología se encuentra actualmente en curso, consiste en dotar de autonomía a los robots móviles, entendiendo como autonomía la facultad del robot de percibir el entorno y transformar dicha información en acciones que le permitan ejecutar su tarea. Una de las acciones que caracterizan a un robot móvil autónomo es la navegación.

La navegación puede definirse como la tarea de guiado de un robot móvil conforme atraviesa su entorno, teniendo como principal objetivo la seguridad, es decir, la ausencia de colisiones con elementos situados en sus proximidades (Muñoz *et al*., 1999). Son varios los planteamientos posibles a la hora de abordar este problema se presentan a continuación, ordenados cronológicamente:

- Navegación estratégica, planificada o deliberativa. Este enfoque genera el plan de movimientos del vehículo a priori, mediante la ejecución de un bucle de tres pasos:
    - Percepción del entorno, que permite construir un modelo del mismo a partir de la información suministrada por los sensores.
    - Planificación del camino, es decir, obtención de aquellos puntos del espacio por los que el robot debe pasar para llegar a su objetivo,

usando el modelo del mundo anterior.

- Seguimiento del camino planificado, para lo que existen diferentes soluciones como la persecución pura o *pure-pursuit*, los polinomios quínticos, etc.

La ventaja de la navegación deliberativa radica en que, al utilizar información del entorno antes de planificar el camino, puede generar caminos óptimos bajo algún criterio (tiempo, distancia, energía,...). Sin embargo, esta ventaja se transforma en un problema desde el punto de vista de la flexibilidad, ya que el camino planificado difícilmente podrá adaptarse a situaciones no previstas en el modelo del mundo. El segundo y el tercer capítulo del presente trabajo examinan más cuidadosamente esta técnica, tanto en el caso monorrobot como multirrobot, por lo que en ellos se incluyen las referencias pertinentes.

- Navegación reactiva. En este tipo de navegación, el vehículo reacciona de manera dinámica y con rapidez ante cambios en el entorno. La navegación reactiva muestra su utilidad en entornos cambiantes, de los que no se posee un conocimiento preciso, pero tiene como contrapartida la imposibilidad de mirar al pasado y al futuro, lo que dificulta la consecución de objetivos globales previamente fijados. Basándose en esta aproximación, en los últimos años han empezado a tomar relevancia los llamados sistemas basados en comportamientos (*behavior-based systems*), que pretenden suplir las carencias de la reactividad, sin dejar a un lado sus ventajas. Los sistemas basados en comportamientos toman su nombre de los elementos básicos sobre los que se apoyan: los comportamientos, que pueden definirse como "patrones de actividad que emergen de interacciones entre el robot y su entorno" (Mataric, 2001). Para conocer con mayor profundidad algunos trabajos destacados o novedosos asociados a esta clase de navegación, pueden consultarse las referencias (Khatib, 1986; Borenstein y Koren, 1989; Arkin, 1989; Brooks, 1986; Stenzel, 2000; Seraji y Howard, 2002; Emery y Balch, 2001).

- Navegación híbrida. Las últimas tendencias a la hora de diseñar estrategias de navegación pretenden aprovechar las ventajas asociadas tanto a la navegación planificada como a la reactiva. Así, la navegación híbrida utiliza, para el alto nivel, la percepción y la planificación propias de la navegación deliberativa, mientras que el bajo nivel se construye en función de

comportamientos que permitan una reacción eficiente. Un ejemplo de estrategia de navegación híbrida sería la empleada en las arquitecturas AuRA (Arkin y Balch, 1997), Saphira (Konolige y Myers, 1998), o la propuesta por Kant y Zucker (Kant y Zucker, 1988).

Dentro de la estrategia de navegación planificada, la mayoría de los métodos de planificación de caminos suponen que dichos caminos se seguirán con una velocidad constante y lo suficientemente baja como para evitar que las características físicas del vehículo (cinemática, dinámica) influyan significativamente en su seguimiento. Puesto que el robot necesita una cierta aceleración inicial (y una deceleración final) para alcanzar la referencia de velocidad impuesta, ésta se pondera según una atenuación variable. Sin embargo, esta suposición no es realista, ya que tanto las características físicas del vehículo, como las restricciones a las que pueda estar sometido debido a requerimientos operacionales, determinan unos valores máximos de velocidad que el robot podrá desarrollar. Al problema de la síntesis del movimiento de un vehículo bajo limitaciones cinemáticas y dinámicas simultáneas se le denomina Problema de la Planificación Cinemático-Dinámica o *Kinodynamic Planning Problem* (Canny *et al.*, 1988). Por tanto, para que la navegación deliberativa aproveche mejor sus posibilidades, la etapa de planificación de caminos debe incorporar también la construcción de un perfil de velocidad asociado al camino, denominándose entonces *planificación de trayectorias*.

Si la navegación es una tarea de gran importancia para un robot que trabaje de manera aislada, cobra especial protagonismo en el caso de los sistemas multirrobot, ya que cada vehículo comparte el entorno con sus compañeros, que por tanto actúan como obstáculos móviles. Por tanto, las técnicas de navegación en sistemas multirrobot deben considerar este hecho a la hora de definir trayectorias para los distintos robots, garantizando que el movimiento de los mismos sea seguro.

1.3 Contribuciones de la tesis

Esta tesis ofrece resultados, tanto teóricos como experimentales, relativos al problema de la planificación de trayectorias en dos dimensiones para sistemas multirrobot formados por robots móviles terrestres. Se parte de la base de que cada vehículo tendrá asociado un camino, compuesto por referencias geométricas que tendrá que alcanzar. Sobre este perfil espacial se construirá un perfil temporal que indique en qué momento deberán alcanzarse las distintas posiciones, teniendo en cuenta las restricciones de velocidad que afectan al movimiento del vehículo. La siguiente etapa consiste en combinar todas las

trayectorias calculadas, basándose en un esquema de prioridades, de manera que los robots puedan moverse simultáneamente sin colisiones. Las contribuciones de la tesis se centran, por tanto, en los aspectos del cálculo de velocidades y la combinación de trayectorias; entre ellas cabe destacar las siguientes:

i) Análisis y clasificación de las diferentes metodologías existentes para la navegación planificada de sistemas formados por robots móviles.

Son muchas las aproximaciones posibles a la hora de plantear el movimiento de grupos de robots móviles terrestres. Es por ello por lo que se ha considerado interesante estudiar las diferencias existentes entre las diversas soluciones que pueden hallarse en la actualidad, y clasificarlas de acuerdo con ciertos criterios. Esta catalogación se plantea desde dos puntos de vista: arquitecturas multirrobot, y estrategias de navegación planificadas. De esta manera, se facilita la comprensión de los objetivos y las aportaciones de cada una de ellas.

ii) Generación y evaluación de diferentes algoritmos para la construcción de perfiles de velocidad.

Se presentan tres algoritmos para el cálculo de velocidades que consideran, tanto las características físicas del vehículo, como las de la tareas que lleva a cabo, incluyendo la presencia de obstáculos móviles en el entorno. A partir del método de generación de velocidades de Muñoz (Muñoz, 1995), se desarrollan tres posibles soluciones, cada una evolución de la anterior; será la última de ellas la que se utilice en la implantación definitiva del sistema multirrobot.

iii) Adaptación del problema de la planificación de trayectorias al caso multirrobot.

El algoritmo de planificación de trayectorias que se presenta permite su extensión al caso multirrobot, si se consideran como obstáculos móviles los vehículos que integran el sistema multirrobot. Así, aplicando un conjunto de prioridades, pueden determinarse las trayectorias para cada uno de los elementos del sistema, que pueden ejecutar su camino con la máxima velocidad permisible en cada instante. La necesidad de incluir mecanismos que resuelvan situaciones inesperadas también se evalúan en este trabajo.

iv) **Obtención de la mejor solución posible en el sistema multirrobot mediante un algoritmo genético.**

La planificación de trayectorias multirrobot puede realizarse empleando ordenamientos diferentes, que darán lugar a soluciones distintas al evaluarlos bajo ciertos criterios definidos para el sistema. Una vez que tales medidas de bondad se establecen, resulta lógico hallar el esquema de prioridades que mejores trayectorias genere para los vehículos. Son varias las posibilidades que existen a la hora de calcular la priorización; tras analizar el problema detalladamente, se ha optado por implantar un algoritmo genético que se encargue de dicha tarea.

v) **Diseño y construcción de un simulador para navegación de sistemas multirrobot.**

Puesto que la experimentación real con sistemas multirrobot complejos es una cuestión costosa, se ha creado una herramienta software que facilite la tarea a la hora de implantar y estudiar sistemas multirrobot; sus características de flexibilidad y generalidad permiten que pueda adaptarse a diferentes tipos de plataformas hardware y software de manera sencilla.

1.4 Marco de realización

El trabajo presentado en esta tesis queda determinado por las actividades de investigación que lleva a cabo, en la línea de la Robótica Móvil, el Grupo de Ingeniería de Sistemas y Automática de la Universidad de Málaga; concretamente, en el diseño y construcción de los robots móviles Auriga-α, Aurora y RAM-2.

La creación del robot Auriga-α ha sido dirigida por el Dr. García Cerezo, contando con una subvención de la Comisión Interministerial de Ciencia y Tecnología (CICYT), dentro del proyecto “Control Inteligente en Robótica Móvil (CIRO)”, TAP-1184-C04-02.

Parte de la tesis ha podido elaborarse gracias a la Beca de Colaboración con Cargo a Créditos de Investigación de la Universidad de Málaga, en la que se trabajó durante los meses de Mayo y Junio de 2001, y a la Beca de Formación de Personal Docente e Investigador de la Consejería de Educación y Ciencia de la Junta de Andalucía, disfrutada por la autora desde Julio de 2001 hasta Febrero de 2002.

Como plataforma hardware y banco de pruebas para experimentos se ha empleado el robot móvil autónomo Auriga-α, desarrollado íntegramente en el Dpto. de Ingeniería de Sistemas y Automática de la Universidad de Málaga. Respecto a la parte software, los algoritmos para la planificación de velocidades se han escrito en lenguaje C, incluyéndose en la arquitectura de control del Auriga-α mediante la extensión NEXUS -diseñada e implantada también en el mismo Dpto. (Fdez. Madrigal y González, 1999)- para el sistema operativo de tiempo real Lynx (Lynx Real-Time Systems Inc., 1993). En las tareas auxiliares de simulación y representación gráfica de datos, así como para el desarrollo del entorno de simulación multirrobot SIMUNA, se ha empleado el programa de cálculo matemático MATLAB (Matlab v6R12, 2002).

Por último, señalar que a lo largo de la tesis todas las magnitudes están expresadas en el Sistema Internacional de Unidades, a no ser que se indique lo contrario explícitamente.

1.5 Organización de la tesis

La tesis está dividida en seis capítulos, tres apéndices y las referencias bibliográficas. Salvo el presente capítulo introductorio, y el dedicado a las conclusiones y futuros desarrollos, los restantes capítulos comienzan con una introducción, que plantea brevemente el problema a estudiar junto con una descripción del contenido, y finalizan con las conclusiones obtenidas a lo largo de su desarrollo.

El segundo capítulo, **Trabajos Relacionados**, se destina a situar convenientemente el problema estudiado en la tesis. Así, por una parte se presentan las principales taxonomías de clasificación de sistemas multirrobot, incluyéndose también algunas de las arquitecturas más conocidas. Por otra parte, se clasifican, desde el punto de vista de la planificación del movimiento, varias metodologías de navegación multirrobot, analizando sus similitudes y diferencias para establecer con mayor claridad qué aporta la solución propuesta.

En el tercer capítulo, **Planificación de Trayectorias para Robots Móviles**, se plantea la necesidad de usar referencias temporales en la navegación de un robot, debido a la existencia de limitaciones de velocidad que rigen su movimiento. Se describe el algoritmo básico del que se ha partido, así como ciertas mejoras que sobre él se efectúan.

El cuarto capítulo, **Planificación de Trayectorias para un Sistema Multirrobot**, explica cómo el algoritmo presentado en el capítulo anterior puede ser extendido a un sistema multirrobot de vehículos; para ello, se proponen tres soluciones para la planificación de trayectorias, y se analizan los resultados obtenidos y las ventajas que aportan. También se incluyen ciertos aspectos relevantes del problema que es necesario tener en cuenta para su completa adaptación a este tipo de sistemas robóticos.

Una vez establecido el algoritmo de planificación de trayectorias multirrobot, se escoge la mejor solución en el capítulo quinto, **Planificación Óptima de Trayectorias Multirrobot**. La elección de un algoritmo genético como herramienta para determinar la solución cuasi-óptima se justifica convenientemente, y también se describe cómo se ha realizado la implantación del mismo, así como los resultados obtenidos tras su aplicación.

El sexto y último capítulo, **Conclusiones y Desarrollos Futuros**, resume los elementos más destacables del trabajo presentado, así como las posibles líneas de investigación derivadas del mismo.

El Apéndice A, **El Robot Autónomo Móvil Auriga-α**, se destina a la descripción del banco de pruebas utilizado para la parte experimental de la tesis, tanto desde el punto de vista físico, como del software que integra su arquitectura de control.

El Apéndice B, denominado **Entorno de Simulación SIMUNA**, presenta un sistema genérico diseñado para simular sistemas multirrobot, y la implantación que del mismo se ha hecho mediante la aplicación MATLAB.

Por último, el Apéndice C, se dedica a la **Arquitectura Robótica Implantada sobre NEXUS**. Así, se ofrece una visión introductoria al sistema de desarrollo de software robótico distribuido NEXUS, incluyendo su evolución posterior denominada BABEL. También se comentan los diferentes elementos creados sobre NEXUS para construir la arquitectura robótica del robot Auriga-α, añadiendo una descripción detallada del módulo encargado de la planificación de trayectorias.

CAPÍTULO 2. Trabajos Relacionados

2.1 Introducción

Antes de abordar la metodología de planificación de trayectorias para robots móviles presentada en esta tesis, y su posterior integración en un sistema multirrobot, resulta conveniente situar ambas cuestiones dentro de los principales trabajos, de similares características, que pueden hallarse en la literatura. Este análisis se ha planteado a dos niveles de profundidad. Por una parte, desde un enfoque global, se examinarán las clasificaciones existentes para sistemas multirrobot/multiagente[3] más notables, a partir de las cuales se mostrarán ciertos rasgos destacables de los mismos, que abarcan aspectos de alto nivel (por ejemplo, toma de decisiones) y bajo nivel (por ejemplo, ancho de banda de las comunicaciones). Esta visión general se cerrará con un abanico de arquitecturas multirrobot ya implantadas.

A continuación, se fijará la atención en la tarea de navegación que los robots móviles de un sistema multirrobot deben llevar a cabo, ya que es el principal objetivo a desarrollar en el sistema multirrobot expuesto en este trabajo; así, se ofrecerá una catalogación de diferentes técnicas de navegación multirrobot ya existentes, utilizando como criterio el mecanismo de planificación de movimiento del vehículo que emplean. Tal clasificación podrá contrastarse con el algoritmo propuesto, y así establecer cuáles son las aportaciones del mismo.

Siguiendo este esquema, el capítulo se organiza en torno a tres apartados fundamentales. El apartado 2.2 explica y compara varias taxonomías de sistemas multirrobot propuestas en la literatura, incluyendo algunas que, si bien no indexan sistemas multirrobot, sí están relacionadas con los mismos, mostrando otros aspectos que también resulta interesante catalogar. El apartado 2.3 describe algunas de las arquitecturas multirrobot que gozan actualmente de mayor relevancia. Respecto a las estrategias de

3. Como ya se indicó en el Capítulo 1, los términos multirrobot y multiagente se emplean comúnmente sin distinción. Las taxonomías que en este apartado se exponen se refieren tanto a sistemas multirrobot como a sistemas robóticos multiagente; por claridad, y mientras no se indique explícitamente lo contrario, se agruparán ambas nomenclaturas bajo el concepto multirrobot.

navegación para este tipo de sistemas, será el apartado 2.4 el destinado a su estudio y clasificación. Como cierre, en el apartado 2.5 se muestran las conclusiones extraídas, que permiten ubicar tanto el sistema multirrobot como el planificador de trayectorias implementados en esta tesis, dentro de sus respectivas áreas.

2.2 Taxonomías de Sistemas Multirrobot

Puesto que son muchas y muy variadas las soluciones que se han planteado, en los últimos años, a la hora de controlar y gestionar un sistema multirrobot, algunos autores han propuesto taxonomías o sistemas de clasificación de estos sistemas que permiten organizar y facilitar la comprensión de sus peculiaridades. La elaboración de taxonomías aporta las siguientes ventajas:

- Definen las características clave que identifican, de manera más precisa y completa, diferentes sistemas multirrobot.
- Permiten establecer relaciones, análisis y pruebas formales para los grupos que de ellas se derivan.
- Abren la posibilidad de realizar comparativas entre sistemas de manera sencilla.
- El espacio posible de diseños de sistemas multirrobot queda acotado; así, al desarrollar un nuevo sistema pueden conocerse aspectos ya cubiertos por soluciones anteriores, y buscar aquellas contribuciones que resulten inéditas.

Son factibles, dentro del dominio multirrobot, diferentes taxonomías, dependiendo de la elección de los ejes de clasificación y del subconjunto concreto sobre el que se apliquen (por ejemplo, sistemas multirrobot compuestos por robots móviles). A continuación se analizarán, con mayor detalle, algunas de las taxonomías de sistemas multirrobot más destacadas. Algunas son específicas para ellos, mientras que otras se refieren a sistemas multiagentes, aunque pueden extenderse al caso robótico. Como resumen pueden consultarse las características más significativas de las clasificaciones comentadas en la Tabla 2.1

Taxonomía de Dudek

La clasificación propuesta por (Dudek *et al*., 1996) se aplica a sistemas multirrobot en general. Utiliza como base para la ordenación las llamadas *dimensiones naturales*, que se refieren a las propiedades del colectivo de robots como conjunto, más que a las características relativas a la arquitectura de cada individuo. Se definen siete ejes de clasificación, que abarcan aspectos muy concretos del sistema: el tamaño del colectivo, diferentes criterios relativos a las comunicaciones (rango, topología, ancho de banda), la reconfigurabilidad del sistema, la capacidad de proceso de cada elemento, y la composición del grupo, referido al grado de homogeneidad hardware y software; dentro de cada eje sólo son posibles un número limitado de valores). Además de proponer la taxonomía, los autores muestran la aplicación práctica de la misma. Por una parte, clasifican con ella algunos problemas tipo en sistemas multirrobot, así como varias de las arquitecturas existentes más notables. Y, por otra parte, la utilizan para probar formalmente que, en ciertas tareas, la eficiencia de un colectivo supera a la de un robot individual.

Taxonomía de Cao

Otra de las taxonomías más conocidas corresponde a la desarrollada por (*Cao et al*., 1997), y que a diferencia de la anterior, circunscribe su ámbito a la robótica móvil cooperativa. También difiere el enfoque ofrecido para la clasificación: organiza la literatura existente basándose en “problemas y soluciones” relativos al mecanismo de cooperación en un sistema multirrobot. Es decir, se plantean cuáles son cuestiones principales a resolver, y qué mecanismos se han encontrado para solventarlas. En concreto, se definen como ejes taxonómicos la arquitectura de grupo, los conflictos por recursos, los orígenes de la cooperación, el aprendizaje y los problemas geométricos; sin embargo, al ser muy generales, algunos de estos criterios revelan cuestiones complejas si se analizan más detalladamente. Así, dentro de la arquitectura de grupo aparecen asuntos como la diferenciación o grado de homogeneidad, la centralización/descentralización del sistema, o las estructuras de comunicación empleadas. Profundizando en los conflictos por recursos, se observa que pueden darse a la hora de compartir objetos, medios de comunicación o espacio; este último caso, el más común, ha dado origen al estudio de técnicas para la planificación de caminos multirrobot y la evitación de colisiones y bloqueos. Por último, en los problemas geométricos, se incluyen el mantenimiento de marcha y formación y, de nuevo, la planificación de caminos multirrobot, ya que también puede definirse bajo esta condición. El aspecto positivo de esta clasificación radica, sobre todo, en que los criterios definidos comprenden gran parte de la investigación realizada hasta la fecha para determinar los mecanismos de cooperación. Sin embargo, como se ha comentado, son ejes interdependientes y muy amplios, lo que puede dificultar las clasificaciones.

Taxonomía de Yuta y Premvuti

La clasificación propuesta por (Yuta y Premvuti, 1992) para sistemas multirrobot autónomos divide a los mismos según sus objetivos sean comunes o no. A partir de este único criterio se declaran tres categorías posibles: sistemas de un único robot (caso degenerado), sistemas con un objetivo común, y sistemas con objetivos individuales. El primer caso constaría de un robot sin idea de la existencia de otros, para el que cualquier otro elemento móvil del entorno es sólo un obstáculo que debe tener en cuenta a la hora de navegar sin riesgos. La segunda situación, en la que todos los integrantes del sistema poseen un objetivo común, puede resolverse mediante alguna solución de tipo centralizado, y por ello no se considera como un sistema completamente autónomo, sino distribuido. Por último, dentro de la clase de robots con objetivos individuales, la interferencia entre los mismos puede resolverse bajo dos mecanismos de decisión, generándose otro eje taxonómico. Por una parte, el robot puede ayudar a otro elemento del sistema, bien sin que medie una petición previa (ayuda activa), bien tras una solicitud (en ese caso, debe decidir si ayudar es lo más conveniente); esta alianza supone que, momentáneamente, los robots abandonan sus tareas individuales en pos de un objetivo común. Por otra parte, los robots pueden ser autónomos y guiarse por el paradigma de la *cooperación moderada* o *modest cooperation* propuesta por los autores, que se resume en el lema "no molestar a los demás"; las molestias posibles o *disturbances* se refieren a conflictos por recursos como herramientas, medios de comunicación y espacio. Intercalando el uso de ambos mecanismos de decisión (de objetivo común y autónomo), se plantea una metodología de cooperación para resolver la tarea de navegación de un sistema multirrobot compuesto por vehículos móviles.

Taxonomía de Stone y Veloso

Aunque con un carácter más general, al estar orientada a sistemas multiagente no necesariamente robóticos, también resulta interesante la taxonomía realizada por Stone y Veloso (Stone y Veloso, 2000). En ella se definen únicamente dos criterios de clasificación: el grado de homogeneidad, y el grado de comunicación. Los agentes son homogéneos si su estructura (sensores, efectores, conocimiento del dominio, funciones de decisión) son idénticos. El grado de comunicación indica si los agentes pueden comunicarse de manera directa entre ellos, y no mediante interacción con el entorno o *estigmergía*[4]. Todos los sistemas multiagente, incluidos los sistemas multirrobot, pueden englobarse en los cuatro tipos de sistemas resultantes, que son, desde el más simple al más complejo: agentes

4. *Estigmergía* es un concepto, tomado de la Biología, que se refiere al hecho de que los agentes pueden tener efectos sobre el entorno que sirvan como indicios determinantes para el comportamiento de otros agentes. Puede ser *sematectónica*, si implica un cambio en las características físicas del ambiente, o *basada en señales*, en la que ciertos elementos se depositan en el entorno, de manera que influyen en la actuación de otros individuos.

homogéneos sin comunicación, agentes heterogéneos sin comunicación, agentes homogéneos con comunicación y agentes heterogéneos con comunicación. Como dominio apropiado sobre el que estudiar las clases de sistemas multiagente definidas, se propone el fútbol robótico -un sistema multirrobot, a la postre-, debido a que contiene muchos aspectos complejos del mundo real, lo que lo convierte en un banco de pruebas idóneo para los MAS.

Taxonomía de Fulbright y Stevens

También se enfoca bajo un prisma multiagente la taxonomía fijada por (Fulbright y Stephens, 1994), que se construye basándose en cómo los agentes comparten recursos o se *acoplan*. Los agentes se definen, desde un punto de vista funcional, como entidades capaces de percibir (sentir e interpretar el entorno), razonar (mantener la información interna y decidir qué acciones emprender según la situación) y actuar (pudiendo modificar el entorno); estas tres capacidades son los recursos a compartir. Según la capacidad que se comparta, surgen tres posibles clasificaciones. Por una parte, la *clasificación de control*, que depende de que la capacidad de actuación sea común o no, y que divide a los agentes en distribuidos y centralizados, siendo casos extremos los agentes autónomos y los sistemas con un único agente, respectivamente. La *clasificación de inferencia*, basada en la división de la aptitud de razonamiento, divide los sistemas multiagente en agentes no-inferenciales, en los que las características necesarias para llevar a cabo el razonamiento están repartidas, y los agentes inferenciales, que sí pueden razonar independientemente de los demás. Por último, la *clasificación de percepción* se basa en la capacidad de percibir del agente: se consideran agentes no perceptuales aquéllos que no son autosuficientes a la hora de sentir el entorno en el que se mueven, y agentes perceptuales los que sí lo son.

Taxonomía de Visser

Otra forma de ordenar MAS, expuesta por Visser en (Visser, 2002), consiste en identificar diferentes arquitecturas posibles para este tipo de sistemas, entendiendo como arquitectura la manera de organizar los agentes del mismo. Siguiendo este criterio, se distinguen siete arquitecturas posibles: control centralizado de un único agente, políticas compartidas de selección de acciones, distribución de tareas entre agentes, agentes locales que no comparten información global, sistemas de agentes con jerarquía, agentes con un modelo global del mundo común, y agentes con comunicación. Para cada una de ellas, se estudia la complejidad de su política de decisión, entendiendo como complejidad de una arquitectura el tamaño total del espacio de políticas posible. Además, este autor aporta también una clasificación de problemas multiagente, que permite seleccionar la arquitectura o arquitecturas más apropiadas para un problema concreto.

Taxonomía de Todt

Restringiéndose a la coordinación de movimientos entre múltiples robots, (Todt *et al.*, 2000) definen cinco ejes taxonómicos: coordinación acoplada/desacoplada, tiempo de coordinación en línea o fuera de línea, existencia de prioridades, criterios de coste, y espacio de trabajo definido. Si bien se trata de una clasificación de bajo nivel, tiene la ventaja de ser aplicable tanto a robots móviles como a manipuladores -no son muchas las taxonomías referidas a sistemas multirrobot en entornos industriales-, y ofrece una revisión bastante amplia de métodos conocidos para la sincronización de desplazamientos en entornos multirrobot.

Taxonomía de Balch

Para aquellos equipos robóticos que emplean técnicas de aprendizaje por refuerzo, (Balch, 1998a) caracteriza tanto la tarea a realizar como la función de recompensa del aprendizaje, de forma que se pueda escoger el tipo de aprendizaje más apropiado para una tarea determinada, y establecer cómo las propiedades de un refuerzo concreto pueden incrementar o disminuir la respuesta del sistema. La taxonomía propuesta para caracterizar las tareas consta de 6 ejes, referidos al tiempo disponible para la ejecución de la misma, el horizonte de tiempo en el que la eficiencia del sistema debe optimizarse, el objeto de la tarea, los límites en los recursos externos, el tipo de movimiento del grupo -siguiendo los criterios expuestos en (Ali, 1999)-, y las capacidades del sistema. Para las recompensas, son cinco los ejes escogidos: la fuente de la recompensa, su relación con la eficiencia, momento en el que se proporciona, recompensa continua/discreta, y globalidad/individualidad de la misma. En ambos casos, al igual que en la taxonomía de Dudek, sólo se permiten ciertos valores para cada eje.

Taxonomía de Ali

Siguiendo con la idea de clasificar tareas, (Ali, 1999) se centra en las que puede llevar a cabo un sistema multiagente robótico móvil. Este autor categoriza las tareas únicamente en función del movimiento relativo de sus elementos, definiendo tres ejes para ello. El primero, Cobertura vs. Convergencia (*Coverage*/*Convergence*), indica si los vehículos tienden a expandirse en el entorno, o por el contrario, se agrupan. El segundo, Movimiento-hacia vs. Movimiento-mientras (*Movement-to*/*Movement-while-maintaining*), expresa cómo han de moverse los robots para lograr la Cobertura o la Convergencia. Por último, Posiciones-conocidas vs. Posiciones-desconocidas (*Known-positions*/*Unknown-positions*) refleja si los vehículos conocen las posiciones, relativas a los demás, que deben mantener. A partir de estas propiedades surgen ocho categorías posibles, a las que se añade una más (otros-movimientos/*other-movement-types*), para aquellos casos que no pueden caracterizarse mediante las mismas.

Conclusión

De lo descrito hasta ahora, se desprende que las clasificaciones son muy diversas, y comprenden características muy variadas, dependiendo del dominio sobre el que trabajen, o la orientación que desee proporcionársele. Sin embargo, sí que es posible distinguir algunos aspectos comunes que aparecen en la mayoría de las taxonomías:

- diferenciación de los integrantes del sistema multirrobot.
- centralización/descentralización del mismo.
- cómo se establece la comunicación entre sus elementos, aunque esta cuestión se aborde desde múltiples perspectivas.
- cómo éstos se mueven en el entorno común.

Por tanto, parece seguro que estos cuatro puntos son relevantes y deben ser tenidos en cuenta, de alguna manera, a la hora de definir y/o clasificar sistemas multirrobot.

Taxonomía	Dominio	Nº Ejes	Descripción
Dudek	Multirrobot	7	Basada en las características del conjunto de robots.
Cao	Robótica móvil cooperativa	5	Basada en problemas y soluciones del mecanismo de cooperación.
Yuta y Premvuti	Multirrobot	2	Definida a partir de los objetivos y los mecanismos de decisión.
Stone y Veloso	Multiagente	2	Estudia la homogeneidad de los agentes y su grado de comunicación.
Fulbright y Stevens	Multiagente	3	Establece tres clasificaciones según el acoplamiento de los agentes.

Tabla 2.1: Taxonomías Multirrobot

Taxonomía	Dominio	Nº Ejes	Descripción
Visser	Multiagente	7	Organiza arquitecturas de sistemas multiagente.
Todt	Multirrobot	5	Limitada a la coordinación de movimientos entre robots.
Balch	Tareas y recompensas	6/5	Útil en sistemas que empleen aprendizaje por refuerzo.
Ali	Tareas para sistemas multirrobot móviles	3	Referida a las características del movimiento relativo de los vehículos.

Tabla 2.1: Taxonomías Multirrobot

2.3 Principales arquitecturas multirrobot

Tras analizar las taxonomías multirrobot más destacadas de la literatura actual, resulta interesante estudiar con mayor detalle algunos de los colectivos que dichas clasificaciones organizan. Los elementos de tales colectivos se apoyan en una estructura hardware/software propia que de manera genérica se denomina arquitectura, y que debe tener en cuenta las peculiaridades de un sistema multirrobot; es decir, una arquitectura robótica diseñada para un único robot no tiene por qué ser válida para ese mismo robot, si éste se agrega a un sistema multirrobot. Por tanto, sería conveniente precisar qué se entiende como *arquitectura multirrobot*, ya que, como se señala en (Coste-Maniére y Simmons, 1997) refiriéndose a sistemas robóticos, la concepción correcta de una arquitectura mejora el desarrollo del sistema, puesto que ayuda en las etapas de especificación, implantación y validación del mismo.

Sin embargo, sólo una de las taxonomías anteriores, la sugerida por Cao para un sistema robótico cooperativo, propone una idea que podría aproximarse a la definición buscada: la arquitectura de grupo. Así, basándose en el eje del mismo nombre, se establece que la arquitectura de grupo "proporciona la infraestructura sobre la que se implantan los comportamientos colectivos, y determina las capacidades y limitaciones del sistema". Esta definición, aunque abarca tanto al software como al hardware, únicamente indica las

funciones de la arquitectura, no sus elementos ni cómo realizar la integración de los mismos. En toda la bibliografía explorada, no se ha encontrado ninguna otra mención a este término, pero sí diferentes soluciones ya implantadas. Esto puede deberse a que los sistemas multirrobot son un campo de investigación reciente, y no existe aún el suficiente bagaje teórico como para poder formalizar los resultados obtenidos; el hecho de que, como se concluye del apartado anterior, no se ha podido consensuar una taxonomía común, refuerza esta idea. Para suplir esta carencia, puede tomarse como base el análisis de los cuatro tipos tradicionales de arquitecturas monorrobot (deliberativa, reactiva, híbrida y basada en comportamientos) realizado por Mataric (Mataric, 2001), del que se deduce que las arquitecturas híbridas y, sobre todo, las basadas en comportamientos, son las que mejor se adaptan al caso multirrobot.

Los siguientes subapartados describen, en orden cronológico aproximado, las características más destacables de las arquitecturas multirrobot con mayor relevancia.

CEBOT

CEBOT (CEllular roBOTics) es una arquitectura robótica distribuida flexible, con aplicaciones en campos tan diversos como la industria, el espacio, o entornos hostiles, y para la que se ha diseñado tanto el software como el hardware (Fukuda *et al*., 1988; Fukuda *et al*., 1990; Kawauchi *et al*., 1992). Fuertemente inspirada por la Biología, CEBOT está formada por *células* básicas, que se unen en *módulos*, que a su vez pueden formar *estructuras*. El paralelismo con la organización de una criatura multicelular queda patente: a partir de las células, se generan los tejidos (módulos), que constituyen la base de los órganos (estructuras). Existen diferentes tipos de células, que se agrupan o separan dinámica y automáticamente dependiendo de la tarea, permitiendo incluso la reparación o sustitución de células dañadas, lo que permite que la arquitectura sea tolerante a fallos. Cada célula tiene su propia inteligencia (conocimiento y base de datos), y puede llevar a cabo una tarea dependiendo de su conocimiento y del conocimiento de las demás, lo que implica que deben existir mecanismos de comunicación entre células. Como se aprecia en la bibliografía seleccionada, CEBOT ha ido evolucionando, y si en sus orígenes estuvo orientada hacia la operación como brazo manipulador, ha ido ampliando su perspectiva, y en la actualidad el objetivo es lograr un sistema que simule un órgano vivo: auto-reparable, auto-evolutivo y auto-organizable.

ACTRESS

ACTor-based Robots and Equipments Synthetic System, o ACTRESS (Asama *et al*., 1989; Matsumoto *et al*., 1990), es un sistema robótico distribuido y autónomo, diseñado para la automatización de tareas de alto nivel. Basado en el formalismo ACTOR (Hewitt *et al*., 1973), sustituye los actores de éste por *robotors* o actores robóticos. Cada robotor es un componente autónomo, con al menos dos funcionalidades básicas. Por una parte, debe poseer la habilidad de tomar decisiones, reconocer el entorno, actuar, y gestionar sus propias condiciones. Por otra parte, será capaz de comunicarse con otros componentes del sistema. Aquellos dispositivos que cumplan estas condiciones, como robots móviles, manipuladores, sensores inteligentes, actuadores inteligentes, equipos especiales, etc., podrán ser considerados robotors, por lo que es posible trabajar simultáneamente con elementos heterogéneos.

INTELIGENCIA DE ENJAMBRE

Si bien no puede considerarse como una arquitectura propiamente dicha, se ha incluido la inteligencia de enjambre o *swarm intelligence* (Beni y Wang, 1989) dentro de esta enumeración, al ser la base sobre la que se apoyan algunas arquitecturas multirrobot. La inteligencia de enjambre es una propiedad de sistemas formados por agentes sin inteligencia, con capacidades individuales limitadas, pero que exhiben un comportamiento colectivo inteligente; normalmente, éste se logra mediante estigmergía. Las hormigas constituyen un ejemplo de un sistema natural con inteligencia de enjambre: mientras que los individuos son insectos simples con memoria limitada y actividad estocástica, el grupo es capaz de realizar tareas complicadas. La inteligencia de enjambre surge de la interacción entre N unidades sólo cuando N supera un valor crítico N_c; es decir, para desarrollar un trabajo complejo, es necesario tener un cierto número de elementos, que además tienen que cooperar. Otros autores (Sugawara y Watanabe, 2002) han suavizado esta definición, estableciendo el concepto de *inteligencia de enjambre generalizada*, en la que se considera que el trabajo efectuado por N individuos cooperantes es mayor que la suma del trabajo realizado por N elementos independientes. Junto con problemas teóricos que las inteligencias de enjambre deben resolver (Beni y Wang, 1991), también se han estudiado áreas importantes, como la autoorganización (Hackwood and Beni, 1992), las comunicaciones (Wang y Beni, 1990), o la posibilidad de utilizar varios enjambres (White y Pagurek, 1998).

GOFER

El proyecto GOFER (Caloud *et al*., 1990) investiga la operación de un grupo de robots móviles -hasta varias docenas-, en entornos interiores, con el fin de automatizar una serie de tareas (transporte de objetos, limpieza, operación de maquinaria...). Además de la arquitectura software, este proyecto también contempla cuestiones hardware, como el diseño de los propios vehículos o las comunicaciones robot-robot y robot-humano. Respecto al software, son tres los puntos más destacables que GOFER aborda: la planificación de las tareas, su distribución, y la planificación de movimientos multirrobot. Para la planificación de tareas se emplean las estructuras de plan (*plan structures*), que agrupan jerarquías de acciones parametrizadas y restricciones que deben cumplirse, y las instancias de plan (*plan instances*), en las que las variables de las estructuras de plan se sustituyen por valores que verifiquen las limitaciones especificadas. El mecanismo de distribución de tareas se basa en el protocolo *Contract Net*, pero se centraliza parcialmente: el sistema central de distribución y planificación de tareas (*CTPS, Central Task Planning and Scheduling System*) ofrece tareas a aquellos robots que se encuentran ociosos, recibe propuestas de resolución por parte de los mismos, y adjudica los trabajos a las mejores ofertas. Por último, el movimiento multirrobot es distribuido, de manera que cada robot es responsable de gestionar su movimiento, buscando el camino más corto hasta su objetivo. Para ello, bajo la suposición de que el entorno es estático, se utiliza un mapa de carreteras, al que se añaden reglas que regulen el tráfico en los cruces y carriles, y técnicas de campos potenciales para evitar colisiones con obstáculos inesperados.

ARQUITECTURA DE NOREILS

La arquitectura propuesta por Noreils (Noreils y Chatila, 1992; Noreils, 1993) está pensada para robots móviles autónomos y heterogéneos. Su diseño permite que un vehículo pueda trabajar de manera autónoma, o bien que forme un equipo para llevar a cabo la tarea pedida. El concepto básico en el que se apoya es la cooperación, que para los autores ocurre cuando varios robots se unen para realizar una tarea que uno sólo no podría. La cooperación consta de una fase de colaboración, en la que se forma el equipo, y otra de coordinación, en la que los vehículos se sincronizan para lograr sus objetivos. Esta suma de colaboración y coordinación se produce en equipos robóticos pequeños, de menos de 20 individuos; a medida que esta cifra se va superando, disminuye la colaboración y se incrementa la coordinación. La arquitectura está organizada en tres niveles: funcional, de control, y de planificación. El primero gestiona las tareas básicas de procesamiento, detección de eventos y reactividad programada. El segundo nivel debe manejar las misiones o tareas que el robot es capaz de ejecutar, y por ello su labor se centra en organizar módulos, ejecutar planes,

generar informes en caso de fallo y asegurar la reactividad. Por último, el nivel de planificación se encarga de tomar decisiones de alto rango, y a su vez se encuentra dividido en dos sistemas: el sistema de coordinación y el sistema de colaboración, que asumen, respectivamente, la sincronización y la formación de equipos.

COMPORTAMIENTOS DE BASE

La arquitectura basada en comportamientos, propuesta por Mataric (Mataric, 1995; Mataric, 1998), introduce los comportamientos de base (*basis behaviors*) para gobernar sistemas multirrobot. Estos comportamientos de base se definen como un conjunto mínimo de comportamientos, con propiedades de composición adecuadas, y que presentan características de necesidad (cada uno de ellos alcanza, o contribuye a alcanzar, un objetivo relevante que no puede ser obtenido mediante otros comportamientos del conjunto, y que no puede reducirse a los mismos), suficiencia (los elementos del conjunto deben garantizar que se logran los objetivos del dominio en el que se trabaje, sin que sean necesarios otros comportamientos de base), simplicidad, localidad, estabilidad, robustez y escalabilidad. Los comportamientos de base que esta autora determina son:

- seguimiento (*following*): un agente sigue a otro.
- recogida (*homing*): buscar una localización concreta.
- vagabundeo (*self-wandering*): los robots vagan evitando colisiones entre ellos o con obstáculos.
- agregación (*aggregation*): los vehículos se agrupan manteniendo una distancia máxima entre cada elemento.
- dispersión (*dispersion*): los robots se separan manteniendo una distancia mínima entre cada uno.

A partir de ellos, pueden aprenderse comportamientos de mayor nivel, uniéndolos de dos formas: comportamientos complementarios y comportamientos contradictorios. En el primer caso, varios comportamientos se activan simultáneamente, cada uno con su propio peso, generando un nuevo comportamiento; la segunda opción se traduce en secuencias temporales de comportamientos que no pueden activarse a la par. Así, los comportamientos de base forman el sustrato para el aprendizaje, lográndose tareas complejas como recolectar, mover objetos coordinadamente o aprender por imitación.

MARTHA

El proyecto europeo MARTHA (Alami *et al*., 1997; Alami *et al*., 1998) se centra en el control y la operación de una gran flota (10-100 unidades) de robots autónomos móviles dedicados a labores de transbordo y descarga en puertos, aeropuertos, etc. La gestión de los vehículos debe ser lo más descentralizada posible, de manera que éstos evolucionen en un entorno abierto, que además no ha sido diseñado expresamente para su tarea, y que se representa mediante un modelo topológico y geométrico. MARTHA está compuesta, desde el punto de vista de la arquitectura general, por una estación central (*CS, Central Station*) y un conjunto de robots que pueden comunicarse con la CS y entre ellos. El papel de la CS consiste en planificar las operaciones de carga/descarga: qué vehículo trabaja en qué lugar, y qué ruta debería utilizar para ello; sin embargo, no interviene en la coordinación de los planes de los robots, ni planifica la trayectoria exacta a seguir. Por tanto, el principal desafío que se presenta es la cooperación multirrobot, entendiendo como tal la que se produce a nivel de plan o programa entre varios robots. Para solventar esta cuestión, dentro de MARTHA se ha desarrollado un esquema genérico, independiente del dominio de aplicación, llamado *Plan-Merging Paradigm* (PMP). Dicho paradigma permite que cada robot produzca un plan compatible con los planes que ejecutan los restantes elementos del sistema, de manera que las nuevas acciones que un vehículo debe realizar se van insertando en su plan después de comprobar que no interfieren de manera problemática con otros robots; las posibles situaciones de retraso, fallo o bloqueo se detectan y tratan de forma robusta. Aunque el proyecto MARTHA ya ha finalizado, se continúa trabajando sobre él, principalmente en mejoras y extensiones sobre PMP (Botelho y Alami, 1999; Botelho y Alami, 2000; Gravot y Alami, 2001).

ALLIANCE/L-ALLIANCE

ALLIANCE es la arquitectura desarrollada por Parker (Parker, 1998) para equipos pequeños o medianos de robots móviles heterogéneos, que desarrollen misiones compuestas por tareas débilmente acopladas, con posibilidad de dependencias temporales. Totalmente distribuida y basada en comportamientos, proporciona tolerancia a fallos, fiabilidad y cooperación adaptativa, en un entorno dinámico. La arquitectura, implantada en cada vehículo, consta de tres capas: comportamientos motivacionales (*motivational behaviours*), conjuntos de comportamientos (*behaviours sets*) y competencia activa (*active competence*). Cada comportamiento emocional se encarga de analizar información proveniente de varias fuentes (sensores, otros elementos, estado del robot) y, en función de ella, selecciona un único conjunto de comportamientos, que será el encargado de cumplir la tarea objetivo. La tercera capa se corresponde con una serie de comportamientos de supervivencia, como la

evitación de obstáculos, y que pueden excitarse o inhibirse independientemente del conjunto de comportamientos que se encuentre activo. ALLIANCE se ha extendido con L-ALLIANCE que, basándose en aprendizaje, ajusta los parámetros de entrada de los comportamientos motivacionales (Parker, 1994).

ARCO

Orientada hacia entornos industriales, la *ARquitectura para la COoperación* de plataformas móviles industriales (Moraleda et al., 1998) es una solución multirrobot distribuida, heterogénea e híbrida. Su objetivo consiste en controlar un grupo de plataformas móviles completamente autónomas con una misión común, que puede descomponerse en tareas del tipo "ir a un lugar" o "ir a un lugar deteniéndose en un punto intermedio". Una estación central se encargará de proporcionar las coordenadas del destino a los diferentes vehículos, que a partir de ese momento serán responsables de elegir la tarea óptima que pueden llevar a cabo, evitando la posibilidad de colisiones, e intentando mejorar la respuesta total del sistema. La estructura interna de cada plataforma consta de dos niveles: planificación y reacción. El primero se dedica a procesar las tareas asignadas y diseñar el correspondiente plan para realizarlas. El plan obtenido es recogido por el segundo nivel, implantado como un conjunto de comportamientos que, al fusionarse, provocan un comportamiento emergente. En el caso de que aparezcan dificultades que impidan cumplir el plan prefijado, el controlador reactivo intentará buscar algún tipo de solución por sí mismo, sin consultar con el planificador; por tanto, la información siempre fluye desde la capa superior hacia la inferior. A estos dos niveles se añaden un módulo de comunicaciones, y otro de percepción, que permitan recoger toda la información necesaria para elaborar y ejecutar los planes.

MICRORROBOTS AUTÓNOMOS

También pueden encontrarse sistemas multirrobot dentro del campo de la Microrobótica (*MARS*, Micro Autonomous Robotic Systems). Un ejemplo de arquitectura para estos sistemas es P-MARS o Programmable MARS (Mitsumoto *et al*., 1995). Como plataforma hardware se elige un tipo de vehículo de 20 $mm.^3$, equipado con un controlador, actuadores, sensores, alimentación y comunicaciones robot-robot o robot-host. Para el software, y de nuevo con la Biología como fuente de ideas, se emplea un algoritmo que simula el comportamiento del sistema inmunológico, de manera que la estrategia del colectivo se adapta para llevar a cabo una tarea. Otra arquitectura de microrrobots es la propuesta por (Dudenhoffer y Bruemmer, 2001) en el Idaho National Engineering and Environmental Laboratory, y que se encarga del hardware y del software. Se utilizan tres tipos de vehículos: los soldados, los sargentos, y el robot padre; los soldados son los más pequeños, controlados por los sargentos, que a su vez son monitorizados por el padre, el

robot de mayor tamaño. El software sigue el paradigma de los campos potenciales sociales. La atracción, establecida a partir de sonidos emitidos por los robots, impide que los elementos del sistema se alejen demasiado, y los atrae si es necesario; la repulsión, dependiente del sonido y de los sensores de evitación, asegura que los vehículos no se aproximan excesivamente entre ellos. De esta manera se consigue un sistema con capacidades de auto-organización, adaptación, inteligencia emergente, o autonomía flexible.

FÚTBOL ROBÓTICO

Como se ha comentado anteriormente, el fútbol robótico o *soccer* es un dominio muy interesante como banco de pruebas para sistemas multiagente, ya que son muchas las cuestiones hardware y software que puede tratar: diseño de agentes autónomos, colaboración multi-agente, razonamiento en tiempo real, fusión de información sensorial... Por ello ha dado lugar a numerosas arquitecturas, que son revisadas con mayor detalle en (Stone y Veloso, 2000). Debido a este auge, en los últimos años se han organizado varias competiciones internacionales. Algunas de ellas son las promovidas por la *Federation of International Robot-soccer Association* (FIRA, 2002), que abarca varias categorías: robots humanoides (HuroSot), robots entrenados por humanos (MiroSot, KepheraSot, NaroSot, RoboSot), y simulación (SimuroSot). Otros campeonatos famosos se engloblan bajo la iniciativa RoboCup (RoboCup, 2002), que trabaja en tres dominios, tanto en simulación como en equipos reales: RoboCup Soccer, cuyo objetivo es desarrollar, para el año 2050, un equipo de robots humanoides completamente autónomos que pueda ganar al equipo humano campeón del mundo de fútbol; RoboCup Rescue, orientado a la búsqueda y el rescate en desastres de grandes dimensiones; y RoboCup Junior, con fines educativos.

2.4 Estrategias de Navegación Multirrobot

Si el apartado anterior se ocupó del estudio de algunas arquitecturas multirrobot muy conocidas, esta sección ahonda en el nivel de detalle y se centra en la principal tarea que deben ejecutar los componentes de un sistema multirrobot de vehículos autónomos: la navegación. Son numerosos los trabajos realizados en este campo, dentro de las diferentes áreas que contempla (planificación de caminos multirrobot, generación y mantenimiento de formaciones, control de tráfico...), pero, paradójicamente, la mayor parte de los mismos se quedan en la simulación, y son pocos los sistemas físicos implantados en la actualidad (Parker, 2000). Puesto que las ideas primordiales de la navegación en el caso de un único vehículo, explicadas en el capítulo de introducción, son similares a las de la navegación multirrobot, en este punto se mencionarán simplemente como recordatorio.

Navegar consiste básicamente en moverse a través de un entorno evitando posibles colisiones con elementos del mismo, tanto estáticos o dinámicos (considerando como tales los restantes elementos del sistema multirrobot). En el caso de un único robot, las aproximaciones clásicas para resolver la cuestión son cuatro: navegación deliberativa, navegación reactiva, navegación híbrida y navegación basada en comportamientos. Mataric (Mataric, 2001) propone un resumen claro y sencillo de las cuatro metodologías:

- navegación deliberativa: piensa detenidamente, actúa después.
- navegación reactiva: no pienses, reacciona.
- navegación híbrida: piensa y actúa independientemente, en paralelo.
- navegación basada en comportamientos: piensa en cómo vas a actuar.

En lo que sigue, y puesto que una de las principales aportaciones de esta tesis consiste en un algoritmo de planificación de trayectorias, se ofrece una clasificación de métodos de navegación multirrobot global basada precisamente en cómo se planea el movimiento de los vehículos. Por tanto, no se incluyen en ella estrategias de navegación multirrobot reactivas ni basadas en comportamientos, aunque sí se mencionan aquellas estrategias híbridas en las que se planifique el movimiento de los robots.

2.4.1. Navegación Multirrobot Global Planificada

Para clasificar los elementos de un dominio, es necesario primero determinar un conjunto de criterios que permitan la diferenciación entre individuos, y que serán elegidos dependiendo del enfoque que se desee dar a la categorización. Por ello, antes de presentar la ordenación de metodologías de navegación multirrobot global planificada, se explican los ejes de clasificación utilizados, resumidos en la Tabla 2.2 para facilitar su comprensión.

En primer lugar, y puesto que una de las aportaciones de la tesis consiste en un algoritmo para la planificación de la trayectoria de un robot -es decir, el cálculo del camino a seguir junto con la velocidad con la que recorrerlo-, los métodos de navegación estudiados se discriminan en función precisamente de cómo abordan la generación del perfil de velocidad. En el presente trabajo, se distingue entre dos tipos de métodos: los que se centran en los aspectos geométricos de la planificación y no consideran como aspecto básico la generación de un perfil de velocidad (normalmente, suponen que el vehículo se mueve a velocidad constante), y los que sí proporcionan una trayectoria completa, analizando diferentes restricciones a las que se ve sometido el movimiento del vehículo.

A un segundo nivel, referido a cómo se realiza la planificación teniendo en cuenta que son varios los robots involucrados, puede distinguirse entre planificación centralizada y planificación desacoplada, según establece Latombe (Latombe, 1991). En la planificación centralizada se calculan simultáneamente los caminos de todos los vehículos, considerando que forman parte de un único robot compuesto. Frente a esta opción, la planificación desacoplada obtiene un camino para cada uno de los robots, y luego analiza las interacciones entre todos. La planificación centralizada se caracteriza por ser completa, aunque computacionalmente muy costosa; la planificación desacoplada, en cambio, presenta menor complejidad, pero puede perder soluciones aunque la metodología de planificación del movimiento para cada vehículo sí sea completa. Es por ello que tradicionalmente se han preferido estas últimas, ya que se ha supuesto que la falta de completitud no es una desventaja significativa. Sin embargo, se han obtenido algunos resultados que contradicen esta hipótesis (Sánchez y Latombe, 2002), a pesar de que la planificación desacoplada sigue siendo útil en situaciones en las que las interacciones entre robots no sean muy limitantes o en aplicaciones distribuidas que carecen de conocimiento del estado global del sistema.

Esta separación entre planificación centralizada y descentralizada propuesta por Latombe no es única y admite variaciones. Algunos autores (Fujimura, 1991; Arai y Ota, 1992) interpretan la planificación centralizada como un único planificador que se encarga del movimiento de todos los vehículos, mientras que la planificación descentralizada se da si cada robot computa su propio camino. Para otros (LaValle y Hutchinson, 1998), los enfoques centralizados y descentralizados son los extremos de un espectro continuo, en el que son válidas soluciones intermedias; otra metodología que recoge aspectos de ambas tendencias es la de Brock y Khatib (Brock y Khatib, 1999), para entornos industriales, donde se coordina una celda multirrobot formada por varios manipuladores partiendo de técnicas descentralizadas, pero empleando la centralización a la hora de resolver conflictos.

A su vez, el criterio de planificación desacoplada puede dividirse en tres subclases posibles de técnicas: priorizadas, coordinación de caminos y presencia de obstáculos móviles.

Las metodologías priorizadas (Erdmann y Lozano-Pérez, 1986; Erdmann y Lozano-Pérez, 1987) consisten en planificar las trayectorias de una en una, tomando aquellos robots cuyo perfil de velocidad ya se ha calculado como obstáculos móviles para los restantes; es decir, el movimiento de cada robot se planifica de manera que evite los obstáculos estáticos del entorno y los obstáculos móviles correspondientes a los vehículos precedentes en la ordenación de prioridades. El orden de elección de los vehículos se

establece mediante una ordenación basada en prioridades dada por algún elemento externo al sistema; esta priorización puede establecerse naturalmente en función de las circunstancias en las que se desenvuelva el sistema: sistemas maestro-esclavo, operaciones de ensamblaje... incluso puede no ser constante, o irrelevante en el caso de robots con tareas que no interfieran. Esta solución está planteada dentro del llamado *espacio de configuraciones espacio-tiempo*, obtenido añadiendo la dimensión temporal al espacio de configuraciones tradicional desarrollado por Lozano-Pérez en (Lozano-Pérez, 1983), y en el que una planificación se obtiene mediante algoritmos de búsqueda sobre dicho espacio. La inclusión del tiempo en el espacio de configuraciones original obliga a tener en cuenta una restricción principal a la hora de realizar las búsquedas: el tiempo sólo avanza, nunca retrocede. Esto significa que los robots no pueden volver atrás en el tiempo ni interactuar consigo mismos; además, de esta limitación se desprende que desde cualquier punto, la región espacio-tiempo alcanzable queda acotada por la velocidad máxima que pueda desarrollar el vehículo. Como se destacó con anterioridad, no es una solución completa: pueden calcularse trayectorias iniciales que impidan encontrar planificaciones válidas para robots posteriores. Los autores han centrado su estudio en dos áreas: la traslación de objetos planares, y sistemas de dos brazos articulados con dos articulaciones planares.

La coordinación de caminos (O'Donnell y Lozano-Pérez, 1989) utiliza métodos de planificación o *scheduling*, en los que el espacio es el recurso compartido; aparte de la evitación de colisiones, debe garantizarse la ausencia de bloqueos, es decir, de situaciones en las que ambos robots esperan una acción del otro para poder continuar. Mediante los denominados *task-completion diagrams* (diagramas de completitud de tareas, o diagramas de coordinación), se establecen simultáneamente las trayectorias de los dos robots garantizando que son seguras y sin bloqueos. Inicialmente se ideó para sistemas formados por dos robots manipuladores en entornos industriales, pero su uso se ha extendido a casos más generales como los sistemas multirrobot. Esta técnica altera los caminos únicamente en velocidad, no geométricamente, aunque proporciona ciertas técnicas para ciertas situaciones en las que resulta más conveniente una modificación en el espacio.

El último eje de clasificación comprende técnicas de planificación del movimiento de un único robot en entornos dinámicos, -es decir, con obstáculos móviles-, ya que, en definitiva, este tipo de sistemas son sistemas multirrobot si se consideran los obstáculos móviles como parte de sus elementos.

<table>
<tr><th>Nivel</th><th>Eje</th><th colspan="2">Valor</th></tr>
<tr><td rowspan="2">I</td><td rowspan="2">Definición del perfil de velocidad</td><td colspan="2">Sin perfil de velocidad</td></tr>
<tr><td colspan="2">Trayectoria</td></tr>
<tr><td rowspan="4">II</td><td rowspan="4">Tipo de planificación</td><td colspan="2">Centralizada</td></tr>
<tr><td rowspan="3">Desacoplada</td><td>Priorizada</td></tr>
<tr><td>Coordinación de caminos</td></tr>
<tr><td>Presencia de obstáculos móviles</td></tr>
</table>

Tabla 2.2: Clasificación de métodos de navegación global multirrobot planificada

En la literatura pueden encontrarse revisiones y clasificaciones relativas a la planificación de movimientos, planteadas desde otros puntos de vista (Latombe, 1991; Fujimura, 1991; Arai y Ota, 1992; Hwang y Ahuja, 1992); la taxonomía aquí presentada se circunscribe exclusivamente, como ya se ha mencionado, a la navegación planificada global y multirrobot.

2.4.1.1. Navegación sin perfil de velocidad

Enfoque centralizado

Bajo este punto de vista, Svetska y Overmars (Svetska y Overmars, 1996) se basan en la construcción de un roadmap[5] para cada uno de los vehículos, sobre el que se generará un roadmap compuesto para todos los robots. A partir de esta estructura, dos nuevos grafos jerárquicos son creados. Primero, el supergrafo plano (*flat*), cuyos nodos son nodos del roadmap compuesto con una distribución concreta de los vehículos, y cuyas aristas representan cómo pasar de un nodo a otro moviendo únicamente un robot, de forma que situaciones conflictivas como bloqueos, etc. se eviten. Y en segundo lugar, se obtienen supergrafos multinivel, que agrupan nodos, ahorrando así memoria. El algoritmo trabaja bien para sistemas con un número de elementos menor o igual a cinco; para sistemas multirrobot mayores, se recomienda la integración del método con técnicas desacopladas.

5. Un roadmap es un grafo o red de curvas unidimensionales, que modelan la conectividad del espacio libre de obstáculos en el que se desenvuelve el robot. Existen diferentes metodologías para su construcción, como los grafos de visibilidad o los diagramas de Voronoi.

En (Parsons y Canny, 1990) puede encontrarse una solución a un caso particular del problema de la mudanza generalizado en dos dimensiones (*generalized movers' problem*); esta propuesta, centralizada y global, se divide en dos etapas. En primer lugar, se realiza una descomposición en celdas, considerando las restricciones debidas a obstáculos estáticos y las causadas por los demás robots. A partir de ella, y en segundo lugar, se construye un grafo de adyacencia sobre el que se realiza una búsqueda, que puede estar guiada por un heurístico, pudiendo adaptarse mejor al caso que se quiere resolver.

Otro planificador de movimientos centralizado que no considera las velocidades de los robots móviles es el expuesto por Hwang (Hwang, 95). Su utilidad se centra en grupos pequeños de robots, tanto móviles como dedicados a tareas de ensamblado. Esta técnica aplica una estrategia de búsqueda eficiente llamada SANDROS, que realiza una búsqueda heurística inicialmente, y exhaustiva posteriormente, en la que la resolución es fijada por el usuario. SANDROS sigue una estructura jerárquica basada en un planificador global y otro local. El primero mantiene una red de subobjetivos (inicialmente con celdas muy grandes), y produce secuencias de subobjetivos; el segundo verifica tales secuencias hasta que encuentra una solución definitiva.

Enfoque desacoplado

- *Planificación Priorizada*

  Algunos autores, como (Liu *et al.*, 1989) proponen un algoritmo para calcular el movimiento coordinado de dos vehículos móviles modelados como círculos, en un entorno con obstáculos estáticos representado mediante un *quadtree*[6]. Establecida la priorización en función del tamaño del vehículo, los conflictos se resuelven utilizando Redes de Petri temporizadas. La principal limitación que presenta este método radica en el hecho de que su extensión a más de dos vehículos móviles resulta bastante complicada.

  Por otra parte, Tournassoud (Tournassoud, 86) separa, para cada robot móvil, el espacio real en "half-spaces", que representan restricciones de movimiento con otros elementos del entorno. Sus inconvenientes se deben a que los vehículos deben seguir trayectorias rectilíneas, y a que necesita un orden de prioridades muy estricto.

  La técnica de campos potenciales también ha sido aplicada al problema de la generación de trayectorias multirrobot (Warren, 1990), de manera que se

6. Un quadtree es un árbol en el que cada nodo se expande *d* dimensiones, de forma que genera 2^d hijos. Su nombre se debe a la aplicación inicial de esta herramienta -el análisis de datos bidimensionales (imágenes)-, en las que cada nodo produce cuatro hijos, de ahí la expresión *quad*tree.

utiliza como técnica de evitación de obstáculos sobre caminos previamente planificados; esta solución planteada añade el tiempo como tercera dimensión al problema tradicional de campos potenciales. A pesar de la eficiencia de esta metodología, y de que tiene en cuenta limitaciones cinemáticas de la velocidad, presenta el inconveniente de no considerar restricciones de tipo dinámico u operacional.

Otros autores (Bennewitz *et al.*, 2001; Bennewitz y Burgard, 2001) se centran en estudiar esquemas de prioridad para métodos de planificación de caminos desacoplados en el caso multirrobot. Para ello, primero se generan todos los caminos, y luego se comprueba si existe algún conflicto; en caso de que así ocurra, se busca un orden de prioridades, óptimo, generado aleatoriamente del que se constata su validez. Además, es posible tener en cuenta, a priori, ciertas restricciones de algunos de los vehículos.

Buckley (Buckley, 1989) también investiga en el establecimiento de un criterio de prioridades que minimice el número de evitaciones de obstáculo necesarias en un sistema multirrobot. Así, partiendo de unos caminos ya fijados, se realiza un análisis de las colisiones en el espacio, incluyendo las posiciones de inicio y fin. A partir de ahí se genera un grafo de prioridades, que determina qué vehículos pueden moverse libremente, cuáles tendrán que modificar su velocidad, y cuáles su camino. Sus carencias se refieren a la no consideración de continuidad en velocidad y aceleración, y la imposibilidad de definir otros criterios (tamaño, velocidad, necesidades de operación...)

- *Coordinación de caminos*

  Dentro de esta modalidad de planificación, (Siméon *et. al*, 2002) presentan un método geométrico que emplea diagramas de coordinación (representación de los lugares en los caminos de los robots en los que suceden colisiones) n-dimensionales para poder determinar, eficientemente, si existe un movimiento coordinado para que los vehículos circulen por sus caminos en ausencia de choques. Se obliga a que los caminos de los robots estén compuestos por rectas y curvas; además, en la solución obtenida los vehículos se mueven de uno en uno, aunque esta última restricción puede solventarse con un postprocesamiento.

  Guo y Parker (Guo y Parker, 2002) modifican parcialmente la idea básica de la coordinación de caminos. Su algoritmo se basa en la descomposición camino-velocidad de Kant y Zucker (Kant y Zucker, 1986), desarrollada con mayor profundidad en el epígrafe siguiente. Cada vehículo calcula su propio camino geométrico, y son las velocidades las que se determinan mediante

diagramas de coordinación. Estos perfiles de velocidad se obtienen de manera distribuida, ya que cada vehículo implanta las funciones de coste de la búsqueda D* que se efectúa sobre el diagrama de coordinación; además, pueden insertarse retrasos y prioridades que impongan esperas a ciertos robots. La solución es aplicable a entornos tanto interiores como exteriores, y permite la replanificación en tiempo real; sin embargo, supone que los vehículos se desplazan a velocidad fija y constante, pudiendo además cambiar de manera instántanea de velocidad, lo que dificulta su aplicación física.

- *Presencia de obstáculos móviles*

  En este apartado, y debido a la enorme influencia que ha tenido sobre otros muchos investigadores, destaca la idea de la *descomposición camino-velocidad* (Kant y Zucker, 1986) que, bajo ciertas restricciones, separa el problema de la planificación de trayectorias (*trajectory planning problem*, TPP) en la planificación de un camino a través de obstáculos estáticos (*path planning problem*, PPP), y en la planificación de un camino a través de los obstáculos móviles (*velocity planning problem*, VPP). Es decir, TPP = PPP + VPP; se busca primero el camino o caminos posibles del PPP, se elige uno, y se calcula la velocidad a lo largo del mismo. Así, las cuatro dimensiones (x,y,z,t) del TPP, se reducen en el PPP al caso tridimensional (x,y,z), y a dos dimensiones (s,t) para el VPP. A pesar de que simplifica enormemente el problema de partida, este algoritmo tiene algunas limitaciones: las velocidades son constantes y definidas a tramos, y los resultados son mejores en caso de pocos obstáculos que además se muevan ortogonalmente al robot. Para aliviar en parte estos problemas, ambos autores ampliaron esta propuesta inicial con una arquitectura híbrida de dos niveles (Kant y Zucker, 1988). El primero se encargaría de la planificación global mediante la descomposición camino-velocidad, mientras que el nivel bajo se encargaría de la planificación local mediante técnicas de campos potenciales. Sin embargo, esta nueva orientación no elimina la principal desventaja de la descomposición camino-velocidad: puede perder soluciones debido a bloqueos, producidos en el caso de que uno de los vehículos finalice su recorrido sobre el camino de otro de los robots.

  Las Figuras 2.1 y 2.2 ilustran más descriptivamente dos casos típicos de dicha problemática. La primera de ellas muestra el caso de un bloqueo que no puede solventarse mediante modificaciones en la velocidad de los robots; la única manera de evitar el bloqueo pasa por modificar, geométricamente, el

camino de alguno de los dos vehículos.

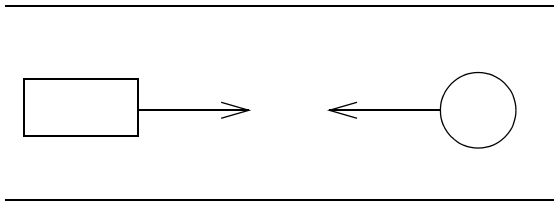


Figura 2.1. Situación de bloqueo: caso irresoluble

La Figura 2.2 refleja unas circunstancias distintas, en las que la planificación del perfil temporal sí puede impedir la obstrucción del camino del robot rectangular por el vehículo circular. En efecto, si el robot rectangular se desplaza más rápidamente que el circular, de manera que sobrepase el área de bloqueo antes, ambos vehículos podrán alcanzar sus destinos de forma segura.

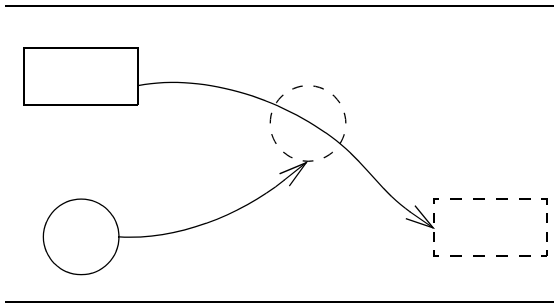


Figura 2.2. Situación de bloqueo: caso resoluble

Hwang y Ju (Hwang y Ju, 2002) plantean un método para la generación de perfiles de velocidad continuos similar a la descomposición camino-velocidad, pero fraccionando el camino completo en porciones que se analizan individualmente. La aportación de esta técnica radica en que, para evitar la discontinuidad del perfil de velocidad obtenido, emplean un modelo de propagación de ondas. A pesar de ello, adolece de ciertas limitaciones: se suponen velocidades constantes, y no se asegura la continuidad en aceleración.

También dentro del área de planificación en presencia de obstáculos móviles, sobresale el trabajo desarrollado por Fujimura, cuyo objetivo consiste en obtener el camino de mínimo tiempo para un robot móvil que se desenvuelve en un entorno poblado por obstáculos móviles. En (Fujimura y Samet, 1989) se presenta una metodología en la que los obstáculos se modelan como polígonos convexos que se mueven en una dirección fija con una velocidad constante, mientras que el robot está limitado por una velocidad tope

superior a la de los obstáculos. Éstos no se tocan entre ellos, ni tampoco al punto destino, que además se supone que también se mueve. Posteriormente, en (Fujimura, 1992) se considera un nuevo tipo de obstáculo dinámico: los obstáculos efímeros (*transient obstacles*), que dependen del tiempo, de manera que pueden aparecer y desaparecer en el entorno; resultan útiles, por ejemplo, en ambientes industriales en los que se deban recoger con piezas de una cinta transportadora. Este algoritmo necesita un conocimiento a priori de la situación, y considera al robot como un punto que se mueve con una cierta velocidad constante máxima. En el caso de que no existan obstáculos efímeros, el método se comporta como un grafo de visibilidad; y además, puede ampliarse para tener en cuenta la presencia de obstáculos móviles tanto lentos como rápidos. Por último, (Fujimura, 1995) presenta una solución basada en roadmaps, de manera que el vehículo sólo puede moverse dentro de ella, bajo un límite superior de velocidad. Esta aproximación se apoya a su vez en dos técnicas: la planificación geométrica de caminos entre obstáculos dinámicos (al estilo de Kant y Zucker, aunque con ciertas variantes), y la búsqueda de caminos en redes dependientes del tiempo. Ambas se combinan de forma que la búsqueda en el mapa de carreteras usa la planificación en el plano espacio-tiempo para analizar los nodos. En principio, no considera restricciones de aceleración, aunque un procesamiento posterior haría viable su utilización.

Apoyado en condiciones como la velocidad constante -inferior a una velocidad tope- de los robots, y la generación de caminos que no observen restricciones cinemáticas, el método de Tsubouchi y Arimoto (Tsubouchi y Arimoto, 1994) desarrolla una búsqueda heurística en el espacio 3D: los obstáculos se modelan como cilindros o cilindros oblicuos, y la solución final pasa por hallar un camino entre ellos que evite las colisiones; en resumen, la idea básica consiste en localizar un camino que sortee dichos cilindros. No existe un conocimiento a priori del movimiento de los obstáculos, por lo que se utiliza una predicción del mismo; asimismo, el algoritmo se repite sucesivamente, para soslayar la influencia de la variación de comportamiento en los vehículos.

2.4.1.2. Navegación con trayectorias

Enfoque centralizado

Bajo este punto de vista destaca el trabajo de Sánchez y Latombe (Sánchez y Latombe, 02), dirigido hacia entornos industriales con un máximo de seis brazos manipuladores, y basados en *roadmaps probabilísticos* (*probabilistic roadmaps*, PRM) generados sobre el espacio de configuraciones compuesto de los robots. También se basa en este tipo de técnicas el método propuesto por Hsu (Hsu et al., 2000), y que para cada solicitud de planificación de una trayectoria genera un PRM desde el punto inicial hasta el objetivo, tomando como nodos o *milestones* puntos del plano estado-tiempo; esta metodología presenta una variante adaptada al caso desacoplado, como se verá posteriormente al comentar el eje "Presencia de Obstáculos Móviles"

Enfoque desacoplado

- *Planificación Priorizada*
  La planificación priorizada admite distintas perspectivas a la hora de su implantación. Algunas soluciones se apoyan en el concepto TPP=PPP+VPP de Kant y Zucker ya estudiado, mientras que otras operan directamente sobre el espacio de estados del vehículo, extendido para albergar la dimensión temporal. A continuación se desglosarán las metodologías más relevantes dentro de cada uno de los planteamientos anteriores.
  Son varias las técnicas relativas a la descomposición camino-velocidad que pueden encontrarse en la literatura. Entre ellas se distingue el trabajo desarrollado por Fraichard y Laugier (Fraichard y Laugier, 1989; Fraichard y Laugier, 1991), que plantean una arquitectura híbrida priorizada para la planificación de movimientos de varios vehículos no holonómicos en un universo dinámico. El nivel superior de la arquitectura se encarga de la planificación de una trayectoria global para cada vehículo, dentro de un horizonte de tiempo preestablecido; el nivel inferior se destina a la resolución de conflictos locales que puedan aparecer durante la ejecución de la trayectoria prevista. Es el planificador global el que aplica la división TPP=PPP+VPP, de manera que el camino se construye teniendo en cuenta las restricciones cinemáticas del vehículo, tras lo cual se le añade el perfil temporal que eluda a los robots colindantes. Sin embargo, algunas exigencias limitan un tanto la versatilidad de esta solución: necesita de un entorno fuertemente estructurado con un código de circulación (es decir, el espacio

de trabajo es similar a red de carreteras o autopistas), el orden de prioridades no es completo, no se tienen en cuenta restricciones de velocidad, y además los perfiles de velocidad generados no son continuos. Para eliminar el problema de los bloqueos, inherente a la descomposición camino-velocidad, (Fraichard, 1998) propone una solución basada en *caminos adyacentes*, paralelos al camino original generado en la fase PPP, a los que el vehículo puede pasar en caso de que la VPP lo requiera para evitar colisiones con otros robots.

Otra técnica fundada en la descomposición camino-velocidad es la desarrollada por Ferrari y otros (Ferrari *et al*., 1998), cuya principal aportación reside en el uso de medidas de calidad de la solución final. Para ello se designan dos tipos de magnitudes: los *factores de impacto de colisión*, que evalúan la calidad del camino de un vehículo, y los *índices de desempeño*, que estiman la calidad total de un plan para el sistema multirrobot. Para cada robot, se aplica un algoritmo que calcula una familia de caminos geométricos a partir de uno inicial (etapa PPP), y se calculan sus índices correspondientes; el proceso se repite mientras existan mejoras en los tiempos de elaboración del plan y de ejecución de los caminos. Cuando dichos tiempos comiencen a empeorar, se pasa a la fase VPP, en la que las colisiones entre robots se resuelven deteniendo algunos de los vehículos, o bien modificando localmente sus caminos. A pesar de la ventaja que supone la generación de parámetros que determinen la bondad de una solución concreta, el método no indica cómo se calcula el perfil de velocidad, y además elimina los choques deteniendo los vehículos, lo que no es siempre necesario.

- *Coordinación de caminos*

  Similar al método propuesto por (Siméon *et al*., 2002) en el caso de la planificación sin trayectorias, Akella y Hutchinson (Akella y Hutchinson, 2002) ofrecen una aproximación que sí construye trayectorias como la unión de un camino y un perfil de velocidad. Esta solución es válida para robots móviles y también en entornos puramente industriales, como celdas de soldadura y pintura en industrias automovilísticas. El problema se resuelve mediante técnicas de optimización, donde el tiempo es la magnitud a minimizar: se formula como programación lineal mixta entera, identificando condiciones necesarias y suficientes para la coordinación de robots sin colisiones.

  Diseñado para ambientes industriales con dos manipuladores, (Lee *et al*.,

1995) presentan una técnica de coordinación de caminos bautizada como *descomposición planificación-coordinación*. Ésta se centra en la planificación de velocidades, por lo que no se concede excesiva importancia a la definición de caminos geométricos. Los perfiles de velocidad usados son trapezoidales, aunque son posibles otros siempre que cumplan las restricciones cinemáticas y dinámicas. La base del método es añadir retrasos en los puntos intermedios entre segmentos del camino. Para ello se crea el *Collected Coordination Space*, que incluye las regiones de colisión; sobre él se determina la curva de coordinación que garantizará la colisión sin que se modifique el camino geométrico, y sin que se alteren los perfiles de velocidad calculados inicialmente, ya que sólo se añaden tiempos de espera.
Por último, Rude y otros (Rude *et al.*, 1995) exponen un método que presta especial atención a la incertidumbre espacial en el movimiento de los vehículos, y que influye en la percepción que cada robot tiene del resto de integrantes del sistema multirrobot. Así, cada robot se modela, en el espacio-tiempo global, como un vector de incertidumbre determinista, que se traduce en un hiper-elipsoide cuatridimensional; por otra parte, las trayectorias se representan como líneas, denominadas *world lines* o líneas del mundo. El problema de la planificación de trayectorias se resuelve si se consigue una configuración apta de las líneas del mundo que eviten la colisión, considerando que alrededor de cada línea debe establecerse una zona de exclusión correspondiente a la incertidumbre asociada. En caso de que un par de líneas interfieran, se genera un vector de desplazamiento temporal, que se añadirá a la trayectoria del robot que ha detectado la colisión, y que provocará una alteración tanto de su perfil geométrico como de su perfil temporal.

- *Presencia de obstáculos móviles*
  Utilizando únicamente información de la velocidad, Fiorini y Shiller (Fiorini y Shiller, 1998) presentan su idea para la generación de trayectorias de un vehículo. Para ello, definen el concepto *Velocity Obstacle*, que mapea el entorno dinámico del robot en su espacio de velocidades, con el que acotan unas zonas llamadas *Velocidades de Evitación Alcanzables* que contienen aquellos valores de velocidad que permiten eludir obstáculos estáticos o dinámicos del vehículo. De esta forma, establecen tres tipos de maniobras posibles: adelantamiento, retraso o divergencia. La planificación de una trayectoria desde un punto de inicio hasta un punto de fin consiste en el encadenamiento de maniobras, que se logra mediante mecanismos de

búsqueda. Se implantan dos mecanismos de búsqueda posibles: búsqueda global, para planificaciones fuera de línea, y búsqueda con heurísticos, para aplicaciones on-line. Las ventajas de este método son múltiples: considera la dinámica del vehículo, utiliza un modelado común para obstáculos estáticos y dinámicos, y como se ha mencionado es válido para aplicaciones en línea y off-line. Sin embargo, sus inconvenientes radican en que no admite restricciones cinemáticas y operacionales, las soluciones dependen de un horizonte de tiempo, el heurístico no garantiza llegar al final, y además modifica los perfiles geométricos.

Los trabajos de Fraichard, Laugier y Scheuer (Fraichard y Laugier, 1992; Fraichard y Scheuer, 1994) aportan una solución basada en añadir la dimensión temporal al espacio de estados del vehículo móvil, de forma que trabajan en el denominado *espacio estado-tiempo*. Este nuevo espacio incluye toda la información relativa a restricciones cinemáticas y dinámicas que afectan a la planificación de velocidades del robot. El paso de una configuración a otra del espacio estado-tiempo se realiza seleccionando un valor de aceleración dentro de un conjunto discretizado de valores posibles, por lo que los perfiles de aceleración obtenidos son discontinuos. Frente a este inconveniente, la metodología produce trayectorias óptimas respecto al tiempo, ya que se obtienen a partir una búsqueda en el grafo de configuraciones parametrizada para minimizar dicho criterio.

También dentro de este eje pueden incluirse los Problemas de Evitación del Asteroide o *Asteroid Avoidance Problems* (Reif y Sharir, 1985), en los que se analiza el comportamiento de un objeto B, modelado como un poliedro convexo, que puede moverse libremente mediante traslaciones con un límite de velocidad, todo ello dentro de un entorno en el que los obstáculos también son poliedros convexos, pero con trayectorias conocidas de velocidad constante; se considera que ni los obstáculos ni el objeto B pueden efectuar rotaciones. Los algoritmos inicialmente planteados por Reif y Sharir, aunque permiten cambios en la velocidad de B, no tienen en cuenta la influencia de posibles restricciones de velocidad. En el caso de que se desee valorar el efecto de dichas limitaciones, surge una variante de la evitación del asteroide: los Problemas de Evitación Cinemático-Dinámica del Asteroide o *Kinodynamic Asteroid Avoidance Problems*. Kindel y otros (Kindel et al., 2000) proponen una solución basada en la construcción de un roadmap probabilístico sobre el espacio estado-tiempo; un estado se define como el par (configuración, velocidad). El algoritmo se fundamenta en un muestreo de tal espacio, escogiendo entradas de control aleatoriamente, de forma que

el PRM se construye como un grafo orientado a lo largo del eje temporal. El inconveniente de este método reside en el hecho de que estima el desplazamiento de los obstáculos como trayectorias rectilíneas de velocidad constante.

Una vez que se ha elaborado la clasificación de técnicas de navegación planificadas para sistemas multirrobot, quedaría como último paso situar la que se presenta en este trabajo. Así, se plantea una metodología deliberativa para la planificación de trayectorias en sistemas multirrobot compuestos por vehículos móviles terrestres heterogéneos que se mueven en dos dimensiones, tanto en entornos exteriores como interiores. En este esquema, las restricciones de velocidad -ya sean físicas u operacionales- que actúan sobre el vehículo se tienen en cuenta a la hora de generar el perfil de velocidad e la trayectoria. Para ello, se emplea un enfoque desacoplado con prioridades, que se apoya en la descomposición camino-velocidad de Kant y Zucker, realizando sobre el plano $s\ x\ t$ el ajuste pertinente de las velocidades para lograr una navegación segura y sin colisiones.

Como resumen de todo lo expuesto, la Figura 2.3 muestra un esquema de la clasificación realizada.

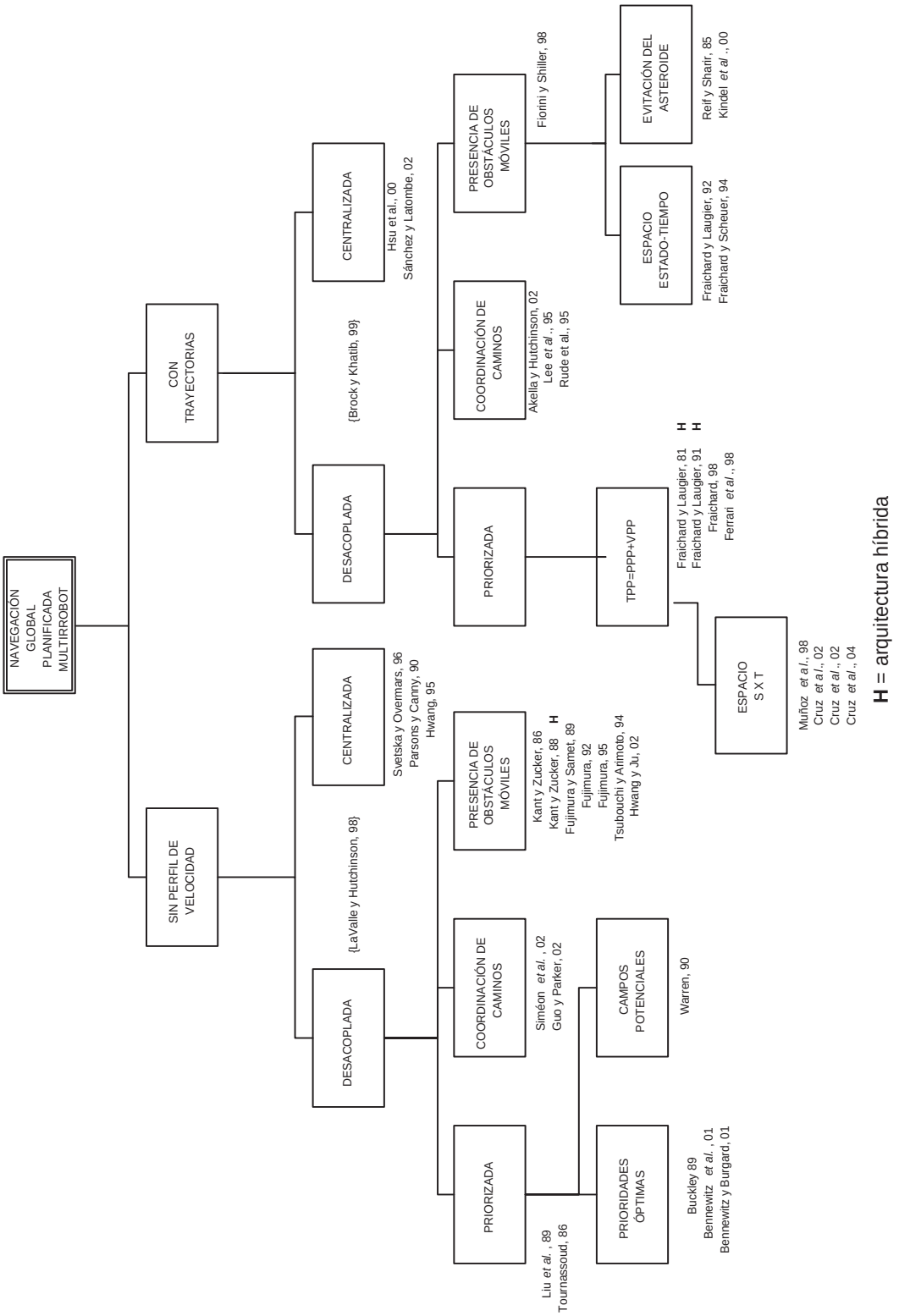


Figura 2.3. Clasificación de metodologías de navegación global multirrobot planificada.

2.5 Conclusiones

Hoy en día son numerosas las arquitecturas multirrobot implantadas, y muy distintos los enfoques bajo los que cada una es diseñada y desarrollada, como se ha expuesto a lo largo de este capítulo. Sin embargo, a pesar de que es un campo en el que continuamente se avanza, aún no existen criterios globales que permitan definir arquitecturas multirrobot de manera que se ajusten a ciertos requerimientos; ni siquiera hay conformidad a la hora de clasificarlas o compararlas. Por ello, con el objetivo de ofrecer una visión amplia y clara del estado del arte relativo a los sistemas multirrobot, se han recogido en este capítulo las principales estructuras de clasificación para entornos multirrobot, así como las arquitecturas más destacadas y conocidas para este tipo de sistemas.

Asimismo, se han analizado multitud de técnicas de navegación para sistemas multirrobot que además, como aportación de la tesis, se han organizado mediante dos ejes taxonómicos: existencia de un perfil de velocidad, y tipo de planificación. Esta misma clasificación se utiliza para localizar, entre todos los métodos estudiados, la navegación multirrobot que aquí se propone, y que será objeto de un estudio mucho más completo en los próximos capítulos.

CAPÍTULO 3. Planificación de Trayectorias para Robots Móviles

3.1. Introducción

El capítulo previo se ha dedicado al estudio de los sistemas multirrobot compuestos por vehículos móviles, analizando diferentes taxonomías, presentando arquitecturas de este tipo consideradas relevantes y, por último, clasificando aquellos sistemas que implantan la navegación de forma planificada. Así, una vez centrado el problema que se desea abordar, puede pasarse a desglosar más detalladamente el mecanismo de planificación de trayectorias para robots móviles que se plantea en esta tesis.

A ello se destina esta capítulo de la presente memoria, que se organiza alrededor de tres apartados principales. El primero comienza con una descripción somera de la cuestión a resolver y de la solución propuesta, incluyendo también una formalización matemática del problema de la planificación de trayectorias, que sirva como base teórica para facilitar la comprensión del resto del capítulo. Los dos siguientes apartados se destinan a profundizar en la metodología. Así, en la tercera sección del capítulo se examina la técnica de generación de los caminos geométricos que seguirá el robot, implantada de manera que garantiza buenas propiedades del camino para su posterior seguimiento. En el cuarto apartado se desglosa la construcción del perfil de velocidad con el se recorrerá el camino geométrico obtenido para el robot, incluyendo también una pormenorización de las distintas limitaciones de velocidad a las que puede verse sometido el vehículo. Por último, el capítulo se cierra con las principales conclusiones que pueden extraerse de todo lo descrito a lo largo del mismo.

3.2. El problema de la planificación de trayectorias para robots móviles

En el capítulo de introducción ya se hizo mención al hecho de que para planificar el curso global de un robot móvil no sólo debe tenerse en cuenta el camino que debe recorrer, sino también la velocidad a la que lo hará, ya que existen diferentes limitaciones de velocidad que afectan significativamente a su movimiento. En el ámbito de la Robótica, la unión de las referencias espaciales y temporales que debe seguir un vehículo se denomina

trayectoria, si bien es cierto que no hay un consenso total a la hora de utilizar este término, y muchas veces se emplea para referirse únicamente al camino geométrico del robot. Como se ha podido observar en todo lo expuesto hasta el momento, será la primera acepción del término la utilizada en esta tesis.

El método de planificación de trayectorias mostrado en este trabajo se fundamenta en el algoritmo de Muñoz (Muñoz, 1995). Esta metodología divide el problema original de la planificación de la trayectoria global para un vehículo móvil en dos: en primer lugar, se genera un camino que evite las colisiones con los obstáculos estáticos del entorno, y que además respete las restricciones no holonómicas[7] del robot; en segundo lugar, se calcula un perfil de velocidades que verifique las restricciones de velocidad debidas al comportamiento cinemático y dinámico del vehículo, así como las limitaciones operacionales asociadas al entorno dinámico en el que se mueve. La elección de esta aproximación, frente a otras posibles ya analizadas en el segundo capítulo de la tesis, se debe principalmente a la importancia que otorga a las consideraciones cinemáticas y dinámicas del robot, lo que se traduce en una mayor seguridad y eficiencia a la hora de seguir las trayectorias calculadas.

Conviene anunciar en este punto que la fase de planificación de velocidades propuesta originalmente por Muñoz, y descrita en lo que sigue, será modificada para sortear los obstáculos móviles presentes en el entorno del robot, de manera que las limitaciones de velocidad referidas a este tipo de obstáculos no se consideran en el momento de generar la trayectoria, sino a posteriori. Así, la planificación temporal puede realizarse a máxima velocidad, y sólo se rectificaría en el caso de que produjese una colisión con otro u otros vehículos; otra ventaja radica en que puede ser extendida, de manera casi directa, al caso multirrobot. Al ser la base del método de evitación de obstáculos móviles, su análisis en profundidad, incluyendo dicha variación, se efectuará en el capítulo próximo.

3.2.1. Formalización del problema

El entorno de trabajo en el cual un robot móvil desempeña su labor puede considerarse como un conjunto de configuraciones (Lozano-Pérez, 1983) en las que puede encontrarse durante instantes de tiempo concretos.

7. Una restricción no holonómica que actúa sobre un robot móvil corresponde a una limitación de las velocidades admisibles que éste podría alcanzar, lo que se traduce en que ciertos movimientos del vehículo en el plano no son realizables.

Un robot es un objeto rígido al cual se le puede asociar un sistema de coordenadas móvil. La situación del vehículo en un momento determinado se establece mediante la relación entre el sistema de coordenadas global F_g, bajo el que se define todo el entorno de trabajo, y su sistema móvil de coordenadas locales asociado F_r (ver Figura 3.1)

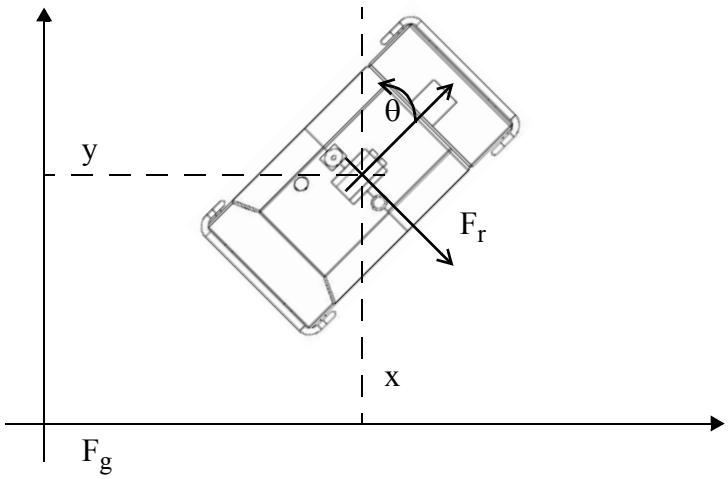


Figura 3.1. Sistemas de coordenadas global y sistema de coordenadas local asociado al robot

Una configuración q de un robot se define como un vector cuyas componentes proporcionan información completa sobre su estado actual. Dicho vector viene dado por dos componentes: la posición p y la orientación θ, de manera que una configuración puede expresarse como:

$$q = (p, \theta) = (x, y, \theta) \tag{3.1}$$

Se denomina espacio de configuraciones C del robot R a todas las configuraciones q que R puede alcanzar en su espacio de trabajo. El subconjunto de C ocupado por R cuando se encuentra en q se denota por $R(q)$. A ello es necesario añadir el hecho de que el espacio de trabajo se encuentra poblado por una serie de obstáculos estáticos, definidos como un conjunto de objetos rígidos $B = \{b_1, b_2, \ldots, b_k\}$ que se distribuyen por el espacio de configuraciones C; el conjunto de configuraciones de C ocupadas por un obstáculo se define por $b_i(q)$, por lo que el espacio libre de obstáculos C_l es el subconjunto del espacio C que viene dado por:

$$C_l = \left\{ q \in C / R(q) \cap \left(\bigcup_{i=1}^{q} b_i(q) \right) = \varnothing \right\} \tag{3.2}$$

Partiendo de estos conceptos, se considera que una ruta será una sucesión de configuraciones q, tal que la primera de ellas sea la configuración origen del robot q_o, y la última corresponda a la configuración objetivo q_f. Todas las configuraciones de la serie deben pertenecer al subconjunto C_l definido en la expresión (3.2). Es decir, una ruta Q_r que conecta la configuración de inicio con la de fin se especificará como:

$$Q_r = \{q_o, \ldots, q_f / q_i \in C_l\} \tag{3.3}$$

Para especificar dicho conjunto Q_r es necesaria la construcción de una función ruta, que se declara de la siguiente forma:

$$\tau:[0,1]\rightarrow C_l \text{ tal que } \tau(0) = q_o\,,\ \tau(1) = q_f \tag{3.4}$$

Además de esta restricción, la función τ deberá ser continua, suponiendo en principio que el robot es omnidireccional, lo que se refleja en:

$$\lim_{s \to s_0} \left\| \tau(s), \tau(s_0) \right\| = 0 \tag{3.5}$$

De todo lo anterior se desprende que la ruta queda definida como una sucesión de segmentos que conforman una línea quebrada cuyos extremos coinciden con las configuraciones de inicio y fin; de esta manera, se determina una vía por la cual puede desplazarse el robot móvil para cumplimentar su misión. Sea $G=\{g_1,\ldots,g_p\}$ dicho conjunto de aristas y extremos de la ruta; su extensión para la eliminación del supuesto de un vehículo omnidireccional se establece como la especificación de una función camino $P(\lambda)$ construida observando las siguientes propiedades:

$$\begin{gathered} \forall(\lambda \in [0,\tau]) \Rightarrow P(\lambda) \in C^2 \\ \forall g_i \in G \Rightarrow \exists \lambda \in \Re / P(\lambda) = g_i \\ P(0) = g_1 \\ P(\tau) = g_p \end{gathered} \tag{3.6}$$

La función camino es una función paramétrica, por lo que un punto p_i se especifica mediante la asignación de un valor concreto al parámetro λ de la función; es decir, una posición dentro del camino viene dada por $p_i=P(\lambda_i)=[P_x(\lambda_i),P_y(\lambda_i)]$. Si además se añaden la orientación θ y la curvatura κ, se obtiene una *postura*, que describe la posición, orientación y curvatura del robot cuando se encuentre situado sobre dicho punto del camino; de esta forma, la postura i-ésima del camino será $q_i=\{p_i,\theta_i,\kappa_i\}$, considerando las dos últimas componentes como:

$$\theta_i = \theta(P(\lambda_i)) = -\tanh\left(\frac{P_x'(\lambda_i)}{P_y'(\lambda_i)}\right) \qquad \kappa_i = \kappa(P(\lambda_i)) = \frac{d^2}{ds}P(\lambda_i) \tag{3.7}$$

Por tanto, una postura constituye una extensión del concepto de configuración, necesaria para poder asignar al camino ciertas propiedades -comentadas en la siguiente sección- que favorezcan su seguimiento.

La discretización de la función camino *P(λ)* se define como una aplicación biyectiva de $\Re$ en $\Re^4$, de forma que a cada valor λ_i del parámetro λ se le asocia una postura q_i. Así se construye una secuencia *Q* de posturas denominada camino.

Sea $L=\{\lambda_1,\lambda_2,...,\lambda_n\}$ el conjunto de los valores posibles del parámetro λ en el intervalo $[0,\Gamma]$, tal que $\lambda_1=0$ y $\lambda_n=\Gamma$, por lo que la definición del conjunto ordenado *Q* como camino es:

$$Q = \{[P(\lambda_i),\theta(P(\lambda_i)),\kappa(P(\lambda_i))]\forall(\lambda_i \in L)\} \tag{3.8}$$

Como puede observarse, en esta definición de camino no se han contemplado ni las propiedades necesarias para que pueda ser seguido correctamente, ni la posible presencia de obstáculos en el entorno. Sin embargo, ambas cuestiones tienen un peso fundamental a la hora de construir buenos caminos; por ello, a continuación, se añaden a la formulación de camino presentada hasta este punto.

Por un lado, como continuación al modelo basado en el espacio de configuraciones, sea $B=\{b_1,b_2,...,b_q\}$ el conjunto de obstáculos, y $R(q_i)$ la región del espacio que ocupa el robot en la configuración q_i. La condición para evitar que el vehículo colisione con algún obstáculo de su espacio de trabajo puede enunciarse así:

$$\forall q_i \in Q \Rightarrow \forall b_j \in B; R(q_i) \cap b_j = \varnothing \tag{3.9}$$

Por otra parte, y desde un punto de vista cinemático, la capacidad de un robot móvil para poder seguir un camino depende exclusivamente de la restricción de no holonomicidad de la curvatura. Esta limitación, si se considera *(dx,dy)* como el cambio de posición efectuado por el robot en un tiempo *dt*, puede concretarse como sigue:

$$-\sin\theta dx + \cos\theta dy = 0 \tag{3.10}$$

Con lo que, si la función camino $P(\lambda)$ no incumple esta condición, el camino Q será seguido apropiadamente. El efecto de la ecuación (3.10) se traduce en la limitación del radio de giro mínimo que el robot puede tomar; por tanto, si el radio de giro del vehículo es ρ_{min}, entonces $P(\lambda)$ debe verificar esta relación:

$$\forall \lambda_i \in [0,\Gamma] \Rightarrow \kappa(P(\lambda_i)) \in \left[-\frac{1}{\rho_{min}},\frac{1}{\rho_{min}}\right] \tag{3.11}$$

cuya consecuencia es la llamada condición de admisibilidad cinemática del camino Q:

$$\forall q_i \in Q \Rightarrow q_i = (x_i,y_i,\theta_i,\kappa_i);\frac{1}{\kappa_i} \geq \left\|\rho_{min}\right\| \tag{3.12}$$

El último paso para lograr la formalización matemática del concepto de trayectoria consiste en fijar una función velocidad $V(s)$, parametrizada por su longitud de arco, continua y acotada; también resultaría conveniente que sus dos primeras derivadas, esto es, la aceleración y las sacudidas, presenten las mismas características. Todo ello se resume en la expresión (3.13):

$$\forall s \in [0,S] \Rightarrow (V(s) \in C^2) \wedge \tag{3.13}$$
$$(V(s) \in [L_i^v(s),L_s^v(s)]) \wedge (V'(s) \in [L_i^a(s),L_s^a(s)]) \wedge (V''(s) \in [L_i^s(s),L_s^s(s)])$$

donde S representa la longitud del camino, mientras que las funciones $L_i^v(s)$, $L_s^v(s)$, $L_i^a(s)$, $L_s^a(s)$, $L_i^s(s)$ y $L_s^s(s)$ se refieren a las cotas inferior y superior de la función velocidad y de sus dos derivadas sucesivas. Dichas cotas se parametrizan respecto al espacio recorrido ya que, debido a las características del camino, del vehículo y de la tarea en cada momento, su valor se modifica a lo largo del mismo.

Al igual que en la fase de construcción del camino geométrico, una vez establecida la función de velocidad $V(s)$ debe procederse a su discretización. Como resultado de esta operación se obtiene el conjunto W de los valores que la referencia de velocidad adopta en cada intervalo de control. Considerando $S=\{s_1,...,s_n\}$ como el conjunto de valores empleados al discretizar $V(s)$, W se define de acuerdo con:

$$W = \{V(s_i) \forall s_i \in S\} = \{v_i, \ldots, v_n\} \tag{3.14}$$

Por fin, la trayectoria $\tilde{Q}$ se construye con los valores de posturas y velocidades obtenidos del camino Q y el conjunto W, y que resulta de la aplicación biyectiva entre ambos conjuntos:

$$\tilde{Q} = QW = \{\tilde{q}_1, \ldots, \tilde{q}_n\} / \tilde{q}_i = (q_i, v_i); q_i \in Q, v_i \in W \tag{3.15}$$

3.3. Planificación geométrica

El planificador espacial tiene como objetivo el cálculo de un camino libre de obstáculos estáticos desde una posición inicial hasta una final, garantizando ciertas propiedades geométricas de manera que posea buenas condiciones que faciliten su seguimiento. Tales condiciones se refieren a la continuidad en posición, orientación y curvatura, a la variación lineal de la curvatura y, finalmente, a la exigencia de acotar este parámetro dentro de los límites impuestos por la restricción no holonómica del vehículo. Para lograrlo, primero se calcula una ruta, sobre la que después se generará el camino definitivo; ambas etapas se explicarán más precisamente en este apartado.

Una ruta se ha definido como un conjunto de subobjetivos, que deben ser alcanzados consecutivamente por el robot para llegar a la localización final. Son numerosos los métodos de planificación de rutas que pueden hallarse en la bibliografía: grafos de visibilidad, diagramas de Voronoi, cilindros rectilíneos generalizados, descomposición en celdas, o campos potenciales (una revisión extendida de estas metodologías puede encontrarse en (Muñoz, 1995)). La técnica elegida en esta tesis se basa en los grafos de visibilidad, y es aplicable a entornos bidimensionales en los que los obstáculos se modelan como polígonos. El grafo se construye tomando como nodos los vértices de los polígonos correspondientes a los obstáculos, junto con la posición de partida y la posición de llegada; los arcos se generan entre aquellos nodos que se consideran *visibles*, entendiendo como tales aquellos que se pueden unir mediante un segmento rectilíneo que no intersecta con ningún obstáculo. Un algoritmo de búsqueda en grafos permitirá elegir la ruta que una la configuración inicial con la configuración final minimizando alguna función de coste. En todo este proceso no se han tenido en cuenta las posibles limitaciones físicas que afectan al movimiento del vehículo, y que pueden tener gran influencia en la capacidad del sistema de guiado del robot para poder ejecutar la ruta calculada. Por tanto, resulta conveniente introducir información cinemática como pauta al calcular una ruta con las mejores características para la generación del camino. Así nace el llamado *Grafo de Visibilidad Cinemático* (Muñoz *et al*., 1994b), una extensión a los grafos de visibilidad originales que sí considera características del vehículo a la hora de obtener la ruta.

En la etapa de generación del camino, partiendo de la ruta computada en el paso anterior, se construye una curva que comienza en el primer punto de la ruta y pasa por todos los puntos que la conforman, para terminar en el último punto de la misma. Esta curva continua se muestrea a intervalos regulares para producir el conjunto de posturas Q que será la referencia espacial utilizada por el algoritmo de seguimiento de caminos del vehículo. Existen diferentes métodos que proporcionan caminos con curvatura continua, pero carecen de una expresión formal cerrada para generar las referencias de curvatura, por lo que requieren unos costes computacionales que impiden su aplicación en tiempo real. Sin embargo, las técnicas de generación de caminos basadas en curvas β-Spline han demostrado su eficiencia para generar caminos con cambios de curvatura suaves, empleando tiempos de cálculo bajos (Muñoz et al. 1994a).

Como indicativo del tiempo necesario para llevar a cabo esta fase, puede resultar interesante aclarar que la construcción del grafo de visibilidad cinemático es una acción costosa computacionalmente (desde unos pocos segundos para entornos simples, hasta minutos en espacios de trabajo complejos), pero que puede realizarse fuera de línea; además la reconstrucción del grafo no es necesaria siempre que el entorno permanezca estático. Frente a ello, el método de generación de caminos con β-Splines se revela mucho más eficiente, siendo capaz de elaborar un camino de 25 metros en 34 mseg. sobre una estación de trabajo SPARC 2.

En resumen, la etapa de planificación geométrica ofrece un amplio campo de estudio, que queda fuera del ámbito de esta tesis. Es por esto por lo que no se hace mayor hincapié en ella; las referencias indicadas permiten su estudio y profundización con mayor detenimiento al lector interesado.

3.4. Planificación de velocidades

Una vez establecido el mecanismo para determinar las referencias espaciales que conformarán el camino a seguir por el vehículo, se procede a detallar cómo se genera el perfil de velocidades que lo transformará en una trayectoria. Antes de abordar las etapas de las que consta la planificación del perfil temporal, resulta conveniente explicar qué limitaciones de velocidad actúan sobre el vehículo, ya que restringen los valores que dicho perfil puede alcanzar.

3.4.1. Limitaciones de velocidad

Las razones que motivan la planificación temporal a lo largo de un camino surgen por la necesidad de satisfacer una serie de restricciones prácticas. Tales limitaciones pueden ser clasificadas en dos categorías, que se especifican a continuación.

3.4.1.1. Limitaciones físicas de velocidad

Las peculiaridades del vehículo, cuando éste se mueve a velocidades elevadas, afectan al proceso de seguimiento del camino; tanto, que en ese caso el error de seguimiento se incrementa significativamente si alguna de las siguientes restricciones no se cumple:

i) *Consideraciones Mecánicas (ME)*. Se deben a las características del sistema de locomoción del robot. Los motores y la cadena de tracción fuerzan unos valores máximos de velocidad y aceleración, que serán los que a la postre el robot pueda desarrollar.

ii) *Consideraciones Cinemáticas (CI)*. Estas limitaciones se deducen a partir del modelo cinemático del vehículo, que acota la velocidad y aceleración tope del punto de guía del vehículo como funciones de la velocidad de cada rueda. La restricción cinemática de velocidad establece que, cuando el robot sigue un camino, la velocidad angular del punto de guía del robot debe ser igual a la velocidad angular de cada una de las ruedas. Expresando esta afirmación matemáticamente, se llega a la ecuación (3.16), donde V es la velocidad del punto de guía, V_i se refiere a la velocidad de la rueda i-ésima, R corresponde al vector de radio de giro actual, y d_i representa el vector distancia desde el punto de guía a la i-ésima rueda.

$$V \leq \frac{\left\|\vec{R}\right\|}{\left\|\vec{R} + d_i\right\|} V_i \tag{3.16}$$

Las restricciones cinemáticas de aceleración se obtienen directamente de la derivación de las restricciones de velocidad cinemáticas.

iii) *Consideraciones Dinámicas (DI)*. Este tipo de limitaciones de velocidad involucra, por una parte, a las cuestiones relativas al comportamiento dinámico de los sistemas de guiado y tracción del robot y, por otra, a las fuerzas que actúan sobre el vehículo en movimiento. En el primer caso, el

sistema de dirección del robot precisa de cierto tiempo para responder ante un cambio de curvatura, por lo que la velocidad del vehículo debe estar limitada por esta circunstancia; sin embargo, debido al elevado número de elementos mecánicos y electrónicos que constituyen el sistema, así como a sus bucles de realimentación internos, es difícil lograr un modelo exacto del mismo, lo que significa que la obtención de la relación entre la velocidad y la variación máxima de la curvatura debe elaborarse para un algoritmo de seguimiento particular, mediante el estudio empírico de diversas situaciones reales. La segunda cuestión, relativa a las fuerzas laterales que actúan sobre el desplazamiento del robot, lleva al problema de la interacción rueda-suelo. Tales fuerzas deforman los neumáticos que están en contacto con el suelo, apareciendo un ángulo de desplazamiento que modifica la velocidad en el punto de guía del vehículo, apartándolo del camino previsto. Aunque existen representaciones para modelar este efecto, dependen de la estimación de parámetros a partir de la experimentación con una clase concreta de neumáticos y de suelo, por lo que establecer un modelo genérico para cualquier combinación rueda-terreno se convierte en una ardua labor.

Dejando a un lado estos inconvenientes a la hora de modelar el comportamiento dinámico del vehículo, el efecto final de las limitaciones dinámicas se traduce en que el robot se desvía del camino planeado. Pero, como tal situación únicamente surge en el caso de que el movimiento se realice a velocidades altas, es posible definir una velocidad tope que, basándose en la aceleración lateral máxima a_L, garantice la ausencia de riesgo de vuelco, según se indica en la expresión (3.17):

$$V \leq \sqrt{a_L \left\| \vec{R} \right\|} \tag{3.17}$$

donde R corresponde al radio de giro actual.

3.4.1.2. Limitaciones operacionales de velocidad

En un entorno dinámico, el robot lleva a cabo su tarea formando parte de un sistema superior, más complejo; debido a ello, debe adaptar su velocidad en función de la tarea que esté desempeñando en un momento dado. Por tanto, existen ciertas limitaciones de velocidad, denominadas operacionales, que deben sopesarse a la hora de calcular el perfil temporal a seguir por el vehículo:

i) *Distancia al objetivo (DO)*. La cercanía del robot al punto de destino obliga a fijar un valor de velocidad seguro, para lograr, de este modo, que el vehículo se detenga antes de introducirse en una zona con peligro de colisión. Ese valor seguro de velocidad es función de dos factores: la deceleración máxima a_{max} que proporciona el sistema de frenado, y la distancia actual al objetivo.

ii) *Velocidad de seguridad (VS)*. Bajo esta denominación se agrupan las restricciones de velocidad que sólo dependen de la interacción del robot con su ámbito de trabajo. Así, puede ser necesario que por motivos de seguridad (presencia de operadores humanos en las cercanías del vehículo), o por la sincronización con ciertos elementos del entorno, el robot navegue a una cierta velocidad a lo largo de un tramo de camino delimitado. Este tipo de limitaciones se define como una función a tramos, que asigna un velocidad máxima constante V_i a cada porción del camino definida por $(s_{i-1},s_i]$, según indica la expresión (3.18):

$$V(s) \leq \begin{cases} V_1 \ si \quad 0 \leq s \leq s_1 \\ \ldots \\ V_N \ si \ s_{N-1} < s \leq s_N \end{cases} \tag{3.18}$$

La evitación de obstáculos móviles constituye un caso especial dentro de estas limitaciones que será analizado en el siguiente capítulo.

3.4.2. Metodología de planificación de velocidades

Como ya se mencionó al formalizar el problema de la planificación de trayectorias, la conversión de un camino Q en una trayectoria $\tilde{Q}$ consiste, básicamente, en añadir una componente de velocidad a todas las posturas del camino inicial. Es decir, cada $q_i = (x_i,y_i,\theta_i,\kappa_i,s_i)$ debe transformarse en $\tilde{q}_i = (x_i,y_i,\theta_i,\kappa_i,s_i,v_i)$, donde v_i es la nueva componente de velocidad de la postura i-ésima. Esta modificación se puede efectuar estableciendo una función o perfil de velocidad *V(s),* parametrizada por la longitud de arco, y que se define en el plano espacio-velocidad (Shiller y Gwo, 1991). En este plano se representan los límites superiores de velocidad para cada una de las posturas q_i del camino Q; dichos márgenes se obtienen examinando las restricciones de velocidad estudiadas en la sección previa. Por tanto, la especificación de *V(s)* se produce de forma que contemple las

acotaciones de velocidad de todas las posturas, consiguiendo así un perfil temporal con características ventajosas para el seguimiento de la trayectoria. Esto equivale a que, en el plano espacio-velocidad, *V(s)* no traspase el área de seguridad determinada por las restricciones de velocidad, según se muestra en la Figura 3.2:

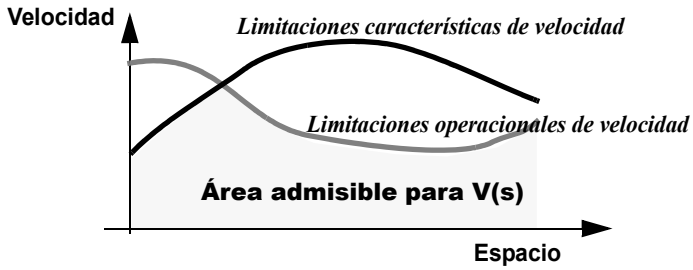


Figura 3.2. Limitaciones de velocidad en el plano espacio-velocidad

La definición de la función de velocidad *V(s)* se realiza en dos pasos consecutivos: primero, se muestrea el camino inicial, para seleccionar ciertas posturas relevantes de las que se computará su valor de velocidad tope; a continuación, tomando como base esas posturas escogidas, se genera una función de velocidad continua, de la que se podrán extraer las componentes de velocidad necesarias para lograr la trayectoria definida. Ambas etapas, denominadas respectivamente *Muestreo del Camino en Velocidad* y *Generación del Perfil de Velocidad*, se investigan con más detenimiento en lo que sigue.

3.4.2.1.Muestreo del Camino en Velocidad

El área válida para la definición de *V(s)* queda acotada por el límite superior de velocidad V_i para cada postura q_i perteneciente al camino, y puede especificarse según la ecuación (3.19):

$$V_i = \min(ME, CI(q_i), DI(q_i), DO(q_i), VS(q_i)) \tag{3.19}$$

donde *ME*, *CI*, *DI*, *DO* y *VS* corresponden, respectivamente, a los valores de velocidad máximos provistos por las restricciones mecánicas, cinemáticas, dinámicas, de distancia al objetivo y de seguridad que se enumeraron anteriormente.

Para muestrear el camino en velocidad, se elige un conjunto de posturas del camino $C=\{q^1,...,q^n\}$, denominado *conjunto de posturas de control*, que divide el camino en un conjunto de segmentos $S=\{S_q,...,S_{p-1}\}$, donde S_i se compone de las posturas del camino entre q^i y q^{i+1}. La selección de las posturas de C se basa en la naturaleza de las limitaciones de velocidad:

1. Las restricciones cinemáticas y dinámicas dependen de la curvatura κ del camino, por lo que la variación de esta magnitud tendrá su importancia al calcular el perfil temporal deseado. Así, las posturas con valores locales máximos o mínimos de curvatura se añaden al conjunto C.

2. En segundo lugar, y para satisfacer las restricciones de velocidad operacionales, se agregan a C las siguientes posturas:
   2.1. *Postura de Velocidad Máxima (PVM)*. Es la postura del camino en la que el vehículo alcanza su velocidad tope, empleando la máxima aceleración a_{max} proporcionada por los motores.
   2.2. *Postura de Frenado (PF)*. Es equivalente a la anterior, pero referida al frenado. Se calcula como la intersección entre la restricción mecánica *ME* y la de distancia al objetivo *DO*.
   2.3. *Posturas Adyacentes (PA)*. Se trata de un conjunto de posturas, formado por aquellas q^i que unen dos porciones limítrofes S_i y S_{i+1}, como se indica en (3.18).

Para construir C, las posturas escogidas según las reglas anteriores se ordenan siguiendo una ley creciente de la distancia s_i; todo el proceso puede resumirse en el siguiente algoritmo:

```
DivideCaminoEnSegmentos(Q)
    Calcula PVM;
    Calcula PF;
    Para i = 2 hasta final(Q) hacer
        Si i == PVM entonces MarcaNuevoSegmento;
        Si i == PF entonces MarcaNuevoSegmento;
        Si i == PA entonces MarcaNuevoSegmento;
        Calcular κ_actual;
        Calcular κ_anterior;
        Si ||κ_actual - κ_anterior|| < ε entonces MarcaNuevoSegmento;
    Fin Para
Fin DivideCaminoEnSegmentos
```

Pero el proceso de muestreo de velocidades aún está incompleto: es necesario añadir una referencia de velocidad máxima a cada miembro q^i de C. Dicho tope corresponde al mínimo valor señalado por todas las restricciones de velocidad en esa postura exacta, conforme se indica en la expresión (3.19). De esta operación resulta un *conjunto de velocidades de control* $V=\{0,v_1,...,v_{p-1},0\}$ para el camino Q, donde la primera componente -siempre nula- corresponde a la velocidad de partida para el segmento S_1, y los restantes elementos v_i son condiciones límite de velocidad entre los segmentos S_i y S_{i-1}, finalizando el camino también con velocidad cero.

Los conjuntos S y V, ambos con representables en el plano espacio-velocidad, son el resultado de la fase de Muestreo del Camino en Velocidad, y serán usados por la etapa de Generación del Perfil de Velocidad para lograr la función $V(s)$ deseada.

3.4.2.2. Generación del Perfil de Velocidad

La función de velocidad $V(s)$ se va a modelar a partir de un conjunto de funciones $\sigma_i(t)$ denominadas *curvas espacio-temporales*, estableciéndose la siguiente relación:

$$V(s) = \bigcup_{i=1}^{p} \frac{d}{dt}(\sigma_i^{-1}(s)) \tag{3.20}$$

La curva i-ésima está asignada al segmento de camino S_i, formado por la sucesión de posturas existentes entre q^i y q^{i+1}. Además, tal curva queda caracterizada por un conjunto de parámetros $\Pi_i=\{v_i,v_{i+1},s_i,t_i\}$, donde v_i y v_{i+1} (pertenecientes al conjunto V) se refieren a las velocidades de las posturas de comienzo y finalización del segmento, s_i corresponde a la longitud del segmento, y por último t_i representa el tiempo en el que debe navegarse el segmento, y que debe calcularse para poder definir completamente la curva espacio-temporal $\sigma_i(t)$.

El método que se va a emplear evalúa cada $\sigma_i(t)$ mediante dos polinomios cúbicos ${}^1\sigma_i(t)$ y ${}^2\sigma_i(t)$, cuya definición se expresa a continuación:

$$\begin{aligned} {}^1\sigma_i(t) &= {}^1\alpha_{i0}t^3 + {}^1\alpha_{i1}t^2 + {}^1\alpha_{i2}t + {}^1\alpha_{i3} \\ {}^2\sigma_i(t) &= {}^2\alpha_{i0}t^3 + {}^2\alpha_{i1}t^2 + {}^2\alpha_{i2}t + {}^2\alpha_{i3} \end{aligned} \tag{3.21}$$

La función ${}^{1}\sigma_i(t)$, con longitud de arco ${}^{1}s_i$, cubre la primera parte del segmento, mientras que ${}^{2}\sigma_i(t)$ describe la parte restante del segmento, equivalente a s_i-${}^{1}s_i$. Ambas cúbicas pueden parametrizarse mediante ${}^{1}\Pi_i=\{v_i, {}^{i}v_m, {}^{1}s_i, {}^{i}t_m\}$ y ${}^{2}\Pi_i=\{{}^{i}v_m, v_{i+1}, {}^{2}s_i, {}^{i}t_m\}$, respectivamente, si se denota como ${}^{i}t_m$ el valor mitad del tiempo t_i, y como ${}^{i}v_m$ el valor de velocidad en el punto de unión de los polinomios.

Al tratarse de funciones cúbicas, tanto los propios polinomios como sus derivadas primera y segunda son continuos, lo que significa que la continuidad en posición, velocidad y aceleración está garantizada dentro de los tramos asociados a ${}^{1}\sigma_i(t)$ y ${}^{2}\sigma_i(t)$. Sin embargo, también es necesario que se mantenga la continuidad en el punto de unión entre las dos cúbicas y entre segmentos adyacentes. Esta situación no es problemática para las derivadas de orden cero y orden uno, pero puede no ocurrir en el caso de las derivadas segundas; por ello, se imponen las siguientes relaciones:

$$\begin{aligned} {}^{1}\sigma''_i(0) &= {}^{2}\sigma''_i({}^{i}t_m) = 0 \\ {}^{1}\sigma''_i({}^{i}t_m) &= {}^{2}\sigma''_i(0) \end{aligned} \tag{3.22}$$

A partir de la formulación matemática indicada en (3.21), y de los parámetros definidos en los conjuntos ${}^{1}\Pi_i$ y ${}^{2}\Pi_i$, ${}^{1}\sigma_i(t)$ y ${}^{2}\sigma_i(t)$ pueden reescribirse matricialmente como sigue:

$$M \times \begin{bmatrix} {}^{1}\alpha_{i0} \\ {}^{1}\alpha_{i1} \\ {}^{1}\alpha_{i2} \\ {}^{1}\alpha_{i3} \end{bmatrix} = \begin{bmatrix} 0 \\ {}^{1}s_i \\ v_i \\ {}^{i}v_m \end{bmatrix} \tag{3.23}$$

$$M \times \begin{bmatrix} {}^{2}\alpha_{i0} \\ {}^{2}\alpha_{i1} \\ {}^{2}\alpha_{i2} \\ {}^{2}\alpha_{i3} \end{bmatrix} = \begin{bmatrix} {}^{1}s_i \\ {}^{2}s_i \\ {}^{i}v_m \\ v_{i+1} \end{bmatrix} \tag{3.24}$$

$$M = \begin{bmatrix} 0 & 0 & 0 & 1 \\ ({}^{i}t_m)^3 & ({}^{i}t_m)^2 & {}^{i}t_m & 1 \\ 0 & 0 & 1 & 0 \\ 3({}^{i}t_m)^2 & 2({}^{i}t_m) & 1 & 0 \end{bmatrix} \tag{3.25}$$

Ahora pueden despejarse los valores para los parámetros que caracterizan a las curvas:

$$\begin{aligned} {}^{i}t_m &= \frac{s_i}{v_i + v_{i+1}} & {}^{i}v_m &= \frac{v_i + v_{i+1}}{2} \\ {}^{1}s_i &= \frac{s_i(5v_i + v_{i+1})}{6(v_i + v_{i+1})} & {}^{2}s_i &= s_i - {}^{1}s_i \end{aligned} \tag{3.26}$$

Por último, resulta imprescindible analizar la relación entre las velocidades de inicio y de fin de cada tramo, de manera que se asegure que dentro de un segmento -concretamente, en el punto de unión de las dos cúbicas- no se viola la restricción de aceleración máxima. Matemáticamente, estas condiciones se expresan como sigue, considerando a_{max} como la aceleración tope que pueden desarrollar los motores del robot:

$$Si\ (v_i < v_{i+1})\ entonces\ v_{i+1} \le \sqrt{v_i^2 + a_{max}s_i}\,;\ si\ no\ v_{i+1} \ge \sqrt{v_i^2 - a_{max}s_i} \tag{3.27}$$

Una vez resueltos los valores desconocidos de los conjuntos de parámetros ${}^{1}\Pi_i$ y ${}^{2}\Pi_i$ usando las expresiones (3.26) y (3.27), el perfil de velocidad *V(s)* puede calcularse a partir de las curvas espacio-temporales de cada porción del camino, conforme se indica en la ecuación (3.20). Las propiedades más destacadas de este perfil temporal radican en su continuidad en velocidad y aceleración, por una parte, y en que además se respetan los límites de velocidad establecidos en el paso previo para cada segmento. A cada una de las posturas q_i de Q se le añadirá una referencia de velocidad extraída de *V(s)*, con lo que se construye completamente la trayectoria $\tilde{Q}$ buscada.

3.4.2.3. Purgado de segmentos innecesarios

La eficiencia de la generación del perfil de velocidad puede incrementarse minimizando el tamaño del conjunto de segmentos del camino *S*. Por ello, tras la etapa de división del camino en segmentos, en el presente trabajos se ha añadido como aportación un procedimiento de purga, mediante el cual se eliminan dos tipos de segmentos que surgen del

proceso de división del camino en segmentos, pero que no aportan información relevante a la hora de construir las referencias temporales. Una de estas clases de segmentos corresponde a los formados por una única postura; la otra afecta a aquellos segmentos consecutivos que verifican el requisito que a continuación se expone.

Sean S_{i-1} y S_i dos segmentos de camino consecutivos pertenecientes a S; sean v_{i-1}, v y v_{i+1} las restricciones de velocidad físicas al comienzo de S_{i-1}, en la postura frontera entre S_{i-1} y S_i, y al final de S_i, respectivamente. Si los valores de v_{i-1}, v y v_{i+1} son todos crecientes (o todos decrecientes), y además cumplen la relación expresada en (3.27), entonces dicha condición también se mantiene entre el primer término v_{i-1} y el último v_{i+1}. Esto significa que la curva espacio-temporal definida considerando S_{i-1} y S_i ofrece las mismas características que la definida usando un segmento con velocidad inicial v_{i-1} y velocidad final v_{i+1}. Por tanto, es posible fundir los segmentos S_{i-1} y S_i en un único tramo, reduciéndose así el tamaño del conjunto S.

Ambas condiciones se combinan dando lugar al siguiente algoritmo:

```
PurgadoDeSegmentos(S,V)
    Para cada (S_i -1, S_i) perteneciente a S
        Eliminar los segmentos de longitud 1 y actualizar V
    Fin Para
    Para cada (S_i -1, S_i) perteneciente a S
        Calcular el incremento de velocidad entre segmentos.
        Determinar el signo del incremento de velocidad.
    Fin Para
    Para cada (S_i -1, S_i) perteneciente a S
        Si incremento y signo de S_i-1 coincide con incremento y signo de S_i
            Unificar el segmento como S_i-1 y actualizar V
        Fin Si
    Fin Para
Fin PurgadoDeSegmentos
```

3.5. Conclusiones

La planificación de trayectorias óptimas para robots móviles en presencia de obstáculos estáticos, y que además ofrezcan buenas propiedades para los subsistemas de guiado y tracción de vehículos reales es un problema complejo, que normalmente demanda elevados requerimientos computacionales. En este capítulo se ha extendido una solución existente al problema basada en la combinación de dos etapas.

Por una parte, una fase de planificación geométrica produce caminos con características adecuadas para seguimiento, basándose en la construcción de grafos de visibilidad cinemáticos, y en la generación de caminos de curvatura continua mediante β-Splines. Por otra parte, las velocidades a lo largo del camino se establecen tras el análisis del conjunto de restricciones de velocidad que actúan sobre el vehículo, y que se deben tanto a cuestiones físicas del robot en sí, como a limitaciones que surgen de su interacción con el entorno. El perfil de velocidad se modela con una función cúbica, que aporta como beneficios tanto la continuidad en aceleración, como la poca exigencia en tiempo de computación -que además se ha mejorado eliminando parte del procesamiento necesario para obtener las referencias de velocidad. Todo ello facilita la aplicación en línea de esta metodología, mientras el vehículo está siguiendo el camino.

CAPÍTULO 4. Planificación de Trayectorias para un Sistema Multirrobot

4.1 Introducción

Una vez conocido el algoritmo de planificación de velocidades para un único robot, en este capítulo se inicia el estudio del proceso que permite transformarlo en un mecanismo válido para la planificación de trayectorias en un sistema con múltiples vehículos. Como parece lógico, la extensión del algoritmo monorrobot al caso multirrobot no puede realizarse de manera directa, y son necesarias ciertas adaptaciones para que la aproximación de base mantenga su utilidad. La principal modificación se refiere al hecho de que, en un entorno multirrobot, la presencia de los restantes vehículos del sistema obliga a considerar y resolver posibles colisiones a la hora de planificar las velocidades; por tanto, es necesario dotar al algoritmo original de mecanismos para la evitación de obstáculos móviles. A partir de este algoritmo de planificación de trayectorias que garantiza la ausencia de colisisiones con otros vehículos del sistema, puede establecerse un planificador de trayectorias multirrobot.

El capítulo se abre, para centrar formalmente el tema estudiado, con el comentario de los aspectos básicos relativos a la complejidad del problema de planificación de trayectorias multirrobot. A continuación, la sección tercera describe la metodología de planificación de trayectorias multirrobot, consistente en proyectar en un plano espacio-temporal la planificación de velocidades obtenida según el algoritmo expuesto en el capítulo previo, analizando y resolviendo los choques que puedan surgir entre robots. En la sección cuarta se presentan los resultados obtenidos aplicando el algoritmo de planificación de trayectorias multirrobot a robots móviles del Departamento de Ingeniería de Sistemas y Automática. La quinta sección se destina a exponer y analizar brevemente qué efectos añadidos implica la integración de la metodología de planificación de velocidades multirrobot en una arquitectura de control híbrida. Para finalizar, las conclusiones extraídas del trabajo mostrado cierran el capítulo.

4.2 Análisis de la complejidad computacional

Antes de examinar en detalle la metodología de planificación de trayectorias multirrobot presentada en la tesis, resulta interesante comentar, aunque de forma breve, los aspectos relativos a la complejidad del problema general del que se desprende; así se obtendrá una mejor comprensión del tipo de cuestión que se está afrontando. El estudio en profundidad de la eficiencia del método en sí quedaría fuera del alcance de este trabajo, por su amplitud y por su pertenencia a otras disciplinas cercanas más bien al área de Complejidad Computacional.

La eficiencia de un algoritmo se mide en términos de los recursos computacionales que consume, normalmente, tiempo de ejecución y espacio de almacenamiento. Ambos pueden expresarse como funciones del tamaño del problema a resolver; a su vez, este tamaño se define como la cantidad de datos necesaria para representar las instancias del mismo. La *complejidad temporal* de un algoritmo es pues la mayor cantidad de tiempo que éste necesita para resolver un problema de un tamaño dado. Puede expresarse usando la siguiente notación, denominada "O-grande": se dice que una función $f(n)$ es $O(g(n))$ siempre que exista una constante c tal que $|f(n)| \leq c|g(n)|$ | para todos los valores de $n \geq 0, c \geq 1$. Por ejemplo, si un algoritmo para ordenar una lista de *n* elementos tiene, en el peor caso, una complejidad temporal de $3n^2$, su complejidad puede expresarse como $O(n^2)$.

A partir de este concepto, puede crearse una clasificación de algoritmos muy simple: *algoritmos de tiempo polinomial* frente a *algoritmos de tiempo exponencial*. Así, para un problema de tamaño *n*, un algoritmo de tiempo polinomial se define como aquél cuya complejidad temporal es de la forma $O(p(n))$, donde $p(n)$ es una función polinomial de n. Cualquier algoritmo cuya complejidad temporal no pueda limitarse por una función polinomial será de tiempo exponencial, incluso cuando su complejidad no siga una función exponencial clásica. La clasificación pone de manifiesto que, en general, los algoritmos polinomiales son mucho más eficientes que los exponenciales; de hecho, un problema se considera intratable si no existe un algoritmo en tiempo polinomial que lo resuelva.

Según esta distinción entre complejidad temporal polinomial y exponencial, los algoritmos pueden agruparse en diferentes conjuntos:

- El conjunto P de algoritmos corresponde a los problemas de decisión que pueden resolverse mediante una máquina de Turing determinista en tiempo polinomial. Un problema de decisión es aquél cuya solución es "sí" o "no". Una máquina de Turing determinista sólo puede ejecutar una secuencia de operaciones, es decir, no admite paralelización. Todas las computadoras modernas son equivalentes a máquinas de Turing deterministas, incluso aunque soporten paralelismo por medio de planificación de tareas o un número finito de CPUs; en la actualidad, se especula con la posibilidad de que los ordenadores cuánticos sí correspondan al modelo de máquina de Turing no determinista. El conjunto P-completo lo integran aquellos problemas a los que puede reducirse todo problema P.

- El conjunto NP de algoritmos está definido como los problemas de decisión para los que se tiene un algoritmo de solución en tiempo polinomial, pero en una máquina de Turing no determinista. Es decir, no hay algoritmo determinista que lo resuelva en tiempo polinomial. En la actualidad se cree que $P \neq NP$; los algoritmos definidos en P son tratables, mientras que los incluidos en NP y superiores son intratables.

- El conjunto NP-completo está formado por los problemas NP a los que pueden reducirse todos los problemas de NP. Para ampliar detalles dentro de la teoría de la NP-completitud, puede consultarse la referencia clásica de Garey y Johnson (Garey y Johnson, 1979).

- El conjunto NP-duro corresponde a aquellos problemas, pertenezcan a NP o no, a los que cualquier problema de NP puede transformarse. Por tanto, los problemas duros de NP son el conjunto NP-completo.

- El conjunto PSPACE de algoritmos lo forman los problemas de decisión para los que las respuestas pueden encontrarse con recursos de almacenamiento (memoria) que son polinomiales en el tamaño de la entrada; el tiempo de ejecución no está restringido. Se sospecha que PSPACE no está contenido en NP, pero no se sabe con certeza. Sobre PSPACE se definen análogamente los conjuntos PSPACE-duro y PSPACE-completo.

En este punto, puede procederse al análisis de la complejidad del método de planificación de trayectorias multirrobot expuesto en este capítulo. Dicho método se engloba dentro del conjunto de los *Dynamic Movers Problems* o Problemas de Mudanzas Dinámicos, es decir, situaciones en las que debe planificarse "el movimiento libre de colisiones de un cuerpo B, que posee libertad para moverse en algún espacio S

bidimensional o tridimensional, que contiene varios obstáculos que se mueven a lo largo de trayectorias conocidas" (Reif y Sharir, 1985). En esta referencia, los autores demuestran que el Problema de la Mudanza Dinámico 3D es intratable: en el caso de que la velocidad del vehículo esté limitada, es PSPACE-duro; si no existen restricciones en la velocidad del robot, el problema es NP-duro.

En la literatura pueden encontrarse algoritmos eficientes que resuelvan subclases derivadas del problema general. Una de ellas corresponde a los Problemas de Evitación del Asteroide, definida en el capítulo segundo de esta tesis. Pues bien, también en (Reif y Sharir, 1985) se demuestra que existen algoritmos en tiempo polinomial para el caso bidimensional, si el robot es un polígono que se mueve entre un número constante de obstáculos; para el caso 3D, si el vehículo es un poliedro convexo, proponen algoritmos exponenciales en el tiempo o polinomiales en el espacio.

4.3 Planificación de trayectorias para un sistema multirrobot

La generación de trayectorias para un sistema multirrobot propuesto en esta tesis se basa en el algoritmo de planificación de velocidades expuesto en el capítulo anterior, unido a la descomposición camino-velocidad ya comentada en el análisis de los trabajos relacionados con el tema de la tesis. Esta nueva planificación de velocidades, desarrollada inicialmente en (Cruz, 1997), ha sido objeto de un estudio posterior más exhaustivo por parte de la autora, dentro de su tarea de investigación en el Departamento de Ingeniería de Sistemas y Automática; los frutos se traducen en diferentes mejoras hasta lograr un algoritmo de planificación de velocidades más exacto y ajustado a la realidad.

Esta sección recoge, para comenzar, una descripción con cierto detalle de la descomposición camino-velocidad, incluyendo también las limitaciones que presenta a la hora de su aplicación en entornos reales. El siguiente subapartado presenta un algoritmo de planificación de velocidades, denominado *a priori*, que corrige parcialmente tales deficiencias (Muñoz *et al*., 1998). La última subsección se dedica al algoritmo *integrado*, que combina de la manera más efectiva la planificación de velocidades estudiada con la descomposición camino-velocidad (Cruz *et al*., 1998; Muñoz *et al*., 1998; Muñoz *et al*., 1999), dando lugar al algoritmo de planificación de trayectorias multirrobot propuesto en el presente trabajo.

4.3.1. La descomposición camino-velocidad

La descomposición camino-velocidad (Kant y Zucker, 1986), también conocida en la literatura como *velocity tuning* o *sintonía de velocidad*, es una estrategia de navegación deliberativa sin perfil de velocidad, que sigue un enfoque desacoplado y considera la presencia de obstáculos móviles, según se explicó en el capítulo segundo. Como se recordará, la descomposición camino-velocidad divide el problema de la planificación de trayectorias en dos subproblemas: la planificación de un camino que evite los obstáculos estáticos, y la planificación de la velocidad a lo largo de ese camino de forma que se sorteen los obstáculos móviles que circulen por su entorno. El fundamento de esta descomposición se encuentra en el propio comportamiento humano. Puede pensarse en el caso de una persona que desea trasladarse de un punto a otro de un determinado entorno: una habitación, una calle... Para ello, primero calcula el punto destino, y después comienza a moverse hacia él con una determinada velocidad que únicamente modifica -incrementándola o decrementándola- en el caso de que un objeto móvil (otra persona, un coche...) se cruce en su camino. De este ejemplo se deduce que tanto el camino elegido como la velocidad a la que se recorre resultan fundamentales para evitar los obstáculos.

La descomposición camino-velocidad se efectúa en tres pasos:

- planificación de los posibles caminos que no colisionen con los objetos estáticos.
- elección de uno de esos caminos.
- planificación de la velocidad para evitar los obstáculos móviles a lo largo de ese camino.

Los dos primeros puntos se refieren a la planificación del camino geométrico. Al tratarse de un problema de planificación espacial, puede resolverse mediante cualquiera de los métodos existentes para tal fin ya conocidos.

El último punto, sin embargo, se refiere a la planificación de velocidades, que es el objeto de estudio principal de esta tesis, por lo que se va a comentar con un poco más de profundidad. La propuesta de Kant y Zucker se basa en el siguiente razonamiento: cualquier objeto que se mueve determina un hipervolumen en el espacio-tiempo. Si se realiza la intersección de ese volumen con el camino generado como resultado de la planificación del camino espacial, se obtienen los intervalos de tiempo en los cuales el robot y el obstáculo se

encuentran en posiciones que colisionan; para ello se utiliza como espacio de representación el llamado plano espacio-tiempo, o plano *s x t*). La forma de estas intersecciones robot- obstáculo, que se denominan *regiones prohibidas*, depende de tres factores principales:

- el tamaño y la forma del obstáculo.
- la trayectoria del obstáculo.
- el camino espacial seguido por el robot.

Las regiones prohibidas son funciones de tiempo difíciles de computar, de forma que, habitualmente, se representan mediante rectángulos: mayor precisión en el cálculo de las mismas implica mayor complejidad de cómputo, sin que los resultados mejoren significativamente. En la Figura 4.1 puede observarse cómo, a partir de la intersección de la trayectoria del obstáculo móvil y el camino del robot, surge una región prohibida, y cómo ésta se plasma en el plano *s x t*.

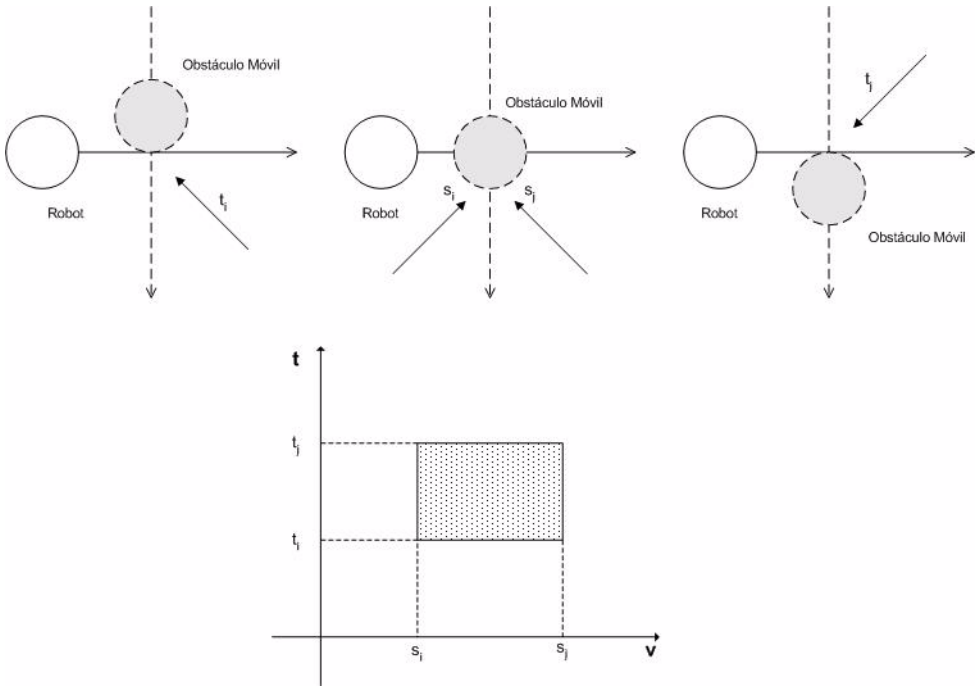


Figura 4.1. Generación de las regiones prohibidas en el plano *s x t*.

Una vez establecidas las zonas de colisión entre el vehículo y los obstáculos móviles, que serán los restantes elementos del sistema multirrobot, se definirá una función de velocidad que garantice la ausencia de choques entre los mismos; la condición suficiente para que no existan colisiones es que la curva que representa la distancia recorrida por el

robot respecto al tiempo no intersecte con las regiones prohibidas ya calculadas. Para ello se crea un grafo de visibilidad sobre la representación del espacio camino-tiempo que incluye regiones prohibidas. Los nodos de este grafo corresponden a los vértices de las zonas vedadas, junto a los puntos inicial y final del camino del robot, mientras que las aristas que unen los distintos nodos se caracterizan por no poder atravesar las áreas prohibidas. La planificación de velocidad final se obtiene mediante la búsqueda en dicho grafo de un camino que una el nodo inicial con el nodo final, como representa la Figura 4.2:

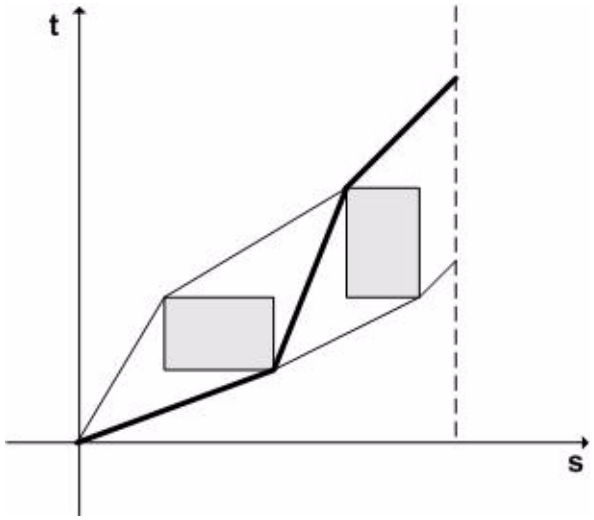


Figura 4.2. Planificación de velocidades según la descomposición camino-velocidad

4.3.1.1. Limitaciones de la descomposición camino-velocidad

Como ya se comentó en el capítulo dedicado a los trabajos relacionados, en algunas situaciones este método es incapaz de determinar una trayectoria, a pesar de que ésta exista. Por ejemplo, cuando un robot se mueve siguiendo a un vehículo, o cuando finaliza su marcha sobre el camino geométrico de otro elemento del sistema.

Además, los propios Kant y Zucker reconocen que la descomposición camino-velocidad proporciona buenas soluciones si los obstáculos se mueven de forma relativamente perpendicular al camino seguido por el robot; de nuevo, esta condición es demasiado limitante y no tiene por qué darse en la realidad. Por ello, las soluciones presentadas a lo largo de la sección resuelven esta traba realizando un análisis intensivo a la hora de verificar los posibles cruces robot-obstáculos así como al construir las regiones prohibidas, de manera que:

- inicialmente, se realiza la detección de cruces entre los trayectos geométricos del robot y de los obstáculos, considerando la posibilidad de que estén muy próximos o sean incluso paralelos.

- a continuación, se construyen las regiones prohibidas asociadas a cada cruce, teniendo en cuenta que éste puede que no sea paralelo.

- por último, las regiones prohibidas generadas se someten a un filtrado, que elimina aquellas que no son válidas, y adapta aquellas que son parcialmente factibles, según se indica en la Figura 4.3. En ella se muestran en rojo aquellas regiones prohibidas incorrectas ya que todas sus coordenadas, espaciales o temporales, quedan fuera del plano *s x t*; serán eliminadas y no se consideran al realizar la planificación de velocidades. Las regiones prohibidas de color naranja corresponden presentan alguna de sus coordenadas exteriores al plano espacio-velocidad, y por ello pueden adaptarse para ser válidas. Un caso particular dentro de este tipo de regiones lo constituye la región siete: aunque sus coordenadas se encuentran parcialmente en el plano *s x t*, no es válida, ya que está situada en el origen de coordenadas, representando así una situación de riesgo en el mismo punto de partida del robot. Finalmente, las regiones prohibidas verdes tienen todas sus coordenadas dentro del plano y no necesitan modificarse.

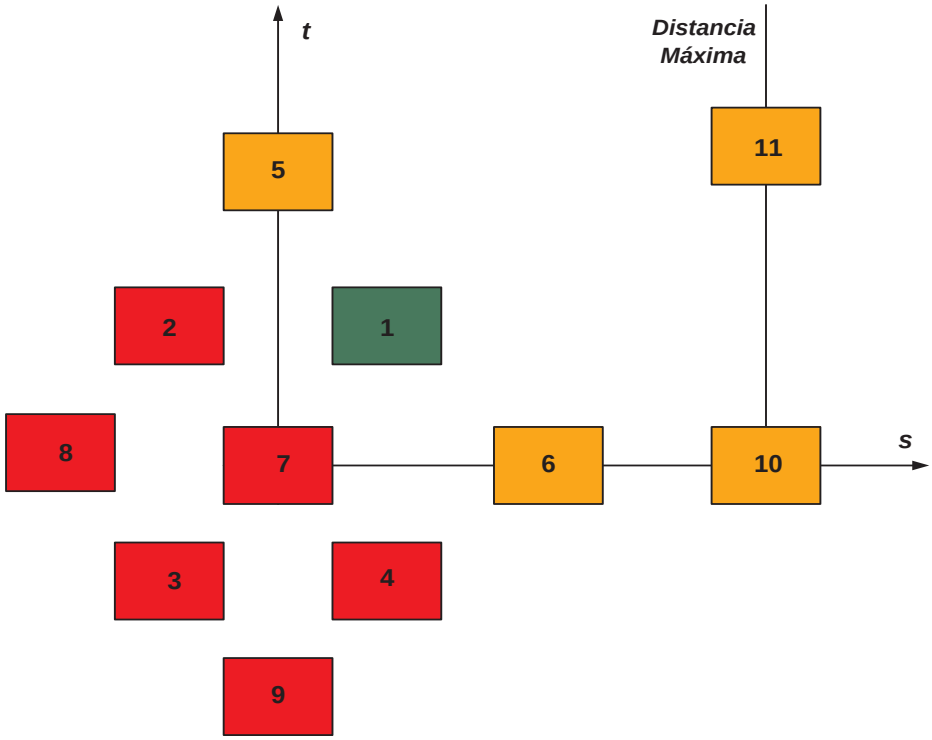


Figura 4.3. Filtrado de regiones prohibidas en el espacio *s x t*.

Por otra parte, todo el desarrollo de la sintonía de velocidad se ha efectuado, hasta el momento, bajo la consideración de que el robot se trata como un vehículo puntual, situación no muy realista. Kant y Zucker proponen como remedio la aproximación de Lozano-Pérez, que permite considerar el robot punto a costa de engrosar convenientemente los obstáculos. Sin embargo, la solución aplicada en el algoritmo propuesto consiste en extender las regiones prohibidas en función del tamaño del robot móvil para el que se computa el perfil de velocidades.

Además de las limitaciones contempladas por los autores de la metodología, a la hora de implantarla aparecen nuevas dificultades. A continuación se desglosan junto con una breve descripción de las distintas soluciones aportadas en esta tesis, y que serán ampliadas en subsecciones posteriores:

- la planificación de velocidad generada se compone de una serie de tramos de velocidad constante entre los que no existe continuidad, lo que supondría la aparición de aceleraciones infinitas. En ninguna de las soluciones propuestas a lo largo del capítulo, tal cuestión supone un problema, ya que la descomposición camino-velocidad se combina adecuadamente con el algoritmo de planificación de velocidades de Muñoz para generar perfiles de velocidad continuos también en aceleración.

- la descomposición camino-velocidad no valora la posibilidad de que existan otras restricciones de velocidad que actúan sobre el robot a lo largo de su camino. Sin embargo, este hecho puede resultar de gran importancia, ya que al unir el perfil de velocidad generado por la sintonía de velocidad con otras limitaciones que resulten más restrictivas, puede verse modificado para respetarlas y, por tanto, dejar de garantizar la ausencia de colisiones. La aplicación de la descomposición camino-velocidad a priori sí resuelve esta dificultad; en el caso de la aplicación integrada, esta dificultad ni siquiera aparece, como se comprobará posteriormente.

- la sintonía de velocidad supone que el camino se recorre en todo momento a velocidad máxima; en la realidad esa situación es imposible, puesto que es necesario cierto tiempo para alcanzar dicha velocidad, según se indica en (Muñoz, 1995). La solución que este mismo autor propone consiste en modificar las regiones prohibidas calculadas mediante un desplazamiento en el eje temporal, para la cual se restan a los límites temporales de la zona

vedada un valor *t* obtenido a partir de aproximaciones de las referencias de velocidad que en ese momento sigue el robot. Al igual que en el apartado anterior, este problema se manifiesta únicamente en la solución a priori, ya que en la solución integrada esta dificultad directamente se obvia.

4.3.2. Aplicación a priori de la descomposición camino-velocidad

La primera solución propuesta para mejorar la sintonía de velocidad de Kant y Zucker se ha denominado *a priori*, ya que se utiliza como herramienta para generar restricciones operacionales del tipo velocidad de seguridad, definidas en el capítulo tercero. Es decir, la descomposición camino-velocidad se aplica *antes* de generar el perfil de velocidad, con el objetivo de establecer nuevas limitaciones de velocidad que habrá que examinar según el algoritmo de planificación de velocidades expuesto con anterioridad.

Pero, como se ha comentado en el subapartado previo, esta idea debe alterarse parcialmente para que considere las características del vehículo y proporcione un perfil de velocidad realmente seguro. En otras palabras, si los límites cinemáticos y dinámicos que afectan al robot no son tenidos en cuenta, la planificación resultante no asegura la evitación de los restantes elementos del sistema multirrobot. Los cambios efectuados sobre la descomposición camino-velocidad para que, al aplicarla a priori, genere restricciones de velocidades, se describen en lo que sigue.

4.3.2.1. Presencia de otras limitaciones de velocidad

Sea *DVG(N,A)* el grafo usado para resolver la planificación de velocidad en el plano espacio-tiempo según se propone en la descomposición camino-velocidad original. En dicho grafo, *N* corresponde al conjunto de nodos, y *A* representa el conjunto de aristas. Sea A_k una arista perteneciente a *A*, y sean $n_s = (s_s,s_t)$ y $n_f = (s_f,t_f)$ sus nodos iniciales y finales, con n_s y n_f perteneciendo a *N*.

La pendiente de la arista A_k define la velocidad con la que se atraviesa la porción de camino entre sus extremos. El valor de la pendiente de toda arista del grafo se encuentra acotado, evidentemente, por la velocidad tope que pueden desarrollar los motores del robot, de manera que:

$$\left|\frac{ds}{dt}\right| = \left|\frac{s_f - s_s}{t_f - t_s}\right| \leq V_{max} \qquad (4.1)$$

Sin embargo, la presencia de otras restricciones de velocidad que actúan sobre el robot móvil modifica la velocidad tope permitida para una arista dada. Por ello, debe emplearse algún tipo de información adicional para calcular una velocidad máxima segura para cada arista del grafo.

Entonces, sea R_k un subconjunto del camino geométrico Q que sigue el robot; R_k se define como:

$$\begin{aligned} R_k &= \{q_s, \ldots q_f\} / s(q_i) \in [s_s, s_f] \\ donde \quad & s(q_i) = s_i \Leftrightarrow q_i = \{x_i, y_i, \theta_i, \kappa_i, s_i\} \end{aligned} \tag{4.2}$$

A partir de esa expresión, la velocidad máxima V_k bajo las restricciones cinemáticas y dinámicas ejercidas sobre A_k puede establecerse como:

$$V_k = min\{min(ME(q), CI(q), DI(q), DO(q), VS(q)); \forall (q \in R_k)\} \tag{4.3}$$

siendo *ME*, *CI*, *DI*, *DO* y *VS* los límites de velocidad físicos y operacionales ya estudiados. La velocidad asociada a la arista A_k se considera válida, y por tanto, susceptible de ser alcanzada por el vehículo con garantías de seguridad, si verifica la siguiente relación:

$$\left| \frac{s_f - s_s}{t_f - t_s} \right| \leq V_k \tag{4.4}$$

Así, aquellas aristas del conjunto de aristas A que no cumplan la expresión 4.4 deben ser eliminadas del mismo y, por tanto, del grafo.

4.3.2.2. Adición de una restricción de aceleración

Como ya se ha comentado, la sintonía de velocidad supone que el robot viaja con una velocidad constante a lo largo de cada arista perteneciente al grafo *DVG(N,A)*. Esta presunción, empero, no es realista, y el robot invierte una cierta cantidad de tiempo en alcanzar una referencia de velocidad. Para reflejar este hecho oportunamente, las regiones prohibidas que modelan los obstáculos móviles en el plano *s x t* deben ser trasladadas hacia abajo a lo largo del eje temporal; resulta necesario, pues, calcular dicho desplazamiento temporal.

Desde un punto de vista cinemático, el tiempo T_c requerido por el robot para adquirir una velocidad de referencia V_{ref} con una aceleración constante a_{max} se define como V_{ref}/a_{max}. El algoritmo de planificación de velocidades desarrollado en el capítulo tercero utiliza un perfil de aceleración determinado por las funciones ${}^1\sigma''_i(t)$ y ${}^2\sigma''_i(t)$, de lo que se desprende que para lograr V_{ref} es necesario un tiempo $T_c = 2V_{ref}/a_{max}$. Para recorrer la arista A_k con extremos n_s y n_f, el robot móvil necesita el tiempo T_c junto con la diferencia entre t_s y t_f. Uniendo todo lo expuesto, el desplazamiento ${}^iT_c$ aplicado a la región prohibida i-ésima puede expresarse como:

$$ {}^iT_c = i\left[\frac{2V_{ref}}{a_{max}} + \rho\right] \tag{4.5}$$

donde V_{ref} y a_{max} se toman como la velocidad y aceleración máximas del vehículo, y ρ es un factor experimental de seguridad que permite ajustar la expresión a casos reales.

4.3.3. Aplicación integrada de la descomposición camino-velocidad

Frente al planteamiento expuesto en el apartado anterior, el enfoque que aquí se presenta está basado en la idea de trasladar la planificación de velocidades *V(s)* directamente al plano *s x t,* que incluirá como regiones prohibidas las zonas de colisión entre el vehículo y los obstáculos móviles que conviven en su entorno. Esta idea se resume en la Figura 4.4:

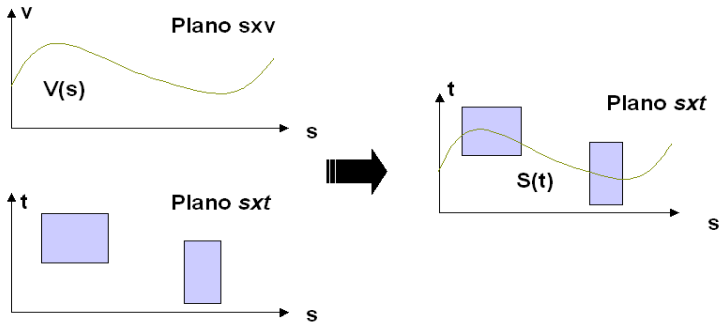


Figura 4.4. Planificación de la función V(s) en el plano espacio-tiempo

Si el perfil de velocidad calculado no interfiere con ninguna región prohibida, puede considerarse seguro; en caso de que aparezcan intersecciones entre ellos, será necesario modificar la planificación temporal de manera que sortee el obstáculo u obstáculos que provocan la colisión. De esta manera, la descomposición camino-velocidad se *integra* junto con el algoritmo de planificación de velocidades para calcular el perfil temporal; es decir, se transforma en una herramienta con la que se computan las velocidades a seguir. Para ello, deben proveerse mecanismos para el mapeado de la función de velocidades en el plano espacio-tiempo, para la detección de intersecciones con las regiones prohibidas, y para la adaptación de la función de velocidad que evite los choques con otros elementos del espacio de trabajo.

Como se declaró en el capítulo tercero, la planificación de velocidades *V(s)* viene dada mediante la unión de las denominadas curvas espacio-temporales. Puesto que originalmente *V(s)* se encuentra definida en el plano espacio-velocidad, es necesario migrarla al plano *s x t*, conforme indica la expresión (4.6):

$$V(t) = \bigcup_{i=1}^{p} \sigma'_i(t) \tag{4.6}$$

También se señaló en el capítulo previo que cada $\sigma_i(t)$ se especifica gracias al conjunto de parámetros $\Pi_i=\{v_i,v_{i+1},s_i,t_i\}$. Por tanto, las componentes (s_i,t_i) permiten la representación de la curva espacio-temporal en el plano *s x t*.

El perfil *V(s)* podrá considerarse seguro si ninguna $\sigma_i(t)$ toca alguna región prohibida. Sea OB_j una región prohibida en el plano *s x t*, que corresponde al j-ésimo cruce entre la trayectoria de un obstáculo móvil y la trayectoria del robot. Debido a que la referencia de velocidad calculada es la máxima que el vehículo puede aplicar en cada momento bajo el efecto de las restricciones físicas y operacionales que actúan sobre él, para evitar el impacto con el obstáculo móvil debe disminuir su velocidad y permitir que éste atraviese la zona de riesgo antes de que el robot la alcance.

Sin embargo, comprobar la existencia de intersecciones entre todas las $\sigma_i(t)$ y OB_j del plano espacio-tiempo es un proceso costoso. En su lugar, el método aquí explicado utiliza un conjunto de áreas *hull-convex*, llamadas *zonas de seguridad* Z_i, que albergan las distintas $\sigma_i(t)$. De esta manera, el algoritmo propuesto emplea una aproximación geométrica para detectar las colisiones entre cada Z_i y OB_j; en caso de que así suceda, tanto la curva espacio-temporal como su zona de seguridad asociada se rectifican para eludir dicha situación peligrosa.

Partiendo de estas ideas básicas, las siguientes subsecciones tratan de los diferentes aspectos que conducen al algoritmo de generación de trayectorias para sistemas multirrobot.

4.3.3.1. Construcción de las zonas de seguridad

La construcción de la zona de seguridad Z_i relativa a la curva $\sigma_i(t)$ se fundamenta en el siguiente lema:

Lema I. *La forma de la función espacio-temporal $\sigma_i(t)$ ligada al segmento S_i es siempre o cóncava o convexa. En otras palabras, los subsegmentos ${}^1s_i(t)$ y ${}^2s_i(t)$ que conforman S_i deben ser, simultáneamente, cóncavos o convexos.*

Prueba. Sean ${}^1\sigma_i(t)$ y ${}^2\sigma_i(t)$ los polinomios cúbicos que integran $\sigma_i(t)$. Sus derivadas segundas se definen como:

$$\begin{aligned} {}^1\sigma''_i(t) &= 6\,{}^1\alpha_{i0}t \\ {}^2\sigma''_i(t) &= 6\,{}^2\alpha_{i0}t + 2\,{}^2\alpha_{i1} \end{aligned} \tag{4.7}$$

donde ${}^1\alpha_{i0}$ se obtiene resolviendo la expresión:

$${}^1\alpha_{i0} = \frac{-(v_i - v_{i+1})(v_i + v_{i+1})^2}{s_i^2} \tag{4.8}$$

De igual forma:

$$\begin{aligned} {}^2\alpha_{i0} &= \frac{(v_i - v_{i+1})(v_i + v_{i+1})^2}{6s_i^2} \\ {}^2\alpha_{i1} &= \frac{-(v_i - v_{i+1})(v_i + v_{i+1})}{2s_i} \end{aligned} \tag{4.9}$$

Será el signo de las segundas derivadas el que determine la concavidad o convexidad de las funciones ${}^1\sigma_i(t)$ y ${}^2\sigma_i(t)$. Éste depende de la relación entre las velocidades de inicio y fin del segmento S_i. Para el caso de segmentos de velocidad creciente, se cumple que $v_i < v_{i+1}$, por lo que ${}^1\sigma_i(t)$ y ${}^2\sigma_i(t)$ son curvas cóncavas. En el caso opuesto, la desigualdad $v_i > v_{i+1}$ fuerza unas derivadas segundas negativas, por lo que ambos subsegmentos son convexos. **Fin de Prueba.**

La aplicación del lema anterior implica que una zona de seguridad Z_i puede modelarse, en el plano espacio-tiempo, como un triángulo, según se ilustra a continuación:

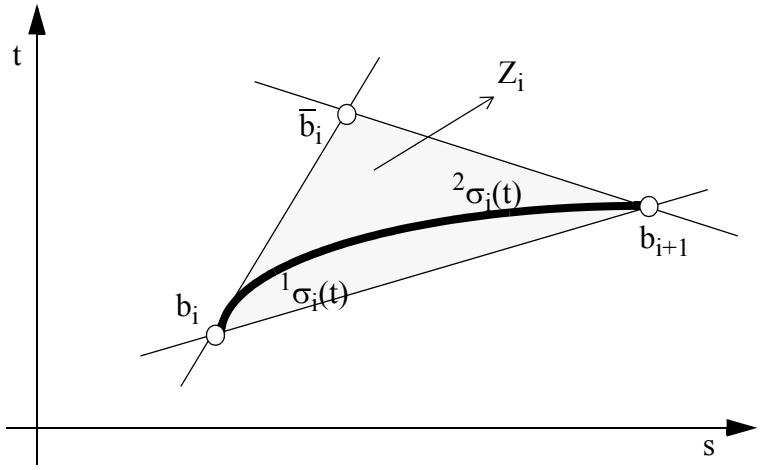


Figura 4.5. Zona de seguridad Z_i relativa a la curva espacio-temporal $\sigma_i(t)$

Esta figura presenta la zona de seguridad como un triángulo sombreado delimitado por las siguientes líneas:

- El segmento de línea que comienza en el punto b_i y finaliza en b_{i+1}, donde ambos puntos se definen como:

$$b_n = \left(\sum_{j=1}^{n} s_j, \sum_{j=1}^{n} t_j\right) \qquad (s_j,t_j) \in \Pi_j \tag{4.10}$$

- La línea tangente a ${}^1\sigma_i(t)$ en el punto b_i, cuya pendiente es v_i.
- La línea tangente a ${}^2\sigma_i(t)$ en el punto b_{i+1}, con pendiente v_{i+1}.

En resumen, la zona de seguridad Z_i queda encuadrada entre los vértices b_i, b_{i+1} y la intersección $\overline{b}_i$ de las tangentes de las funciones que componen la curva espacio temporal.

4.3.3.2. Modificación de las zonas de seguridad

A la hora de modificar las zonas de seguridad, son dos las opciones que se han evaluado. La primera de ellas desplaza la zona de seguridad una cierta cantidad constante de tiempo, hasta que la colisión con la región prohibida haya desaparecido. La segunda posibilidad, más exacta, determina qué tiempo es necesario para sortear el obstáculo; en caso de que la zona de seguridad no pueda asumir tal deceleración debido a las restricciones establecidas en (3.27), propaga hacia las zonas de seguridad previas el tiempo restante. Ambas soluciones se explican con mayor profundidad a continuación.

Modificación basada en tiempo constante

Como se ha mencionado a lo largo del capítulo, el robot debe aminorar la velocidad cuando se localiza una interferencia entre una zona de seguridad Z_i y un obstáculo móvil OB_j. En el plano *s x t*, esto significa que Z_i se mueve hasta que $\mathrm{OB_j}$ quede bajo ella. Un cambio en la zona de seguridad Z_i implica un cambio en sus funciones ${}^{1}\sigma_i(t)$ y ${}^{2}\sigma_i(t)$, y obviamente, los valores que las definen, esto es, $\{v_i, {}^{i}v_m, {}^{1}s_i, {}^{i}t_m\}$ y $\{{}^{i}v_m, v_{i+1}, {}^{2}s_i, {}^{i}t_m\}$ también varían. Esta situación incrementa el tiempo de navegación *t* para recorrer el segmento de camino S_i, por lo que las velocidades de inicio y fin del mismo deben ser recalculadas. Las relaciones entre velocidades y tiempo de las fórmulas de la expresión 3.26 indican cómo determinar estos nuevos valores, dependiendo de la velocidad creciente o decreciente del segmento S_i:

$$\begin{aligned} v_i < v_{i+1} &\Rightarrow v_{i+1} = \frac{s_i}{\tilde{t}_i} - v_i \\ v_i > v_{i+1} &\Rightarrow v_i = \frac{s_i}{\tilde{t}_i} - v_{i+1} \end{aligned} \tag{4.11}$$

donde $\tilde{t}_i = t_i + \Delta t_i$ es el nuevo tiempo de navegación para el segmento. En segundo lugar, es necesario considerar los efectos laterales producidos por estas modificaciones en los segmentos adyacentes: concretamente, en el segmento posterior S_{i+1}, en caso de que S_i sea un tramo de velocidad creciente, o en el segmento previo S_{i-1}, si S_i es un tramo de velocidad decreciente.

De esta manera, el algoritmo de evitación de colisiones entre zonas de seguridad y regiones prohibidas se implementa así:

```
EvitaObstáculoMóvil(S,V,OB_j,Π_i)
    Z_i = CalculaZonaDeSeguridad(v_i,v_i+1,s_i)
        Mientras Colisión(OB_j,Z_i) hacer
            t_i = t_i + PASO
            Si v_i < v_i+1 entonces
                v_i+1 = (s_i/t_i)-v_i
                Modifica segmento S_i+1
            Si no
                v_i= (s_i/t_i)-v_i+1
                Modifica segmento previo S_i-1
            Fin Si
        Fin Mientras
Fin EvitaObstáculoMóvil
```

Como se observa, los parámetros que *EvitaObstáculoMóvil* precisa son el obstáculo representado por la región prohibida OB_j, el conjunto de segmentos del camino S, el conjunto de velocidades de control V, y el conjunto de datos Π_i ligados al segmento actual S_i. La primera operación a realizar es el cálculo de la zona de seguridad Z_i para el i-ésimo segmento. Una vez que Z_i se ha construido, dos acciones se llevan a cabo iterativamente en un bucle: primero se modifica el tiempo de navegación t_i para S_i, y después se obtienen los nuevos valores de la velocidad de inicio o fin del segmento, teniendo en cuenta los efectos laterales sobre los segmentos contiguos. El bucle finaliza cuando no se detecta ninguna colisión con el obstáculo OB_j. El resultado del algoritmo es el conjunto de velocidades de control actualizadas, lo cual avala la evitación del choque.

Para finalizar, señalar que el valor PASO representa el incremento de tiempo necesario para navegar a lo largo de S_i evitando los vehículos móviles en sus cercanías. Esta valor se fija mediante reglas obtenidas por experimentación, y se basa en la distribución de los obstáculos en el plano *s x t*.

Modificación basada en propagación hacia atrás del tiempo

Bajo este enfoque, y manteniendo la idea de que el robot debe reducir su marcha si se detecta una colisión entre una zona de seguridad Z_i y un obstáculo móvil OB_j, el segmento de camino S_i correspondiente a Z_i se reemplaza por dos nuevos segmentos de camino denotados SA_i y SB_i, de forma que se logra eludir el obstáculo OB_j a través de un nuevo punto espacio-temporal b_k. El esquema de esta situación, representado en el plano *s x t*, se muestra en la Figura 4.6:

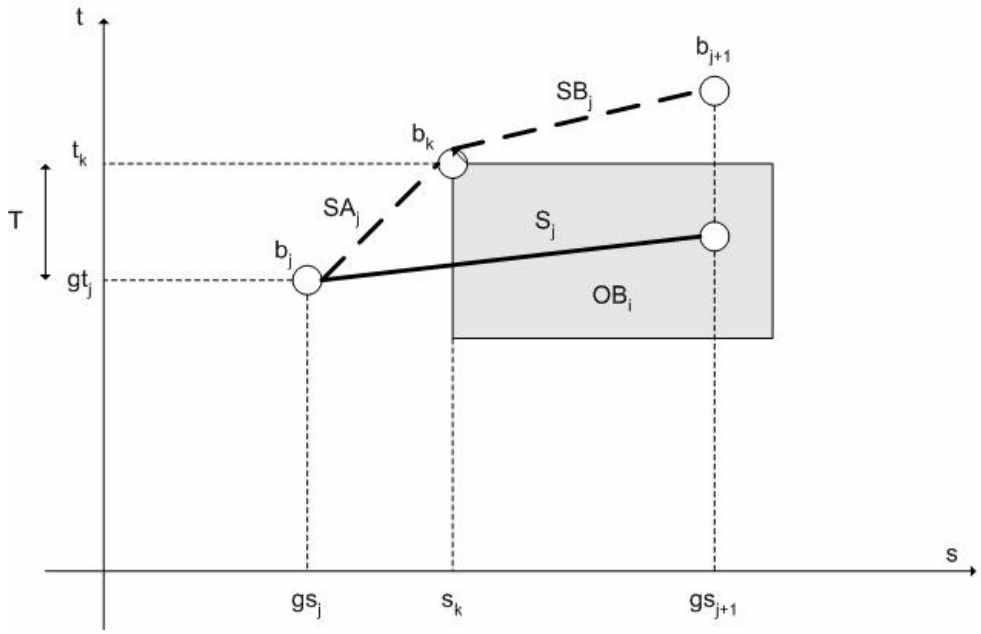


Figura 4.6. Situación de colisión y modificación del segmento de camino

El hecho de añadir b_k como base para la evitación de la colisión implica que tanto el conjunto S de segmentos del camino, como el conjunto V de velocidades de control deben actualizarse para incluirlo. Así, si v_k es la referencia de velocidad en el punto b_k, entonces:

$$\begin{aligned} S &= \{S_1,\ldots, S_{i-1},SA_i,SB_i,S_{i+1},\ldots,S_p\} \\ V &= \{0,v_1\ldots, v_i,v_k,v_{i+1},\ldots,v_{p-1},0\} \end{aligned} \tag{4.12}$$

Los conjuntos de parámetros para los nuevos segmentos SA_i y SB_i vienen dados por $\Pi_i^A = \{v_i, v_k, s_i^A, t_i^A\}$ y $\Pi_i^B = \{v_k, v_{i+1}, s_i^B, t_i^B\}$, donde v_i y v_{i+1} son las velocidades planificadas para b_j y b_{j+1}, respectivamente.

El valor de velocidad v_k debe garantizar que la navegación del segmento de camino SA_i se completa en un tiempo T, obtenido a partir de la siguiente expresión:

$$v_k = \frac{s_j^A}{T} - v_j \tag{4.13}$$

donde T, denominado *tiempo de evitación*, se establece como $T = t_k - gt_j$, según puede comprobarse en la Figura 4.6. Es decir, T equivale al tiempo transcurrido entre el punto espacio-temporal b_j y el nuevo punto b_k añadido para poder sortear el obstáculo móvil con el que colisiona el robot.

Una vez calculada la velocidad v_k con la que se inicia la circulación del segmento SA_i, resulta imprescindible comprobar si la relación con la velocidad v_i de fin del segmento verifica la restricción de aceleración indicada en 3.27. En caso de que no sea cierta, v_k deberá modificarse, pero si el valor modificado no asegura un tiempo de navegación t_i^A para SA_i mayor que el tiempo T, entonces el obstáculo actual no será superado. Esta dificultad se solventa modificando, de manera iterativa, las referencias de velocidad de los segmentos de camino anteriores. Esta operación se efectúa mediante la propagación hacia atrás del tiempo necesario para evitar el obstáculo, y que además mantiene la restricción de aceleración; dicho valor temporal que se propaga recibe el nombre de *tiempo restante*.

El proceso total se detalla en un nuevo algoritmo *EvitaObstáculoMóvil*, compuesto por dos partes diferenciadas. La primera inserta en el conjunto S de segmentos de camino los nuevos segmentos, calcula el valor de velocidad v_k, y por último determina el tiempo ΔT, referido a la diferencia entre el tiempo de evitación T para evitar el obstáculo OB_j y el tiempo real t_i^A para recorrer SA_i impuesto por la restricción de aceleración. La segunda parte, implantada como un bucle, se encarga de realizar, si es necesario, la propagación hacia atrás del tiempo.

EvitaObstáculoMóvil(S,V,OB_j,i)
- Selecciona S_i como el i-ésimo elemento de S
- Dividir S_i en SA_i y SB_i y actualizar S
- Calcular T
- Calcular v_k usando la expresión 4.13 y añadirla al conjunto V
- Comprobar que V cumple la restricción de aceleración 3.27
- Calcular el tiempo de navegación t_i^A para SA_i conforme a las expresiones 3.26
- $\Delta T = T - t_i^A$
- Mientras (($\Delta T > 0$) y ($i > 0$)) hacer
  - $i = i - 1$
  - Calcular vi a partir de ΔT y la expresión 4.13
  - Comprobar que V cumple la restricción de aceleración 3.27
  - Calcular el tiempo de navegación t_i para S_i según las expresiones 3.26
  - Actualizar ΔT
- Fin Mientras

Fin *EvitaObstáculoMóvil*

4.3.3.3. Algoritmo de planificación de trayectorias multirrobot

Para que resulte de utilidad, la metodología de evitación de obstáculos móviles debe integrarse con el procedimiento de planificación de velocidades expuesto en el capítulo previo. Surge así un algoritmo separado en tres etapas, correspondientes al proceso de muestreo de velocidades, al sorteo de obstáculos móviles, y finalmente a la generación de velocidades.

Así, en la primera fase se divide el camino Q en el conjunto S de segmentos de camino, y se establece el conjunto de velocidades de control V, cuyos componentes aseguran que se verifica la restricción de aceleración (3.27).

La segunda etapa se destina a la evitación de los obstáculos móviles que pululan en el entorno del vehículo. Para ello, en primer lugar se definen los conjuntos de regiones prohibidas OB y de zonas de seguridad Z en el plano espacio-tiempo. Acto seguido, el algoritmo comprueba las colisiones entre los elementos de ambos conjuntos; siempre que se detecte algún choque, los conjuntos S y V se modifican para solventar la situación.

Por último, la tercera etapa construye el perfil de velocidad $V(s)$ y lo funde con el camino Q, para así obtener la trayectoria deseada $\tilde{Q}$. Estos tres pasos puede esquematizarse en el algoritmo *ConvierteCaminoEnTrayectoria*:

ConvierteCaminoEnTrayectoria(*Q,OT*)
/* Planificación de velocidades */
$\tilde{Q}$ = {}
{S} = DivideCaminoEnSegmentos(Q)
{S} = PurgadoDeSegmentos(S)
{V} = CalculaVelocidadesDeControl(S)

/* Evitación de obstáculos móviles */
{OB} = CalculaRegionesProhibidas(S,OT)
{Z} = CalculaZonasSeguridad(S,V)
Para cada S_i perteneciente a S hacer
Para cada OB_j perteneciente a OB hacer
Si Colision(Z_i,OB_j) entonces
{V,S} = EvitaObstáculoMóvil(S,V,OB_j,i)
Fin Si
Fin Para
Fin Para

/* Generación del perfil de velocidad y construcción de la trayectoria*/
Para cada S_i perteneciente a S hacer
Calcular ${}^1\sigma_i(t)$ y ${}^2\sigma_i(t)$
${}^1\sigma_i(t) = \text{Componer}({}^1\sigma_i(t),{}^2\sigma_i(t))$
$V_i(s) = \sigma'_i(\sigma_i^{-1}(s))$
$\tilde{Q} = Q \cup \{S_i, V_i(s \in S_i)\}$
Fin Para
Fin *ConvierteCaminoEnTrayectoria*

Es necesario tener en cuenta que evitar una región prohibida OB_j puede provocar colisiones entre zonas de seguridad Z_k con {k=0,...,i-1} y regiones prohibidas OB_l con {l=0,...,j-1}. Por ello, los bucles anidados que resuelven la evitación de obstáculos móviles deben reiniciarse en el caso de que la función *EvitaObstáculoMóvil* rectifique la planificación de velocidad original.

El procedimiento *ConvierteCaminoEnTrayectoria* genera la trayectoria para un único vehículo. Para poder establecer las trayectorias asociadas a un sistema multirrobot, debe aplicarse dicho procedimiento a cada uno de los robots integrantes del mismo, considerando que el resto de vehículos se comportan como obstáculos móviles. En otras palabras, la planificación de trayectorias para un sistema multirrobot se realiza bajo un

enfoque desacoplado priorizado, según se expuso en el capítulo segundo de esta tesis, en el que la trayectoria para cada vehículo se calcula mediante la metodología de planificación de velocidades presentada en esta sección. Como colofón a todo lo expuesto en esta sección, a continuación se muestra el algoritmo *PlanificaciónTrayectoriasMultirrobot*:

PlanificaciónTrayectoriasMultirrobot(Robots)
    {R'} = DeterminaOrdenacion{Robots}
    Para cada R'_i perteneciente a R'
        {Q_i,OT} = ExtraeCaminoYObstaculos{Robots}
        ConvierteCaminoEnTrayectoria(Q_i,OT)
    Fin Mientras
Fin *PlanificaciónTrayectoriasMultirrobot*

El único factor que quedaría pendiente en el algoritmo de planificación de trayectorias multirrobot es el establecimiento del orden en el que se computaría la trayectoria para cada vehículo; este tema, y las implicaciones que lleva consigo, se trata mucho más profundamente en el capítulo próximo.

4.3.4. Comparativa entre soluciones

De entre las tres soluciones planteadas -a priori, integrada con modificación constante del tiempo, e integrada con propagación hacia atrás del tiempo-, las dos últimas presentan la ventaja de proporcionar directamente la trayectoria del robot móvil, al combinar la descomposición camino-velocidad junto con el algoritmo de planificación de velocidades de Muñoz. Mientras, la solución a priori es una herramienta para generar restricciones de velocidad de seguridad, que serán tenidas en cuenta posteriormente, de alguna manera, para obtener el perfil de velocidad definitivo.

Respecto a las dos soluciones integradas, resulta más precisa la planificación con propagación del tiempo, puesto que se ajusta exactamente a la situación del plano *s x t* para sortear los obstáculos móviles. Por contra, la metodología integrada basada en tiempo constante, al utilizar incrementos de tiempo constantes, puede generar perfiles de velocidad no tan ajustados a la realidad, aparte de necesitar la estimación del parámetro PASO.

4.4. Experimentación y Resultados

A continuación, se van a mostrar y comentar los resultados experimentales obtenidos de la aplicación del algoritmo planificador de velocidades. En el desarrollo de las tales pruebas ha resultado muy útil la aplicación de un pequeño interfaz gráfico desarrollado sobre Matlab, que facilita en gran medida la ejecución del cálculo de trayectorias, así como la recogida de los datos obtenidos. Por ello, antes de comenzar con interpretación de los experimentos propiamente dichos, se va a dedicar una subsección a una descripción más completa del mismo.

4.4.1. Interfaz Gráfico sobre Matlab

El interfaz gráfico creado presenta la siguiente apariencia inicial:

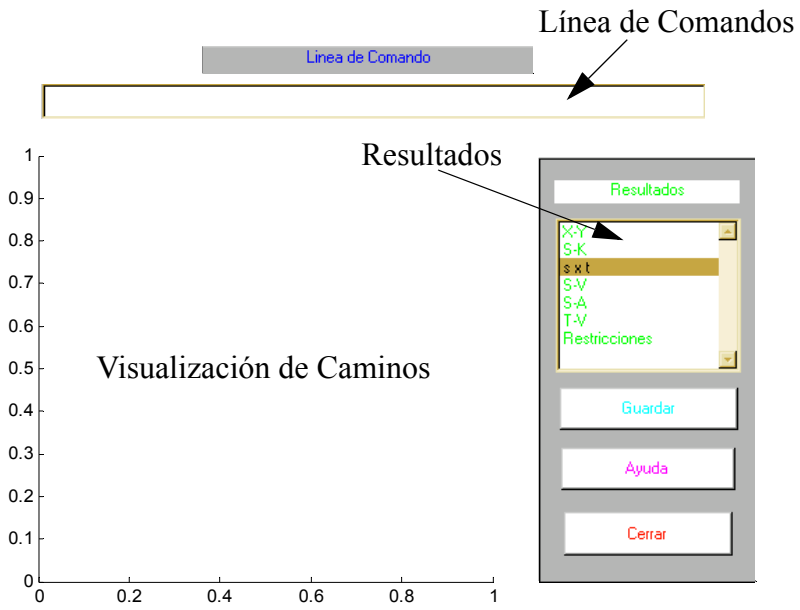


Figura 4.7. Interfaz Gráfico del Planificador de Velocidades: ventana inicial

En esta ventana inicial se distinguen tres zonas claramente diferenciadas: la *Línea de Comandos*, el área de *Visualización de Caminos*, y el área de *Resultados*; en este mismo orden se va a explicar su funcionalidad.

Para poder trabajar con esta herramienta, es necesario que el primer paso consista en la introducción, en la línea de comandos, de los datos del sistema multirrobot cuyas trayectorias se desean calcular, con el siguiente formato:

camino_robot_actual {tamaño_robot_anterior_i trayectoria_robot_anterior_i} (4.14)

El parámetro *camino_robot_actual* es obligatorio, y concuerda con el camino geométrico del vehículo para el que se está calculando la trayectoria. El siguiente parámetro, cuya multiplicidad es cero o más, corresponde en realidad, a pares de valores relativos a los robots que preceden al actual en el sistema; por un parte, se incluye su tamaño, expresado como el diámetro del círculo en el que se inscribiría el obstáculo móvil; por otra parte, la trayectoria que sigue, entendiendo como trayectoria la unión del camino espacial y el perfil temporal asociado al mismo. Tanto los caminos como las trayectorias se suponen almacenados en ficheros con un formato matricial, en el que las columnas indican las componentes de posición, orientación, curvatura, distancia, velocidad, aceleración, y tiempo de las diferentes posturas del camino, representadas en las filas de dicha matriz.

La zona de visualización de caminos muestra, sobre un plano del entorno que puede modificarse convenientemente, los caminos geométricos de cada uno de los elementos que intervienen en el cálculo de las trayectorias para el robot actual; cada vehículo se representa con un código de color distinto.

Por último, el área de Resultados ofrece cuatro posibilidades:

- seleccionar la información que se desea contemplar. En efecto, una vez calculada la trayectoria para el vehículo actual, puede visualizarse gráficamente en una ventana separada, de siete maneras diferentes:

  - Coordenadas X del camino frente a coordenadas Y del camino (X-Y).

  - Espacio recorrido frente a curvatura del camino (S-K).

  - Plano *s x t* (s x t).

  - Espacio recorrido frente a perfil de velocidad (S-V).

  - Espacio recorrido frente a perfil de aceleración (S-A).

  - Tiempo transcurrido frente a perfil de velocidad (T-V).

- Restricciones de velocidad: el perfil de velocidad obtenido se representa, en el plano espacio-velocidad, junto con las principales limitaciones de velocidad que afectan al vehículo.

Todas estas representaciones indican, además, los segmentos en los que se divide el camino para poder establecer el perfil temporal según el algoritmo de Muñoz.

- guardar la nueva trayectoria, en el mismo formato matricial de los ficheros de entrada.
- obtener información de ayuda para el manejo del interfaz gráfico.
- cerrar la ventana del interfaz gráfico.

Para completar esta descripción del interfaz gráfico desarrollado, la Figura 4.8 muestra un ejemplo de utilización del interfaz gráfico. Se observa que en la línea de comando se dan las instrucciones para calcular la trayectoria para el vehículo con camino denominado *trasun10.tra*, en presencia del robot con trayectoria *trasun8.tra*, cuyas dimensiones se inscriben en un círculo de diámetro igual a un metro.

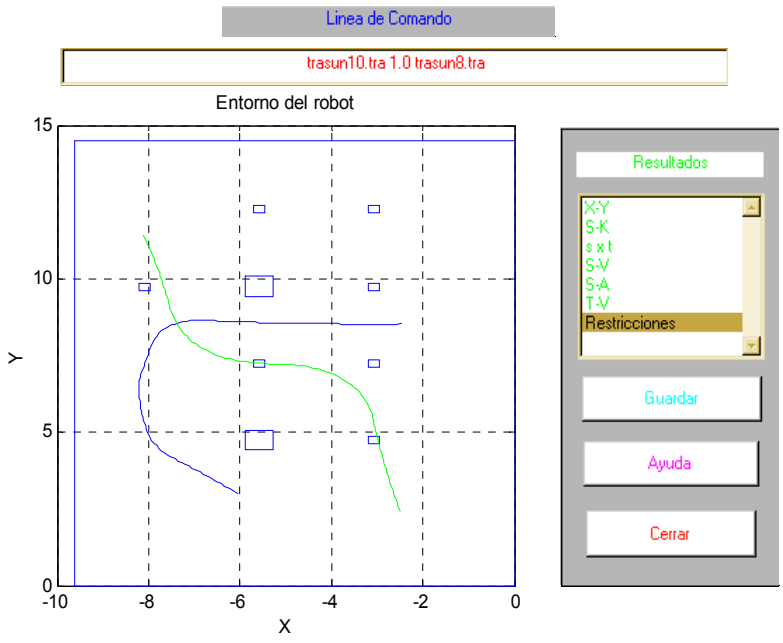


Figura 4.8. Ejemplo de utilización del interfaz gráfico.

Toda la gama de resultados posibles se resume en la tabla 4.1

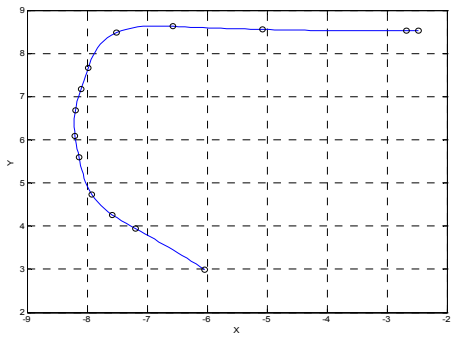
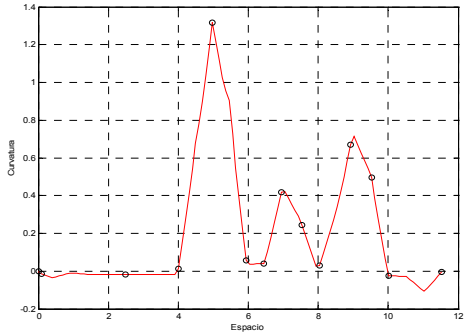
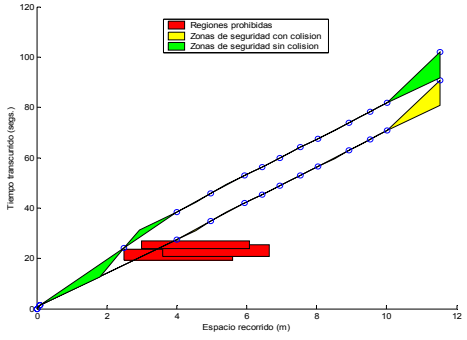
Tipo de Resultado	Representación Gráfica
X-Y	
S-K	
s x t	

Tabla 4.1: Resultados ofrecidos por el interfaz gráfico

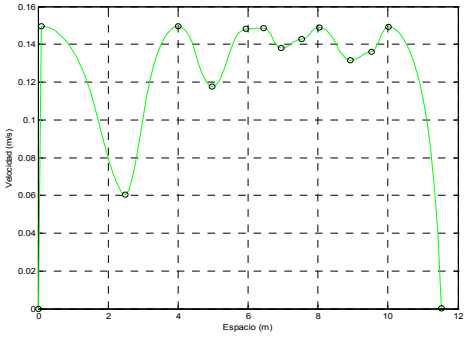
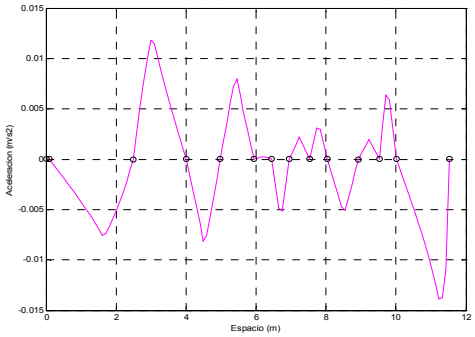
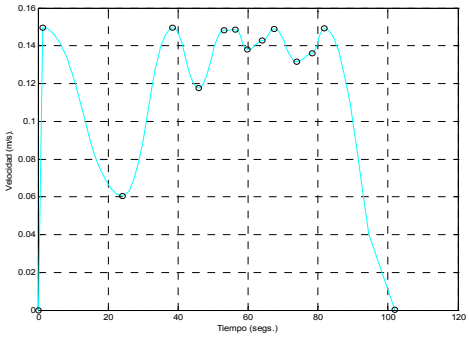
Tipo de Resultado	Representación Gráfica
S-V	
S-A	
T-V	

Tabla 4.1: Resultados ofrecidos por el interfaz gráfico

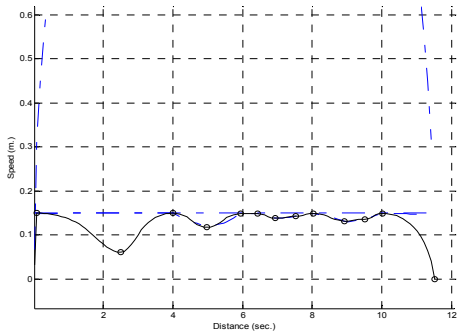
Tipo de Resultado	Representación Gráfica
Restricciones de Velocidad	

Tabla 4.1: Resultados ofrecidos por el interfaz gráfico

4.4.2. Pruebas experimentales

Las pruebas experimentales se han realizado sobre el robot Auriga-α (Figura 4.9), diseñado y construido íntegramente en el Departamento de Ingeniería de Sistemas y Automática de la Universidad de Málaga, sobre el que se ha implantado una arquitectura robótica mediante el software de desarrollo de aplicaciones robóticas distribuidas NEXUS, labor también del mismo departamento. Tanto el robot como NEXUS se describen con mayor detalle en los Apéndices A y C, respectivamente.

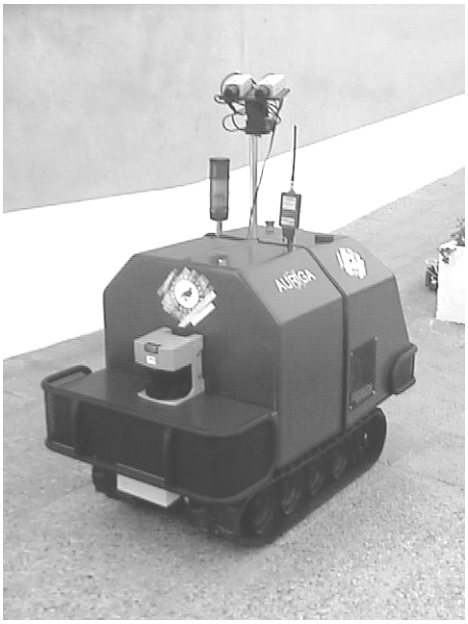


Figura 4.9. Robot autónomo Auriga-α

El entorno en el que se han desarrollado las pruebas se muestra en la Figura 4.10. Como se observa, se trata de un espacio rectangular, en cuya zona central existen dos columnas. El camino geométrico del robot móvil Auriga-α, así como los de los vehículos móviles que interactúan con él en este ambiente, deben sortear tales obstáculos estáticos.

Figura 4.10. Entorno real de experimentación

Los resultados de experimentación se han organizado como sigue. El primer experimento consiste en un recorrido del Auriga-α que es en gran parte rectilíneo, sorteando un par de obstáculos móviles. En las dos siguientes pruebas, sin embargo, el camino geométrico del Auriga-α se complica, formando un "ocho" alrededor de las dos columnas centrales del espacio de trabajo; la diferencia entre ambas radica en la velocidad tope con la que Auriga-α sigue su camino.

4.4.2.1. Prueba experimental nº 1

En la primera prueba intervienen dos obstáculos móviles que interfieren en el camino de Auriga-α, según se muestra en la Figura 4.11:

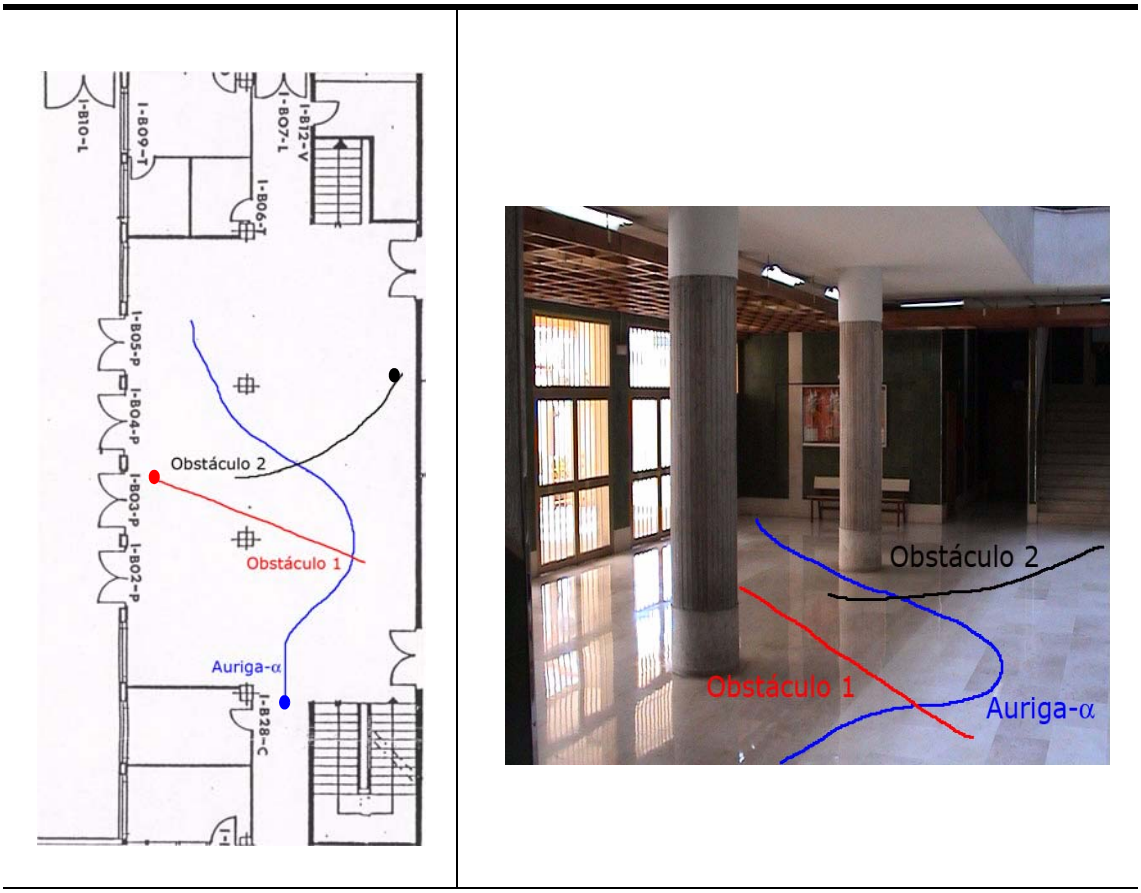


Figura 4.11. Caminos geométricos de Auriga-α y los obstáculos móviles para el Experimento 1

La velocidad punta considerada para el vehículo en este caso es de 0.2 m/s, que se alcanzaría rápidamente si Auriga-α se encontrase en un entorno sin obstáculos. En la Figura 4.12 se representa en trazo azul continuo el perfil de velocidad calculado para Auriga-α, y en trazo rojo discontinuo el perfil de velocidad seguido:

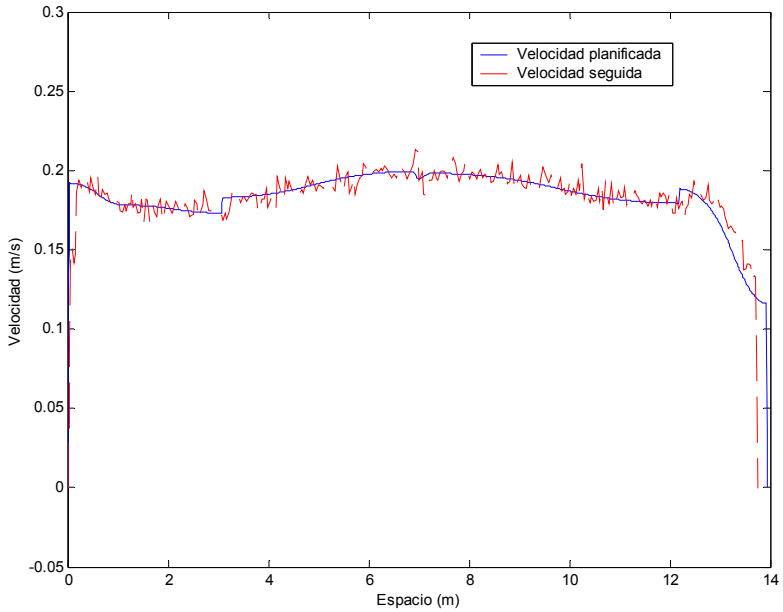


Figura 4.12. Velocidad planificada y seguida para Auriga-α en ausencia de obstáculos móviles

El estudio del plano *sxt* lleva a la situación indicada en la Figura 4.13:

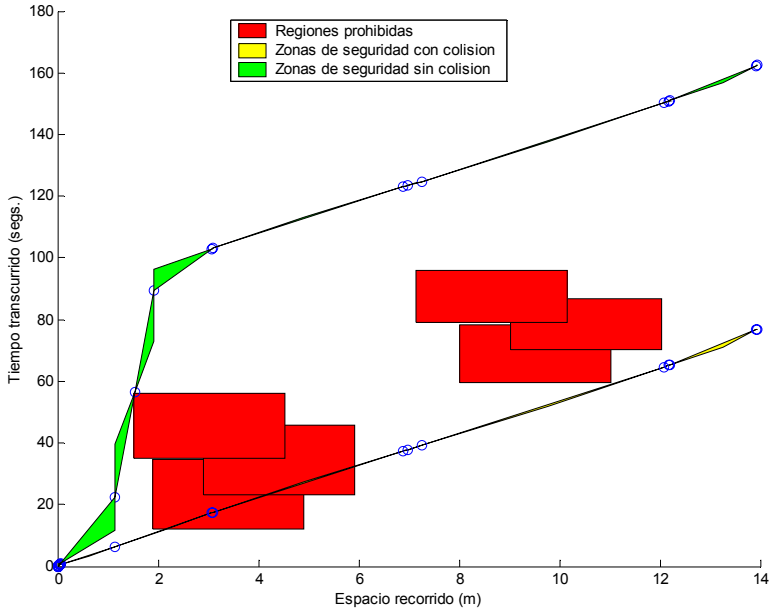


Figura 4.13. Plano espacio-temporal asociado al Experimento 1

Como se observa, existen dos áreas conflictivas, correspondientes a los dos únicos cruces geométricos entre el camino del Auriga-α y las trayectorias de los obstáculos móviles. En concreto, una de las regiones prohibidas perteneciente a la primera zona problemática intersecta con la planificación inicial de velocidad (la máxima posible) calculada para el robot, que se representa mediante las zonas de seguridad amarillas. Por ello, es necesario modificar este perfil de tiempos original para eludir la colisión y garantizar una navegación segura; este nuevo plan de velocidades, más lento, se representa en la gráfica mediante las zonas de seguridad verdes. Dicha replanificación de velocidades se representa, frente al espacio y al tiempo, en las Figuras 4.14 y 4.15:

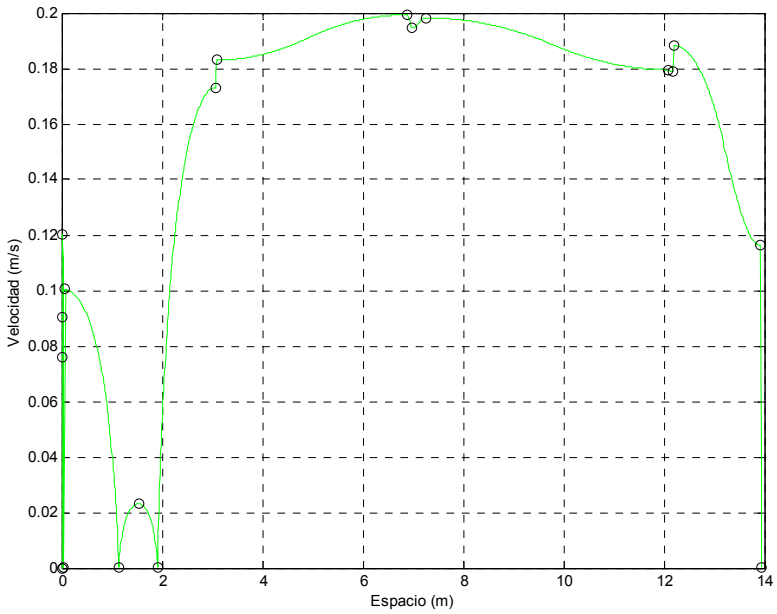


Figura 4.14. Velocidad replanificada frente a espacio recorrido

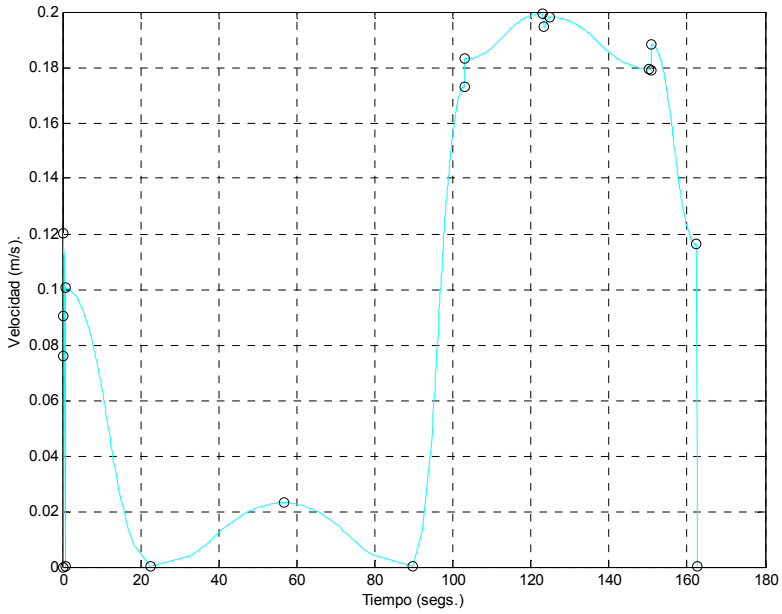


Figura 4.15. Velocidad replanificada frente a tiempo de navegación

Y, cuando el vehículo real sigue dicha planificación, se obtiene la siguiente ejecución, cuya precisión como puede observarse sigue siendo muy ajustada:

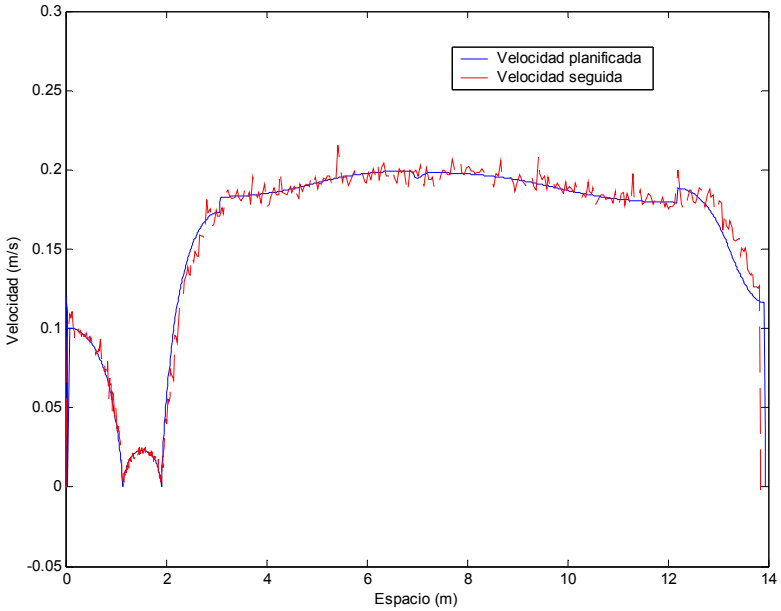


Figura 4.16. Velocidad replanificada y velocidad seguida

4.4.2.2. Prueba experimental nº 2

La segunda prueba consiste en recorrer un camino que transita entre las columnas del entorno de trabajo, en presencia de dos obstáculos móviles, considerando que la velocidad tope que puede desarrollar Auriga-α es de 0.3 m. Gráficamente, la situación se resume en la siguiente Figura 4.17:

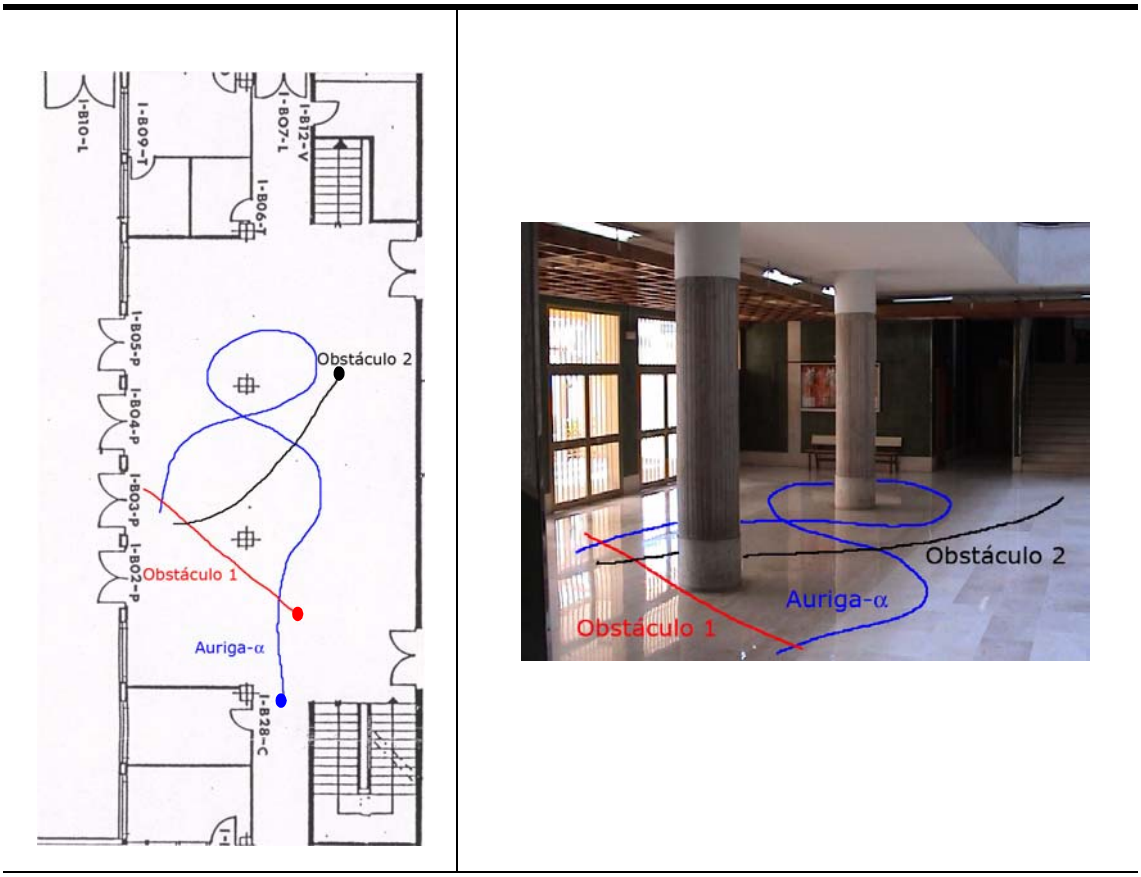


Figura 4.17. Caminos geométricos de Auriga-α y los obstáculos móviles para el Experimento 2

De manera que si Auriga-α pudiera recorrer su camino sin la interacción de los vehículos móviles, el perfil de velocidad aplicado y seguido sería:

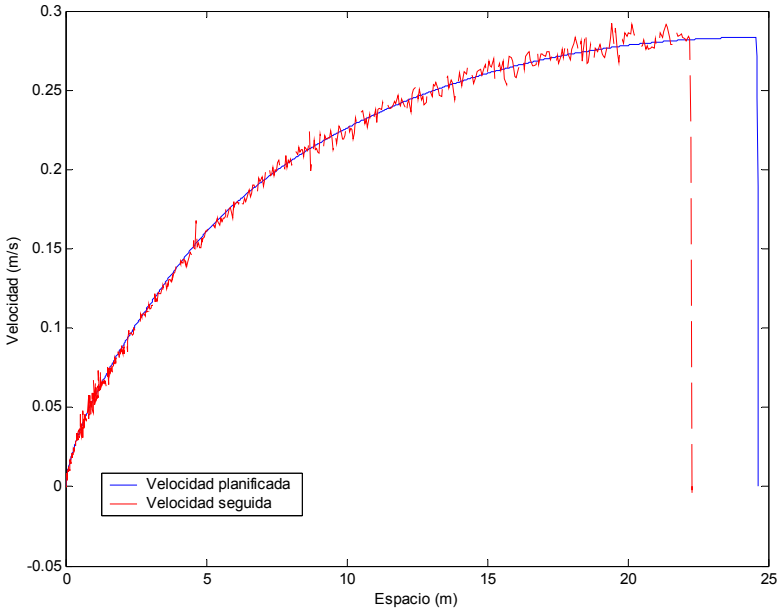


Figura 4.18. Velocidad planificada y seguida para Auriga-α en ausencia de obstáculos móviles

Sin embargo, como ésta no es la situación real de trabajo, será necesario evaluar cómo influye la presencia de los obstáculos móviles sobre esta planificación de velocidades, y para ello se estudia el plano *s x t*:

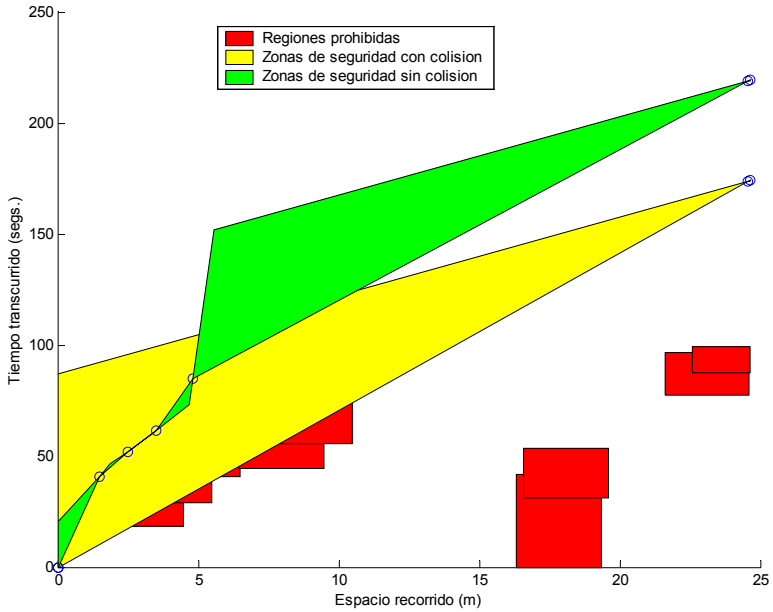


Figura 4.19. Plano espacio-temporal asociado al Experimento 2

En este caso, la planificación inicial (zonas de seguridad amarillas) resulta conflictiva y provocaría una colisión con los obstáculos móviles. Evidentemente, es necesaria una replanificación de velocidades que elimine cualquier probabilidad de choque, representada por las zonas de seguridad verdes. Para mayor claridad, las siguientes Figuras (4.20 y 4.21) muestran, en primer lugar, las regiones prohibidas junto con las zonas de seguridad de la replanificación; en segundo lugar, un detalle aumentado de la evitación por parte de las mismas de las áreas conflictivas:

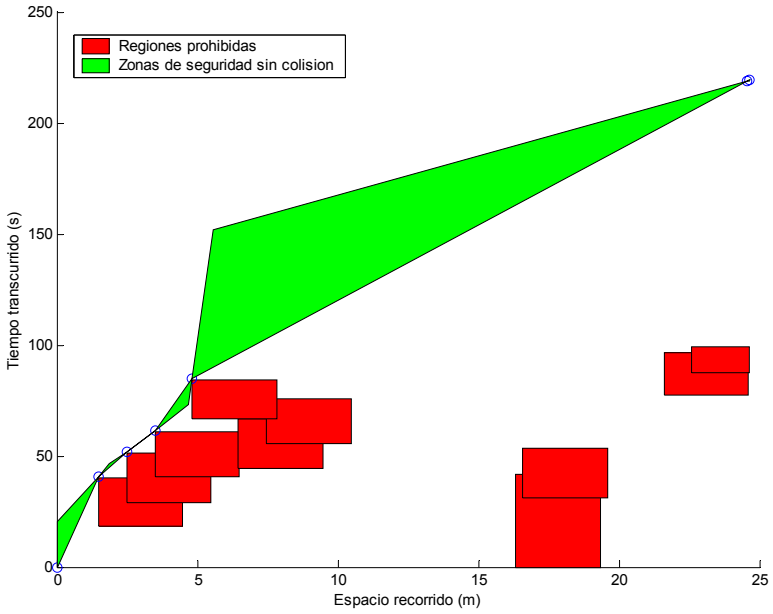


Figura 4.20. Zonas de seguridad que evitan la colisión en el plano *sxt*

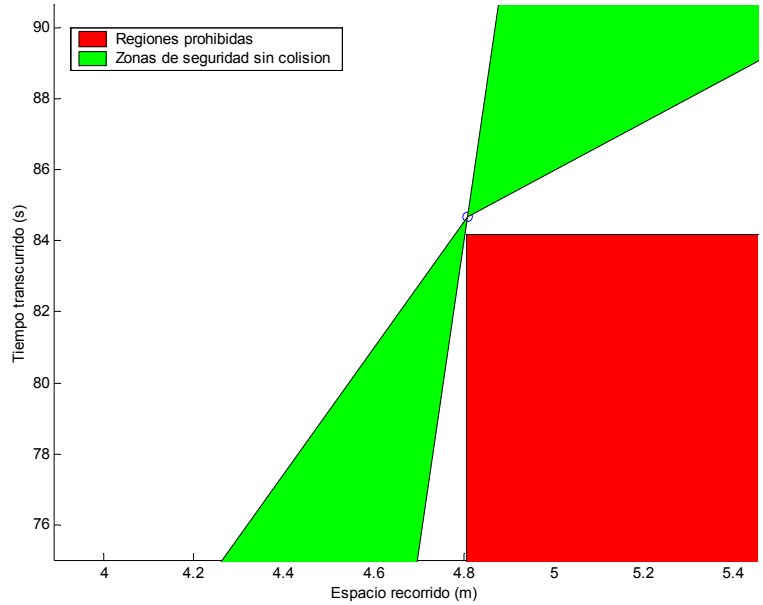


Figura 4.21. Detalle de la zona de seguridad que evita la colisión en el plano *sxt*

El nuevo perfil de velocidades tendría la siguiente forma:

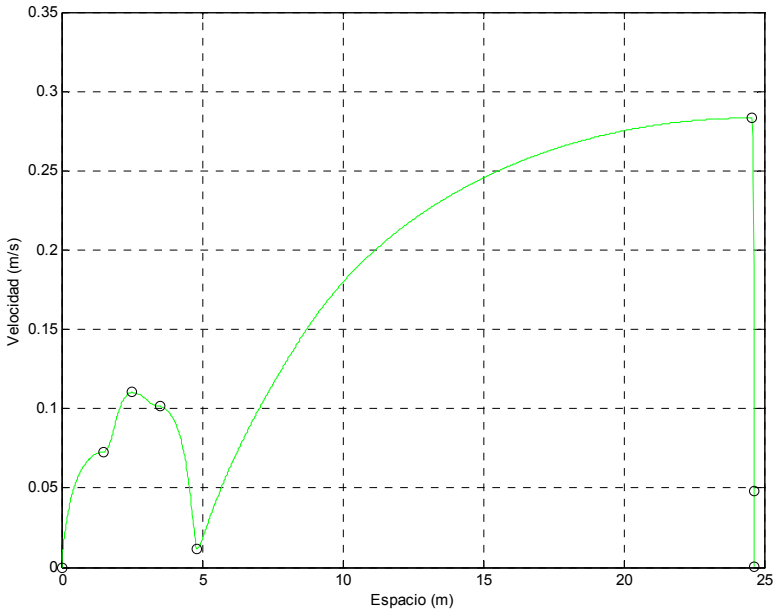


Figura 4.22. Velocidad replanificada frente a espacio recorrido

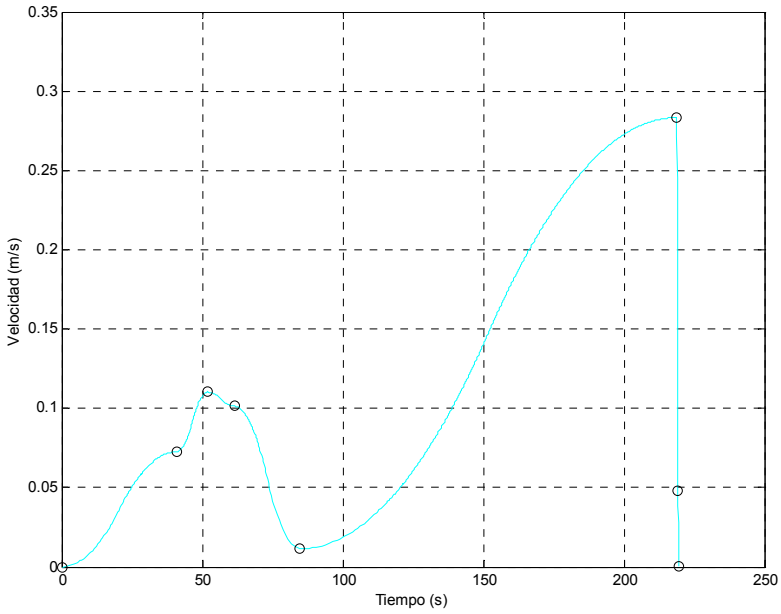


Figura 4.23. Velocidad replanificada frente a tiempo de navegación

cuyo seguimiento, por parte del vehículo real, es el siguiente:

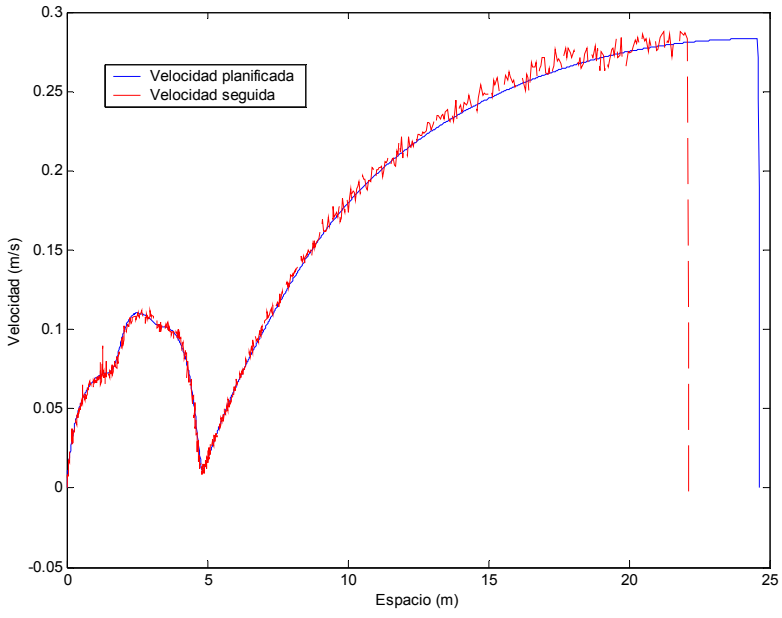


Figura 4.24. Velocidad replanificada y velocidad seguida

4.4.2.3. Prueba experimental nº 3

En esta última prueba, Auriga-α debe efectuar el mismo camino que en el apartado anterior, mientras que los obstáculos móviles han variado su posición ligeramente. La Figura 4.25 muestra esta nueva situación, tanto tanto de manera esquemática como sobre el entorno de trabajo.

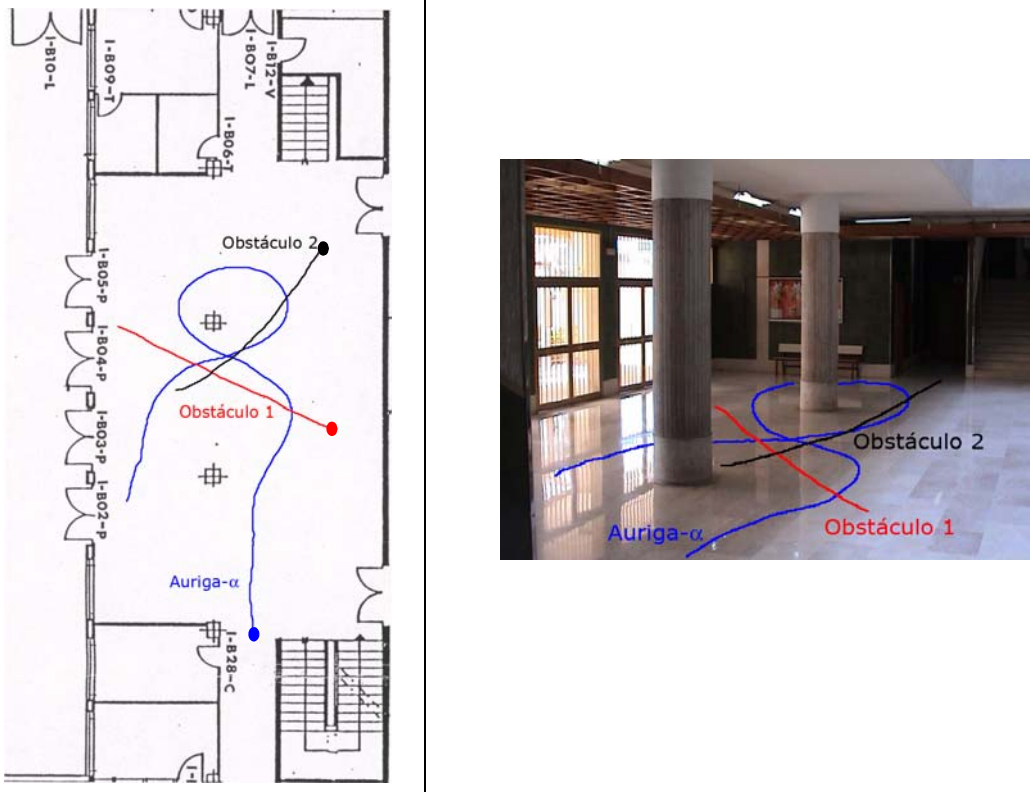


Figura 4.25. Caminos geométricos de Auriga-α y los obstáculos móviles para el Experimento 3

El trayecto del robot Auriga-α se muestra en la siguiente secuencia de imágenes:

Figura 4.26. Recorrido real efectuado por Auriga-α

Para esta situación experimental, y en ausencia de obstáculos móviles, el vehículo puede alcanzar su velocidad punta (en este caso, 0.15 m/s) en la mayor parte del trayecto. La Figura 4.27 muestra, como siempre, en trazo azul continuo el perfil de velocidad calculado para Auriga-α, y en trazo rojo discontinuo el perfil de velocidad que se sigue realmente.

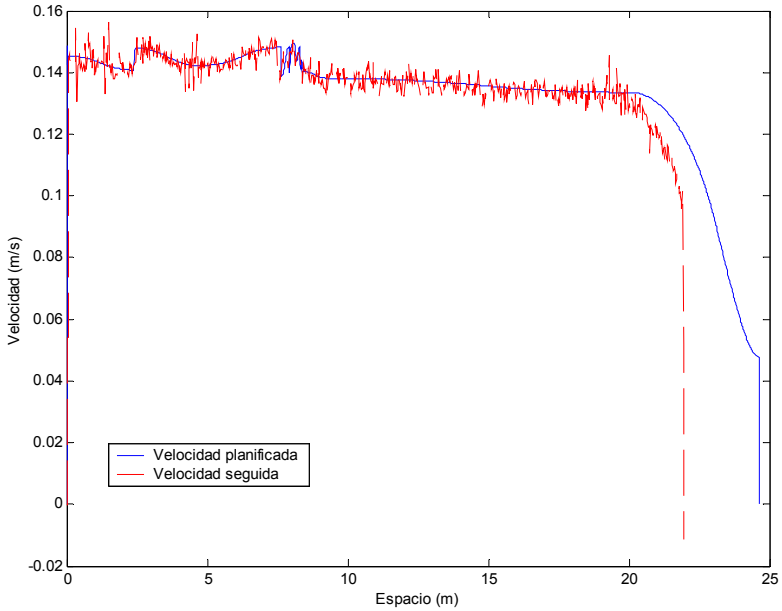


Figura 4.27. Velocidad planificada y seguida para Auriga-α en ausencia de obstáculos móviles

Puede observarse que, aunque en líneas generales el perfil de velocidad real se ajusta con bastante precisión al perfil de velocidad planificado, en el último tramo del camino este hecho deja de producirse. La razón estriba en que el algoritmo de seguimiento de caminos implantado sobre el vehículo (concretamente, el método *pure-pursuit*), entiende que se alcanza el objetivo final del trayecto geométrico antes de lo que realmente le corresponde.

Sin embargo, cuando se tiene en cuenta la presencia de los dos obstáculos móviles, aparecen cinco cruces geométricos entre los obstáculos y el camino del Auriga-α, que podrán dar lugar a situaciones de riesgo si éste último no los recorre con la velocidad apropiada. Dichos cruces entre Auriga-α y los obstáculos se traducen en dos conjuntos de regiones prohibidas. El primero de ellos se produce cuando el Auriga-α ha recorrido una distancia de entre 5 y 10 metros, y el tiempo transcurrido pasa de 0 a 90 segundos aproximadamente; el segundo surge cuando el vehículo ha avanzado un poco más, entre 15 y 25 metros, en un intervalo de tiempo un poco más reducido. La relación entre los caminos/trayectorias de robot móvil y obstáculos, y las regiones prohibidas generadas se indican en la Figura 4.28: el primer conjunto se corresponde con la línea magenta de la imagen, mientras que el segundo conjunto se representa en la misma con una línea verde.

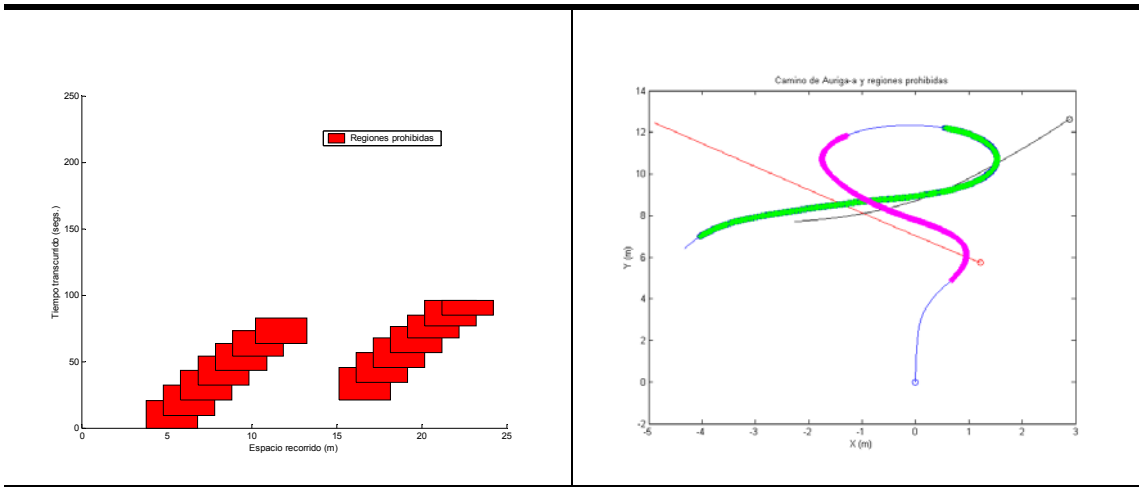


Figura 4.28. Regiones prohibidas en el plano *sxt* y su localización sobre el entorno real

Ahora sí, se analizan en el plano *s x t* las posibles colisiones entre las regiones prohibidas originadas por los cruces espacio-temporales y las zonas de seguridad correspondientes al perfil de velocidad inicialmente asignado a Auriga-α. Dicho análisis se muestra en la Figura 4.29:

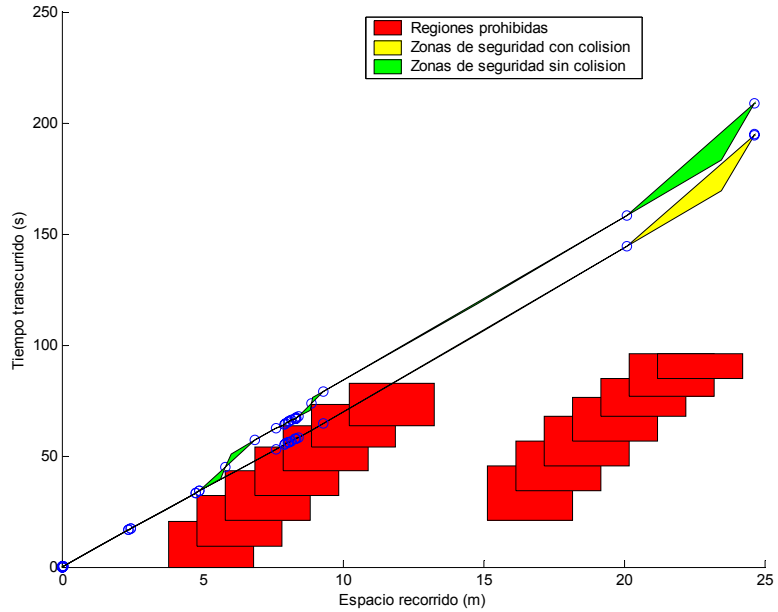


Figura 4.29. Plano espacio-temporal asociado al Experimento 3

Como se observa, existe una colisión entre las zonas de seguridad y las regiones prohibidas, lo que fuerza a la creación de un nuevo segmento que sortee la región prohibida, elevando las zonas de seguridad. El nuevo perfil de velocidad, que sí garantiza la ausencia de colisiones, se representa en las siguiente Figuras frente al espacio y frente al tiempo, respectivamente:

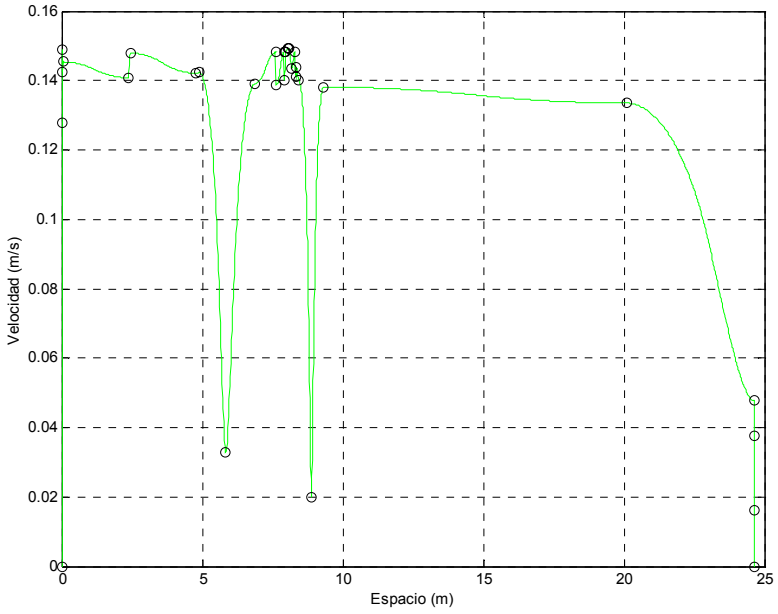


Figura 4.30. Velocidad replanificada frente a espacio recorrido

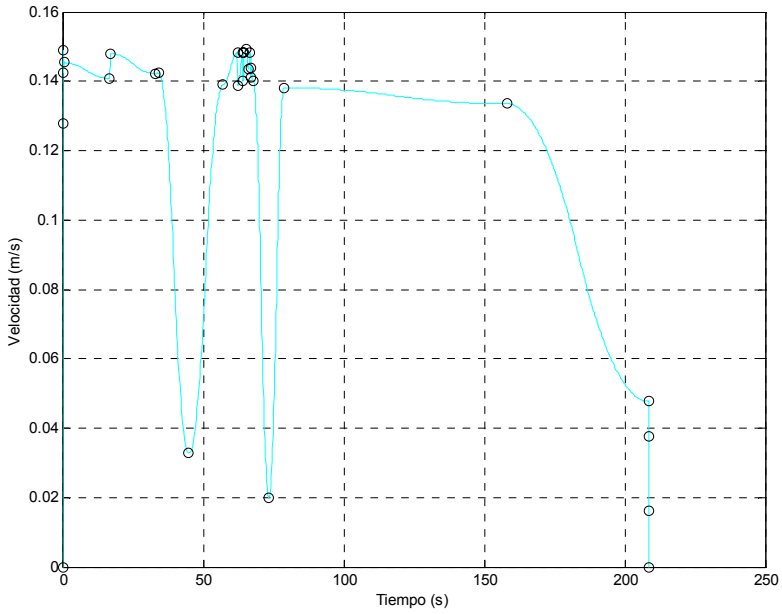


Figura 4.31. Velocidad replanificada frente a tiempo de navegación

La ejecución, por parte del vehículo, de la replanificación de velocidades que evita las colisiones con los obstáculos móviles, se realiza de nuevo satisfactoriamente, según indica la Figura 4.32:

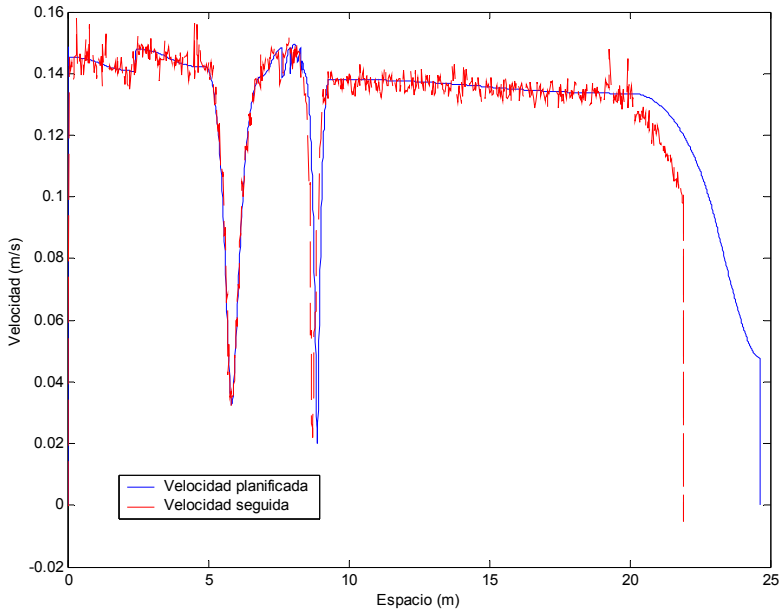


Figura 4.32. Velocidad replanificada y velocidad seguida

4.5 Efectos de la replanificación de velocidades

Hasta el momento, el trabajo presentado se ha centrado en dos aspectos centrales: el desarrollo de un algoritmo de planificación de velocidades para robots en presencia de obstáculos móviles, y la aplicación del mismo a sistemas multirrobot integrados por varios vehículos. Ambos casos se han planteado bajo el paradigma de navegación deliberativa que, como se mencionó en el capítulo de introducción, utiliza información conocida a priori para elaborar una planificación completa del movimiento a realizar por el robot.

Sin embargo, la limitación de la navegación estratégica es clara: el plan trazado originalmente puede verse abortado ante situaciones inesperadas, de las que se carecía de información a priori. Por tanto, para que la navegación planificada resulte de utilidad en la práctica, es necesario que provea de técnicas para facilitar la adaptación ante aquellos imprevistos, que aunque no se produzcan con frecuencia, pueden surgir durante el desempeño de la tarea del vehículo.

Por tanto, para que la planificación de trayectorias deliberativa para sistemas multirrobot presentada en esta tesis sea lo más efectiva posible, debería integrarse en un arquitectura robótica reactiva o híbrida, que le permita reaccionar ante acontecimientos de los que, a priori, carece de información y, por tanto, no contempla en su plan de movimientos. La arquitectura elegida ha sido la implantada en el robot Auriga-α, diseñada y desarrollada completamente en el Dpto. de Ingeniería de Sistemas y Automática de la Universidad de Málaga. Se trata de una arquitectura híbrida y modular, dividida en tres niveles que abarcan tareas de planificación, coordinación y ejecución de misiones. Para su construcción se ha empleado NEXUS, un sistema de integración de software robótico desarrollado también en el mismo Departamento (Fernández Madrigal y González, 1999). En el Apéndice C se explican con mayor detalle tanto la arquitectura en sí, como la implantación del mecanismo de planificación de velocidades multirrobot.

El hecho de que, gracias a las posibilidades de reacción que ofrece la arquitectura híbrida de Auriga-α, se alteren los perfiles temporales generados en un principio para cada vehículo da lugar a unos efectos laterales que pueden tener importantes consecuencias sobre el funcionamiento previsto del sistema multirrobot. A continuación se analizan con mayor detalle tales efectos.

Los métodos que alteran la planificación inicial para ceñirse a los nuevos requerimientos se denominan *técnicas de replanificación*. La replanificación de una trayectoria inicial puede efectuarse tanto en el espacio como en el tiempo; en el primer caso, se altera el camino geométrico que iba a recorrer el robot, mientras que en el segundo se reforma el perfil de velocidad con que lo seguiría. Como el tema central de esta tesis se refiere a la planificación temporal, la replanificación que aquí se propone será una replanificación de velocidades. Por otra parte, en muchas ocasiones, al término "replanificación" se le asocia el adjetivo "local", para indicar que el ámbito de la corrección afecta únicamente a las cercanías del punto en el que se encuentra el robot. Sin embargo, cuando se modifica la velocidad a la que el vehículo se mueve, el efecto de este cambio perdura a lo largo del tiempo, más allá del entorno próximo del vehículo. Por tanto, se considera que la replanificación aquí expuesta no se trata de una replanificación local en el sentido de la expresión anteriormente citado.

Cuando se replanifica el perfil temporal asociado a alguno de los vehículos de un sistema multirrobot, la disminución de velocidad calculada para que éste pueda eludir una situación de riesgo implica también que todas las zonas conflictivas posteriores a la misma en las que el vehículo se vea involucrado ya no se producen bajo las mismas condiciones iniciales; por tanto, no hay garantía de que los choques que el perfil de velocidad original evitaba ahora no aparezcan. Es decir, la replanificación local puede influir decisivamente en el ulterior comportamiento de los vehículos del sistema.

Existen dos soluciones opuestas para remediar esta situación. Una de ellas consiste en que, una vez detectado el peligro de colisión entre vehículos, se detengan todos los robots del sistema y se efectúe un nuevo cálculo de las trayectorias globales para cada uno de los robots a partir de ese punto. La otra posibilidad dejaría totalmente en manos de la replanificación local el funcionamiento del sistema multirrobot, de forma que, en caso de que suceda algún problema imprevisto -incluyendo los debidos a replanificaciones locales previas-, se modifiquen los perfiles de velocidad de los vehículos afectados mientras que el resto del sistema sigue su curso. La primera solución es más segura, pero tiene la desventaja de ser más costosa temporalmente, puesto que supone reproducir todo el proceso de generación de trayectorias, aunque sea sobre caminos más cortos; además, necesita de algún mecanismo de comunicación para avisar de la situación de parada y replanificación a todos los vehículos. Frente a ella, la segunda opción ahorra tiempo de cómputo, no requiere ningún tipo de comunicación interna o externa, y dota de mayor autonomía al sistema,

siempre que no haya que replanificar localmente un número excesivo de veces. Entre ambas soluciones extremas, existirían mecanismos intermedios para solventar la replanificación, como puede ser el cálculo del nuevo plan de velocidades sólo para los vehículos que se vean afectados por algún acontecimiento potencialmente peligroso no considerado inicialmente.

La actuación más deseable, en caso de que acontezca alguna situación inesperada que obligue a recalcular el perfil temporal de uno o varios elementos del sistema multirrobot, pasaría por no realizar ningún tipo de acción correctora sobre las trayectorias originales de los vehículos, de manera que sea el propio sistema el que evolucione y replanifique de nuevo, si fuera necesario. La posibilidad de volver a generar todas las trayectorias de los robots resulta más costosa, aunque no quedaría descartada para estructuras multirrobots que sí empleen comunicaciones entre vehículos o con elementos externos.

4.6 Conclusiones

Este capítulo se centra en el estudio de una metodología de planificación de trayectorias para robots móviles aplicable a los componentes de un sistema multirrobot. Tomando como punto de partida un algoritmo de planificación de velocidades monorrobot, se ha incorporado el planteamiento de la descomposición camino-velocidad, de manera que en el cálculo de la trayectoria de un vehículo, los restantes elementos del sistema pueden ser tratados como obstáculos móviles, permitiendo así la generación de un perfil de velocidad seguro.

La unión de ambos enfoques da lugar a dos interpretaciones diferentes, que se traducen en dos metodologías diferentes: a priori e integrada. Por una parte, la metodología a priori emplea la descomposición camino-velocidad como herramienta para la construcción de restricciones de velocidad, que serán tenidas en cuenta por el algoritmo de planificación de velocidades de Muñoz. Bajo la técnica integrada, sin embargo, la planificación temporal se realiza a la par que se analizan las posibles colisiones, todo ello en el plano espacio-tiempo asociado a la sintonía de velocidad. Para esta tarea se crean unas estructuras auxiliares, bautizadas como zonas de seguridad, que representan al perfil de velocidades, pero sin exigir su cálculo completo. En el caso de que se produzcan situaciones de riesgo, se modifica la situación de las zonas de seguridad para que sorteen las regiones prohibidas, de manera que las velocidades generadas garantizan la ausencia de choques con los vehículos.

A partir de aquí, se determina la técnica de planificación de trayectorias multirrobot, que sigue el paradigma de la planificación desacoplada priorizada.

Por último, se valoran las consecuencias que sobre la política de navegación planteada tiene el hecho de considerar extensiones que suplan las carencias intrínsecas a las estrategias deliberativas. Esto origina modificaciones en las condiciones sobre las que se realizó la planificación del perfil temporal, que es necesario tener en cuenta; por ello, se examinan y evalúan diferentes soluciones para esta cuestión. Así, para garantizar la utilidad de la planificación de trayectorias multirrobot planteada, debe incluirse en algún tipo de arquitectura que permita reaccionar en caso de situaciones no esperadas, como es el caso de la arquitectura desarrollada e implantada sobre el robot Auriga-α.

CAPÍTULO 5. Planificación Óptima de Trayectorias Multirrobot

5.1. Introducción

Dado un sistema multirrobot integrado por una serie de robots móviles, se desea hallar la secuencia en la que se calculan sus trayectorias aplicando el algoritmo de planificación de velocidades presentado en el capítulo cuarto, para de este modo lograr la optimización de algún criterio fijado. Dicha secuencia debe ser acorde con la planificación desacoplada priorizada, cuyo enfoque, así como los principales trabajos desarrollados bajo tal paradigma, fueron descritos en la sección 2.4.1. Como se recordará, con este planteamiento se calculan las trayectorias de los robots de manera independiente siguiendo unas prioridades establecidas, de forma que los vehículos cuya trayectoria se obtiene primero pasan a comportarse como obstáculos móviles para los robots que les siguen en la ordenación. Es decir, para el vehículo *i-ésimo* de la ordenación se computa la trayectoria considerando que los *i-1* vehículos que le preceden en el esquema de prioridades son obstáculos móviles. Los casos extremos corresponden al primer y al último vehículos de la ordenación. El primero de los robots de cualquier ordenación puede moverse a la máxima velocidad que le permitan las restricciones de velocidad a las que se encuentra sujeto, puesto que no existen obstáculos móviles que afecten su movimiento; sin embargo, el último robot debe considerar, además de sus propias limitaciones de velocidad, la influencia que sobre él tienen todos los robots que le anteceden en la ordenación y para los que previamente se han calculado las trayectorias, por lo que su recorrido puede verse enlentecido dependiendo del número de choques con sus predecesores que deba sortear.

Los criterios bajo los cuales se persigue la optimización pueden ser muy variados. A continuación se describen algunos de ellos, bastante generales, aunque serían posibles otros más adaptados a situaciones concretas:

- minimización del tiempo total empleado por los vehículos para ejecutar sus respectivas trayectorias.
- consumo energético de los vehículos.
- desgaste producido por la realización de maniobras bruscas.
- favorecimiento de ciertos vehículos: los más rápidos, los más grandes,...

Sin pérdida de generalidad, el criterio cuya minimización se ha utilizado a lo largo de este trabajo es el tiempo de navegación de los robots que forman el sistema. Además, el esquema de prioridades óptimo debe evitar los bloqueos físicos entre vehículos, uno de los problemas de los que adolece la descomposición camino-velocidad comentados en apartados anteriores.

Como se verá más adelante, la búsqueda de la solución óptima es un problema de optimización combinatoria NP-completo, por lo que no existe un algoritmo que en tiempo polinomial sea capaz de proporcionar la solución deseada. Para estos casos resulta útil el uso de técnicas heurísticas, que aunque no den con el óptimo del problema, sí pueden hallar una solución buena (óptima o *cuasi-óptima*) empleando una cantidad de recursos aceptable. Dentro de todas las técnicas heurísticas existentes, los algoritmos genéticos se han revelado como la herramienta que mejor se ajusta al tipo de problema estudiado. Así, el planificador de trayectorias multirrobot incorpora un algoritmo genético encargado de establecer el orden de prioridades en el que se aplicará la metodología de planificación de velocidades propuesta en apartados anteriores.

Este capítulo se estructura en las siguientes secciones. La sección 5.2 propone, a lo largo de sus diversos apartados, la identificación del planteamiento desacoplado priorizado como un problema de optimización combinatoria, lo que permite adjudicarle ciertas características relativas a su complejidad y a otras cuestiones clave. En la sección 5.3 se describen, de manera concisa, las principales técnicas heurísticas disponibles en la actualidad, y se justifica la elección de los algoritmos genéticos como método para establecer el orden de prioridades buscado. A continuación, el apartado 5.4 se refiere a los detalles de implantación del algoritmo genético diseñado. Los resultados experimentales obtenidos tras su aplicación se resumen en la sección 5.5. El capítulo se remata con las conclusiones que pueden extraerse de todo lo que en él se ha desarrollado. Por último, hay que señalar que este trabajo puede encontrarse, más resumidamente debido a las limitaciones de espacio propias de la publicación, en (Cruz-Martín et al., 2004).

5.2. Planteamiento del problema como optimización combinatoria

El planteamiento desacoplado priorizado puede incluirse dentro de la optimización combinatoria -véase (Papadimitriou y Steigliz, 1998) para profundizar en este tema-. Esta clase de problemas, que trabaja con variables discretas, busca la solución óptima dentro de un espacio de búsqueda finito o infinito numerable (normalmente, un entero, conjunto, permutación o grafo). Algunos problemas típicos de optimización combinatoria son el

problema de la asignación, el problema del camino más corto (*shortest path problem*, *SPP*), el problema del mínimo árbol de expansión (*minimum spanning tree*, *MST*), la programación lineal y la programación entera (*linear programming*, *LP*; *integer programming*, *IP*), el problema de la mochila (*knapsack problem*, *KP*) y el problema del viajante (*traveling salesman problem*, *TSP*).

Otro de estos problemas de optimización combinatoria es el problema del camino hamiltoniano (*hamiltonian path problem*, *HPP*). El objetivo del HPP consiste en, dado un grafo, determinar si existe un camino hamiltoniano entre dos nodos, es decir, un camino entre ambos que visite cada nodo del grafo exactamente una vez. Pues bien, la ordenación al generar las trayectorias en un sistema multirrobot puede considerarse como una instancia del mismo. Esta conclusión se extrae tras analizar más exhaustivamente diferentes cuestiones, según se explica en los siguientes apartados.

5.2.1. Complejidad del HPP

En general, se considera que el HPP es NP-completo tanto para grafos no dirigidos como para grafos dirigidos[8] (Garey y Johnson, 1979; Papadimitriou y Steinglitz, 1998). Sin embargo, para ciertos casos sí existe una solución polinomial: si el grafo es de tipo *tournament*, si el grafo es un grafo-arista, o bien si ningún grado de incidencia/excedencia supera la unidad. Como se verá en el siguiente subapartado, el grafo que representa el problema de la asignación de prioridades en el sistema multirrobot no se ajusta a ninguna de esas tipologías, por lo que puede tratarse como un problema NP-completo.

Se observa que la formulación del problema del camino hamiltoniano, tal y como se planteó anteriormente, corresponde a un problema de decisión, es decir, un problema con sólo dos soluciones posibles: sí o no. Empero, su utilidad para la ordenación de los robots de un sistema se basa en hallar el camino hamiltoniano de menor coste, por lo que sería necesario expresar el HPP como un problema de optimización. Aunque la Teoría de la NP-completitud estudia exclusivamente problemas de decisión, sus implicaciones pueden extenderse a los problemas de optimización, ya que los primeros pueden derivarse de éstos últimos. Así, si un problema de optimización busca la solución mínima entre todas las

8. Los conceptos sobre Teoría de Grafos que aparecen en este apartado pueden consultarse en las referencias (Dienstel, 2000; Eric's Weisstein's World of Mathematics, 2003; PlanetMath, 2003).

posibles, puede asociársele el problema de decisión que pregunte si existe una solución cuyo coste no sea mayor que cierto límite *L*. De igual manera se podría obtener el problema de decisión correspondiente a un problema de maximización, sustituyendo "no mayor que" por "al menos".

La principal característica de esta relación "decisión-optimización" radica en el hecho de que, siempre que la función de coste pueda evaluarse de manera sencilla, el problema de decisión no puede ser más complejo que el de optimización. Por ejemplo, si pudiera hallarse un camino hamiltoniano de coste mínimo para el HPP en tiempo polinomial, igualmente podría resolverse el problema de decisión en tiempo polinomial, calculando el camino de menor coste y comparando ese coste con el límite *L*. Por tanto, al demostrar que el HPP es NP-completo, se establece automáticamente que el problema de optimización correspondiente es, por lo menos, tan complejo como él. La relación entre problema de decisión y de optimización es, en algunos casos, mucho más estrecha: para muchos problemas de decisión, se demuestra que no son menos complejos que sus problemas de optimización.

5.2.2. Reducción del problema al HPP

Muchos problemas de ordenación se reducen típicamente al problema del camino hamiltoniano, y la asignación de prioridades en un sistema multirrobot es uno de ellos. La analogía existente entre este problema y un grafo sobre el que se busque un camino hamiltoniano se pormenoriza en lo que sigue:

- Los vehículos del sistema multirrobot se representan como los nodos del grafo.
- Las relaciones entre vehículos corresponden a las aristas del grafo. Estos arcos son dirigidos, de manera que la existencia de un arco *i* -> *j* entre los nodos *i* y *j* indica que primero se calcula la trayectoria para el vehículo *i*, y después para el vehículo *j* tomando *i* como obstáculo móvil. En principio, se supondrá que todos los robots se relacionan con los restantes sin ninguna limitación, por lo que el grafo que representa el problema será un grafo completamente dirigido. Sin embargo, podrían suprimirse algunas aristas para reflejar la existencia de restricciones en las precedencias de vehículos, debidas a situaciones como:
    - Posibilidad elevada de bloqueo. Inicialmente, no se conoce con certeza la existencia de un bloqueo entre dos robots, sólo se sabe si

el camino de uno termina sobre o en las cercanías del recorrido de otro. El bloqueo aparecerá dependiendo de las velocidades con las que los vehículos recorran sus respectivos caminos. Pero puede ser deseable cortar de raíz la posibilidad de que surja una obstrucción en los caminos, eliminando la arista que permite que el robot potencialmente causante del bloqueo preceda al otro.

- Favorecimiento de ciertos vehículos, siguiendo los criterios auxiliares que se comentaron anteriormente.

- Los costes asociados a la arista *i->j* se computan como el tiempo que tardaría el robot *j* en recorrer su camino, teniendo como obstáculo móvil en su entorno al robot *i*. Estos costes no son fijos, ya que ese tiempo también dependerá de los vehículos que antecedan al robot *i*. Explicado con un ejemplo, dadas las ordenaciones *[A-E-B-C-D]* y *[A-B-E-C-D]*, el coste de la arista *C->D* en ambos será diferente, ya que el orden de los robots predecesores también lo es.

- Un camino hamiltoniano en el grafo consistirá en un recorrido que incluye a cada uno de los nodos una única vez. Dicho camino representará una ordenación de vehículos, de manera que si un robot r_i precede en el camino hallado a un robot r_j, la trayectoria de r_i se generará antes que la de r_j.

Como se deduce de lo expuesto a lo largo de este apartado, la ordenación de la generación de trayectorias para un sistema multirrobot equivale a determinar el camino hamiltoniano de menor coste en el grafo que representa el sistema multirrobot.

5.2.3. Resolución del problema mediante análisis exhaustivo

Para cualquier problema de optimización combinatoria existe siempre un procedimiento obvio para hallar la solución óptima deseada: explorar el espacio de búsqueda completo, obtener el coste de todas las soluciones posibles, y escoger la mejor de todas. Por desgracia, la explosión combinatoria que caracteriza a este tipo de problemas impide la aplicación de tales métodos de fuerza bruta, tanto por la complejidad temporal que suponen como por la capacidad de almacenamiento que demandan.

Para ilustrar el coste que supondría resolver el problema de la asignación de prioridades con técnicas de este tipo, se presentan a continuación los resultados exhaustivos obtenidos para sistemas multirrobot de tres, cuatro, cinco, seis, siete y ocho elementos; en sistemas de orden mayor, la complejidad del problema causa que el tiempo de cálculo se dispare. Los análisis se han efectuado utilizando el software científico MATLAB, sobre un procesador Pentium a 1 GHz. Aunque evidentemente no se trata de la implantación más rápida, no influye de manera relevante en lo que se pretende demostrar con los resultados: la implantación de técnicas de fuerza bruta no resulta factible debido a su carácter exponencial.

La siguiente tabla muestra los tiempos empleados tanto para la generación del espacio de búsqueda del problema (esto es, dados *n* robots, obtener las *n!* permutaciones posibles), como para el análisis exhaustivo de cada una de las ordenaciones generadas. Las medidas de tiempo se expresan en segundos, y corresponden a la media tomada tras varias llamadas a programa, puesto que obviamente aparecen pequeñas diferencias entre ejecuciones sucesivas.

Nº de robots	Nº de soluciones	Tiempo de generación del espacio de búsqueda	Tiempo de análisis del espacio de búsqueda
3	6	0.012	8.1894
4	24	0.012	86.004
5	120	0.036	570.5066667
6	720	0.1886	4642.776000
7	5040	8.6562	46965.961000
8	40320	1158	-

Tabla 5.1: Explosión combinatoria del problema de la asignación de prioridades

En principio, parece que la etapa de generación del espacio de búsqueda consume muy poco tiempo frente a la de análisis. Sin embargo, conforme el número de robots crece, ese tiempo aumenta de manera exponencial, como puede observarse para el sistema con 8 vehículos. La fase de análisis del espacio de soluciones crece rápidamente, y para más de siete robots resulta impracticable.

5.2.4. Otros problemas relacionados

En lo que sigue, se exponen algunos otros problemas relacionados, de alguna manera, con la asignación de prioridades para un sistema con múltiples vehículos. Se trata de problemas relativos a la ordenación de robots, cuyo estudio ha permitido llegar a la conclusión de que dicho problema en realidad es una instancia del HPP.

- *El problema del viajante*. La diferencia entre el HPP y el TSP es la misma que existe entre camino hamiltoniano y ciclo hamiltoniano: aunque ambos deber recorrer todos los nodos del grafo una única vez, el ciclo hamiltoniano debe retornar al nodo de partida, mientras que para el camino hamiltoniano esa condición no se exige. Al igual que el camino hamiltoniano, el TSP es un problema NP-completo; algunas de sus variantes son:
    - TSP simétrico: la distancia entre el nodo *i* y el nodo *j* es la misma que la distancia entre el nodo *j* y el *i*.
    - TSP asimétrico (ATSP): la distancia del nodo *i* al *j* puede ser distinta de la distancia del *j* al *i*.
    - TSP geométrico (GTSP): encontrar una solución del TSP que no supere cierta extensión límite *L*, tomando como distancia entre nodos la euclidiana.
    - TSP extendido (TSE): a partir de una ruta parcial, se desea conocer si es posible construir el circuito hamiltoniano completo sin exceder un cierto coste.

  El TSP es un problema extensamente estudiado en la literatura; en (Traveling Salesman Problem Homepage, 2003) puede encontrarse un completo listado de referencias acerca del problema, así como otra información de utilidad.

- El problema del ordenamiento secuencial. El problema del ordenamiento secuencial (*sequential ordering problem*, *SOP*) consiste en obtener un camino hamiltoniano mínimo en un grafo dirigido cuyos nodos y arcos tienen pesos, teniendo en cuenta además relaciones de precedencia entre nodos. El SOP es un problema NP-completo, considerado como variante del TSP -incluso puede plantearse como una extensión del ATSP-, que encuentra su aplicación en campos como la planificación de producción, sistemas de fabricación flexible o enrutamiento de vehículos. Una descripción más completa puede hallarse en (Gambardella y Lorigo, 2000), donde se incluye una resolución del problema mediante heurísticos.

- El problema del enrutamiento de vehículos. Por último, el problema del enrutamiento de vehículos (*vehicle routing problem*, *VRP*) se refiere a una clase de problemas NP-completos, cuyo objetivo se centra en establecer un conjunto de rutas mínimas para una flota de vehículos basada en uno o varios almacenes, de manera que sirvan a un conjunto de clientes geográficamente dispersos. Se menciona en este apartado simplemente para aclarar que, a pesar del nombre, no refleja el problema de la asignación de prioridades, puesto que su objetivo no es proporcionar una ordenación.

5.3. Técnicas heurísticas y algoritmos genéticos

A la hora de resolver problemas de optimización combinatoria, pueden encontrarse tres tipos de enfoques: exacto, heurístico e híbrido. Las técnicas exactas persiguen el hallazgo de la solución óptima al problema; algunas de las más conocidas son el método simplex, ramificación y acotación (*branch and bound*), o programación dinámica. Sin embargo, la complejidad temporal de muchos de los problemas de optimización combinatoria impiden el uso de tales métodos, ya que el tiempo de cómputo resulta excesivo. Por ello, muchas veces resulta mucho más interesante encontrar soluciones factibles -que satisfagan las restricciones del problema- y que, aunque no sean las óptimas, se aproximen a ella en un tiempo de cálculo razonable. Así buscan las soluciones los métodos heurísticos (Díaz *et. al*, 1996), como el recocido simulado (*simulated annealing*, *SA*), la búsqueda tabú (*tabu search*, *TS*), las redes neuronales (*neural networks*, *NN*) o los algoritmos genéticos (*genetic algorithms*, *GA*).

Como se indicó en apartados previos, el HPP -y otros problemas de optimización combinatoria- es NP-completo, lo que significa que los únicos algoritmos conocidos para resolverlo requieren un tiempo de cómputo exponencial. Pero, debido al interés práctico, financiero o estratégico que muchos de estos problemas suscitan, encontrar una solución que pueda calcularse en un período de tiempo aceptable es una tarea primordial. Para estos casos, las técnicas heurísticas parecen una opción muy aconsejable ya que, además de proporcionar una buena solución de manera rápida, ofrecen las siguientes ventajas:

- Son especialmente útiles si se carece de datos fiables, en caso de que existan restricciones de tiempo o espacio que satisfacer, o bien cuando no sea imprescindible descubrir la solución óptima.
- La flexibilidad para manejar las características del problema es mucho mayor que con otras técnicas.

- Normalmente, su comprensión es sencilla, sobre todo cuando deben usarse por personal no experto.

La principal contrapartida de las técnicas heurísticas se debe a la falta de garantía de la calidad de la solución lograda, por lo que no es posible saber cuán próxima se encontrará al óptimo.

De entre todas las heurísticas anteriormente mencionadas, los algoritmos genéticos han sido los escogidos para afrontar el problema de la asignación de prioridades del sistema multirrobot: a los beneficios propios de cualquier heurístico, añaden su buen comportamiento en la resolución del TSP -que puede considerarse una variante del HPP (Garey y Johnson, 1979)- mientras que otros métodos, como programación dinámica o branch-and-bound, no son tan efectivos en cuestiones de complejidad.

Básicamente, los GA pueden definirse como "algoritmos que codifican una solución potencial a un problema específico mediante una estructura de datos simple similar a un cromosoma, y aplican operadores de recombinación a estas estructuras para preservar la información crítica" (Whitley, 1993). Esta formulación tiene dos posibles lecturas. Ciñéndose al modelo originalmente propuesto por Holland (Holland, 1975), un algoritmo genético trabaja con cromosomas binarios de longitud fija, y dos operadores géneticos únicos: cruzamiento y mutación. En un sentido más amplio, un algoritmo genético parte de una población inicial aleatoria codificada con alguna estructura de datos (que no tiene que ser, de manera obligada, binaria o de longitud constante), y mediante operadores genéticos que seleccionen y recombinen individuos de la población, genera nuevos individuos dentro del espacio de búsqueda. Este último planteamiento, que se discute con mayor detalle en (Michalewicz, 1992), es el adoptado en esta tesis, y será expuesto en subsecciones venideras.

Para finalizar, conviene comentar que los GA se han aplicado a la planificación de trayectorias multirrobot, pero de manera distinta a como aquí se propone. Así, los algoritmos genéticos no se emplean como herramientas para la obtención de un esquema de prioridades, sino como medios de generación de caminos/trayectorias bajo circunstancias muy diversas: entornos dinámicos (Leung y Zalzala, 1994), manipuladores robóticos (Monteiro y Madrid, 1999; De la Cueva y Ramos, 1998), aprendizaje automático (Gallardo *et al*., 1997) y planificación descentralizada de caminos (Shibata *et al.*, 1992; Shibata y Fukuda, 1993).

5.4. Implantación del algoritmo genético

Esta subsección se detiene en los aspectos de implantación del algoritmo genético concreto que resuelve el problema de la asignación de prioridades en la planificación de trayectorias multirrobot. Son cinco los puntos que definen las líneas básicas de dicha implantación, y se detallan en lo que sigue.

5.4.1. Codificación de las soluciones

Para el HPP, la representación de individuos basada en una codificación binaria no es la más indicada. En primer lugar, no corresponde a una representación natural del problema, puesto que no resulta lógico pensar en permutaciones como cadenas binarias. En segundo lugar, y lo que resulta más grave, deja abierta la posibilidad de que, tras la aplicación de los operadores genéticos, surjan cromosomas erróneos (por ejemplo, ordenaciones en las que se visita el mismo nodo del grafo dos veces); esta situación obligaría a establecer mecanismos que corrigieran tales fallos tras la recombinación de los cromosomas, lo que supone un coste computacional añadido.

Es por este motivo por lo que se ha empleado una codificación entera de las soluciones, de forma que un individuo representa una permutación cualquiera perteneciente al espacio de soluciones. Así, la posición dentro del cromosoma señala el orden en el que se calcula la trayectoria, mientras que el valor de la posición equivale al vehículo cuya trayectoria se genera.

Para ilustrar esta idea, se propone el siguiente ejemplo. Dado un sistema multirrobot compuesto por cinco elementos, un esquema de prioridades válido podría representarse mediante el individuo [5 3 1 4 2]. El primer robot en obtener la trayectoria sería el vehículo 5; a continuación, el robot 3 obtendría su perfil geométrico-espacial considerando al robot 5 como obstáculo móvil; tras él, se computaría la trayectoria del vehículo 1 bajo la influencia de los vehículos móviles 5 y 3, y así sucesivamente.

5.4.2. Población inicial

La población inicial de la que parte el GA es un subconjunto del espacio de búsqueda total elegido aleatoriamente: para un sistema multirrobot de n elementos, la población inicial constaría de x elementos tomados de las $n!$ permutaciones existentes.

5.4.3. Función de evaluación

La función de evaluación -o función de *fitness*, ya que ambos términos suelen emplearse indistintamente (Whitley, 1993)- es una medida de la eficiencia de cualquier cromosoma respecto a un conjunto de parámetros que define el propio problema que se desea resolver. Es decir, la función de evaluación determina el grado de bondad de un cromosoma de la población.

Para la asignación de prioridades en un sistema multirrobot, la función de evaluación se establece como sigue. Todas las trayectorias involucradas en un cromosoma se calculan, respetando el orden establecido. Como se explicó en el capítulo tres, cada trayectoria posee una componente temporal que guarda el tiempo de navegación empleado por el vehículo para recorrer su camino. El valor que la función de evaluación toma es la suma de los tiempos de navegación de los distintas trayectorias del cromosoma.

Puede deducirse de lo expuesto en el párrafo anterior que la función de evaluación cumple uno de los requisitos exigibles al planificador de trayectorias multirrobot: encontrar la secuencia de robots que minimice cierto criterio; en este caso, el tiempo de navegación de todos los vehículos del sistema. La otra condición que debía verificarse -la ausencia de bloqueos- también se garantiza: la función de evaluación asigna a aquellos esquemas de prioridades que correspondan a situaciones de este tipo un valor muy elevado, lo que los convierte en individuos con escasas posibilidades a la hora de propagarse durante la ejecución del algoritmo genético.

5.4.4. Operadores genéticos

Los operadores genéticos utilizados en el algoritmo genético implantado son los de selección, cruzamiento y mutación; serán descritos en lo que sigue con mayor detenimiento.

- Operador de selección. El operador de selección elige aquellos individuos de la población actual que son más apropiados para la reproducción, de manera que los cromosomas mejor evaluados tienen mayores oportunidades de ser seleccionados para procrear y, por tanto, de transmitir sus características a posteriores generaciones. El operador de selección utilizado para este caso es la selección mediante ruleta (*roulette wheel*), propuesto por Holland en su trabajo original (Holland, 1975). La idea consiste en asignar a cada cromosoma de la población una porción de una rueda de ruleta proporcional al valor de su función de evaluación; de esta manera, los más aptos tendrán

mayor probabilidad de ser escogidos en cada tirada de la ruleta. En la literatura pueden encontrarse otros operadores de selección, como el de torneo (t*ournament selection*), o la selección basada en ranking (*ranking selection*).

- Operador de cruzamiento. El cruzamiento es responsable del intercambio genético entre dos cromosomas seleccionados que generan, con una cierta probabilidad, descendientes que combinan las mejores características de ambos progenitores. El algoritmo clásico de Holland utiliza cruzamiento de un punto, inspirado en procesos biológicos, pero este tipo de cruce no es apropiado para el HPP ya que no impide la codificación de individuos erróneos. Existen varios operadores que sí respetan los requerimientos, como el cruzamiento parcialmente mezclado (*partially mixed crossover*, *PMX*), el cruzamiento heurístico (*heuristics crossover*, *HX*), la recombinación de aristas (*edge recombination*, *ER*) o el cruzamiento basado en orden uniforme (*uniform order-based crossover*, *OX*); este último ha sido escogido para la implantación del algoritmo genético debido a su sencillez y a que ninguno de ellos presenta ventajas destacables frente a los demás. Brevemente descrito, el PMX crea una plantilla binaria aleatoria de longitud igual a la de los padres. Uno de los hijos hereda la información del primer padre en todas aquellas posiciones correspondientes a 1 en la plantilla binaria; el resto de la secuencia se completa a partir de los valores del segundo padre que aún no están presentes en el hijo, manteniendo el orden que presentan en dicho padre. El otro hijo se obtiene de manera similar, pero invirtiendo el papel de los progenitores. Para clarificar el funcionamiento del operador PMX, la Figura 5.1 muestra un ejemplo del cruce de dos individuos.

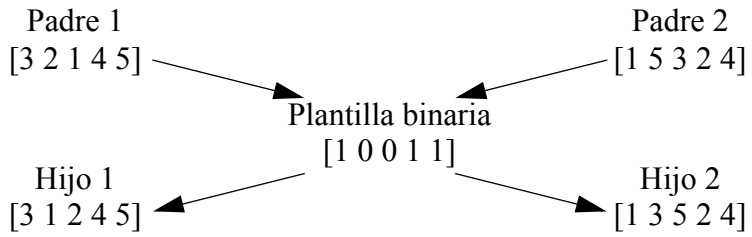


Figura 5.1. Cruzamiento mediante el operador genético PMX

- Operador de mutación. El objetivo de la mutación es introducir divergencia con una cierta probabilidad en una población que tiende a converger. De esta manera se garantiza la oportunidad de escapar de un mínimo local en el que pudiera haberse caído. El operador de mutación escogido en este trabajo es el intercambio recíproco (*reciprocal exchange*), que simplemente intercambia los valores de dos posiciones aleatorias del cromosoma.

5.4.5. Elitismo

La aplicación del algoritmo genético tal y como se ha explicado hasta el momento presenta una dificultad: el mejor elemento hallado en una generación puede desaparecer en la siguiente. Es decir, no se garantiza la permanencia de la mejor solución obtenida en una generación, ya que las operaciones de cruzamiento y mutación pueden tener como consecuencia su eliminación de la población actual. Para evitar esta situación, se emplea el *elitismo*, una estrategia consistente en copiar el mejor individuo de una población (denominado *superindividuo*) en la generación sucesora, forzando así que pueda reproducirse y generar a su vez buenas soluciones.

5.4.6. Otros parámetros

Otros parámetros son también de importancia al diseñar un algoritmo genético: tamaño de la población, número de generaciones y probabilidad de que se produzcan operaciones tanto de cruzamiento como de mutación. No existen reglas establecidas a la hora de asignarles un valor, ya que muchas veces éste dependerá de las características intrínsecas del problema a tratar; sin embargo, un punto de partida puede ser el conjunto de valores propuestos en (GA Tutorial Home Page, 2004). El detalle de los parámetros empleados en el algoritmo genético desarrollado en la tesis se realiza en la subsección 5.5.3 dedicada al examen de los resultados obtenidos en su aplicación.

5.5. Experimentación y Resultados

Los experimentos se han ejecutado sobre una máquina PC a 1 GHz bajo Windows 2000, usando el software matemático para ciencia e ingeniería MATLAB, en su versión R12. Como ya se mencionó con anterioridad, esta herramienta software no produce la implantación más eficiente, pero resulta muy cómoda de manejar y ofrece utilidades gráficas que permiten analizar los resultados fácilmente.

El banco de pruebas establecido corresponde a un espacio de trabajo en el que se desenvuelven siete robots, cuyos caminos geométricos aparecen en la Figura 4.3, con los puntos de inicio marcados con una *o* y los puntos de fin señalados con una *x*. Puede apreciarse que, si bien algunos vehículos siguen caminos independientes (caso del Robot 7), otros deben guiarse a través de zonas en las que abundan los cruces.

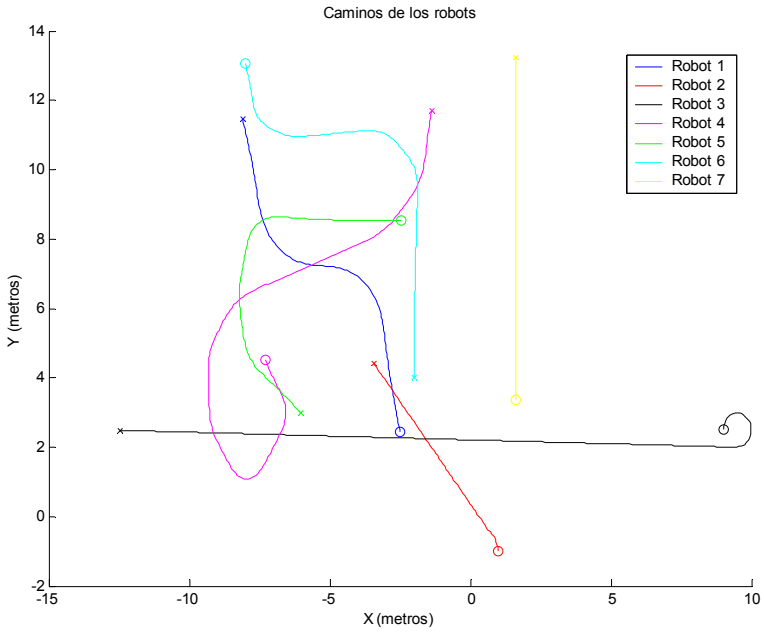


Figura 5.2. Caminos geométricos de los vehículos móviles

Este sistema multirrobot de siete elementos ha dado lugar a los tres tipos de pruebas que se muestran dentro de esta subsección. En primer lugar, como paso deseable para la comprobación de la bondad del algoritmo genético implantado, se ha llevado a cabo el análisis exhaustivo del espacio de soluciones del sistema multirrobot. A continuación, se demuestra también que la búsqueda aleatoria de soluciones no debe considerarse un método efectivo para solventar el problema. Por fin, se analizan los resultados producidos por el algoritmo genético que determina la mejor ordenación de vehículos.

5.5.1. Análisis del espacio de soluciones

El análisis del espacio de soluciones, necesario para poder estimar la eficiencia del algoritmo desarrollado, supone realizar una búsqueda exhaustiva sobre el mismo, calculando el coste asociado a cada una de las ordenaciones que lo componen obtenido según el criterio de menor tiempo de navegación de los vehículos del sistema. Los datos

obtenidos tras dicha búsqueda se presentan en la Tabla 5.2. En ella se observa que el menor coste posible es de 169.94 segundos, y se encuentra asociado a 280 permutaciones, lo que supone un 5.55% del espacio de soluciones; el peor coste, debido a situaciones de bloqueo imposibles de gestionar para el sistema, corresponde a 10^6 segundos, ocupando casi un 72% del total. Por tanto, la proporción entre malas soluciones y buenas soluciones es llamativamente más favorable a las primeras.

Coste	Ocurrencias	Porcentaje
1.6994e+002	280	5.55%
1.8573e+002	280	5.55%
1.9200e+002	280	5.55%
2.1842e+002	126	2.5%
2.2151e+002	21	0.4%
2.4563e+002	21	0.4%
2.5555e+002	84	1.66%
2.7333e+002	63	1.25%
2.7432e+002	21	0.4%
2.7756e+002	21	0.4%
2.8306e+002	21	0.4%
3.2687e+002	28	0.55%
4.2210e+002	42	0.83%
4.2218e+002	126	2.5%
1.0000e+006	3626	71.94%

Tabla 5.2: Análisis de costes en el espacio de soluciones de un sistema multirrobot de siete elementos

Gráficamente, la información incluida en la Tabla 5.2 se muestra en las Figuras 5.3 y 5.4. En ambas figuras, las permutaciones se sitúan a lo largo del eje de abcisas, sin seguir ningún orden concreto puesto que ningún orden puede establecerse inicialmente sobre ellas; el eje de ordenadas refleja el coste asociado a cada ordenación. La Figura 5.3 representa los costes reales del espacio de búsqueda, donde las ordenaciones válidas para el sistema multirrobot conviven con las correspondientes a situaciones cuyo coste asignado es muy elevado; se ha indicado con un asterisco el mejor coste.

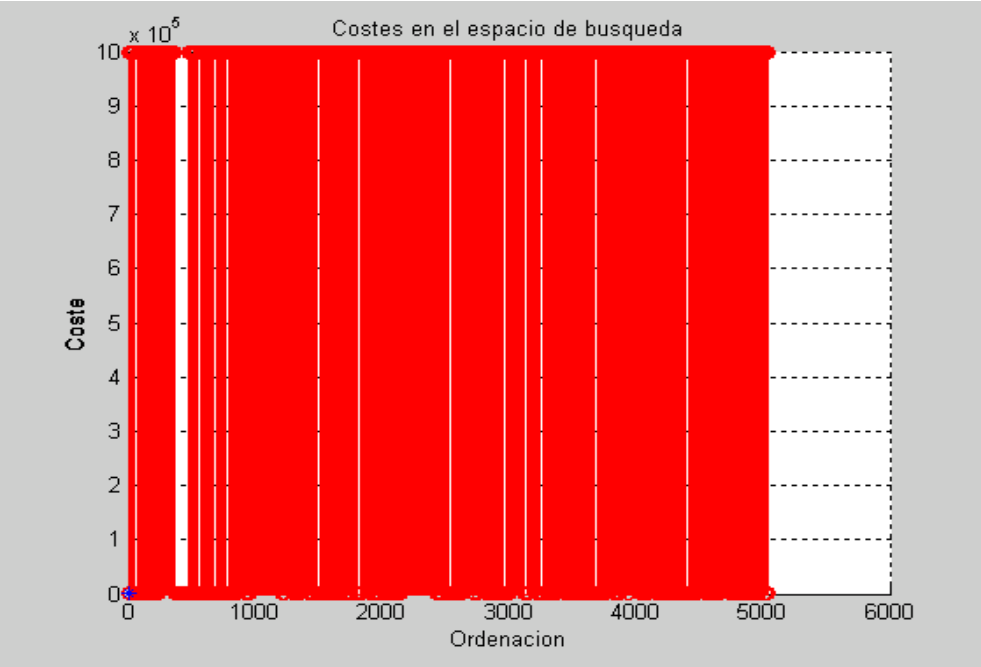


Figura 5.3. Costes reales en el espacio de búsqueda

La gran desigualdad entre los costes de las situaciones de bloqueo y los de las restantes ordenaciones impide que la distribución de estos últimos pueda apreciarse sobre la gráfica anterior. Por ello, en la Figura 5.4 se han escalado los costes, sustituyendo los de las ordenaciones de bloqueo por el máximo coste de las ordenaciones válidas; los nuevos valores escalados se representan como círculos claros en la imagen, mientras que el menor coste posible sigue marcándose con un asterisco. Las dos gráficas se complementan y ofrecen así una visión más clara del espacio de búsqueda al que se enfrentará el algoritmo genético desarrollado.

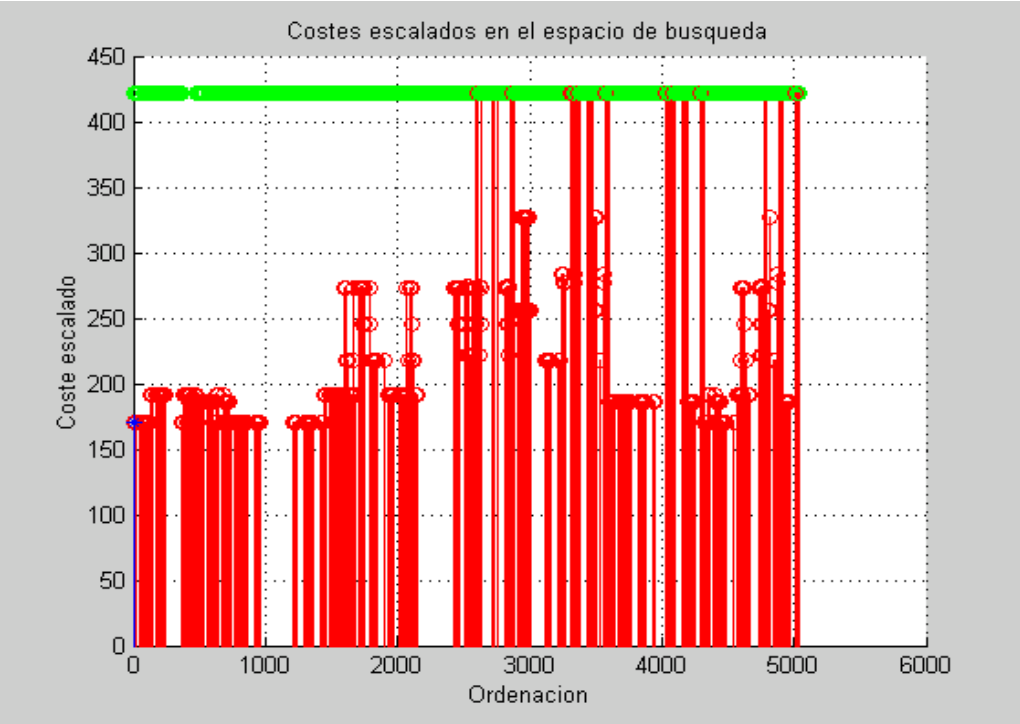


Figura 5.4. Costes escalados en el espacio de búsqueda

5.5.2. Resultados mediante búsqueda aleatoria

Existe una forma muy simple de resolver el problema de la planificación de trayectorias multirrobot: seleccionar, aleatoriamente, una ordenación del espacio de soluciones, y tomarla como solución válida. Evidentemente, no parece una buena idea, ya que depende totalmente de la distribución de las soluciones, y esta es una información que se desconoce en cualquier problema de mediana entidad que se aborde.

En esta subsección se va a corroborar la escasa utilidad de la búsqueda aleatoria, examinando los datos del espacio de búsqueda que se han obtenido para el sistema multirrobot de siete elementos propuesto. El hecho de que el 72% de las soluciones posibles sea desechable ya es un indicador del mal resultado que la búsqueda aleatoria proporciona, puesto que la probabilidad de obtener no ya la solución óptima, sino una solución válida, es bastante reducida. Las pruebas concretas realizadas arrojan unos datos similares: tras la ejecución de 150 búsquedas aleatorias, en el 80% de los casos la ordenación encontrada es

la peor, mientras que sólo el 3.33% de las soluciones halladas corresponden a una ordenación óptima. Por tanto, la búsqueda aleatoria no puede ser considerada como un método realista para determinar la mejor planificación de trayectorias en un sistema multirrobot.

5.5.3. Resultados mediante el uso del algoritmo genético

Por fin, en esta subsección se estudiarán los resultados proporcionados por el algoritmo genético cuyo desarrollo se ha explicado a lo largo del capítulo. Antes de la experimentación, los parámetros del mismo se ha ajustado con los valores expuestos en la Tabla 5.3:

Parámetro	Valor
Población inicial	10 individuos
Nº de generaciones	20 generaciones
Operador de selección	Ruleta
Probabilidad de cruce	0.8
Operador de cruce	Basado en orden uniforme
Probabilidad de mutación	0.1
Operador de mutación	Intercambio recíproco
Elitismo	1 individuo

Tabla 5.3: Parámetros del algoritmo genético implantado

Son dos las consecuencias generales más destacables que pueden extraerse de las sucesivas ejecuciones del algoritmo genético. En primer lugar, en el 100% de los casos se ha obtenido una solución válida, de los que el 90% corresponde a la óptima; hay que tener en cuenta que se parte de una población inicial generada aleatoriamente, por lo que la mayor parte de sus individuos serán no válidos, como se desprende del estudio del espacio de búsqueda efectuado anteriormente. En segundo lugar, el tiempo empleado en cada generación ronda de media los 135.5109778 segundos, lo que implica que, para producir 20 generaciones y resolver el problema, se necesitan en torno a los 45 minutos; si se compara con las casi siete horas que requiere la búsqueda exhaustiva, la mejora es clara.

Para finalizar, se comenta con mayor detalle uno de los experimentos realizados (figuras 5.5 y 5.6), que puede tomarse como un caso típico de la aplicación del algoritmo genético implantado. El mejor cromosoma obtenido en la primera generación no corresponde a la solución óptima, pero el avance del algoritmo genético logra mejoras sucesivas hasta llegar a la mejor ordenación posible, según se observa en la Figura 5.5. El hecho de que la permutación óptima surja en la última generación podría indicar que el parámetro que indica el número de generaciones necesarias para que el algoritmo genético consiga un buen valor no es correcto; sin embargo, en la gran mayoría de los casos estudiados la mejor solución no se demora tanto, por lo que puede considerarse que 20 generaciones son suficientes para encontrarla.

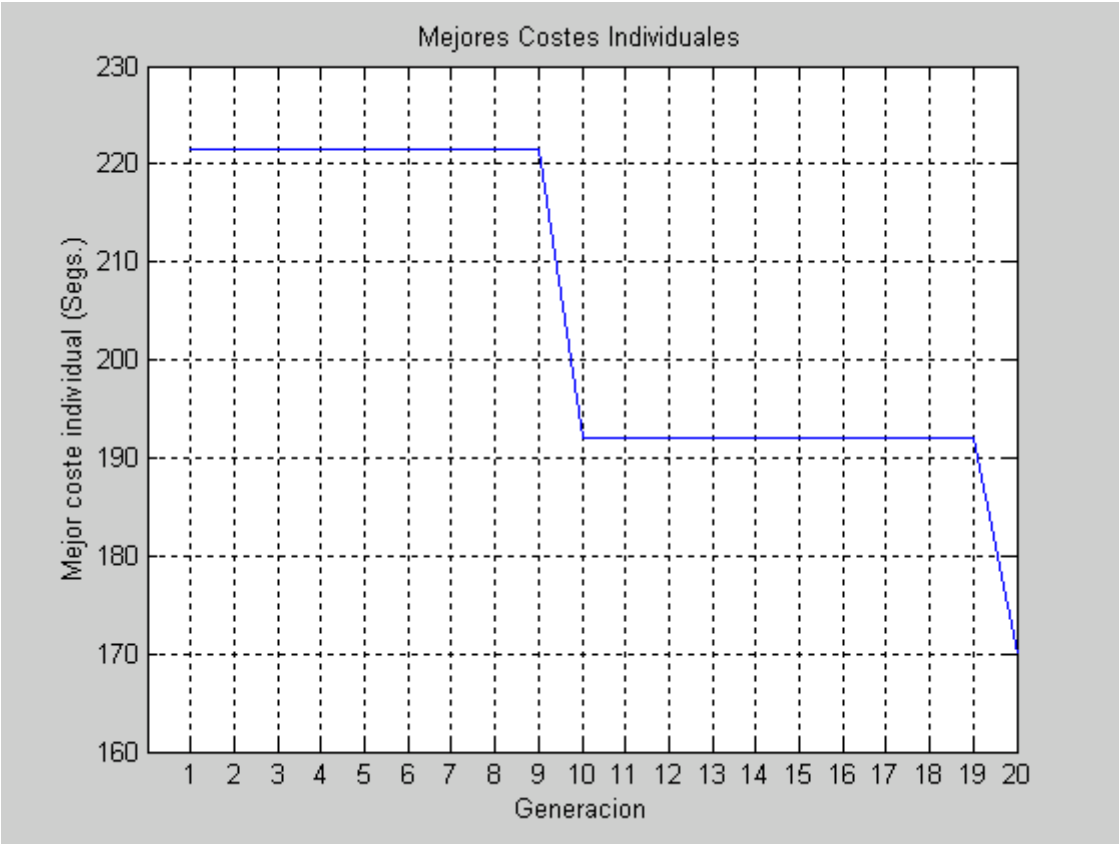


Figura 5.5. Evolución del mejor coste individual

Por otra parte, en la Figura 5.6 se muestra la evolución del coste medio de cada generación, es decir, el valor medio de todos los cromosomas que componen una generación. La tendencia en esta gráfica, aunque descendente, presenta algunos picos, siendo el más señalado el que se produce en la decimoctava generación. Estas variaciones tan elevadas son debidas a la actuación de los operadores de cruce y mutación, que pueden generar individuos peores que los encontrados hasta el momento, y cuyo coste dispare el

valor medio de la generación. No debe pensarse que éste es un efecto negativo, al contrario: demuestra que el cruce y la mutación producen individuos diferentes a los que pueblan mayoritariamente una población, aumentando las posibilidades de escapar de mínimos locales o de poblaciones que convergen prematuramente. Este factor, combinado con el elitismo, permite mantener buenos individuos a lo largo de generaciones sucesivas sin cerrar la posibilidad de encontrar otros mejores.

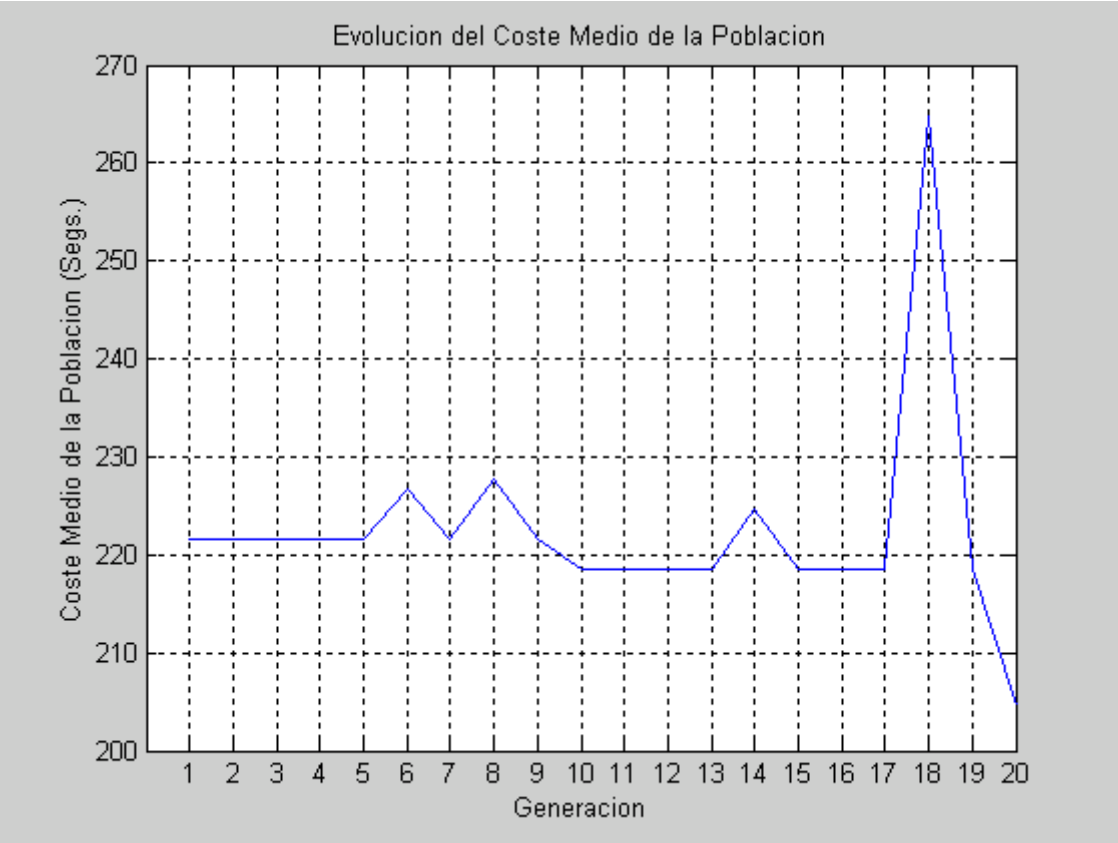


Figura 5.6. Evolución del coste medio asociado a una generación

Los perfiles de velocidad de cada vehículo según la ordenación determinada por el algoritmo genético pueden verse en la Figura 5.7. En principio, todos los vehículos pueden seguir su camino a la velocidad máxima permitida por las restricciones cinemáticas y dinámicas que actúan sobre ellos; sin embargo, los vehículos 4 y 5 deben enlentecer su marcha para esquivar situaciones conflictivas. Así, el vehículo 4 sortea la zona de riesgo alrededor de los cruces con los robots 1 y 5; por otra parte, el vehículo 6 evita el choque con el robot 2 que se produciría al final de su recorrido si no redujese su velocidad.

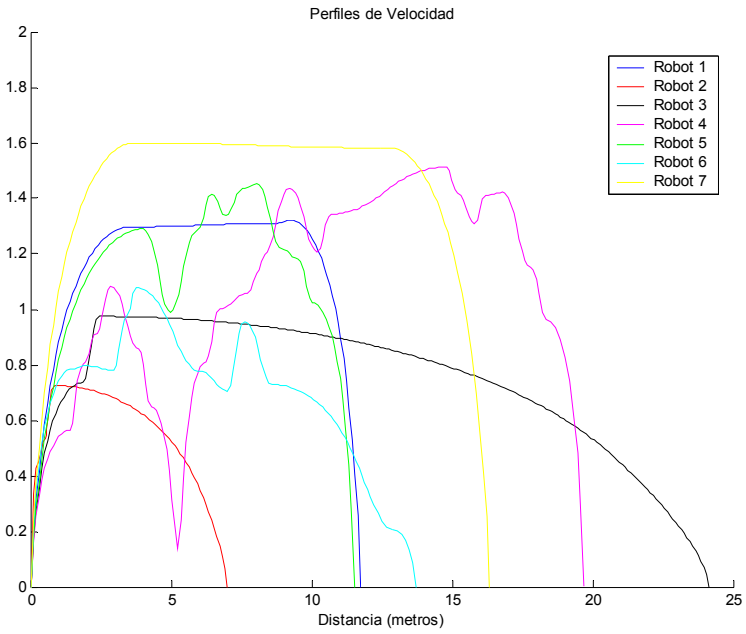


Figura 5.7. Perfiles de velocidad de los robots del sistema según la ordenación hallada por el GA

5.6 Conclusiones

En el capítulo, se propone la identificación del planteamiento de navegación desacoplado priorizado con el problema del camino hamiltoniano o HPP. Ello implica que la determinación del esquema de prioridades empleado para el cálculo de las trayectorias del sistema multirrobot debe tratarse como un problema de optimización combinatoria NP-completo, lo que descarta su resolución mediante aproximaciones basadas en fuerza bruta. Para el tipo de problema a resolver, las técnicas heurísticas parecen ser las más apropiadas, destacando entre ellas los algoritmos genéticos como las más idóneas para el caso concreto del HPP. Por tanto, se ha implantado un algoritmo genético, adaptando los diferentes operadores genéticos y parámetros de funcionamiento al problema de la obtención de las prioridades multirrobot; los resultados obtenidos tras su aplicación señalan que el algoritmo genético procura soluciones óptimas o casi óptimas, proporcionando además una reducción en el tiempo de cálculo realmente significativa.

CAPÍTULO 6. Conclusiones y Trabajos Futuros

6.1. Conclusiones

El trabajo presentado en esta tesis puede resumirse, de manera muy esquematizada, como un procedimiento destinado al cálculo de las mejores trayectorias para los vehículos que conforman un sistema multirrobot. Para ello, han sido dos los puntos principales de estudio. Por una parte, se ha desarrollado una técnica completa de planificación de velocidades multirrobot; por otra, se ha establecido un mecanismo que permita seleccionar, de entre todas las posibles, las mejores trayectorias de velocidad para cada uno de los elementos del sistema.

La técnica de planificación de velocidades para sistemas de múltiples vehículos introducida se basa en el algoritmo de planificación de velocidades monorrobot propuesto por Muñoz, que ofrece como una de sus aportaciones la consideración de las restricciones de velocidad que actúan sobre el vehículo -ya sean de carácter mecánico, cinemático, dinámico u operacional- y que tienen una influencia significativa en el momento de seguir un camino. Este método se ha mejorado y extendido para que pueda usarse de forma combinada con la conocida descomposición camino-velocidad de Kant y Zucker. El resultado de esta unión es una metodología de planificación de velocidades en presencia de obstáculos móviles, que tiene en cuenta las restricciones existentes sobre el movimiento del vehículo, y cuyo cálculo es sencillo y poco costoso computacionalmente. El hecho de que el algoritmo analice la existencia de otros vehículos en el entorno del robot, hace posible que pueda adaptarse a sistemas multirrobot, surgiendo así el mecanismo de planificación de trayectorias multirrobot desacoplado priorizado planteado en la tesis.

La planificación de trayectorias multirrobot se ha incorporado a la arquitectura de control del robot Auriga-α, implantada bajo el sistema de integración de software robótico NEXUS, desarrollado en su totalidad en el Departamento de Ingeniería de Sistemas y Automática. Para ello, se ha construido un módulo NEXUS, denominado ICE_Velocidad, que genera el plan de velocidad para un vehículo móvil a partir del camino geométrico que va a seguir; junto con este servicio básico, se comentan dos servicios añadidos, que

describen más ampliamente los errores que pueden surgir en la ejecución del módulo, por una parte, y que implantan un pequeño interfaz de ventanas que simplifica el manejo del mismo, por otra parte. Los efectos laterales que dicha integración provoca son también analizados, y se proponen soluciones que permitan solventarlos.

Para un sistema multirrobot dado, diferentes ordenaciones al aplicar la técnica de planificación de trayectorias originarán diferentes conjuntos de trayectorias que permitan la navegación de sus elementos. Si esas trayectorias se evalúan según algún tipo de criterio, elegido para la ocasión, resultará que unas trayectorias serán mejores que otras. En esa coyuntura, sería conveniente que la metodología de planificación de trayectorias proporcionara las mejores soluciones, medidas con las normas definidas para el caso. Así, la construcción de los perfiles de velocidad óptimos se ha efectuado en dos etapas. En primer lugar, se ha identificado el tipo de problema a que corresponde, llegándose a la conclusión de que la planificación de trayectorias multirrobot es una instancia del problema del camino hamiltoniano, que a su vez es un problema de optimización combinatoria. Al tratarse de un problema NP-completo, una buena aproximación para resolverlo la proporcionan las técnicas heurísticas, de entre las que se han elegido los algoritmos genéticos como herramienta para determinar la mejor ordenación de los vehículos. Por supuesto, se ha llevado a cabo la implantación de un algoritmo genético concreto para este problema, que ha ofrecido buenos resultados.

Junto con estas cuestiones fundamentales para la planificación de trayectorias multirrobot, se ha realizado un trabajo de documentación exhaustivo, que ha dado origen a un sistema taxonómico para la clasificación de estrategias de navegación deliberativas que pueden hallarse en la literatura, con el que además se ha situado convenientemente la solución propuesta dentro de todas las soluciones de navegación multirrobot analizadas. A ello se añade el estudio, también bastante extenso, de los principales métodos de clasificación de sistemas multirrobot, así como de las arquitecturas multirrobot más conocidas en la actualidad.

Por último, y como herramienta auxiliar que facilite la prueba y comparación de los resultados obtenidos, se ha desarrollado SIMUNA: una herramienta de simulación multirrobot general, flexible, simple y no propietaria, diseñada con técnicas orientadas a objeto y que puede ejecutarse en cualquier plataforma. Para lograr la concurrencia propia de los entornos multirrobot se ha empleado un Sistema de Eventos Discretos, por lo que puede

utilizarse en equipos no multitarea. SIMUNA ha sido creada para constituir la base de desarrollos complejos, que puedan adaptarse lo más posible al sistema físico a simular; su utilidad se muestra especialmente en aquellos casos en los que desarrollar un simulador para un problema concreto es menos costoso que adaptar una herramienta ya creada.

6.2. Trabajos Futuros

Puesto que, como se ha podido comprobar, son varios los aspectos tratados para conseguir el algoritmo de planificación de trayectorias multirrobot definitivo, los trabajos futuros que pueden completarlo se han dividido en función de la cuestión a la que atañen. A continuación se desglosan más concretamente:

6.2.1. Planificación de velocidades multirrobot

El campo más amplio para nuevas investigaciones se corresponde con la planificación de trayectorias multirrobot, ya que no resulta complicado añadir nuevas funcionalidades al mismo, que mejoren su eficiencia y sus oportunidades de aplicación a distintas situaciones. Entre estas posibles extensiones, se destacan las que siguen:

- la incorporación de un sistema de comunicaciones que permita el trasvase de información entre los integrantes del sistema. De esta manera, y como ya se indicó en el capítulo cuatro, se abren nuevas posibilidades a la hora de la resolución de conflictos, planificación y desempeño de tareas, etc.

- en la arquitectura de control robótico implantada sobre Auriga-α se ha diseñado una maniobra de evitación local de obstáculos móviles, basada en tres etapas consecutivas: estimación del tamaño y movimiento del obstáculo (mediante extrapolación lineal o filtro de Kalman), análisis de colisión, y generación de reacción (Fernández Ramos, 2001). Este último paso es el más importante, ya que implica la replanificación de la velocidad del vehículo de manera que sortee un obstáculo móvil imprevisto. La solución aplicada hasta el momento toma como nueva referencia de velocidad un valor escogido entre la velocidad original del vehículo, y otro valor obtenido tras la aplicación de técnicas borrosas. Sin embargo, esta idea presenta dos problemas fundamentales: no se tiene en cuenta la velocidad real del robot, ni si la restricción de aceleración máxima que actúa sobre el robot permite llegar a la nueva referencia calculada. El efecto de estas limitaciones se

traduce en que la replanificación de velocidades puede ser insuficiente para eludir la colisión. Como trabajos futuros, resultaría interesante evaluar las propuestas de (Garnier y Fraichard, 1997), basadas en el uso de tablas borrosas cuyas entradas corresponden al estado actual del robot y su entorno, mientras que las salidas se refieren a la velocidad y aceleración del robot. En la misma línea, las técnicas utilizadas por Protzel, Holve y otros (Protzel et al., 1993; Holve et al., 1995a; Holve et al., 1995b) establecen varias tablas borrosas según la velocidad y aceleración del vehículo. Por otra parte, también resultaría interesante el uso de mecanismos de generación automática de tablas borrosas.

6.2.2. Establecimiento de la solución óptima

De cara a mejorar y ampliar el mecanismo de determinación del esquema de prioridades que da lugar a la mejor solución, podría hacerse mayor hincapié en los siguientes aspectos:

- comparar los resultados obtenidos empleando otros operadores genéticos. Aquí se incluiría la modificación de la función de evaluación de los cromosomas, para lograr una mayor eficiencia. Para esta tarea, sería de utilidad determinar la complejidad computacional del algoritmo de planificación de trayectorias multirrobot.
- estudiar el comportamiento del algoritmo genético para sistemas multirrobot de grandes dimensiones, integrados por decenas o incluso centenares de vehículos.
- evaluar la posibilidad de paralelización del algoritmo genético, para así lograr una mayor velocidad de cálculo.

6.2.3. Simulador de entornos multirrobot

El simulador de entornos multirrobot SIMUNA principalmente adolece, en la actualidad, de la falta de interfaces con el usuario que faciliten la comunicación entre ambos:

- por un lado, un interfaz gráfico que permita la gestión del simulador de manera sencilla y cómoda.

- por otra parte, un generador de informes configurable a voluntad, que devuelva la información asociada a las simulaciones dependiendo de las necesidades del usuario de la aplicación.

Apéndice A. El Robot Autónomo Móvil Auriga-α

A.1. Descripción general de Auriga-α

El robot autónomo móvil Auriga-α (Pedraza, 2000; Pedraza y otros, 1999) es el primer vehículo desarrollado en su totalidad por el Departamento de Ingeniería de Sistemas y Automática de la Universidad de Málaga; con él se pretende establecer un banco de pruebas efectivo para la investigación en el campo de la Robótica móvil. Las Figuras A.1 y A.2 muestran dos vistas del robot móvil: la primera corresponde a una imagen fotográfica del mismo, mientras que en la segunda se observa el interior del vehículo, sin la carcasa que habitualmente lo protege.

Figura A.1. El Robot Autónomo Móvil Auriga-α

El robot es apto tanto para la navegación interior como en entornos exteriores. Gracias a su sistema de locomoción, que le permite desplazarse en suelos irregulares y poco o nada preparados, es en este último tipo de ambientes donde encuentra un mayor rango de aplicaciones, como inspección, vigilancia o reconocimiento del terreno.

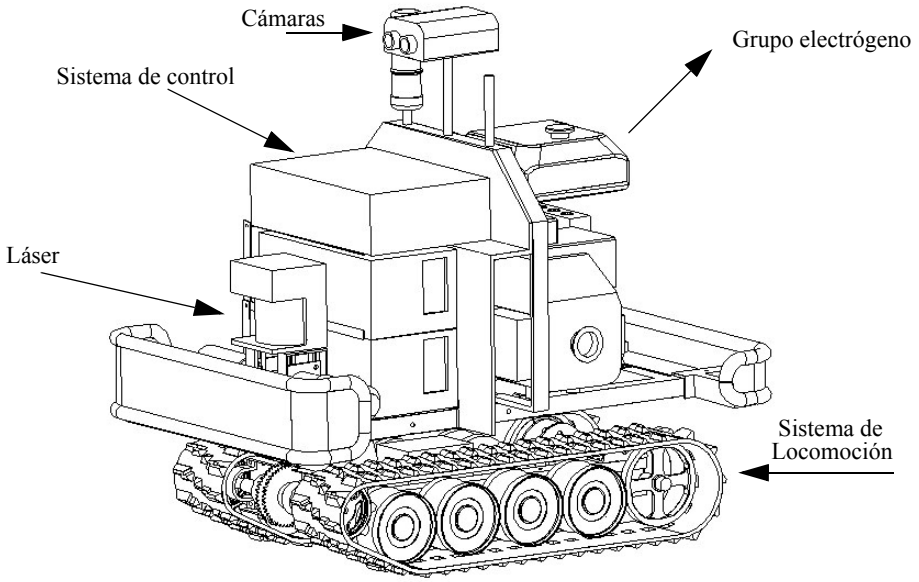


Figura A.2. Estructura interna de Auriga-α

El robot se ha construido sobre la base de una carretilla todoterreno de tipo oruga destinada al transporte de material. Estructuralmente, en el robot se diferencian dos partes principales: el sistema de locomoción, y el cuerpo del vehículo. A su vez, el cuerpo del robot se encuentra dividido en dos sectores: uno, en el que se aloja el equipo de alimentación autónoma, y otro, que alberga los sistemas de computación y electrónica. Una carcasa de poliéster, compuesta por tres piezas independientes, cubre el cuerpo del vehículo aislando sus elementos de los agentes externos; además, para reforzar la seguridad, se han añadido dos parachoques de hierro en la parte anterior y posterior del mismo. Las Figuras A.3 y A.4 presentan una vista delantera y otra trasera en las que se observa la integración de los elementos internos del robot y la carcasa.

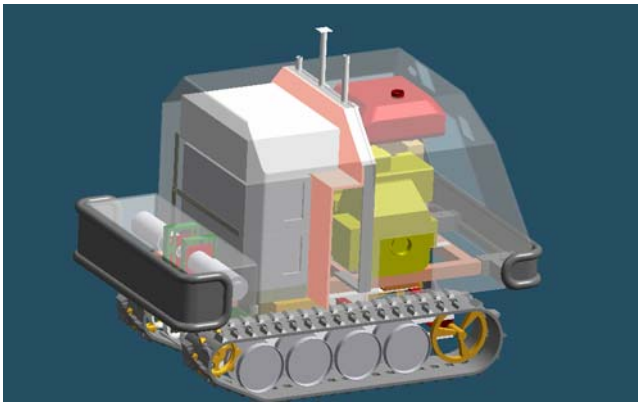


Figura A.3. Vista frontal del interior y la carcasa de Auriga-α

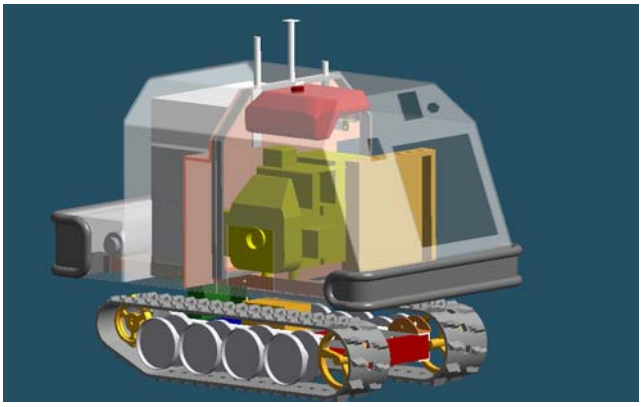


Figura A.4. Vista posterior del interior y la carcasa de Auriga-α

A.2. Sistemas básicos de Auriga-α

El funcionamiento de Auriga-a depende de una serie de sistemas con una función concreta, que interactúan entre ellos para lograr que el vehículo se comporte de la manera deseada. A continuación se enumeran tales sistemas, a la par que se comenta brevemente la tarea desempeñada por cada uno de ellos:

- Sistema mecánico y de locomoción. Diferentes elementos mecánicos se conectan entre sí para permitir el movimiento relativo de unos respecto a otros; la morfología del robot queda establecida por la manera en que tales elementos se unen.
- Sistema de actuación. Su labor consiste en provocar el movimiento relativo de los elementos del sistema mecánico, siguiendo las consignas determinadas por el sistema de control.

- Sistema sensorial. Se ocupa de recopilar información tanto del entorno exterior como del estado interno del robot. Su misión resulta fundamental para poder navegar correctamente dentro de entornos parcialmente conocidos, puesto que la realimentación sensorial posibilita la reacción del vehículo ante situaciones inesperadas.

- Sistema de control. Es el encargado de generar las consignas que el sistema de actuación aplicará para llevar a cabo la tarea indicada. Las consignas producidas dependen, por una parte, de la misión a realizar, y por otra, del conocimiento del entorno y del estado del robot proporcionado por el sistema sensorial. El sistema de control no tiene que localizarse forzosamente en el propio robot móvil, sino que es posible situarlo e incluso distribuirlo en varias plataformas, siempre que exista un sistema gestor de comunicaciones que lo soporte.

- Sistema de comunicaciones. Transfiere la información entre los diferentes elementos del robot, incluyendo la que se establece entre el robot y el sistema que genera las tareas a ejecutar.

- Sistema de alimentación. Proporciona la energía necesaria para el funcionamiento de los sistemas descritos. De él depende, en gran medida, la autonomía del robot móvil.

Estos sistemas, así como las relaciones e interacciones entre los mismos, se esquematizan en la Figura A.5.

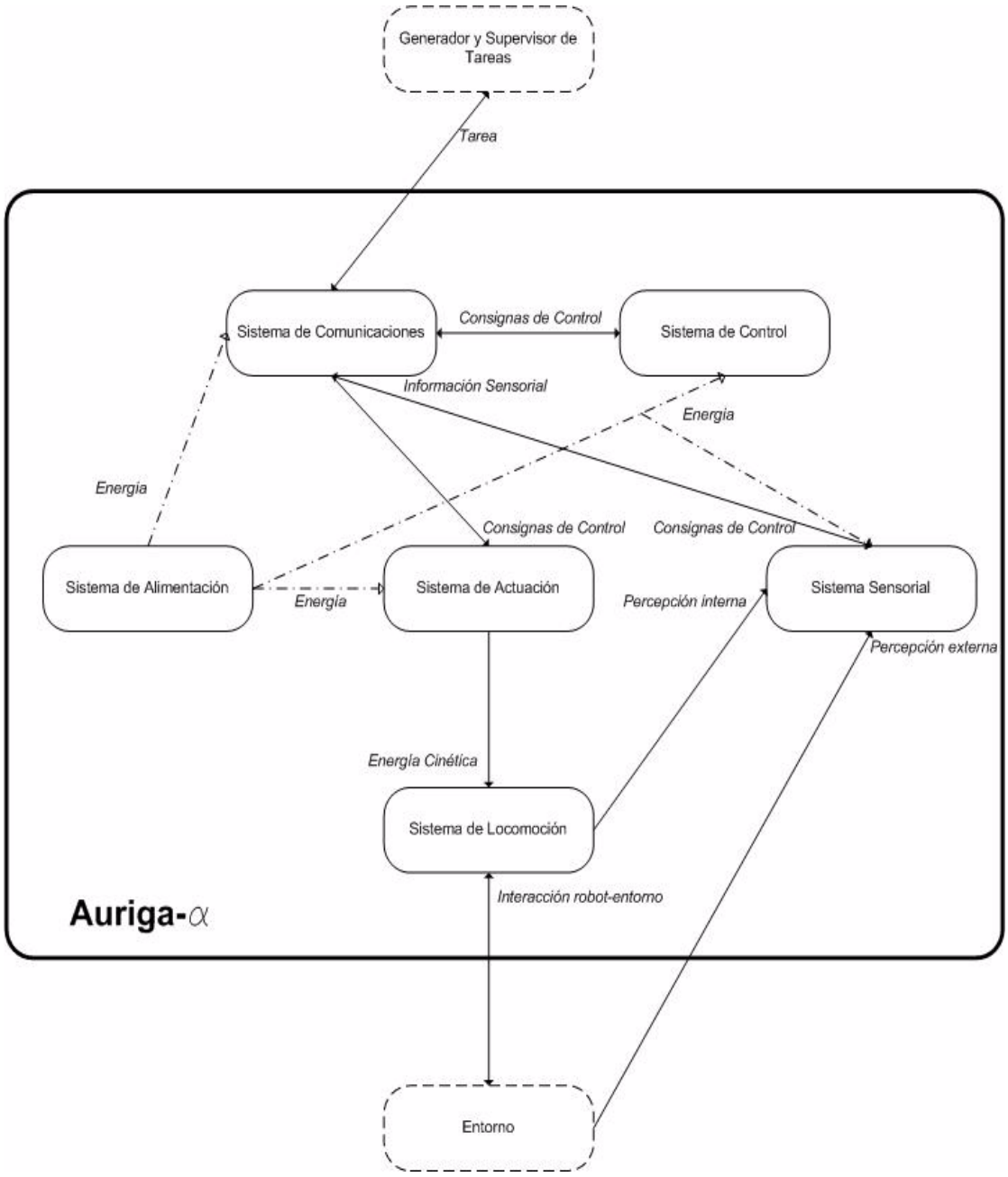


Figura A.5. Sistemas básicos de Auriga-α

Las siguientes subsecciones se centran más profundamente en las características particulares de los diferentes sistemas que componen Auriga-α.

A.2.1. Sistema de actuación

El sistema de actuación recibe las consignas generadas por el sistema de control, para poder llevar a término la misión encomendada al robot en su entorno particular. El sistema de actuación transforma tales consignas en órdenes entendibles por el sistema de locomoción, y así el vehículo responde de manera adecuada.

A.2.2. Sistema de locomoción

El sistema de locomoción de Auriga-α consta de dos cadenas de caucho sobre las que este vehículo se apoya, desplazándose gracias al movimiento de las mismas y su rozamiento con el suelo.

El sistema de tracción de las cadenas original, formado por un motor de gasolina y mecanismo de giro por embrague-freno, se ha sustituido por un motor eléctrico, reductora planetaria y transmisión por cadena metálica, para cada uno de los ejes que actúan sobre tales cadenas.

A.2.3. Sistema de alimentación

El robot garantiza la alimentación de todo su equipamiento a partir de dos fuentes diferentes pero equivalentes entre sí:

- un grupo electrógeno compuesto por un motor de gasolina de dos tiempos de 220 cc. acoplado a un generador trifásico de 400 V y 4 Kw de potencia. Este sistema asegura el funcionamiento autónomo del vehículo durante varias horas ininterrumpidas.
- conexión a red fija trifásica de 400 V a través de una clavija de potencia de 16 A, situada en la parte posterior del vehículo. Así se permite el funcionamiento en entornos cerrados y de prueba que no requieran un funcionamiento autónomo del robot, como pueden ser laboratorios de investigación; de esta manera, se evita la contaminación acústica y ambiental asociada al funcionamiento del grupo electrógeno.

Un cuadro eléctrico, sito en la parte trasera del robot y accesible mediante una puerta en la carcasa de poliéster, facilita la operación de selección de la fuente de alimentación, y además aloja los dispositivos de protección de la instalación, el generador, el transformador y las etapas de potencia, junto con los elementos de control de la potencia a los motores y el accionamiento del freno eléctrico. Las etapas de potencia de los motores disponen de una salida de tensión (± 10 V) proporcional a la intensidad consumida por el motor al que alimentan.

A.2.4. Sistema sensorial

Los sensores que componen el sistema sensorial pueden dividirse en dos grupos, según el tipo de percepción que recojan. Así, los sensores internos proveen información sobre el estado interno del robot, mientras que los sensores externos reúnen información acerca del entorno del vehículo.

A.2.4.1. Sistema sensorial interno

Auriga-α dispone de los siguientes sensores internos:

- Un codificador óptico incremental de 2000 cuentas por revolución.
- Dos conjuntos de acelerómetros, dispuestos a ambos lados del robot de forma simétrica con respecto al centro de gravedad del vehículo, y que permiten obtener la aceleración de ese punto del robot.
- Un sistema giroscópico para la medición de las velocidades angulares del vehículo según las tres direcciones del espacio.

A.2.4.2. Sistema sensorial externo

La función del sistema sensorial externo consiste en recolectar información sobre el estado del entorno del robot, de manera que se puedan detectar obstáculos y situaciones de riesgo, y además calcular su posición. Los elementos que componen este sistema son:

- Una brújula electrónica que obtiene la orientación del robot respecto al eje magnético de la tierra. Este sensor se ha situado sobre un mástil sobre el cuerpo del robot.

- Un sistema de cámaras integrado por dos cámaras CCD color, que también se ha colocado en un mástil para una mejor percepción del entorno ante el vehículo.

- Un telémetro láser instalado en la parte delantera inferior del vehículo, que genera un mapa bidimensional del entorno en un rango de 180º; el modelo elegido es el LMS-200, de Sick-Optic Electronic (Sick-Optic Electronic, 1985). Este dispositivo es el que mayor información proporciona para la estrategia de navegación implantada sobre el vehículo. Además, los escáner láser ofrecen muy buenas prestaciones para su aplicación a robots móviles, en aspectos como precisión, fiabilidad o tiempo de procesamiento. Su precio, hasta hace poco elevado en comparación con otros sensores, resulta cada vez más asequible gracias a la aparición en el mercado de dispositivos semicondutores comerciales. Desgraciadamente, estos sensores también presentan limitaciones, siendo la más destacable su incapacidad para detectar superficies de cristal, que permanecen invisibles para él.

A.2.5. Sistema de control

El sistema de control de Auriga-α está formado, en realidad, por dos sistemas de computación con características y tareas claramente diferenciadas. Por un lado, un sistema basado en DSP que realiza el control de bajo nivel y la monitorización de los sensores externos y los internos más simples; por otro lado, un sistema basado en PC encargado de la planificación, el control de alto nivel y el tratamiento de la información de los sensores más complejos, tales como el sensor láser. Además, podrían emplearse sistema de control externos, conectados al vehículo mediante el sistema de comunicaciones.

A.2.5.1. Sistema basado en DSP

El sistema basado en DSP es un sistema modular constituido por los siguientes elementos:

- una placa madre con un DSP TMS320C31 de Texas Instruments.

- una placa de memoria ROM de programa de 32 kpalabras de capacidad.

- una placa de entrada-salida analógica, con dos canales de entrada y dos canales de salida de 32 bits.

- una placa de control de motores de dos ejes.

- un programa que realiza el control de bajo nivel y la adquisición de sensores analógicos.

El DSP, que se inicializa con la puesta en marcha del robot, presenta dos modos de funcionamiento: manual y automático.

Cuando el DSP trabaja en modo manual, las consignas de actuación se introducen a través de una palanca de mandos digital. Así, en base a los comandos de incremento de velocidad lineal y angular deseadas, el DSP envía las consignas apropiadas al sistema de actuación. El manejo de la palanca directamente desde el sistema DSP permite un control más directo y fiable, sin necesidad de usar el sistema basado en PC. En cualquier momento, el operador puede tomar el control del robot mediante una secuencia de movimientos en la palanca de mandos que pasan el DSP a modo manual.

En el modo automático de control de DSP, las consignas de actuación son recibidas del sistema basado en PC. En este modo de funcionamiento, si no hay comunicación entre el PC y el DSP en un intervalo de dos segundos, se considera que se ha producido un fallo grave en el sistema.

A.2.5.2. Sistema basado en PC

El sistema basado en PC se dedica a las tareas de alto nivel, así como a la interpretación de la información recibida por el escáner láser. Consta de dos computadores PC industriales, denominados PC1 y PC2, con un hardware similar: Pentium a 133 MHz, 32 Mb de RAM, disco duro de 1.28 Gb, y tarjeta de red Ethernet que permite interconectarlos en red a través de un concentrador. Además, PC1 incorpora una tarjeta de entrada-salida digital ACL-175, y una tarjeta de expansión de cuatro puertos serie para la conexión de nuevos dispositivos.

El sistema operativo instalado en ambos PCs es LynxOS (Lyns Real-Time Systems, 1993), un sistema tipo Unix de tiempo real que cumple la normativa POSIX. Sobre él se ha construido una arquitectura de control utilizando el sistema de desarrollo de software robótico NEXUS. Tanto NEXUS, como la arquitectura implantada y los elementos añadidos para la planificación de velocidades se estudian con detalle en el Apéndice C.

A.2.6. Sistema de comunicaciones

El sistema de comunicaciones de Auriga-α se ha implantado a dos niveles, acordes con los dos niveles del sistema de control. Así, existe un sistema de comunicaciones que comunica los PCs con el exterior, y un sistema de comunicaciones interno asociado al DSP. Esta concepción de comunicación interna del robot y externa con el resto del sistema facilita la escalabilidad del sistema de control, y habilita una posible distribución del mismo, de manera que se realice parte del control del vehículo en un sistema externo.

Entrando en un poco más de detalle, los PCs constituyen una red de área local que se conecta con el exterior a través de Ethernet radio. Por su parte, el DSP se comunica con ellos a través de puerto serie, mediante un protocolo de comunicación diseñado especialmente para el robot (Fernández y Pedraza, 1998). Se trata de un protocolo de tipo maestro-esclavo, en el que el PC siempre lleva el control de la comunicación.

Apéndice B. Entorno de Simulación Multirrobot SIMUNA

B.1 Introducción

Si en cualquier área científica la experimentación en entornos reales tiene un papel fundamental, en el campo de la Robótica cobra una especial relevancia, ya que la realidad es el mejor muestrario de todas las posibles situaciones a las que debe enfrentarse un prototipo. A pesar de ello, la simulación robótica también resulta importante, puesto que permite el desarrollo de los estadios iniciales de un sistema de forma fácil, rápida y barata (Balch, 1998b). En el caso de los sistemas multirrobot, que son especialmente complejos y exigentes a la hora de realizar experimentos reales, el hecho de tener una herramienta de simulación alivia en gran medida el esfuerzo de diseño e implantación.

Son varias las herramientas de simulación para robots móviles -tanto comerciales como *shareware* y *freeware*- que pueden encontrarse actualmente: Webots (K-Team, 2002), Kephera Simulator (Kephera Simulator HomePage, 1997), JavaBots/TeamBots (Balch and Ram, 1998), Rossum's Playhouse RP1 (Rossum's Playhouse, 2002), Mission Lab (Mission Lab, 2002) Player/Stage (Player/Stage, 2003)... Estas soluciones estándar, aunque normalmente potentes, completas y probadas, no siempre se ajustan a las características de sistemas distintos de los que constituyeron el origen de su diseño: sólo pueden aplicarse a determinados robots, o únicamente admiten cierto lenguaje de programación, deben correr bajo un sistema operativo o un procesador específico o, si están preparadas para un único robot, es necesaria una adaptación al caso multirrobot.

Por ello, en este apéndice se presenta SIMUNA (SIstema MUltirrobot de Navegación Autónoma), un conjunto simple de clases de objetos que pueden ser usadas como la base de diseños de simulación más complejos, adaptables a cualquier sistema multirrobot. Su mayor aportación radica en su flexibilidad, ya que permite integrar diferentes elementos cuya implantación dependa del sistema multirrobot a simular; asimismo, se trata de una solución no propietaria, por lo que no está atada a ningún lenguaje de programación o plataforma hardware o software. Además de ser el entorno sobre el que se han realizado parte de las simulaciones necesarias para el desarrollo de esta tesis, SIMUNA ha dado lugar a dos publicaciones en congresos internacionales (Cruz *et al*., 2002a; Cruz *et al*., 2002b).

El apéndice se estructura como sigue. Los apartados segundo y tercero analizan, respectivamente, dos cuestiones importantes del diseño realizado: la utilización de la programación orientada a objetos para el desarrollo de la solución propuesta, y el tratamiento dado a las cuestiones de concurrencia que surgen en cualquier sistema multirrobot. La sección 2.4 se centra en la estructura y componentes principales de SIMUNA. Los experimentos y resultados obtenidos mediante la implantación de SIMUNA en MATLAB se muestran en el apartado 2.5. Finalmente, las conclusiones que pueden extraerse de la herramienta presentada se razonan en la sexta sección.

B.2 Enfoque Orientado a Objetos

Las fases de análisis y diseño de SIMUNA se han llevado a cabo bajo el conocido paradigma orientado a objetos, con el objetivo de lograr un software de calidad. La calidad del software puede evaluarse mediante cinco criterios principales: *corrección*, *robustez*, *extendibilidad*, *reusabilidad* y *compatibilidad* (Meyer, 1998). Tales criterios se explican brevemente a continuación:

- corrección y robustez: aseguran que el software desarrollado verifica su especificación previa (corrección), y que su funcionalidad queda garantizada en situaciones no previstas en tales requerimientos (robustez). Ambos términos se denominan a veces de manera conjunta como *fiabilidad*.
- extendibilidad: software extensible es aquél que puede adecuarse fácilmente a cambios en su especificación.
- reusabilidad: permite que el código pueda utilizarse en aplicaciones distintas a la proyectada inicialmente.
- compatibilidad: el software es compatible si puede integrarse para funcionar con otros programas.

La programación orientada a objetos proporciona técnicas que permiten lograr código extensible, reusable y compatible. Sin embargo, la corrección y la robustez dependen además de la metodología empleada en las etapas de análisis y diseño. Una herramienta eficaz en ambos pasos es el lenguaje gráfico *Unified Modelling Language* o *UML* (Fowler y Scott, 2000). UML consiste en una serie de diagramas (casos de uso, clases, secuencias,...) que recogen tanto los aspectos estáticos como los dinámicos del sistema a construir. Cada diagrama se refiere a una característica precisa, y todos ellos ofrecen una

visión global del sistema completo; en las próximas secciones se incluirán algunas de los diagramas generados durante la construcción de SIMUNA. La combinación de POO y UML conducen a un mejor desarrollo de proyectos (Cantor, 1998), una cuestión importante en el caso de sistemas complejos o que involucren a un equipo de programadores.

B.3 Concurrencia

La concurrencia es una característica inherente a los sistemas multirrobot: los robots que lo forman pueden moverse de manera simultánea. En consecuencia, a la hora de implantar un simulador multirrobot debe tenerse este hecho en cuenta. La herramienta propuesta en este trabajo gestiona la concurrencia de los sistemas multirrobot mediante un Sistema de Eventos Discretos (SED).

Un SED se define como un método de simulación diseñado para modelar sistemas con un comportamiento no determinístico debido a componentes aleatorios, en los que tanto el tiempo como el espacio de estados son discretos. Sus elementos son (Muñoz, 1998):

- *Entidades*: elementos del sistema como máquinas, piezas, clientes o, en este caso, robots. Pueden clasificarse como entidades *permanentes*, si su tiempo de vida es igual al del experimento con el modelo, o *temporales*, si se crean y se destruyen a lo largo de la simulación.
- *Atributos*: características de las entidades, que distinguen unas de otras.
- *Conjuntos*: una colección de entidades, normalmente temporales. Si se asocian a una entidad permanente, esta unión recibe el nombre de *recurso*.
- *Actividades*: representan los cambios en el sistema, por lo que puede decirse que la pauta del mismo depende de un correcto modelado de sus actividades, tanto en su forma como en su secuencia. Una actividad queda definida por un conjunto de sucesos instantáneos denominados *eventos*.
- *Estado*: valor que toman los atributos de las entidades del modelo, el cual se usa como criterio para seleccionar la próxima actividad a ser lanzada.

A la hora de simular un SED, resulta necesario incluir un *Planificador* o *Scheduler*, cuya labor será controlar la ejecución de la simulación, de acuerdo con el algoritmo indicado en la Figura B.1. En otras palabras, interviene como un reloj que marca qué debe hacerse en el instante actual, considerando que en un SED el tiempo se incrementa

arbitrariamente dependiendo de la actividad actual. Como se manifiesta en el algoritmo, cada iteración del bucle corresponde a la ocurrencia de un evento. Por tanto, es la secuencia de eventos la que decide cuándo suceden las cosas en el sistema, de lo que se desprende que, si los eventos están convenientemente ordenados (en otras palabras, si las actividades se han modelado convenientemente), se logra simular la concurrencia.

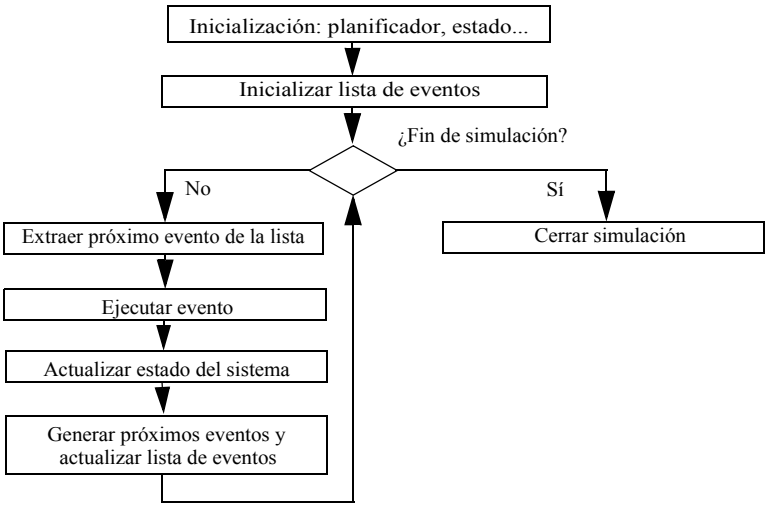


Figura B.1. Algoritmo del Planificador de un SED

B.4 Descripción de SIMUNA

En esta sección se pormenorizan algunos aspectos de la herramienta de simulación presentada. Con este objetivo, en primer lugar se ofrece una visión global de su funcionamiento; a continuación, y desde el planteamiento orientado a objetos que guía su diseño, se detallan las clases más importantes que componen SIMUNA.

B.4.1. Mecanismo de simulación

El comportamiento del simulador multirrobot se describe, mediante notación UML, en la Figura B.2. Este diagrama, denominado *Diagrama de Nivel de Contexto*, muestra los elementos más destacables del simulador y las relaciones entre ellos. Así, puede verse que SIMUNA contiene una serie de robots y un monitor, cuya evolución es controlada

por un Sistema de Eventos Discretos. Aparte de estos componentes básicos, existen otras clases menores también incluidas en el diagrama, como un generador de informes o un interfaz gráfico, cuyo papel es facilitar la obtención e interpretación de los resultados extraídos de las ejecuciones de SIMUNA.

El Diagrama de Nivel de Contexto también contiene elementos que pueden considerarse externos al simulador, pero que de alguna forma toman parte en la simulación, relacionándose con los componentes internos ya comentados. Dichos elementos son un generador de trayectorias, implantado según el método propuesto en el capítulo tercero; un manejador de ficheros, y un interfaz de usuario; por supuesto, es posible incluir otras clases, dependiendo de las necesidades del sistema multirrobot que se desee simular. Es decir, el planteamiento de SIMUNA es extensible y compatible, ya que permite al diseñador añadir elementos diferentes a los aquí propuestos.

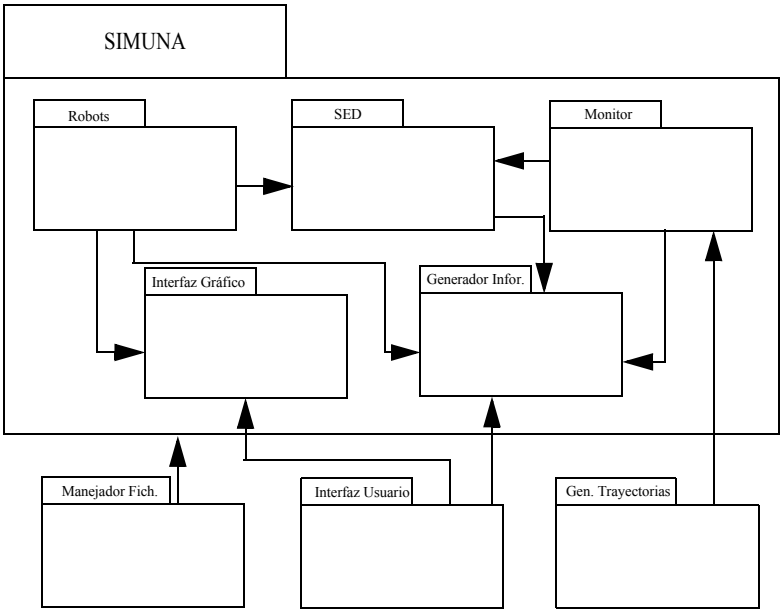


Figura B.2. Diagrama de Nivel de Contexto

El funcionamiento del simulador multirrobot es el siguiente, siempre teniendo en cuenta que la tarea que se está implantando es la navegación para robots móviles, expuesta a lo largo de esta tesis. Así, el monitor consigue una trayectoria para cada robot del sistema, asegurando por tanto que no se producen colisiones entre ellos. Después, los robots comienzan a moverse siguiendo su camino con la velocidad calculada, comunicándose entre ellos si es necesario. Si surge algún problema, cada vehículo intentará solventarlo por sus propios medios; en caso de que no sea posible, informará de la situación al monitor, que proveerá una solución. La simulación finaliza cuando todos los vehículos del sistema

alcanzan su posición final. Durante la misma, los componentes dedicados a recolectar información, como el generador de informes o el interfaz gráfico, se encontrarán activos; además, también podrán ser llamados algunos elementos externos del simulador. Como la concurrencia se gestiona a través de un Sistema de Eventos Discretos, dichos eventos deben ser generados, conforme la simulación avanza, por los propios componentes del sistema multirrobot; por ejemplo, un vehículo recibe un mensaje porque otro robot lo envió añadiendo al SED el evento *envía_mensaje*. Todo ello se resume en el *Diagrama de Clases* de la Figura B.3, en el que se representan, desde una perspectiva conceptual, las clases involucradas en el funcionamiento de SIMUNA.

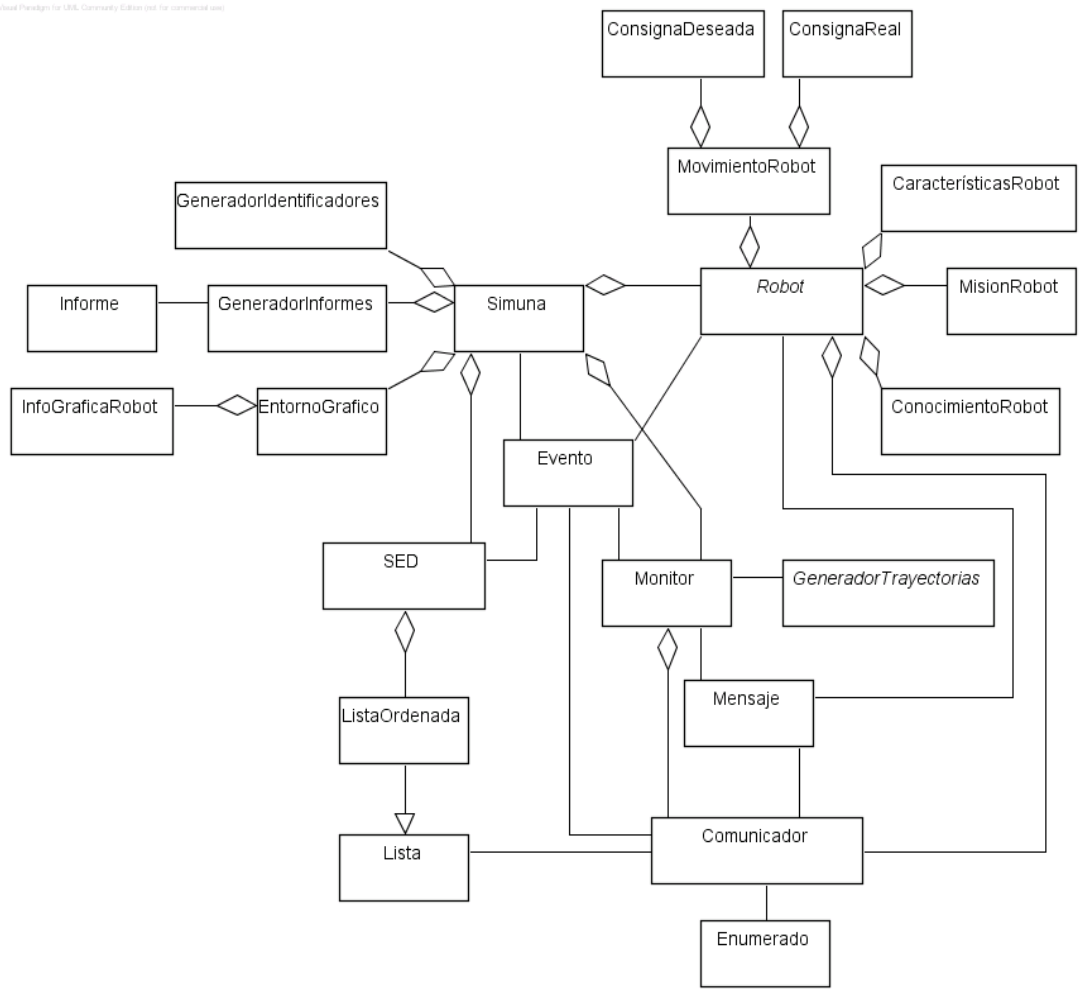


Figura B.3. Diagrama de Clases completo de SIMUNA (perspectiva conceptual)

Si el sistema multirrobot a simular tuviera que desempeñar otra tarea, sólo deberían modificarse dos puntos de toda la herramienta de simulación. Por un lado, la funcionalidad interna de los robots y/o el monitor, es decir, el código que implanta el mecanismo de navegación sería sustituido o incrementado con el software que defina el nuevo comportamiento. Por otro lado, los eventos que deban generar los componentes del sistema multirrobot serán diferentes, para poder reflejar ese cambio.

B.4.2. Clases fundamentales

Aunque como se desprende de la Figura B.3 son numerosas las clases de objetos que conforman SIMUNA, en esta subsección sólo se hará mención a aquéllas que componen su núcleo y que son determinantes para el funcionamiento de la herramienta. Estas clases destacadas se detallan a continuación:

- *SIMUNA*. Esta es la clase que alberga el sistema multirrobot en sí; puede entenderse como la representación del entorno físico en el que evoluciona el sistema. Por ello, contiene uno o varios objetos *Robot*, un objeto *Monitor*, y otros objetos recolectores de información como un *GeneradorInformes* o *InterfazGráfico*. Por último, puesto que la simulación está guiada por un Sistema de Eventos Discretos, también se incluye un objeto *SED* que controle toda la ejecución. La relación entre todos estos objetos queda explicada en el diagrama de nivel de contexto de la Figura B.2.

- *Monitor*. El objeto *Monitor* incluido en *SIMUNA* debe solucionar los conflictos que se den a raíz de la interacción entre los robots del sistema. Primero debe solicitar las trayectorias de los diferentes vehículos a un Generador de Trayectorias, para después enviar cada una de ellas al destinatario correspondiente; tras esto, queda a la espera de recibir los mensajes de finalización de tarea de los elementos del sistema multirrobot. Las trayectorias de los robots se calculan de manera que las colisiones entre ellos se eviten, pero a pesar de ello, la ausencia de choques no puede garantizarse completamente, ya que ni la información empleada para generar dichas trayectorias es siempre exacta, ni los vehículos las siguen con total exactitud. Por tanto, es posible que se produzcan colisiones entre robots según éstos se mueven en el entorno. En principio, cada vehículo del sistema debería ser capaz de analizar y resolver situaciones inesperadas y potencialmente peligrosas que puedan acontecer; si no fuera así, solicitaría ayuda al *Monitor*, que calcularía la respuesta apropiada para el vehículo en

cuestión, vigilando también su influencia en el comportamiento global del sistema. De esto se deduce que el grado de intervención del *Monitor* en el sistema multirrobot es directamente proporcional al grado de autonomía de los componentes del mismo.

- *SED*. Clase correspondiente al Sistema de Eventos Discretos que, como se ha explicado a lo largo de este apéndice, actúa como la espina dorsal del mecanismo de simulación. Contiene dos propiedades: una *lista_de_eventos*, ordenada respecto al tiempo de los objetos de tipo *Evento* que pertenecen a dicha lista, y un *tiempo*, que señala el tiempo actual de Planificador del SED. Las operaciones asociadas a esta clase controlan cómo el Sistema de Eventos Discreto va avanzando en el tiempo, y cómo finaliza, de manera que la simulación se lance, dirija y termine de manera correcta.

- *Evento*. Según se expuso anteriormente, los eventos son la traducción de las sucesos que surgen a lo largo de la vida del sistema multirrobot físico a la herramienta de simulación. Se han implementado como una clase con cuatro propiedades: *autor*, el elemento del sistema que origina el evento; *acción*, que indica qué clase de acción representa el evento; *tiempo*, un marcador del instante en el que el evento ocurre en el sistema; y por último, *info*, propiedad que contiene información adicional sobre el evento y que puede ser de interés en ciertos casos. Son cuatro los eventos que se han definido en la implantación de SIMUNA presentada en este apéndice, y se recogen en la tabla B.1.

Evento	Acción	Generado por...	Analizado por...
llegada_mensaje	Se ha recibido un mensaje	Comunicador	Robot, Monitor
nueva_consigna	El robot lee una nueva consigna de su trayectoria	Robot	Robot
sigue_consigna	El robot debe seguir la última referencia leída	Robot	Robot
preparar_robots	El monitor obtiene las trayectorias de los robots y enviárselas.	Simuna	Monitor

Tabla B.1: Eventos definidos en la implantación de SIMUNA

- *Robot*. Clase que representa cualquier robot del sistema y sus particularidades. Debe existir una instancia de la clase por cada robot que intervenga en el sistema multirrobot; si éste se encontrase formado por diferentes clases de vehículos, la clase *Robot* debería ser abstracta, de forma que las diferencias entre robots pudieran considerarse mediante herencia. El diseño de esta clase depende fuertemente del sistema multirrobot concreto que se desee simular.

- *Comunicador*. Puesto que algunos elementos del sistema multirrobot, como los vehículos o el monitor, necesitan intercambiar mensajes en ciertos instantes, es necesario que las comunicaciones entre los mismos también se simulen, para lo que se ha diseñado una clase específica; así, un objeto *Comunicador* permite el envío y la recepción de mensajes. Las propiedades incluidas en esta clase son un valor de *time_out*, un *estado*, y una *lista_de_mensajes* que encola los mensajes recibidos; como métodos, ofrece una operación *escribe_mensaje*, que introduce un mensaje en la *lista_de_mensajes*, y un método *lee_mensaje*, que extrae un mensaje de la misma.

Todo esto se representa gráficamente en el Diagrama de Clases de la Figura B.4, donde las clases expuestas en la subsección aparecen detalladas, desde una perspectiva ya de implantación, incluyendo sus atributos y métodos.

Figura B.4. Diagrama de Clases del núcleo de SIMUNA (perspectiva de implantación)

B.5 Experimentos

Una vez explicada, desde el punto de vista teórico, la herramienta de simulación SIMUNA, se aborda en esta sección su implantación en un lenguaje de programación. Para la codificación, realizada en un PC a 1 GHz bajo Windows 2000, se ha elegido el software de computación matemática MATLAB 6 R12, ya que permite obtener resultados de forma relativamente rápida y fácil, ofreciendo además buenas utilidades gráficas. Sin embargo, cualquier otro lenguaje orientado a objetos (Java, C++,...), en cualquier otra plataforma (Linux, Windows 9x, Lynx,...) sería factible.

Para facilitar la comprensión del funcionamiento de SIMUNA se ha escogido un sencillo experimento, en el que dos robots móviles (RAM-2 y Auriga-α, ambos desarrollados en el Departamento de Ingeniería de Sistemas y Automática de la Universidad de Málaga), navegan simultáneamente en un espacio de trabajo común. La Figura B.5 muestra la situación inicial, en la que los dos vehículos se encuentran en sus posiciones de comienzo; RAM-2 se dibuja en gris oscuro, Auriga-α en un tono más claro de gris, y de igual manera se representan los caminos que deben seguir. Tales caminos se cruzan en dos puntos; puesto que la planificación espacial no evita las colisiones, el perfil de velocidad deberá asegurar que no se producen choques. El movimiento de los robots a lo largo de sus respectivos caminos, eludiendo las situaciones de riesgo, se esboza en las Figuras B.6 y B.7. Por último, en la Figura B.8 los vehículos alcanzan su objetivo.

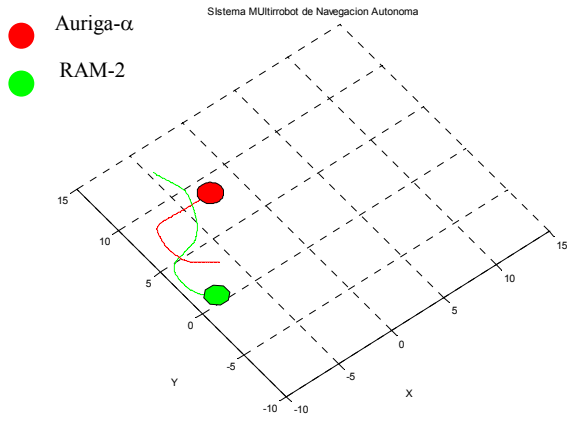


Figura B.5. Situación Inicial

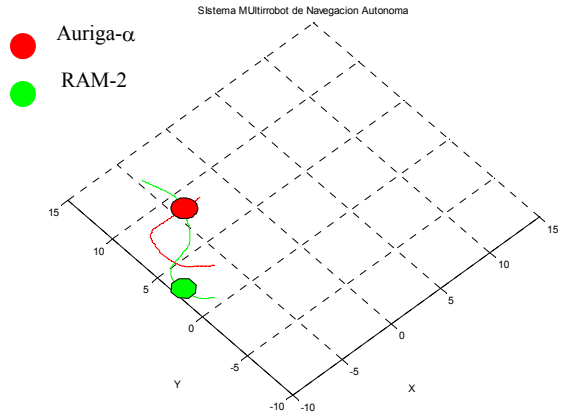


Figura B.6. Evitación del primer cruce

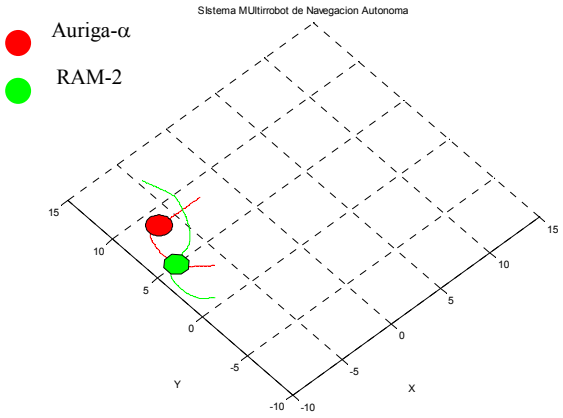


Figura B.7. Evitación del segundo cruce

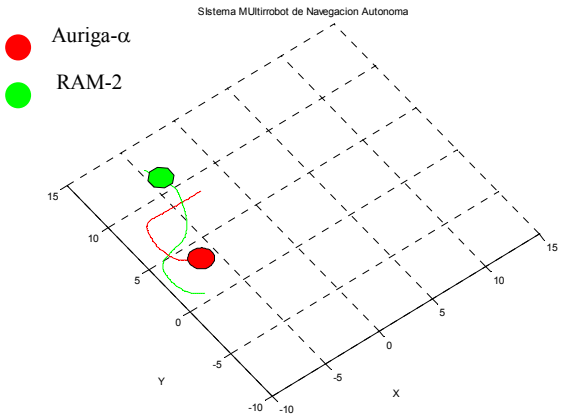


Figura B.8. Situación final

Es ahora, en el momento de la implantación, cuando los aspectos clave relativos al sistema multirrobot físico que se simula y que no pueden considerarse dentro del esquema general de SIMUNA, deben estudiarse e incluirse en el simulador. Primeramente será necesario analizar qué eventos deben insertarse en el SED, y qué elementos del sistema serán los encargados de lanzarlos; después puede comenzarse la codificación de las clases. En este ejemplo la navegación es la tarea a realizar, por lo que los robots del sistema multirrobot deben seguir la trayectoria previamente calculada: el sistema de control de movimiento recibe la información extraída de la trayectoria, y dicha referencia es seguida. Por otra parte, cada referencia debe mantenerse un cierto tiempo, para así garantizar que el control de movimiento del vehículo es capaz de alcanzarla. Siempre que una de estas situaciones se produzca, un objeto *Robot* generará un *Evento* (moverse a una nueva referencia, o seguir una ya obtenida); estos eventos se añadirán al SED, pasando a formar parte de la simulación. El SED seguirá corriendo hasta que no se produzcan nuevos eventos, lo que significa que ambos robots han llegado al final de sus caminos.

B.6. Conclusiones

En este apéndice se ha presentado SIMUNA, una herramienta de simulación multirrobot general, flexible, simple y no propietaria, diseñada con técnicas orientadas a objeto y que puede ejecutarse en cualquier plataforma. Para lograr la concurrencia propia de los entornos multirrobot se ha empleado un Sistema de Eventos Discretos, por lo que puede utilizarse en equipos no multitarea. SIMUNA ha sido creada para constituir la base de desarrollos complejos, que puedan adaptarse lo más posible al sistema físico a simular; su utilidad se muestra especialmente en aquellos casos en los que desarrollar un simulador para un problema concreto es menos costoso que adaptar una herramienta ya creada.

Apéndice C. Arquitectura Robótica Implantada sobre NEXUS

En este apéndice se describe el sistema de integración de software robótico NEXUS, desarrollado en su totalidad en el Departamento de Ingeniería de Sistemas y Automática de la Universidad de Málaga (Fernández Madrigal y González, 1999), así como la evolución que lo ha transformado en el sistema de desarrollo BABEL. A continuación, una vez conocida la base que sustenta la construcción de arquitecturas de control robótico, se comenta la arquitectura concreta para el robot Auriga-α diseñada e implantada sobre NEXUS (Fernández Ramos, 2001). Por último, se describe el módulo ICE encargado de planificar las velocidades para un único robot, según el algoritmo presentado en esta tesis (Cruz, 1999). El apéndice se cierra con las conclusiones más destacables de lo presentado a lo largo del mismo.

C.1. Sistema de integración software NEXUS

Los sistemas multirrobot constituyen un campo multidisciplinar, en el que para gobernar el comportamiento de sus elementos es necesaria la integración de aplicaciones desarrolladas, en la mayoría de los casos, por un grupo heterogéneo de personas, y relativas a áreas tan dispares como control, visión, inteligencia artificial, teleoperación, comunicaciones, etc. Habitualmente, estas aplicaciones se encuentran sujetas tanto al hardware concreto de los vehículos móviles que componen el sistema multirrobot, como al sistema operativo que permite su gestión; sin embargo, existe mayor flexibilidad al elegir el software en el que cada una de ellas se programa, lo que puede suponer un problema a la hora de unirlas para lograr el correcto funcionamiento del sistema multirrobot. Así, resultaría beneficioso el empleo de un sistema de integración de software abierto que permita la interacción de todas las aplicaciones. Cualquier sistema de este tipo debería ofrecer las siguientes características:

- Ser de propósito general, tanto software como hardware. Así, desde la perspectiva del software, se permitiría la integración del mayor número de áreas posibles para establecer arquitecturas de control robóticas más ricas y completas. Desde el punto de vista hardware, se extendería la posibilidad de trabajar con diferentes tipos de vehículos.

- Capacidad de distribución entre varias computadores, para repartir la carga de procesamiento y lograr el máximo rendimiento.
- Proporcionar un interfaz sencillo y amplio, que permita a los desarrolladores del sistema incorporar sus programas cómodamente a la arquitectura general, así como utilizar aplicaciones ya integradas en ella.
- Garantizar la protección entre los programas conectados y el sistema principal, para evitar conflictos que provoquen comportamientos no deseados.
- Aprovechar las ventajas del sistema operativo escogido para la implantación. En este caso, se trata del sistema de tiempo real LynxOS, basado en Unix.

Dentro del proyecto de investigación "Sistema de navegación autónoma y operación para un robot móvil equipado con un manipulador" (TAP95-0775-E), llevado a cabo en el Departamento de Ingeniería de Sistemas y Automática de la Universidad de Málaga, se ha diseñado e implantado la extensión de sistemas operativos de tiempo real NEXUS. NEXUS es un sistema de desarrollo e integración de aplicaciones de control para robots móviles que conforma una base para labores de investigación de más alto nivel, como puedan ser los sistemas multirrobot. Las propias siglas que componen el nombre de NEXUS[9] explican su filosofía:

- **Not-centralized**. NEXUS se divide en dos capas independientes: una de ellas se encarga del mantenimiento de la estructura interna, mientras que la otra se dedica a la implantación de la arquitectura de control de robot (entendiendo como arquitectura la *aplicación* que se ejecuta sobre NEXUS). En principio, no existe jerarquía alguna entre los gestores de NEXUS, ni tampoco entre los componentes de la aplicación, aunque ésta puede diseñar un sistema jerárquico de módulos si así lo desea.
- **EXperimental**. Con NEXUS, es posible utilizar los conceptos y técnicas más relevantes en el campo de las arquitecturas de control de robots móviles, tales como teleoperación, arquitecturas híbridas, árboles de tareas, ejecución en tiempo real, recuperación de errores... Es decir, permite la implantación de muchas y muy diferentes experiencias.
- **Upgradeable**. Pueden añadirse módulos fácilmente, incrementando así la funcionalidad del sistema con nuevos servicios.

9. NEXUS es un acrónimo: Not-centralized, EXperimental, Upgradeable, Static-binded.

- **Static-binded**. Cada módulo de la aplicación debe conocer la existencia de los restantes módulos previamente a su conexión a la arquitectura. Es decir, la asignación de servicios a peticiones de servicios se efectúa en tiempo de compilación de los módulos (*ligaduras estáticas*), y no en tiempo de ejecución.

La Figura C.1 muestra un esquema de NEXUS, con sus principales componentes. En él se observa la división en dos subsistemas: el *subsistema dependiente de la tarea*, y el *subsistema de administración*. Ambos se desglosan en los dos subapartados siguientes.

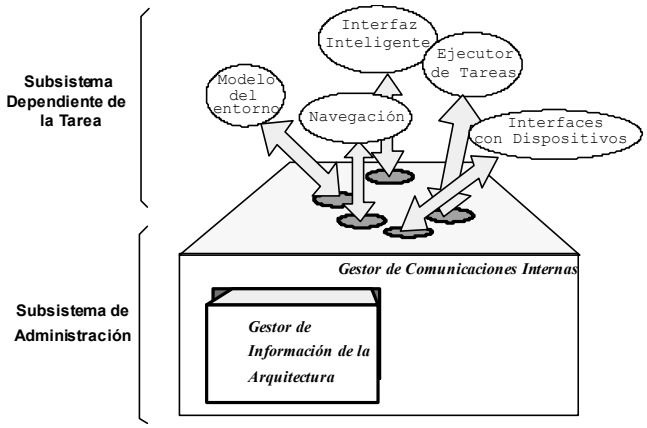


Figura C.1. Principales componentes de NEXUS

C.1.1. Subsistema dependiente de la tarea

El subsistema dependiente de la tarea define la arquitectura de control o aplicación como tal. Se divide en varias *Unidades Conceptuales* (UC), que son agrupaciones de programas que se ocupan de labores relacionadas entre sí; por ejemplo, las de percepción, las de ejecución de tareas, etc. No hay ninguna UC más privilegiada que otra, por lo que puede considerarse que NEXUS es absolutamente distribuido. Cada UC es, para las demás, una caja negra que ofrece un conjunto de servicios; su esquema interno se presenta en la Figura C.2.

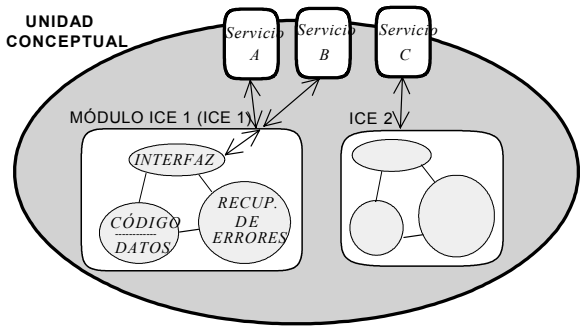


Figura C.2. Estructura interna de una Unidad Conceptual genérica

Los módulos o programas de usuario pertenecientes a las UC se denominan *módulos ICE* (de *Interface-Code-Error recovery*), y proporcionan ciertos servicios al resto de la arquitectura. Por ejemplo, un módulo ICE llamado "Grafos-AH", conectado a la UC "Modelado del Mundo", provee los servicios necesarios para modelar el mundo del robot mediante grafos jerárquicos anotados, incluyendo servicios de búsqueda de caminos entre dos puntos, de anotación sobre lugares u objetos del mundo, etc. Con el fin de facilitar su manejo, los servicios se agrupan en *grupos de servicios*; por defecto, toda UC contiene un grupo de "Servicios Misceláneos", que engloba todos aquellos servicios que no pueden asignarse a un grupo concreto.

Respecto a los módulos ICE, la separación en tres partes de la Interfaz, el Código y la Recuperación de Errores no es arbitraria. El Interfaz se ocupa de las comunicaciones con la UC a la que pertenezca el módulo, del tratamiento de las peticiones de servicios que vayan llegando, y de las cuestiones relativas a la conexión, desconexión, inicialización y finalización del módulo. Todas estas tareas se facilitan al usuario en una librería ya compilada, de forma que sólo es necesario enlazar la parte denominada código del ICE con dicha librería para realizar la mayor parte del trabajo de conexión del ICE a la arquitectura. El Recuperador de Errores comprende las rutinas necesarias para solventar los errores cometidos en la ejecución de algún servicio, o para propagarlos al que realizó la petición del servicio en caso de que el error sea irrecuperable. De todo lo expuesto se deduce que el usuario únicamente debe implantar el Código del ICE (los servicios que definen la funcionalidad de su programa) y el Recuperador de Errores, ya que todo lo demás corre a cuenta de NEXUS.

C.1.2. Subsistema de administración

El subsistema de administración es el más próximo al hardware y al sistema operativo, encargándose de las labores de mantenimiento de la propia estructura de NEXUS. En él además se realizan las comunicaciones entre procesos y entre computadoras, puesto que debido al carácter distribuido de NEXUS, los módulos ICE que componen una aplicación pueden ejecutarse en distintas máquinas. Para llevar a cabo todas estas tareas se dispone de dos administradores principales: el *gestor de comunicaciones internas* (GCI), y el *gestor de creación de la aplicación* (GCA).

Así, las comunicaciones entre los proceso agrupados en una UC se envían hacia el GCI de la capa inferior, el cual las dirige a sus destinos y se ocupa de enrutarlas por la red de ordenadores cuando es necesario. Por otra parte, el GCA mantiene toda la información sobre la estructura de la aplicación (datos sobre las UCs y los módulos que contienen), así como sobre la conexión de nuevos programas de usuario, y sobre el registro de servicios y grupos de servicios.

C.1.3. La evolución de NEXUS: el sistema de desarrollo BABEL

Aparte de la especificación de diseño explicada en el subapartado anterior, NEXUS incluye un paquete software (varias APIs y herramientas) que permite tanto la implantación en lenguaje C de los diseños para el sistema operativo LynxOS, como la depuración de los mismos. Además, se añade un mecanismo de comunicaciones apropiado para la distribución del software, construido sobre los protocolos TCP/IP.

Sin embargo, NEXUS presenta ciertas limitaciones a causa de la inherente heterogeneidad del software robótico. Es por ello por lo que se han extendido ampliamente las capacidades de NEXUS, que ahora se encuentra parcialmente contenido en un nuevo marco para el desarrollo de software robótico distribuido complejo: BABEL (Fernández Madrigal, 2003; Departamento de Ingeniería de Sistemas y Automática, 2004). BABEL soporta el desarrollo de software para múltiples lenguajes de programación (C, C++, Java, Lisp,...), plataformas (sistemas operativos de tiempo real, MS-Windows) y tecnologías de comunicación (TCP/IP, CORBA).

El propósito de BABEL no es ofertar un nuevo entorno único que agrupe las cuestiones de ejecución, comunicación, acceso al hardware y programación; tampoco promueve la utilización de algún otro sistema de desarrollo ya conocido. Aunque este tipo de enfoque ofrece interesantes ventajas (por ejemplo, facilita un control más estricto de los requerimientos de la especificación, mejorando así la validación, la eficiencia, etc.), tiene el inconveniente de que impone a diseñadores y programadores el aprendizaje de nuevas técnicas y paradigmas; es decir, tiende a la homogenización de los usuarios. Además, estos planteamientos homogéneos se ajustan sin problemas a proyectos robóticos pequeños, pero la experiencia en la gestión de proyectos de mayor dimensión indica que su adaptación a las diferentes necesidades que surgen en los mismos no es siempre satisfactoria. Por todo ello, BABEL permite armonizar la integración de sistemas de desarrollo existentes, de manera que formen parte de un esquema coherente; así se dota de mayor flexibilidad a la gestión y creación de un software caracterizado por la diversidad de personal y de disciplinas que lo conforman.

Los principales elementos que forman el sistema de desarrollo BABEL aparecen en la Figura C.3.

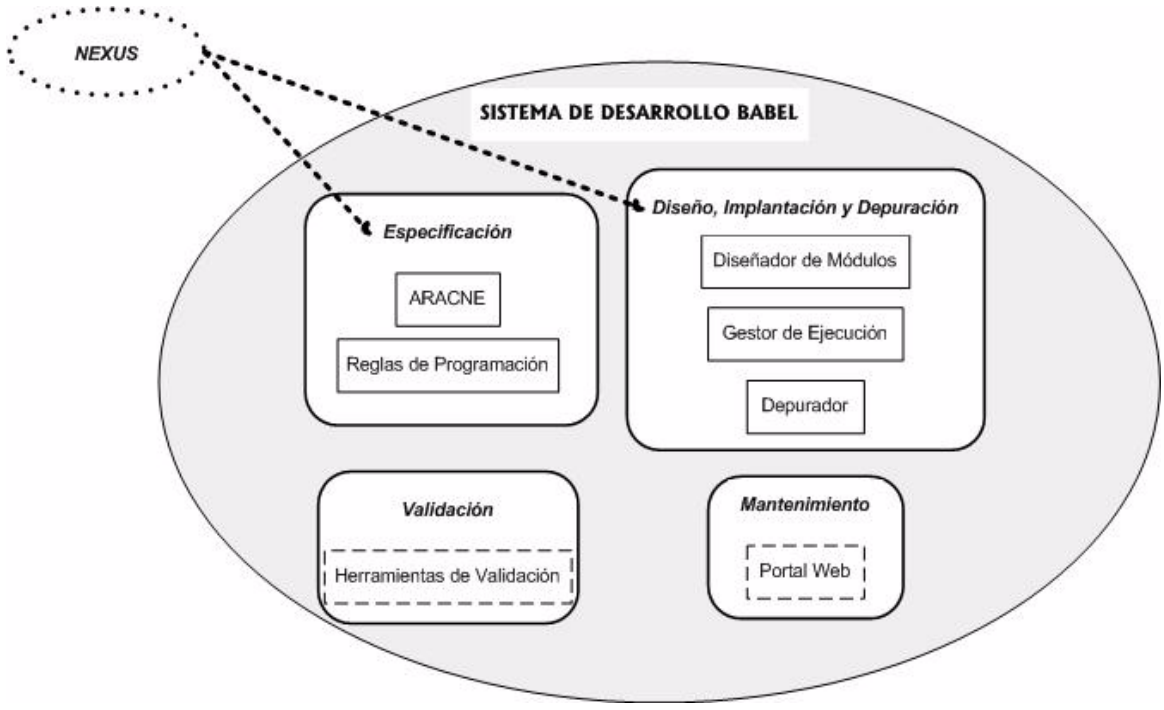


Figura C.3. Sistema de Desarrollo Babel

En ella se aprecia que ciertos elementos pertenecientes a NEXUS aún forman parte de BABEL. Así, su especificación de diseño se ha integrado en una nueva especificación de diseño de software robótico bautizada como ARACNE. ARACNE fuerza un diseño que mejora la reusabilidad, dependencia y eficiencia del código, pero manteniendo la flexibilidad para integrar programas heterogéneos; dicha flexibilidad se logra debido a la simplicidad de su estructura, así como al carácter opcional de muchos de sus parámetros. A ella se han añadido un conjunto de reglas de programación que proporcionan una guía a los nuevos programadores de módulos BABEL, puesto que los módulos se mantienen como las piezas fundamentales del software robótico generado. Por otra parte, las APIs y algunas utilidades de NEXUS siguen empleándose en las tareas de diseño, implantación y depuración de módulos, que ahora se realizan mediante herramientas visuales (*Module Designer*, *Execution Manager* y *Debugger*), lo que facilita enormemente la construcción de software. La Figura C.3 incluye también dos apartados dedicados a la validación de módulos y a su mantenimiento; ambos se están desarrollando en la actualidad, por lo que las herramientas asociadas se muestran en trazo discontinuo. La validación pretende analizar y comprobar, de manera automática, las restricciones temporales asociadas a un sistema de software robótico BABEL a partir de ciertas especificaciones de tiempo ligadas a cada uno de los módulos que la componen. El área de mantenimiento se basa en un portal web que ofrece a los programadores la posibilidad de administrar sus propios módulos, así como el acceso a la biblioteca de módulos BABEL implantados, y accesorios como news, manuales, etc.

C.2. Arquitectura de control implantada

La arquitectura de control desarrollada sobre NEXUS es de tipo híbrido, de manera que se integran aspectos característicos de las estrategias de control planificadas, junto con la ejecución de acciones simples basadas en información sensorial propias de las técnicas reactivas. La Figura C.4 muestra un esquema de la arquitectura diseñada, en la que los niveles de planificación y reacción comentados se conjugan gracias a la acción de un nivel intermedio que coordina la interacción entre ambos. La funcionalidad asociada a cada nivel la desempeñan cuatro sistemas principales:

- en el nivel de planificación, el *Sistema de Generación y Supervisión de Tareas* establece la tarea que debe realizar el robot.
- el *Sistema Gestor de Tareas*, perteneciente al nivel de coordinación, analiza tareas complejas, desglosándolas, por una parte, en comportamientos a ejecutar y, por otra parte, en situaciones ante las que cambiar de actuación.

- el nivel de ejecución comprende dos sistemas: el *Sistema de Reconocimiento de Contextos*, y el *Sistema de Ejecución de Comportamientos*. El primero identifica los diferentes escenarios en los que puede encontrarse inmerso el robot, por lo que necesita que los sensores del mismo le suministren información del entorno circundante. El segundo sistema se ocupa de generar las consignas adecuadas para los actuadores del vehículo.

Además de los tres niveles de la arquitectura híbrida (de los que se han implantado los dos últimos, suponiendo que el nivel de planificación está a cargo de un sistema externo), en la figura se incluyen los sistemas de actuadores y sensores del robot, que permiten, respectivamente, el desplazamiento del vehículo en su entorno y la recopilación de información acerca del mismo.

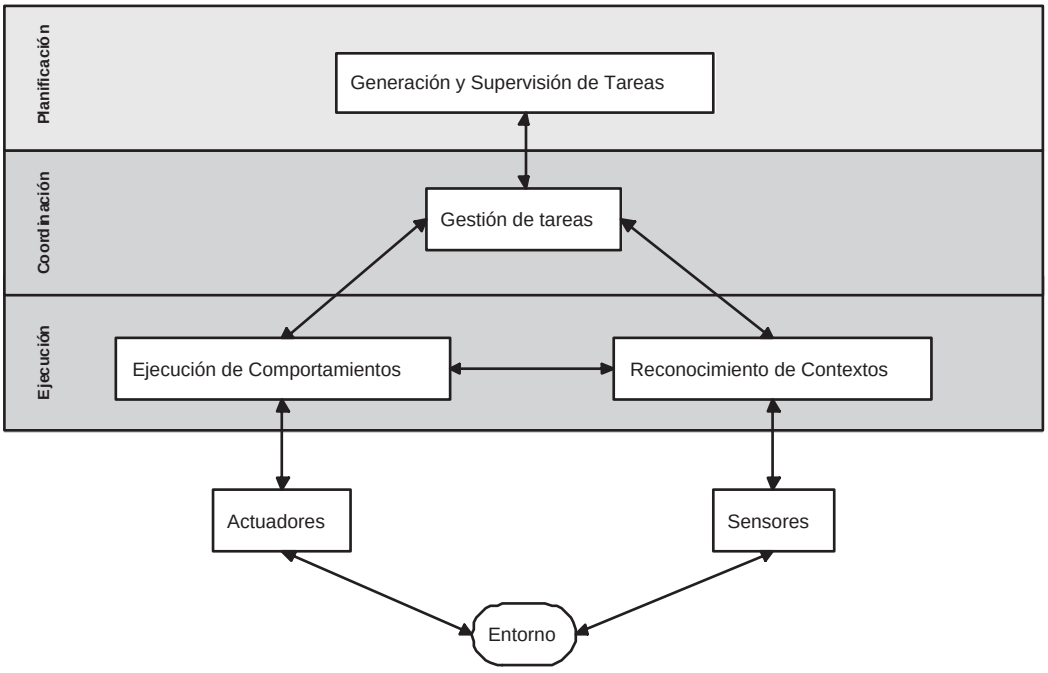


Figura C.4. Arquitectura de control implantada

Este esquema global no hace mención alguna acerca de los elementos relativos a NEXUS que se han desarrollado para lograr la implantación de la arquitectura. La Figura C.5 refina dicho esquema mostrando qué Unidades Conceptuales y módulos ICE se han creado para cada sistema concreto. Como se observa, por cada sistema existe una Unidad

Conceptual asociada que agrupa los módulos ICE necesarios para cumplir su cometido; además, se ha añadido una Unidad Conceptual dedicada a registrar los datos de ejecución vinculados a los restantes módulos, y generar así los informes oportunos para evaluar el funcionamiento de la arquitectura.

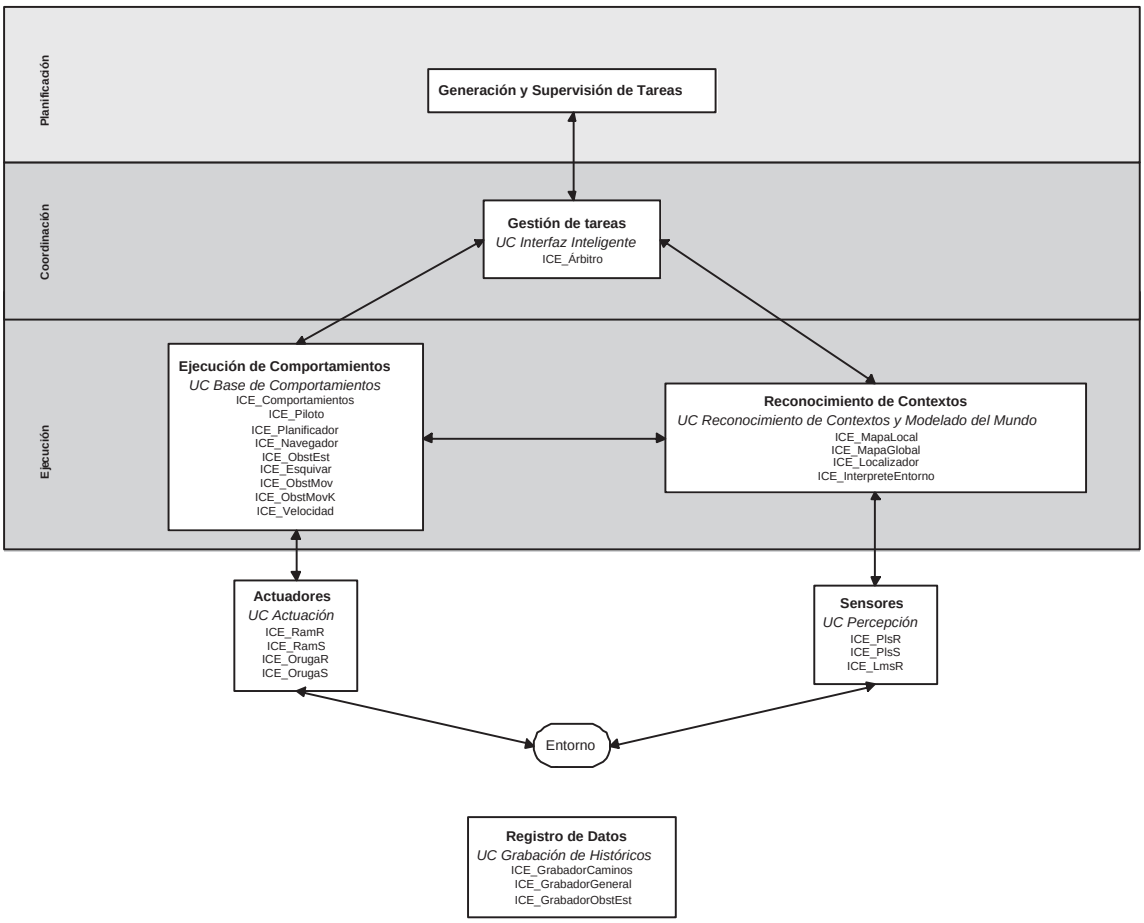


Figura C.5. Unidades Conceptuales y Módulos NEXUS implantados

C.3. Módulo ICE asociado a la planificación de velocidades

Esta sección comenta el módulo de planificación de velocidades para robots móviles *ICE_Velocidad*, desarrollado en el sistema de generación e integración de arquitecturas robóticas NEXUS. Se resume la tarea de diseño y programación realizada en la construcción del módulo, sin entrar en detalles de bajo nivel; para ese tipo de consultas, relativas a aspectos muy concretos de la implantación, se remite a la información recogida por la autora en el informe técnico (Cruz, 1999).

El módulo ICE_Velocidad se engloba, dentro de la arquitectura de control explicada en la sección anterior, en la Unidad Conceptual Base de Comportamientos. Su tarea consiste en la generación, a partir de un camino del vehículo conocido previamente, de las consignas de velocidad lineal, aceleración lineal y tiempo que deben aplicarse en cada una de las posturas que componen tal camino. Estos valores respetan las limitaciones de velocidad a las que se ve sometido el robot, tanto por sus características físicas (restricciones mecánicas, cinemáticas y dinámicas) como operacionales (restricciones debidas a la presencia de otros obstáculos móviles en el entorno de trabajo). Es decir, la planificación de velocidades calculada asegura que el robot realiza una navegación segura.

El módulo ofrece tres servicios: planificar las velocidades, consultar un error, y un entorno gráfico que facilite su manejo. El primero, *Planificar_Velocidad*, es el servicio principal, que calcula el plan de velocidades para un camino específico, devolviéndola en un fichero de trayectoria. El segundo servicio, llamado *Consultar_Error*, informa del resultado de la planificación si se desea obtener una explicación un poco más ampliada del resultado de la misma. Este servicio puede ser útil en caso de que se produzca un error al planificar la velocidad, y sea necesario averiguar qué lo produjo o cómo solventarlo. Por último, se ha implementado un servicio gráfico *Interfaz_Gráfico_Velocidad*, que permite llamar a los dos servicios anteriores cómodamente desde un interfaz de ventanas. Los tres servicios se pormenorizan en las subsecciones siguientes:

C.3.1. Servicio *Planificar_Velocidad*

Es el servicio principal del módulo, ya que es el encargado de realizar la planificación de velocidad para el robot. Para poder determinar las velocidades, el servicio debe conocer el camino sobre el cual se planifica, si existen obstáculos móviles en el entorno del robot, y en el caso de que así sea, sus tamaños y trayectorias correspondientes. Con estos datos, el servicio llama al programa planificador de velocidades -el que realmente calcula los valores de velocidad, aceleración y tiempo- y devuelve como resultado el nombre del fichero donde se almacena la trayectoria obtenida. Por tanto, a partir de un fichero de camino del robot (que sigue el formato estándar establecido, esto es, una matriz cuyas columnas representan la x, y, orientación, curvatura, distancia, velocidad lineal, aceleración lineal y tiempo de cada una de las posturas que corresponden a las filas de dicha matriz), se obtiene un fichero con el mismo formato, pero cuyas columnas correspondientes a la velocidad, aceleración y tiempo toman ahora los valores calculados por el planificador de velocidades.

C.3.2. Servicio *Consultar_Error*

Ya se ha mencionado que una de las cuestiones de mayor importancia al implementar un módulo NEXUS es el manejo de errores. La filosofía de este sistema implica que cada rutina servidora debe encargarse de gestionar el mayor número de errores posible, propagando jerárquicamente hacia niveles superiores aquellos que no pueda solventar. Si se produce un error, éste se devuelve como un código en uno de los campos de la estructura de datos de salida.

En el caso del módulo ICE_Velocidad, éste código puede resultar muy escueto cuando la planificación de velocidades no se ha realizado correctamente, y resulta conveniente conocer algo más de la causa que provocó el error, o de cómo resolverlo. Para situaciones en las que interese averiguar qué ocurrió al planificar, puede llamarse al servicio *Consultar_Error*. Este servicio proporciona, a partir del código de error que devuelve NEXUS, una cadena que contiene una explicación más detallada del resultado de la ejecución. La tabla C.1 muestra los mensajes de error que pueden aparecer tras una llamada al servicio:

Mensaje de Error
Planificación correcta
El número de argumentos para realizar la planificación es insuficiente. La sintaxis correcta es la siguiente: *camino_robot* *{tamaño_obstáculo, trayectoria_obstáculo}*
Error de apertura del fichero del camino.
Error de apertura del fichero de trayectoria del obstáculo.
El número de obstáculos móviles considerado al planificar las velocidades supera el máximo permitido por el planificador.
La longitud del fichero del camino del robot supera el máximo permitido por el planificador.
La longitud del fichero de la trayectoria del obstáculo supera el máximo permitido por el planificador.
Error al escribir el fichero de la trayectoria calculada para el robot.
El número de segmentos creados tras la división del camino supera el máximo permitido por el planificador.

Tabla C.1: Mensajes de error ofrecidos por el servicio *Consultar_Error*

Mensaje de Error
El número de cruces entre los caminos del planificador y los obstáculos móviles supera el máximo permitido por el planificador.
Existe una colisión entre el robot y algún obstáculo móvil al inicio del camino.
Imposible insertar el nuevo segmento necesario para la evitación del obstáculo móvil.
La longitud del segmento supera el máximo permitido por el planificador.
La longitud del primer subsegmento supera el máximo permitido por el planificador.
La longitud del segundo subsegmento supera el máximo permitido por el planificador.

Tabla C.1: Mensajes de error ofrecidos por el servicio *Consultar_Error*

C.3.3. Servicio *Interfaz_Gráfico_Velocidad*

El módulo planificador de velocidades puede tener dos usos: por una parte, puede incluirse dentro de una arquitectura de navegación compleja, como elemento constructor de un plan de velocidades para el vehículo. Pero, por otra parte, también puede utilizarse de forma aislada, como generador de trayectorias que se combinen después con programas de simulación, etc. Para éste último caso, se ha implementado un interfaz de ventanas sencillo de manejar para el usuario.

Así, cuando se llama al módulo ICE_Velocidad, aparece una ventana con dos botones: planificar y consultar error. El primero realiza la planificación de velocidades según los datos de entrada que se hayan introducido en la llamada al servicio, dentro del fichero *entorno_velocidad.c*. El segundo abre una ventana con el mensaje de error correspondiente a la última planificación de velocidades realizada; este punto es importante: para consultar el error, previamente es necesario planificar las velocidades.

El entorno gráfico desarrollado admite múltiples ampliaciones: mejor colocación y presentación de las ventanas, introducción de los datos mediante el interfaz, etc. Estas carencias se deben a que la idea principal a la hora de implementarlo fue obtener un modo rápido de manejo del módulo, sin prestar atención a detalles que, en ese momento, no resultaban fundamentales. Sin embargo, todos esos puntos siguen abiertos para futuras ampliaciones.

Referencias

[1] Akella S. y Hutchinson S. (2002) "Coordinating the Motions of Multiple Robots with Specified Trajectories". *Procs. of the IEEE International Conference on Robotics and Automation*, Washington DC, EE.UU., Mayo.

[2] Alami R., Fleury S., Herrb M., Ingrand F., Qutub S. (1997) "Operating a Large Fleet of Mobile Robots using the Plan-Merging Paradigm". *Procs. of the IEEE International Conference on Robotics and Automation*, Albuquerque, Nuevo Méjico, EE.UU.

[3] Alami R., Fleury S., Herrb M., Ingrand F., Robert F. (1998) "Multi-Robot Cooperation in the MARTHA Project". *IEEE Robotics and Automation Magazine*, Special issue "Robotics and Automation in Europe", Marzo.

[4] Ali K. S. (1999) "Multiagent Telerobotics: Matching Systems to Tasks". PhD Thesis. Georgia Institute of Technology.

[5] Arai T. y Ota J. (1992) "Motion Planning of Multiple Mobile Robots". *Procs. of the IEEE International Conference on Robotics and Automation*, Raleigh, Carolina del Norte, 7-10 Julio, pp. 1761-1768.

[6] Arai T., Pagello E., Parker L. E. (2002) "Advances in Multirobot Systems". *IEEE Transactions on Robotics and Automation*, vol. 18, Nº 5, Octubre.

[7] Arkin R. C. (1989) "Motor Schema-Based Mobile Robot Navigation". *The International Journal of Robotics Research*, Agosto 1989, pp. 92-112.

[8] Arkin R. C. y Balch T. (1997) "AuRA: Principles and Practice in Review". *Journal of Experimental and Theoretical Artificial Intelligence*, Vol.9, Nº 2-3, pp. 175-189.

[9] Asama H., Matsumoto A., Ishida Y. (1989) "Design of an Autonomous and Distributed Robot System: ACTRESS". *Procs. of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, Tsukuba, Japón, 4-6 Septiembre, pp. 283-290.

[10] Balch T. (1998a) "Taxonomies of Multirobot Task and Reward". Technical Report, Carnegie Mellon University.

[11] Balch T. (1998b) "Robots Move".AAAI Spring Symposium.

[12] Balch T. y Ram A. (1998) “Integrating Robotic Technologies with JavaBots”. Working Notes on the AAAI 1998 Spring Symposium, Standford, California.

[13] Beni G. y Wang J. (1989) “Swarm Intelligence in Cellular Robotic Systems”. *Procs. of the NATO Advanced Workshop on Robots and Biological Systems*, Il Ciocco, Toscana, Italia.

[14] Beni G. y Wang J. (1991) “Theoretical Problems for the Realization of Distributed Robotic Systems”. *Procs. of the IEEE International Conference on Robotics and Automation*, Sacramento, California, EE.UU.

[15] Bennewitz M., Burgard W., Thrun S. (2001) “Optimizing Schedules for Prioritized Path Planning of Multi-Robot Systems”. *Procs. of the IEEE International Conference on Robotics and Automation*, Seúl, Corea, 21-26 Mayo.

[16] Bennewitz M. y Burgard W. (2001) “Finding Solvable Priority Schemes for Decoupled Path Planning Techniques for Teams of Mobile Robots”. *Procs. of the 9th International Symposium on Intelligent Robotic Systems*, LAAS-CNRS, Toulouse, Francia, 18-20 Julio.

[17] Borenstein J. y Koren Y. (1989) “Real-time Obstacle Avoidance for Fast Mobile Robots”. *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 19, Nº 5, Sept/ Oct 1989, pp. 1179-1187.

[18] Botelho S. C. y Alami R. (1999) “M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement”. *Procs. of the IEEE International Conference on Robotics and Automation*, Detroit, Michigan, EE.UU.

[19] Botelho S. C. y Alami R. (2000) “A multi-robot cooperative task achievement system”. *Procs. of the IEEE International Conference on Robotics and Automation*, San Francisco, California, EE.UU.

[20] Brock O. y Khatib O. (1999) “Real-Time Obstacle Avoidance and Motion Coordination in a Multi-Robot Workcell”. *IEEE International Symposium on Assembly and Task Planning*, Oporto, Portugal, Julio, pp. 274-279.

[21] Brooks R. A. (1986) “A Robust Layered Control System for a Mobile Robot”. *IEEE Journal of Robotics and Automation*, Vol. RA-2, Nº 1, pp. 14-23.

[22] Buckley S. J. (1989) “Fast Motion Planning for Multiple Moving Robots”. *Procs. of the IEEE International Conference on Robotics and Automation*, Scottsdale, Arizona, EE.UU, pp. 322-326.

[23] Caloud P., Choi W., Latombe J. C., Le Pape C., Yim M. (1990) "Indoor Automation with Many Mobile Robots". *IEEE International Workshop on Intelligent Robots and Systems*.

[24] Cantor M. (2000) "Object-oriented Project Management with UML". John Wiley & Sons.

[25] Cao Y. U., Fukunaga A. S., Kahng A. B. (1997) "Cooperative mobile robotics: antecedents and directions". *Autonomous Robots,* 4, 1-23.

[26] Costa i Castelló R., Basáñez Villaluenga L., Suárez Feijóo R. (1995) "Planificación y Control en Entornos Multi-Robot". *Actas del 4º Congreso de la Asociación Española de Robótica y Automatización, Tecnologías de la Producción*, Zaragoza, España, pp. 123-131.

[27] Coste-Manière E. y Simmons R. (2000) "Architecture, the Backbone of Robotic Systems". *Procs. of the IEEE International Conference on Robotics and Automation*, San Francisco, California, EE.UU.

[28] Cruz Martín A. (1997) "Desarrollo e Implantación de un Algoritmo para la Planificación de Velocidades". Proyecto Fin de Carrera. Universidad de Málaga.

[29] Cruz A., Muñoz V., García Cerezo A., Ollero A. (1.998): "Moving Obstacle Avoidance Algorithm for Mobile Robots under Speed Restrictions", en *Selected Papers of the IFAC Workshop on the Intelligent Components for Vehicles (ICV'98)*, ISBN 0-08043232-8, Pergamon Press, Gran Bretaña.

[30] Cruz Martín A. (1999) "ICE Velocidad". Informe Técnico del Departamento de Ingeniería de Sistemas y Automática. Universidad de Málaga.

[31] Cruz A., Muñoz V. F., García Cerezo A. (2002a) "A Flexible Simulation Approach for Multirobot Systems". *15th International Federation of Automatic Control World Congress*, Barcelona, España, 21-26 Julio.

[32] Cruz A., Muñoz V. F., García Cererzo A. (2002b) "An Adaptable Multirobot System Simulation Schema". *Engineering of Intelligent Systems Conference*, Málaga, España, 24-27 Septiembre.

[33] Cruz-Martín A., Muñoz V., García-Cerezo A. (2004) "Genetic Algorithm Based Multirobot Trajectory Planning". *10th International Symposium on Robotics and Applications (ISORA'04)*, Sevilla, España, 28 Junio-1 Julio. *Aceptado para publicación*.

[34] De la Cueva V. y Ramos F., "Cooperative Genetic Algorithms: A New Approach to Solve the Path Planning Problem for Cooperative Robotic Manipulators sharing the same Workspace". *Procs. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Victoria, B.C., Canadá, Octubre.

[35] Departamento de Ingeniería de Sistemas y Automática (2004) http:// www.isa.uma.es

[36] Diccionario de la Lengua Española (1992). Real Academia Española, Vigésimo primera edición, Madrid.

[37] Canny J., Donald B., Reif J., Xavier P. (1988) "On the Complexity of Kinodynamic Planning", *29th Symposium on the Foundations of Computer Science*, White Plains, Nueva York, EE.UU, Octubre.

[38] Dienstel, R. (2000) "Graph Theory. Second Edition". Springer.

[39] Dudek G., Jenkin M. J. M., Milios E., Wilkes D. (1996) "A taxonomy for multi-agent robotics". *Autonomous Robots*, 3, 375-397.

[40] Dudenhoffer D. D. y Bruemmer D. J. (2001) "Command and Control Architectures for Autonomous Micro-Robotic Forces". INEEL/EXT-2001-00232, Idaho National Engineering and Environmental Laboratory, Idaho Falls, Idaho.

[41] Emery R. y Balch T. (2001) "Behavior-Based Control of a Non-Holonomic Robot in Pushing Tasks". *Procs. of the IEEE International Conference on Robotics and Automation*, Seúl, Corea, 21-26 Mayo.

[42] Erdmann M. y Lozano-Pérez T. (1986) "On Multiple Moving Objects". *Procs. of the IEEE International Conference on Robotics and Automation*, Vol. 3, Abril, pp. 1419-1424.

[43] Erdmann M. y Lozano-Pérez T. (1987) "On Multiple Moving Objects". Algorithmica, 2(4), pp. 477-521

[44] Eric Weissteins's World of Mathematics (2003) http://mathworld.wolfram.com

[45] Fernández Madrigal J. A. y González J. (1999) "Sistema Abierto para la Implantación de Arquitecturas Software de Robots. Manual de Referencia". Departamento de Ingeniería de Sistemas y Automática, Universidad de Málaga. ISBN 84-699-1636-X.

[46] Fernández Madrigal, J. A. (2003) "The BABEL Development System for Integrating Heterogeneous Robotic Software". Informe Técnico del Departamento de Ingeniería de Sistemas y Automática. Universidad de Málaga.

[47] Fernández R. y Pedraza S. (1998) "Protocolo PC-MX31". Informe Técnico del Departamento de Ingeniería de Sistemas y Automática. Universidad de Málaga.

[48] Fernández Ramos, R. N. (2001) "Navegación Autónoma de Robots Móviles en Entornos Parcialmente Conocidos". Tesis Doctoral. Universidad de Málaga.

[49] Federation of International Robot-Soccer Association, FIRA (2002). http:// www.fira.net.

[50] Ferrari C., Pagello E., Ota J., Arai T. (1998) "Multirobot motion coordination in space and time". *Robotics and Autonomous Systems*, 25, pp. 219-229.

[51] Fiorini P. y Shiller Z. (1998) "Motion Planning in Dynamic Environments using Velocity Obstacles". *The International Journal of Robotics Research*, Julio.

[52] Fowler M. y Scott K. (2000) "UML Distilled". Segunda Edición. Addison-Wesley. ISBN 0-201-65783-X.

[53] Fraichard T. y Laugier C. (1989) "Planning Movements for Several Coordinated Vehicles". *Procs. of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, Tsukuba, Japón, 4-6 Septiembre.

[54] Fraichard T. y Laugier C. (1991) "On line reactive planning for a non holonomic mobile in a dynamic world". *Procs. of the IEEE International Conference on Robotics and Automation*.

[55] Fraichard T. y Laugier C. (1992) "Kinodynamic planning in a structured and time-varying 2D workspace". *Procs. of the IEEE International Conference on Robotics and Automation*, Niza, Francia.

[56] Fraichard T. y Scheuer A. (1994) "Car-Like Robots and Moving Obstacles". *Procs. of the IEEE International Conference on Robotics and Automation*.

[57] Fraichard T. (1998) "Trajectory Planning Amidst Moving Obstacles: Path-Velocity Decomposition Revisited". *Journal of the Brazilian Computer Science Society*, Special issue on Robotics, Vol. 4, Nº 3, Abril.

[58] Franklin S. y Graesser A. (1996) “Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents”. *Procs. of the Third International Workshop on Agent Theories, Architectures and Languages*, Springer-Verlag.

[59] Fujimura K. y Samet H. (1989) “Time-Minimal Paths among Moving Obstacles”. *Procs. of the IEEE International Conference on Robotics and Automation*, Vol. 2, Scottsdale, Arizona, pp. 1110-1115.

[60] Fujimura K. (1991) “Motion Planning in Dynamic Environments”. Springer-Verlag.

[61] Fujimura K. (1992) “On Motion Planning amidst Transient Obstacles”. *Procs. of the IEEE International Conference on Robotics and Automation*, Niza, Francia, Mayo.

[62] Fujimura K. (1995) “Time-Minimum Routes in Time Dependent Networks”. *IEEE Transactions on Robotics and Automation*, Vol. 11, Nº 3, Junio.

[63] Fukuda T., Nakagawa S., Kawauchi Y., Buss M. (1988) “Self Organizing Robots Based on Cell Structures - CEBOT”. *IEEE International Workshop on Intelligent Robots*.

[64] Fukuda T., Kawauchi Y., Asama H. (1990) “Analysis and Evaluation of Cellular Robotics (CEBOT) as a Distributed Intelligent System by Communication Information Amount”. *IEEE International Workshop on Intelligent Robots and Systems*.

[65] Fulbright R. y Stephens L. M. (1994) “Classification of Multiagent Systems”. USC Technical Report ECE 06-94-02.

[66] GA Tutorial Home Page (2004), http://www.estec.esa.nl/outreach/gatutor.

[67] Gallardo G., Colomina O., Flórez F., Arques P., Company P., Rizo R. (1997) “Control Local de Robots Móviles basado en Métodos Estadísticos y Algoritmos Genéticos”, *Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA’97)*, Torremolinos, Málaga, Noviembre.

[68] Gambardella L. M. y Dorigo M. (2000) “An Ant Colony System Hybridized with a New Local Search for the Sequential Ordering Problem”. INFORMS Journal on Computing, vol. 12(3), pp. 237-255.

[69] Garey M. R. y Johnson D. S. (1979) “Computers and Intractability. A Guide to the Theory of NP-Completeness”. W. H. Freeman and Company.

[70] Garnier P. y Fraichard T. (1997) "A Fuzzy Motion Controller for a Car-like Vehicle". Rapport de Recherche nº 3200, INRIA, Junio.

[71] Gravot F. y Alami R. (2001) "An extension of the Plan-Merging Paradigm for multi-robot coordination". *Procs. of the IEEE International Conference on Robotics and Automation*, Seúl, Corea, 21-26 Mayo.

[72] Guo Y. y Parker L. E. (2002) "A Distributed and Optimal Motion Planning Approach for Multiple Mobile Robots". *Procs. of the IEEE International Conference on Robotics and Automation*, Washington DC, EE.UU., Mayo.

[73] Hackwood S. y Beni G. (1992) "Self-organization of sensors for swarm intelligence". *Procs. of the IEEE International Conference on Robotics and Automation*, pp. 819-829.

[74] Hewitt C., Bishop P., Greif I., Smith B., Matson T., Steiger R. (1973) "A Universal Modular Actor Formalism for Artificial Intelligence". *Procs. of the International Joint Conference on Artificial Intelligence*, Palo Alto, California, pp. 235-245.

[75] Holland J. (1975) "Adaptation in Natural and Artificial Systems", University of Michigan Press.

[76] Holve R., Protzel P., Naab K. (1995a) "Generating Fuzzy Rules for the Acceleration Control of an Adaptative Cruise Control System". NAFIPS Conference, 19-22 Junio, Berkeley, California, EE.UU.

[77] Holve R., Protzel P., Bernasch J., Naab K. (1995b) "Adaptative Fuzzy Control for Driver Assistance in Car-Following". *3rd European Congress on Intelligent Techniques and Soft Computin (EUFIT'95)*, Aachen, Alemania, Agosto 1995, pp. 1149-1153.

[78] Hsu D., Kindel R., Latombe J. C., Rock S. (2000) "Randomized Kinodynamic Motion Planning with Moving Obstacles". *The Fourth International Workshop on Algorithmic Foundations of Robotics*.

[79] Hwang K. S. y Ju M. Y. (2002) "Speed planning for a Maneuvring Motion". *Journal of Intelligent and Robotic Systems*, 33: 25-44.

[80] Hwang Y. K. y Ahuja N. (1992) "Gross Motion Planning - A Survey". *ACM Computing Surveys*, Vol. 24, No. 3, Septiembre.

[81] Hwang Y. K. (1995) "Motion Planning for Multiple Moving Objects". *Procs. od the IEEE International Symposium on Assembly and Task Planning*, pp. 400-405.

[82] K-Team (2002) http://www.k-team.com/software/webots.html

[83] Kant K. y Zucker S. W. (1986) "Toward Efficient Trajectory Planning: The Path-Velocity Decomposition". *The International Journal of Robotics Research*, Vol. 5, Nº 3, Fall.

[84] Kant K. y Zucker S. W. (1988) "Planning collision free trajectories in time-varying environments: a two level hierarchy". *Procs. of the IEEE International Conference on Robotics and Automation*, pp. 1644-1649.

[85] Kawauchi Y., Inaba M., Fukuda T. (1992) "A Strategy of Self-Organization for Cellular Robotic System (CEBOT)". *Procs. of the IEEE International Conference on Robotics and Automation*, Raleigh, Carolina del Norte, 7-10 Julio, pp. 1558-1565.

[86] Kephera Simulator HomePage (1997) http://diwww.epfl.ch/lami/team/michel/khep-sim/

[87] Khatib O. (1986) "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots". *The International Journal of Robotics Research*, Vol. 5, Nº 1, Spring.

[88] Kindel R., Hsu D., Latombe J. C., Rock S. (2000) "Kinodynamic Motion Planning amidst Moving Obstacles". *Procs. of the IEEE International Conference on Robotics and Automation*, San Francisco, California, EE.UU., Abril.

[89] Konolige K. y Myers K. (1998) "The Saphira Architecture for Autonomous Mobile Robots". En *Artificial Intelligence and Mobile Robotics: Case Studies of Succesful Robot Systems*. Eds. D. Kortenkamp, R. P. Bonasso y R. Murphy. MIT Press.

[90] Latombe J. C. (1991) "Robot Motion Planning" Kluwer Academic Publishers. ISBN 0-7923-9129-2

[91] LaValle S. M. y Hutchinson S. A. (1998) "Optimal Motion Planning for Multiple Robots Having Independent Goals". *IEEE Transactions on Robotics and Automation*, Vol.14, no. 6, Diciembre, pp. 912-925.

[92] Lee J. , Nam H. S., Lyou J. (1995) "A Practical Collision-Free Trajectory Planning for Two Robot Systems". *Procs. of the IEEE International Conference on Robotics and Automation*.

[93] Leung C. H. y Zalzala A. M. S. (1994) "A Genetic Solution for the Motion of Wheeled Robotic Systems in Dynamic Environments", *International Conference on Contro*l, Vol. 1, pp. 760-764, Marzo.

[94] Liu Y, Kuroda S., Naniwa T., Naborio H., Arimoto S (1989) "A Practical Algorithm for Planning Collision-Free Coordinated Motion of Multiple Mobile Robots". *Procs. of the IEEE International Conference on Robotics and Automation*, pp. 327-332.

[95] Lynx Real-Time Systems, Inc. (1993) "LynxOs Application Writer's Guide".

[96] Lozano-Pérez T. (1983) "Spatial Planning: A Configuration Space Approach". *IEEE Transactions on Computers*, Vol. 32, pp. 108-120.

[97] Martínez Sánchez, M. A. (2001) "Integración de Brazos Manipuladores en Robots Móviles". Tesis Doctoral. Universidad de Málaga.

[98] Mataric M. J. (1995) "Designing and Understanding Adaptive Group Behavior". *Adaptive Behavior*, 4:1, Diciembre, pp. 51-80.

[99] Mataric M. J. (1998) "Coordination and learning in multirobot systems". *IEEE Intelligent Systems*, Marzo/Abril, pp. 6-8.

[100] Mataric M. J. (2001) "Learning in behavior-based multi-robot systems: policies, models, and other agents". *Journal of Cognitive Systems Research*, 2, pp- 81-93.

[101] Matlab v6R12 (2002) http://www.mathworks.com.

[102] Matsumoto A., Asama H., Ishida Y. (1990) "Communication in the Autonomous and Decentralized Robot System ACTRESS". *Procs. of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pp. 835-840

[103] Merriam-Webster On-line (2002). http://www.m-w.com.

[104] Meyer B. (1998) "Object-oriented software construction". Prentice-Hall.

[105] Michalewicz A. (1992) "Genetic Algorithms+Data Structures=Evolution Programs". Springer-Verlag.

[106] Mission Lab (2002) http://www.cc.gatech.edu/ai/robot-lab/research/MissionLab/

[107] Mitsumoto N., Fukuda T., Shimojima K., Ogawa K. (1995) "Micro Autonomous Robotic System and Biologically Inspired Immune Swarm Strategy as a Multi Agent Robotic System". *IEEE International Conference on Robotics and Automation*.

[108] Monteiro D. C. y Madrid M. K. (1999) "Planning of Robot Trajectories with Genetic Algorithms". *IEEE First Workshop on Robot Motion and Control (RoMoCo'99)*.

[109] Moraleda E., Matía F., Puente E. A. (1998) "ARCO: Architecture for Autonomous Mobile Platforms Co-operation in Industrial Environments". *3rd IFAC Symposium on Intelligent Autonomous Vehicles (IAV'98)*, Madrid, 25-27 Marzo.

[110] Muñoz V., Ollero A., Prado M., Simón A. (1994a) "Mobile Robot Trajectory Planning with Dynamic and Kinematic Constraints". *Procs. of the IEEE International Conference on Robotics and Automation*, 4, pp. 2802-2807.

[111] Muñoz V., Martínez J. L., Ollero A. (1994b) "Navigation with Uncertainty Position in the RAM-1 Mobile Robot". *IFAC Symposium on Artificial Intelligence in Real Time Control*, Valencia, Spain.

[112] Muñoz Martínez V. F. (1995) "Planificación de Trayectorias para Robots Móviles". Tesis Doctoral. Universidad de Málaga.

[113] Muñoz Martínez V. F. (1998) "Sistemas de Eventos Discretos". Apuntes de clase para el Curso de Doctorado "Técnicas de Modelado y Computacionales, y de Análisis en Ingeniería". Departamento de Ingeniería de Sistemas y Automática. Universidad de Málaga.

[114] Muñoz V., Cruz A., Ollero A., García Cerezo A. (1998) "Speed Planning Method for Mobile Robots under Motion Constraints". *3rd IFAC Symposium on Intelligent Autonomous Vehicles (IAV'98)*, Madrid, 25-27 Marzo.

[115] Muñoz V., Cruz A., García Cerezo A. (1.998): "Speed Planning and Generation Approach based on the Path- Time Space for Mobile Robots", *IEEE International Conference on Robotics and Automation (ICRA'98)*, Leuven, Bélgica.

[116] Muñoz V. F., García Cerezo A., Cruz A. (1999) "A Mobile Robots Trajectory Planning Approach under Motion Restrictions". *Special Issue on Intelligent Autonomous Vehicles of the Journal of Integrated Computer-Aided Engineering*, Vol. 6, No. 4, pp 331-347.

[117] Noreils F. R. (1993) "Toward a Robot Architecture Integrating Cooperation between Mobile Robots: Application to Indoor Environment". *The International Journal of Robotics Research*, Vol. 12, No. 1, Febrero.

[118] Noreils F. R. y Chatila R. G. (1995) "Plan Execution monitoring and control architecture for mobile robots". *IEEE Transactions on Robotics and Automation*, Vol.11, no. 2, Abril, pp. 255-266.

[119] Papadimitriou C. H. y Steigliz K. (1998) "Combinatorial Optimization. Algorithms and Complexity". Dover Publications.

[120] Parker L. E. (2000) "Current State of the Art in Distributed Robot Systems". Distributed Autonomous Robotic Systems, Springer, pp. 3-12.

[121] Pedraza S. (2000) "Modelado y Análisis de Vehículos de Cadenas para la Navegación Autónoma". Tesis Doctoral. Universidad de Málaga.

[122] Pedraza S., Fernández R., García Cerezo A. (1999) "Robot móvil traccionado por cadenas con capacidad de operación autónoma y teleoperada". Solicitud de patente nº P9902511.

[123] PlanetMath (2003) http://planetmath.org

[124] Player/Stage (2003) http://playerstage.sourceforge.net/

[125] Protzel P., Holve R., Bernasch J., Naab K. (1993) "Fuzzy Distance Control for Intelligent Vehicle Guidance". *12th Anual Meeting of the North American Fuzzy Information Processing Society*, Allentown, Pennsylvania, 22-25 Agosto, pp. 87-91.

[126] O'Donnell P. A. y Lozano-Pérez T. (1989) "Deadlock-Free and Collision Free Coordination of Two Robot Manipulators". *Procs. of the IEEE/RSJ International Conference on Robotics and Automation*, Scottsdale, Arizona, EE.UU., Mayo, pp. 484-489.

[127] Ollero Baturone, A. (1991) "Evolución y Perspectivas de la Robótica". Lección Inaugural Curso 1991-1992, Universidad de Málaga.

[128] Parker L. E. (1994) "Heterogeneneus Multi-Robot Cooperation". PhD Thesis, Massachussets Institute of Technology, Artificial Intelligence Laboratory.

[129] Parker L. E. (1998) "ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation". *IEEE Transactions on Robotics and Automation*, 14(2).

[130] Parsons D. y Canny J. (1990) "A Motion Planner for Multiple Mobile Robots". *Procs. of the 7th IEEE International Conference on Robotics and Automation*, Cinncinnati, Ohio, EE.UU.

[131] Reif J. y Sharir M. (1985) "Motion Planning in the Presence of Moving Obstacles". *Procs. of the 26th Annual IEEE Symposium on Foundations of Computer Science*, Portland, Oregón, EE.UU., Octubre, pp.144-154.

[132] RoboCup (2002) http://www.robocup.org.

[133] Rossum's Playhouse (2002) http://rossum.sourceforge.net/sim.html

[134] Rude M., Löwer J., Rupp T. (1995) "Coordination of Mobile Robots by Estimating Relative Spatial and Temporal Uncertainties". *2nd IFAC Conference on Intelligent Autonomous Vehicles,* Helsinki, Finlandia, Junio.

[135] Sánchez G. y Latombe J. C. (2002) "Using a PRM Planner to Compare Centralized and Decoupled Planning for Multi-Robot Systems". *Procs. of the IEEE/RSJ International Conference on Robotics and Automation*, Washington, DC, EE.UU., Mayo.

[136] Seraji H. y Howard A. (2002) "Behavior-Based Robot Navigation on Challenging Terrain: A Fuzzy Logic Approach". *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 3, Junio.

[137] Shibata T., Fukuda T., Kozuge K., Arai F. (1992) "Selfish and Corrdinative Planning for Multiple Mobile Robots by Genetic Algorithm". *Procs. of the 31st Conference on Decision and Control*, Tucson, AZ, Diciembre.

[138] Shibata T. y Fukuda T. (1993) "Coordinative Behavior in Evolutionary Multi-Agent-Robot System". *Procs. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Yokohama, Japón.

[139] Shiller Z. y Gwo Y. (1991) "Dynamic Planning of Autonomous Vehicles". *IEEE Transactions on Robotics and Automation*, 7(2).

[140] Sick-Optic Electronic (1985) Manual de Referencia del Scanner Laser PLS. Alemania.

[141] Siméon T., Leroy S., Laumond J. P. (2002) "Path Coordination for Multiple Mobile Robots: A Resolution Complete Algorithm". *IEEE Transactions On Robotics and Automation*, Vol. 18, No. 1, Febrero.

[142] Stenzel R. (2000) "A behavior-based control architecture". *Int. Conference on Systems, Man and Cybernetics*, pp. 3235-3240.

[143] Stone P. y Veloso M. (2000) "Multiagent Systems: A Survey from a Machine Learning Perspective". *Autonomous Robotics*, volume 8, number 3, Julio.

[144] Sugawara K. y Watanabe T. (2002) "Swarming Robots - Foraging Behavior of Simple Multi-robot System". *Procs. of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, EPFL, Lausana, Suiza, Octubre.

[145] Svetska P. y Overmars M. H. (1996) "Multi-Robot Path Planning with Super-Graphs". *Symposium on Robotics and Cybernetics*, pp. 482-487.

[146] Todt E., Raush G., Suárez R. (2000). "Analysis and Classification of Multiple Robot Coordination Methods". *Procs. of the IEEE International Conference on Robotics and Automation*, San Francisco, California, EE.UU.

[147] Tournassoud P. (1986) "A Strategy for Obstacle Avoidance and its Application to Multi-Robot Systems". *Procs. of the IEEE International Conference on Robotics and Automation*, Vol. 3, pp. 1224-1229, Abril.

[148] Traveling Salesman Problem Homepage (2003) http://www.math.princeton.edu/tsp

[149] Tsubouchi T. y Arimoto S. (1994) "Behaviour of a Mobile Robot Navigated by an 'Iterated Forecast and Planning' Scheme in the Presence of Multiple Moving Obstacles. *Procs. of the IEEE International Conference on Robotics and Automation*, San Diego, California, EE.UU., pp. 2470-2475, Mayo.

[150] Visser A. (2002) "Organisation and Design of Autonomous Systems", Capítulo 12. http://www.science.uva.nl/~arnoud/OOAS/

[151] Wang J. y Beni G. (1990) "Distributed computing problems in cellular robotic systems". *Procs. of the IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, pp. 819-826.

[152] Warren C. W. (1990) "Multiple Robot Path Coordination Using Artificial Potential Fields". *Procs. of the IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 500-505.

[153] White T. y Pagurek B. (1998) "Towards Multi-Swarm Problem Solving in Networks". *International Conference on Multi-Agent Systems*.

[154] Whitley D. (1993) “A Genetic Algorithm Tutorial”. Computer Science Department, Colorado State University.

[155] Wooldridge M. y Ciancarini P. (2001) “Agent-Oriented Software Engineering: The State of the Art”. *Agent-Oriented Software Engineering* (eds. P. Ciancarini, M. Wooldridge). Springer-Verlag Lecture Notes in AI, volumen 1957.

[156] Yuta S., Premvuti S. (1992) “Coordinating Autonomous and Centralized Decision Making to Achieve Cooperative Behaviors Between Multiple Mobile Robots”. *Procs. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Raleigh, Carolina del Norte, 7-10 Julio, pp. 1566-1574.