Universidad
Politécnica de Valencia

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS,
COMPUTADORES Y AUTOMÁTICA

Modelado, Detección de Colisiones y Planificación de Movimientos en Sistemas Robotizados mediante Volúmenes Esféricos

-TESIS DOCTORAL-

Por : D. Martin Mellado Arteche

Director : D. Josep Tornero i Montserrat

Valencia, 20 de Mayo de 1996
(Re-editada el 23 de Octubre de 2015)

A la memoria de mi padre.

Agradecimientos

En primer lugar, deseo expresar un sincero reconocimiento a mi Director de Tesis, D. Josep Tornero i Montserrat, sin cuya dirección esta Tesis Doctoral no habría sido posible. Casi diez años de colaboración con él me han servido para orientar mi forma de actuar en la Universidad y fuera de ella.

Es difícil expresar con palabras mi gratitud a Edu por su inestimable ayuda; sólo me queda solicitar excusas por el humo extra que ha tenido que soportar debido a esta Tesis. En un agradecimiento extensible a todos mis compañeros de departamento, quiero hacer mención especial a dos de ellos, Julián y Ángel, con los que he compartido momentos inolvidables en el TAC y a los que debo agradecer su amistad.

Tampoco puedo olvidar a aquellos titulados (o futuros titulados) que, realizando su PFC en el Laboratorio de Robótica, han colaborado con sus implementaciones, en mayor o menor medida, a la elaboración de esta Tesis Doctoral, así como a superar los malos ratos. Nombrarlos a todos representa correr el riesgo de olvidarse de alguno, pero creo que más de un nombre merece, cuanto menos, figurar aquí: Antonio Sánchez, Salvador Navarro, Ricardo Díaz, Santiago Martínez, Enrique Ballester, Roberto Guzmán, Jesús Aparicio, Javier Traver, Miguel Pla, Javier Granados, ...

Mi madre me ha alentado cuando lo he necesitado y me ha apoyado siempre sin pedir nada a cambio. Nunca lo olvido. Gracias.

Por último, sin que por ello sea menos importante, la fuerza necesaria para terminar este trabajo la he conseguido gracias al cariño de Toñi. Seguro que pronto compartiremos los tres muchos momentos muy felices.

ÍNDICE

LISTA DE FIGURAS

PRÓLOGO:

CAPÍTULO 1:

CAPÍTULO 2:

CAPÍTULO 3:

CAPÍTULO 4:

CAPÍTULO 5:

EPÍLOGO:

LISTA DE TABLAS

CAPÍTULO 3:

CAPÍTULO 4:

PRÓLOGO

Como una entrada a esta memoria, en este prólogo se incluye una breve introducción a la problemática de la planificación de movimientos y la motivación, objetivos y planteamiento de la Tesis Doctoral realizada, presentando el resto de la memoria.

—Sigue sin haber vocaciones, ¿eh?
—Oh, hay algunas, pero solamente en las categorías donde la necesidad no es vital. Lo que necesitamos son investigadores. Usted lo sabe muy bien. El problema es que con los robots prohibidos en la Tierra, el dedicarse a la robótica se ha convertido en algo impopular.
—El maldito complejo de Frankenstein —dijo Bogert, imitando conscientemente una de las frases favoritas del otro.

"Lenny". Isaac Asimov, 1957.

I. Introducción: Aspectos en la Planificación de Movimientos

En los últimos años existe un interés creciente en dotar al robot la capacidad de planificar sus propios movimientos cuando está llevando a cabo alguna tarea, de forma que se consiga la autonomía del mismo. Esta es, *grosso* modo, la filosofía que subyace en los estudios de planificación de movimientos (*motion planning*). No es difícil comprender la importancia de dicha empresa, como tampoco es complicado ser consciente de su envergadura. Efectivamente, salvo casos muy sencillos, muy bien estudiados, no es fácil dotar al robot de la inteligencia necesaria para ejecutar tareas con una total independencia. Precisamente, un aspecto clave en el tema de la planificación de movimientos, que permitirá o impedirá su aplicación industrial, es la complejidad computacional de los algoritmos utilizados.

Conseguir una mayor automatización facilitaría al mismo tiempo la programación simbólica de los sistemas robotizados, describiendo de forma declarativa el objetivo de una tarea, sin entrar en cómo hay que resolver la tarea, ni las acciones que deba realizar el robot para resolverla.

La planificación de movimientos es un campo muy amplio. Por ejemplo, en una típica tarea industrial de ensamblaje, acciones como coger elementos, transportarlos y posicionarlos, pueden ser realizadas por un robot. Para ello se debe conocer cómo son y dónde están todos los componentes de la escena (piezas a transportar y obstáculos fijos del entorno, así como el mismo robot). Esta información podría ser suministrada por el usuario mediante un sistema CAD/CAM, pero también puede ser generada automáticamente con el uso de sensorización. Normalmente la forma de los objetos se aproxima de alguna manera para reducir la complejidad del problema, representando los objetos mediante algún modelo.

Para evitar que el robot (y su carga) colisione con los obstáculos, habrá que realizar el cálculo de distancias entre los modelos que los representen. Es entonces cuando la planificación de movimientos puede entrar en juego, con objeto de obtener por dónde debe moverse el robot (es decir, el camino a recorrer por el robot) para no colisionar con los obstáculos de su entorno. La resolución de este problema, denominado problema básico de la planificación de movimientos o planificación del camino (*path planning*), especifica un camino geométrico libre de colisiones, pudiéndose realizar en el espacio de configuraciones o directamente en el espacio cartesiano. En este problema se deben considerar la evitación de colisiones con los obstáculos estáticos y los rangos de trabajo de las articulaciones del sistema robotizado.

Sin embargo, esta situación no es de las más complejas; podría ser mucho más sofisticada. De hecho no es nada extraño que pueda haber más de un robot cooperando en la realización de una tarea, o es posible que los obstáculos tengan algún tipo de

movilidad ([War90] y [Fio95] presentan trabajos en estos dos campos). En tales circunstancias el entorno es variable, por lo que la planificación no será, tan sólo, función de la geometría del problema, sino también del tiempo, debiendo obtenerse, además del camino a recorrer por el robot, las características de ese movimiento (velocidades y aceleraciones fundamentalmente). Este nuevo problema, denominado generación de trayectorias (*trajectory generation*), debe ajustar unas funciones temporales suaves (con segunda derivada continua) al camino geométrico en el espacio de las variables de articulación ([K&Z86], [FS+95]). En este problema, al considerarse el tiempo, hay que tener en cuenta los límites de los actuadores (velocidades y aceleraciones) y la evitación de colisiones con obstáculos dinámicos o móviles.

Adicionalmente el robot puede tener múltiples articulaciones, manos con varios grados de libertad, restricciones en sus movimientos u otras consideraciones que pueden aumentar considerablemente la complejidad del problema. Finalmente, la tarea podría incluir algunos criterios de optimización de algún índice de prestaciones (casi siempre se trata de minimizar el tiempo total de un ciclo de trabajo, la longitud del camino a recorrer por un punto característico, el volumen barrido por el móvil, etc.). Sin embargo, el planteamiento de una solución común a todos estos aspectos es inabordable en el contexto de un único trabajo.

La planificación de movimientos está muy ligada a otros campos afines, como el problema del seguimiento del camino (*path tracking*), que es la implementación en tiempo real del movimiento mediante un control con realimentación ([Kha86], [GC+95]). Esta fase debe incluir la calibración, monitorización y sensorización del robot, así como, evidentemente, el control ejercido sobre los actuadores del mismo.

Todos estos aspectos están fuertemente relacionados entre sí, pudiéndose esquematizar con una arquitectura jerárquica como la que se muestra en la Figura P.1. Entre los diferentes niveles existe una interacción que permite tener siempre disponibles en un nivel, como resultados de un módulo de otro nivel, los datos de entrada requeridos.

Por encima del nivel de la planificación de movimientos deben considerarse aquellos niveles que permiten al usuario realizar la programación del robot. En concreto, el nivel superior es el sistema de programación, sobre el que interactúa el usuario para especificar de forma declarativa las tareas que debe realizar el robot. Trabajar a este nivel permite definir tareas del estilo "ensambla las piezas A y B mediante el tornillo C". La planificación de tareas para el ensamblado automático de componentes es un tema tratado desde hace mucho tiempo [Ern61], [Ino74]. Existen numerosos sistemas, métodos y lenguajes para programar los sistemas robotizados ([L-P83a], [PC+93]), destacando sobremanera aquéllos que incluyen sistemas CAD para la simulación de los movimientos del robot ([Agb95]).

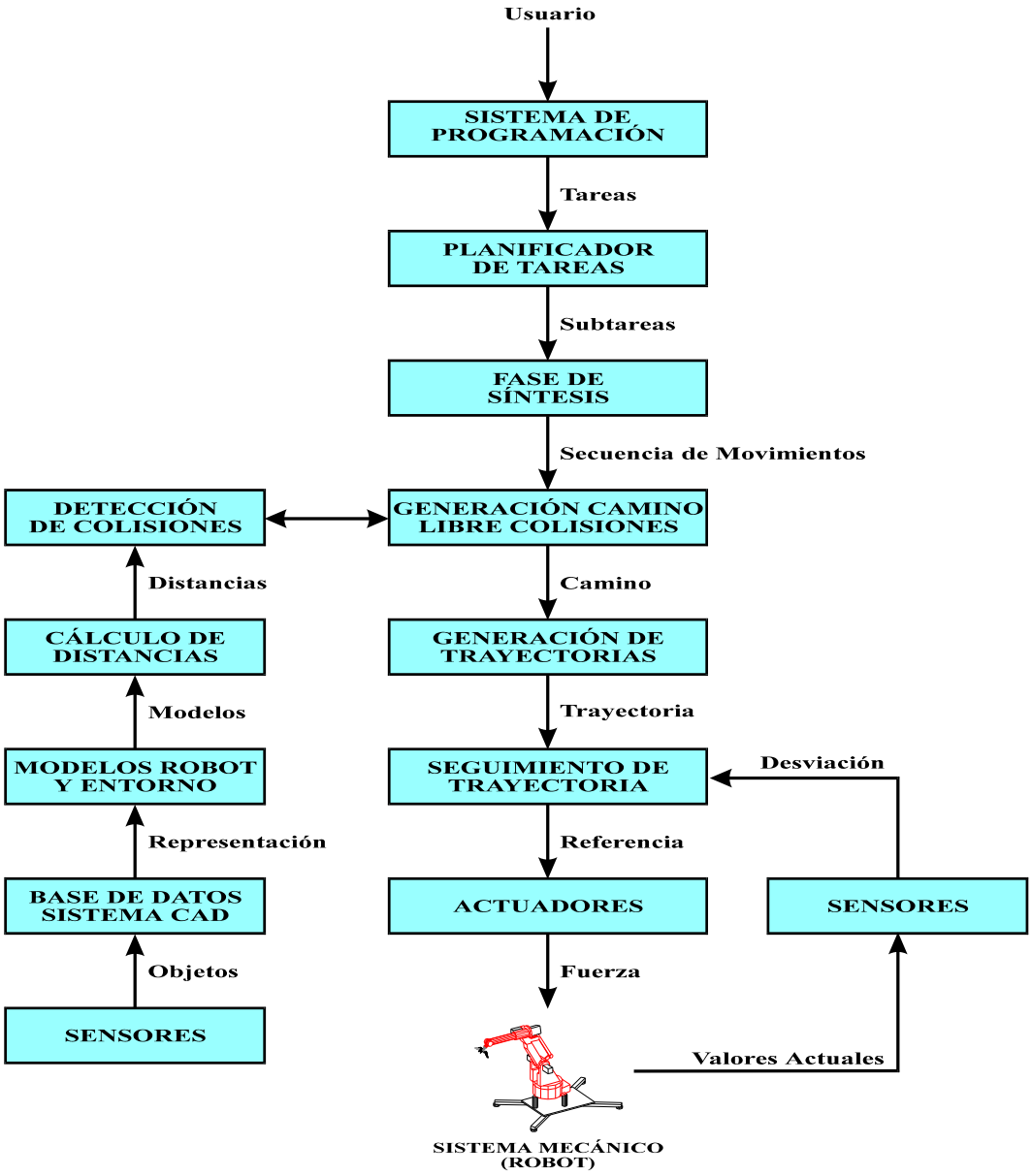


Figura P.1. Integración de la Planificación de Movimientos en el Proceso Global de la Automatización de la Robótica. *La entrada a la planificación de movimientos es una secuencia de movimientos del robot, obtenidos en un nivel superior a partir de un planificador de tareas y la fase de síntesis. La planificación de movimientos se basa en la detección de colisiones, a su vez fundamentada en el cálculo de distancias y los modelos del robot y su entorno, que pueden ser obtenidos de los sistemas CAD y/o sensores externos. A partir del camino generado se obtiene una trayectoria, cuyo seguimiento se realiza mediante un control con realimentación de la desviación.*

En el siguiente nivel, el planificador de tareas resuelve la forma en que debe realizarse esta tarea, descomponiendo esta tarea principal en una serie de subtareas, como por ejemplo, "coge la pieza A y déjala en una posición conocida", "coge la pieza B y déjala sobre la pieza A" y "coge el tornillo C y ajusta ambas piezas con él". Esta fase, denominada secuenciación del ensamblado o planificación de la secuencia del ensamblado, ha sido ampliamente estudiada ([HMS91], [W&R91], [D&G92], [WLL92] y [WK+95]).

Cada una de estas subtareas se sintetiza en una secuencia de movimientos ("aproximación a pieza A", "movimiento a posición de agarre de pieza A", etcétera) y acciones de robot ("movimiento libre", "movimiento rectilíneo", "cerrar pinza", ...), dando lugar a un programa robot. Dentro de este campo se encuentran los estudios realizados para determinar el agarre óptimo de piezas ([PLP90], [F&C92], [PSF93] y [Bic95]).

Son las acciones definidas en un programa robot, especialmente los movimientos de la configuración actual del robot a otra nueva configuración, los que se deben transformar a un problema de planificación de movimientos.

Como ya se ha comentado, en paralelo a la planificación de movimientos se encuentra la detección de colisiones, utilizada fundamentalmente para la generación de un camino libre de colisiones. Para detectar y evitar estas colisiones, se debe disponer de un modelo del robot y su entorno, sobre los que se aplicarán el cálculo de distancias. Los modelos del robot y de su entorno pueden ser tomados de la representación de los mismos de un sistema CAD así como generados directa o indirectamente (a través del sistema CAD) mediante sistemas de visión artificial ([Ike93], [Ina93], [K&I93], [KII94] y [G&T95]) o por medio de sensores de diferentes tipos ([N&Y92], [MDW93], [L&C94], [FBE94], [Eve95], [Sud95] y [K&K95]). La posible incorporación de este tipo de sensorización es necesaria para dotar a los robots de una flexibilidad cada vez más necesaria en los procesos industriales. Además la integración sensorial es necesaria para la monitorización en tiempo real de las acciones ejecutadas por el robot.

II. Motivación y Objetivos de la Tesis Doctoral

En todo campo de investigación, siempre se encuentra un hito que marca el estado en que se encuentra dicho campo. En la planificación de movimientos, la publicación de "Robot Motion Planning" [Lat91] por Latombe, autor de numerosas aportaciones en este campo, marca el final de una etapa, así como el inicio de otra bastante diferente. [Lat91] revisa el estado del arte en la planificación de movimientos muy acertado, estudiando las diferentes alternativas para solucionar el problema básico y diferentes extensiones del mismo. A partir de entonces, los investigadores, si bien no han dado la espalda a este

problema, si que han centrado sus esfuerzos en otras variantes del problema no consideradas hasta entonces o propuestas como campos abiertos en [Lat91].

Sin embargo, pese a todos los esfuerzos realizados en el campo de la planificación de movimientos, los resultados de estas investigaciones han tenido poco impacto en el mundo industrial. Realmente, la disponibilidad de algoritmos de planificación de movimientos suficientemente rápidos para su implantación en sistemas comerciales es muy reciente, y siempre hay un retraso entre el desarrollo de las nuevas tecnologías y su incorporación en productos industriales. De hecho, todavía no se disponen de aplicaciones industriales que incluyan de forma integrada la planificación de movimientos libres de colisiones, aunque empiezan a aparecer comercialmente sistemas simuladores gráficos que incluyen algunas prestaciones para la detección de colisiones, como un primer paso hacia la planificación de movimientos.

Probablemente, las dos razones fundamentales de la falta de aplicación de la planificación de movimientos a la robótica tradicional son la dificultad de construir y mantener modelos geométricos precisos del entorno y el tiempo requerido para producir una planificación de movimientos aceptable para robots y entornos razonablemente complejos. Lozano-Pérez [L-P93], otro de los pioneros en el campo de la planificación de movimientos en sistemas robotizados, resalta principalmente el primer aspecto frente al segundo, quizá generalmente más citado, por la dificultad que lleva consigo el generar los modelos a partir de datos sensoriales, lo que da lugar a que cada vez se aplique más la planificación de movimientos en campos donde los modelos ya están disponibles y la velocidad no es un aspecto crítico (el diseño mecánico, los gráficos y la simulación por ordenador, la química, la biología e incluso la medicina).

Por tanto, se puede concluir que, aún existen dos puntos abiertos dentro de la planificación de movimientos libres de colisión, que tienen gran influencia a la hora de desarrollar procedimientos eficientes en entornos complejos y sobre los que resulta necesario concentrar esfuerzos de investigación. Estos dos puntos son:

- La representación y modelado del robot, así como de los objetos (obstáculos) que se encuentren en su región de trabajo.
- La posibilidad de disponer de procedimientos rápidos para la detección de colisiones entre el robot y los obstáculos.

En el estudio y el desarrollo de nuevos métodos para estos dos aspectos, así como en la búsqueda de nuevas técnicas que faciliten su aplicación a la planificación de movimientos de sistemas robotizados, se centra el trabajo de investigación desarrollado durante los últimos años por el autor de esta Tesis Doctoral.

III. Planteamiento de la Tesis Doctoral

Esta Tesis Doctoral presenta un nuevo enfoque al problema de la planificación de movimientos que permite trabajar directamente en el espacio cartesiano, con sistemas multi-robot y entornos complejos, basado en la aplicación de varios métodos de optimización inéditos dentro de las técnicas de planificación de movimientos con campos potenciales y en el desarrollo de procedimientos rápidos de detección de colisiones, mediante las simplificaciones realizadas al utilizar un modelo original basado en esferas.

En concreto, el presente trabajo comienza introduciendo una nueva técnica de modelado de cualquier tipo de objetos, pero con altas prestaciones para el modelado de sistemas robotizados, que facilita la detección de colisiones entre el robot y los obstáculos de su área de trabajo y posibilita su aplicación en problemas de planificación de movimientos para entornos industriales.

Los modelos que se plantean en este trabajo se componen de infinitas esferas y su volumen sirve como envolvente de los objetos reales. Para poder trabajar con estos modelos se extienden los conceptos tradicionales del álgebra vectorial a un espacio esférico (Sección 2.3.1). Estos volúmenes esféricos se definen mediante un número reducido de esferas de control. La aplicación de unas relaciones lineales entre estas esferas de control produce unos modelos esféricos, que se han denominado poli-esferas (Sección 2.3.2), que se pueden considerar como una extensión de los politopos convencionales (poliedros convexos). Estas relaciones lineales se pueden aplicar con más de un parámetro o grado de libertad, produciendo modelos más o menos complejos. La formalización de estos modelos, cuya base se recoge en [T&M94], ha permitido extender trabajos anteriores ([THK91] y [HKT92]).

Aplicando a las esferas de control relaciones cuadráticas o cúbicas, se definen las esferoides (Secciones 2.4.1-3), lo que permite extender al espacio esférico el concepto de *splines*, muy utilizado en la interpolación de curvas para sistemas CAD. Estos modelos resultan muy adecuados para modelar brazos-robot articulados, tal como se muestra en [M&T93]. También en este caso se pueden considerar diferentes grados de libertad en los parámetros, permitiendo obtener un conjunto muy amplio de modelos (Sección 2.4.4).

Una vez se dispone de la definición matemática de estos modelos geométricos sencillos pero potentes, se debe determinar una técnica de calcular estos volúmenes esféricos como envolventes de objetos reales. Para ello se desarrolla un nuevo método que permite obtener la poli-esfera mínima envolvente mediante una optimización con dos niveles: el nivel superior, basado en el método *Downhill Simplex*, busca los centros de las esferas de control que permitan obtener un volumen mínimo; el nivel inferior, mediante la utilización de la Transformada de Hough, determina los radios de las esferas que producen el mínimo volumen envolvente posible (Sección 3.1.1).

Este método, cuyos fundamentos se mostraron en [B&T95a] y [B&T95b], se generaliza en este trabajo para su aplicación a cualquier tipo de poli-esferas. Además, una inicialización adecuada, mediante un preproceso de agrupamiento o *clustering* de los puntos de entrada, mejora notablemente los resultados iniciales, tanto en la velocidad de cálculo como en el volumen obtenido (Sección 3.1.2). Los resultados de esta optimización para determinar volúmenes mínimos en el contexto de la robótica se han publicado en [BTM96].

Para simplificar aún más los modelos utilizados, se desarrolla un método, basado en la determinación de la dimensión o grados de libertad del volumen generado, que permite reducir el número de esferas necesarias para definir un objeto mediante una poli-esfera como volumen esférico envolvente (Sección 3.2). Este método de reducción y simplificación de poli-esferas también está incluido en [T&M94].

Por otra parte, el cálculo de esferoides como modelos envolventes se puede realizar utilizando únicamente técnicas de agrupamiento (Sección 3.3). Las esferoides presentan el inconveniente de ser en general modelos no convexos, por lo que sólo se puede garantizar que engloba completamente un objeto realizando una comprobación de una manera exhaustiva. Un estudio empírico permite detectar los casos más conflictivos al aplicarse como envolventes de brazos-robot articulados y determinar unos factores de corrección que presentan buenos resultados.

Los sistemas robotizados suelen presentar una naturaleza jerárquica con diferentes niveles: células, sistemas, sub-sistemas, elementos, bloques, primitivas, etcétera. Una estructura jerárquica extendida con estos niveles y diferentes grados de precisión en sus modelos (esferoides y poli-esferas como representación aproximada y poliedros como representación exacta), se introduce en este trabajo (Sección 3.4). Esta estructura, generada automáticamente mediante el algoritmo desarrollado para el cálculo de volúmenes envolventes, resulta altamente provechosa para la implementación de un procedimiento rápido de detección de colisiones, tal como se mostró en [TM+92].

El hecho de recurrir a aproximaciones con envolventes esféricas se debe a la ventaja que estos modelos presentan frente a los poliedros (utilizados convencionalmente) a la hora de considerar la sencillez, tanto desde el punto de vista matemático y geométrico del modelo, como de sus posibilidades para el desarrollo de procedimientos de detección de colisiones, lo que sobrepone la pérdida de precisión inicial del nuevo método. Para ello se generalizan los algoritmos utilizados sobre modelos tradicionales, obteniéndose algoritmos rápidos de detección de colisiones basados en el cálculo de distancias y optimizados para el caso de las poli-esferas (Sección 4.3). Cuando en los modelos intervienen esferoides, se dispone de ecuaciones no lineales, por lo que el cálculo de distancias se ha resuelto mediante técnicas de optimización clásicas (Sección 4.4).

En este trabajo, como ampliación de las bases que se presentaron en [TM+92], se concreta una metodología que permite aplicar la detección de colisiones entre modelos básicos a una estructura jerárquica extendida que representa una célula con más de un sistema controlable (célula multi-robot) para resolver el problema de la auto-colisión (Sección 4.5). Esta metodología tiene unas características que posibilitan su aplicación en

tiempo real: la respuesta es rápida y anticipativa, entendiendo por tal que todo movimiento de cualquier robot ha sido validado antes de su ejecución, por lo que no necesariamente debe realizarse en un proceso de simulación off-line. Para garantizar estas características, el procedimiento permite fijar dinámicamente las velocidades de los movimientos de los sistemas, de forma que cuanto más cercano se encuentre un robot a un obstáculo (u otro robot que actúa como obstáculo para el primero) más lento será su movimiento, permitiendo utilizar modelos más precisos con mayor tiempo de respuesta.

El objetivo de disponer de modelos sencillos y procedimientos rápidos de cálculo de distancias es conseguir métodos eficientes y rápidos de planificación de movimientos libres de colisiones en sistemas robotizados. De hecho, la técnica de planificación de movimientos denominada de campos potenciales artificiales, basada fundamentalmente en el cálculo de distancias, se va a beneficiar enormemente de estos dos aspectos, sacando muy buen partido cuando los potenciales atractivos y repulsivos se definen para los modelos esféricos (Sección 5.2).

Cuando se dispone de un móvil con traslación y rotación, la mayoría de trabajos resuelven el problema convirtiéndolo en un nuevo problema en el espacio de configuraciones. Sin embargo, trabajar en este espacio sólo produce buenos resultados para entornos conocidos y estáticos, ya que los objetos se deben convertir al espacio de configuraciones del robot, dificultándose el desarrollo de aplicaciones en tiempo real. En este trabajo se introduce, a partir de una representación homogénea de posición y orientación, una nueva definición de campos potenciales con rotación que permite realizar la planificación de un móvil directamente en el espacio cartesiano, evitando la conversión al espacio de configuraciones (Sección 5.3).

Por otra parte, la necesidad de experimentar nuevas formas de realizar la planificación sobre entornos modelados con campos potenciales, ha motivado a la aplicación, por primera vez en este contexto, diversos métodos de optimización clásicos (Sección 5.4). Para ello, se consideran tres métodos diferentes, uno basado en el gradiente, otro en el Hessiano y un tercero como combinación de ambos. [M&T96] recoge las aportaciones obtenidas al aplicar estos métodos, así como la definición de potenciales con rotación en el espacio cartesiano.

De cara a validar los aspectos teóricos sobre un sistema robotizado industrial, resulta necesario disponer de sistemas experimentales que permitan realizar pruebas sobre aplicaciones prototipo, ya que los sistemas comerciales suelen ser bastante cerrados. El desarrollo de diverso software para la programación, simulación y control de sistemas robotizados (Sección I del Epílogo) ha sido publicado en un diversas publicaciones, resumidas por ejemplo en [M&V94], [M&V95] y [V&M95].

IV. Presentación de la Memoria

La memoria se ha organizado en este prólogo, que incluye una breve introducción a la problemática de la planificación de movimientos y la motivación, objetivos y planteamiento de la Tesis Doctoral realizada, más cinco capítulos y un epílogo.

El primer capítulo realiza una exposición del problema de la planificación de movimientos que incluye la definición de un problema básico, los métodos de modelado que se han utilizado para resolver este problema o ampliaciones sobre el mismo y una clasificación de las técnicas utilizadas para la resolución del problema.

El segundo capítulo se centra inicialmente en modelos basados en esferas relacionados con el trabajo desarrollado en esta Tesis Doctoral y se introducen los modelos esféricos que se utilizan en los siguientes capítulos, las poli-esferas y las esferoides.

El objetivo del tercer capítulo es presentar procedimientos que permitan generar automáticamente el modelado esférico del mundo (el robot y los objetos de su entorno). En concreto se verán la aplicación de técnicas de agrupamiento al modelado esférico y la estructura jerárquica de representación del mundo. También se plantean métodos de reducción de los modelos esféricos de cara a obtener modelos más simples.

El cuarto capítulo hace una revisión de las técnicas de cálculo de distancias y detección de colisiones para modelos tradicionales y posteriormente se extienden estas técnicas a los modelos esféricos presentados anteriormente. En los algoritmos se introducen varias mejoras que permiten obtener buenos resultados para los modelos tratados. Finalmente se presenta una metodología basada en la estructura jerárquica extendida anterior para resolver la detección de colisiones de una célula multi-robot.

En el quinto capítulo se analiza la técnica de planificación de movimientos basada en campos potenciales, con especial interés en su aplicación a los modelos esféricos. La formalización de esta técnica para el caso de un móvil con rotación y el estudio y análisis de la utilización de técnicas de optimización clásicas para resolver la planificación de movimientos con campos potenciales artificiales son los temas tratados en este capítulo.

Finalmente, el epílogo muestra diversos sistemas experimentales realizados con el objetivo de corroborar el trabajo desarrollado mediante unas aplicaciones prototipo, presenta las conclusiones obtenidas en este trabajo, resume las principales aportaciones realizadas y contempla una autocrítica al trabajo y ampliaciones futuras a realizar.

La memoria contiene 242 referencias listadas al final de la misma, si bien no se incluyen por motivos obvios aquellas con una orientación hacia el campo de la ciencia ficción.

√

CAPÍTULO 1. EL MODELADO EN LA PLANIFICACIÓN DE MOVIMIENTOS

Este capítulo, que debe considerarse como un estado del arte en la planificación de movimientos, realiza una exposición del problema de la planificación de movimientos que incluye la definición de un problema básico, los métodos de modelado que se han utilizado para resolver este problema o ampliaciones sobre el mismo y una clasificación de las técnicas utilizadas para la resolución del problema.

> *—La acusación está estableciéndose en una posición desde la cual pueda demostrar que el EZ-27 no era un robot ordinario. Era el primero de su tipo que era ofrecido al público. Era un modelo experimental que necesitaba ser probado sobre la marcha, y la universidad era el único medio decente de proporcionarle esa prueba. Eso parecerá plausible a la luz de los intensos esfuerzos del doctor Lanning para colocar el robot, y la buena voluntad de la U.S. Robots alquilándolo por tan poco. La acusación argumentará luego que las pruebas sobre el terreno demostraron que Easy era un fracaso. ¿Ve ahora usted la finalidad de haber empezado con todo esto?*
> *—Pero EZ-27 era un modelo perfectamente bueno —argumentó Robertson—. Era el que hacía veintisiete en la producción.*
> *—Lo cual es realmente un punto contra nosotros —dijo sombríamente la defensa— ¿Qué fue lo que falló con los primeros veintiséis? Obviamente algo. ¿Por qué entonces no podía haber también algo malo en el que hacía veintisiete?*
>
> "Esclavo en Galeras". Isaac Asimov, 1957.

1.1. El Problema de la Planificación de Movimientos

De forma muy general, un robot es un dispositivo mecánico, que puede adoptar formas muy diferentes (puede tener varias articulaciones, puede disponer de uno o varios brazos, puede o no tener ruedas, etc.), que suele tener actuadores y sensores. La versatilidad y rendimiento de estos instrumentos radica en su capacidad de ser programados para que lleven a cabo tareas diferentes. El robot funcionará en un determinado espacio de trabajo, limitado, dentro del mundo. Dicho espacio no está normalmente desierto, sino que en él existen diversos objetos. Los objetos y el robot están sometidos, cómo no, a las leyes físicas. Para que el robot pueda realizar tareas en el contexto de dicho entorno, necesitará moverse. Este movimiento del robot debe estar restringido por las propiedades, no sólo del robot, sino del entorno que lo envuelve. Es deseable que el robot esté integrado lo mejor posible en el espacio en que opera, de forma que pueda realizar su trabajo sin entrar en conflicto con los elementos que existen a su alrededor, dentro de su área de actuación.

Todo esto enfatiza el hecho de que la planificación de movimientos no consiste en un problema simple y bien definido, sino más bien en una colección de muchos problemas, cada uno de los cuales puede verse como una variante de otro. Es pues poco factible (desde un punto de vista conceptual y, sobre todo, computacional) encontrar una definición general del problema que pueda cubrir todos los casos posibles.

1.1.1. Problema Básico de Planificación de Movimientos

La mayoría de las técnicas de planificación de movimientos se han estudiado y planteado para problemas muy particulares (robots planares en 2D, entornos específicos, ...). Plantear un problema común que sea resuelto por todas estas técnicas de forma eficiente resulta cuanto menos complicado, si no imposible. En esta seción se va a presentar el problema básico de planificación de movimientos que permitirá mostrar los planteamientos generales de las técnicas de planificación de movimientos, que se comentarán posteriormente.

En el *problema básico de planificación de movimientos* definido por Latombe [Lat91] se asume que el robot es el único objeto móvil en el espacio de trabajo y se ignoran las propiedades dinámicas del robot, evitando temas temporales. Así mismo, una simplificación adicional consiste en considerar que el robot es un objeto rígido simple, es decir, un objeto cuyos puntos son fijos unos respecto a otros. Los movimientos se restringen a aquellos sin contacto, para ignorar temas relacionados con la interacción mecánica entre dos objetos físicos. Estas suposiciones transforman el problema de planificación de movimientos físico en un problema puramente geométrico. Los obstáculos son las únicas restricciones a los movimientos de este objeto.

El problema básico de planificación de movimientos resultante de estas simplificaciones es el siguiente:

> *Sea A un objeto rígido simple, el robot, moviéndose en un espacio Euclídeo W, su área de trabajo, representado como* $\Re^n$.
>
> *Sean* $B_1,...,B_k$ *objetos rígidos fijos distribuidos en W, llamados obstáculos.*
>
> *Asúmase que la geometría de A,* $B_1,...,B_k$ *y sus localizaciones en W son conocidas de forma precisa. Asúmase además que ninguna restricción cinemática limita los movimientos de A (se dice que A es un objeto de vuelo libre).*
>
> *El problema se define como: dadas una posición y orientación inicial y una posición y orientación destino de A en W, generar un camino* τ *que especifique una secuencia continua de posiciones y orientaciones de A evitando cualquier contacto con los obstáculos, que empiece en la posición y orientación inicial y termine en la posición y orientación destino.*
>
> *Si dicho camino no existe se debe responder con fallo.*

La Figura 1.1 muestra el planteamiento de un problema básico y una posible solución al mismo. En este ejemplo, el espacio de trabajo del móvil es bidimensional, pero éste dispone de tres grados de libertad, dos ejes de traslación para el cambio de posición $\mathbf{q}_x$, $\mathbf{q}_y$ y un giro para el cambio de orientación $\mathbf{q}_\theta$. Aunque pueda parecer que este ejemplo sólo puede ser significativo para el caso de un AGV con estos tres grados de libertad, esto no es cierto: este ejemplo puede ser la instanciación de cualquier robot con tres grados de libertad mediante sus tres variables de articulación: $\mathbf{q}_x$, $\mathbf{q}_y$ y $\mathbf{q}_\theta$.

Otros autores definen el problema básico de planififcación de movimientos con más diferencias en la notación que en el concepto básico. Por ejemplo, [Til90] lo define como:

> *Se tiene un espacio X de posibles posiciones (o configuraciones) de un robot. Se considera un robot móvil cuya posición* $x \in X$ *puede representarse por sus coordenadas con respecto a un sistema de referencia dado. Como datos se tiene un estado inicial* $\mathbf{x}_0 \in X$ *y un punto final* $\mathbf{x}_f \in X$. *Se desea calcular un camino* $\mathbf{x}(t)$, $t \in [0,t_f]$ *tal que* $\mathbf{x}(0)=\mathbf{x}_0$, $\mathbf{x}(t_f)=\mathbf{x}_f$ *y* $\forall t \in [0,t_f]$ $\mathbf{x}(t)$ *permanece fuera de un conjunto dado de posiciones "inadmisibles" (conjunto de obstáculos), O. Es decir,* $\mathbf{x}(t) \in X\text{-}O$, $t \in [0,t_f]$.

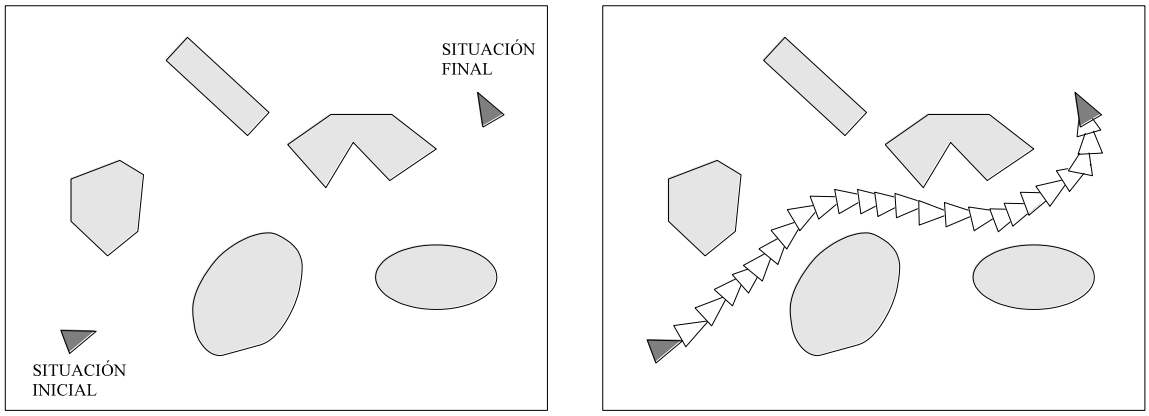


Figura 1.1. Problema Básico de Planificación de Movimientos. *En la Figura de la izquierda se muestra el planteamiento del problema para un espacio bidimensional donde el móvil, un triángulo, dispone de tres grados de libertad, dos de posición y uno de giro. El móvil debe ir de una situación inicial a otra final, sin que durante su movimiento se produzca colisión alguna con ninguno de los obstáculos que se encuentran en la zona de trabajo del móvil. Los obstáculos pueden ser modelados de diferentes formas: convexos, cóncavos, curvos, elípticos, etcétera. En la Figura de la derecha se muestra una posible solución de un camino del móvil libre de colisiones que se visualiza como una secuencia discreta de posiciones y orientaciones intermedias.*

[H&Z94] también enuncian el problema de forma similar:

> *Dado un objeto con una configuración (posición y orientación) inicial, una configuración destino, y un conjunto de obstáculos distribuidos en el espacio, encontrar un camino continuo para el objeto desde la configuración inicial hasta la configuración destino, sin colisionar con ningún obstáculo a lo largo del camino*.

Evidentemente, el problema básico conlleva una gran simplificación. Sin embargo, la navegación de robots móviles (por ejemplo, vehículos autoguiados) son casos prácticamente reales del problema básico. Además, un estudio inicial del problema básico permite aplicar posteriormente técnicas para su solución a problemas más completos y reales.

1.1.2. Extensiones del Problema Básico

Como se ha visto en la definición del problema básico, se han hecho tantas simplificaciones que se pone en duda la utilidad de las soluciones; es decir, puede resultar bastante difícil reducir un problema actual de robótica a una instancia de dicho problema básico, resolverlo y adaptar las soluciones encontradas para satisfacer las

condiciones que el problema original planteaba. Por este motivo se incluye ahora, de forma informal, una breve discusión de varias líneas generales en las que es posible extender el problema.

Múltiples Objetos Móviles

Bajo este epígrafe podemos incluir tres ampliaciones del problema que eliminan la suposición realizada sobre la característica estática de los obstáculos y la unicidad de móvil en el entorno:

- Movimiento de los obstáculos.
  Hay que añadir una dimensión, el tiempo, al espacio de configuraciones del robot para poder obtener una función continua en el tiempo que especifique la configuración del robot en cada instante de tiempo. Si no existe ninguna restricción en la velocidad y la aceleración del robot, y el movimiento de cada obstáculo es conocido de antemano, es bastante directo ampliar los métodos que tratan el problema básico para resolver este nuevo problema [ELP86] y [Fio95]. Sin embargo, si se imponen limitaciones, por ejemplo, en la velocidad o en la aceleración, habrá que traducirlas en restricciones geométricas en la pendiente y la curvatura de la trayectoria como función de la dimensión temporal, y el problema se vuelve más complejo [K&Z86].

- Operación simultánea de varios robots en el mismo espacio (*robots múltiples*). Cada robot supone un obstáculo potencial para los demás.
  Este escenario difiere del que presenta obstáculos en movimiento en que el movimiento de cada uno de los robots debe ser planificado, mientras que el movimiento de los obstáculos no se sometían a control. Una forma de tratar con múltiples robots consiste en considerarlos a todos ellos como un solo robot de varios cuerpos (*single bodied robot*) [BLL89]. Así, el espacio de configuraciones compuesto estará formado por el producto de los espacios de configuraciones de los robots individuales. Otra aproximación consiste en planificar el movimiento de forma separada para cada robot, de forma más o menos independiente de los otros, y considerar, en una segunda fase, la interacción entre caminos [OLP89]. También se puede plantear la programación de los diferentes robots de forma secuencial, pasando a ser considerados obstáculos móviles para los robots aún no programados. Mediante esta aproximación, puede reducirse la complejidad computacional si bien a costa de perder completitud.

- *Robots articulados*, constituidos por varios objetos rígidos unidos por articulaciones.
  Un robot articulado se compone de varios objetos rígidos unidos por articulaciones prismáticas o de revolución (el ejemplo típico sería un brazo-robot). Cada articulación restringe el movimiento relativo entre los dos objetos que une. De una forma bastante directa es posible extender algunos métodos de planificación para el problema básico, al menos de una forma teórica [Bur88]. En la práctica el problema que se presenta es que la dimensión del espacio de configuraciones crece con el número de articulaciones.

Restricciones Cinemáticas

Las únicas restricciones que el problema básico imponía al movimiento del robot venían dadas por la sola presencia de los obstáculos; sin embargo, en muchas ocasiones se quieren imponer, de forma adicional, restricciones cinemáticas al movimiento del robot. Se puede hablar de restricciones holonómicas y no-holonómicas.

Si una configuración es representada por una lista de parámetros de cardinal mínimo, una restricción de igualdad *holonómica* es una relación de igualdad entre estos parámetros que puede resolverse para uno de ellos [ABF88]. Tal relación reduce en 1 la dimensión del actual espacio de configuraciones. Un conjunto de k restricciones holonómicas independientes lo reduce en k dimensiones.

Una restricción de igualdad *no-holonómica* es una ecuación no integrable que incluye los parámetros de configuración y sus derivadas (parámetros de velocidad) [Lau86]. Aunque tal restricción no reduce la dimensión del espacio de configuraciones alcanzables por el robot, sí reduce la dimensión del espacio de, por ejemplo, el espacio de direcciones de velocidad en una configuración dada.

Incertidumbre

Rara vez es posible aceptar el hecho de que la geometría del robot y de los obstáculos, así como la localización de los mismos es conocida con precisión. Tampoco es admisible, en general, la asunción de la capacidad del robot de seguir fielmente los caminos que el planificador genera. Tanto los modelos geométricos como el control del robot son imperfectos, aunque, a menudo, no suele ser preocupante esta imperfección, pues es pequeña en relación con la tolerancia de la tarea a realizar.

En un caso extremo, el robot podría tener poco o ningún conocimiento a priori sobre el espacio de trabajo, y tendría que confiar en sus sensores en tiempo de ejecución para obtener la información necesaria para la realización de la tarea [LMT84]. Pero cuando más incompleto es el conocimiento inicial del entorno, más se aparta el problema del tema de la planificación de movimientos. Una situación intermedia se da cuando existen errores en el control del robot y en los modelos geométricos iniciales, pero estos errores están contenidos dentro de regiones acotadas [Can89].

Objetos Movibles

Antes ya se ha indicado la posibilidad de movimiento de los objetos, pero ahora nos referimos a aquellos objetos que el robot puede desplazar. De esta forma, si no existe un camino que lleve el robot a la posición objetivo, dada una configuración del entorno, el robot podría crear una nueva disposición, moviendo alguno de estos objetos movibles.

Con ello se alternaría entre movimientos de tránsito y movimientos de transferencia; en el primero el robot se movería solo, en el segundo, junto a un objeto movible [L&A89]. Estos dos tipos de movimientos tendrían lugar en espacios de configuraciones diferentes: uno en el espacio de configuraciones del robot, otro en el espacio de configuraciones de la unión del robot y del objeto movible moviéndose con él. El problema principal estribará en determinar qué espacio usar y cuándo cambiar de uno al otro. Agarrar los objetos es una forma en que el robot puede desplazar un objeto movible, pero no la única; también es posible que el robot empuje el objeto. Cada una de estas maneras de mover objetos requiere unas capacidades por parte del robot.

Si se tienen objetos que pueden moverse, podemos hablar de nuevos tipos de problemas de planificación en los que el objetivo a perseguir no es alcanzar una configuración determinada del robot sino una disposición espacial de un conjunto de piezas (los objetos movibles) [ASL89].

1.2. Modelos en la Planificación de Movimientos

La planificación de movimientos, generación automática de trayectorias, detección de colisiones, etc., son aspectos fundamentales hoy en día en tareas de sistemas robotizados. Su desarrollo ha venido influenciado por diversos aspectos, siendo quizá uno de los más importantes el modelado del robot y su entorno o área de trabajo. Un buen modelado puede y debe mejorar la velocidad de cálculo computacional, facilitar el cálculo de distancias y otros aspectos geométricos. A continuación se presenta una perspectiva histórica de la evolución de la planificación de movimientos y los métodos más destacados, haciendo especial hincapié en el tipo de modelado utilizado.

1.2.1. Evolución del Modelado en la Planificación de Movimientos

El primer sistema robotizado móvil con capacidad de planificación, llamado *Shakey*, modelaba el espacio de trabajo y las acciones del robot expresándolas en un lenguaje de *cálculo predicativo* [Nil69]. Este sistema presentó el método *del grafo de visibilidad (Grafo-V)* para planificar los movimientos del robot.

Utilizando matrices en *coordenadas homogéneas*, en [A&P75] se calculan las posiciones de diferentes objetos a partir de relaciones espaciales simbólicas entre los rasgos distintivos de dichos objetos. Una extensión de este método, con más relaciones, permitió implementar posteriormente un lenguaje de programación de robots de alto nivel, el Rapt [PAB80].

Casi simultáneamente, se realizaron los dos primeros intentos para desarrollar sistemas integrados para la programación automática de robots: el Lama [L-P76] y el Autopass [L&W77]. Ambos sistemas partían de la descripción de un montaje mecánico, mediante los modelos geométricos de las diferentes piezas del montaje y las relaciones entre estas piezas en la estructura final. La tarea del sistema era generar automáticamente los programas del robot para montar la estructura. Aunque dichos sistemas nunca se implementaron completamente, contribuyeron a enfatizar la importancia del modelado y razonamiento geométrico en la planificación de robots, así como a resaltar los problemas claves en la planificación de movimientos en el contexto de montajes mecánicos.

Taylor [Tay76] investigó el problema de la planificación de movimientos en presencia de *incertidumbre*. Propuso un método, conocido como *refinamiento esquemático*, basado en técnicas de propagación numérica de la incertidumbre. Un aproximación similar se presentó en [L-P76], mientras que una mejora posterior, basada en técnicas de propagación simbólica en vez de numérica se describe en [Bro82].

Un nuevo método de modelar un robot y su zona de trabajo, presentado en [Udu77], consistía en utilizar un espacio "apropiado" donde el robot se podía comprimir a un único punto. Este método fue expuesto de una forma más sistemática en [LPW79], aplicándolo a robots poliédricos (o poligonales) sin rotación, moviéndose entre obstáculos poliédricos (o poligonales). Con este tipo de modelado, propusieron un algoritmo de planificación de movimientos, considerado como la primera aportación a la planificación de movimientos "exacta". Los obstáculos son modificados en función del tamaño y la orientación del robot. Posteriormente, Lozano-Pérez llamó al modelo resultante *espacio de configuraciones*, noción tomada de la Mecánica que se popularizó dentro de la planificación de movimientos. Uno de los problemas de trabajar en el espacio de configuraciones es lo costoso que puede resultar generar este espacio. [Kav95] utiliza la Transformada de Fourier Rápida (FFT) para transformar los obstáculos a este espacio.

Lozano-Pérez amplió sus trabajos anteriores e introdujo el principio de la técnica de *descomposición celular* aproximada para modelar los obstáculos en el área de trabajo [L-P81] y [L-P83b]. En [Cha82] se utilizó por primera vez una técnica para modelar con una descomposición celular exacta del área de trabajo libre de obstáculos en células convexas. La idea de descomposición celular exacta se extendió posteriormente como *espacio de configuraciones libre*. Con esta técnica, Chatila estudió la planificación de movimientos con conocimiento incompleto para un robot móvil representado como un punto en un área de trabajo bi-dimensional. El planificador trabajaba en línea, actualizando periódicamente la descomposición para considerar la nueva información suministrada por los sensores.

Posteriormente, en [Gou84] se describe la primera implementación de un planificador de movimientos para un brazo robot con articulaciones de revolución con un modelo basado en el método de descomposición celular aproximada. El método consiste en aproximar el espacio libre como un conjunto de células rectangulares obtenidas por la división del rango de posiciones angulares posibles para cada articulación en pequeños subintervalos.

Estos métodos para modelar el espacio de configuraciones libre se han estudiado y refinado por otros investigadores. Por ejemplo, la utilización de conos generalizados en [Bro83a]; la representación en arboles octales del espacio libre tridimensional se utiliza en [Fav84] y [Her86]; corredores rectangulares y sus intersecciones para obstáculos rectangulares en [Mad86]; áreas convexas para un objeto puntual en [S&W86]; cilindros generalizados y polígonos convexos en [KZB85]; discos circulares para crear el diagrama de Voronoi generalizado definido por la localización y la forma de los obstáculos en [T&S89]; árboles de código multivalor, de estructura similar a árboles 2^n, en [P&R91].

Independiente, pero casi simultáneamente, se presentaron dos métodos clasificados dentro de la técnica del mapa de carreteras, la *retracción* [O&Y82] como una aproximación teórica y la técnica del *camino-libre* [Bro83b] como un método de planificación más empírico.

Mason reforzó la importancia de incluir *movimientos para empujar objetos* en ciertas tareas de manipulación [Mas82]. Su trabajo inicial se enfocó en modelar los mecanismos para empujar bajo condiciones *quasi*-estáticas, ampliándolo a la planificación de movimientos.

La primera vez que se describió un método exacto para la planificación de caminos libres de un objeto poligonal con posibilidad de trasladarse y rotar en un espacio bi-dimensional con obstáculos poligonales fue en el primer artículo de los cinco que componen la llamada Piano Movers' Problem Series [S&S83a]. En el segundo artículo de esta serie [S&S83b] se establece el primer límite superior en la complejidad temporal de la planificación de movimientos en un espacio libre semi-algebraico de cualquier dimensión fija.

En [L&P83] los objetos se modelan mediante superficies planas, cilíndricas y esféricas para poder definir el posicionamiento de la garra de una mano mecánica sobre objetos. Ello dio lugar a un planificador que trabaja en dos fases: en una primera, el planificador genera varios agarres y valora su viabilidad usando un conjunto de pruebas empíricas. En una segunda fase, comprueba la accesibilidad global para cada agarre, planificando un movimiento en el espacio de configuraciones de la mano.

En [LMT84] se describe el *encadenamiento hacia atrás pre-imagen* en planificación de movimientos con incertidumbres ocasionadas por inexactitud en el modelado de los objetos, errores en el control y ruido en la sensorización. Por primera vez, esta aproximación consideró movimientos completos en la planificación, por lo que se considera la primera aproximación válida a la *planificación de movimientos correcta* (*fine motion planning*). En el artículo original, la aproximación era esencialmente un esquema teórico, no implementado, pero el impacto de este trabajo en el campo de la planificación de movimientos ha sido tal, que no es extraño encontrar referencias a esta técnica denominándola problema LMT o planificación LMT. Desde entonces, la teoría e implementación de esta técnica ha avanzado bastante.

Una aproximación implementada a la planificación de movimientos con incertidumbre, basada en el *aprendizaje inductivo* se describe en [D&L84]. La técnica consiste en ejecutar la misma tarea varias veces, combinando las trazas de ejecución en una estrategia general.

Otro método de representar el espacio libre, en el que Khatib fue el pionero, es la técnica de los *campos potenciales* [Kha80] y [Kha86]. Aunque su primera implementación fue como un módulo para evitar colisiones en tiempo real dentro del controlador de un robot, la técnica se extendió para modelar el espacio de trabajo en la planificación de movimientos, estando actualmente en gran auge. Con una variante de los campos potenciales, en [F&T87] y [F&T89] se encuentra la implementación de un sistema práctico capaz de planificar movimientos de un brazo manipulador con ocho articulaciones moviéndose entre tubos verticales. [K&V88] modelan el entorno con una función potencial artificial supercuadrática con n-elipses modificadas como contornos isopotenciales. Otra forma de modelar el entorno para crear una función potencial artificial es mediante ecuaciones frontera de politopos [H&A88].

El método de campos potenciales, como método local, presenta el inconveniente de la posible aparición de mínimos locales. Existen muchos estudios que tienden a utilizar modelos que eviten este problema. La utilización de una *función de navegación*, función potencial libre de mínimos locales, se debe a Koditschek [Kod87], estudiada profundamente junto con Rimon [R&K88], [R&K89], [R&K90] y [Rim91]. Combinando el método de campos potenciales con técnicas aleatorias para escapar de mínimos locales, en [B&L89a], [B&L89b] y [B&L90] se explica un planificador para generar movimientos libres para robots con un elevado número de grados de libertad. En [K&K91] se utiliza una representación potencial armónica para evitar mínimos locales.

[CBW90] sustituyen el modelo de distribución de carga de Khatib por un potencial bajo condiciones de frontera de Diritchlet, obteniendo una mejora en el método, pese a existir la posibilidad de presentar zonas planas en el modelo. Una variante diferente se presenta en [T&B91], que utiliza campos potenciales eléctricos escalares en 2D, modelando los caminos como flujos de corriente desde una fuente de corriente (punto origen) a otra de signo contrario (punto destino) en un medio conductor evitando elementos no conductores (obstáculos).

Laumond [Lau86] y [Lau87] consideró el problema de planificar movimientos libres para robots móviles con un modelo *no-holonómicos*, similares a automóviles. Produjo como resultado interesante que un camino libre para un robot de vuelo libre en un espacio bi-dimensional siempre se puede transformar en un camino libre para un robot no-holonómico teniendo la misma geometría y moviéndose en el mismo espacio de trabajo, introduciendo maniobras simples de marcha atrás. Li y Canny [L&C89] fueron los primeros en aplicar herramientas desarrolladas en teoría de control de sistemas no lineales a robots no-holonómicos, haciendo posible generalizar los resultados de Laumond.

En [L-P87] se describe un sistema robótico implementado, el Handey, que integra un planificador de agarre y un planificador de movimientos. Este sistema es capaz de planificar los movimientos de un robot manipulador para construir montajes simples hechos de objetos poliédricos, y ejecutar los movimientos suponiendo que no hay incertidumbre. Parte del conocimiento inicial acerca del área de trabajo se obtiene usando un sistema de visión. Además, el sistema es capaz de planificar el re-agarre de un objeto, si ningún agarre en la localización inicial del objeto es compatible con un agarre en la localización final.

La técnica del Grafo-V, introducida como la primera técnica en la planificación de movimientos, ha sido utilizada durante muchos años. Recientemente, se ha presentado un método que aumenta la flexibilidad de dicha técnica, el *grafo tangente (Grafo-T)* [L&A94]. El Grafo-T permite incluir obstáculos modelados con caras curvas, no solo poligonales como el Grafo-V, y se basa en considerar las rectas de tangencia entre los obstáculos. En el mismo artículo también se presenta el grafo tangente extendido, que depende de los obstáculos en el área de trabajo real, no en el espacio de configuraciones. De esta forma se dispone de mayor flexibilidad para diferentes tamaños de robot y/o distancias de seguridad.

Como se puede ver, existen un gran número de métodos para resolver el problema básico de planificación de movimientos, si bien no todos lo resuelven con completa generalidad. Por ejemplo, algunos requieren que el área de trabajo sea bidimensional y los objetos poligonales. En grandes rasgos, los métodos se basan en tres técnicas básicas: el mapa de carreteras, la descomposición celular y los campos potenciales artificiales. Normalmente, estas técnicas se basan en el concepto del *espacio de configuraciones*, tratado en la siguiente sección.

1.2.2. Modelado en el Espacio de Configuraciones

Una de las formas más utilizadas para modelar un robot y su entorno es la utilización del espacio de configuraciones. El espacio de configuraciones es un espacio apropiado, por ejemplo, el espacio de configuraciones de un robot, que permite representar el robot como un punto, así como realizar transformaciones de los obstáculos a este espacio. Estas transformaciones convierten el problema de planificación de movimientos de un objeto dimensional al de planificar movimientos de un punto. Además, las restricciones de los movimientos del robot son más explícitas en el espacio de sus configuraciones. Por contra, como inconveniente frente a trabajar en el espacio cartesiano, cuando el entorno es variable, resulta muy costoso utilizar este modelo para aplicaciones *on-line*.

Considérese el problema básico. Sea el robot A, en una posición y orientación dada, descrito como un subconjunto compacto de W, y los obstáculos $B_1,...,B_k$ como subconjuntos cerrados de W. Sea F_A el sistema de coordenadas local de A y F_W el sistema de coordenadas de W. F_A es un sistema móvil mientras que F_W es fijo. Por definición, si

A es rígido, cada punto a de A tiene una posición fija respecto a F_A, pero su posición en W depende de la posición y orientación de F_A respecto a F_W. Como los obstáculos $B_1,...,B_k$ son objetos rígidos y fijos en W, cada punto de cualquiera de los obstáculos tiene una posición fija respecto a F_W.

La configuración de un objeto es la especificación de la posición de cada punto de este objeto respecto a un sistema de referencia fijo. Por tanto, la configuración **q** de A consiste en determinar la posición Υ y orientación Θ de F_A respecto a F_W. El espacio de configuraciones de A es el espacio C de todas las configuraciones posibles de A. El espacio de W ocupado por A en la configuración **q** es $A_{\mathbf{q}}$, mientras que $a_{\mathbf{q}}$ representa el punto a de A para la configuración **q** en W.

Se puede describir una configuración mediante una lista de parámetros reales. Por ejemplo, la posición Υ es el vector de n coordenadas del origen de F_A respecto a F_W. La orientación Θ se puede describir como una matriz n×n cuyas columnas son los vectores unitarios a lo largo de los ejes de F_A expresados respecto a F_W. Realmente, esta matriz de rotación se puede obtener mediante una secuencia de rotaciones básicas. El número de rotaciones básicas necesarias para poder obtener cualquier orientación depende del espacio de trabajo en que se esté trabajando. Por ejemplo, para un espacio bi-dimensional, existe un único parámetro de rotación, mientras que para el espacio tri-dimensional, hay tres parámetros de orientación. La dimensión de C, m, se define como el número mínimo de parámetros independientes necesarios para definir **q**. Por tanto, $\mathbf{q}=(\Upsilon,\Theta)$ se representa únicamente con un vector de m parámetros. Por ejemplo, para n=2, m es tres (dos parámetros para posición y uno para orientación), mientras que para n=3, m es seis (tres para posición y tres para orientación).

Se define un camino de A desde una configuración inicial $\mathbf{q}_i$ a una destino $\mathbf{q}_d$ como una función continua

$$\tau:[0,1] \rightarrow C \quad \text{con} \quad \tau(0)=\mathbf{q}_i, \qquad \tau(1)=\mathbf{q}_d$$

La continuidad de τ implica que para todo $t_0 \in [0,1]$ se cumple

$$\lim_{t \rightarrow t0} d(\tau(t), \tau(t_0)) = 0$$

donde $d:C\times C \rightarrow \Re$ es una función distancia, definida por ejemplo como

$$d(\mathbf{q}, \mathbf{q}') = \max_{a \in A} \left\| a_{\mathbf{q}} - a_{\mathbf{q}'} \right\|$$

siendo $\|a-b\|$ la distancia Euclídea entre dos puntos a, b en $\Re^n$.

A es un *objeto de vuelo libre* si, en ausencia de obstáculos, cualquier camino definido así es realizable por A.

La definición previa de un camino no toma en consideración a los obstáculos. Lo que se pretende es determinar los movimientos que solucionan el problema básico cuando existen obstáculos en el área de trabajo. Para ello, hay que transformar los obstáculos al espacio de configuraciones y obtener el subconjunto de C formado por las configuraciones libres de contacto.

Cada obstáculo B_i, i=1,...,k en el espacio W se transforma en una región en C, llamada *C-obstáculo*, dada por $CB_i = \left\{ \mathbf{q} \in C : A_{\mathbf{q}} \cap B_i \neq 0 \right\}$.

Cuando A es un punto (*robot puntual*) no tiene sentido hablar de su orientación, por lo que el espacio de configuraciones de A es una copia de $\Re^n$, siendo un espacio Euclídeo. Los C-obstáculos son idénticos a los obstáculos en W.

Cuando A es un disco o un objeto dimensional que se puede trasladar libremente sin rotación (el origen de F_A puede seguir cualquier camino en W pero siempre con una orientación fija respecto a F_W), el espacio de configuraciones también es $\Re^n$, pero los C-obstáculos se obtienen ampliando los obstáculos con la forma de A, tal como se ilustra en la Figura 1.2.

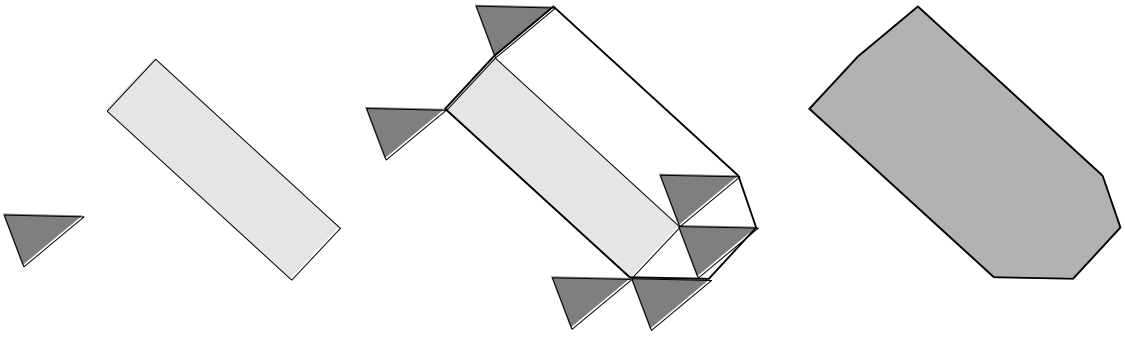


Figura 1.2. Método de Obtención de un C-obstáculo. *El obstáculo, en este caso un rectángulo, se debe agrandar con la zona delimitada por el área barrida por un punto de referencia del móvil, en este caso un triángulo, cuando se consideran todas las posibles posiciones en las que el móvil mantiene contacto con el obstáculo. En el ejemplo, el punto de referencia del móvil es el vértice del ángulo más agudo del triángulo y el C-obstáculo obtenido se muestra an la gráfica de la derecha. Planificar el movimiento del móvil con el obstáculo original es similar a planificar el movimiento de un punto con el C-obstáculo así generado.*

La unión de todos los C-obstáculos, $\bigcup_{i=1}^{k} CB_i$, se llama *región C-obstáculo* y el conjunto

$$C_l = C \sim \bigcup_{i=1}^{k} CB_i = \left\{ \mathbf{q} \in C : A_{\mathbf{q}} \cap \left(\bigcup_{i=1}^{k} CB_i \right) = 0 \right\}$$

se denomina *espacio de configuraciones libre*. Cualquier configuración de C_l se llama *configuración libre*.

Un *camino libre* entre dos configuraciones $\mathbf{q}_i$ y $\mathbf{q}_d$ es una función continua $\tau:[0,1] \rightarrow C_l$, con $\tau(0)=\mathbf{q}_i$ y $\tau(1)=\mathbf{q}_d$. Dos configuraciones pertenecen a la misma componente conectada de C_l, si y sólo si están conectadas por un camino libre.

Dadas unas configuraciones inicial y final, el problema básico de la planificación de movimientos consiste en obtener un camino libre entre estas dos configuraciones si es que pertenecen a la misma componente conectada de C_l, o devolver fallo en otro caso.

1.3. Técnicas de Planificación de Movimientos

Han sido muchas las técnicas que se han ido investigando sobre el tema de la planificación de movimientos. Como es normal, cada una de ellas tiene sus aspectos positivos, pero también sus restricciones. Las limitaciones pueden consistir en que la implementación de una técnica en un caso real no es suficientemente eficiente, lo que impide su aplicación en tiempo real. O es posible, como sucede en muchos casos, que se parta de hipótesis muy simplistas, como puede ser suponer un espacio bi-dimensional, objetos poligonales, etc.

En esta sección se da una pequeña visión de las principales técnicas existentes. A pesar de que existen muchas técnicas, la mayoría de ellas se pueden clasificar dentro de tres categorías: mapa de carreteras, descomposición celular y campos potenciales. Una descripción muy completa se puede encontrar en [Lat91].

1.3.1. Mapa de Carreteras

El método del *mapa de carreteras* (*roadmap*) consiste en obtener la conectividad del espacio libre del robot en una red de curvas unidimensionales en el espacio libre C_l. Una vez se ha construido el mapa *R*, se utiliza como un conjunto de caminos estándares, por lo que la planificación de movimientos se reduce a conectar las configuraciones inicial y final a puntos de *R* y buscar un camino en *R* que una esos puntos. El camino solución, si existe, es la concatenación de tres subcaminos: uno conectando la configuración inicial

con el mapa, otro contenido en el mapa y un tercero conectando el mapa a la configuración destino.

Varios métodos se han propuesto basados en esta idea general, que calculan diferentes tipos de mapas de carreteras, tales como el *grafo de visibilidad*, el *diagrama de Voronoi*, la *red de camino libre* y *el mapa de la silueta*. A continuación se comenta uno de ellos.

El método del grafo de visibilidad es uno de los primeros métodos de planificación de movimientos. Se utiliza fundamentalmente en espacios de configuraciones bidimensionales con C-obstáculos poligonales, ya que su extensión a dimensiones mayores resulta sumamente compleja. Es un grafo no dirigido *G* cuyos nodos son las configuraciones inicial y final y todos los vértices de todos los C-obstáculos. Los arcos de *G* son todos los segmentos rectos que conecten dos nodos sin intersección con algun C-obstáculo. La Figura 1.3 muestra un ejemplo de grafo de visibilidad. El mapa de carreteras *R* consiste en aquellos arcos de *G* formados por sólo los vértices de los C-obstáculos. El resto de arcos conectan el mapa con las configuraciones inicial y final. El camino más corto (según métrica Euclídea en $\Re^2$) en *G* que una las configuraciones inicial y final será el camino libre resultado (realmente es un *camino semi-libre*, ya que se permiten contactos con los obstáculos). El camino, si existe, es una línea poligonal que conecta la configuración inicial con la final através de los vértices de los C-obstáculos.

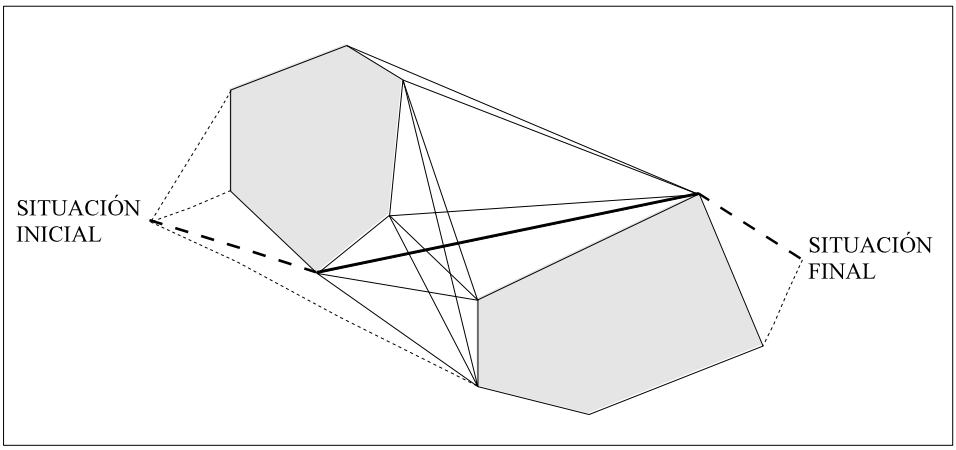


Figura 1.3. Grafo de Visibilidad. *El espacio de configuraciones libre está limitado externamente por un rectángulo e internamente por dos polígonos (C-obstáculos). El mapa de carreteras se forma uniendo mediante rectas todos los pares de vértices de los C-obstáculos visibles (dos vértices son visibles cuando la recta que los une no intersecciona con el interior de ningún C-obstáculo). Las configuraciones inicial y final se unen a aquellos vértices visibles desde estas configuraciones (líneas a puntos). Con un mayor espesor se muestra el camino más corto obtenido sobre este mapa.*

1.3.2. Descomposición Celular

Los métodos de *descomposición celular* (*Cell-Decomposition*) son quizá los más utilizados durante la aparición de los primeros estudios sobre la planificación de movimientos. Consisten en descomponer el espacio de configuraciones libre del robot en regiones simples, llamadas células, tal que resulte fácil generar un camino libre entre dos configuraciones de la misma célula. Para representar las relaciones de adyacencia entre las células se construye un grafo no dirigido. Este grafo, llamado grafo de conectividad, tiene como nodos las células obtenidas del espacio de configuraciones libre y existe un arco entre dos nodos si y sólo si sus células correspondientes son adyacentes. Una búsqueda en el grafo produce una secuencia de células llamada canal. Un camino libre continuo se puede calcular a partir de esta secuencia. La descomposición celular puede ser exacta y aproximada:

- Los métodos de *descomposición celular exacta* (Figura 1.4) descomponen el espacio de configuraciones libre en células cuya unión produce exactamente este espacio. La frontera de una célula suele corresponder a algun aspecto crítico, tal como un cambio repentino en las restricciones del movimiento del robot. El espacio de configuraciones libre se descompone en cálulas. Un grafo de conectividad permite obtener un canal, a partir del cual se puede obtener un camino libre.

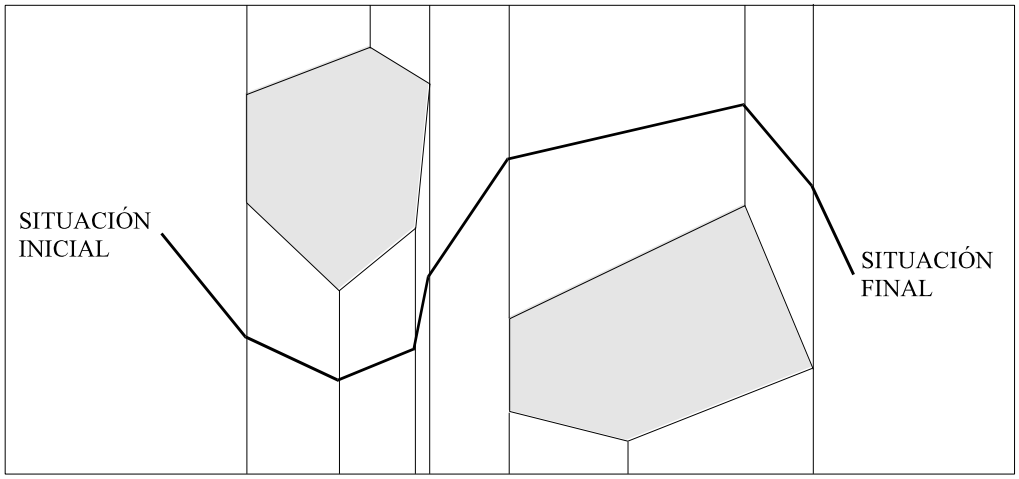


Figura 1.4. Descomposición Celular Exacta. *El espacio de configuraciones es el mismo que en la Figura 1.3, descomponiéndose exactamente en células trapezoidales. Estas células se consiguen mediante proyecciones verticales de los vértices de los C-obstáculos. Etiquetando estas células y formando un grafo de conectividad con ellas, es posible obtener un canal de células que conecten la célula donde se está la configuración inicial con aquella donde está la destino. Sobre este canal se determina un camino que lleve de la configuración inicial a la destino, por ejemplo, mediante los puntos medios de las fronteras de cada dos células sucesivas del canal.*

- Los métodos de *descomposición celular aproximada* producen células de formas predefinidas cuya unión está estrictamente incluida en el espacio de configuraciones libre. La frontera de una célula no caracteriza una discontinuidad de ningún tipo ni tiene ningún sentido físico. La Figura 1.5 muestra un ejemplo de método de descomposición celular aproximada basada en la representación de árbol cuaternario (árbol octal en tres dimensiones). Normalmente, este método opera de forma jerárquica, usando una baja resolución al principio y refinándola hasta que se encuentre un camino adecuado o se alcance un límite de resolución.

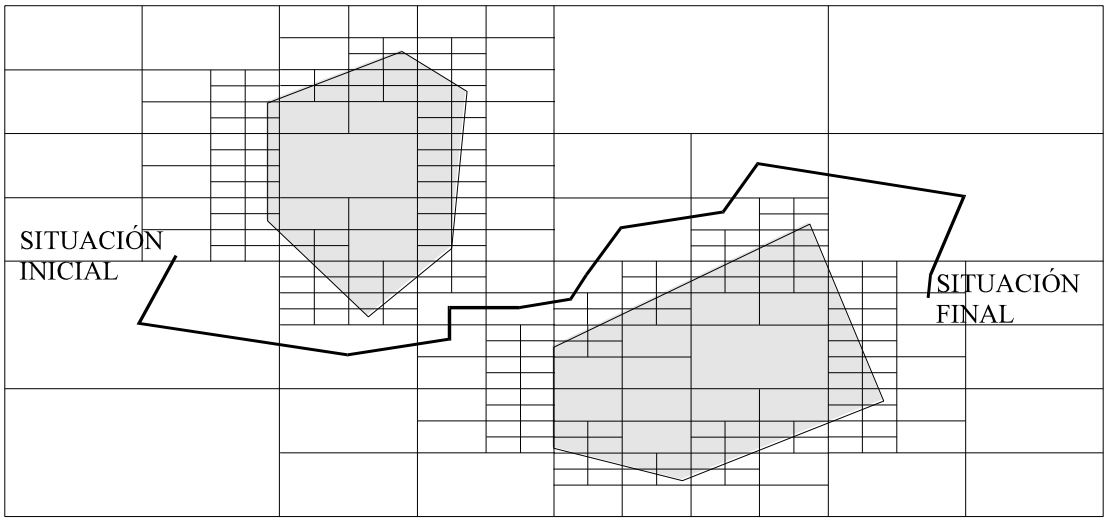


Figura 1.5. Descomposición Celular Aproximada. *El espacio de configuraciones es el mismo que en la Figura 1.3. El rectángulo exterior se descompone recursivamente en rectángulos más pequeños, generándose cuatro rectángulos en cada descomposición, por lo que esta técnica se denomina quad-tree, al poder ser representado por un árbol de grado 4 (en 3D se utilizan cubos, denominándose oct-tree al poder ser representado por árboles de grado 8). Los nuevos rectángulos se marcan como llenos (los que caen completamente dentro de un C-obstáculo), vacios (los que caen completamente en el espacio libre) o parciales (aquellos que una parte cae dentro de un C-obstáculo y otra fuera), de forma que solo se subdividen los parciales. Tras terminar la subdivisión para un nivel de precisión dado, sólo las células vacías se utilizan para generar el grafo de conectividad. Si una búsqueda en el grafo termina con éxito, se puede calcular fácilmente un camino libre, por ejemplo uniendo los centros de estas células. Si no es así, o bien la resolución es insuficiente o bien no existe ningún camino libre que conecte las configuraciones inicial y final.*

1.3.3. Campos Potenciales Artificiales

En la técnica de *campos potenciales artificiales* el robot o móvil es tratado como una partícula bajo la influencia de un campo potencial artificial U(**q**) producido por la configuración destino y los obstáculos. U(**q**) se define típicamente (aunque no es la única aproximación) como la suma de un *potencial atractivo*, que atrae el robot hacia la configuración final, y un *potencial repulsivo*, que tiende a alejarlo de los obstáculos. Bajo estos potenciales, el movimiento se planifica de forma iterativa, determinando secuencias de configuraciones del móvil $\mathbf{q}_i$. En cada iteración, la fuerza artificial $\mathbf{F}(\mathbf{q}_i)=-\nabla(U(\mathbf{q}_i))$, inducida por la función potencial en la configuración actual, es utilizada para determinar la siguiente configuración, ya que se considera como la dirección más prometedora de movimiento hacia el destino(*). La Figura 1.6 muestra la noción de potenciales artificiales.

Cómo se mueve el móvil inmerso en un campo potencial artificial también se explica utilizando comparaciones. Si observamos un campo potencial dibujado (por ejemplo, en un espacio 3D donde la componente en Z represente el valor del campo para cada punto (x,y)), se puede ver como éste consiste en una serie de elevaciones más notorias que corresponden a los obstáculos, y una pendiente más o menos pronunciada, pero continua, con el punto más bajo en la configuración final. Visto así, el camino que seguiría el móvil colocado en cualquier punto de este campo sería la que buscase siempre el punto de mayor pendiente, no pudiendo subir a las elevaciones que representan obstáculos. Y de esta forma tan sencilla se entiende cómo esta técnica permite que el móvil se mueva hacia el destino, evitando colisionar con los obstáculos. Una experiencia bastante cotidiana acabará por dejar clara la idea: una pelota dejada en lo alto de una pendiente tiende, por leyes físicas, a descender, buscando siempre la menor altura en la dirección de la mayor pendiente. Símiles hidráulicos o eléctricos también resultan ilustrativos.

Khatib [Kha86] fue el primero en estudiar esta técnica, desarrollada inicialmente para aplicaciones "on-line", en las que el robot no tiene un modelo previo de los obstáculos sino que los detecta durante el movimiento. Como tal aplicación en tiempo real, es posible la presencia de mínimos locales que atrapen el móvil en una determinada configuración. Con un modelo anterior del espacio de trabajo, es posible utilizar técnicas basadas en campos potenciales, que sean locales con una visión global. La técnica de planificación basada en potenciales puede o no asumir una función potencial específica. Además es aplicable tanto al espacio cartesiano 3D como al espacio de configuraciones.

(*) El gradiente proporciona la dirección de la pendiente más pronunciada hacia arriba. La dirección opuesta del gradiente es la dirección del descenso más pronunciado. Esta es la razón por la que se considera el gradiente cambiado de signo, ya que se busca un mínimo, no un máximo.

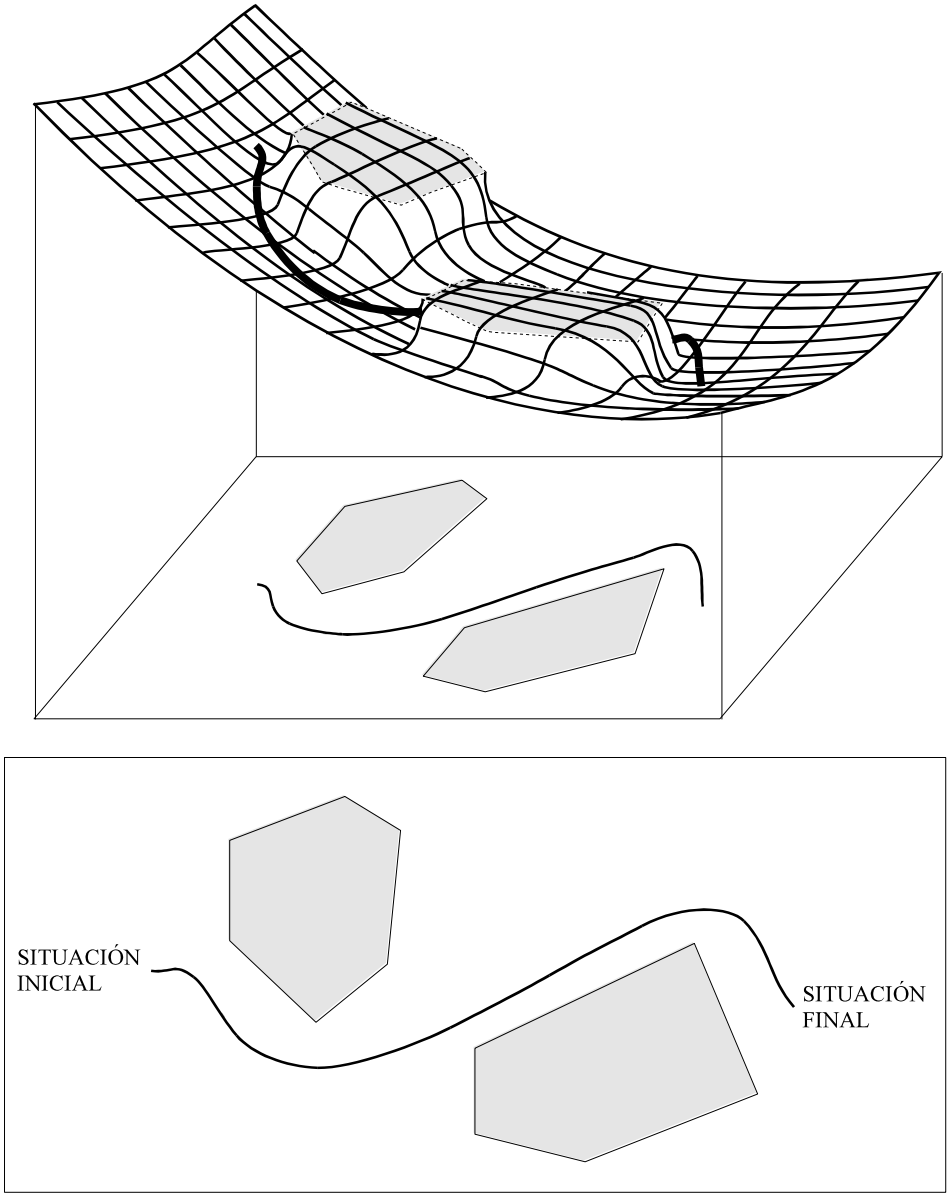


Figura 1.6. Campos Potenciales Artificiales. *El espacio de configuraciones es el mismo que en la Figura 1.3. En la gráfica superior se muestra una representación tridimensional del potencial resultante como suma de un potencial atractivo, que tiene un mínimo global en la configuración destino y un potencial repulsivo no nulo sólo a cierta distancia de los C-obstáculos. Un camino entre las configuraciones inicial y destino se construye siguiendo el gradiente negado del potencial total.*

1.3.4. Métodos Locales y Globales

Los métodos del mapa de carreteras y de descomposición en celdas reducen el problema de encontrar un camino libre a realizar una búsqueda en grafo previo análisis de la conectividad del espacio libre. Los métodos de campos potenciales no incluyen ningún tipo de preprocesamiento para capturar la conectividad del espacio libre. En cada paso, se mueve de una a otra, utilizando como criterio para seleccionar la siguiente configuración el gradiente de la función potencial. Ahora bien, tal gradiente depende normalmente del contenido del espacio en una vecindad de la configuración actual del robot. Por eso los métodos de campos potenciales se suelen denominar *locales*, mientras que los métodos de roadmap y descomposición en celdas se conocen como métodos *globales*.

De esta forma, un método de campos potenciales podría ejecutarse mientras el robot se mueve, calculando en cada paso el gradiente negativo del potencial, definido como una función tanto de la distancia a la configuración destino como de las distancias a los obstáculos, medidas éstas por medio de sensores de alcance. Sin embargo el carácter local de los métodos basados en campos potenciales no es completo. Por ejemplo, podría utilizarse una técnica que evite mínimos locales (como se ha descrito anteriormente). En dicho caso, calcular la función potencial con ausencia de mínimos locales necesita considerar la geometría del espacio libre, no en una vecindad, sino en conjunto. Y en este sentido este método sería tan global como el de roadmap o el de descomposición en celdas.

De forma similar, si los métodos de descomposición en celdas o roadmap se restringen de forma que se apliquen tan sólo en un subconjunto del espacio de configuraciones alrededor de la configuración actual del móvil, lo que se habrá hecho no es más que hacer locales estos métodos globales. Es posible actuar de esta forma, tratando configuraciones intermedias como configuraciones finales dentro de un ámbito local y obteniendo el camino completo mediante concatenación de estos caminos locales.

Además existen técnica que no se pueden catalogar en uno sólo de estos métodos, como [H&Z94] que combinan, de una interesante forma, los métodos de descomposición en celdas y campos potenciales artificiales. Empiezan por reducir el problema a 2-D indicando que aunque el robot y los obstáculos son tridimensionales, pueden proyectarse y considerarlos objetos bi-dimensionales, asumiendo que si no existe colisión en el caso bi-dimensional, tampoco lo habrá en el problema 3-D original. Los obstáculos son modelados como polígonos (cóncavos o convexos) y el móvil es rectangular. Por supuesto asumen un entorno de trabajo completamente conocido a priori. La idea fundamental consiste en utilizar un método de descomposición celular jerárquica, con celdas hexagonales, para encontrar el camino libre de colisiones. El uso de celdas hexagonales se justifica por las ventajas que ello comporta (posibilidad de dividir recursivamente un hexágono, simplificación del procedimiento de búsqueda, las distancias del centro de un hexágono a los centros de sus seis celdas vecinas es el mismo, etc). El campo potencial artificial se utiliza como función para guiar la búsqueda. A las celdas se les asocia un atributo que describe su estado. Cuando se tiene que pasar de una celda a alguna de sus vecinas se calcula el potencial en el centro de las

celdas candidatas (las que tengan un estado que permita que el móvil pase a ellas). Será seleccionada la celda con el menor potencial. Realizan un tratamiento para evitar mínimos locales, por medio del uso de un atributo que permita recordar por qué celda se ha pasado, para evitar ciclos infinitos, o asignando un potencial infinito a la celda que es seleccionada por segunda vez.

√

CAPÍTULO 2. MODELOS ESFÉRICOS

Este capítulo, tras comentar inicialmente otros trabajos con modelos basados en esferas, presenta la definición matemática de unos modelos esféricos nuevos, las poli-esferas y las esferoides, que serán la base del desarrollo de procedimientos de detección de colisiones y planificación de movimientos en capítulos posteriores.

Había desatornillado la placa pectoral del más cercano a ellos mientras hablaba, e insertó la pequeña esfera de unos cinco centímetros que contenía la pequeña chispa de energía atómica que era la vida de un robot. Le costó encajarla, pero lo consiguió al fin, y volvió a atornillar la placa con un esfuerzo. Los radio controles de los modelos más modernos no se conocían todavía hace diez años. Luego repitió la operación con los otros cinco.

"Círculo Vicioso". Isaac Asimov, 1942.

2.1. Modelado Tradicional

En el modelado geométrico y sólido existen principalmente dos esquemas de representación [Hof89], la representación de fronteras (*B-rep*) y la geometría sólida constructiva (*CSG*). Las características básicas de cada tipo son:

- *B-rep*: representa un sólido mediante la descripción de su superficie, para obtener información completa del interior y el exterior de un objeto. Esta representación da dos tipos de descripciones, la geométrica (parámetros necesarios para describir caras, aristas y vértices) y la topológica (adyacencias e incidencias de vértices, aristas y caras). La descripción geométrica se basa en una descripción analítica de vértices (coordenadas), aristas (ecuaciones de las aristas) y caras (semiespacios).

- *CSG*: representa los sólidos mediante la teoría de conjuntos con expresiones lógicas sobre objetos primitivos. Como operaciones en *CSG* se tienen la rotación, la traslación, la unión regularizada, la diferencia regularizada y la intersección regularizada. Las primitivas típicas de *CSG* son paralepípedo, esfera, cilindro, cono, prisma triangular y en algunos casos el toro. Para crear una primitiva sólo se requiere especificar sus dimensiones, ya que se suelen definir mediante instanciación de primitivas puras. Cada primitiva tiene un sistema de coordenadas local asociado que se utiliza para situarla respecto a un sistema de coordenadas global (del mundo) mediante traslaciones y rotaciones. Un objeto se genera localizando primitivas y realizando operaciones lógicas regularizadas entre ellas.

Un método mixto entre estos dos esquemas de modelado consiste en representar las primitivas mediante objetos *B-rep* y realizar operaciones *CSG* sobre éstas.

Otras técnicas, como las descomposiciones regular e irregular del espacio, arboles octáles (*oct-trees*) y la enumeración ocupacional del espacio suelen utilizarse para aplicaciones muy concretas, como es el caso de análisis por elementos finitos.

Las primitivas típicas de *CSG*, excepción hecha del toro, son todas convexas, por lo que se pueden definir mediante politopos. Un politopo es el casco convexo de un conjunto de puntos. Si **P** es un conjunto de puntos $\{p_0,...,p_n\}$, el casco convexo de **P** es un conjunto infinito de puntos expresado como

$$\mathrm{Co}\mathbf{P} = \left\{ p \in \Re^3 : p = \sum_{i=0}^{n} \lambda_i p_i, \quad p_i \in \mathbf{P}, \quad 0 \le \lambda_i, i = 0,\ldots,n, \quad \sum_{i=0}^{n} \lambda_i = 1 \right\}$$

Por tanto, un politopo puede verse como un poliedro convexo generado a partir de sus vértices. Téngase en cuenta que objetos con caras curvas se pueden representar mediante aproximaciones poliédricas. Así, cilindros, conos, esferas, etc. están generados de manera

aproximada por politopos. Por otra parte, los objetos no convexos siempre se pueden descomponer en la unión de objetos convexos.

2.2. Modelos Esféricos

En el modelado tradicional se trabaja con un modelo basado en un elemento básico de volumen nulo: el punto. Las técnicas de modelado esférico se basan en considerar un elemento básico casi igual de sencillo que el punto pero con volumen propio: la esfera. La utilización de la esfera permitirá mantener la misma complejidad que el modelo tradicional pero aumentando las posibilidades del modelo, ya que su elemento básico es más completo. Por ejemplo, con el modelo basado en puntos, se requieren al menos cuatro elementos básicos para definir un volumen, ya que un punto no tiene volumen, dos puntos definen un segmento sin volumen y tres puntos dan a lo sumo un triángulo, es decir, un área. Otro aspecto importante es que trabajar con modelos esféricos debe incluir siempre al modelado tradicional definido con puntos, ya que la esfera de radio nulo es un punto.

La esfera tiene tres propiedades intrínsecas que la hacen muy útil para el modelado de objetos en el marco en que nos encontramos:

- La esfera queda descrita por un único parámetro que la caracteriza completamente, el radio
- El movimiento de una esfera se describe completamente mediante el movimiento de un único punto, el centro.
- Para la esfera no existen rotaciones, los movimientos son siempre de traslación.

Muchos problemas en diferentes áreas se han resuelto con mayor facilidad para el caso particular de la esfera, por resultar muy costosos con otro tipo de objetos. De hecho es posible encontrar numerosos antecedentes en los que se usa la esfera como primitiva de representación en campos tales como gráficos por ordenador, geometría computacional y en algunos casos de robótica. Sin embargo, en la mayoría de casos se utilizan círculos en el plano o aproximaciones groseras del objeto con una única esfera. En otros casos se representa un modelo con un conjunto de esferas sin posibilidad de mejorar la representación con mayor precisión [Abr88], [Tha88]. En [Dai89] se utiliza el casco esférico (esfera envolvente) para aproximar objetos, y cuando éstos están en movimiento, el casco esférico generalizado envuelve el casco esférico de las posiciones inicial y final. Elementos basados en esferas, como cilindros con semiesferas en sus extremos, se ha utilizado en [J&G85], [GJK88] y [S&M89], ya que dan una buena aproximación de los elementos de un brazo-robot. Otros autores que han utilizado representaciones esféricas sencillas son [Wid74], [O&B79], [HSS83], [S&S83c] y [O&Y82].

Más recientemente, otros investigadores como Bonner [Bon89] y del Pobil [dPo91] han trabajado con modelos esféricos para aplicaciones de detección-evitación de colisiones y planificación de movimientos en sistemas robotizados. Posteriormente, Hubbard ([Hub93], [Hub94] y [Hub95]) y Zelinsky ([Zel94]) presentaron, en el campo de los gráficos por computador, modelos esféricos, denominados *sphere-trees* y *distance space bubbles*, muy similares ambos al modelo de del Pobil que se comenta acontinuación. [dPo91] fijó las características adecuadas que debe cumplir un buen modelado esférico:

- Debe ser *mejorable*, es decir, a partir de un modelo se puede obtener otro con menor error. El error se define como la diferencia entre la representación del modelo y el objeto real. Por tanto, la representación debe converger a un modelo de error nulo.
- Debe ser *óptimo*, es decir, el que tenga el menor error para el número de esferas que lo componen.
- Debe ser *equilibrado*, es decir, el error debe estar repartido homogéneamente sobre el objeto, sin zonas de acumulación de error.

Para conseguir estos objetivos, Bonner y del Pobil han utilizado modelos esféricos discretos jerárquicos, si bien desde diferentes perspectivas. En el siguiente apartado se resumen los trabajos realizados por estos autores.

2.2.1. Modelos Esféricos Jerárquicos-Discretos

Los modelos esféricos de Bonner y de del Pobil tienen unos principios muy similares. Ambos utilizan un modelado esférico *jerárquico* basado en sucesivas aproximaciones del modelo, de forma que cada aproximación es una mejora del modelo anterior. También se puede entender que ambos modelos esféricos son *discretos*, ya que para un nivel de aproximación dado, la representación está determinada por un número finito de esferas. Además tanto uno como otro obtienen una *representación exterior* por exceso y una *representación interior* por empaquetamiento (Bonner los denomina límites superior e inferior).

Bonner llamó a su modelo esférico *SSA* (*Successive Spherical Approximation*) [Bon89] y tiene su aplicación únicamente sobre poliedros convexos. La representación SSA de un objeto está compuesta de una serie de niveles sucesivamente detallados que van desde una esfera envolviendo todo el objeto a las propias caras del objeto. Todas las aproximaciones están referidas a un punto central común, elegido arbitrariamente siempre y cuando esté dentro del objeto, resultando adecuado utilizar el centroide del objeto. Cada nivel está compuesto de dos límites, denominados límite superior que envuelve completamente la porción modelada del objeto y el límite inferior contenida completamente por la parte modelada del objeto. El modelo del objeto menos preciso está compuesto por su esfera envolvente (límite superior) cuyo radio está definido por la máxima distancia del punto central a un vértice y su esfera envuelta (límite inferior) cuyo radio está definido por la mínima distancia del punto central a las caras del objeto.

El siguiente nivel de precisión, la aproximación envolvente de las caras, divide las esferas anteriores en sectores rectangulares según las caras del objeto. Cada sector se define

mediante dos rangos angulares, según un sistema de coordenadas esféricas global que localiza cada cara respecto al punto central. Estos dos rangos se determinan calculando las coordenadas esféricas de cada vértice del objeto y obteniendo los valores mínimo y máximo de cada ángulo. El límite superior de una cara es el sector esférico que contiene completamente esta cara de una esfera cuyo radio es la máxima distancia del punto central a la cara. El límite inferior de una cara es el sector esférico contenido completamente por esta cara de una esfera cuyo radio es la mínima distancia del punto central a la cara. La representación es, por tanto, un conjunto finito de esferas, tantas como caras tenga el objeto, si bien se utiliza únicamente un sector esférico de cada esfera.

La aproximación de caras divididas descompone cada sector esférico de la aproximación envolvente de las caras en un conjunto de subsectores, resultando una aproximación más cercana a las caras del objeto. La descomposición se realiza dividiendo la diferencia entre los radios de los sectores esféricos de los límites superior e inferior por cierto número fijo y asignando un subsector a cada incremento. Realmente se puede utilizar cualquier otro método razonable de descomposición del sector esférico.

La última aproximación, la aproximación por caras, es el modelo exacto del objeto mediante sus caras, donde coinciden los límites superior e inferior. La Figura 2.1 muestra esquemáticamente los cuatro niveles de aproximación propuestos por esta técnica.

El modelo esférico SSA tiene varios inconvenientes: por una parte, la utilización de sectores esféricos en vez de esferas completas complica en gran medida su aplicación, además de la relativa complejidad de su generación; para que la aproximación de caras divididas tenga una buena precisión, se requiere un número elevado de sectores esféricos, con el consiguiente aumento de memoria requerida, ya que cada sector esférico debe almacenar los cuatro parámetros de una esfera y otros cuatro de los rangos angulares del sector; para elementos desproporcionados en dimensiones, tal como la columna de la Figura 2.2, que tiene una dimensión mucho mayor que las otras, las aproximaciones de todos los niveles darían resultados bastante malos. Una posible solución a este último problema consiste en realizar una división del objeto respecto a su dimensión mayor, modelando posteriormente cada una de sus partes. Evidentemente, además de aumentar aún más la cantidad de memoria necesitada, esta solución trae el problema de tener que distinguir entre caras interiores, que no se necesitaría modelar y caras exteriores del objeto.

La utilidad de este tipo de modelado esférico en la detección de colisiones y la planificación de movimientos según la filosofía de generación-y-prueba en sistemas robotizados se puede encontrar en [B&K88], [B&K89a], [B&K89b] y [B&K89c].

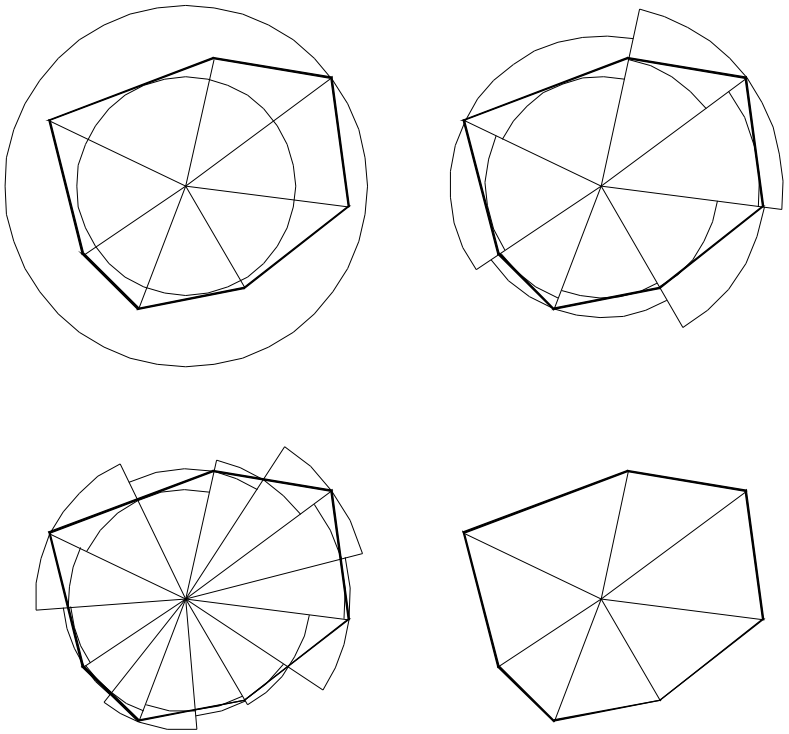


Figura 2.1. Modelo Esférico SSA de Bonner. *En el nivel 0 (figura superior izquierda) se obtienen la esfera envolvente y la esfera envuelta del objeto respecto a un punto del interior del objeto. En el nivel 1 (figura superior derecha) se obtienen un sector esférico envolvente y un sector esférico envuelto por cada una de las caras del objeto. En el nivel 2 (figura inferior izquierda) estos sectores esféricos se descomponen cada uno en dos sectores esféricos de diferentes esferas. En el último nivel se alcanza el modelo exacto del objeto, un poliedro convexo (figura inferior derecha). Por claridad, la figura muestra el caso para un polígono convexo en 2D.*

Frente al modelo esférico SSA, del Pobil propone una jerarquía basada en un algoritmo recursivo de refinamiento que busca en cada paso mejorar una aproximación sustituyendo la esfera "peor" del modelo (la esfera que tiene un mayor volumen no contenido por el objeto) mediante dos nuevas esferas. Por otra parte, del Pobil utiliza también dos modelos, un modelo exterior que busca un recubrimiento de la frontera del objeto (no de todo el interior el objeto) y un modelo interior para obtener un conjunto de esferas que estén incluidas dentro de la frontera del objeto. El proceso de obtener el modelo esférico aproximado se denomina esferización.

Este modelo esférico tridimensional está basado fundamentalmente en un modelo circular bidimensional, ya que los objetos considerados son generados por un barrido de traslación de un polígono convexo. Esta restricción posibilita por otra parte una gran simplicación: cubriendo una sección del objeto mediante un conjunto de círculos es relativamente fácil obtener las esferas que cubren al objeto. Por tanto, para modelar objetos tridimensionales se realiza una traslación del recubrimiento 2D para obtener el recubrimiento 3D.

Sobre polígonos 2D se plantean dos tipos de recubrimiento con discos o círculos, el recubrimiento de la frontera y el recubrimiento de toda el área incluida en el interior del polígono. Para obtener el modelo exterior del objeto se realiza inicialmente un recubrimiento de la frontera de la base del objeto utilizando un conjunto de discos. Una traslación de estos discos produce un conjunto de cilindros generalizados, que son sustituidos en una segunda fase por un conjunto de esferas con idéntico radio. Este conjunto de esferas recubre toda la frontera lateral del objeto, denominada SIDE, pero sobre las bases superior e inferior del mismo, denominadas TOP, debe realizarse un recubrimiento de toda el área incluida en el interior de las mismas. La Figura 2.2 muestra esquemáticamente el método de construcción del modelo esférico exterior de un objeto con el recubrimiento del TOP y del SIDE mediante esferas.

En ciertos casos puede resultar conveniente realizar un tercer recubrimiento sobre dos nuevas regiones, denominadas TIP, adyacentes a las dos bases del objeto, obtenidas por subdivisión de la frontera lateral. Sobre estas zonas se realiza un recubrimiento similar al del SIDE, pero independiente del mismo.

El modelo esférico interior se basa en un diagrama de Voronoi de las aristas del polígono base. En concreto se determina un conjunto finito de discos cuyos centros se encuentran en el diagrama de Voronoi y son tangentes al menos a dos lados del polígono, siendo este conjunto el recubrimiento interior del polígono. Una extensión al caso 3D, con generación de cilindros generalizados a partir del recubrimiento interior de la base y realizando la conversión de estos cilindros generalizados a un número finito de esferas permite obtener el modelo esférico interior del objeto.

Del Pobil realiza un estudio de la precisión y convergencia de sus modelos, obteniendo que el error cometido, obviamente, se acerca a cero cuando el número de esferas tiende a infinito. Sin embargo, si bien esta propiedad es deseable, tal como se vió anteriormente, el modelo esférico exterior de una columna similar a la mostrada en la Figura 2.2, con 250cm de alto, el error cometido es de 8.82cm (3.528%) cuando se utilizan cien esferas. Esta convergencia no es muy rápida, ya que para 20 esferas, [dPS92a] muestra que esta distancia máxima de error es aproximadamente 13cm (5.2%), por lo que el error sólo se reduce en un 1.7% al multiplicar por 5 el número de esferas. Evidentemente, el número de esferas, y por tanto la memoria, a utlilizar para alcanzar grandes precisiones de representación puede llegar a ser muy elevado para entornos mínimamente complejos.

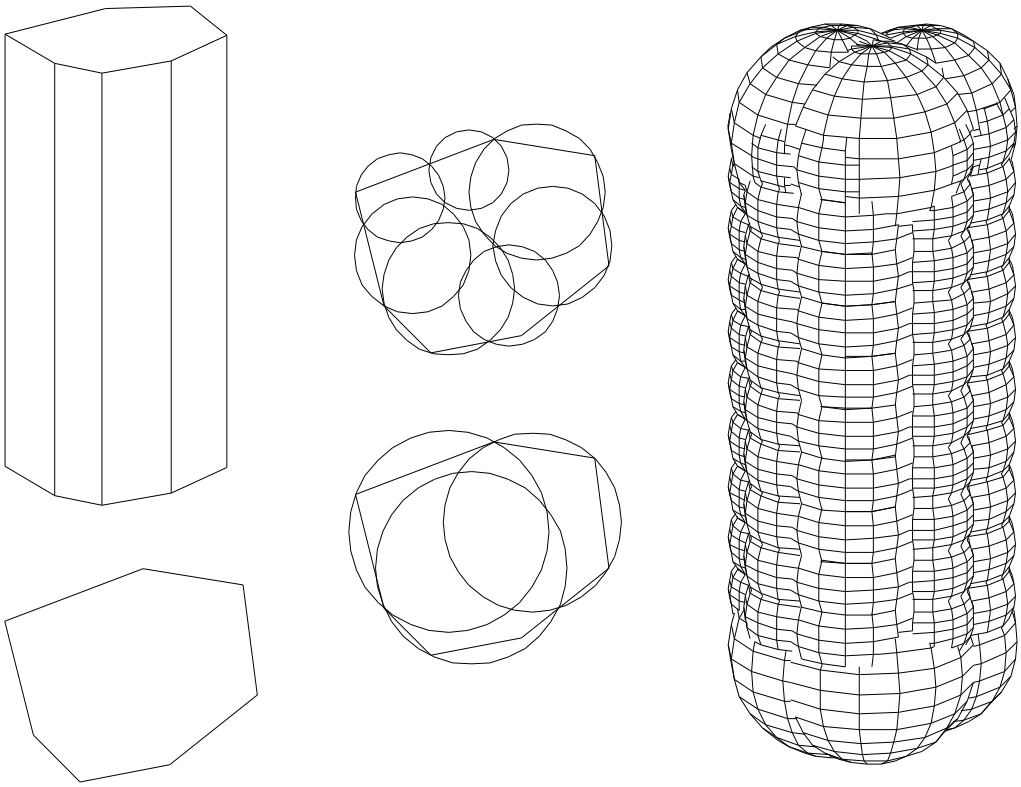


Figura 2.2. Modelo Esférico de del Pobil. *La columna a modelar se muestra en la figura de la izquierda, con la base a partir de la cual se ha generado mediante un barrido de traslación. En la parte central se puede observar el recubrimiento de frontera realizado para modelar el SIDE mediante siete discos (figura central superior) y el recubrimiento global del TOP realizado con tres discos (figura central inferior). Repitiendo el modelo del TOP para las bases superior e inferior y el modelo del SIDE diez veces para las caras laterales del objeto, en la figura de la izquierda se puede ver el modelo esférico exterior de la columna obtenido con 76 esferas.*

La utilización de este modelado esférico en el campo de la detección de colisiones y la planificación de movimientos en sistemas robotizados se puede encontrar en [dPS92a], [dPS92b], [dPE93], [dPS94a] y [dPS94b]. Como esta técnica de modelado sólo es aplicable a prismas rectos obtenidos por un barrido traslacional de polígonos convexos, los últimos trabajos de del Pobil tienden a resolver este inconveniente, mediante extensiones para considerar el recubrimiento de polígonos no convexos [MdP95a] y polígonos generalizados [MdP95b].

2.3. Poli-Esferas

Frente a los dos modelos comentados anteriormente, en esta sección se presenta un nuevo modelado esférico, las *poli-esferas*. Este tipo de modelado esférico se puede entender como *continuo*, ya que para cualquier nivel de aproximación deseado, el modelo está compuesto de infinitas esferas, frente a los modelos esféricos discretos anteriores que utilizan un número finito de esferas para cada nivel. El concepto de poli-esferas se basa en una aproximación para el modelado geométrico denominada inicialmente los *politopos esféricos* o *s-topos*, que se presentó en [THK90] y [THK91]. Con esta aproximación, los objetos (y en particular los elementos de un sistema robotizado) se aproximan por un número infinito de esferas, produciendo volúmenes de forma esférica. Estos trabajos previos adolecían de una falta de formalización y generalización, trabajando y aplicando relaciones lineales de forma independiente a los centros y radios de las esferas. En esta sección se introduce una normalización general para este tipo de modelos, que facilita su utilización en los siguientes capítulos.

2.3.1. Espacio Vectorial Esférico

Es bien conocido que la esfera se define mediante cuatro parámetros, por lo que se puede representar como un punto en $\Re^4$ mediante una 4-tupla, representada por $s=(x,y,z,r)\in\Re^4$ cuyo centro se denota por $s.c=(s.x,s.y,s.z)\in\Re^3$ y su radio por $s.r\in\Re$ y que se define como el conjunto de puntos cuya distancia al centro de la esfera es menor o igual al radio de la esfera:

$$s=\left\{p\in\Re^3:\left\|p-s.c\right\|\le s.r\right\}$$

siendo $\|\bullet\|$ la norma Euclídea en $\Re^3$.

Esta representación de las esferas constrasta con otras representaciones utilizadas recientemente por otros autores como [Ped88], [DMT92], [BC+91], [BC+92] y [D&G94], que han representado una esfera como un punto en $\Re^4$ (o un círculo como un punto en $\Re^3$) mediante la 4-tupla (x,y,z,R) donde el radio de la esfera es $r=x^2+y^2+z^2-R$. Si bien este tipo de representación tiene sus ventajas en la geometría computacional, no es así para el modelado que se presenta en este capítulo.

El conjunto de todas las esferas posibles en el espacio 3D se denomina *Espacio Esférico*, denotado por Ω, cuya esfera origen $s_\varnothing$ es la *esfera nula* formada por cuatro componentes nulas.

Dada una esfera, s_i, se define su *vector esférico desde el origen*, $\mathbf{s}_i$, como una 4-tupla con las mismas cuatro componentes que la esfera s_i. Un vector esférico tiene como sentido

geométrico la unión de su *esfera definitoria* con un *cono tangente* a esta esfera cuyo vértice es el origen. Se denomina *circunferencia tangente del vector esférico* a la circunferencia de tangencia entre el cono tangente y la esfera definitoria. Se denomina *eje* del vector esférico $\mathbf{s}_i$ al vector tridimensional $\mathbf{v}_i$ que va desde el origen al centro de la esfera s_i y *eje unitario* a este vector dividido por su norma, $\hat{\mathbf{v}}_i = \mathbf{v}_i / \|\mathbf{v}_i\|$. Se denomina *grado de convergencia* del vector esférico $\mathbf{s}_i$ al valor $\eta_i = s_i.r / \|\mathbf{v}_i\|$ y *ángulo de convergencia* a $\gamma_i=\sin^{-1}\eta_i$. Para valores del grado de convergencia igual o mayor que uno ($\eta_i \geq 1$) el vector esférico es degenerado, no existiendo ni cono ni circunferencia tangente. Se denomina $\hat{\mathbf{n}}_i$ a cualquier vector unitario normal al cono tangente (apuntando hacia el exterior) del vector esférico $\mathbf{s}_i$ por lo que cumple que $\hat{\mathbf{n}}_i \cdot \hat{\mathbf{v}}_i = -|\eta_i|$. Se define el vector esférico nulo, $\mathbf{s}_\varnothing$, a aquél con sus cuatro componentes nulas. La Figura 2.3 muestra éstos y otros parámetros de los vectores esféricos.

Dadas dos esferas s_i, s_j, se define el vector esférico $\mathbf{s}_{ij}$ de la esfera s_i a la esfera s_j, como el vector esférico desde el origen a la esfera resultado de restar a cada componente de la esfera s_j la correspondiente componente de s_i, por lo que siempre existe una esfera s_k tal que $\mathbf{s}_k=\mathbf{s}_{ij}$. En este caso, el eje del vector esférico $\mathbf{s}_{ij}$ es el vector tridimensional $\mathbf{v}_{ij}$ que une los centros de las esferas (siendo su vector unitario el eje unitario), el grado de convergencia es $\eta_{ij} = \left(s_j.r - s_i.r\right) / \|\mathbf{v}_{ij}\|$ y el ángulo de convergencia $\gamma_{ij}=\sin^{-1}\eta_{ij}$. Nótese que un vector generado así puede tener radio negativo en su esfera definitoria.

Sobre el conjunto de vectores esféricos se definen las siguientes operaciones internas:

- La suma (resta) de vectores esféricos es el vector esférico resultante de realizar la suma (resta) de sus componentes:

$$\mathbf{s}_i \pm \mathbf{s}_j = \mathbf{s}_k: \quad \begin{array}{l} \mathbf{s}_k.x = \mathbf{s}_i.x \pm \mathbf{s}_j.x, \;\; \mathbf{s}_k.y = \mathbf{s}_i.y \pm \mathbf{s}_j.y, \\ \mathbf{s}_k.z = \mathbf{s}_i.z \pm \mathbf{s}_j.z, \;\; \mathbf{s}_k.r = \mathbf{s}_i.r \pm \mathbf{s}_j.r \end{array}$$

- El producto de un vector esférico por un real se define mediante el producto de cada componente por el escalar:

$$\lambda\mathbf{s}_i = \mathbf{s}_k: \quad \begin{array}{l} \mathbf{s}_k.x = \lambda\mathbf{s}_i.x, \;\; \mathbf{s}_k.y = \lambda\mathbf{s}_i.y, \\ \mathbf{s}_k.z = \lambda\mathbf{s}_i.z, \;\; \mathbf{s}_k.r = \lambda\mathbf{s}_i.r \end{array}$$

Estas operaciones cumplen las siguientes propiedades:

- Conmutativa: $\mathbf{s}_i \pm \mathbf{s}_j = \mathbf{s}_j \pm \mathbf{s}_i$
- Asociativa: $(\mathbf{s}_i \pm \mathbf{s}_j) \pm \mathbf{s}_k = \mathbf{s}_i \pm (\mathbf{s}_j \pm \mathbf{s}_k)$
- Distributiva: $\lambda(\mathbf{s}_i \pm \mathbf{s}_j) = \lambda\mathbf{s}_i \pm \lambda\mathbf{s}_j$

Además se define la suma (resta) de una esfera y un vector esférico como la esfera resultado de sumar (restar) sus componentes:

$$s_i \pm \mathbf{s}_j = s_k: \quad \begin{array}{l} s_k.x = s_i.x \pm \mathbf{s}_j.x, \;\; s_k.y = s_i.y \pm \mathbf{s}_j.y, \\ s_k.z = s_i.z \pm \mathbf{s}_j.z, \;\; s_k.r = s_i.r \pm \mathbf{s}_j.r \end{array}$$

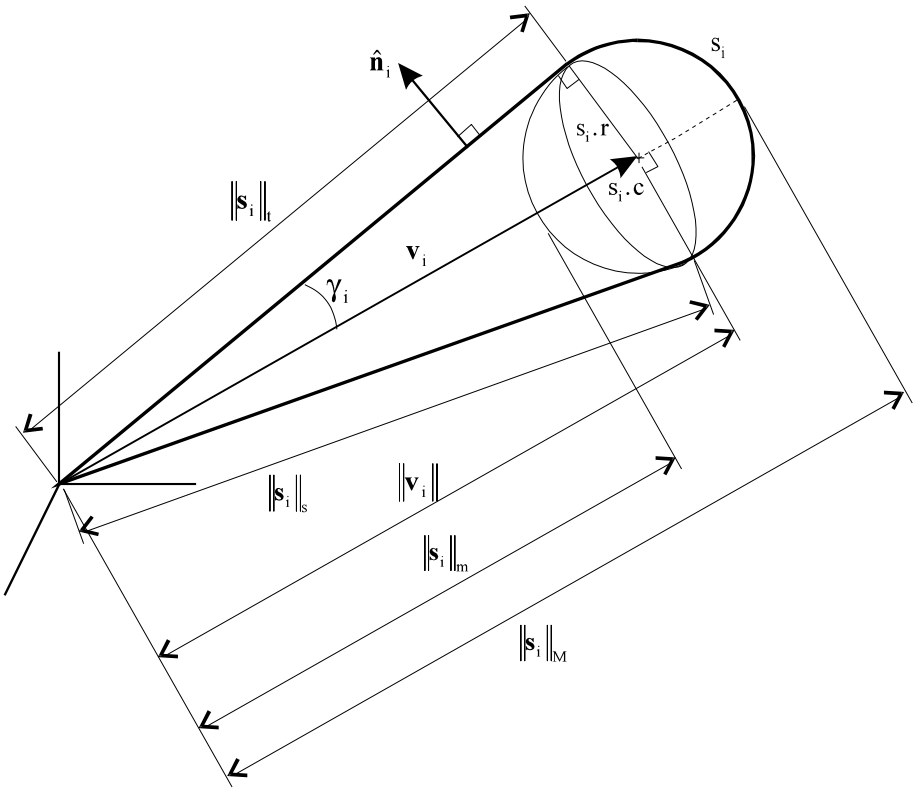


Figura 2.3. Sentido Geométrico y Parámetros de un Vector Esférico. *Un vector esférico* $\mathbf{s}_i$ *viene definido por una esfera* $\mathbf{s}_i$ *y su cono tangente con vértice en el origen. El eje del vector esférico es el vector tridimensional* $\mathbf{v}_i$ *que va desde el origen al centro de la esfera. El ángulo de convergencia* γ_i *es el semiángulo del cono y el grado de convergencia* η_i *es el seno de este ángulo. El vector* $\hat{\mathbf{n}}_i$ *es perpendicular al cono tangente del vector esférico. La norma esférica euclídea* $\|\mathbf{s}_i\|_s$ *es la distancia del origen a un punto del círculo formado por la intersección de la esfera definitoria con un plano que pasa por el centro de la esfera y tiene como vector normal el eje del vector esférico; la norma esférica tangente* $\|\mathbf{s}_i\|_t$ *es la distancia del origen a la circunferencia de tangencia entre el cono y la esfera (ambos valores son diferentes al módulo del eje* $\|\mathbf{v}_i\|$*), mientras que las normas esféricas mínima* $\|\mathbf{s}_i\|_m$ *y máxima* $\|\mathbf{s}_i\|_M$ *son las distancias del origen al punto de la esfera más cercano y lejano respectivamente.*

Se define la *norma esférica euclídea* de un vector esférico $\mathbf{s}_{ij}$, representada por $\left\|\mathbf{s}_{ij}\right\|_s$, como la norma euclídea de la 4-tupla representativa del vector esférico, es decir:

$$\left\|\mathbf{s}_{ij}\right\|_s = \sqrt{\left(\mathbf{s}_{ij}.x\right)^2 + \left(\mathbf{s}_{ij}.y\right)^2 + \left(\mathbf{s}_{ij}.z\right)^2 + \left(\mathbf{s}_{ij}.r\right)^2}$$

Para ciertos casos, esta norma no es muy significativa como medida de distancia, por lo que se pueden definir también las siguientes tres pseudo-normas:

- *Pseudo-norma esférica tangente* de un vector esférico $\|\mathbf{s}_{ij}\|_t$ según:

$$\|\mathbf{s}_{ij}\|_t = \sqrt{(\mathbf{s}_{ij}.x)^2 + (\mathbf{s}_{ij}.y)^2 + (\mathbf{s}_{ij}.z)^2 - (\mathbf{s}_{ij}.r)^2}$$

- *Pseudo-norma esférica mínima* de un vector esférico $\|\mathbf{s}_{ij}\|_m$ según:

$$\|\mathbf{s}_{ij}\|_m = \|\mathbf{v}_{ij}\| - \mathbf{s}_{ij}.r$$

- *Pseudo-norma esférica máxima* de un vector esférico $\|\mathbf{s}_{ij}\|_M$ según:

$$\|\mathbf{s}_{ij}\|_M = \|\mathbf{v}_{ij}\| + \mathbf{s}_{ij}.r$$

La norma esférica euclídea representa la distancia euclídea del origen a un punto del círculo formado por la intersección de la esfera definitoria con un plano que pasa por el centro de la esfera y tiene como vector normal el eje del vector esférico. La pseudo-norma esférica tangente representa la distancia euclídea del origen a un punto de la circunferencia tangente del vector esférico (longitud exterior del cono tangente). La pseudo-norma esférica mínima (máxima) se corresponde con la mínima (máxima) distancia del origen a la esfera definitoria (el significado de estas normas se puede ver en la Figura 2.3).

Un vector esférico es *unitario respecto a una norma esférica* cuando su norma esférica es uno. Por tanto existen cuatro formas de normalizar los vectores esféricos, cada una de ellas con un significado diferente. La Figura 2.4 muestra comparativamente diferentes vectores esféricos unitarios respecto a cada una de estas normas.

Es de resaltar que las tres últimas normas definidas son pseudo-normas por que no cumplen una de las tres propiedades de las normas, en concreto la que expresa que la norma de la suma de vectores es menor o igual a la suma de normas. La utilidad de estas diferentes normas esféricas se verá al analizar el cálculo de distancias entre objetos modelados con poli-esferas.

Dados n vectores esféricos respecto a una misma esfera s_0, $\{\mathbf{s}_{01},...,\mathbf{s}_{0n}\}$, se dice que son *linealmente independientes* si cualquier combinación lineal no trivial es no nula, es decir,

$$\sum_{i=1}^{n} \alpha_i \mathbf{s}_{0i} = 0 \quad \text{solo si} \quad \alpha_i = 0, i = 1, \ldots, n$$

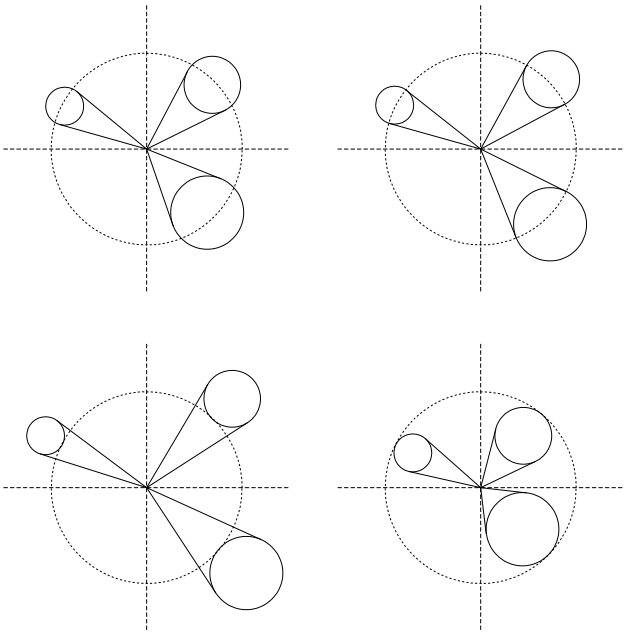


Figura 2.4. Vectores Esféricos Unitarios Respecto a Diferentes Normas Esféricas. *Los vectores esféricos de la figura superior izquierda son unitarios respecto a la norma esférica euclídea, cumpliendo todos ellos que la frontera de su intersección con la esfera unitaria (esfera centrada en el origen y de radio unidad) es una circunferencia con el mismo radio que la esfera definitoria del vector esférico. Los vectores esféricos de la figura superior derecha son unitarios respecto a la norma esférica tangente, cumpliendo todos ellos que la frontera de su intersección con la esfera unitaria es la circunferencia tangente del vector esférico. Los vectores esféricos de la figura inferior izquierda son unitarios respecto a la norma esférica mínima, cumpliendo todos ellos que sus esferas definitorias son tangentes exteriormente a la esfera unitaria. Los vectores esféricos de la figura inferior derecha son unitarios respecto a la norma esférica máxima, cumpliendo todos ellos que sus esferas definitorias son tangentes interiormente a la esfera unitaria.*

En cualquier otro caso son *linealmente dependientes*, y por lo menos uno de los vectores esféricos es una combinación lineal de los demás, es decir

$$\exists \alpha_j \neq 0 : \mathbf{s}_{0j} = \sum_{\substack{i=1 \\ i \neq j}}^{n} \beta_i \mathbf{s}_{0i}, \quad \beta_i = \frac{-\alpha_i}{\alpha_j}, i = 1, \ldots, n, i \neq j$$

Se define el *producto escalar esférico* entre dos vectores esféricos no nulos, $\mathbf{s}_i$ y $\mathbf{s}_j$ de la siguiente forma:

$$\mathbf{s}_i \cdot \mathbf{s}_j = \left\| \mathbf{s}_i \right\|_s \left\| \mathbf{s}_j \right\|_s \cos \beta_{ij}$$

siendo β_{ij} el mínimo ángulo que forma el eje del vector esférico $\mathbf{s}_i$ con el cono tangente del vector esférico $\mathbf{s}_j$ y cuyo coseno se define como:

$$\cos\beta_{ij} = \cos\left(\alpha_{ij} - \gamma_j\right) = \cos\alpha_{ij}\cos\gamma_j + \sin\alpha_{ij}\sin\gamma_j$$

siendo α_{ij} el ángulo que forman los ejes $\mathbf{v}_i$ y $\mathbf{v}_j$ de los vectores esféricos $\mathbf{s}_i$ y $\mathbf{s}_j$, cuyo coseno es $\cos\alpha_{ij} = \dfrac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\|\|\mathbf{v}_j\|}$ y γ_j el ángulo de convergencia del vector esférico $\mathbf{s}_j$. Nótese que este producto escalar esférico sólo es conmutativo cuando ambos vectores esféricos tienen igual grado de convergencia.

Se dice que el vector esférico $\mathbf{s}_i$ es perpendicular al vector esférico $\mathbf{s}_j$ cuando $\mathbf{s}_i \cdot \mathbf{s}_j = 0$, es decir, el eje de $\mathbf{s}_i$ es perpendicular al cono tangente de $\mathbf{s}_j$, por lo que se debe verificar que $\hat{\mathbf{v}}_i \cdot \hat{\mathbf{v}}_j = -\left|\eta_j\right|$.

Se define el *producto vectorial esférico* entre dos vectores esféricos no nulos, $\mathbf{s}_i$ y $\mathbf{s}_j$ como aquel vector esférico $\mathbf{s}_k$ que es perpendicual a $\mathbf{s}_i$ y $\mathbf{s}_j$:

$$\mathbf{s}_k = \mathbf{s}_i \times \mathbf{s}_j \Leftrightarrow \mathbf{s}_k \cdot \mathbf{s}_i = 0 \text{ y } \mathbf{s}_k \cdot \mathbf{s}_j = 0$$

De las dos condiciones se puede plantear el siguiente sistema de ecuaciones:

$$\begin{cases} \hat{\mathbf{v}}_k \cdot \hat{\mathbf{v}}_i = -\left|\eta_i\right| \\ \hat{\mathbf{v}}_k \cdot \hat{\mathbf{v}}_j = -\left|\eta_j\right| \end{cases} \Rightarrow \begin{cases} \hat{\mathbf{v}}_k.x\,\hat{\mathbf{v}}_i.x + \hat{\mathbf{v}}_k.y\,\hat{\mathbf{v}}_i.y + \hat{\mathbf{v}}_k.z\,\hat{\mathbf{v}}_i.z = -\left|\eta_i\right| \\ \hat{\mathbf{v}}_k.x\,\hat{\mathbf{v}}_j.x + \hat{\mathbf{v}}_k.y\,\hat{\mathbf{v}}_j.y + \hat{\mathbf{v}}_k.z\,\hat{\mathbf{v}}_j.z = -\left|\eta_j\right| \end{cases}$$

Se puede comprobar que un múltiplo del vector unitario solución del sistema es el vector obtenido mediante el siguiente determinante:

$$\begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ \hat{\mathbf{v}}_i.x - \dfrac{\left|\eta_i\right|}{\hat{\mathbf{v}}_i.z\,\hat{\mathbf{v}}_j.y - \hat{\mathbf{v}}_i.y\,\hat{\mathbf{v}}_j.z} & \hat{\mathbf{v}}_i.y & \hat{\mathbf{v}}_i.z \\ \hat{\mathbf{v}}_j.x - \dfrac{\left|\eta_j\right|}{\hat{\mathbf{v}}_i.z\,\hat{\mathbf{v}}_j.y - \hat{\mathbf{v}}_i.y\,\hat{\mathbf{v}}_j.z} & \hat{\mathbf{v}}_j.y & \hat{\mathbf{v}}_j.z \end{vmatrix}$$

Sin embargo, resulta más interesante expresar el eje unitario $\hat{\mathbf{v}}_k$ respecto al sistema formado por $\hat{\mathbf{v}}_i$, $\hat{\mathbf{v}}_j$ y $\hat{\mathbf{v}}_i \times \hat{\mathbf{v}}_j$ según $\hat{\mathbf{v}}_k = \alpha\hat{\mathbf{v}}_i + \beta\hat{\mathbf{v}}_j + \gamma\left(\hat{\mathbf{v}}_i \times \hat{\mathbf{v}}_j\right)$ que se obtiene a partir de:

$$\begin{cases} \hat{\mathbf{v}}_k \cdot \hat{\mathbf{v}}_i = \alpha + \beta\left(\hat{\mathbf{v}}_i \cdot \hat{\mathbf{v}}_j\right) = -\left|\eta_i\right| \\ \hat{\mathbf{v}}_k \cdot \hat{\mathbf{v}}_j = \alpha\left(\hat{\mathbf{v}}_i \cdot \hat{\mathbf{v}}_j\right) + \beta = -\left|\eta_j\right| \end{cases} \Rightarrow \begin{cases} \alpha = \dfrac{\left|\eta_j\right|\left(\hat{\mathbf{v}}_i \cdot \hat{\mathbf{v}}_j\right) - \left|\eta_i\right|}{\left\|\hat{\mathbf{v}}_i \times \hat{\mathbf{v}}_j\right\|^2} \\ \beta = \dfrac{\left|\eta_i\right|\left(\hat{\mathbf{v}}_i \cdot \hat{\mathbf{v}}_j\right) - \left|\eta_j\right|}{\left\|\hat{\mathbf{v}}_i \times \hat{\mathbf{v}}_j\right\|^2} \end{cases}$$

La condición de vector unitario fija:

$$\hat{\mathbf{v}}_k \cdot \hat{\mathbf{v}}_k = 1 \Rightarrow \alpha^2 + \beta^2 + 2\alpha\beta\left(\hat{\mathbf{v}}_i \cdot \hat{\mathbf{v}}_j\right) + \gamma^2\left\|\hat{\mathbf{v}}_i \times \hat{\mathbf{v}}_j\right\|^2 = 1$$

de donde despejando se obtienen dos valores de γ, descartándose el valor negativo (que corresponde al eje del producto vectorial esférico en orden contrario, $\mathbf{s}_j \times \mathbf{s}_i$).

Una vez fijado el eje del vector esférico resultado del producto vectorial esférico, el radio de la esfera definitoria del vector $\mathbf{s}_k$ se puede fijar para que cumpla:

$$\mathbf{s}_k.\mathrm{r} = \left\|\mathbf{v}_k\right\| \sin\gamma_{kij}, \quad \text{con } \cos\gamma_{kij} = \frac{\mathbf{v}_k}{\left\|\mathbf{v}_k\right\|} \cdot \frac{\mathbf{v}_i \times \mathbf{v}_j}{\left\|\mathbf{v}_i \times \mathbf{v}_j\right\|}$$

2.3.2. Poli-Esferas. Elementos Básicos

Una *poli-esfera* es el casco convexo de un conjunto finito de esferas, $\mathbf{S}=\{s_0,...,s_n\}$ y se define como

$$\mathrm{Co}\mathbf{S} = \left\{ s \in \Omega : s = \sum_{i=0}^{n} \lambda_i s_i,\ s_i \in \mathbf{S},\quad 0 \le \lambda_i, i = 0,\ldots,n \quad \sum_{i=0}^{n} \lambda_i = 1 \right\}$$

De las condiciones en λ_i, se puede eliminar $\lambda_0 : \lambda_0 = 1 - \sum_{i=1}^{n} \lambda_i$. Substituyendo en la ecuación anterior, da la expresión basada en vectores esféricos que se utilizará para poli-esferas:

$$\mathrm{Co}\mathbf{S} = \left\{ \begin{array}{l} s \in \Omega : s = s_0 + \sum_{i=1}^{n} \lambda_i \mathbf{s}_{0i}, \\ \mathbf{s}_{0i} = (s_i - s_0), s_0 \in \mathbf{S}, s_i \in \mathbf{S}, i = 1,\ldots,n, \\ 0 \le \lambda_i, i = 1,\ldots,n \sum_{i=1}^{n} \lambda_i \le 1 \end{array} \right\}$$

Una poli-esfera formada a partir de un conjunto de esferas $\{s_0,...,s_n\}$ se denomina $S_{0\text{-}n}$. Para las poli-esferas *simples* (las de orden menor o igual a cuatro) las esferas se listan explícitamente. Por ejemplo, S_{012} es una poli-esfera definida a partir de las esferas s_0, s_1 y s_2.

Las esferas que definen la poli-esfera se llaman sus *vértices esféricos*. Si se toma un vértice esférico, preferentemente la esfera con menor radio (por ejemplo s_0), como *esfera primaria*, hay n vectores esféricos ($\mathbf{s}_{01}$, ..., $\mathbf{s}_{0n}$) que definen la poli-esfera. La esfera primaria sirve para situar la poli-esfera, así como para producir un efecto *offset* sobre la misma (aumento de radio). El *orden* de una poli-esfera es el número de esferas que la definen, y el *grado* es el número de vectores esféricos, cumpliéndose que grado=orden-1. Los valores de orden y grado se utilizarán posteriormente para reducir la complejidad de las poli-esferas.

Una poli-esfera contiene un conjunto infinito de esferas, llamado el conjunto de *esferas barridas*, obtenidas según la relación lineal del casco convexo. Una esfera barrida que pertenezca a una poli-esfera se obtiene fijando un valor para los parámetros: por ejemplo, la esfera obtenida en $S_{0\text{-}n}$ para un conjunto de valores fijos $\lambda_1,...,\lambda_n$ se denomina $S_{0\text{-}n}(\lambda_1,...,\lambda_n)$. La expresión matricial que permite obtener una esfera barrida de una poli-esfera $S_{0\text{-}n}$ para los valores $\lambda_1,...,\lambda_n$ es

$$S_{0-n}(\lambda_1,\ldots,\lambda_n)=\begin{bmatrix}1 & \lambda_1 & \cdots & \lambda_n\end{bmatrix}\begin{bmatrix}1 & -1 & \cdots & \cdots & -1\\ 0 & 1 & 0 & \cdots & 0\\ 0 & \ddots & \ddots & \ddots & \vdots\\ \vdots & \ddots & \ddots & 1 & 0\\ 0 & \cdots & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}s_0\\ \vdots\\ s_n\end{bmatrix}=s_0+\begin{bmatrix}\lambda_1 & \cdots & \lambda_n\end{bmatrix}\begin{bmatrix}\mathbf{s}_{01}\\ \vdots\\ \mathbf{s}_{0n}\end{bmatrix}$$

Nótese que el conjunto de esferas barridas no incluye todas las posibles esferas que están incluidas dentro de la poli-esfera, sino sólo aquellas generadas por esta última ecuación.

El conjunto de todas las posibles esferas que se encuentran dentro de una esfera s se denomina $S_\infty(s)$ y se puede expresar como:

$$S_\infty(s)=\left\{s_i\in\Omega:\left\|s.c-s_i.c\right\|\le s.r-s_i.r\right\}$$

El conjunto de todas las posibles esferas que están incluidas en una poli-esfera $S_{0\text{-}n}$ se denota por $S_\infty(S_{0\text{-}n})$ y se puede expresar como:

$$S_\infty(S_{0\text{-}n})=\left\{s\in\Omega:\exists\lambda_1,...,\lambda_n,\ 0\le\lambda_i,i=1,...,n,\ \sum_{i=1}^{n}\lambda_i\le 1,\ s\in S_\infty(S_{0\text{-}n}(\lambda_1,...,\lambda_n))\right\}$$

Una poli-esfera es *sobre-especificada* si uno o más de sus vértices esféricos se pueden eliminar sin cambiar el volumen ocupado por su casco convexo. Este o estos vértices esféricos son *redundantes*. Formalmente, una poli-esfera $S_{0\text{-}n}$ es sobre-especificada si se cumple que:

$$\exists i,\ 0\le i\le n: s_i\in S_\infty(S_{0,..,i-1,i+1,...,n})$$

Entonces, se cumple que $S_\infty(S_{0\text{-}n})=S_\infty(S_{0,\ldots,i\text{-}1,i+1,\ldots,n})$. Una poli-esfera que no es sobre-especificada es *válida*.

Eliminando los vectores esféricos linealmente dependientes, se puede encontrar un conjunto mínimo de vectores esféricos que formen una *base* de la poli-esfera $S_{0\text{-}n}$. Las esferas utilizadas para definir la base de $S_{0\text{-}n}$ se llaman sus *esferas básicas*. El número de elementos de la base se denomina la *dimensión* de la poli-esfera $S_{0\text{-}n}$, representado por φ, y es el grado mínimo de la poli-esfera. Nótese que φ no puede ser mayor que 4, la dimensión del Espacio Esférico Ω.

Una poli-esfera $S_{0\text{-}n}$ formado a partir de $\{s_0,\ldots,s_n\}$ con dimensión φ es *sobre-dimensionada* si $\varphi<n$. En otro caso, ($\varphi=n$), es bien-dimensionada. Una poli-esfera sobre-dimensionada puede *reducirse* a una poli-esfera bien-dimensionada formada por $\varphi+1$ esferas básicas y que ocupe el mismo volumen. Por tanto, la poli-esfera se puede representar con sólo φ parámetros $(\lambda'_1,\ldots,\lambda'_\varphi)$ en vez de con n parámetros $(\lambda_1,\ldots,\lambda_n)$.

Una poli-esfera que es válida y bien-dimensionada se denomina *irreducible*. A continuación se presentan brevemente las poli-esferas simples :

- La poli-esfera más simple es una esfera (o *mono-esfera*). Una poli-esfera formada por una esfera s_0 se denota por S_0 y es idéntica a esa esfera. Su dimensión es cero ($\varphi=0$).

- Una poli-esfera formada por dos esferas s_0 y s_1, es una *bi-esfera*, denotada por S_{01} (Figura 2.5). La dimensión de una bi-esfera irreducible es 1 ($\varphi=1$). La expresión que describe el conjunto de esferas barridas que definen la bi-esfera se obtiene reescribiendo la ecuación general de poli-esferas de la siguiente forma,

$$S_{01} = \left\{ s \in \Omega : s = s_0 + \lambda_1 \mathbf{s}_{01},\ 0 \le \lambda_1 \le 1 \right\}$$

  Las esferas barridas por una bi-esfera se obtienen con la ecuación

$$S_{01}(\lambda_1) = \begin{bmatrix} 1 & \lambda_1 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \end{bmatrix} = s_0 + \lambda_1 \mathbf{s}_{01}$$

  Por tanto, una bi-esfera puede ser engendrada a partir de una esfera primaria s_0 y un vector esférico $\mathbf{s}_{01}$.

  Existen diversos parámetros que pueden utilizarse para describir la bi-esfera S_{01}, basados fundamentalmente en los parámteros de los vectores esféricos:

  - La *esfera primaria* es la esfera de menor radio, s_0
  - El *offset* de la biesfera es el radio de la esfera primaria, $s_0.r$
  - El *eje esférico* de la bi-esfera viene determinada por el vector esférico $\mathbf{s}_{01}$

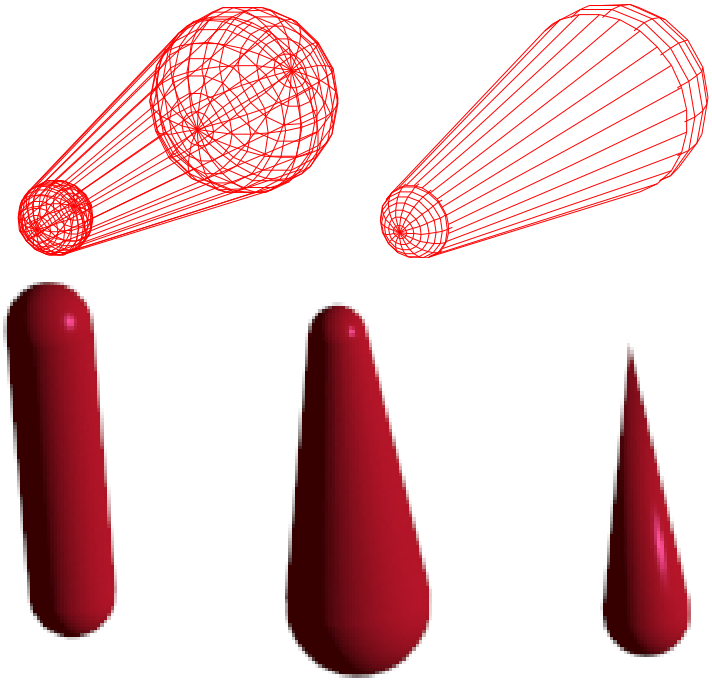


Figura 2.5. Bi-Esferas. *En la parte superior se muestra las representaciones alámbrica y superficial de un bi-esfera, donde se puede observar claramente que la bi-esfera está generada a partir de dos esferas. En la parte inferior se muestran tres posibilidades de modelado con bi-esferas, mediante una representación sombreada de un modelo sólido, donde la bi-esfera de la izquierda se forma a partir de dos esferas de idéntico radio, la central de dos esferas de radio diferente y en la bi-esfera de la izquierda una esfera tiene radio nulo, es decir, es un punto.*

- Los *módulos euclídeo, mínimo y máximo* de la bi-esfera vienen dados por las normas esféricas euclídea, mínima y máxima de su vector esférico, $\|\mathbf{s}_{01}\|_s, \|\mathbf{s}_{01}\|_m, \|\mathbf{s}_{01}\|_M$
- El *eje*, el *grado de convergencia* y el *ángulo de convergencia* de la bi-esfera vienen dados por sus correspondientes valores de su eje esférico, $\mathbf{v}_{01}$, η_{01} y γ_{01}.

Cuando el vector esférico de la bi-esfera es degenerado (valores del grado de convergencia igual o mayor que uno) el volumen degenera en el de una simple esfera igual a la más grande de las dos esferas extremas de la bi-esfera.

- Una *tri-esfera* es una poli-esfera formada a partir de tres esferas, s_0, s_1 y s_2, y se denota por S_{012} (Figura 2.6). Una tri-esfera irreducible tiene dimensión 2 (φ=2). La expresión que describe el conjunto de las esferas barridas en una tri-esfera es

$$S_{012} = \{s \in \Omega : s = s_0 + \lambda_1 \mathbf{s}_{01} + \lambda_2 \mathbf{s}_{02},\ 0 \le \lambda_1, \lambda_2,\ \lambda_1 + \lambda_2 \le 1\}$$

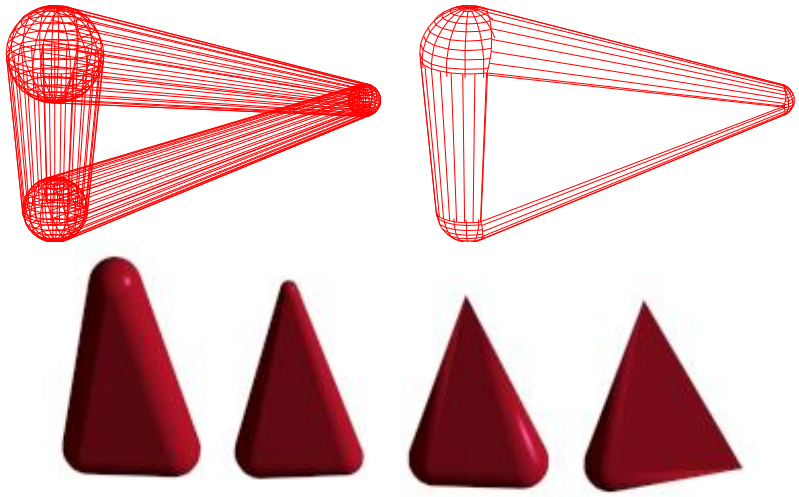


Figura 2.6. Tri-Esferas. *En la parte superior se muestra las representaciones alámbrica y superficial de un tri-esfera, donde se puede observar claramente que la tri-esfera está generada a partir de tres esferas. En la parte inferior se muestran cuatro posibilidades de modelado con tri-esferas, mediante una representación sombreada de un modelo sólido, donde de izquierda a derecha, la primera la tri-esfera se forma a partir de tres esferas de idéntico radio, la segunda de tres esferas de radio diferente, la siguiente tiene dos de sus esferas con el mismo radio y otra de radio nulo y en la última tri-esfera hay dos esferas de radio nulo, probocando que una arista esférica sea un segmento rectilíneo.*

Las esferas barridas por una tri-esfera se obtienen con la ecuación

$$S_{012}(\lambda_1,\lambda_2) = \begin{bmatrix} 1 & \lambda_1 & \lambda_2 \end{bmatrix} \begin{bmatrix} 1 & -1 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \end{bmatrix} = s_0 + \begin{bmatrix} \lambda_1 & \lambda_2 \end{bmatrix} \begin{bmatrix} \mathbf{s}_{01} \\ \mathbf{s}_{02} \end{bmatrix}$$

Por tanto, una tri-esfera puede ser engendrada a partir de una esfera primaria s_0 y dos vectores esféricos $\mathbf{s}_{01}$ y $\mathbf{s}_{02}$. Existen diversos parámetros que pueden utilizarse para describir la tri-esfera S_{012}:

- La *esfera primaria* es la esfera de menor radio, s_0
- A las tres bi-esferas que delimitan la tri-esfera se les llama *aristas esféricas*
- El *offset* de la tri-esfera es el radio de la esfera primaria, $s_0.r$
- El *plano de centros* es el plano definido por los centros de las tres esferas, y la *normal al plano de centros* es el vector unitario perpendicular a los ejes de las tres aristas esféricas y cumple $\hat{\mathbf{n}}_{012} = \hat{\mathbf{v}}_{01} \times \hat{\mathbf{v}}_{02}$
- Cada uno de los dos *planos tangentes* de una tri-esfera está definido por un vector normal que es el eje de uno de los *vectores esféricos normales* a la tri-esfera, definidos por $\mathbf{s}_{012} = \mathbf{s}_{01} \times \mathbf{s}_{02}$ y que es perpendiculares ambos a las aristas esféricas.

- El *grado de convergencia* de la tri-esfera se define mediante el valor $\eta_{012} = \hat{\mathbf{n}}_{012} \cdot \hat{\mathbf{v}}_{012}$ donde $\mathbf{v}_{012}$ es el eje del vector esférico normal $\mathbf{s}_{012}$.
- El *ángulo de convergencia* de la tri-esfera viene dado por $\gamma_{012} = \cos^{-1} \eta_{012}$

Una tri-esfera puede degenerar en una simple bi-esfera cuando uno de sus vértices esféricos está completamente contenido por la bi-esfera definida por los otros dos vértices esféricos.

■ Una *tetra-esfera* es una poli-esfera formada por cuatro esferas (Figura 2.7). La notación para una tetra-esfera es similar a la de los casos anteriores: una tetra-esfera formada por los vértices esféricos s_0, s_1, s_2 y s_3 se denota por el símbolo S_{0123}. La dimensión de una tetra-esfera irreducible es 3 (φ=3). La ecuación para una tetra-esfera es similar a la de la tri-esfera,

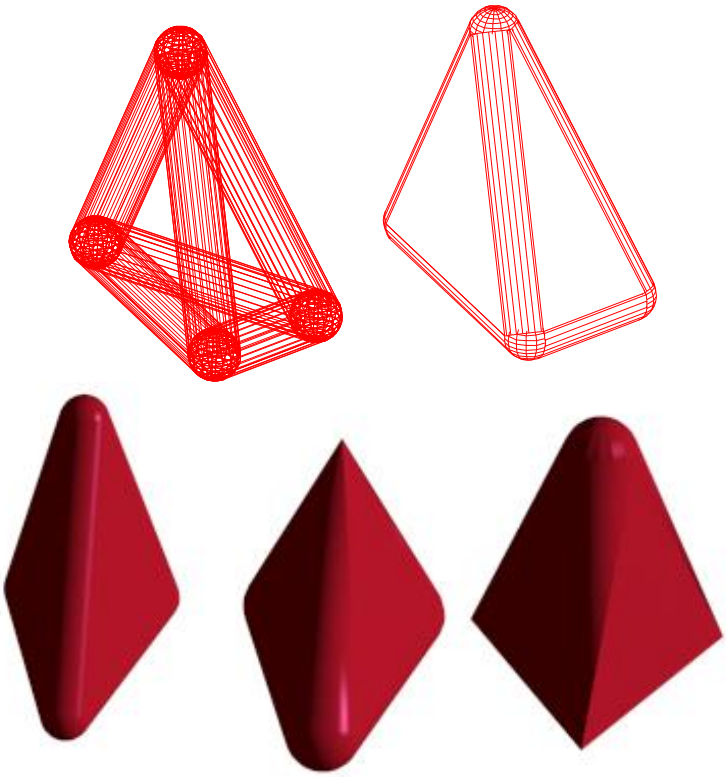


Figura 2.7. Tetra-Esferas. *En la parte superior se muestra las representaciones alámbrica y superficial de un tetra-esfera, donde se puede observar claramente que la tetra-esfera está generada a partir de cuatro esferas. En la parte inferior se muestran tres posibilidades de modelado con tetra-esferas, mediante una representación sombreada de un modelo sólido, donde la tetra-esfera de la izquierda se forma a partir de cuatro esferas de idéntico radio, la central tiene una esfera de radio nulo y la tetra-esfera de la derecha tiene tres esferas de radio nulos, produciendo que tres aristas esféricas sean segmentos rectilíneos.*

$$S_{0123} = \left\{ s \in \Omega : s = s_0 + \lambda_1 \mathbf{s}_{01} + \lambda_2 \mathbf{s}_{02} + \lambda_3 \mathbf{s}_{03},\ 0 \le \lambda_1, \lambda_2, \lambda_3,\ \lambda_1 + \lambda_2 + \lambda_3 \le 1 \right\}$$

Las esferas barridas por una tetra-esfera se obtienen con la ecuación

$$S_{0123}(\lambda_1, \lambda_2, \lambda_3) = \begin{bmatrix} 1 & \lambda_1 & \lambda_2 & \lambda_3 \end{bmatrix} \begin{bmatrix} 1 & -1 & -1 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = s_0 + \begin{bmatrix} \lambda_1 & \lambda_2 & \lambda_3 \end{bmatrix} \begin{bmatrix} \mathbf{s}_{01} \\ \mathbf{s}_{02} \\ \mathbf{s}_{03} \end{bmatrix}$$

Por tanto, una tetra-esfera puede ser generada a partir de una esfera primaria s_0 y tres vectores esféricos $\mathbf{s}_{01}$, $\mathbf{s}_{02}$ y $\mathbf{s}_{03}$.

- Una *penta-esfera* es una poli-esfera formada a partir de cinco vértices esféricos s_0, s_1, s_2, s_3 y s_4, denotada por $S_{0\text{-}4}$. Cuando sea irreducible, su dimensión es 4. La ecuación de la penta-esfera es

$$S_{0-4} = \left\{ s \in \Omega : s = s_0 + \sum_{i=1}^{4} \lambda_i \mathbf{s}_{0i},\ 0 \le \lambda_1, \lambda_2, \lambda_3, \lambda_4,\ \sum_{i=1}^{4} \lambda_i \le 1 \right\}$$

Las esferas barridas por una tetra-esfera se obtienen con la ecuación

$$S_{0-4}(\lambda_1, \ldots, \lambda_4) = \begin{bmatrix} 1 & \lambda_1 & \cdots & \lambda_4 \end{bmatrix} \begin{bmatrix} 1 & -1 & -1 & -1 & -1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_0 \\ \vdots \\ s_4 \end{bmatrix} = s_0 + \begin{bmatrix} \lambda_1 & \cdots & \lambda_4 \end{bmatrix} \begin{bmatrix} \mathbf{s}_{01} \\ \vdots \\ \mathbf{s}_{04} \end{bmatrix}$$

El volumen engendrado por una penta-esfera puede ser obtenido siempre a partir de una o dos tetra-esferas. Sin embargo, las esferas barridas no, ya que su ecuación tiene una componente en el radio que la diferencia, generando esferas en su interior.

2.4. Modelos Esféricos No Lineales

Las poli-esferas definidas en la sección anterior se basan en la utilización de una relación lineal en el espacio paramétrico a partir de vectores esféricos obtenidos de un conjunto de esferas. Es posible ampliar considerablemente el dominio de objetos representables a partir de un conjunto de esferas utilizando relaciones no lineales de segundo o tercer grado

(e incluso grados mayores) en el espacio paramétrico. El conjunto de volúmenes representable de esta forma constituye una extensión esférica a los cilindros generalizados [KZB85] [K&N90] y los conos generalizados [Bro83a], y serán de utilidad para modelar brazos-robot completos, con utilidad en la detección de colisiones con obstáculos de su entorno y/o otros brazos-robot.

2.4.1. Esferoides Cuadráticas

Considerando una relación cuadrática en un único parámetro λ, para generar un volumen se requieren tres restricciones, es decir tres esferas, s_0, s_1, s_2, denominadas *esferas de control*, o lo que es lo mismo, una esfera primaria s_0 y dos vectores esféricos, $\mathbf{s}_{01}$ y $\mathbf{s}_{02}$, denominando *esferoide cuadrática*, S^2_{012}, al volumen generado, compuesto de infinitas esferas según:

$$S^2_{012} = \left\{ s \in \Omega : s = s_0 + \lambda \mathbf{s}_{01} + \lambda^2 \mathbf{s}_{02},\ 0 \le \lambda \le 1 \right\}$$

La formulación matricial de la esferoide cuadrática que define todas las esferas barridas es:

$$S^2_{012}(\lambda) = \begin{bmatrix} 1 & \lambda & \lambda^2 \end{bmatrix} \begin{bmatrix} s_0 \\ \mathbf{s}_{01} \\ \mathbf{s}_{02} \end{bmatrix},\ 0 \le \lambda \le 1$$

Las esferas de control s_0, s_1 y s_2 se pueden obtener a partir de las dos esferas extremas de la esferoide cuadrática, σ_0 y σ_2, y de una esfera interior σ_1, necesaria para completar la definición del volumen, de forma que $S^2_{012}(0) = \sigma_0$, $S^2_{012}(1) = \sigma_2$ y $S^2_{012}(\lambda^*) = \sigma_1$ para un λ^* fijo entre 0 y 1. El valor de λ^* se considera relativo a las normas de los vectores esféricos $\mathbf{s}_{01}$ y $\mathbf{s}_{12}$, de forma que si se toman $d_{01}=\|\sigma_{01}\|_s$, $d_{12}=\|\sigma_{12}\|_s$ y $d_{02}=d_{01}+d_{12}$ entonces $\lambda^*=d_{01}/d_{02}$. Con este planteamiento se pueden obtener las tres ecuaciones siguientes:

$$\left.\begin{array}{ll} \lambda = 0 & \Rightarrow S^2_{012}(0) = s_0 = \sigma_0 \\ \lambda^* = \dfrac{d_{01}}{d_{02}} & \Rightarrow S^2_{012}\left(\dfrac{d_{01}}{d_{02}}\right) = s_0 + \mathbf{s}_{01}\dfrac{d_{01}}{d_{02}} + \mathbf{s}_{02}\dfrac{d_{01}^2}{d_{02}^2} = \sigma_1 \\ \lambda = 1 & \Rightarrow S^2_{012}(1) = s_0 + \mathbf{s}_{01} + \mathbf{s}_{02} = \sigma_2 \end{array}\right\} \Rightarrow \left\{\begin{array}{l} s_0 = \sigma_0 \\ s_1 = \sigma_0 + \dfrac{d_{02}^2}{d_{01}d_{12}}\boldsymbol{\sigma}_{01} - \dfrac{d_{02}}{d_{12}}\boldsymbol{\sigma}_{02} \\ s_2 = \sigma_0 - \dfrac{d_{02}^2}{d_{01}d_{12}}\boldsymbol{\sigma}_{01} + \dfrac{d_{02}}{d_{12}}\boldsymbol{\sigma}_{02} \end{array}\right.$$

Por lo que la esferoide cuadrática se puede expresar también en función de su esfera extrema inicial σ_0, y de dos vectores esféricos, σ_{01} definido desde σ_0 a una esfera intermedia σ_1 y σ_{02} definido desde σ_0 a la esfera extrema final σ_2 como:

$$S^2_{012}(\lambda) = \begin{bmatrix} 1 & \lambda & \lambda^2 \end{bmatrix} \begin{bmatrix} s_0 \\ \mathbf{s}_{01} \\ \mathbf{s}_{02} \end{bmatrix} = \begin{bmatrix} 1 & \lambda & \lambda^2 \end{bmatrix} \begin{bmatrix} P_2 \end{bmatrix} \begin{bmatrix} \sigma_0 \\ \boldsymbol{\sigma}_{01} \\ \boldsymbol{\sigma}_{02} \end{bmatrix}, \quad 0 \le \lambda \le 1 \quad \text{con}$$

$$\begin{bmatrix} P_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \dfrac{d_{02}^2}{d_{01} d_{12}} & -\dfrac{d_{01}}{d_{12}} \\ 0 & -\dfrac{d_{02}^2}{d_{01} d_{12}} & \dfrac{d_{02}}{d_{12}} \end{bmatrix}$$

En la Figura 2.8 se ilustran las posibilidades de modelado mediante esferoides cuadráticas.

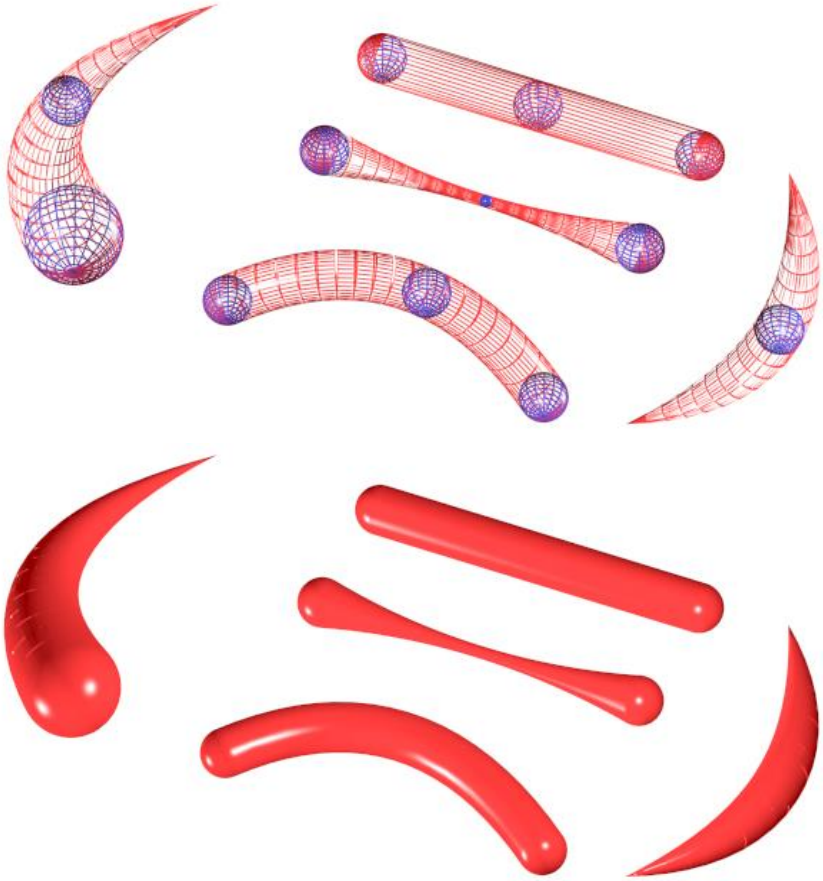


Figura 2.8. Esferoides Cuadráticas. *En la parte superior se muestra la representación alámbrica de cinco esferoides cuadráticas, donde se puede observar que todas están generadas a partir de tres esferas, dos esferas extremo y una intermedia. En la parte inferior se muestran una representación sombreada de sus modelos sólidos. Los diferentes ejemplos ilustran la potencia de modelado que se puede alcanzar en el caso de las esferoides cuadráticas, incluyendo como caso particular a la bi-esfera cuando los centros y radios de las esferas mantienen una relación lineal.*

2.4.2. Esferoides Cúbicas

Si para utilizar una relación cuadrática se necesitan tres esferas, con cuatro esferas se puede utilizar una relación cúbica, obteniendo una *esferoide cúbica*. En este caso se parte de las cuatro esferas de control, s_0, s_1, s_2 y s_3, o bien de una esfera primaria s_0 y tres vectores esféricos, $\mathbf{s}_{01}$, $\mathbf{s}_{02}$ y $\mathbf{s}_{03}$, siendo S^3_{0123}, el volumen generado por la esferoide cúbica, compuesto de infinitas esferas según:

$$S^3_{0123} = \left\{ s \in \Omega : s = s_0 + \lambda \mathbf{s}_{01} + \lambda^2 \mathbf{s}_{02} + \lambda^3 \mathbf{s}_{03},\ 0 \le \lambda \le 1 \right\}$$

La formulación matricial de la esferoide cúbica que define todas las esferas barridas es:

$$S^3_{0123}(\lambda) = \begin{bmatrix} 1 & \lambda & \lambda^2 & \lambda^3 \end{bmatrix} \begin{bmatrix} s_0 \\ \mathbf{s}_{01} \\ \mathbf{s}_{02} \\ \mathbf{s}_{03} \end{bmatrix},\ 0 \le \lambda \le 1$$

También en este caso las esferas de control s_0, s_1, s_2 y s_3 se pueden obtener a partir de las dos esferas extremas de la esferoide cúbica, σ_0 y σ_3, y de dos esferas interiores σ_1 y σ_2, necesarias para completar la definición del volumen, partiendo de $S^3_{0123}(0) = \sigma_0$, $S^3_{0123}(1) = \sigma_3$ y $S^3_{0123}(\lambda^*) = \sigma_1$ y $S^3_{0123}(\lambda') = \sigma_2$ para un λ^* y λ' fijo entre 0 y 1. Los valores de λ^* y λ' se consideran relativos a las normas de los vectores esféricos $\mathbf{s}_{01}$, $\mathbf{s}_{12}$ y $\mathbf{s}_{23}$, de forma que si se toman $d_{01}=\|\sigma_{01}\|_s$, $d_{12}=\|\sigma_{12}\|_s$, $d_{23}=\|\sigma_{23}\|_s$, $d_{02}=d_{01}+d_{12}$, $d_{13}=d_{12}+d_{23}$ y $d_{03}=d_{01}+d_{12}+d_{23}$ entonces $\lambda^*=d_{01}/d_{03}$ y $\lambda'=d_{02}/d_{03}$. Con este planteamiento se pueden obtener las cuatro ecuaciones siguientes:

$$\left.\begin{aligned}
&\lambda = 0 \quad \Rightarrow S^3_{0123}(0) = s_0 = \sigma_0 \\
&\lambda^* = \frac{d_{01}}{d_{03}} \Rightarrow S^3_{0123}\left(\frac{d_{01}}{d_{03}}\right) = s_0 + \mathbf{s}_{01}\frac{d_{01}}{d_{03}} + \mathbf{s}_{02}\frac{d_{01}^2}{d_{03}^2} + \mathbf{s}_{03}\frac{d_{01}^3}{d_{03}^3} = \sigma_1 \\
&\lambda' = \frac{d_{02}}{d_{03}} \Rightarrow S^3_{0123}\left(\frac{d_{02}}{d_{03}}\right) = s_0 + \mathbf{s}_{01}\frac{d_{02}}{d_{03}} + \mathbf{s}_{02}\frac{d_{02}^2}{d_{03}^2} + \mathbf{s}_{03}\frac{d_{02}^3}{d_{03}^3} = \sigma_2 \\
&\lambda = 1 \quad \Rightarrow S^3_{0123}(1) = s_0 + \mathbf{s}_{01} + \mathbf{s}_{02} + \mathbf{s}_{03} = \sigma_3
\end{aligned}\right\}$$

de donde se obtiene

$$\begin{cases} s_0 = \sigma_0 \\ s_1 = \sigma_0 + \dfrac{d_{02}\,d_{03}^2}{d_{01}\,d_{12}\,d_{13}}\boldsymbol{\sigma}_{01} - \dfrac{d_{01}\,d_{03}^2}{d_{02}\,d_{12}\,d_{23}}\boldsymbol{\sigma}_{02} + \dfrac{d_{01}\,d_{02}}{d_{13}\,d_{23}}\boldsymbol{\sigma}_{03} \\ s_2 = \sigma_0 - \dfrac{d_{03}^2(d_{02}+d_{03})}{d_{01}\,d_{12}\,d_{13}}\boldsymbol{\sigma}_{01} + \dfrac{d_{03}^2(d_{01}+d_{03})}{d_{02}\,d_{12}\,d_{23}}\boldsymbol{\sigma}_{02} - \dfrac{d_{03}(d_{01}+d_{02})}{d_{13}\,d_{23}}\boldsymbol{\sigma}_{03} \\ s_3 = \sigma_0 + \dfrac{d_{03}^3}{d_{01}\,d_{12}\,d_{13}}\boldsymbol{\sigma}_{01} - \dfrac{d_{03}^3}{d_{02}\,d_{12}\,d_{23}}\boldsymbol{\sigma}_{02} + \dfrac{d_{03}^2}{d_{13}\,d_{23}}\boldsymbol{\sigma}_{03} \end{cases}$$

Por lo que la esferoide cúbica se puede expresar también en función de su esfera extrema inicial σ_0, y de los vectores esféricos, $\boldsymbol{\sigma}_{01}$ y $\boldsymbol{\sigma}_{02}$ definidos desde σ_0 a unas esferas intermedias σ_1 y σ_2, y $\boldsymbol{\sigma}_{03}$ definido desde σ_0 a la esfera extrema final σ_3 como:

$$S_{0123}^3(\lambda) = \begin{bmatrix} 1 & \lambda & \lambda^2 & \lambda^3 \end{bmatrix} \begin{bmatrix} s_0 \\ \mathbf{s}_{01} \\ \mathbf{s}_{02} \\ \mathbf{s}_{03} \end{bmatrix} = \begin{bmatrix} 1 & \lambda & \lambda^2 & \lambda^3 \end{bmatrix} \begin{bmatrix} P_3 \end{bmatrix} \begin{bmatrix} \sigma_0 \\ \boldsymbol{\sigma}_{01} \\ \boldsymbol{\sigma}_{02} \\ \boldsymbol{\sigma}_{03} \end{bmatrix},\ 0 \le \lambda \le 1$$

donde

$$\begin{bmatrix} P_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \dfrac{d_{02}\,d_{03}^2}{d_{01}\,d_{12}\,d_{13}} & -\dfrac{d_{01}\,d_{03}^2}{d_{02}\,d_{12}\,d_{23}} & \dfrac{d_{01}\,d_{02}}{d_{13}\,d_{23}} \\ 0 & -\dfrac{d_{03}^2(d_{02}+d_{03})}{d_{01}\,d_{12}\,d_{13}} & \dfrac{d_{03}^2(d_{01}+d_{03})}{d_{02}\,d_{12}\,d_{23}} & -\dfrac{d_{03}(d_{01}+d_{02})}{d_{13}\,d_{23}} \\ 0 & \dfrac{d_{03}^3}{d_{01}\,d_{12}\,d_{13}} & -\dfrac{d_{03}^3}{d_{02}\,d_{12}\,d_{23}} & \dfrac{d_{03}^2}{d_{13}\,d_{23}} \end{bmatrix}$$

En la Figura 2.9 se ilustran las posibilidades de modelado mediante esferoides cúbicas. Es de resaltar que las esferoides cúbicas amplían notoriamente la flexibilidad de modelado, ya que al no existir relación de coplanaridad entre los centros de las esferas, se pueden alcanzar modelos que se retuerzan en el espacio o que presenten inflexiones entre concavidad y convexidad. Por contra, en el caso de las esferoides cuadráticas, esto no era posible, teniendo siempre relación de coplanaridad entre centros (y entre cualquier trio de dimensiones incluyendo el radio) y el volumen generado presentaba propiedades de concavidad o convexidad, pero nunca ambas.

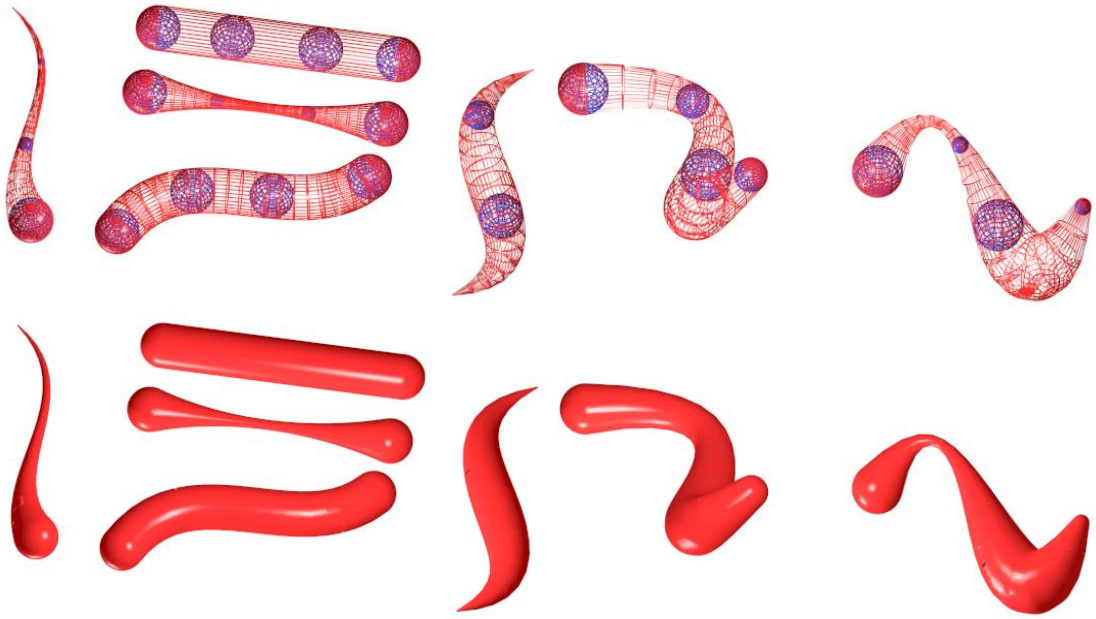


Figura 2.9. Esferoides Cúbicas. *En la parte superior se muestra la representación alámbrica de siete esferoides cúbicas, donde se puede observar que todas están generadas a partir de cuatro esferas, dos esferas extremo y dos intermedias. En la parte inferior se muestran una representación sombreada de sus modelos sólidos. Los diferentes ejemplos ilustran la potencia de modelado que se puede alcanzar en el caso de las esferoides cúbicas, incluyendo como caso particular a las esferoides cuadráticas. En los dos casos de la derecha, los centros no son coplanares, obteniéndose volúmenes que se retuercen en el espacio, con zonas cóncavas y convexas.*

2.4.3. Esferoides Spline

Cuando el número de esferas de control supera las cuatro, no resulta interesante ampliar el grado excesivamente, por lo que se puede utilizar la conocida técnica de *interpolación por splines*. Esta técnica se basa en conseguir una serie de segmentos mediante formulación cúbica que en global generen un único modelo con interpolación de los datos de entrada. Normalmente esta técnica es muy utilizada en sistemas CAD para el modelado de curvas [dBo78], [BBB87], donde los datos de entrada son puntos a interpolar. Igualmente se ha venido utilizando para la especificación de trayectorias en sistemas robotizados, bien la trayectoria del elemento terminal de un brazo-robot en el espacio cartesiano, bien la trayectoria del robot en el espacio de configuraciones [W&J88] [M&T91] [VTM92].

Existen muchos métodos de interpolación por segmentos, con diferentes bases, como por ejemplo, exigir que los segmentos consecutivos tengan la primera derivada

continua. El éxito de la técnica de splines es tomar como premisa que dichos segmentos tengan tanto primera como segunda derivada continua. Así se consigue "suavizar" el modelo resultante, sin apenas diferenciación entre los segmentos.

La aplicación de la interpolación mediante splines para el caso de modelos esféricos [M&T93], requiere para interpolar m+1 esferas de control $\sigma_0,...,\sigma_m$, una secuencia de segmentos cúbicos, cuyo modelo global se denomina *esferoide spline*. Cada uno de los m segmentos será una esferoide cúbica, que queda completamente determinada por cuatro coeficientes esféricos, $S_i^3(\lambda) = s_{i,0} + \mathbf{s}_{i,01}\lambda + \mathbf{s}_{i,02}\lambda^2 + \mathbf{s}_{i,03}\lambda^3$, i=0,...,m-1, recorriendo la esferoide $S_i^3(\lambda)$ la zona que va desde la esfera σ_i a la esfera σ_{i+1}. Al haber m segmentos, en total se necesitan 4m ecuaciones para poder resolver las 4m incógnitas existentes.

En cada uno de los m-1 esferas interiores, $\sigma_1,...,\sigma_{m-1}$, donde se conectan dos segmentos, se va a exigir que se cumplan cuatro condiciones para i=1,...,m-1:

- El valor inicial del (siguiente) segmento, $S_i^3(\lambda)$, es la esfera a interpolar, σ_i:
$$S_i^3(0) = \sigma_i \Rightarrow s_{i,0} = \sigma_i$$

- El valor final del (anterior) segmento, $S_{i-1}^3(\lambda)$, es la esfera a interpolar, σ_i:
$$S_{i-1}^3(1) = \sigma_i \Rightarrow s_{i-1,0} + \mathbf{s}_{i-1,01} + \mathbf{s}_{i-1,02} + \mathbf{s}_{i-1,03} = \sigma_i$$

- Dos segmentos consecutivos, $S_{i-1}^3(\lambda)$ y $S_i^3(\lambda)$, tienen continuidad en primera derivada:
$$\left.\frac{dS_{i-1}^3(\lambda)}{d\lambda}\right|_{\lambda=1} = \left.\frac{dS_i^3(\lambda)}{d\lambda}\right|_{\lambda=0} \Rightarrow \mathbf{s}_{i-1,01} + 2\mathbf{s}_{i-1,02} + 3\mathbf{s}_{i-1,03} = \mathbf{s}_{i,01}$$

- Dos segmentos consecutivos, $S_{i-1}^3(\lambda)$ y $S_i^3(\lambda)$, tienen continuidad en segunda derivada:
$$\left.\frac{d^2S_{i-1}^3(\lambda)}{d\lambda^2}\right|_{\lambda=1} = \left.\frac{d^2S_i^3(\lambda)}{d\lambda^2}\right|_{\lambda=0} \Rightarrow 2\mathbf{s}_{i-1,02} + 6\mathbf{s}_{i-1,03} = 2\mathbf{s}_{i,02}$$

Además, en las esferas extremas σ_0 y σ_m tenemos las siguientes condiciones:

- El valor inicial del primer segmento, $S_0^3(\lambda)$, es la primera esfera a interpolar, σ_0:
$$S_0^3(0) = \sigma_0 \Rightarrow s_{i-1,0} = \sigma_0$$

- El valor final del último segmento, $S_{m-1}^3(\lambda)$, es la última esfera a interpolar, σ_m:
$$S_{m-1}^3(1) = \sigma_m \Rightarrow s_{m-1,0} + \mathbf{s}_{m-1,01} + \mathbf{s}_{m-1,02} + \mathbf{s}_{m-1,03} = \sigma_m$$

Con todas estas condiciones se han obtenido un total de 4m-2 ecuaciones, por lo que se requieren dos condiciones más para poder resolver el sistema.

Existe una gran variedad de formas para fijar las dos condiciones que faltan, pero normalmente estas condiciones se aplican en los extremos de la curva, y dan lugar a diferentes tipos de splines. La técnica más utilizada es la llamada *splines cúbicas naturales*, que exigen segunda derivada en los extremos nula. Esto implica dos nuevas ecuaciones:

- El valor inicial de la segunda derivada del primer segmento, $S_0^3(\lambda)$, es el vector esférico nulo:

$$\left.\frac{d^2 S_0^3(\lambda)}{d\lambda^2}\right|_{\lambda=0} = \mathbf{s}_\varnothing \Rightarrow 2\mathbf{s}_{0,02} = \mathbf{s}_\varnothing$$

- El valor final de la segunda derivada del último segmento, $S_{m-1}^3(\lambda)$, es el vector esférico nulo:

$$\left.\frac{d^2 S_{m-1}^3(\lambda)}{d\lambda^2}\right|_{\lambda=1} = \mathbf{s}_\varnothing \Rightarrow 2\mathbf{s}_{m-1,02} + 6\mathbf{s}_{m-1,03} = \mathbf{s}_\varnothing$$

Resolver este sistema implica resolver un sistema de 4m ecuaciones con 4m incógnitas (las esferas $s_{i,0}$ y los vectores esféricos $\mathbf{s}_{i,01}$, $\mathbf{s}_{i,02}$ y $\mathbf{s}_{i,03}$ para cada uno de los m segmentos), por lo que es bastante complejo para m grande. Existen métodos más adecuados para resolver las splines [BBB87] mediante dos niveles:

- Replantear el sistema de 4m ecuaciones con 4 m incógnitas a un sistema de m+1 ecuaciones con m+1 incógnitas nuevas, denominadas en el caso esférico $\mathbf{S}_i$, i=0,...,m, que representan el valor de los vectores esféricos derivada en las esferas a interpolar.
- Resolver el nuevo sistema y plantear un caso de interpolación que parta de derivadas conocidas, como por ejemplo el de Hermite, que tiene una resolución cómoda.

En cierta forma, lo que se está consiguiendo con este método es obtener cuales son las derivadas en las esferas de control de la esferoide spline segmentada que se quiere encontrar para que mantenga continuidad en su segunda derivada.

Para el caso esférico, el sistema de m+1 ecuaciones que se obtiene es

$$\begin{bmatrix} 2 & 1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ 1 & 4 & 1 & \ddots & & & & & \vdots \\ 0 & 1 & 4 & 1 & \ddots & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & & \ddots & 1 & 4 & 1 & 0 \\ \vdots & & & & & \ddots & 1 & 4 & 1 \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} \mathbf{S}_0 \\ \mathbf{S}_1 \\ \mathbf{S}_2 \\ \vdots \\ \mathbf{S}_{m-2} \\ \mathbf{S}_{m-1} \\ \mathbf{S}_m \end{bmatrix} = \begin{bmatrix} 3\sigma_{01} \\ 3\sigma_{02} \\ 3\sigma_{13} \\ 3\sigma_{24} \\ \vdots \\ 3\sigma_{m-4,m-2} \\ 3\sigma_{m-3,m-1} \\ 3\sigma_{m-2,m} \\ 3\sigma_{m-1,m} \end{bmatrix}$$

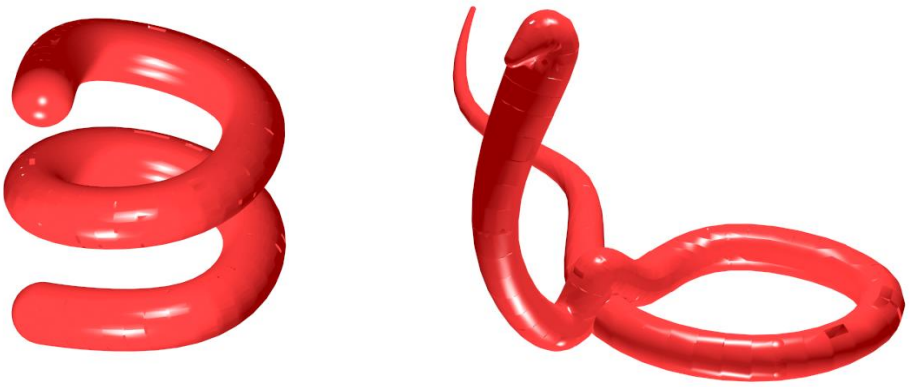


Figura 2.10. Esferoides Spline. *En esta figura se muestran la representación sombreada de dos modelos sólidos de esferoides spline. La esferoide spline de la izquierda está generada a partir de ocho esferas de control, dando una forma de tipo muelle, donde el radio se ha mantenido constante. En la esferoide spline de la derecha se puede apreciar claramente la variación de radio de la misma, obteniéndose un modelo con forma de serpiente.*

Aparte de que este método de resolución es más rápido que el inicial, otra ventaja adicional es que la matriz de este sistema de ecuaciones está bien-condicionada. Resolviendo este sistema, se obtienen los vectores esféricos derivada, $\mathbf{S}_i$, i=0,...,m, que se utilizan para resolver todas las esferoides cúbicas sustituyéndose en las ecuaciones siguientes:

$$s_{i,0} = \sigma_i$$
$$\mathbf{s}_{i,01} = \mathbf{S}_i$$
$$\mathbf{s}_{i,02} = 3\sigma_{i,i+1} - 2\mathbf{S}_i - \mathbf{S}_{i+1}$$
$$\mathbf{s}_{i,03} = 2\sigma_{i,i+1} + \mathbf{S}_i + \mathbf{S}_{i+1}$$

Dos ejemplos de esferoides spline aparecen en la Figura 2.10. Obviamente, por construcción la esferoide spline está compuesta de una secuencia de esferoides cúbicas que actúan como segmentos, pero manteniendo continuidad hasta segunda derivada.

2.4.4. Extensiones Esféricas Bi-Paramétricas y Tri-Paramétricas

Las esferoides definidas en la secciones anteriores se basan en la utilización de una relación cuadrática o cúbica mono-paramétrica a partir de vectores esféricos obtenidos de un conjunto de esferas. Es posible ampliar considerablemente el dominio de objetos representables a partir de un conjunto de esferas utilizando relaciones de segundo o tercer grado (e incluso grados mayores) sobre dos o más parámetros. El conjunto de volúmenes representable de esta forma serán de utilidad para modelar barridos en las trayectorias de movimiento de un robot y así detectar colisiones con los obstáculos de su entorno.

Considerando una relación cuadrática con dos parámetros se obtiene la denominada *superficie esferoide bi-cuadrática,* S^{22}_{0-8}, definida por nueve vectores esféricos $\mathbf{s}_0$, ... , $\mathbf{s}_8$:

$$S^{22}_{0-8}(\lambda_1,\lambda_2)=\left\{s\in\Omega: s=\sum_{i=0}^{2}\sum_{j=0}^{2}\lambda_1^i\lambda_2^j\mathbf{s}_{3i+j}, 0\le\lambda_1,\lambda_2\le 1\right\}$$

Cuando la relación es cúbica, se requieren dieciséis vectores esféricos, dando la llamada *superficie esferoide bi-cúbica,* S^{33}_{0-15}, como la mostrada en la Figura 2.11:

$$S^{33}_{0-15}(\lambda_1,\lambda_2)=\left\{s\in\Omega: s=\sum_{i=0}^{3}\sum_{j=0}^{3}\lambda_1^i\lambda_2^j\mathbf{s}_{4i+j}, 0\le\lambda_1,\lambda_2\le 1\right\}$$

También se puede generar una *superficie esferoide cuadrática-cúbica*, siendo cuadrática en un parámetro y cúbica en otro, por lo que requiere un total de doce vectores esféricos:

$$S^{23}_{0-11}(\lambda_1,\lambda_2)=\left\{s\in\Omega: s=\sum_{i=0}^{2}\sum_{j=0}^{3}\lambda_1^i\lambda_2^j\mathbf{s}_{4i+j}, 0\le\lambda_1,\lambda_2\le 1\right\}$$

Como generalización de las esferoides splines, también se pueden generar *superficies esferoides spline* como unión de "parches" de superficies esferoides bi-cúbicas, manteniendo continuidad hasta la segunda derivada.

Si se consideran tres parámetros, se tienen relaciones similares, dando lugar a *hiper-volúmenes esferoides tri-cuadráticos, tri-cúbicos* o de una combinación de órdenes. Así se tiene, por ejemplo, que un hiper-volumen esferoide tri-cuadrático tendría la formulación siguiente:

$$S^{222}_{0-26}(\lambda_1,\lambda_2,\lambda_3)=\left\{s\in\Omega: s=\sum_{i=0}^{2}\sum_{j=0}^{2}\sum_{k=0}^{2}\lambda_1^i\lambda_2^j\lambda_3^k\mathbf{s}_{9i+3j+k}, 0\le\lambda_1,\lambda_2,\lambda_3\le 1\right\}$$

Se puede observar que con una complejidad sencilla, relaciones cuadráticas y/o cúbicas, se puede alcanzar un dominio de objetos muy elevado, requiriendo muy poca memoria para una precisión aceptable.

Figura 2.11. Superficie Esferoide Bi-Cúbica. *En la figura se muestra una representación sombreada de una superficie esferoide bi-cúbica. Esta superficie esferoide bi-cúbica está definida mediante una esfera yseis vectores esféricos. Una de sus propiedades es que sus cuatro contornos volumétricos son esferoides cúbicas. Con este tipo de superficies esferoides se puede llegar a representar un dominio de objetos muy elevado con un coste pequeño (ecuaciones de orden cúbico).*

√

CAPÍTULO 3.
MODELADO DE SISTEMAS ROBOTIZADOS

El objetivo de este capítulo es presentar procedimientos que permiten generar automáticamente el modelado esférico del mundo (el robot y los objetos de su entorno). En concreto se ha desarrollado una técnica de minimización del volumen envolvente que presenta la originalidad de trabajar en dos niveles, el superior basado en el método de optimización Downhill Simplex y el inferior en la transformada de Hough. Para obtener unos valores iniciales adecuados, así como para obtener un modelo con mayor rapidez se aplican técnicas de agrupamiento al cálculo del modelo esférico. Finalmente se desarrolla una estructura jerárquica extendida con diferentes grados de precisión para la representación de una célula flexible. También se plantean métodos de reducción de los modelos esféricos de cara a obtener modelos más simples.

Robbie asintió con la cabeza..., un pequeño paralelepípedo con bordes y ángulos redondeados unido a otro paralelepípedo similar pero mucho más grande que le servía de torso mediante un tallo corto y flexible..., y se puso obedientemente cara al árbol. Una delgada película metálica descendió sobre sus resplandecientes ojos, y del interior de su cuerpo brotó un regular y resonante tictaqueo.

"Robbie". Isaac Asimov, 1940.

3.1. Generación de Poli-Esferas Envolventes

Para las aplicaciones de detección de colisiones y planificación de movimientos en sistemas robotizados, los modelos esféricos presentados en el capítulo anterior se utilizan como envolventes de los sistemas reales, de forma que faciliten el cálculo de distancias. Dos posibilidades pueden surgir, objetos del entorno conocidos y objetos desconocidos que son obtenidos por sensorización. En el primer caso, se parte de un modelo lo más exacto posible del sistema real, lo que posibilita un procesamiento *off-line* para buscar el modelo envolvente óptimo. Es el caso de aquellos elementos del entorno del sistema robotizado fijos, como máquinas-herramientas, dispositivos de transporte, sistemas de almacenaje, etcétera.

En el segundo caso, se trata de objetos desconocidos que aparecen repentinamente en el entorno del sistema robotizado, siendo el sistema de sensorización disponible el que se encarga de obtener puntos de la superficie del objeto, a partir de los cuales hay que obtener, en el mínimo tiempo posible, un modelo envolvente. En las siguientes secciones se analiza la forma de obtener modelos esféricos envolventes y se distinguen técnicas para cada uno de los casos posibles.

3.1.1. Cálculo de una Poli-Esfera Envolvente con Volumen Mínimo

Para obtener un modelo envolvente de un objeto conocido mediante poli-esferas, se parte de una representación lo más precisa posible del objeto, mediante un conjunto de puntos de la frontera del mismo. El problema consiste en una minimización de la función volumen, donde las variables son las coordenadas y radios de los vértices esféricos de la poli-esfera resultante.

El problema se plantea como:

> *Dado un conjunto de puntos* $\mathbf{P}=\{p_0,...,p_N\}$ *que definen un objeto incluido en el politopo convexo* $P_{0\text{-}N}$*, encontrar la poli-esfera de orden* $n+1$ *y volumen mínimo,* $S_{0\text{-}n}$ *de forma que* $P_\infty(P_{0\text{-}N}) \subset S_\infty(S_{0\text{-}n})$

El problema de minimización se puede resolver con la siguiente filosofía:

1.- *Fijar un conjunto inicial de* $n+1$ *puntos,* **P'***, siguiendo algún criterio*
2.- *Calcular la poli-esfera* $S_{0\text{-}n}$ *cuyos vértices esféricos tienen como centros los puntos de* **P'** *y los radios son tales que producen el mínimo volumen necesario para englobar completamente* $P_{0\text{-}n}$
3.- *Recalcular* **P'** *en una dirección que lleve a un volumen menor*
4.- *Repetir los pasos 2..3 hasta convergencia*

Para poder aplicar este proceso hay que determinar:

- Los valores iniciales
- La forma de calcular los radios que producen el mínimo volumen necesario
- La obtención de una dirección adecuada para modificar los centros de las esferas

En [BTM96] se ha presentado, como generalización aplicable a poli-esferas del método expuesto en [B&T95a] y [B&T95b], un procedimiento automático y general capaz de generar la mínima poli-esfera necesaria para envolver a un objeto real dado. Esta técnica encuentra, partiendo de un conjunto de puntos que representan al objeto a aproximar, y del tipo de envolvente deseado, el modelo de ese tipo con mínimo volumen que envuelve a todos los puntos.

La técnica de minimización de volúmenes en cuestión, está desarrollada para que pueda modelar objetos mediante su envolvente, utilizando diversas modelos geométricos, como por ejemplo, poli-esferas, politopos, poliedros, etc.

El método para encontrar el modelo de mínimo volumen, parte de un conjunto de puntos pertenecientes a la superficie del objeto real a modelar, así como la estructura geométrica deseada. La minimización del volumen se consigue en dos niveles jerárquicos: el nivel superior está basado en el método de optimización *Downhill Simplex* [N&M65] y se encarga de buscar la posición y la orientación del modelo envolvente que minimiza una función volumen. El nivel inferior consiste en la aplicación de la Transformada de Hough [FGL85], y utilizando los parámetros proporcionados por el nivel superior (la posición y orientación del modelo), se obtiene la envolvente mínima (para esa posición y orientación), que modela completamente al objeto. El volumen de la envolvente generada será devuelto al nivel superior. Así pues, la mínima envolvente se obtiene a través de un proceso iterativo, donde el nivel superior irá llamando al nivel inferior, hasta encontrar la posición y orientación del modelo mínimo.

El problema de la aproximación del volumen implica que se dispone de un conjunto de N puntos de la superficie del objeto real. Inicialmente, se puede suponer que estos puntos son obtenidos de la base de datos de un sistema CAD. El modelo a obtener finalmente debe ser el de mínimo volumen con la restricción de que todos los puntos del objeto a modelar se encuentren en su interior.

Evidentemente, la complejidad del problema depende de la complejidad del volumen utilizado para la envolvente. Por ejemplo, si el tipo de envolvente deseado es una esfera, se requieren cuatro parámetros (3 para el centro y uno para el radio), mientras que para un cilindro se requieren siete parámetros, o para un politopo con n vértices se requieren 3n parámetros. El caso más general, es el de una poli-esfera con n vértices esféricos, lo que implica un total de 4n parámetros necesarios. Obviamente, el caso más simple implica un modelo con baja precisión, mientras que a mayor complejidad se tiene un modelo más preciso del objeto real. Las poli-esferas presentan en general un buen compromiso entre complejidad y precisión.

Una vez seleccionado el tipo de modelo, el problema de optimización plantea minimizar una función volumen, que depende de m parámetros con N restricciones (una por punto conocido). Las restricciones son ecuaciones cuadráticas, representando que la distancia de un punto al objeto debe ser menor o igual a cero. La función volumen a utilizar generalmente es una ecuación cúbica en m parámetros. Esto posibilita utilizar técnicas de programación no lineal, si bien la complejidad de estas técnicas crece al tomarse valores mayores de m y N.

El método utilizado resuelve este problema mediante dos niveles jerárquicos: el nivel superior se basa en la optimización de una función con m-k parámetros sin restricciones, pudiendo utilizarse el método de optimización Downhill Simplex. El nivel inferior minimiza la función volumen para k parámetros (el nivel superior ha fijado m-k parámetros) con N restricciones. Este nivel consiste en la aplicación de la Transformada de Hough. El primer paso consiste en determinar los k parámetros que no se deben considerar en el nivel superior pero sí en el nivel inferior. Las poli-esferas se adaptan perfectamente a este planteamiento, ya que los k parámetros serán los radios de las n esferas vértices, con k=n.

Para una poli-esfera $S_{0\text{-}n}$ con n+1 vértices esféricos, el problema global tiene m=4(n+1) parámetros, con k=n+1 radios como parámetros utilizados por el nivel inferior, y para el nivel superior q=m-k=3(n+1) parámetros.

El volumen de una poli-esfera se obtiene distinguiendo inicialmente la parte superior e inferior de la poli-esfera. La parte superior consiste en regiones curvas y planas limitadas por aristas en forma de arcos circulares y segmentos rectilíneos. Proyectando un rayo vertical hacia abajo desde cada punto de una arista se generan paredes verticales asociadas a las aristas. Estas paredes descomponen el espacio bajo la envolvente superior en prismas curvos, cuyos volúmenes se pueden calcular en tiempo constante. El número de prismas curvos está linealmente relacionado con el número de caras de la poli-esfera. La suma de estos volúmenes proporciona el volumen total por debajo de la parte superior de la poli-esfera. Realizando el mismo proceso se puede obtener el volumen bajo la parte inferior de la poli-esfera. El volumen ocupado por la poli-esfera se obtiene mediante la diferencia entre ambos volúmenes. Para un caso sencillo como el de la bi-esfera, este volumen se puede calcular mucho más fácilmente, al ser una envolvente de revolución, mediante la siguiente fórmula:

$$V_{S_{01}} = \frac{2}{3}\pi\left(s_0.r^3 + s_1.r^3\right) + \frac{4}{3}\gamma_{01}\left(s_0.r^3 - s_1.r^3\right) + \frac{\pi}{4}\left\|\mathbf{v}_{01}\right\|\cos\gamma_{01}\left(s_0.r + s_1.r\right)$$

En el nivel superior, el método Downhill Simplex trabaja en un espacio q-dimensional con una variable $C=(c_0,...,c_n)$, el conjunto de centros de los vértices esféricos, por lo que q=3(n+1). El proceso parte inicialmente de un simplex, que consiste en q+1 puntos expresados como C_0 y $C_i=C_0+\delta\mathbf{e}_i$, i=1,...,q, donde C_0 es un cierto valor inicial, $\mathbf{e}_1$,..., $\mathbf{e}_q$ son q vectores unitarios linealmente independientes y δ es un parámetro elegido según una escala dimensional característica del problema.

En un primer paso, el nivel superior llama al nivel inferior para calcular los volúmenes mínimos obtenibles por poli-esferas definidas según cada uno de estos q+1 puntos q-dimensionales (conjuntos de centros de esferas). Entonces, y para cada paso del proceso iterativo, el método downhill simplex refleja el punto q-dimensional del simplex que tenga mayor volumen a través de la cara opuesta del simplex hasta un nuevo punto con volumen menor. Por tanto, en el siguiente paso, sólo se ha de encontrar la poli-esfera con mínimo volumen para este nuevo conjunto de centros. Si con esta reflexión no se hubiera producido una mejora en el volumen, se puede expandir o contraer el nuevo punto en su dirección de movimiento. Si tampoco se alcanza una mejora en el volumen, el simplex se puede contraer en todas sus direcciones, acercándose a un mínimo de la función volumen, para lo que se deben realiza q nuevas búsquedas de mínimos volúmenes. Se puede demostrar que una secuencia apropiada de tales pasos siempre converge a un mínimo de la función.

Por otra parte, el nivel inferior será el encargado de, dados los centros de esfera suministrados por el nivel superior, generar la poli-esfera con mínimo volumen posible, considerando las N restricciones impuestas por los N puntos de entrada. Para generar la poli-esfera con mínimo volumen, se requiere sólo calcular los radios de las esferas. Una vez obtenidos éstos, se devuelve al nivel superior el volumen ocupado por la poli-esfera generada. La clave de este nivel se centra en cómo obtener los radios apropiados para que el volumen de la poli-esfera con centros fijados por el nivel superior sea mínimo, cumpliendo siempre las N restricciones impuestas por los puntos de entrada. Este problema se puede resolver mediante la Transformada de Hough.

Para ello, cada punto de entrada p_i se proyecta perpendicularmente a la estructura con dimensión (n+1) formada por los centros de C, $c_0,...,c_n$, calculando el punto normal según:

$$p_i^{\perp} = c_0 + \sum_{j=1}^{n} \lambda_{ij} (c_j - c_0)$$

Este punto normal se puede obtener resolviendo el siguiente sistemas de ecuaciones lineal:

$$\left(p_i^{\perp} - p_i\right) \cdot \left(c_j - c_0\right) = 0, \quad j = 1,...,n$$

De esta forma, para cada punto p_i se obtiene un conjunto de parámetros $(\lambda_{i1},...,\lambda_{in})$, siendo necesario un parámetro adicional d_i para expresar completamente el punto p_i respecto a la estructura (n+1)-dimensional:

$$d_i = \left\| p_i^{\perp} - p_i \right\|$$

Con este proceso, cada punto p_i se puede expresar como el vector $(\lambda_{i1},...,\lambda_{in},d_i)$ respecto a la estructura aportada por el nivel superior.

Se denomina *Half-Way Points (HWP)* a aquellos puntos $p_i=(\lambda_{i1},...,\lambda_{in},d_i)$ que cumplen:

$$\lambda_{ij} > 0, j = 1,\ldots,n, \sum_{j=1}^{n}\lambda_{ij} < 1$$

Del resto de puntos, denominados *Outer-Way Points (OWP)*, se pueden descartar algunos:

- Puntos $p_i=(\lambda_{i1},...,\lambda_{in},d_i)$ tal que $\lambda_{ij} \le 0,\ j = 1,\ldots,n$ fijando $d_0^{min} = \max\left(\left\|p_i - c_0\right\|\right)$.
- Puntos $p_i=(\lambda_{i1},...,\lambda_{in},d_i)$ con $\lambda_{ij} \ge 1,\ \lambda_{ij} \ge 1+\sum_{\substack{k=1\\k\neq j}}^{n}\lambda_{ik}$ fijando $d_j^{min} = \max\left(\left\|p_i - c_j\right\|\right)$.

donde $d_0^{min},\ldots,d_n^{min}$ representan los mínimos radios para los centros $c_0,...,c_n$ que generan una poli-esfera envolvente de todos los puntos OWP. Nótese que siempre se debe verificar que $d_j^{min} \ge 0, j = 0,...,n$.

Los puntos HWP $p_i=(\lambda_{i1},...,\lambda_{in},d_i)$ se transforman en hiperplanos mediante la Transformada de Hough, según:

$$\frac{r_j - r_0}{1} = \frac{d_i - r_0}{\lambda_{ij}} \quad \Rightarrow\ r_j = \left(1-\frac{1}{\lambda_{ij}}\right)r_0 + \frac{d_i}{\lambda_{ij}},\ j = 1,\ldots,n$$

donde $r_0,...,r_n$ son las variables del hiperplano. Estas variables deben ser mayores o igual a cero porque representan los radios incógnitas de los vértices esféricos $s_0,...,s_n$ de la poli-esfera a obtener. A estos hiperplanos se les denomina *Hough Hyper-Planes (HHP)*.

Los hiperplanos HHP cumplen las siguientes características:

- Las pendientes de las rectas en el plano definidas por r_0, r_j son siempre negativas: Todo HHP generado de un HWP $p_i=(\lambda_{i1},..,\lambda_{in},d_i)$, con $\lambda_{ij} > 0,\ j = 1,\ldots,n, \sum_{j=1}^{n}\lambda_{ij} < 1$ cumple que $\frac{1}{\lambda_{ij}} > 1 \ \rightarrow\ 1-\frac{1}{\lambda_{ij}} < 0,\ j = 1,\ldots,n$
- Un HHP determina en $r_0,...,r_n$ el semi-hiperespacio $r_j \ge \left(1-\frac{1}{\lambda_{ij}}\right)r_0 + \frac{d_i}{\lambda_{ij}},\ j = 1,\ldots,n$ donde todos los puntos $(r_0,...,r_n)$ situados en este semi-hiperespacio generan una poli-esfera que envuelve a todos los HWP p_i.

Para los puntos OWP se definen nuevos hiperplanos con $r_j = d_j^{min}$, $j = 1,...,n$ que producen nuevos semi-hiperespacios $r_j \geq d_j^{min}$, $j = 1,...,n$

La unión de todos estos semi-hiperespacios obtenidos determina la región válida de los radios de los vértices esféricos de la poli-esfera con centros $c_0,...,c_n$ que envuelven completamente todos los puntos de entrada. Esta región se denomina *Valid Radii Zone (VRZ)*. Por construcción, se puede comprobar que para cualquier conjunto de valores $(r_0,...,r_n)$, que haga fallar la ecuación de un semi-hiperespacio, existe por lo menos un punto $p_i=(\lambda_{i1},...,\lambda_{in},d_i)$ que no está envuelto por la poli-esfera con vértices esféricos definidos mediante centros $(c_0,...,c_n)$ y radios $(r_0,...,r_n)$.

Una vez determinada la región donde se encuentra los valores de los radios que hacen posible una poli-esfera que envuelva completamente a los puntos de entrada, hay que determinar cuál es la poli-esfera de volumen mínimo dentro de esta zona. Es decir, cuál es el punto $(r_0,...,r_n)$ de esta región que representa una poli-esfera con volumen mínimo. El punto que produce un volumen mínimo, además de cumplir todas las ecuaciones de los semi-hiperespacios, debe verificar al menos la ecuación de un hiperplano para encontrarse en el límite del semi-hiperespacio: si no fuera así, al menos un radio r_j se puede reducir, manteniendo el punto en la región válida y obteniendo una poli-esfera con volumen mínimo que sigue envolviendo a todos los puntos de entrada.

El conjunto de valores $r_0,...,r_n$ que se encuentran en la región VRZ y verifican al menos la ecuación de un hiperplano se denomina *Minimum Volume Locus (MVL)*, es decir el lugar de valores donde va a encontrarse la poli-esfera con mínimo volumen. Nótese que al tener cualquier recta en este hiperplano pendiente negativa, el MVL es convexo visto desde el punto $(r_0,...,r_n)=(0,...,0)$, o cóncavo desde $(r_0,...,r_n)=(+\infty,...,+\infty)$.

Por tanto, una vez fijados los centros por el nivel superior, el nivel inferior calcula los radios que determinan completamente la poli-esfera con mínimo volumen que envuelve a todos los puntos de entrada realizando una búsqueda de aquel punto del MVL con menor resultado en la función volumen. Los radios calculados y el valor del volumen son los resultados devueltos por este nivel al nivel superior.

Por ejemplo, para obtener la bi-esfera (n=1) con mínimo volumen que envuelve a un conjunto de puntos de entrada se tiene:

- La proyección de cada punto de entrada en la línea definida por c_0,c_1 se obtiene mediante la ecuación $\left(p_i^{\perp} - p_i\right)\cdot\left(c_1 - c_0\right) = 0$ dando $\lambda_i = \dfrac{\left(p_i - c_0\right)\left(c_1 - c_0\right)}{\left\|c_1 - c_0\right\|^2}$
- El parámetro d_i mediante $d_i = \left\|p_i^{\perp} - p_i\right\|$ con $p_i^{\perp} = c_0 + \lambda_i(c_1 - c_0)$
- Para aquellos $p_i=(\lambda_i,d_i)$ con $\lambda_i \leq 0$ se determina $d_0^{min} = \max\left(\left\|p_i - c_0\right\|\right)$
- Para aquellos $p_i=(\lambda_i,d_i)$ con $\lambda_i \geq 1$ se determina $d_1^{min} = \max\left(\left\|p_i - c_1\right\|\right)$

- Los HHP serán líneas rectas en el espacio $r_0 \times r_1$ definidas por la ecuación:

$$r_1 = \left(1 - \frac{1}{\lambda_i}\right) r_0 + \frac{d_i}{\lambda_i}$$

- El VRZ viene delimitado por las ecuaciones de todos los semiespacios:

$$r_1 \geq \left(1 - \frac{1}{\lambda_i}\right) r_0 + \frac{d_i}{\lambda_i},\ r_0 \geq d_0^{min} \text{ y } r_1 \geq d_1^{min}$$

- El MVL son aquellos puntos del VRZ que pertenecen al menos a una recta HHP, es decir, que verifican la condición de igualdad en al menos una de las ecuaciones de los semi-espacios.

La Figura 3.1 muestra el espacio $r_0 \times r_1$ con la región VRZ y el lugar MVL. Cada par (r_0, r_1) representa los dos radios que determinan completamente una bi-esfera (los centros vienen fijados por el nivel superior), de la cual se puede calcular su volumen. La envolvente de mínimo volumen se encontrará con un proceso de búsqueda barriendo el MVL hasta encontrar el mínimo volumen. Para esta búsqueda se puede realizar un barrido de todo el MVL o bien aplicar una técnica divide-y-conquistarás.

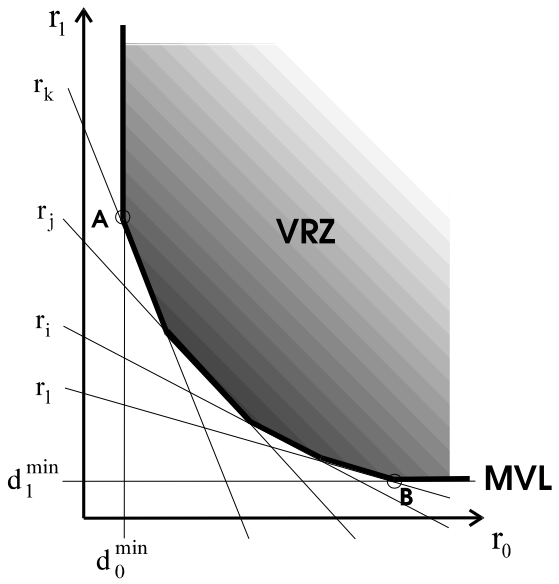


Figura 3.1. Cálculo de la Bi-Esfera con Volumen Mínimo. *En el espacio de los radios $r_0 \times r_1$, cada punto de entrada HWP genera, mediante la Transformada de Hough, un semiespacio delimitado por una recta r_i y los puntos OWP generan los semiespacios que fijan dos radios mínimos. La intersección de todos estos semiespacios fija la VRZ, donde estarán las bi-esferas que envuelven a los puntos de entrada. El volumen de las bi-esferas será mayor cuando más se alejen del origen (zonas claras) y menor cuanto estén más cerca del origen al tener radios menores (zonas oscuras). Para determinar aquella bi-esfera con mínimo volumen, se debe realizar una búsqueda en la zona del MVL desde el punto A al B.*

Para obtener la tri-esfera (n=2) con mínimo volumen que envuelve a un conjunto de puntos de entrada se tiene:

- La proyección de cada punto de entrada en el triángulo definido por c_0,c_1,c_2 se obtiene mediante el sistema de ecuaciones
$\begin{cases}\left(p_i^{\perp}-p_i\right)\cdot\left(c_1-c_0\right)=0\\ \left(p_i^{\perp}-p_i\right)\cdot\left(c_2-c_0\right)=0\end{cases}$ cuya resolución da λ_{i1}, λ_{i2}

- El parámetro d_i es $d_i=\left\|p_i^{\perp}-p_i\right\|$ con $p_i^{\perp}=c_0+\lambda_{i1}(c_1-c_0)+\lambda_{i2}(c_2-c_0)$

- Para aquellos $p_i=(\lambda_{i1},\lambda_{i2},d_i)$ con $\lambda_{i1}\leq 0$, $\lambda_{i2}\leq 0$ se calcula $d_0^{min}=\max\left(\left\|p_i-c_0\right\|\right)$

- Para aquellos $p_i=(\lambda_{i1},\lambda_{i2},d_i)$ con $\lambda_{i1}\geq 1$, $\lambda_{i1}\geq 1+\lambda_{i2}$ se calcula
$$d_1^{min}=\max\left(\left\|p_i-c_1\right\|\right)$$

- Para aquellos $p_i=(\lambda_{i1},\lambda_{i2},d_i)$ con $\lambda_{i2}\geq 1$, $\lambda_{i2}\geq 1+\lambda_{i1}$ se calcula
$$d_2^{min}=\max\left(\left\|p_i-c_2\right\|\right)$$

- Los HHP serán planos en el espacio $r_0\times r_1\times r_2$ definidos por las ecuaciones:
$$r_1=\left(1-\frac{1}{\lambda_{i1}}\right)r_0+\frac{d_i}{\lambda_{i1}}\quad y\quad r_2=\left(1-\frac{1}{\lambda_{i2}}\right)r_0+\frac{d_i}{\lambda_{i2}}$$

- El VRZ viene delimitado por las ecuaciones de todos los semiespacios:
$$r_1\geq\left(1-\frac{1}{\lambda_{i1}}\right)r_0+\frac{d_i}{\lambda_{i1}},\ r_2\geq\left(1-\frac{1}{\lambda_{i2}}\right)r_0+\frac{d_i}{\lambda_{i2}},\ r_0\geq d_0^{min},\ r_1\geq d_1^{min}\ y\ r_1\geq d_1^{min}$$

- El MVL son aquellos puntos del VRZ que pertenecen al menos a un plano HHP.

Esta técnica se ha implementado en un programa C para comprobar sus resultados. Por ejemplo, la Figura 3.2 muestra cuatro pasos para conseguir la bi-esfera con mínimo volumen envolvente de la herramienta de un sistema robotizado. De la herramienta, modelada en un sistema CAD, se han obtenido todos los puntos necesarios para definirla (112 puntos). Resulta necesario recordar que, por la propiedad de convexidad que cumple la bi-esfera (las poli-esferas en general), si una bi-esfera envuelve un conjunto de puntos, todo objeto compuesto de elementos convexos definidos con estos puntos está envuelto en la bi-esfera. Cuando el objeto no es convexo, como el caso de la pinza, siempre se puede encontrar una descomposición en elementos convexos. En este caso, la bi-esfera calculada envuelve completamente a todos los elementos convexos que forman la pinza.

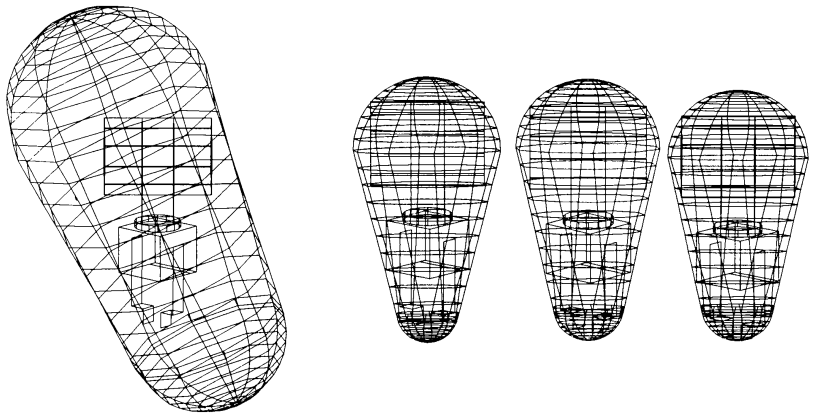


Figura 3.2. Ejemplo de Aplicación del Cálculo de la Bi-Esfera con Volumen Mínimo Envolvente de una Herramienta. *En los cuatro pasos mostrados en la gráfica se puede observar la evolución de la bi-esfera hasta alcanzar el mínimo volumen que engloba completamente a una herramienta tipo pinza de un sistema robotizado.*

3.1.2. Aplicación de Técnicas de Agrupamiento

La técnica mostrada en la sección anterior resuelve el problema de optimización descomponiéndolo en dos niveles: el nivel superior utiliza el método Downhill Simplex para mover los centros de las esferas vértices de la poli-esfera y así minimizar el volumen alcanzable por la poli-esfera para la que el nivel inferior calcula los radios. Los radios se obtienen mediante una búsqueda en el lugar generado mediante la aplicación de la Transformada de Hough.

La implementación de esta técnica ha mostrado los siguientes problemas:

- El método Downhill Simplex puede quedar atrapado en un mínimo local.
- Es un método muy sensible a los valores iniciales, tanto el resultado como el coste del proceso.
- Para un número de puntos de entrada alto, el tiempo de ejecución es muy elevado.

Respecto al tercer punto, el coste computacional es lineal al número de puntos de entrada, pero la ejecución de pruebas experimentales con puntos de entrada generados aleatoriamente produce los tiempos que se muestran en la Figura 3.3. Se puede concluir que los tiempos de ejecución, por ejemplo veinte minutos para calcular la bi-esfera de volumen mínimo que envuelve a 2500 puntos dados, lo hacen totalmente inoperante, no sólo para aplicaciones en tiempo real, sino incluso casi para aplicaciones off-line. Por ejemplo, calcular la bi-esfera envolvente mínima de un modelo tan sencillo como la herramienta de la Figura 3.2 tarda casi 25 segundos.

Estos tres problemas vienen directamente relacionados con el valor inicial de prueba que se seleccione, por lo que a continuación se va a proponer la aplicación de técnicas de agrupamiento para aportar un valor inicial suficientemente bueno (un modelo inicial que englobe los puntos de entrada con un volumen que no sea desproporcionado).

Las técnicas de *clasificación automática* o *agrupamiento* (*clustering*) [J&D88] están destinadas a producir agrupaciones de elementos descritos por un determinado número de características. Una vez realizadas las agrupaciones, los elementos de un mismo grupo tendrán características que los harán similares entre sí y que los diferenciarán de los elementos de los demás grupos. Estas técnicas son muy utilizadas en tratamiento estadístico de datos y suelen ser un paso previo a técnicas de análisis como la factorización o el análisis de varianza, o incluso pueden ser utilizadas como una técnica de análisis de datos.

La forma de agrupamiento óptima sería aquella en la que la suma de las "distancias entre los elementos" de un mismo grupo es mínima. El problema surge al tratar de definir el concepto "distancia entre elementos", al ser una propiedad que depende de la naturaleza de dichos elementos. Por lo general, dados unos valores numéricos para las características de los elementos, se suele partir de la distancia euclídea o de una función de ésta.

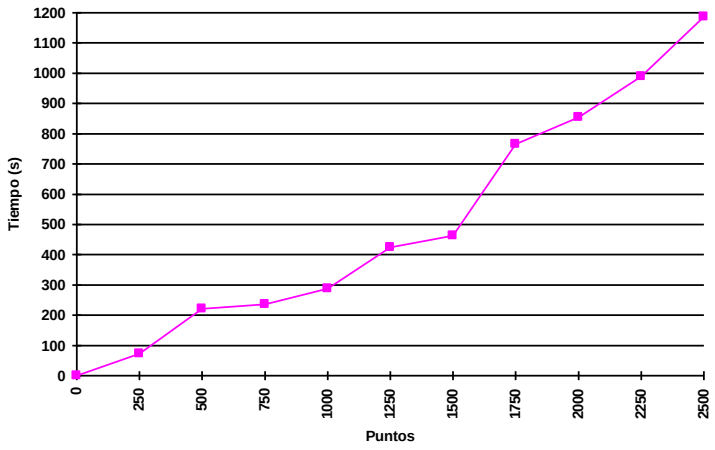


Figura 3.3. Tiempos de Ejecución del Cálculo de la Bi-Esfera Mínima. *Si bien el coste computacional es lineal al número de puntos de entrada, en la gráfica se puede observar que el tiempo medio de cálculo necesario para generar la bi-esfera de volumen mínimo que engloba a 2500 puntos de entrada se acerca a los veinte minutos. Para obtener esta gráfica se ha ejecutado un programa en C en un ordenador Pentium a 160MHz y se han obtenido los tiempos medios de 100 pruebas con puntos distribuidos aleatoriamente en el espacio.*

Para determinar el agrupamiento óptimo sería necesario desarrollar todas las agrupaciones posibles, calcular distancias entre elementos y coger aquel agrupamiento en el que la suma de distancias, dentro de un grupo, es mínima. Sin embargo, el número de agrupaciones posibles para un conjunto de elementos es muy elevado, incluso para conjuntos pequeños, y crece exponencialmente con el tamaño de éstos, por lo que no es aconsejable utilizar esta forma exhaustiva de búsqueda. Los métodos de agrupamiento no pretenden conseguir la solución óptima formalmente, sino que son métodos algorítmicos que pretenden buscar una de las mejores soluciones posibles, repartiendo los elementos de un conjunto de forma que la distancia entre los elementos que forman parte de un mismo grupo se minimice.

Antes de realizar un agrupamiento puede ser necesario realizar una normalización o reescalado. Esta necesidad puede surgir si la naturaleza de las variables o características que definen cada uno de los elementos es diferente, por lo que se puede primar alguna de las variables. El agrupamiento obtenido puede ser distinto según la escala que se asigne a cada variable. Existen dos familias de métodos algorítmicos para realizar los agrupamientos:

- *Métodos Ascendentes o Jerárquicos*: también son llamados *aglomerativos* porque se basan en la construcción de grupos por aglomeración sucesiva de objetos.

- *Métodos Descendentes o no Jerárquicos*: consisten en realizar un agrupamiento inicial e ir mejorando este agrupamiento sucesivamente.

Los métodos ascendentes consisten en agrupar los elementos de forma que en cada paso se agrupen aquellos elementos que estén mas cerca entre sí. El algoritmo termina cuando queda un número determinado de grupos o cuando la distancia entre los elementos a agrupar es menor que un valor determinado.

Como elementos también hay que considerar los grupos ya formados en pasos anteriores, con lo cual surge el problema de cómo calcular la distancia de un punto a un grupo o de un grupo a otro grupo, influyendo esto en el resultado final del algoritmo. Estas distancias entre grupos, considerando a un elemento como un grupo de un único elemento, se pueden considerar de distintas formas:

- La mínima distancia entre un elemento de cada grupo, si bien presenta el inconveniente de no distinguir entre grupos que están muy cercanos o entre grupos encadenados.
- La máxima distancia entre un elemento de cada grupo, si bien tiende a formar grupos del mismo diámetro y se ve afectado por valores atípicos.
- La distancia al grupo centroide o al grupo medio.
- Considerar la distancia entre dos grupos como la inversa de la densidad del grupo que formarían ambos, siendo esa densidad el número de puntos dividido por el volumen que ocupa todo el grupo. Esta distancia tiende a formar grupos compactos y resulta complicada de calcular.

El algoritmo del *salto mínimo* es uno de los métodos ascendentes más utilizados, teniendo los siguientes pasos:

1. *Obtener una matriz triangular superior de distancias entre todos los grupos (inicialmente elementos)*
2. *Determinar los dos grupos que tengan la mínima distancia dentro de la matriz*
3. *Agrupar los dos grupos en un grupo, reduciendo en uno la dimensión de la matriz, recalculando las distancias de ese nuevo grupo a todos los demás*
4. *Repetir los pasos 2-3 hasta que se alcance un número predefinido de grupos*

El algoritmo del salto mínimo parece adecuado para obtener un modelo aproximado de objetos mediante poli-esferas envolventes. Los elementos a agrupar serán puntos tridimensionales (el conjunto de vértices que definen al objeto). Agrupando estos vértices se puede simplificar el problema del cálculo de envolventes sugiriendo formas para la envolvente del modelo. El reescalado o la normalización de los elementos, solo cabría aplicarlos si por alguna razón se quiere ponderar alguna dimensión sobre las otras.

Los métodos ascendentes tienen una ventaja importante y es que permiten aplicar distintas formas de calcular las distancias entre los grupos, pudiéndose elegir aquella que más interese. El método que mejores resultados ofrece para generar modelos envolventes es el de la distancia al grupo centroide.

Para envolver un objeto mediante una poli-esfera se puede aplicar este método, agrupando los puntos hasta que se obtengan el número de grupos necesarios para el tipo de poli-esfera deseado: dos grupos para la bi-esfera, tres para la tri-esfera o n para una poli-esfera con n vértices esféricos. Posteriormente, para cada grupo de puntos generado calcula la mínima esfera envolvente, con lo que la poli-esfera queda formada. Sin embargo, el coste computacional de estas técnicas es exponencial, por lo que sólo deben aplicarse cuando el número de puntos de entrada es muy reducido.

Frente a los métodos de agrupamiento ascendente, los métodos de agrupamiento descendiente realizan una subdivisión del espacio en n grupos, e iterativamente mejorar el agrupamiento desplazando elementos de un grupo a otro, hasta que el agrupamiento converja en un agrupamiento estable.

El algoritmo de agrupamiento descendente más utilizado, llamado de las *nubes dinámicas*, es el siguiente:

1. *Se seleccionan inicialmente n puntos como centros iniciales de agrupamiento*
2. *Se asigna cada uno de los elementos al centro que esté más próximo*
3. *Se recalculan los centros de masa de los grupos formados*
4. *Repetir los pasos 2-3 hasta obtener un agrupamiento estable, es decir, que se repita el agrupamiento de la iteración anterior*

La elección de estos centros iniciales determinará la velocidad de convergencia e incluso el resultado final. Sin embargo, para la generación de envolventes mediante poli-esferas

resulta sencillo determinar unos centros iniciales adecuados. Por ejemplo, para encontrar una bi-esfera se pueden tomar como valores iniciales los dos puntos más alejados, para una tri-esfera el triángulo con mayor área, o para una tetra-esfera el tetraedro con mayor volumen.

Como se puede observar en la Figura 3.4, que muestra los tiempos medios de ejecución del algoritmo de las nubes dinámicas, el orden es lineal al número de puntos de entrada. Sin embargo, los tiempos de ejecución dependen de la distribución que tengan los puntos de entrada. En la gráfica de la derecha se puede observar el comportamiento temporal del programa para un agrupamiento con cuatro centros, distinguiéndose cuatro tipos de pruebas según los puntos formen grupos claramente diferenciables o grupos solapados. Se puede observar claramente que la pendiente de la gráfica es mucho mayor para puntos distribuidos en grupos totalmente solapados que para puntos previamente semi-distribuidos en grupos.

La primera utilidad de la técnica de agrupamiento de las nubes dinámicas es para determinar unos centros adecuados como valores iniciales del nivel superior (método de optimización Downhill Simplex) del proceso para calcular una poli-esfera de volumen mínimo envolvente de un conjunto de puntos de entrada. A partir de estos centros, el nivel inferior, mediante la utilización de la Transformada de Hough, calcula los radios que producen la poli-esfera con mínimo volumen, repitiéndose el proceso para nuevos centros recalculados por el nivel superior. Al tomar unos centros iniciales adecuados, el proceso, no sólo va a resultar más rápido, sino que además los resultados serán mejores, ya que se reduce enormemente la posibilidad de que el nivel superior quede atrapado en un mínimo local.

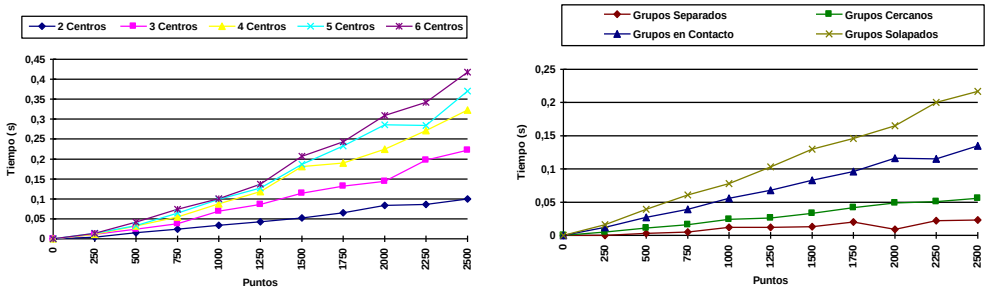


Figura 3.4. Coste del Algoritmo de Agrupamiento de las Nubes Dinámicas. *En la gráfica izquierda se muestra el tiempo medio de ejecución del algoritmo de las nubes dinámicas respecto al número de puntos de entrada para desde 2 a 10 centros, siendo lineal al número de puntos de entrada. El tiempo necesario para agrupar 1000 y 2500 puntos en dos grupos es de 56 y 157 milisegundos respectivamente. En la gráfica derecha se muestra la influencia que tiene, en los tiempos de cálculo del agrupamiento, la distribución en que se encuentren los puntos de entrada, para el caso concreto de agrupamiento a cuatro grupos. Las gráficas se han obtenido tomando los valores medios de 500 pruebas de un programa C ejecutado en un ordenador Pentium a 160MHz.*

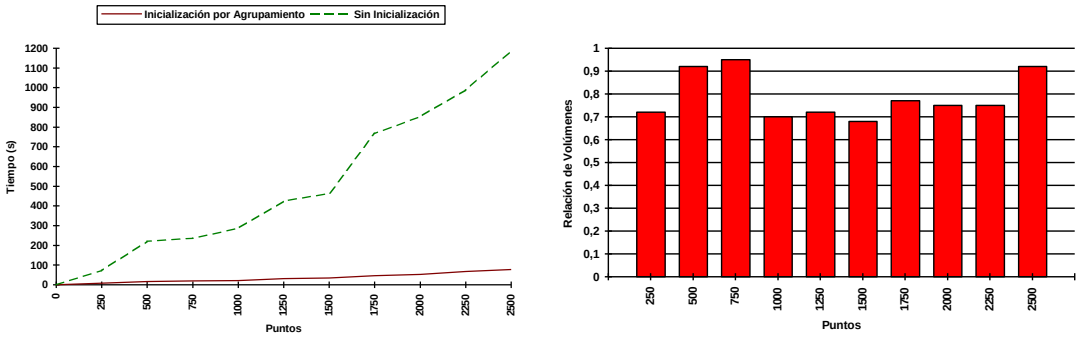
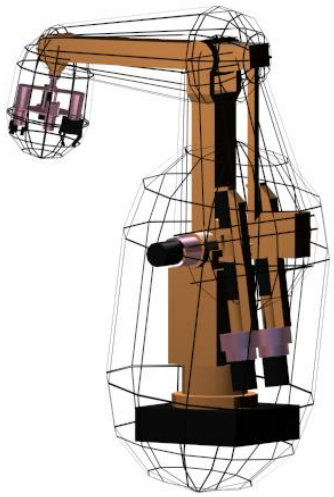


Figura 3.5. Mejoras Obtenidas al Aplicar Inicialización por Agrupamiento. *En la gráfica izquierda se muestra comparativamente el tiempo medio de ejecución del método para calcular una bi-esfera envolvente de puntos generados aleatoriamente cuando se utiliza inicialización por agrupamiento con el algoritmo de las nubes dinámicas frente a no utilizar esta inicialización. En la gráfica derecha se muestra la relación de volúmenes obtenidos (volumen obtenido con inicialización por agrupamiento dividido por el volumen obtenido sin inicialización por agrupamiento). Obsérvese que la mejora ronda el 75%, al evitarse mínimos locales en el proceso de optimización por Downhill Simplex. Estos resultados se obtienen como media de 100 ejecuciones con puntos aleatorios de un programa C en un ordenador Pentium a 160MHz.*

La Figura 3.5. muestra comparativamente las mejoras obtenidas al aplicar un proceso de inicialización por agrupamiento al método del cálculo de la bi-esfera de volumen mínimo que engloba unos puntos de entrada. En la gráfica izquierda se comparan los tiempos medios de ejecución del programa con y sin inicialización por agrupamiento. Como ejemplo significativo, calcular la bi-esfera de mínimo volumen envolvente de 2500 puntos baja de casi veinte minutos a poco más de 75 segundos. En la gráfica de la derecha se muestran las mejoras medias conseguidas en los volúmenes obtenidos, al rededor del 75%, al reducir la posibilidad de quedar atrapado en un mínimo local.

Para reducir estos tiempos, se pueden considerar únicamente una serie de puntos característicos, eliminando aquellos puntos que no van a influir en el modelo envolvente final. Formalmente, se debería considerar el mínimo número de puntos que define un politopo envolvente del robot real y obtener el modelo esférico envolvente de este politopo. En la práctica, no va a ser necesario realizar este proceso, sino que fácilmente se pueden descartar aquellos puntos que no influyen en el modelo envolvente, por ser vértices de zonas cóncavas del objeto.



Elemento	Puntos de Definición	Puntos Característicos	Tiempo Cálculo Bi-Esfera
Base y Cuerpo	505	72	0.72s
Brazo	118	28	0.28s
Antebrazo	99	52	0.52s
Muñeca y Mano	645	34	0.34s

Figura 3.6. Envolvente de un Brazo-Robot Industrial Mediante Bi-Esferas. *La aplicación del método para calcular la bi-esfera de mínimo volumen envolvente a los puntos característicos que definen cada elemento de este brazo articulado industrial (robot IRB L6 de ABB) permite obtener un modelo sencillo envolvente del mismo. En la tabla se puede observar el coste de realizar este modelado, dando un total de 1.86s modelar completamente todos los elementos del robot. Los tiempos de cálculo se obtienen de la ejecución de un programa C en un Pentium a 160MHz.*

Este método puede aplicarse ahora al cálculo de modelos envolventes de objetos conocidos a priori en un proceso off-line. Dentro de los objetos conocidos se puede considerar tanto el caso de elementos fijos del entorno, como de los elementos del mismo sistema robotizado. Por ejemplo, en la Figura 3.6 se muestra un modelo de un brazo-robot industrial mediante bi-esferas envolventes de sus elementos, junto con una tabla que muestra el tiempo de cálculo de cada elemento. Este modelado se realiza una vez para el robot y los modelos envolventes se asocian a los elementos reales del robot, de forma que un movimiento del robot a una configuración diferente implique sólo actualizar la posición y orientación de las envolventes.

Analizando los tiempos de cálculo (1.86s para modelar todos los elementos del robot), se deduce claramente que la obtención de un modelo envolvente con volumen mínimo se puede realizar solamente para un proceso off-line, pero nunca para una aplicación on-line. Este tipo de aplicaciones, con restricciones de tiempo real, pueden darse en dos casos típicos:

- La aplicación sobre un brazo-robot articulado para obtener un único modelo envolvente, que se actualice según la configuración nueva del robot. El modelo se debe ir regenerando según el robot se mueve, al cambiar de posición los puntos que definen al brazo-robot.
- Si se desea obtener la envolvente de un objeto desconocido del que el sistema de sensorización capta puntos de su superficie. El modelo debe ser modificado para envolver todos los puntos captados por los sensores en cada instante de tiempo, para que el sistema robotizado pueda aplicar con éxito un proceso de evitación de colisiones.

Estos casos requieren una velocidad de cálculo elevada, pese a que a la calidad del modelo envolvente no sea óptima. Este compromiso impide utilizar en aplicaciones on-line el proceso de minimización del volumen envolvente. En su lugar, si para cada grupo generado por el algoritmo de las nubes dinámicas se obtiene la esfera mínima que lo envuelve, se dispone de un modelo aproximado que sirve inicialmente, mientras que un proceso off-line no lo mejore.

El problema de determinar la mínima esfera envolvente de un conjunto de puntos es un problema que, con diferentes nombres (*spanning ball*, *minimum sphere*), ha sido tradicionalmente estudiado en el campo de la geometría computacional ([R&T57], [Mel85], [Meg89], [Sky91], [Wel91], [ESZ93]) y que también presenta un coste lineal al número de puntos. El tiempo de cálculo total para generar una primera aproximación envolvente de un objeto será la suma del coste de agrupamiento en un número de centros determinado y del coste de generación de las esferas mínimas que envuelven a cada grupo. La Figura 3.7 muestra la evolución de este coste respecto al número de puntos de entrada.

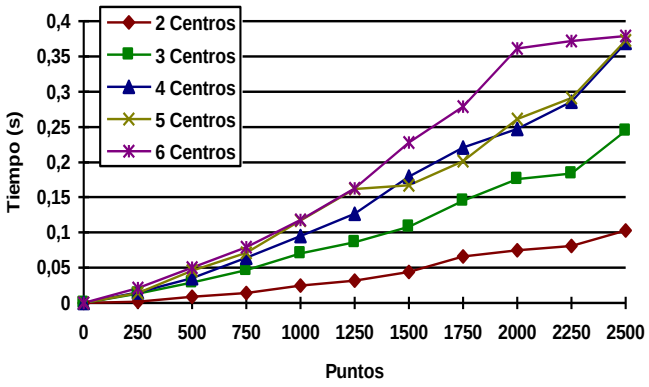
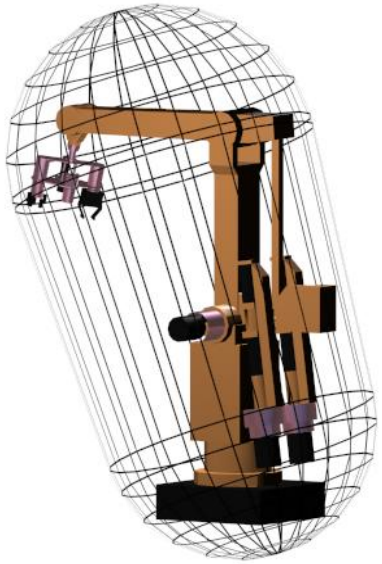


Figura 3.7. Tiempo de Ejecución del Cálculo de una Poli-Esfera Envolvente. *La obtención de una poli-esfera envolvente mediante la utilización del algoritmo de las nubes dinámicas más el cálculo de la esfera mínima envolvente de cada grupo generado tiene un coste lineal al número de puntos de entrada. Como ejemplo significativo, el tiempo medio necesario para generar una bi-esfera envolvente de 1500 puntos es de 82 milisegundos. La gráfica se ha obtenido tomando los valores medios de 500 pruebas de un programa C ejecutado en un ordenador Pentium a 160MHz.*

El estudio realizado hasta ahora se basa en considerar el caso peor, ya que al ser el coste lineal al número de puntos disponibles, los tiempos serán bajos para pocos puntos y crecerán según aumente el número de puntos de entrada. El estudio trata de reducir los tiempos cuando hay un número de puntos de entrada elevado, del orden de 2000 o 2500. Sin embargo, para el modelado de sistemas robotizados, el número de puntos característicos no va a resultar tan elevado. Por ejemplo, en la Figura 3.8 se muestra un brazo-robot completo (con 114 puntos característicos) modelado mediante una bi-esfera envolvente, junto con una tabla de tiempos de cálculo del modelo (el modelado se realiza en 3.3ms). A diferencia de los modelos obtenidos en la Figura 3.6, este modelado se debe recalcular cada vez que el robot se mueve a una configuración diferente.



Puntos de Definición	Puntos Característicos	Tiempo Cálculo de la Bi-Esfera Envolvente
1367	114	3.3ms

Figura 3.8. Envolvente de un Brazo-Robot Industrial Mediante una Bi-Esfera. *La aplicación del método para calcular la bi-esfera de envolvente a los puntos característicos que definen este brazo articulado industrial (robot IRB L6 de ABB) permite obtener un modelo sencillo envolvente del mismo. En la tabla se puede observar el coste de realizar este modelado, dando un total de 3.3ms para modelar completamente el robot. Los tiempos de cálculo se obtienen de la ejecución de un programa C en un Pentium a 160MHz.*

3.2. Simplificación de Poli-Esferas

Al aplicar el agrupamiento para obtener un modelo envolvente mediante poli-esferas, puede producirse grupos de puntos y por tanto esferas, que no se requieran para definir el volumen del modelo. Esto se produce principalmente cuando los puntos son captados por un sistema de sensorización, por lo que puntos erróneos pueden producir esferas dentro de los objetos. Estas esferas redundantes se deben eliminar para simplificar el modelo. Por otra parte, también pueden surgir esferas que sean combinación lineal de otras esferas, permitiendo también que se pueda reducir el modelo.

La simplificación de las poli-esferas se debe realizar en dos pasos [T&M94]: primero eliminar los vértices esféricos redundantes de las poli-esferas sobre-especificadas, obteniendo una poli-esfera válida. Entonces se determinan sus esferas básicas y su dimensión para obtener una poli-esfera bien-dimensionada, que al ser válida, es irreducible.

3.2.1. Eliminación de Vértices Esféricos Redundantes

El problema de determinar si una poli-esfera es sobre-especificada consiste en determinar para cada uno de sus vértices esféricos, si éste está contenido en el volumen generado por la poli-esfera formada por el resto de vértices esféricos. Si es cierto, el vértice esférico es redundante y se puede eliminar, manteniéndose el mismo volumen.

Una forma fácil de determinar si la esfera s_0 está contenida en el volumen generado por una poli-esfera $S_{0\text{-}n}$ con vértices esféricos $S=\{s_0,...,s_n\}$ consiste en los siguientes pasos:

- Considerar el conjunto de n esferas $S'=\{s_1-\mathbf{s}_0,...,s_n-\mathbf{s}_0\}$
- Si la poli-esfera $S'_{1\text{-}n}$ contiene al Origen, entonces $S_{0\text{-}n}$ contiene a s_0.

Ahora el problema se reduce a determinar si el origen de coordenadas está incluido en el volumen generado por una poli-esfera. Para ello, se calcula la distancia al origen de la poli-esfera y si esta distancia es positiva, el origen es exterior a la poli-esfera, mientras que si es negativa, está incluido.

3.2.2. Dimensión y Esferas Básicas de Poli-Esferas

Si se determina la base de una poli-esfera, los vértices esféricos no incluídos en ella serán combinación lineal de las esferas básicas. Planteando estas relaciones lineales en el espacio de los parámetros como nuevas condiciones del volumen generado, se pueden eliminar. El siguiente algoritmo permite determinar la dimensión y las esferas básicas de una poli-esfera.

Dado un conjunto de esferas $S=\{s_0,...,s_n\}$, construir las siguientes matrices, a partir de s_0, la esfera de mínimo radio:

$$[\Phi]_{4,n} = \left[\mathbf{s}_{01} \ \dots \ \mathbf{s}_{0n}\right]$$
$$[\Psi]_{3,n} = \left[\mathbf{v}_{01} \ \dots \ \mathbf{v}_{0n}\right]$$

Llámense φ=rango($[\Phi]$) y ς=rango($[\Psi]$). Nótese que $[\Psi]$ está formado por las tres primeras filas de $[\Phi]$ (los ejes de los vectores esféricos que forman $[\Phi]$) y por tanto, φ será igual a ς o a ς+1. Los valores de φ y ς se pueden utilizar para determinar la dimensión de la poli-esfera $S_{0\text{-}n}$. Ocho casos posibles se pueden presentar, listados en la siguiente tabla:

φ	ς	**CASO**
0	0	Una esfera
1	0	Bi-esfera degenerada (esferas concéntricas)
1	1	Una bi-esfera
2	1	Tri-esfera degenerada (bi-esferas con centros alineados)
2	2	Una tri-esfera
3	2	Tetra-esfera degenerada (Tri-esferas con planos base coplanares)
3	3	Una tetra-esfera
4	3	Una penta-esfera

Se pueden distinguir dos casos característicos:

a) $\varphi=\varsigma$. Para este caso, habrá un subconjunto de S con φ esferas, $\{\sigma_1,...,\sigma_\varphi\}$ cuyos vectores esféricos respecto a s_0, $\sigma_{01},...,\sigma_{0\varphi}$ son linealmente independientes. El resto de vectores esféricos son combinación lineal de ellos, es decir,

$$\mathbf{s}_{0i} = \alpha_{i1}\boldsymbol{\sigma}_{01} + \cdots + \alpha_{i\varphi}\boldsymbol{\sigma}_{0\varphi},\ i = 1,...,n$$

Entonces la dimensión de $S_{0\text{-}n}$ es φ y el conjunto de φ+1 esferas, $\{s_0,\sigma_1,...,\sigma_\varphi\}$ son las esferas básicas de la poli-esfera $S_{0\text{-}n}$. Como se muestra en la tabla, φ puede tomar cuatro valores diferentes:

- φ=0 implica la reducción a una esfera
- φ=1 quiere decir que se puede reducir a una bi-esfera
- φ=2 con reducción a una tri-esfera
- φ=3 reduciéndose a una tetra-esfera

b) $\varphi=\varsigma+1$. En este supuesto, habrá un subconjunto de S con ς esferas, $\{\sigma_1,...,\sigma_\varsigma\}$ para las que los centros de todas las esferas verificarán que

$$\mathbf{v}_{0i} = \alpha_{i1}\mathbf{v}_{01} + \cdots + \alpha_{i\varphi}\mathbf{v}_{0\varphi},\ i = 1,...,n$$

Pero sólo si una nueva esfera, σ_φ, se considera, la relación es cierta para las esferas completas (considerando los radios):

$$\mathbf{s}_{0i} = \alpha_{i1}\boldsymbol{\sigma}_{01} + \cdots + \alpha_{i\varsigma}\boldsymbol{\sigma}_{0\varsigma} + \alpha_{i\varphi}\boldsymbol{\sigma}_{0\varphi},\ i = 1,...,n$$

Entonces, la dimensión de $S_{0\text{-}n}$ es φ y un conjunto de $\varphi+1$ esferas, $\{s_0,\sigma_1,...,\sigma_\varsigma,\sigma_\varphi\}$ son las esferas básicas de la poli-esfera $S_{0\text{-}n}$. Sin embargo, uno de sus centros es combinación lineal de los otros. Los cuatro posibles valores de φ dan lugar a los siguientes casos:

- $\varphi=1$: reducible a una biesfera degenerada con esferas concéntricas (con el volumen igual al de la esfera de mayor radio)
- $\varphi=2$: reducible a una tri-esfera degenerada (bi-esferas con ejes alineados)
- $\varphi=3$: reducible a tetra-esfera degenerada (tri-esferas con planos base coplanares)
- $\varphi=4$: reducible a una penta-esfera

3.2.3. Reducción de Poli-Esferas

Las esferas básicas de la poliesfera $S_{0\text{-}n}$ se pueden determinar diagonalizando la matriz $[\Phi]$, lo que da una matriz triangular superior de la forma siguiente:

$$[\Phi] \triangleright \begin{bmatrix} I_{\varphi,\varphi} & \alpha_{\varphi,n-\varphi} \\ 0_{4-\varphi,\varphi} & 0_{4-\varphi,n-\varphi} \end{bmatrix}$$

donde φ es la dimension de la poli-esfera $S_{0\text{-}n}$; $[I]_{\varphi,\varphi}$ es la matriz identidad $\varphi\times\varphi$; $[\alpha]_{\varphi,n\text{-}\varphi}$ es una matriz $\varphi\times(n\text{-}\varphi)$ de coeficientes reales; $[0]_{4\text{-}\varphi,\varphi}$, $[0]_{4\text{-}\varphi,n\text{-}\varphi}$ son matrices nulas con sus correspondientes dimensiones.

Apartir de la matriz $[\alpha]_{\varphi,n\text{-}\varphi}$, con elementos

$$[\alpha] = \begin{bmatrix} \alpha_{1,\varphi+1} & \cdots & \alpha_{1,n} \\ \vdots & & \vdots \\ \alpha_{\varphi,\varphi+1} & \cdots & \alpha_{\varphi,n} \end{bmatrix}$$

se puede obtener la siguiente expresión para las $n\text{-}\varphi$ esferas que no son esferas básicas

$$\sigma_j = s_0 + \sum_{i=1}^{\varphi} \alpha_{i,j}\boldsymbol{\sigma}_{0i},\ j = \varphi+1,...,n$$

A continuación se presentan tres casos bastante habituales de reducción de poli-esferas: la reducción a una bi-esfera, a una tri-esfera y a una tetra-esfera.

1) Cuando φ=1, suponiendo $\sigma_1=\sigma_\varphi=s_1$, la poli-esfera $S_{0\text{-}n}$ se puede reducir a la bi-esfera S_{01}. La matriz obtenida después del proceso de diagonalización tiene la siguiente forma:

$$[\Phi] \triangleright \begin{bmatrix} 1 & \alpha_{1,2} & \cdots & \alpha_{1,n} \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

Y por tanto $s_i=s_0+\alpha_{1,i}\sigma_{01}$ i=2,...,n

Evidentemente, la bi-esfera generada por $\{s_0,s_1\}$ con fórmula $S_{01}=s_0+\lambda\sigma_{01}$, $\lambda\in\Lambda_1$ contiene al resto de esferas. Para este caso particular, $\Lambda_1=\{\lambda: \lambda_{min}\leq\lambda\leq\lambda_{max}\}$. El rango Λ_1 del parámetro λ se puede calcular como sigue

$$\lambda_{min} = \alpha_{1,j} = \min \alpha_{1,i} \quad i = 0,\ldots,n$$
$$\lambda_{max} = \alpha_{1,k} = \max \alpha_{1,i} \quad i = 0,\ldots,n$$

con $\alpha_{1,0}$=0 y $\alpha_{1,1}$=1.

Si se requiere una formulación normalizada en el rango del parámetro, la bi-esfera se puede definir a partir de las dos esferas extremos s_i, s_j, obtenidas en el proceso definido por las ecuaciones para obtener λ_1 y λ_2, de la forma siguiente

$$S_{ij} = s_i + \lambda \mathbf{s}_{ij} \quad 0 \leq \lambda \leq 1$$

2) Cuando φ=2, suponiendo $\sigma_1=s_1$ y $\sigma_2=\sigma_\varphi=s_2$, la poli-esfera $S_{0\text{-}n}$ se puede reducir a una tri-esfera S_{012}. La fórmula de la tri-esfera será:

$$S_{012} = s_0 + \lambda_1\boldsymbol{\sigma}_{01} + \lambda_2\boldsymbol{\sigma}_{02},\ \lambda_1,\lambda_2 \in \Lambda_{12}$$

La matriz obtenida tras la diagonalización será:

$$[\Phi] \triangleright \begin{bmatrix} 1 & 0 & \alpha_{1,3} & \cdots & \alpha_{1,n} \\ 0 & 1 & \alpha_{2,3} & \cdots & \alpha_{2,n} \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

Y por tanto, todas las esferas se pueden expresar como combinación lineal de las tres esferas básicas:

$$s_i = s_0 + \alpha_{1,i}\boldsymbol{\sigma}_{01} + \alpha_{2,i}\boldsymbol{\sigma}_{02},\ i = 3,\ldots,n$$

El subespacio $\Lambda_{12} \subset \Re^2$ está limitado por rectas que unen los vértices esféricos en el espacio paramétrico. El semiespacio definido por la arista esférica que une s_i con s_j formulada como el semiespacio limitado por una recta en términos de λ_1, λ_2 se puede calcular a partir de la siguiente ecuación (tomando $\alpha_{1,0}=\alpha_{2,0}=0$):

$$\begin{vmatrix} \lambda_1 & \alpha_{1,i} & \alpha_{1,j} \\ \lambda_2 & \alpha_{2,i} & \alpha_{2,j} \\ 1 & 1 & 1 \end{vmatrix} \geq 0$$

Siguiendo este procedimiento, se calculan todas las rectas en Λ_{12} que corresponden a aristas esféricas de la poliesfera a partir de sus vértices esféricos.

3) Cuando $\varphi=3$, suponiendo $\sigma_1=s_1$, $\sigma_2=s_2$ y $\sigma_3=\sigma_\varphi=s_3$, la poli-esfera $S_{0\text{-}n}$ se puede reducir a la tetra-esfera S_{0123}. La fórmula de la tetra-esfera será:

$$S_{0123} = s_0 + \lambda_1 \boldsymbol{\sigma}_{01} + \lambda_2 \boldsymbol{\sigma}_{02} + \lambda_3 \boldsymbol{\sigma}_{03},\ \lambda_1, \lambda_2, \lambda_3 \in \Lambda_{123}$$

La matriz obtenida tras la diagonalización será:

$$[\Phi] \triangleright \begin{bmatrix} 1 & 0 & 0 & \alpha_{1,4} & \cdots & \alpha_{1,n} \\ 0 & 1 & 0 & \alpha_{2,4} & \cdots & \alpha_{2,n} \\ 0 & 0 & 1 & \alpha_{3,4} & \cdots & \alpha_{3,n} \\ 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

Y por tanto, todas las esferas se pueden expresar como combinación lineal de las cuatro esferas básicas:

$$s_i = s_0 + \alpha_{1,i} \boldsymbol{\sigma}_{01} + \alpha_{2,i} \boldsymbol{\sigma}_{02} + \alpha_{3,i} \boldsymbol{\sigma}_{03},\ i = 4,\ldots,n$$

El subespacio $\Lambda_{123} \subset \Re^3$ está limitado por planos que unen los vértices esféricos en el espacio paramétrico. El semiespacio definido por el plano esférico que une s_i, s_j y s_k formulado como el semiespacio limitado por un plano en términos de λ_1, λ_2, λ_3 se puede calcular a partir de la siguiente ecuación (tomano $\alpha_{1,0}=\alpha_{2,0}=\alpha_{3,0}=0$):

$$\begin{vmatrix} \lambda_1 & (\alpha_{1,j} - \alpha_{1,i}) & (\alpha_{1,k} - \alpha_{1,i}) \\ \lambda_2 & (\alpha_{2,j} - \alpha_{2,i}) & (\alpha_{2,k} - \alpha_{2,i}) \\ \lambda_3 & (\alpha_{3,j} - \alpha_{3,i}) & (\alpha_{3,k} - \alpha_{3,i}) \end{vmatrix} \geq \begin{vmatrix} \alpha_{1,i} & \alpha_{1,j} & \alpha_{1,k} \\ \alpha_{2,i} & \alpha_{2,j} & \alpha_{2,k} \\ \alpha_{3,i} & \alpha_{3,j} & \alpha_{3,k} \end{vmatrix}$$

Siguiendo este procedimiento, se pueden calcular todos lo planos en Λ_{123} que corresponden a planos esféricos de la poliesfera a partir de sus vértices esféricos.

Es importante remarcar que al aplicar diagonalización, suele resultar necesario realizar intercamnbios de filas o columnas en la matriz. Estos cambios representan lo siguiente:

- Si se intercambian las columnas i, j, las esferas s_i, s_j se deben intercambiar. Al final del proceso, el conjunto de esferas $\{s_0,...,s_n\}$ que generaba la poli-esfera $S_{0\text{-}n}$ ha sufrido un proceso de reordenación.
- Si se intercambian las filas i, j, las coordenadas i, j se deben "intercambiar", teniendo en cuenta que las coordenadas x, y, z y r son respectivamente 1, 2, 3 y 4. Este intercambio se debe realizar sólo lógicamente, considerando una matriz de permutaciones [P] que afecte a la matriz $[\alpha]$ en las ecuaciones anteriores. Cuando la coordenada r haya sufrido algún intercambio, se presentará alguno de los casos degenerados explicados anteriormente ($\varphi=\varsigma+1$).

Por tanto, una poli-esfera $S_{0\text{-}n}$ con vértices esféricos $\mathbf{S}=\{s_0,...,s_n\}$ definida según:

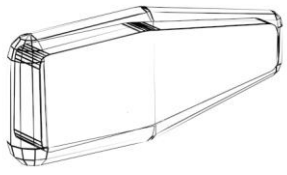
$$S_{0-n}=\left\{s\in\Omega:s=s_0+\sum_{i=1}^{n}\lambda_i\mathbf{s}_{0i},\mathbf{s}_{0i}=s_i-s_0,s_i\in\mathbf{S},0\le\lambda_i,i=1,\ldots,n,\sum_{i=1}^{n}\lambda_i\le 1\right\}$$

se puede reducir, ocupando el mismo volumen, a una poli-esfera $S'_{0\text{-}\varphi}$ con vértices esféricos $\mathbf{S}'=\{s'_0,...,s'_\varphi\}$, subconjunto de $\mathbf{S}$, con φ=rango($[\Phi]$)<4 definida según:

$$S'_{0-\varphi}=\left\{s\in\Omega:s=s'_0+\sum_{i=1}^{\varphi}\lambda_i\mathbf{s}'_{0i},\mathbf{s}'_{0i}=s'_i-s'_0,s'_i\in\mathbf{S}',\sum_{i=1}^{\varphi}\alpha_{ij}\lambda_i\ge\beta_j,j=1,\ldots,n\right\}$$

Esta nueva poli-esfera irreducible sólo requiere φ parámetros λ_i para relacionar sus φ+1 esferas, frente a los n parámetros λ_i necesarios anteriormente para relacionar sus n+1 esferas. Además, el número de ecuaciones lineales que delimitan los rangos de los parámetros λ_i se ha reducido de n+1 a n.

Como ejemplo de aplicación, considérese el segundo elemento de un brazo robot PUMA estándar como el mostrado en la Figura 3.9, del cual se ha generado como modelo envolvente una poli-esfera con los vértices esféricos que se muestran en la tabla de la figura.



	x	y	z	r
s_0	-57.50	-58.50	0.00	20.00
s_1	20.00	-58.50	0.00	16.00
s_2	-57.50	58.50	0.00	20.00
s_3	20.00	58.50	0.00	16.00
s_4	155.00	-17.55	0.00	8.00
s_5	155.00	17.55	0.00	8.00

Figura 3.9. Envolvente del Segundo Elemento de un Robot PUMA. *La envolvente generada por agrupamiento está definida mediante los 6 vértices esféricos mostrados en la tabla de la derecha.*

Su formulación como una poli-esfera de cinco dimensiones, $S_{0\text{-}5}$ es:

$$S_{0-5} = s_0 + \sum_{i=1}^{5} \lambda_i \mathbf{s}_{0i},\ 0 \le \lambda_i,\ \sum_{i=1}^{5} \lambda_i \le 1$$

Aplicando la diagonalización a la matriz $[\Phi]$ se tiene:

$$[\Phi] = \begin{bmatrix} 77.5 & 0 & 77.5 & 155 & 155 \\ 0 & 117 & 117 & 40.95 & 76.05 \\ 0 & 0 & 0 & 0 & 0 \\ -4 & 0 & -4 & -12 & -12 \end{bmatrix} \triangleright \begin{bmatrix} 1 & 0 & 1 & 2 & 2 \\ 0 & 1 & 1 & 0.35 & 0.65 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Como el rango es dos ($\varphi=\varsigma=2$), existe una tri-esfera que engloba al volumen esférico generado por $S_{0\text{-}5}$. Considérese la siguiente tri-esfera, $S_{012}=s_0+\lambda_1\mathbf{s}_{01}+\lambda_2\mathbf{s}_{01}$, donde λ_1, λ_2 van a estar limitados por las ecuaciones de las rectas que unen las siguientes esferas:

- La arista esférica que une s_0 con s_1: $\begin{vmatrix} \lambda_1 & 0 & 1 \\ \lambda_2 & 0 & 0 \\ 1 & 1 & 1 \end{vmatrix} \ge 0$ que da la condición $\lambda_2 \ge 0$
- La arista esférica que une s_1 con s_4 $\begin{vmatrix} \lambda_1 & 1 & 2 \\ \lambda_2 & 0 & 0.35 \\ 1 & 1 & 1 \end{vmatrix} \ge 0$ da la condición $0.35\lambda_1 - \lambda_2 \le 0.35$
- La arista esférica que une s_4 con s_5 $\begin{vmatrix} \lambda_1 & 2 & 2 \\ \lambda_2 & 0.35 & 0.65 \\ 1 & 1 & 1 \end{vmatrix} \ge 0$ da la condición $\lambda_1 \le 2$
- La arista esférica que une s_5 con s_3 $\begin{vmatrix} \lambda_1 & 2 & 1 \\ \lambda_2 & 0.65 & 1 \\ 1 & 1 & 1 \end{vmatrix} \ge 0$ da la condición $0.35\lambda_1 + \lambda_2 \le 1.35$
- La arista esférica que une s_3 con s_2 $\begin{vmatrix} \lambda_1 & 1 & 0 \\ \lambda_2 & 1 & 1 \\ 1 & 1 & 1 \end{vmatrix} \ge 0$ da la condición $\lambda_2 \le 1$

- La arista esférica que une s_2 con s_0 $\begin{vmatrix} \lambda_1 & 0 & 0 \\ \lambda_2 & 1 & 0 \\ 1 & 1 & 1 \end{vmatrix} \geq 0$ da la condición $\lambda_1 \geq 0$

Tras aplicar el método de reducción propuesto, se obtiene la siguiente tri-esfera:

$$S_{012} = s_0 + \lambda_1 \mathbf{s}_{01} + \lambda_2 \mathbf{s}_{02}$$

con

$$\lambda_1, \lambda_2 \in \Lambda_{123} = \{0 \leq \lambda_1 \leq 2,\ 0 \leq \lambda_2 \leq 1,\ 0.35\lambda_1 - \lambda_2 \leq 0.35,\ 0.35\lambda_1 + \lambda_2 \leq 1.35\}$$

que presenta el mismo volumen que $S_{0\text{-}5}$.

Aplicando el método de reducción desarrollado, el mismo volumen se representa en el ejemplo anterior, utilizando sólo 3 esferas con 2 parámetros y 6 condiciones, en vez de 6 esferas con 5 parámetros y 6 condiciones. En general, para n+1 esferas, su reducción a tri-esfera representa eliminar n-2 esferas y n-2 parámetros, manteniendo el mismo número de condiciones de los parámetros.

3.3. Generación de Esferoides Envolventes

La segunda posibilidad de modelar un objeto, a partir de las esferas de control envolventes de los conjuntos de puntos obtenidos por agrupación, es mediante esferoides. Este tipo de modelado va a resultar principalmente adecuado para brazos-robot articulados, por la forma *"zigzagueante"* que pueden adoptar. El número de esferas de control envolvente de los conjuntos generados en la fase de agrupamiento y el tipo de esferoide que se utiliza como envolvente del sistema robotizado vienen directamente relacionados según:

- Para modelar el robot mediante una esferoide cuadrática se necesitan tres esferas de control
- Para modelar el robot mediante una esferoide cúbica se necesitan cuatro esferas de control
- Para modelar el robot mediante una esferoide spline se necesitan cinco o más esferas de control

La esferoide obtenida dependerá de los grupos obtenidos en el proceso de agrupamiento, por lo que la elección de los puntos iniciales para realizar el agrupamiento va a ser decisiva. Para que el modelo sea adecuado, las esferas de control deben situarse en las proximidades de las articulaciones. Teniendo en cuenta que sobre los ejes de estas

articulaciones se colocan los sistemas de coordenadas locales de los elementos, los puntos orígenes de estos sistemas resultan adecuados para iniciar las agrupaciones. Además, durante los movimientos del robot, se conoce dónde se encuentran estos puntos.

Por tanto, para obtener una esferoide envolvente de un brazo-robot en una configuración dada, se realiza un agrupamiento partiendo de los puntos característicos situados en los orígenes de los sistemas de coordenadas locales de los elementos. Alrededor de estos puntos, el algoritmo de las nubes dinámicas generará grupos de puntos sobre los que posteriormente se generan las esferas de volumen mínimo que los envuelven. Las esferas generadas se encontrarán situadas siempre a lo largo del robot, siendo su situación simétrica cuando el robot lo sea. Con estas esferas se pueden obtener fácilmente las esferoides.

El coste de obtener una esferoide va a ser similar al de realizar un agrupamiento y calcular la esfera mínima de cada grupo, ya que obtener la esferoide a partir de estas esferas es un proceso cuyo tiempo de cálculo no es significativo frente al tiempo anterior (centésimas de milisegundo). Pero por la idiosincrasia del problema, los puntos característicos que definen un brazo-robot se encuentran semiagrupados inicialmente respecto a los ejes de articulación, por lo que los tiempos de cálculo son muy bajos. La Figura 3.10 muestra la evolución de estos tiempos según el número de puntos de entrada para las esferoides cuadráticas, cúbicas y splines de cinco y seis esferas de control.

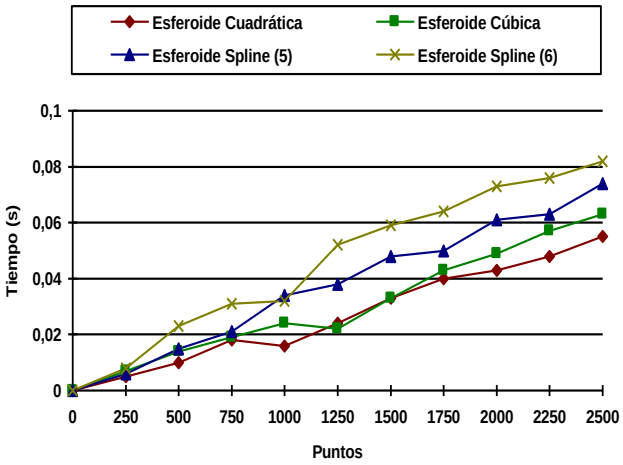


Figura 3.10. Tiempo de Ejecución del Cálculo de una Esferoide Envolvente. *La obtención de una esferoide envolvente mediante la utilización del algoritmo de las nubes dinámicas más el cálculo de la esfera mínima envolvente de cada grupo generado tiene un coste lineal al número de puntos de entrada. El tiempo medio necesario para generar una esferoide cuadrática, cúbica o spline de cinco o seis esferas de control envolvente de 2500 puntos es de 55, 63, 74 y 82 milisegundos respectivamente. La gráfica se ha obtenido tomando los valores medios de 500 pruebas de un programa C ejecutado en un ordenador Pentium a 160MHz con puntos de entrada semidistribuidos aleatoriamente alrededor de tantos centros aleatorios como esferas de control se deban conseguir.*

La utilización de modelos esféricos envolventes tiene como aplicación el modelar un brazo-robot completo, de forma que el modelo se debe recalcular cuando se mueva el robot. Aunque los tiempos pueden considerarse como suficientemente rápidos (menos de una centésima de segundo cuando se dispone de hasta 250 puntos característicos) el proceso de generación se puede acelerar mediante alguna de las dos aproximaciones siguientes:

- Generar las esferas de control una vez para una configuración inicial y aplicar la cinemática directa sobre ellas cuando el robot se mueva, generando la nueva esferoide envolvente. Como los puntos característicos varían de posición al moverse el robot, para evitar que algunos puntos se salgan de las esferas de control, se puede aumentar el radio de las mismas. Este problema no aparece si las esferas de control se generan para el peor caso posible, aquel que produce radios de esferas mayores, pero produce que en los casos mejores las esferas estén sobredimensionadas.
- El movimiento del robot producirá cambios en la posición de los puntos característicos pero empíricamente se ha comprobado que estos cambios no altera el grupo al que pertenecen. Por tanto, si ante un movimiento del robot los grupo siguen manteniendo los mismos puntos característicos, bastaría con aplicar la cinemática directa sobre estos puntos y realizar únicamente el cálculo de la esfera mínima que envuelve a este grupo.

El principal problema que plantean las esferoides es que no son objetos convexos, por lo que, pese a estar generados por esferas que envuelven a todos los puntos característicos de un robot, no se garantiza que envuelvan completamente al mismo. Empíricamente se ha comprobado que el caso peor se produce cuando dos elementos unidos por una articulación presentan un giro de 90º. Este problema se puede resolver de dos formas:

- Aumentando el radio de la esfera de control que se encuentra en este eje de articulación. El incremento del radio será mayor cuando la variable de articulación más se acerque a 90º. Sin embargo, el incremento puede llegar a ser grande (20% del radio). Es la solución a utilizar para mantener modelos del tipo esferoide cuadrática o cúbica.
- Introduciendo nuevas esferas de control en zonas intermedias de los elementos, de forma que se fuerza a que el modelo se aproxime más al robot. Eso lleva a que los modelos a utilizar sean esferoides splines con varias esferas de control. Adicionalmente se pueden aumentar los radios de las esferas por seguridad.

Las Figuras 3.11 a y b muestran como se han modelado dos brazos-robot industriales con esferoides cuadrática, cúbica y spline de cinco y seis esferas de control. Para los casos cuadrático y cúbico resultó necesario aplicar una corrección a los radios de las esferas de control. En ambos casos, la esferoide cúbica es la que peor modelo presenta, debido a la mala distribución que presentan las esferas de control a lo largo del robot (las dos últimas esferas de control se encuentran muy próximas). Los tiempos de cálculo de estos modelos son muy bajos, tardando 4ms para el modelo más completo. Para el modelado sólo se han considerado la parte articulada de los robots, es decir, sin considerar la plataforma, la base y el cuerpo, siguiendo la filosofía de modelado jerárquico que se presenta en la siguiente sección.

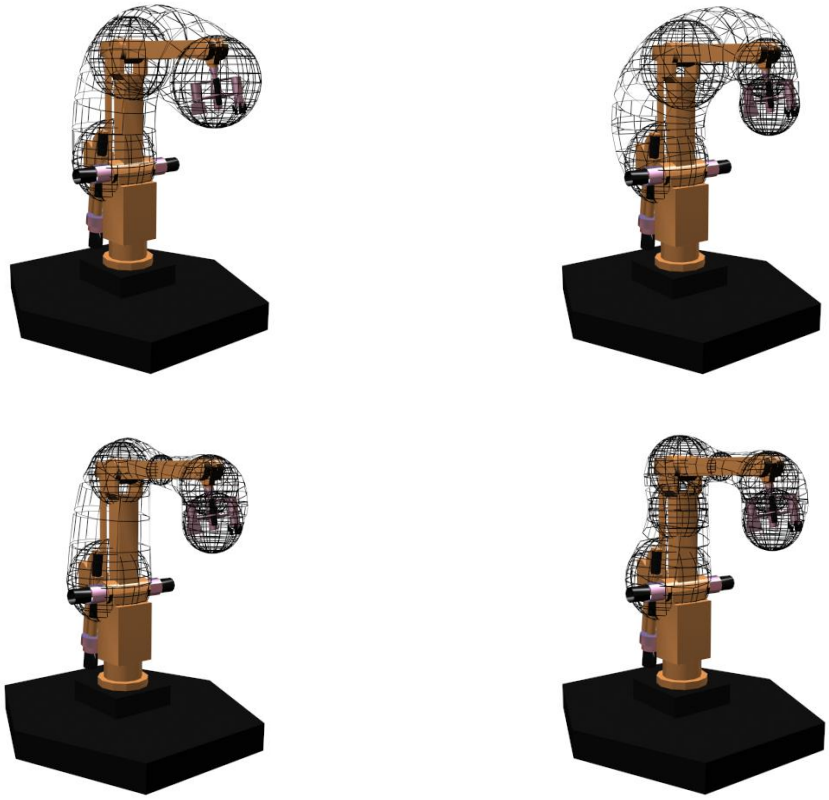


Figura 3.11a. Modelado con Esferoides de un Brazo-Robot Industrial. *En esta gráfica se muestran los modelos con esferoides cuadrática, cúbica y spline con cinco y seis esferas de control del robot IRB L6 de ABB. Sólo para los dos primeros casos se ha requerido aplicar un factor de corrección a los radios de las esferas de control (incremento del 5% y 20% del radio para la esfera de control situada en el codo en los casos cuadrático y cúbico respectivamente). En las gráficas, con las esferas de control solapadas a la esferoide generada, puede observarse que el caso cúbico es el que menos se ajusta a la forma de este robot. Los modelos se han obtenido a partir de 124 puntos característicos del robot con un programa C ejecutado en un ordenador Pentium a 160MHz dando unos tiempo de ejecución de 2.5ms, 3ms, 3.5ms y 4ms para los casos cuadrático, cúbico y spline de cinco y seis esferas de control respectivamente.*

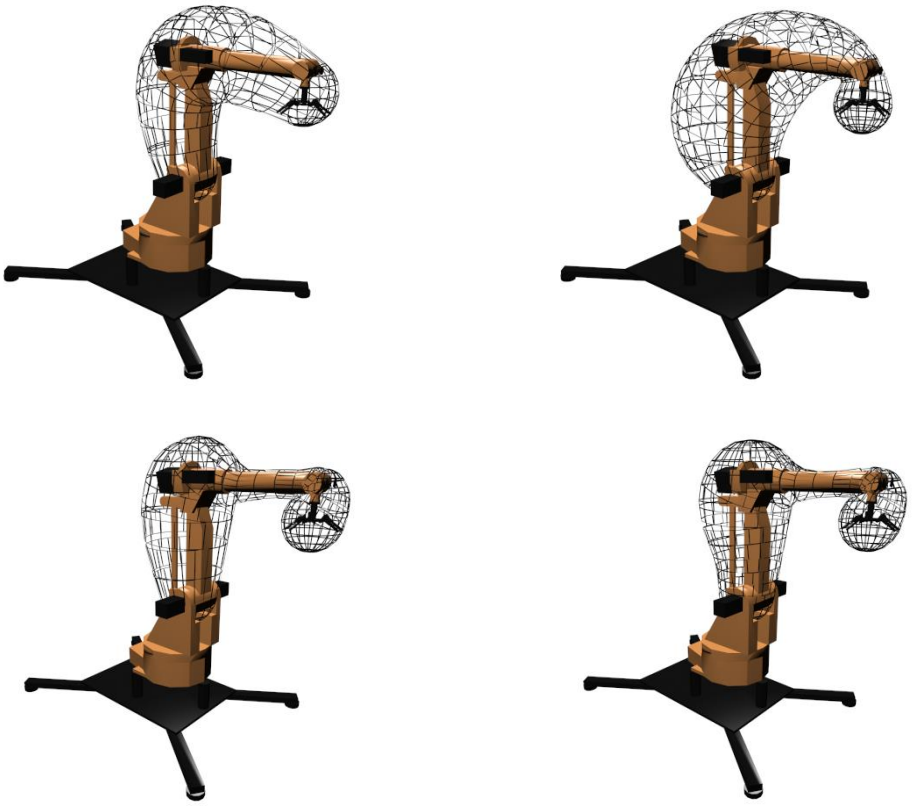


Figura 3.11b. Modelado con Esferoides de un Brazo-Robot Industrial. *En esta gráfica se muestran los modelos con esferoides cuadrática, cúbica y spline con cinco y seis esferas de control del robot IRB1500 de ABB. Sólo para los dos primeros casos se ha requerido aplicar un factor de corrección a los radios de las esferas de control (incremento del 5% del radio para la esfera de control situada en el codo en el caso cuadrático y del 20% del radio para la esfera de control situada en el codo y del 5% para la esfera de control situada en la muñeca para el caso cúbico). De las gráficas se puede deducir que el caso cúbico es de nuevo el que menos se ajusta a la forma de este robot. Los modelos se han obtenido a partir de 82 puntos característicos del robot con un programa C ejecutado en un ordenador Pentium a 160MHz dando unos tiempo de ejecución de 1.5ms, 2ms, 2.25ms y 2.5ms para los casos cuadrático, cúbico y spline de cinco y seis esferas de control respectivamente.*

3.4. Modelado Jerárquico de Sistemas Robotizados

El modelado de un sistema robotizado y su entorno implica representar los distintos objetos que aparecen en dicho sistema con distintos modelos, los cuales tendrán una mayor o menor precisión. Esto introduce el concepto de modelo jerárquico o árbol de ensamblaje [Mye81], [Fav86], [Qui94]. Además, debido a la naturaleza propia de los sistemas robotizados, para describirlos resulta conveniente utilizar una estructura jerárquica. En esta estructura se definen diferentes niveles: células, sistemas, sub-sistemas, elementos, ...

Las células son el nivel superior de la estructura, incluyendo robots con ejes externos, líneas de ensamblado, conjuntos robot/máquina-herramienta, etcétera, correspondiendo los siguientes niveles a brazos-robot, vehículos autoguiados, máquinas-herramienta, cintas transportadoras para el nivel sistema y finalmente los elementos del brazo-robot, componentes de máquinas-herramientas y vehículos autoguiados, etcétera para el nivel elemento. A partir de este nivel, pese a que los elementos son sistemas rígidos, se pueden definir otros niveles: los elementos se descomponen en bloques, éstos se forman con primitivas y éstas se definen (o se pueden aproximar) mediante caras planas u otras formas geométricas de representación.

Las células y los sistemas deben tener un modelo cinemático, basado en matrices de transformación homogénea y en los parámetros de Denavit-Hartenberg [D&H55], que se utiliza para determinar la posición y orientación de sus elementos. Para ello, se describen puntos característicos de cada elemento respecto a unos sistemas de referencia locales a los elementos. Estos sistemas de referencia, basados en la representación de Denavit-Hartenberg, se sitúan sobre los ejes de articulación que unen dos elementos. Las matrices anteriores son las que relacionan la posición y orientación de cada sistema de coordenadas local con el sistema previo, en función de la variable de esa articulación [Cra88].

Un sistema podrá ser modelado, por ejemplo, mediante una simple esfera que lo envuelva totalmente, pero parece claro que en determinadas aplicaciones este modelo resulta muy inexacto, por lo que se debe aumentar la calidad del modelo, representando el sistema mediante una bi-esfera. Evidentemente, la precisión se ha incrementado, pero, si se requiere mayor fidelidad en el modelo, hay que aumentar más la complejidad del volumen envolvente, utilizando esferoides o bien poli-esferas con más vértices esféricos. Por contra, utilizar envolventes más complejas cada vez, produce que los tiempos de procesamiento de las aplicaciones se incremente considerablemente. Por lo tanto, los volúmenes que sean excesivamente complejos pueden llegar a ser, en general, contraproducentes.

Sin embargo, podemos aumentar la calidad de la aproximación, modelando independientemente cada uno de los elementos que componen el sistema en cuestión, y además, también de un modo jerárquico. Así cada elemento puede ser envuelto a través de un esfera, una bi-esfera, una tri-esfera, etc.

En este modelado jerárquico, se contemplan todas las posibilidades anteriormente expuestas, es decir, modelado en los niveles célula, sistema, elemento, bloque y primitiva. La estructura jerárquica permite modelar varios objetos independientemente en un cierto nivel, o como una única entidad en el nivel superior. Por ejemplo, los elementos de un brazo-robot se pueden modelar independientemente en el nivel elemento o globalmente en el nivel sistema. Aunque es difícil de conseguir, siempre que sea posible, el volumen del modelo en un cierto nivel debe cotener al volumen de la unión de modelos usados en el nivel inferior. Para los objetos del nivel elemento, los volúmenes dependerán de la forma y las dimensiones de los objetos; para niveles superiores, además habrá que tener en cuenta la configuración en cada instante, obtenida mediante el modelo cinemático.

La topología de la estructura jerárquica se puede almacenar en un árbol cuya raíz será el nivel más alto, normalmente la célula, mientras que las hojas serán todas las primitivas que la forman. Los nodos intermedios representarán diferentes niveles de descomposición (sistemas, elementos, bloques, ...). La descomposición de un nodo en sub-nodos no tiene por que ser única, pudiendo haber diferentes formas de descomonerlo en paralelo. La aplicación de detección de colisiones que utilice este árbol será la encargada de seleccionar qué subdivisión escoger dinámicamente, no sólo según la configuración actual del nodo como hace [HC+92], sino considerando además otros nodos del mismo árbol o de otros árboles que sean los obstáculos. Para obtener un mayor rendimiento, se consideran niveles intermedios entre los niveles sistema y elemento, para permitir la agrupación de diferentes elementos en un sub-sistema.

Respecto al modelo geométrico con que se modela cada nodo, tradicionalmente se han utilizado los modelos poliédricos [Pas89], [Cam89], [Hub93]. Una estructura jerárquica extendida [TM+92], con diferentes grados de precisión, se puede formar utilizando poli-esferas (que incluyen a los modelos poliédricos) y esferoides. Esta estructura se puede generar automáticamente a partir de los modelos exactos del sistema robotizado, disponibles en la base de datos de un sistema CAD, mediante las técnicas presentadas para el cálculo de volúmenes esféricos envolventes. Cada modelo en un nivel dado se obtiene calculando el volumen del tipo considerado que envuelva todas las partes de este nivel. Un ejemplo de los modelos para un elemento herramienta con tres pinzas se muestra en la Figura 3.12.

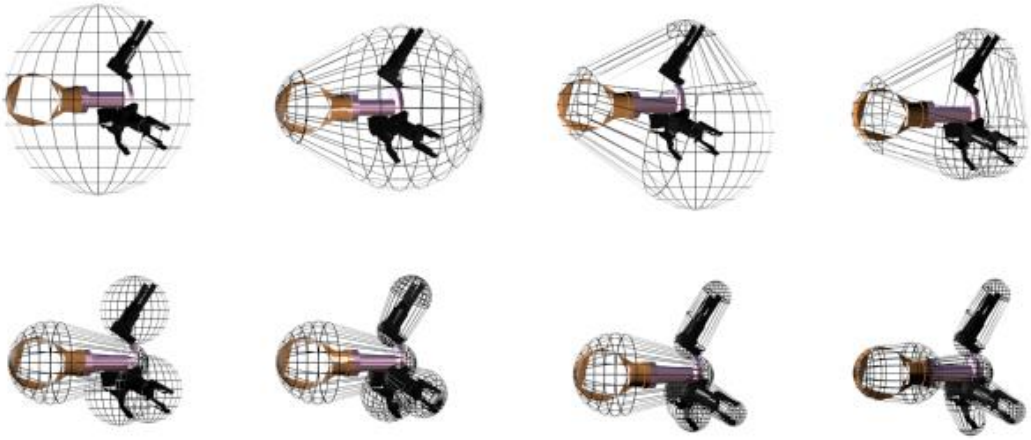


Figura 3.12. Diferentes Grados de Precisión de Modelos Envolventes de un Elemento. *En la gráfica se muestra un herramienta con tres pinzas modelada en dos niveles: para el nivel superior se considera toda la herramienta con los grados de precisión esfera, bi-esfera, tri-esfera y tetra-esfera; para el nivel inferior, se muestran una descomposición con una pieza de sujeción (modelada con una bi-esfera) y las tres pinzas (izquierda a derecha con esferas, bi-esferas y tri-esferas) y otra descomposición con dos elementos de sujeción (modelados con bi-esferas) y las tres pinzas (modeladas con tri-esferas).*

Para los niveles elemento, bloque y primitiva, este proceso se puede realizar off-line, obteniendo el modelo envolvente de mínimo volumen. Para estos niveles, los volúmenes esféricos más adecuados son las poli-esferas, cuyos vértices esféricos se expresan en función del sistema de coordenadas local del elemento correspondiente. La representación cinemática comentada anteriormente se utiliza para recalcular la nueva posición de los vértices esféricos cuando el robot se mueva. El nivel elemento no debe ser literalmente una descomposición en los elementos del robot, ya que puede interesar agrupar elementos muy cercanos en un único elemento, como suele ocurrir con la muñeca y la mano, incluida la herramienta, o con la base y el cuerpo. Lo mismo ocurre con las primitivas, pudiéndose modelar un grupo de primitivas con un único modelo envolvente.

Para el modelado en los niveles célula, sistema y sub-sistema, hay que tener en cuenta el hecho de que un cambio en la configuración del sistema hace que el modelo deba actualizarse, a menos que el sistema se modele previamente incluyendo todas sus posibles configuraciones, cosa que resultaría excesivamente costoso. Es en los niveles sistema y sub-sistema donde las esferoides aportan mayores ventajas.

Si bien en cada nivel se pueden aplicar todos los grados de precisión, esto puede ser contraproducente de cara a su aplicación en la detección de colisiones. Por ejemplo, modelar un elemento alargado como el antebrazo de un robot mediante una esfera produce un modelo envolvente con una distribución del error muy irregular, ya que la

distancia de los puntos extremos del elemento a la esfera es muy pequeña mientras que un punto intermedio está muy lejano de su envolvente. Esto puede producir que el módulo de detección de colisiones, al comprobar inicialmente la distancia de la envolvente a un objeto se crea que el elemento está muy cercano cuando está muy lejano o vicerversa. Por tanto, es conveniente eliminar aquellos modelos que no se acoplen bien a la forma de los elementos, y generar la estructura jerárquica extendida con sólo aquellos cuya forma se asemeje al elemento en cuestión. La misma solución se debe aplicar en cualquier otro nivel.

La Figura 3.13 muestra la topología, representada con un grafo AND-OR en forma de árbol, de la estructura jerárquica dinámica extendida generada para un brazo-robot industrial. Los modelos utilizados según diferentes grados de precisión para cada nodo se muestran en la Tabla 3.1, siendo el modelo más costoso de calcular, para los puntos característicos seleccionados, la poli-esfera de seis esferas de control de todo el robot, con un coste próximo a los 18ms. La utilización de esta estructura jerárquica dinámica extendida va a presentar claras ventajas en la detección de colisiones. Para ello, el módulo de detección de colisiones (sección 4.4) es el encargado de recorrer el árbol de forma correcta.

Aunque pueda parecer que la estructura jerárquica planteada sólo sirve para brazos-robot articulados, esto no es cierto: modelar un AGV es modelar un sistema que puede tener sus propios elementos articulados (torreta con cámara de visión, portapalets, etc). El modelo cinemático es el del propio AGV más el de estos elementos articulados. La situación inicial y el modelo exacto del que se parte son los propios del AGV.

El modelo jerárquico del mundo, también resuelve el modelado de los obstáculos, con el mismo nivel jerárquico que el dado en el modelado de robots. Por ejemplo, una mesa se modela globalmente a nivel sistema y mediante sus patas y el tablero superior a nivel elemento. Además, no existe distinción alguna sobre si los obstáculos son estáticos (cinemática nula) o dinámicos (aplicación de un modelo cinemático concreto).

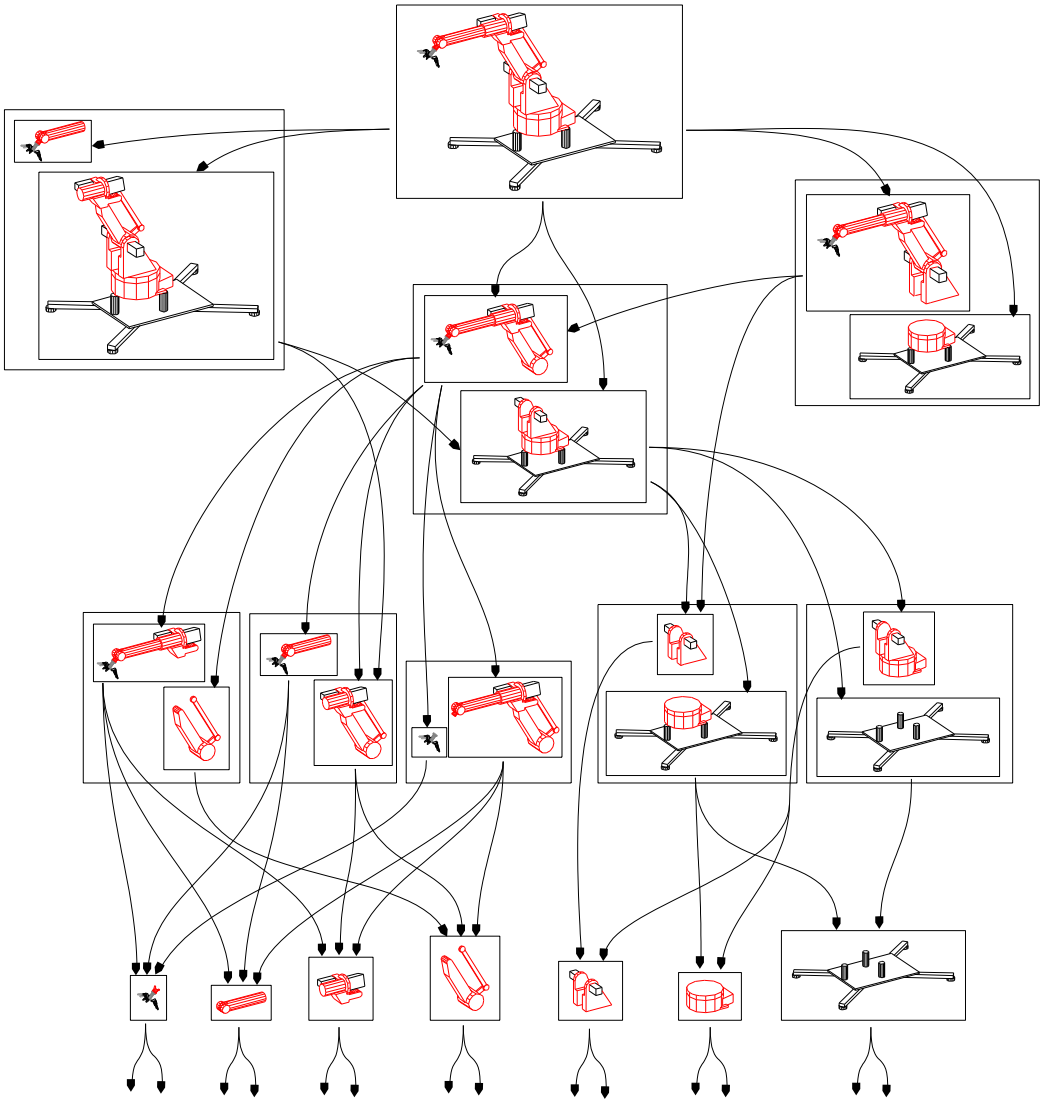
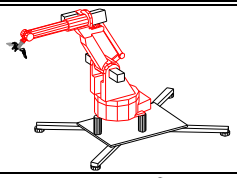
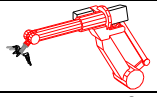
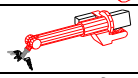
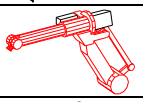
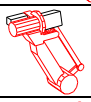
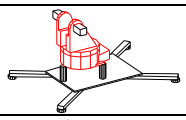
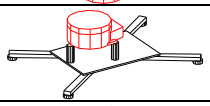
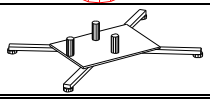


Figura 3.13. Definición del Modelado Jerárquico de un Sistema Robotizado Industrial. *En la gráfica se muestra una parte del árbol de definición del modelo jerárquico de un brazo-robot (IRB1500 de ABB). La descomposición realizada es en los niveles sistema (raíz del árbol), sub-sistema, sub-sub-sistema y elemento, pudiéndose descomponer además en bloques y primitivas (hojas del árbol). En un nodo del árbol pueden surgir diferentes descomposiciones en paralelo, indicadas con diferentes inicios de flechas. Para cada nodo existen más de un modelo envolvente, según el grado de precisión deseado, tal como se indica en la Tabla 3.1.*

Tabla 3.1. *Modelos Esféricos Utilizados para el Modelado de los Nodos del Árbol de la Figura 3.13.*

Nodo del Árbol	Puntos Totales	Puntos Caract.	Modelos Utilizados según Grado de Precisón		
	886	122	Esfera	Poli-Esfera (6)	
	618	82	Esferoide Cuadrática	Esferoide Cúbica	Esferoide Spline
	496	76	Bi-Esfera	Tri-Esfera	
	316	58	Esferoide Cuadrática	Esferoide Spline	Tri-Esfera
	241	56	Bi-Esfera	Tetra-Esfera	
	377	56	Esferoide Cuadrática	Bi-Esfera	Tri-Esfera
	268	66	Esfera	Penta-Esfera	
	105	34	Esfera	Bi-Esfera	Tri-Esfera
	204	56	Penta-Esfera		
	302	36	Esfera	Tetra-Esfera	
	75	24	Bi-Esfera	Tri-Esfera	
	119	34	Esfera	Bi-Esfera	
	122	34	Bi-Esfera	Tetra-Esfera	
	64	24	Esfera	Tri-Esfera	
	41	28	Esfera	Bi-Esfera	
	163	51	Penta-Esfera		

√

CAPÍTULO 4. CÁLCULO DE DISTANCIAS Y DETECCIÓN DE COLISIONES CON MODELOS ESFÉRICOS

Este capítulo hace una revisión de las técnicas de cálculo de distancias y detección de colisiones para modelos tradicionales y posteriormente se extienden estas técnicas a los modelos esféricos presentados anteriormente. En los algoritmos se introducen varias mejoras que permiten obtener buenos resultados para los modelos tratados. Finalmente se presenta una metodología basada en la estructura jerárquica extendida del capítulo anterior para resolver la detección de colisiones de una célula multi-robot.

—Está demasiado lejos, Greg. Es casi un kilómetro.
—Tranquilo —respondió Powell—. No olvides la gravedad mercuriana y un brazo de acero como lanzador. Espera, ¿quieres?
Los ojos del robot estaban midiendo la distancia con mecánica precisión estereoscópica.

"Círculo Vicioso". Isaac Asimov, 1942.

4.1. Detección de Colisiones y Cálculo de Distancias

Por ser campos muy afines, suele ser fácil confundir los conceptos de detección de colisiones y cálculo de distancias entre objetos, pero realmente son diferentes. El cálculo de distancias es el proceso que permite obtener la mínima distancia de separación entre dos objetos. Opcionalmente, puede devolver también la intensidad de penetración entre los objetos cuando interfieren, así como el llamado vector de colisión, definido por los dos puntos más cercanos entre los objetos, perteneciendo un punto a cada objeto. La detección de colisiones es el proceso que permite determinar si un objeto móvil interfiere (colisiona) en algún instante de su movimiento con algún otro objeto (estático o móvil), siendo conveniente que devuelva los instantes de inicio y fin de colisión.

Pese a que el cálculo de distancias se suele utilizar para la detección de colisiones, también tiene otro tipo de aplicaciones, como la planificación de movimientos mediante campos potenciales. Por otra parte, la detección de colisiones puede no requerir el cálculo de distancias, realizando únicamente comprobaciones de interferencia entre los objetos implicados.

4.1.1. Detección de Colisiones

La detección de colisiones se debe utilizar para verificar que un camino generado en la fase de planificación de movimientos es correcto, llamándose a este tipo de planificación de movimientos de *generación-y-prueba*. Una solución al problema de la detección de colisiones puede ser [C&K86]:

- *Eficiente* si es capaz de decidir cuándo en tiempo y espacio se debe volver a comprobar la colisión entre objetos.
- *Completa* cuando se garantice que no hay ninguna colisión que no se detecte.
- *Útil* si informa precisamente de los puntos en el espacio y en el tiempo en los que los objetos empiezan y finalizan a colisionar.

Las soluciones eficientes, completas y útiles permiten una corrección en la planificación de movimientos denominada *con proposición-y-corrección*. Para ello dos objetos se deben clasificar en tres categorías:

- *Separados*, cuando su intersección es vacía.
- *En contacto*, si su intersección está formada sólo por puntos de sus fronteras.
- *Interfiriendo*, la intersección de los objetos incluye puntos de sus interiores.

El inicio de una colisión es una transición de separados a en contacto, mientras que el fin de una colisión es la transición de en contacto a separados. Una transición de

separados a interfiriendo (o viceversa) durante un intervalo de tiempo indica que el inicio (o fin) de la colisión ha ocurrido dentro del intervalo.

Las comparaciones *exhaustivas*, mediante algoritmos de fuerza bruta, entre dos objetos para detectar sus posibles colisiones deben evitarse en lo posible al ser muy costosas computacionalmente. Una forma de evitarlo es envolver cada objeto con una aproximación más simple o con una jerarquía de objetos, como en [Mye81], o en [Fav86] que introduce el concepto de *árbol de ensamblaje*.

En general los métodos de detección de colisiones deben ser *conservativos* en el sentido de que puede ser permisible que devuelvan colisión erróneamente si no existe, pero no pueden cometer el error de no detectar una colisión. Por tanto, la aproximación más fina (con más nivel de detalle) puede ser aquella que envuelva al objeto real mediante una distancia de seguridad o margen de error, aunque pueda ocurrir que se rechacen caminos válidos entre envolventes de los objetos.

Las colisiones se deben detectar tanto en tiempo como en espacio, por lo que considerar intervalos de tiempo fijos puede provocar que colisiones breves empiecen y terminen dentro de uno de los intervalos de tiempo. Por eso, considerar zonas de seguridad con un adecuado intervalo de tiempo garantiza que todas las colisiones se puedan detectar. Por ejemplo si los objetos se agrandan con una distancia de seguridad *ds* y un intervalo de tiempo adecuado *dt*, función de la velocidad de los objetos, se asegurará que ningún objeto se mueve más que una distancia *ds* durante un tiempo *dt*, por lo que no hay que volver a comprobar la posibilidad de colisión dentro de este intervalo temporal. Con esta técnica hay que encontrar un valor *ds* que no sea muy grande (los intervalos de tiempo serán grandes pero se rechazarán muchos caminos válidos) ni muy pequeño (intervalos de tiempo muy pequeños y tiempo de cálculo muy elevado).

[HA+95] define la detección de colisiones como el conjunto de técnicas usadas para asegurar si objetos móviles colisionan entre ellos, a partir de sus formas y movimientos. En robótica, estas técnicas se utilizan para planificar caminos válidos del robot mediante una planificación de movimientos con generación-y-prueba. Esto contrasta con las técnicas de planificación de movimientos llamadas globales, que realizan una búsqueda en el conjunto de configuraciones seguras. En la detección de colisiones existen dos paradigmas:

- Las técnicas basadas en la *búsqueda de raíces*, cuando los caminos de los objetos móviles se define en términos de un parámetro y se usan técnicas de búsqueda de raíces para calcular el valor del parámetro en que se produce una colisión. Las técnicas de este tipo se basan en que existen sólo dos tipos no degenerados de colisión [L-P83b]: cara-vértice y arista-arista, y los objetos a analizar son convexos o están descompuestos en subconjuntos convexos. Dentro de este grupo se encuentran también [Boy79], [Can86], [G&H89], [Cam85] y [Bar89].

- Las técnicas basadas en un *incremento temporal*, cuando se discretiza una escala temporal y en cada paso de una trayectoria se aplica una detección estática de

colisiones. Las colisiones se detectan determinando si la distancia que separa los objetos es nula, mediante una búsqueda exhaustiva o casi exhaustiva entre cada par de elementos. Particularmente dentro de este grupo de técnicas se pueden encontrar dos métodos de resolución:

a) Métodos de cálculo de distancia mediante *geometría computacional*, que optimizan la complejidad computacional asimptótica del peor-caso, como una función del tamaño total de los objetos mediante estructuras de datos adecuadas. Sin embargo, como estos métodos buscan mejorar la complejidad computacional asimptótica, su eficiencia práctica para poliedros con número de vértices relativamente pequeños no es muy elevada. Dentro de este grupo se encuentra [D&K85].

b) Métodos *iterativos con cálculo de distancia*, que buscan una solución que se mueve por las fronteras de los objetos, minimizando una función objetivo cuadrática, la distancia euclídea. La utilización de programación cuadrática para resolver el problema se ha utilizado en [H&S88] y [Bob89], mientras que la utilización de funciones soporte se ha utilizado en [GJK88], [G&J85] y [G&F90].

Las técnicas basadas en un incremento temporal se pueden acelerar mediante cinco tipos de mejoras [HC+92]:

(i) *Aproximación de Primitivas*: es un método off-line que reduce la complejidad del cálculo a realizar en cada paso. Consiste en aproximar los objetos con primitivas geométricas. Al ser una aproximación, las distancias no son exactas y se pueden descartar caminos válidos. Son ejemplos de esta mejora los algoritmos de [Lin94], [GSF94] y [HC+92].

(ii) La *Inicialización* en un paso con el resultado del paso anterior, reduciendo on-line la complejidad de cálculo en cada paso. El grado de mejora depende de la distancia recorrida por el móvil durante un incremento temporal. Son ejemplos de esta mejora los algoritmos de [GJK88] y [L&C91].

(iii) *Actualización de las Distancias*, donde para un camino discreto se resta a la última distancia calculada la mayor longitud recorrida por el móvil. Si el resultado es menor que un cierto umbral, se recalcula la distancia entre los objetos. Así se reduce on-line la cantidad de pasos en los que realizar cálculo para un camino completo. Sin embargo, la nueva distancia es aproximada, no real. Ejemplo de esta mejora son los algoritmos de [C&C86], [Fav89], [HA+95] y [Fio95].

(iv) *Jerarquías Estáticas*, aproximando off-line los objetos mediante árboles de ensamblaje, donde para cada nivel hay una aproximación definida, siendo peor la aproximación para niveles más altos. Así se reduce la cantidad de cálculo para

cada paso. Son ejemplos de esta mejora las diferentes vertientes de [Mye81], [Pas89], [Fav89], [Cam89], [Hub93] y [Qui94].

(v) *Manipulador Virtual*, donde para un manipulador de n grados de libertad se genera al manipulador virtual de k grados de libertad sustituyendo los últimos elementos por su volumen barrido al dejar libres las articulaciones k+1,...,n. Un manipulador virtual de k grados de libertad es una aproximación más gruesa al manipulador original que el manipulador virtual de k+1 grados de libertad. Así se reduce off-line la cantidad de cálculos a realizar en cada paso. Una vez más, [Fav89] también saca partido de esta mejora.

Estas cinco mejoras se pueden clasificar según sean on-line (inicialización y actualización de las distancias) u off-line (aproximación de primitivas, jerarquías estáticas y manipuladores virtuales) o bien según reduzcan la cantidad de pasos en que realizar cálculos (actualización de distancias, jerarquías estáticas y manipulador virtual) o reduzcan la complejidad de los cálculos en cada paso (aproximación de primitivas e inicialización).

Cuando la trayectoria del móvil no se conoce o sólo se conoce parcialmente a priori, sólo las técnicas basadas en incremento temporal se pueden utilizar, obteniendo posiciones y velocidades directamente de los sensores. Por contra, el problema de estas técnicas repetitivas es que las mismas pruebas se deben realizar para cada incremento temporal, aunque se produzcan sólo pequeños cambios entre los objetos. Algunos métodos iterativos [GJK88], [L&C91] y [Lin94], usan la solución del instante anterior para inicializar la búsqueda en el siguiente instante, produciendo una convergencia muy rápida, ya que la inicialización está muy próxima a la solución, siendo muy adecuados para utilizar en una trayectoria completa.

[Boy79] estudió la detección de colisiones entre un poliedro que se mueve, bien con traslación uniforme, bien con rotación uniforme, con obstáculos poliédricos. Se basa en una búsqueda exhaustiva de intersecciones entre cada arista de un poliedro y todas las caras del otro, por lo que no obtiene distancias ni intensidades de penetración. Esta búsqueda exhaustiva de las raíces de las restricciones algebraicas que describen todos los posibles contactos cara-vértice y arista-arista se realiza comprobando si las raíces se encuentran dentro de las restricciones expresadas por las fronteras de los poliedros. Para traslaciones obtiene ecuaciones lineales y para las rotaciones ecuaciones cuadráticas. Sin embargo, no hace mención de ninguna implementación de estas ideas.

Canny ([Can84], [Can86] y [Can88]) presenta un algoritmo capaz de detectar si un poliedro que se mueve en línea recta a velocidad constante y velocidad angular casi constante (un recta en el espacio de configuraciones del móvil, es decir, movimientos aproximadamente inerciales del móvil) colisiona con otros poliedros estáticos, devolviendo los puntos de colisión. Para ello representa la orientación mediante cuaterniones frente a la representación tradicional con ángulos (normalmente de Euler), con lo que elimina funciones trigonométricas, sustituidas por restricciones algebraicas puras. Para no aumentar la complejidad al usar cuaterniones (7 dimensiones en vez de

6) proyecta el espacio 7-dimensional a un hiperspacio 6-dimensional mediante la propiedad del escalado de cuaterniones. Para cada tipo de colisiones definidas en [L-P83b], (cara-vertice, vértice-cara y arista-arista), encuentra una condición en la configuración del móvil, expresada mediante cuaterniones, que definen las superficies frontera entre el espacio libre y el de colisiones con obstáculos en el espacio de configuraciones. El algoritmo, de búsqueda exhaustiva de raíces, no se basa en el cálculo de distancias, sino en condiciones expresadas como ecuaciones cúbicas en un parámetro que si son ciertas devuelven colisión. Las colisiones se obtienen resolviendo una ecuación de quinto grado. Finalmente el algoritmo se amplia para considerar más de un obstáculo móvil.

En [Cam85] se describen varios métodos para la detección de colisiones entre poliedros, pero ninguno de ellos calcula los puntos de colisión cuando hay traslación y rotación. Los caminos se describen mediante segmentos lineales y usa una estructura de datos jerárquica en el espacio 4D tiempo-espacio basada en extrusiones (un poliedro extruido es un hiperpoliedro en 4D). La estructura de datos se forma rápidamente comprobando intersecciones en 4D mediante la poda de secciones enteras. Como ampliación a estos trabajos, en [C&C86] se define la mínima distancia de traslación entre 2 objetos como la longitud de la traslación relativa más corta que hace que los objetos estén en contacto, por lo que devuelve la distancia entre los objetos cuando no interseccionan o la intensidad de penetración cuando interseccionan. Este artículo presenta un algoritmo cuando los objetos son poliédricos, basado en la distancia entre el origen y la diferencia de Minkowski de los poliedros definida mediante semiespacios. También en [C&K86] se amplia el trabajo original, presentando un algoritmo completo y útil para la detección de colisiones entre dos objetos que devuelve el inicio y el final de las colisiones, basándose en los límites de distancia y velocidad y usando la mínima distancia de traslación entre los objetos de [C&C86]. La idea es determinar el tiempo mínimo en que se debe comprobar la colisión entre objetos según la velocidad con que se desplaza el móvil. Muestra un par de ejemplos entre dos esferas cuando una de ellas se traslada o gira respecto a la otra.

Los trabajos de Cameron continúan en esta línea y así en [Cam89] presenta los *s-bounds* como volúmenes envolventes de las primitivas que forman un objeto CSG. Con la definición de operaciones de pseudo-intersección entre los objetos, sólo se debe realizar una búsqueda de intersección en detalle sobre ciertas zonas. En [Cam90] se extiende este trabajo utilizando 4 dimensiones al considerar el volumen espacio-temporal barrido por un objeto a lo largo de su trayectoria, si bien el coste computacional de esta técnica parece elevado [Cam91].

Existen otros trabajos en 4D al considera el tiempo como una cuarta dimensión, como en [J&P87] que resuelve la colisión entre politopos 4D mediante una generalización del algoritmo de [D&K85] cuando los objetos poliédricos se mueven con traslaciones lineales.

Otra alternativa a la detección de colisiones se basa en el *volumen barrido* por los objetos durante su movimiento [PBB83]. La detección de colisiones se realiza comprobando las intersecciones entre los volúmenes barridos con los obstáculos estáticos. Cuando sólo hay un móvil, bastará con comprobar que no existe intersección entre su volumen barrido y los obstáculos, sin necesidad de realizar una comprobación temporal. Por contra, cuando hay más de un móvil, las intersecciones hay que verificarlas dentro del espacio temporal. [F&H93] simplifica el problema mediante subdivisiones de la trayectoria que son aproximadas mediante volúmenes convexos barridos en cada tramo.

[HA+95] estudia el problema de la detección de colisiones entre un número grande de objetos dinámicos, considerando todas las interacciones dos a dos entre objetos y manteniendo una estructura que refleja la inminencia o urgencia de colisión para cada par. Los objetos se determinan mediante sus posiciones y límites en las velocidades y aceleraciones relativas tal como se definen en [FHA90]. También enfocan la cuestión de determinar el intervalo de tiempo mínimo en el que se debe comprobar si se ha producido colisión para asegurar que toda colisión es detectada. Muestra una experiencia del llamado *Algoritmo de Urgencia Dinámica*, que predice colisiones, en el caso de muchas esferas moviéndose simultánea y aleatoriamente. También mediante cotas en las velocidades y aceleraciones de los objetos, en [Hub93] se restringe la zona donde pueden encontrarse los objetos tras un intervalo de tiempo, pudiendo comprobar entonces si se producen interferencia entre los mismos.

La programación lineal y cuadrática también se ha utilizado para resolver la detección de colisiones. En [Bar89], mediante la programación lineal se muestra una solución simple y eficiente a la detección de colisiones para objetos que se mueven con traslaciones puras. La complejidad en el peor caso es lineal al número total de vértices, tal como muestra en varias ejemplos. [H&S88] utiliza técnicas estándares de programación cuadrática al plantear el problema de minimizar la norma del vector entre dos puntos de dos poliedros, con una función donde las restricciones son las caras de los poliedros. De forma análoga, [Bob89] utiliza la programación cuadrática con condiciones especiales, que relacionan las restricciones activas con el gradiente de la función objetivo. El comportamiento es lineal al número de restricciones, pero la complejidad en el peor caso es cuadrática.

Para resolver el caso de detección de colisiones on-line de sistemas robotizados múltiples [HC+92] utiliza jerarquías dinámicas que se ajustan a la configuración actual del robot para calcular el vector de colisión. Pese a presentar diferentes aproximaciones, incluyendo esferas y cilindros esféricos, considera sólo las cajas contenedoras de los elementos del robot por dar una mejor aproximación al vector de colisión. A partir de las cajas contenedoras de los elementos del robot se genera la jerarquía dinámica mediante un árbol de cajas contenedoras. Los resultados se comparan con otros tipos de mejoras posibles clasificadas en varias categorías.

Varios son los trabajos de detección de colisiones en sistemas de simulación. [B&D83] describe el sistema GRASP para la programación off-line de robots con verificación de si los programas robot producen colisiones entre robot y obstáculos mediante una búsqueda exhaustiva de intersecciones entre todas las caras de cada par de objetos usando geometría vectorial estándar. [UOT83] considera el problema de la intersección dividiendo el espacio en pixels y almacenando para cada pixel los objetos que lo contienen, siendo una técnica muy costosa para elevadas precisiones. [GSF94] muestra un sistema de simulación que realiza la detección de si dos objetos colisionan mediante cuatro pasos, de más rápido a menos: (i) verificar la matriz de interés de colisión, que para cada par de objetos almacena si es posible la colisión o no, (ii) calcula la colisión entre pares de elementos *contenedores* (mínima caja que contiene al objeto), (iii) calcula la colisión entre voxels y (iv) calcula la colisión entre pares de caras. Para el tercer punto, se tiene un modelo geométrico jerárquico que ordena los polígonos espacialmente usando voxels.

4.1.2. Cálculo de Distancias

Dobkin y Kirkpatrick [D&K85] analizan algoritmos de cálculo de distancias entre politopos. La separación entre dos poliedros convexos se define como la mínima distancia de un punto de uno a un punto del otro, sin ser necesariamente puntos extremos. El artículo presenta un algoritmo de tiempo lineal como complejidad asintótica o caso-peor para encontrar el par de puntos de distancia mínima entre dos poliedros convexos. Para ello se basa en una descripción jerárquica de los poliedros: una secuencia finita de aproximaciones poliédricas, progresivamente más fina, de forma que cada aproximación contenga a su predecesora. De esta forma un poliedro se almacena en una estructura de datos tipo árbol recursivo. Una extensión de este trabajo [D&K90], que requiere un preprocesamiento de orden lineal, permite determinar el cálculo de distancias entre poliedros convexos con un orden igual al producto de los logaritmos de los vértices de cada objeto.

El algoritmo de [D&K90] es uno de las mejores cotas teóricas de la complejidad computacional caso-peor, pero al aplicar el cálculo de distancias a la detección de colisiones, normalmente, no es tan importante la complejidad computacional en el caso-peor, sino que suelen interesar algoritmos heurísticamente buenos (de coste aproximadamente lineal) aunque tengan mayor complejidad asintótica. [Tor95] muestra que en la detección de colisiones hay dos estrategias de acotación espacial para determinar las zonas de un objeto con mayor posibilidad de colisión con otros objetos, las basadas en la *delimitación de un volumen*, donde se encuentran las mejoras por jerarquías estáticas comentadas en el apartado anterior y las basadas en la *restricción de los elementos fronterizos de los objetos que pueden entrar en contacto*, como los métodos de clasificación de aristas que se comentan a continuación.

Meyer introdujo en [Mey86] el algoritmo de *clasificación de aristas* que determina la mínima distancia entre dos paralepípedos rectangulares (*boxes*). Se basa en compartimentar el espacio respecto a un *box* en 27 regiones: la caja en sí, 8 zonas (una por vértice) tipo cónico, 12 zonas (una por arista) tipo cuña, 6 zonas (una por cara) tipo

barra. Para un punto de cada una de estas zonas, se conoce cual es el elemento más cercano al punto, por lo que de la otra caja se analiza cada arista en qué zonas cae, realizándose un tratamiento directo para cada zona. Cuando la arista cae en varias zonas se subdivide en subaristas que caen cada una en una zona, a no ser que se detecte que la arista apunte hacia afuera de la primera caja. Frente a algoritmos de búsqueda exhaustiva, que tratan todos los elementos (caras, aristas y vértices) igual, el algoritmo de la clasificación de aristas realiza un tratamiento especial según la situación relativa de cada elemento de una caja respecto a la otra caja. También explica cómo plantear el problema como programación u optimización no lineal.

En [L&C91] y [L&C92] se presenta un algoritmo para el cálculo de distancias entre politopos que tiene sus raíces en el método anterior de Meyer. Se basa en encontrar los dos elementos (cara, arista, vértice) más cercanos, mediante los tres criterios de aplicabilidad de [Don84], que si fallan dan lugar a otro par de elementos más cercanos. El coste es heurísticamente menor que lineal al número de vértices, pero resulta especialmente interesante cuando, conocidos los dos elementos más próximos, algún politopo (o ambos) se mueve, ya que los elementos más próximos se suelen mantener para movimientos incrementales. En este caso el coste es constante, produciéndose no más de un cambio durante cada paso del algoritmo. Resulta conveniente realizar un preproceso para evitar casos como un vértice intersección de más de cuatro o cinco aristas, caras limitadas por más de cuatro o cinco aristas, etcétera. Este trabajo inicial se completa en [Lin94] donde se presenta un algoritmo para la detección de colisiones y la determinación de contactos entre modelos geométricos, descritos mediante fronteras lineales o curvas (superficies splines racionales o funciones algebraicas), con movimientos rígidos. El núcleo del algoritmo es un método incremental simple y rápido para calcular la distancia entre poliedros convexos que utiliza las propiedades de convexidad para establecer unos criterios de aplicabilidad y verificar los elementos más cercanos. Realiza un preproceso para que el número de elementos vecinos de un elemento sea constante y garantizar ejecuciones en tiempo constante para cada prueba. Para politopos no convexos usa una representación jerárquica de la medida de la distancia. Además describe dos métodos para reducir la frecuencia de la realización de la detección de colisiones entre pares de objetos móviles: utilizando una cola de prioridades ordenada por la mínima cota en tiempo de colisión y utilizando un test de solapamiento entre cajas contenedoras. Finalmente presenta un algoritmo global oportunístico de planificación de movimientos [C&L93] que usa el algoritmo del cálculo de la distancia incremental para obtener un esqueleto mono-dimensional sobre el que se debe mover el móvil.

Lin, junto a Manocha, ha extendido recientemente sus trabajos para entornos dinámicos con múltiples objetos complejos mediante una técnica de reducción de dimensión para podar el número de pruebas a realizar [CL+94] y [CL+95]; para cualquier tipo de objetos B-rep mediante una jerarquía de cajas contenedoras mediante una variante de la representación oct-tree [PML94]; para poliedros no convexos mediante barridos y podas jerárquicos [LMP95]; para objetos curvos definidos mediante superficies NURBS y Bézier [L&M95].

En [T&T94] también se extiende esta técnica para modelos poliédricos generales, no necesariamente convexos, evitando además el preprocesamiento inicial de [Lin94] que puede crear numerosas caras y aristas ficticias con el consiguiente aumento de coste computacional en los algoritmos de determinación de interferencias. Cada predicado de un criterio de aplicabilidad se transforma en la evaluación del signo de un determinante, lo que aumenta la robustez del algoritmo. Además, muestran la forma de obtener una cota inferior de la distancia entre los poliedros no-convexos. La utilización de una estrategia de poda mediante la representación de un grafo esférico de orientaciones de caras en [J&T95] permite reducir el espacio de búsqueda de la distancia entre los objetos para la detección de colisiones con movimientos translacionales. Una poda de pruebas en la detección de colisiones utilizando descomposición celular se utiliza en [E&T95].

Gilbert es uno de los investigadores más productivos en el campo del cálculo de distancias. En [G&J85] formulan un problema de planificación de movimientos como control óptimo, donde las distancias son restricciones activas para la detección de colisiones. Para ello realizan un estudio de la continuidad y derivabilidad de la distancia como función del estado del sistema, obteniendo que sólo son continuamente diferenciables cuando los objetos son modelados mediante conjuntos convexos. Los resultados los aplican a un móvil 2D con rotación. En [J&G85] aplica los resultados anteriores considerando el espacio 3D y un control óptimo de tiempo mínimo.

Posteriormente, en [GJK88] se presenta un algoritmo para determinar la distancia entre conjuntos compactos, optimizado para politopos 3D. El algoritmo es eficiente y aproximadamente lineal en el número de puntos total que definen los politopos. Además devuelve los puntos más cercanos de los politopos o una cota a la intensidad de penetración entre ellos. Para aplicaciones de detección de colisiones se puede inicializar de una forma especial en función del resultado anterior para reducir el coste computacional. Los autores vuelven a utilizar la extensión esférica de politopos (con radio constante) como en [J&G85] para garantizar una distancia de seguridad. El algoritmo se basa en la diferencia de Minkowski y la evaluación de una función soporte que se puede expresar, al igual que su solución, mediante productos escalares de los vértices que definen la diferencia de Minkowski. El algoritmo es iterativo, evaluando para cada paso la función soporte y su solución, inicializándose según una dirección como por ejemplo la del centroide. En cada paso, para una dirección, considera la solución soporte y los vértices mediante los que se definen.

En [G&H89] explica un algoritmo iterativo para detección de colisiones entre dos objetos convexos cuyos movimientos se definen por un camino con traslación y orientación en el espacio de configuraciones. Ambos objetos se pueden mover simultáneamente de forma independiente. El algoritmo devuelve que no hay colisión o el primer punto donde colisionan. Estudia los casos 2D y 3D. El algoritmo se basa en determinar, para una dirección dada, el hueco de menor espesor según los hiperplanos definidos por la función soporte en cada politopo, así como el espesor de este hueco, que es la mínima distancia entre los politopos en la dirección dada. Plantear esta distancia en función de la variable de configuración de los politopos y resolver la ecuación que hace la distancia nula, devuelve si no hay colisión en el intervalo a medir

o por el contrario cuándo volver a considerar otra dirección para evaluar su función soporte.

Finalmente, en [G&F90] generaliza el algoritmo de cálculo de distancias de [GJK88] para cualquier tipo de objetos convexos en 3D, incluyendo objetos curvos, evitando errores por aproximación de politopos. Consiste en una serie de reglas que permiten obtener tanto la función soporte como la solución soporte de objetos complejos en base a las funciones soporte y soluciones soporte de los elementos básicos componentes de esos objetos. Para ello presenta fórmulas de diversos objetos básicos, tanto objetos 3D como elementos 2D en el espacio 3D. Así consigue reducir el tiempo al calcular la función soporte y la solución soporte con fórmulas directas en vez de tener que realizar comparaciones con productos escalares de puntos por el vector dirección en cada paso.

Algunos de los colaboradores de Gilbert, como Keerthi, han seguido trabajando en estos temas: en [S&K92] analiza la complejidad computacional de cuatro problemas de proximidad entre politopos convexos: a) comprobar si dos politopos interseccionan, b) si se tocan, en caso de no interseccionar, c) la mínima distancia entre ellos d) y si interseccionan, la intensidad de penetración entre ellos, mientras que en [SSK93] presenta algoritmos para calcular la intensidad de penetración entre objetos planares y poliédricos cuando los objetos colisionan.

También se han utilizado técnicas de optimización en el cálculo de distancias. [ZRL92] realiza el cálculo de distancias entre politopos convexos 3D basado en una minimización de la función distancia obteniendo una sucesión de direcciones óptimas de búsqueda sobre las frontera de los politopos, para dar lugar a los dos puntos más cercanos. El algoritmo combina el método de la proyección del gradiente con una dirección de búsqueda óptima adicional cuando el primer método da fenómenos zigzagueantes. En este artículo, los objetos se modelan mediante politopos definidos por semiespacios.

Normalmente, al calcular la distancia entre dos objetos se suele considerar la norma euclídea entre dos puntos, cada uno perteneciente a la frontera de un objeto. Sin embargo, hay trabajos que utilizan otros tipos de normas al considerar la distancia, como la norma-1 o la norma-∞ ([K&S89a] y [K&S89b]).

4.2. Cálculo de Distancias entre Politopos

En la sección anterior se han repasado las técnicas más importantes de detección de colisiones y cálculo de distancias. En esta sección se analizarán dos conceptos utilizados posteriormente para aplicar sobre los modelos esféricos: la diferencia de Minkowski y la función soporte.

4.2.1. Diferencia de Minkowski y Función Soporte

La diferencia de Minkowski entre dos poliedros convexos es un poliedro convexo cuyos puntos frontera representan configuraciones de contacto entre los poliedros originales, mientras que los puntos del interior representan configuraciones de intersección. Por tanto, la mínima distancia de traslación entre dos poliedros es igual a la mínima distancia de traslación entre el Origen y la diferencia de Minkowski entre los poliedros. Las caras del poliedro generado mediante la diferencia de Minkowski se corresponden a uno de los tipos de contacto definidos en [L-P83b]: cara-vértice, vértice-cara y arista-arista.

El cálculo de distancias entre politopos desarrollado en [GJK88] aprovecha el hecho de que la distancia mínima entre dos objetos es la misma que la distancia más corta desde el Origen al objeto formado por la Diferencia de Minkowski de los dos objetos. Para ello, dados dos conjuntos de puntos $\mathbf{P}_1$ y $\mathbf{P}_2$, que definen dos politopos, se forma un nuevo politopo con el conjunto de puntos siguiente:

$$\mathbf{P}_{21} = \mathbf{P}_2 - \mathbf{P}_1 = \{p : p = p_2 - p_1,\ \ p_1 \in \mathbf{P}_1, p_2 \in \mathbf{P}_2\}$$

Aunque fue Wolfe [Wol76] el primero en utilizar una función soporte para el cálculo de distancias sobre la Diferencia de Minkowski de dos politopos, su utilización en robótica se plantea por primera vez en [GJK88]. La *función soporte* $h_{\mathbf{P}}(\eta)$ de un politopo $\mathbf{P} \subset \Re^m$ en una dirección $\eta \in \Re^m$ es el máximo producto escalar de un punto de $\mathbf{P}$ por η:

$$h_{\mathbf{P}}(\eta) = \max_{p \in \mathbf{P}} \eta \cdot p,\ \eta \in \Re^m$$

Una *solución soporte* de un politopo para una dirección, $p_{\mathbf{P}}(\eta)$, es aquel vértice del politopo que devuelve la función soporte (máximo producto escalar con el vector director), es decir el punto del politopo más lejano en la dirección dada:

$$p_{\mathbf{P}}(\eta) \in \mathbf{P} : \eta \cdot p_{\mathbf{P}}(\eta) = h_{\mathbf{P}}(\eta)$$

Tanto la función soporte como la solución soporte se pueden definir mediante los vértices del politopo.

El algoritmo de cálculo de distancia entre los politopos **P** y **Q** basado en la diferencia de Minkowski y la función soporte se descompone en un subalgoritmo de cálculo de distancias y en un algoritmo iterativo que para cada paso se aprovecha la dirección "probablemente más adecuada" para calcular el punto más cercano y considerar sólo los vértices necesarios para definirlo. La dirección probablemente más adecuada viene definida por el punto más cercano al origen encontrado hasta ahora.

Básicamente, el algoritmo tiene los siguientes pasos:

1. *Obtener* **P-Q** *a partir de la diferencia de Minkowski entre* **P** *y* **Q**.
2. *Inicializar un conjunto* $\mathbf{V}_0$ *con el punto* $v_0=p_{\mathbf{P-Q}}$ *solución soporte de* **P-Q** *para la dirección* $c_{\mathbf{P}}-c_{\mathbf{Q}}$, *siendo* $c_{\mathbf{P}}$ *y* $c_{\mathbf{Q}}$ *los centroides de* **P** *y* **Q**, *es decir, con el vértice de* **P-Q** *que de máximo producto escalar con* $c_{\mathbf{P}}-c_{\mathbf{Q}}$.
3. *Calcula el punto* v_i *del conjunto* $\mathbf{V}_i$ *más cercano al origen mediante el subalgoritmo de cálculo de distancia. Se considera* $-v_i$ *la dirección "probablemente más adecuada" de la distancia del origen a* **P-Q**.
4. *Obtener* v_{i+1} *como la solución soporte de* **P-Q** *para la dirección* $-v_i$, *es decir el vértice de* **P-Q** *más cercano al origen en la dirección* $-v_i$.
5. *Actualizar el conjunto* $\mathbf{V}_{i+1}$ *incluyendo* v_{i+1} *en* $\mathbf{V}_i$ *y eliminando los vértices no necesarios para definir* v_{i+1}.
6. *Incrementar* i *e ir al paso 3 si no se cumple la condición de convergencia (por ejemplo,* $|(\|v_{i+1}\|^2-(h_{\mathbf{P-Q}}(-v_{i+1}))^2)|$ *menor que cierta tolerancia).*

El subalgoritmo del cálculo de distancia cálcula el punto más cercano al origen de un politopo con hasta cuatro vértices. Además, devuelve los únicos vértices necesarios para definir ese punto: por ejemplo, si el punto más cercano es uno de los vértices del politopo, sólo ese vértice se necesita para definir el punto más cercano; si el vértice está en una arista (cara) solo sus dos (tres) vértices extremos se necesitan. Sólo se requieren cuatro vértices cuando el punto está en el interior del objeto, lo que produce una colisión.

4.3. Cálculo de Distancias entre Poli-Esferas

Pese a que las poli-esferas son una generalización de los politopos, permitiendo un dominio de modelos mucho más amplio, esto no complica el cálculo de distancias. En ambos problemas, cálculo de distancias entre politopos y entre poli-esferas, se debe encontrar el par de elementos más cercanos: en el primer caso los dos puntos más cercanos y en el segundo, las dos esferas más cercanas. Esto implica que para idéntica complejidad del problema, con poli-esferas se tiene la ventaja de poder calcular la distancia entre objetos más generales y por tanto más complejos.

El algoritmo de [GJK88] del apartado anterior o bien calcula la distancia más corta entre dos politopos, devolviendo además un par de puntos (uno por politopo) que definen esta distancia y el vector de colisión, o bien devuelve que existe colisión entre los politopos. Sin embargo, este algoritmo presenta los tres problemas siguientes:

- No es general para poder ser aplicado a poli-esferas.
- No devuelve la intensidad de penetración cuando los objetos colisionan, lo que será necesario conocer cuando se utilice el cálculo de distancias para la detección de colisiones con estructuras jerárquicas extendidas de sistemas robotizados.
- No devuelve todas los posibles pares de elementos (puntos en este caso) que definen la distancia entre los objetos (politopos en este caso): si existe más de una solución, devuelve aleatoriamente una de ellas. Por ejemplo, cuando un objeto tiene una arista paralela a un plano del otro objeto, la distancia más corta entre los objetos viene determinada por ese par arista/plano: sin embargo, el algoritmo devuelve aleatoriamente un par de puntos solución (uno perteneciente a la arista de un objeto y otro al plano del segundo objeto) de los infinitos pares solución existentes. Para la utilización de este módulo del cálculo de distancias en la planificación de movimientos de objetos con rotación, va a resultar necesario determinar completamente los elementos que definen el par de elementos más cercanos entre las objetos.

El problema de encontrar la distancia más corta entre dos poli-esferas, $S_{0\text{-}n}$ y S'_{0-m} se puede formalizar, al igual que el cálculo de distancias entre dos politopos, como un problema de minimización:

$$d(S_{0\text{-}n}, S'_{0-m}) = \min_{\substack{s_i \in S_{0\text{-}n} \\ s'_j \in S'_{0-m}}} d(s_i, s'_j)$$

siendo $d(s_i, s'_j)$ la distancia entre las esferas s_i y s'_j, es decir, la norma esférica mínima, $\left\| \mathbf{s}_{ij} \right\|_m$, del vector esférico $\mathbf{s}_{ij}$ definido por las esferas s_i y s'_j.

Formulando en el espacio de los parámetros queda:

$$d(S_{0\text{-}n}, S'_{0-m}) = \min_{\substack{\lambda_i, i=1,\ldots,n \\ \lambda'_j, j=1,\ldots,m}} d(S_{0\text{-}n}(\lambda_1,\ldots,\lambda_n), S'_{0-m}(\lambda'_0,\ldots,\lambda'_m))$$

Muchos métodos numéricos se pueden aplicar al problema, pero un algoritmo para una solución geométrica muy rápida se muestra en [THK91], formalizada en [HKT92] como una extensión esférica del cálculo de distancias entre politopos comentado en la sección anterior. Con este algoritmo se resuelve el primer problema, la generalización del cálculo de distancias entre politopos al cálculo de distancias entre poli-esferas, pero permanenecen los dos últimos problemas.

A continuación se muestra otra forma de realizar una extensión esférica del algoritmo del cálculo de distancias entre politopos para poder aplicarlo al cálculo de distancias entre dos poli-esferas. Este nuevo algoritmo devuelve la intensidad de penetración cuando los objetos colisionan, así como los elementos necesarios para definir completamente la distancia entre las poli-esferas. Al igual que en el planteamiento original, el método se basa en unos algoritmos básicos, que permiten calcular la distancia al origen o la intensidad de penetración del origen de los cuatro tipos básicos de poli-esferas (esfera, bi-esfera, tri-esfera y tetra-esfera) y un algoritmo iterativo que se basa en los algoritmos básicos.

4.3.1. Algoritmos Básicos de Cálculo de Distancias al Origen

Los algoritmos básicos para el cálculo de distancias entre poli-esferas son cuatro algoritmos diferentes pero interdependientes, consistentes en el cálculo de la distancia al origen de una esfera, una bi-esfera, una tri-esfera y una tetra-esfera. Estos cuatro algoritmos se utilizan para encontrar la distancia entre una poli-esfera y el origen ya que éste se basa en los algoritmos de poli-esferas con menor número de vértices.

Distancia a una Esfera

Al hablar de distancia a un objeto nos referimos a la mínima distancia entre el Origen y este objeto. Lo podemos denotar por $DIST_O$(objeto).

Para una esfera s_i, este cálculo es trivial:

$$DIST_O(s_i)=||\mathbf{s}_i||_m$$

Nótese que si el origen está incluido en la esfera s_i, la distancia devuelve la intensidad de penetración.

Se define la distancia máxima del origen a la esfera s_i como $DIST_{OM}(s_i)=||\mathbf{s}_i||_M$.

Distancia a una Bi-Esfera

La distancia del Origen a una bi-esfera S_{01} es igual a la distancia del origen a un esfera s_i perteneciente al conjunto de esferas barridas por la bi-esfera, es decir, $s_i(\lambda)=S_{01}(\lambda)$ para cierto λ, de forma que los vectores esféricos $\mathbf{s}_i(\lambda)$ y $\mathbf{s}_{01}$ son perpendiculares, cumpliendo

$$\mathbf{s}_i(\lambda)\cdot\mathbf{s}_{01}=0$$

lo que da lugar a la ecuación siguiente:

$$\cos\beta_i(\lambda) = \cos\alpha_i(\lambda)\cos\gamma_{01} + \sin\alpha_i(\lambda)\sin\gamma_{01} = 0$$

siendo $\beta_i(\lambda)$ el ángulo que forma el eje $\mathbf{v}_i(\lambda)$ del vector esférico $\mathbf{s}_i(\lambda)$ con el cono tangente del vector esférico $\mathbf{s}_{01}$; $\alpha_i(\lambda)$ el ángulo que forman los ejes $\mathbf{v}_i(\lambda)$ y $\mathbf{v}_{01}$ y γ_{01} el ángulo de convergencia del vector esférico $\mathbf{s}_{01}$. Sustituyendo en la ecuación anterior $\cos\alpha_i(\lambda) = \dfrac{\mathbf{v}_i(\lambda)\cdot\mathbf{v}_{01}}{\|\mathbf{v}_i(\lambda)\|\|\mathbf{v}_{01}\|}, \sin\gamma_{01} = \dfrac{\mathbf{s}_{01}.\mathrm{r}}{\|\mathbf{v}_{01}\|}$:

$$\mathbf{s}_{01}.\mathrm{r}^2(\mathbf{v}_i(\lambda)\cdot\mathbf{v}_i(\lambda)) = (\mathbf{v}_i(\lambda)\cdot\mathbf{v}_{01})^2$$

Sustituyendo $\mathbf{v}_i(\lambda) = \mathbf{v}_0 + \lambda\mathbf{v}_{01}$, (siendo $\mathbf{v}_0$ el vector del origen al centro de s_0) y operando:

$$\|\mathbf{v}_{01}\|^2\left(\|\mathbf{v}_{01}\|^2 - \mathbf{s}_{01}.\mathrm{r}^2\right)\lambda^2 + 2\left(\|\mathbf{v}_{01}\|^2 - \mathbf{s}_{01}.\mathrm{r}^2\right)\left(\mathbf{v}_0\cdot\mathbf{v}_{01}\right)\lambda + \left(\mathbf{v}_0\cdot\mathbf{v}_{01}\right)^2 - \mathbf{s}_{01}.\mathrm{r}^2\|\mathbf{v}_0\|^2 = 0$$

Resolviendo la ecuación de segundo grado obtenida y simplificando con las expresiones $\|\mathbf{a}\|^2\|\mathbf{b}\|^2 - (\mathbf{a}\cdot\mathbf{b})^2 = (\mathbf{a}\times\mathbf{b})\cdot(\mathbf{a}\times\mathbf{b}) = \|\mathbf{a}\times\mathbf{b}\|^2$ y $\mathrm{tg}\,\gamma_{01} = \dfrac{\mathbf{s}_{01}.\mathrm{r}}{\sqrt{\mathbf{s}_{01}.\mathrm{r}^2 - \|\mathbf{v}_{01}\|^2}}$ se tiene:

$$\lambda = \frac{-\left(\mathbf{v}_0\cdot\mathbf{v}_{01}\right)}{\|\mathbf{v}_{01}\|^2} \pm \mathrm{tg}\,\gamma_{01}\frac{\|\mathbf{v}_0\times\mathbf{v}_{01}\|}{\|\mathbf{v}_{01}\|^2} = \lambda^{\perp} \pm \Delta\lambda$$

El primer término de la solución $\lambda^{\perp}$ representa aquel valor de λ que hace perpendicular a los ejes $\mathbf{v}_i(\lambda)$ y $\mathbf{v}_{01}$, dando la esfera barrida por la bi-esfera con centro más cercano al origen, mientras que el segundo término $\Delta\lambda$ sirve como corrector para que la perpendicularidad sea entre el eje $\mathbf{v}_i(\lambda)$ y el cono tangente del vector esférico $\mathbf{s}_{01}$. Estas dos soluciones representan los vectores esféricos que son perpendiculares al cono tangente de la bi-esfera por fuera y por dentro, tal como se muestra en la Figura 4.1. Es fácil comprobar que la solución adecuada es sumando ambos términos, tanto si $\mathrm{tg}\,\gamma_{01}$ es positivo como negativo, por lo que no es necesario comprobar ambas soluciones y descartar una.

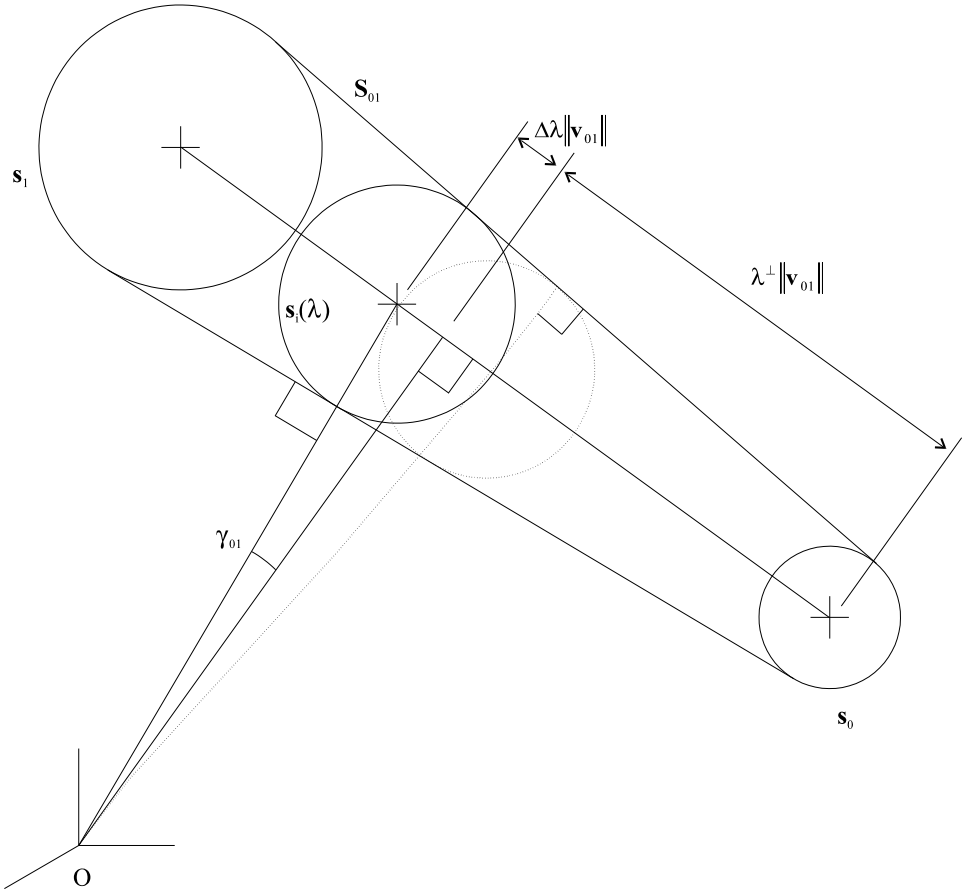


Figura 4.1. Distancia del Origen a una Bi-Esfera. *El cálculo de la esfera barrida por la bi-esfera más próxima al origen se realiza en dos pasos: primero se determina aquella esfera que tenga como centro el punto del eje de la bi-esfera más cercano al origen y después se corrige esta posición para encontrar la esfera deseada. De esta forma se obtiene una esfera cuyo vector esférico es perpendicular al vector esférico de la bi-esfera. El vector esférico dibujado a puntos se descarta por ser su eje perpendicular al cono tangente de la bi-esfera por dentro de la misma.*

Al mismo resultado se llega en [THK91] mediante una interpretación geométrica directa del problema o si se considera que la derivada respecto a λ de la mínima distancia del origen al vector esférico $\mathbf{s}_i(\lambda)$ debe ser cero:

$$\frac{d}{d\lambda}\left\|\mathbf{s}_i(\lambda)\right\|_m = \frac{d}{d\lambda}\left\|s_0 + \lambda\mathbf{s}_{01}\right\|_m = 0$$

lo que produce una ecuación de segundo grado idéntica a la anterior.

Finalmente, si $\lambda \in [0,1]$ entonces $DIST_O(S_{01})=DIST_O(s_i(\lambda))$. Si el parámetro λ está fuera del rango [0,1], se tiene que si $\lambda<0$, $DIST_O(S_{01})=DIST_O(s_0)$ y por el contrario, si $\lambda>1$, $DIST_O(S_{01})=DIST_O(s_1)$. Nótese que si el origen está incluido en la bi-esfera S_{01}, la distancia devuelve la intensidad de penetración, al basarse en el cálculo de la distancia del origen a una esfera.

Se define la distancia máxima del origen a la bi-esfera $DIST_{OM}(S_{01})$ a aquella obtenida utilizando $\lambda-\Delta\lambda$ y la distancia máxima a la esfera en vez de la distancia a la esfera. La Tabla 4.1. resume el cálculo de ambas distancias.

Tabla 4.1. *Cálculo de las distancias mínima y máxima de una bi-esfera al origen*

$$\lambda^{\perp} = \frac{-\left(\mathbf{v}_0 \cdot \mathbf{v}_{01}\right)}{\left\|\mathbf{v}_{01}\right\|^2}$$

$$\Delta\lambda = \operatorname{tg}\gamma_{01}\frac{\left\|\mathbf{v}_0 \times \mathbf{v}_{01}\right\|}{\left\|\mathbf{v}_{01}\right\|^2}$$

$\lambda = \lambda^{\perp} + \Delta\lambda$		$\lambda_M = \lambda^{\perp} - \Delta\lambda$	
$0 \le \lambda \le 1$	$DIST_O(S_{01})=DIST_O(s_i(\lambda))$	$0 \le \lambda_M \le 1$	$DIST_{OM}(S_{01})=DIST_{OM}(s_i(\lambda_M))$
$\lambda<0$	$DIST_O(S_{01})=DIST_O(s_0)$	$\lambda_M<0$	$DIST_{OM}(S_{01})=DIST_{OM}(s_0)$
$\lambda>1$	$DIST_O(S_{01})=DIST_O(s_1)$	$\lambda_M>1$	$DIST_{OM}(S_{01})=DIST_{OM}(s_1)$

Nota: Sombreadas se encuentran las condiciones que representan fuera de rango

Distancia a una Tri-Esfera

La dirección de la distancia más corta del origen a la tri-esfera $\mathbf{S}_{012}(\lambda_1,\lambda_2)$ viene dada por la dirección del eje de uno de los vectores esféricos normales a la tri-esfera, resultado de los productos vectoriales esféricos $\mathbf{s}_{012}=\mathbf{s}_{01}\times\mathbf{s}_{02}$ o $\mathbf{s}_{021}=\mathbf{s}_{02}\times\mathbf{s}_{01}$. Para determinar cual de los dos casos posibles es el adecuado, basta comprobar si $\mathbf{v}_0 \cdot \left(\mathbf{v}_{01} \times \mathbf{v}_{02}\right) < 0$, en cuyo caso se considera $\mathbf{s}_{012}$, mientras que en caso contrario, se considera $\mathbf{s}_{021}$. El siguiente desarrollo presupone que el vector adecuado es $\mathbf{s}_{012}$, sustituyéndose por $\mathbf{s}_{021}$ en caso necesario.

Sea $s_{ij}=\mathbf{S}_{012}(\lambda_i,\lambda_j)$ la esfera barrida por la tri-esfera más cercana al origen y sea $d_{ij}=\|s_{ij}.c\|$ la distancia del origen al centro de la esfera s_{ij}. Es evidente que se debe cumplir:

$$-d_{ij}\,\hat{\mathbf{v}}_{012} = \mathbf{v}_0 + \lambda_i\,\mathbf{v}_{01} + \lambda_j\,\mathbf{v}_{02}$$

Multiplicando esta ecuación escalarmente por $\mathbf{v}_{01}\times\mathbf{v}_{012}$ y teniendo en cuenta que $\mathbf{a}\cdot(\mathbf{a}\times\mathbf{b})=0$ ya que **a** y **a**×**b** son perpendiculares da:

$$0=\mathbf{v}_0\cdot\left(\mathbf{v}_{01}\times\mathbf{v}_{012}\right)+0+\lambda_j\,\mathbf{v}_{02}\cdot\left(\mathbf{v}_{01}\times\mathbf{v}_{012}\right)\Rightarrow\lambda_j=-\frac{\mathbf{v}_0\cdot\left(\mathbf{v}_{01}\times\mathbf{v}_{012}\right)}{\mathbf{v}_{02}\cdot\left(\mathbf{v}_{01}\times\mathbf{v}_{012}\right)}$$

De la misma manera, si la misma ecuación se multiplica ahora por $\mathbf{v}_{02}\times\mathbf{v}_{012}$ se tiene:

$$\lambda_i=-\frac{\mathbf{v}_0\cdot\left(\mathbf{v}_{02}\times\mathbf{v}_{012}\right)}{\mathbf{v}_{01}\cdot\left(\mathbf{v}_{02}\times\mathbf{v}_{012}\right)}$$

Con lo que se obtienen los valores λ_i, λ_j que definen la esfera s_{ij} barrida por la tri-esfera S_{012} más cercana al origen. Finalmente, los parámetros λ_i, λ_j pueden estar dentro o fuera del rango delimitado por $\lambda_i\geq0$, $\lambda_j\geq0$, $\lambda_i+\lambda_j\leq1$ presentándose diferentes casos, ilustrados en la Tabla 4.2.

Tabla 4.2. *Cálculo de la distancia mínima de una tri-esfera al origen*

Si $\mathbf{v}_0\cdot\left(\mathbf{v}_{01}\times\mathbf{v}_{02}\right)<0$ entonces

$\mathbf{v}_i=\left(\mathbf{v}_{02}\times\mathbf{v}_{012}\right)$, $\mathbf{v}_j=\left(\mathbf{v}_{01}\times\mathbf{v}_{012}\right)$

si no

$\mathbf{v}_i=\left(\mathbf{v}_{02}\times\mathbf{v}_{021}\right)$, $\mathbf{v}_j=\left(\mathbf{v}_{01}\times\mathbf{v}_{021}\right)$

$\lambda_i=-\frac{\mathbf{v}_0\cdot\mathbf{v}_i}{\mathbf{v}_{01}\cdot\mathbf{v}_i}$, $\lambda_j=-\frac{\mathbf{v}_0\cdot\mathbf{v}_j}{\mathbf{v}_{02}\cdot\mathbf{v}_j}$

$0\leq\lambda_i$	$0\leq\lambda_j$	$\lambda_i+\lambda_j\leq1$	$DIST_O(S_{012})=DIST_O(s_{ij})$
$\lambda_i<0$	$0\leq\lambda_j$	$\lambda_i+\lambda_j\leq1$	$DIST_O(S_{012})=DIST_O(S_{02})$
$0\leq\lambda_i$	$\lambda_j<0$	$\lambda_i+\lambda_j\leq1$	$DIST_O(S_{012})=DIST_O(S_{01})$
$0\leq\lambda_i$	$0\leq\lambda_j$	$1<\lambda_i+\lambda_j$	$DIST_O(S_{012})=DIST_O(S_{12})$
$\lambda_i<0$	$\lambda_j<0$	$\lambda_i+\lambda_j\leq1$	$DIST_O(S_{012})=\min(DIST_O(S_{01}), DIST_O(S_{02}))$
$\lambda_i<0$	$0\leq\lambda_j$	$1<\lambda_i+\lambda_j$	$DIST_O(S_{012})=\min(DIST_O(S_{02}), DIST_O(S_{12}))$
$0\leq\lambda_i$	$\lambda_j<0$	$1<\lambda_i+\lambda_j$	$DIST_O(S_{012})=\min(DIST_O(S_{01}), DIST_O(S_{12}))$
$\lambda_i<0$	$\lambda_j<0$	$1<\lambda_i+\lambda_j$	IMPOSIBLE

Nota: Sombreadas se encuentran las condiciones que representan fuera de rango

Como el cálculo de la distancia del origen a una tri-esfera se basa en el cálculo de la distancia del origen a una esfera o a una bi-esfera, si el origen está incluido en la tri-esfera, el algoritmo devuelve la intensidad de penetración.

Se define la distancia máxima a la tri-esfera, $DIST_{OM}(S_{012})$ a aquella calculada usando el eje del otro vector esférico normal y las distancias máximas a la esfera y la bi-esfera cuando corresponda. Su cálculo viene definido en la Tabla 4.3.

Tabla 4.3. *Cálculo de la distancia máxima de una tri-esfera al origen*

Si $\mathbf{v}_0 \cdot (\mathbf{v}_{01} \times \mathbf{v}_{02}) > 0$ entonces

$$\mathbf{v}_i = (\mathbf{v}_{02} \times \mathbf{v}_{012}),\ \mathbf{v}_j = (\mathbf{v}_{01} \times \mathbf{v}_{012})$$

si no

$$\mathbf{v}_i = (\mathbf{v}_{02} \times \mathbf{v}_{021}),\ \mathbf{v}_j = (\mathbf{v}_{01} \times \mathbf{v}_{021})$$

$$\lambda_i = -\frac{\mathbf{v}_0 \cdot \mathbf{v}_i}{\mathbf{v}_{01} \cdot \mathbf{v}_i},\ \lambda_j = -\frac{\mathbf{v}_0 \cdot \mathbf{v}_j}{\mathbf{v}_{02} \cdot \mathbf{v}_j}$$

$0 \le \lambda_i$	$0 \le \lambda_j$	$\lambda_i+\lambda_j \le 1$	$DIST_{OM}(S_{012})=DIST_{OM}(s_{ij})$
$\lambda_i<0$	$0 \le \lambda_j$	$\lambda_i+\lambda_j \le 1$	$DIST_{OM}(S_{012})=DIST_{OM}(S_{02})$
$0 \le \lambda_i$	$\lambda_j<0$	$\lambda_i+\lambda_j \le 1$	$DIST_{OM}(S_{012})=DIST_{OM}(S_{01})$
$0 \le \lambda_i$	$0 \le \lambda_j$	$1<\lambda_i+\lambda_j$	$DIST_{OM}(S_{012})=DIST_{OM}(S_{12})$
$\lambda_i<0$	$\lambda_j<0$	$\lambda_i+\lambda_j \le 1$	$DIST_{OM}(S_{012})=\min(DIST_{OM}(S_{01}), DIST_{OM}(S_{02}))$
$\lambda_i<0$	$0 \le \lambda_j$	$1<\lambda_i+\lambda_j$	$DIST_{OM}(S_{012})=\min(DIST_{OM}(S_{02}), DIST_{OM}(S_{12}))$
$0 \le \lambda_i$	$\lambda_j<0$	$1<\lambda_i+\lambda_j$	$DIST_{OM}(S_{012})=\min(DIST_{OM}(S_{01}), DIST_{OM}(S_{12}))$
$\lambda_i<0$	$\lambda_j<0$	$1<\lambda_i+\lambda_j$	IMPOSIBLE

Nota: Sombreadas se encuentran las condiciones que representan fuera de rango

Distancia a una Tetra-Esfera

Como el origen es un punto del espacio tridimensional, se puede expresar como una combinación lineal en función de los vectores eje $\mathbf{v}_{01}$, $\mathbf{v}_{02}$ y $\mathbf{v}_{03}$ según:

$$\mathbf{0} = \mathbf{v}_0 + \lambda_i \mathbf{v}_{01} + \lambda_j \mathbf{v}_{02} + \lambda_k \mathbf{v}_{03}$$

Multiplicando esta ecuación escalarmente por $\mathbf{v}_{02} \times \mathbf{v}_{03}$, $\mathbf{v}_{01} \times \mathbf{v}_{03}$, y $\mathbf{v}_{01} \times \mathbf{v}_{02}$, se obtiene:

$$\lambda_i = \frac{-\mathbf{v}_0 \cdot (\mathbf{v}_{02} \times \mathbf{v}_{03})}{\mathbf{v}_{01} \cdot (\mathbf{v}_{02} \times \mathbf{v}_{03})}, \lambda_j = \frac{-\mathbf{v}_0 \cdot (\mathbf{v}_{01} \times \mathbf{v}_{03})}{\mathbf{v}_{02} \cdot (\mathbf{v}_{01} \times \mathbf{v}_{03})}, \lambda_k = \frac{-\mathbf{v}_0 \cdot (\mathbf{v}_{01} \times \mathbf{v}_{02})}{\mathbf{v}_{03} \cdot (\mathbf{v}_{01} \times \mathbf{v}_{02})}$$

Si estos tres parámetros son no negativos y su suma es menor o igual que la unidad, el origen se encuentra dentro del tetraedro formado por los centros de las esferas que definen la tetra-esfera. En este caso se debe devolver la intensidad de penetración que corresponde a la mínima de las distancias máximas a las tri-esferas de la tetra-esfera:

$DIST_{OM}(S_{0123})$=-min($DIST_{OM}(S_{012})$,$DIST_{OM}(S_{013})$,$DIST_{OM}(S_{023})$,$DIST_{OM}(S_{123})$)

Si los parámetros se salen del rango establecido, la distancia resultado se obtiene en función de distancias a tri-esferas, devolviéndose cuando así ocurra la intensidad de penetración. En total, pueden presentarse los dieciséis casos posibles que se muestran en la Tabla 4.4.

Tabla 4.4. *Cálculo de la distancia de una tetra-esfera al origen (devolviendo la intensidad de penetración cuando se produce colisión).*

$\lambda_i = \frac{-\mathbf{v}_0 \cdot (\mathbf{v}_{02} \times \mathbf{v}_{03})}{\mathbf{v}_{01} \cdot (\mathbf{v}_{02} \times \mathbf{v}_{03})}, \lambda_j = \frac{-\mathbf{v}_0 \cdot (\mathbf{v}_{01} \times \mathbf{v}_{03})}{\mathbf{v}_{02} \cdot (\mathbf{v}_{01} \times \mathbf{v}_{03})}, \lambda_k = \frac{-\mathbf{v}_0 \cdot (\mathbf{v}_{01} \times \mathbf{v}_{02})}{\mathbf{v}_{03} \cdot (\mathbf{v}_{01} \times \mathbf{v}_{02})}$				
$0 \le \lambda_i$	$0 \le \lambda_j$	$0 \le \lambda_k$	$\lambda_i+\lambda_j+\lambda_k \le 1$	$DIST_O(S_{0123})=DIST_{OM}(S_{0123})$
$\lambda_i<0$	$0 \le \lambda_j$	$0 \le \lambda_k$	$\lambda_i+\lambda_j+\lambda_k \le 1$	$DIST_O(S_{0123})=DIST_O(S_{023})$
$0 \le \lambda_i$	$\lambda_j<0$	$0 \le \lambda_k$	$\lambda_i+\lambda_j+\lambda_k \le 1$	$DIST_O(S_{0123})=DIST_O(S_{013})$
$0 \le \lambda_i$	$0 \le \lambda_j$	$\lambda_k<0$	$\lambda_i+\lambda_j+\lambda_k \le 1$	$DIST_O(S_{0123})=DIST_O(S_{012})$
$0 \le \lambda_i$	$0 \le \lambda_j$	$0 \le \lambda_k$	$1<\lambda_i+\lambda_j+\lambda_k$	$DIST_O(S_{0123})=DIST_O(S_{123})$
$\lambda_i<0$	$\lambda_j<0$	$0 \le \lambda_k$	$\lambda_i+\lambda_j+\lambda_k \le 1$	$DIST_O(S_{0123})=\min(DIST_O(S_{023}),DIST_O(S_{013}))$
$\lambda_i<0$	$0 \le \lambda_j$	$\lambda_k<0$	$\lambda_i+\lambda_j+\lambda_k \le 1$	$DIST_O(S_{0123})=\min(DIST_O(S_{023}),DIST_O(S_{012}))$
$\lambda_i<0$	$0 \le \lambda_j$	$0 \le \lambda_k$	$1<\lambda_i+\lambda_j+\lambda_k$	$DIST_O(S_{0123})=\min(DIST_O(S_{023}),DIST_O(S_{123}))$
$0 \le \lambda_i$	$\lambda_j<0$	$\lambda_k<0$	$\lambda_i+\lambda_j+\lambda_k \le 1$	$DIST_O(S_{0123})=\min(DIST_O(S_{013}),DIST_O(S_{012}))$
$0 \le \lambda_i$	$\lambda_j<0$	$0 \le \lambda_k$	$1<\lambda_i+\lambda_j+\lambda_k$	$DIST_O(S_{0123})=\min(DIST_O(S_{013}),DIST_O(S_{123}))$
$0 \le \lambda_i$	$0 \le \lambda_j$	$\lambda_k<0$	$1<\lambda_i+\lambda_j+\lambda_k$	$DIST_O(S_{0123})=\min(DIST_O(S_{012}),DIST_O(S_{123}))$
$\lambda_i<0$	$\lambda_j<0$	$\lambda_k<0$	$\lambda_i+\lambda_j+\lambda_k \le 1$	$DIST_O(S_{0123})=\min(DIST_O(S_{023}),DIST_O(S_{013}),DIST_O(S_{012}))$
$\lambda_i<0$	$\lambda_j<0$	$0 \le \lambda_k$	$1<\lambda_i+\lambda_j+\lambda_k$	$DIST_O(S_{0123})=\min(DIST_O(S_{023}),DIST_O(S_{013}),DIST_O(S_{123}))$
$\lambda_i<0$	$0 \le \lambda_j$	$\lambda_k<0$	$1<\lambda_i+\lambda_j+\lambda_k$	$DIST_O(S_{0123})=\min(DIST_O(S_{023}),DIST_O(S_{012}),DIST_O(S_{123}))$
$0 \le \lambda_i$	$\lambda_j<0$	$\lambda_k<0$	$1<\lambda_i+\lambda_j+\lambda_k$	$DIST_O(S_{0123})=\min(DIST_O(S_{013}),DIST_O(S_{012}),DIST_O(S_{123}))$
$\lambda_i<0$	$\lambda_j<0$	$\lambda_k<0$	$1<\lambda_i+\lambda_j+\lambda_k$	IMPOSIBLE

Nota: Sombreadas se encuentran las condiciones que representan fuera de rango

Estos cuatro algoritmos básicos, que permiten calcular la distancia al origen de las cuatro clases de poli-esferas básicas, se han implementado en un programa en C y ejecutado en un ordenador Pentium a 160MHz sobre cien elementos generados aleatoriamente, obteniéndose los tiempos de ejecución resumidos en la Tabla 4.5.

Tabla 4.5. *Evaluación de tiempos para el cálculo de la distancia de poli-esferas básicas al origen.*

Poli-esfera	Tiempo Mínimo	Tiempo Medio	Tiempo Máximo
Esfera	0.0	0.001	0.001
Bi-esfera	0.014	0.015	0.016
Tri-esfera	0.065	0.065	0.066
Tetra-esfera	0.182	0.183	0.188

Nota: Los tiempos se muestran en milisegundos

Es de destacar lo reducido que son estos tiempos, ya que muchas veces se utilizarán estos elementos básicos para modelar objetos. Por ejemplo, para planificar los movimientos de un vehículo autoguiado, suele ser aconsejable envolver el mismo con un modelo sencillo que, además de servir como zona de seguridad, permita el cálculo de distancias rápido para poder aplicar en tiempo real y comprobar los datos previstos por el planificador off-line con los resultados medidos con sensores de distancia y actuar en consecuencia. Ello hace que muchas veces se modele el AGV como una esfera y los obstáculos como elementos de pocas esferas, pudiéndose obtener cálculos de distancias entre ellos en pocos milisegundos.

4.3.2. Cálculo de Distancias entre Poli-Esferas

Se define la *Diferencia de Minkowski Esférica*, a partir de dos poli-esferas S'_{0-m} y S^*_{0-k}, con vértices esféricos $\mathbf{S'}=\{s'_0,\ldots,s'_m\}$ y $\mathbf{S^*}=\{s^*_0,\ldots,s^*_k\}$ respectivamente, entre las que se quiere calcular la distancia, como aquella poli-esfera definida por los vértices esféricos $\mathbf{S}=\{s_0,\ldots,s_n\}$ con n=(m+1)×(k+1)-1 según:

$$s_{i\times(k+1)+j} = s'_i \sim s^*_j, i = 0,\ldots,m, j = 0,\ldots,k$$

siendo el operador binario Diferencia de Minkowski Esférica (~) para dos esferas como la esfera resultante de realizar la diferencia de centros y suma de radios:

$$s_2 \sim s_1 = s{:}s.c = s_2.c - s_1.c, s.r = s_1.r + s_2.r$$

De esta forma, el cálculo de la distancia entre las dos poli-esferas se calcula a partir de la distancia de la poli-esfera diferencia de Minkowski al origen.

Se define la *función soporte esférica* $h_S(\eta)$ de una poli-esfera $S_{0\text{-}n}$ definida a partir del conjunto finito de esferas $\mathbf{S}=\{s_0,...,s_n\}$ en una dirección $\eta\in\Re^3$ como el máximo resultado de realizar el producto escalar del centro de una esfera de **S** por el vector η y sumarle el radio de la esfera:

$$h_S(\eta) = \max_{s_i \in \mathbf{S}} \left\{ \eta \cdot s_i.c + s_i.r \right\}, \eta \in \Re^3$$

Una *solución soporte esférica* de una poli-esfera para una dirección $\eta\in\Re^3$, denominada $s_S(\eta)$, es aquella esfera vértice de la poli-esfera que devuelve la función soporte esférica (máximo producto escalar de su centro con el vector director más su radio), es decir la esfera de la poli-esfera más lejana en la dirección dada:

$$s_S(\eta) \in \mathbf{S}: \eta \cdot s_S(\eta).c + s_S(\eta).r = h_S(\eta)$$

Tanto la función soporte esférica como la solución soporte esférica se pueden definir mediante los vértices esféricos de la poli-esfera.

El algoritmo de cálculo de distancia entre las poli-esferas $S_{0\text{-}n}$ y S'_{0-m}, con vértices esféricos $\mathbf{S}=\{s_0,...,s_n\}$ y $\mathbf{S'}=\{s'_0,...,s'_m\}$ respectivamente, basado en la diferencia de Minkowski esférica y la función soporte esférica, utiliza los algoritmos básicos de cálculo de distancias a poli-esferas de hasta cuatro vértices esféricos de la sección anterior.

La forma de actuar del algoritmo es iterativa, aprovechando para cada paso la dirección "probablemente más adecuada" para calcular la esfera más cercana y considerar sólo los vértices esféricos necesarios para definirla. La dirección probablemente más adecuada viene definida por el centro de la esfera más cercana al origen encontrada hasta ahora.

El algoritmo devuelve la intensidad de penetración cuando hay colisión, ya que los algoritmos básicos la calculan. Además, genera los elementos necesarios para definir completamente las dos esferas más próximas cuando la solución no es única.

Básicamente, el algoritmo tiene los pasos que se muestran a continuación:

1. *Obtener* **S** *como la diferencia de Minkowski esférica entre* **S'** *y* $\mathbf{S}^*$ *de una forma ordenada, para que sea posible recuperar, a partir una esfera* s *de* **S**, *el par de esferas* s' *de* **S'** *y* s^* *de* $\mathbf{S}^*$ *que la han generado. Si* **S** *está vacío, terminar con mensaje al respecto; si sólo tiene una esfera, terminar devolviendo la distancia* $d_i=\|\mathbf{s}\|_m$.
2. *Para i=0, inicializar un conjunto* $\boldsymbol{\Sigma}_i$ *con la esfera* σ_i *de* **S** *más cercana al origen; inicializar* $\eta_i=-\sigma_i.c$ *y* $g_i=-\sigma_i.c\cdot\sigma_i.c+\sigma_i.r$*; inicializar* s_i *con la esfera de* **S** *solución soporte esférica de* η_i*:* $s_i=s_S(\eta_i)$ *y eliminarla de* **S***; inicializar* $h_i=-\eta_i\cdot s_i.c+s_i.r$*; inicializar* d_i *con* $\|\sigma_i\|_m$*; eliminar de* **S** *la esfera* s_i *y toda esfera* s_j *tal que* $\eta_i\cdot s_j.c+s_j.r<g_i$
3. *Si* $h_i<g_i$ ***terminar*** *con:*
   - σ_i *la esfera más cercana al origen,* $\sigma_i.c$ *el vector de colisión,* d_i *la distancia entre las poli-esferas*
   - *Añadir a* $\boldsymbol{\Sigma}_i$ *toda esfera* s_j *de* **S** *tal que* $\eta_i\cdot s_j.c+s_j.r=g_i$
   - *Recuperar todas las esferas* s' *de* **S'** *y* s^* *de* $\mathbf{S}^*$ *que definen las esferas de* $\boldsymbol{\Sigma}_i$
4. *Generar* $\boldsymbol{\Sigma}_{i+1}=\boldsymbol{\Sigma}_i\cup\{s_i\}$
5. *Calcular la esfera* σ_{i+1} *de la poli-esfera definida mediante* $\boldsymbol{\Sigma}_{i+1}$ *más cercana al origen mediante los algoritmos básicos de cálculo de distancias al origen; actualizar* $d_{i+1}=\|\sigma_{i+1}\|_m$, $\eta_{i+1}=-\sigma_{i+1}.c$ *y* $g_{i+1}=\eta_{i+1}\cdot\sigma_i.c+\sigma_i.r$
6. *Eliminar de* $\boldsymbol{\Sigma}_{i+1}$ *los vértices esféricos no necesarios para definir* σ_{i+1}
7. *Si* $\boldsymbol{\Sigma}_{i+1}$ *tiene cuatro vértices* ***terminar*** *con:*
   - *Mensaje de colisión con* σ_{i+1} *la esfera más cercana al origen,* $\sigma_{i+1}.c$ *el vector de colisión y* d_{i+1} *la intensidad de penetración*
8. *Si* **S** *está vacío* ***terminar*** *con:*
   - σ_{i+1} *la esfera más cercana al origen,* $\sigma_{i+1}.c$ *el vector de colisión,* d_{i+1} *la distancia entre las poli-esferas*
   - *Añadir a* $\boldsymbol{\Sigma}_{i+1}$ *toda esfera* s_j *de* **S** *tal que* $\eta_{i+1}\cdot s_j.c+s_j.r=g_{i+1}$
   - *Recuperar todas las esferas* s' *de* **S'** *y* s^* *de* $\mathbf{S}^*$ *que definen las esferas de* $\boldsymbol{\Sigma}_{i+1}$
9. *Calcular* s_{i+1} *como la esfera de* **S** *solución soporte esférica de* η_{i+1}*:* $s_{i+1}=s_S(\eta_{i+1})$*; actualizar* $h_{i+1}=\eta_{i+1}\cdot s_{i+1}.c+s_{i+1}.r$*; eliminar de* **S** *la esfera* s_{i+1} *y toda esfera* s_j *tal que* $\eta_{i+1}\cdot s_j.c+s_j.r<g_{i+1}$
10. *Incrementar i e ir al paso 3*

La terminación del algoritmo en los pasos 3 y 8 se realiza con dos conjuntos de esferas de **S'** y $\mathbf{S}^*$ que representan los elementos necesarios para definir completamente las esferas más proximas cuando la solución no es única. La terminación en el paso 7 se realiza devolviendo la intensidad de penetración como una distancia de valor negativo. La Figura 4.2 ilustra el algoritmo.

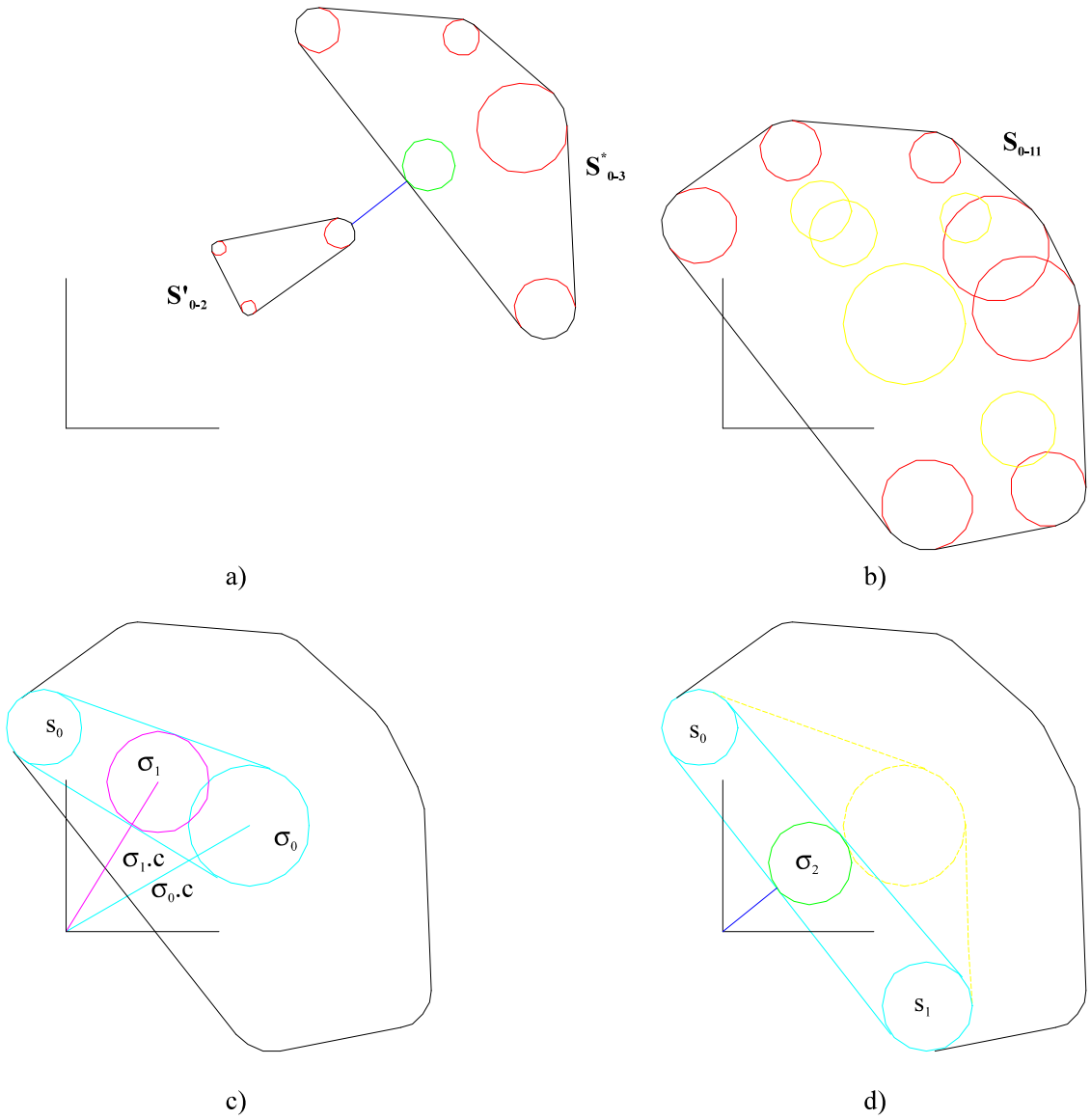


Figura 4.2. Pasos del Algoritmo del Cálculo de Distancias entre Poli-Esferas. *El primer paso para calcular la distancia entre las poliesferas* $S'_{0\text{-}2}$ *y* $S^{*}_{0\text{-}3}$ *(figura a) es calcular la poli-esfera diferencia de Minkowski esférica,* $S_{0\text{-}11}$*, (figura b), para encontrar la distancia al origen. Después, (figura c) se calcula la esfera* σ_0 *más cercana al origen y* s_0 *como la solución soporte del vector* $\sigma_0.c$*, para definir una bi-esfera de la que se calcula* σ_1 *como la esfera más cercana al origen. Se repite el proceso (figura d) para determinar* s_1 *como la solución soporte de* $\sigma_1.c$ *y* σ_2 *como la esfera más cercana al origen de la tri-esfera definida mediante* σ_0*,* s_0*,* s_1*, eliminando* σ_0*, ya que no resulta necesaria para obtener* σ_2*. Como la función soporte respecto a* $\sigma_2.c$ *no se mejora, el algoritmo termina devolviendo la distancia, el vector de colisión y recuperando los elementos originales que producen esta distancia.*

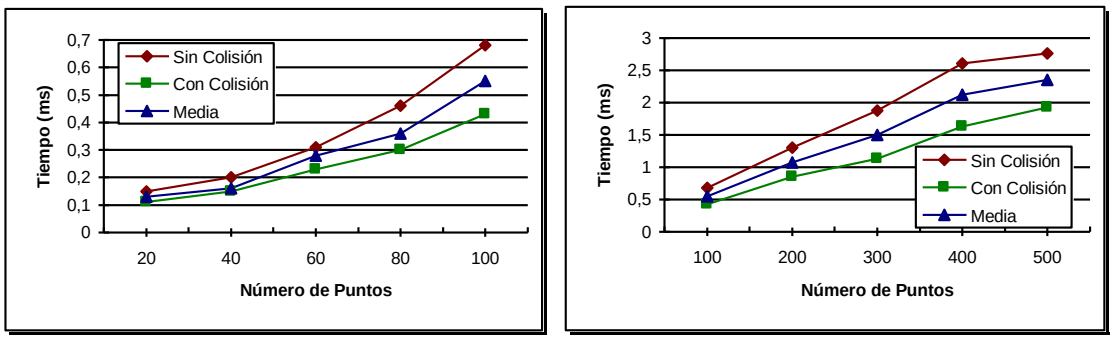


Figura 4.3. Coste Computacional del Algoritmo del Cálculo de Distancias entre Poli-Esferas. *El coste computacional es aproximadamente lineal al número de puntos total utilizado para definir la poli-esfera diferencia de Minkowski esférica. Además se puede apreciar que el coste es mayor para determinar la distancia cuando no hay colisión que cuando hay colisión. El algoritmo es tan rápido (menos de 3 milisegundos para obtener la distancia al origen de una poli-esfera definida con 500 vértices esféricos) por su rápida convergencia: no suele requerir más de tres o cuatro iteraciones.*

Este algoritmo se ha implementado en un programa en C y ejecutado en un Pentium a 160MHz, usando una poli-esfera de hasta 500 vértices esféricos generados aleatoriamente. En la Figura 4.3. se muestran los resultados de coste computacional, donde se puede ver que éste es lineal al número total de vértices, produciendo resultados muy rápidos (menos de 0.7 (3) milisegundos para calcular la distancia al origen de una poli-esfera definida mediante 100 (500) vértices esféricos. La rapidez de este algoritmo se basa en la convergencia tan rápida que tiene, necesitando normalmente sólo 3 ó 4 iteraciones para determinar la distancia mínima. Además, en la gráfica se puede observar que resulta más costoso obtener la distancia cuando no hay colisión que determinar la intensidad de penetración cuando se produce colisión.

4.3.3. Cálculo de Distancias con Reducción de Poli-Esferas

A partir del ejemplo mostrado en la sección anterior (Figura 4.2), se puede deducir que al aplicar la diferencia de Minkowski esférica entre dos poli-esferas, en general aparecen esferas redundantes, no necesarias para definir el volumen ocupado por la poli-esfera resultante, es decir, que la nueva poli-esfera es reducible. Esto puede sugerir la conveniencia de realizar un proceso de reducción de la poli-esfera generada por la diferencia de Minkowski esférica antes de calcular su distancia al origen.

Recordando (Sección 3.3.3) que el resultado de reducir una poli-esfera $\boldsymbol{\Sigma}_{0\text{-}n}$ definida a partir de un conjunto de esferas $\boldsymbol{\Sigma}=\{\sigma_0,...,\sigma_n\}$ es una poli-esfera $S_{0\text{-}\varphi}$ que ocupa el mismo volumen, definida a partir de un conjunto de esferas $\mathbf{S}=\{s_0,...,s_\varphi\}$, subconjunto de $\boldsymbol{\Sigma}$, con φ=rango([Φ])<4 según:

$$S_{0-\varphi} = \left\{ s \in \Omega : s = s_0 + \sum_{i=1}^{\varphi} \lambda_i \mathbf{s}_{0i}, \mathbf{s}_{0i} = s_i - s_0, s_i \in \mathbf{S}, \sum_{i=1}^{\varphi} \alpha_{ij} \lambda_i \geq \beta_j, j = 1,\ldots,m \right\}$$

Esta nueva poli-esfera irreducible sólo requiere φ parámetros λ_i para relacionar sus φ+1 esferas, frente a los n parámetros λ_i necesarios anteriormente para relacionar sus n+1 esferas. Además, el número de ecuaciones lineales que delimitan los rangos de los parámetros λ_i se ha reducido de n+1 a m con m<n.

Como ambas poli-esferas representan el mismo volumen en el espacio tridimensional, la distancia al origen de $\Sigma_{0\text{-}n}$ y $S_{0\text{-}\varphi}$ es la misma.

Como φ es menor que n, se puede deducir erróneamente que el coste computacional de calcular la distancia al origen de $S_{0\text{-}\varphi}$ es menor que el de calcular la distancia al origen de la poli-esfera $\Sigma_{0\text{-}n}$. Sin embargo, la distancia al origen de $S_{0\text{-}\varphi}$ no depende sólo de las esferas $\{s_0,\ldots,s_\varphi\}$, sino también de las m ecuaciones $\sum_{i=1}^{\varphi} \alpha_{ij} \lambda_i \geq \beta_j$, que limitan el rango de los parámetros, con lo que los costes computacionales son similares. Si además se añade el coste computacional de realizar una reducción basada en la diagonalización de la matriz, resulta que reducir la poli-esfera resultado de la diferencia de Minkowski esférica no presenta mejoras frente al algoritmo utilizado en la sección anterior. Esto es debido a que el algoritmo anterior presenta una convergencia muy rápida: en cuatro o cinco iteraciones ya ha encontrado la esfera solución. Además, en cada paso elimina un número elevado de esferas a considerar.

4.4. Cálculo de Distancias con Esferoides

La utilización de modelos esféricos no lineales, como esferoides cuadráticas y/o cúbicas, hace que se puedan presentar dos casos diferentes para el cálculo de distancias, la distancia de una esferoide a una poli-esfera y la distancia entre dos esferoides. Si bien para ambos casos se podría buscar una solución analítica, bien pretendiendo determinar el vector esférico perpendicular a las superficies externas de la esferoide y la poli-esfera, o determinando los parámetros que hacen la derivada nula, estos métodos resultan muy laboriosos, por lo que se ha optado por el planteamiento de un problema de minimización multi-dimensional o bi-dimensional.

Calcular la distancia entre una esferoide S^{g}_{0-g} de grado g y una poli-esfera $S_{0\text{-}n}$ con n vértices esféricos es un problema de optimización multi-dimensional:

$$\min_{\substack{0\le\lambda\le 1,\\ 0\le\lambda_i\le 1,\\ i=1,\ldots,n}} \mathrm{DIST}(S^{g}_{0-g}(\lambda), S_{0-n}(\lambda_1,,\lambda_n))$$

Para resolver este caso se puede utilizar como técnica de minimización el método Downhill Simplex que ya se utilizó, en otro contexto, en el capítulo anterior. En principio, las derivadas no están disponibles, ya que son difíciles de obtener.

La técnica Downhill Simplex es un método de optimización local, por lo que el resultado obtenido puede ser un mínimo local pero no global. Sin embargo, al aplicar esta técnica al cálculo de distancias entre esferoides y poli-esferas, se conoce el máximo número de mínimos locales que se pueden presentar, lo que simplifica mucho el problema. De hecho, la distancia entre una esferoide cuadrática y una esfera puede presentar hasta dos mínimos, aumentando el número de mínimos locales progresivamente según aumente el número de vértices esféricos de la poliesfera, dando un total de n+1 mínimos locales al calcular la distancia entre una esferoide cuadrática y una poli-esfera con n vértices esféricos. De la misma forma, la distancia entre una esferoide cúbica y una poli-esfera con n vértices esféricos puede presentar como máximo n+2 mínimos locales. En el caso del cálculo de la distancia entre una esferoide spline con m esferas de control y una poli-esfera con n vértices esféricos, cada uno de los m-1 segmentos es una esferoide cúbica que podría presentar n+2 mínimos locales, dando un total de (m-1)(n+2) mínimos.

Con una implementación del algoritmo Downhill Simplex en un programa en C y ejecutado en un Pentium a 160MHz, se ha aplicado al cálculo de distancias entre las esferoides cuadrática y cúbica envolventes del robot IRB L6 mostradas en la Figura 3.11a del capítulo anterior a una poli-esfera. En la Tabla 4.6 se muestra el tiempo medio de ejecución sobre 500 pruebas realizadas con el robot en una configuración aleatoria y una poli-esfera de hasta 8 vértices esféricos generados aleatoriamente. La precisión con que se obtiene la distancia es 1mm.

Calcular la distancia entre dos esferoides S^{g}_{0-g} y S^{h}_{0-h} de grados g y h es un problema de optimización bi-variable al que se puede aplicar de nuevo el método Downhill Simplex:

$$\min_{0\le\lambda_1,\lambda_2\le 1} \mathrm{DIST}(S^{g}_{0-g}(\lambda_1), S^{h}_{0-h}(\lambda_2))$$

El número máximo de mínimos locales que se pueden presentar entre dos esferoides cuadráticas es cuatro, entre dos cúbicas es nueve mientras que entre una cuadrática y una cúbica es seis. Para una esferoide spline con n esferas de control y una esferoide cuadrática es 6(n-1), con una esferoide cúbica es 9(n-1) y con otra esferoide spline con m esferas de control es 9(n-1)(m-1).

Tabla 4.6. *Evaluación de tiempos para el cálculo de la distancia de la esferoide cúbica envolvente de un robot y una poli-esfera.*

Número de Vértices Esféricos de la Poli-Esfera	1	2	3	4	5	6	7	8
Tiempo Medio de Cálculo para la Esferoide Cuadrática	0.4	0.5	0.9	1.5	1.8	2.5	3.0	3.9
Tiempo Medio de Cálculo para la Esferoide Cúbica	0.6	0.8	1.4	2.2	2.7	3.8	4.6	5.9

Nota: Los tiempos se muestran en milisegundos

Aplicando la implementación del algoritmo Downhill Simplex, ejecutado en un Pentium a 160MHz, al cálculo de distancias entre las esferoides cuadráticas y cúbicas envolventes de los robots IRB L6 e IRB 1500 mostradas en las Figuras 3.11a y 3.11b del capítulo anterior, el tiempo medio de ejecución sobre 500 pruebas realizadas con ambos robots en configuraciones aleatorias ha sido de 2.6ms (entre dos esferoides cuadráticas), 4ms (entre una esferoide cuadrática y una cúbica) y 6ms (entre dos esferoides cúbicas). La precisión con que se obtiene la distancia es 1mm.

4.5. Detección de Colisiones con el Modelo Jerárquico

La implementación de un módulo eficiente de detección de colisiones exige el establecimiento de un compromiso entre la precisión del modelo utilizado y la eficiencia temporal que se exija al módulo de cálculo de distancias. Para una aplicación donde el tiempo no sea un factor primordial, como la verificación en la fase de simulación de un programa robot off-line, se podrían calcular tantas configuraciones intermedias por las que se mueve el robot como se deseen y para cada una de ellas calcular la distancia del robot a los obstáculos de su entorno utilizando modelos exactos. La precisión con que se trabaja será función de la frecuencia con que se realice la detección de colisiones (una cada segundo, cada décima o centésima de segundo, etcétera) y la velocidad de movimiento. Por ejemplo, para una velocidad típica de 1m/s y una frecuencia de detección de colisiones de 0.1s, se realizan diez detecciones de colisión para un movimiento de la herramienta de un metro, es decir, una cada 10cm de movimiento de la herramienta. Evidentemente, aunque dos configuraciones del robot no produzcan colisiones, no se asegura que configuraciones intermedias no las produzcan, por lo que la frecuencia debe ser elevada para trabajar con una precisión aceptable. Además de tener que realizarse muchas verificaciones de la detección de colisiones, al

utilizarse modelos exactos del robot y los obstáculos, el tiempo de cálculo será muy elevado para cada detección de colisiones.

Por otra parte, la detección de colisiones tampoco debe ser una aplicación completamente en tiempo real, comprobando si la configuración en que se encuentra el robot en este preciso instante colisiona o no con los obstáculos de su entorno: un mensaje del tipo *"el robot está colisionando en este instante con el obstáculo x"* seguramente producirá resultados desastrosos.

Por tanto, la detección de colisiones debe resultar tan rápida que resulte transparente para poder aplicarse on-line, dando la impresión de ser en tiempo real, pero trabajando de forma anticipativa, para asegurar que el siguiente movimiento se puede realizar porque no produce colisión. De esta forma se intenta que el robot esté en movimiento continuo sin tiempos de espera, pero la detección de colisiones se esté aplicando a configuraciones futuras por las que el robot va a pasar.

La frecuencia con que se trabaja debe determinarla el propio detector de colisiones, de forma que si las distancias son pequeñas y no va a tener suficiente tiempo para detectar la posible colisión en la siguiente configuración, reduzca la velocidad de movimiento del robot para disponer de más tiempo. Esta filosofía es la que utilizamos en la vida cotidiana cuando, por ejemplo, conducimos un vehículo: según nos aproximamos a otro vehículo reducimos la velocidad para disponer de un tiempo de reacción suficiente para poder evitar un posible choque. Un técnico programador de robots realiza la misma función cuando prueba un programa, reduciendo o aumentando la velocidad de ejecución en función de la distancia a los objetos. Cuando el robot está cercano a las piezas que va a manipular o sobre las que va a realizar cierta operación, el programador reduce la velocidad de ejecución y realiza una comprobación de posibles colisiones visualmente. Si cree que va a producirse una colisión durante un movimiento, detiene el programa antes de que se produzca y lo corrige. Por contra, cuando las distancias son grandes, el operario puede aumentar la velocidad de ejecución para no ralentizar el proceso de prueba del programa.

Disponer de un detector de colisiones suficientemente rápido para aplicar on-line, no sólo reducirá los tiempos de verificación de un programa off-line en la fase de simulación, sino que podrá utilizarse para la programación on-line del robot. Un programador suele introducir las posiciones de movimiento mediante el guiado del brazo con una unidad de programación que normalmente dispone de un *teach-pendant* para mover el robot. Para introducir una instrucción de movimiento del robot a partir de la configuración actual ya programada, el programador indica, mediante parámetros relativos o absolutos, otra configuración para programarla, o realiza movimientos del robot por articulaciones con el *teach-pendant* y alcanza la nueva configuración. El camino intermedio por el que ha pasado el robot no se almacena, y en su lugar sólo se guardan las dos configuraciones extremas. Durante la ejecución de esta instrucción, se realizará una interpolación (en el espacio de articulaciones o en el espacio cartesiando según el tipo de movimiento seleccionado), produciéndose el movimiento del robot. Un detector de colisiones on-line debe verificar si la instrucción introducida producirá una colisión en ejecución, y en su caso dar un mensaje para impedir esta instrucción de movimiento. Evidentemente, este

proceso puede realizarse sólo para obstáculos conocidos y fijos del entorno del robot. Cuando el obstáculo es desconocido, o puede cabiar de forma y posición (por ejemplo, otro robot o un AGV), se requiere una simulación off-line de todos los elementos que intervengan para asegurar la inexistencia de colisiones.

Una primera aproximación para acelerar el proceso de la detección de colisiones se muestra en [TM+92] y consiste en considerar una estructura jerárquica extendida con diferentes grados de precisión utilizando modelos envolventes esféricos para determinar si hay colisión entre un sistema robotizado y un obstáculo. El procedimiento de detección de colisiones comienza testeando modelos globales del sistema robotizado completo (nivel célula) con la menor precisión disponible (por ejemplo, esferas). Mientras no se detecte colisión se sigue manteniendo este modelo, ya que permite el cálculo de distancias de forma muy rápida. Si se detecta una colisión, se deben considerar modelos globales con menor precisión (por ejemplo, bi-esferas). Cuando en un nivel se ha producido colisión para el grado de mayor precisión disponible, se desciende de nivel, para considerar, por ejemplo, brazos-robot en el nivel sistema. De nuevo se recorre de menor a mayor precisión este nivel (esfera, bi-esfera, tri-esfera, ...). Si persiste la colisión, se repite el proceso para el nivel elemento, comparándose todos los elementos de un sistema con todos los elementos del otro sistema. El procedimiento termina cuando no se detecte colisión o cuando los modelos de mayor precisión disponibles a nivel elemento devuelvan colisión. La ventaja de utilizar diferentes grados de precisión se muestra en [T&B92] para una célula multi-robot, comparando la detección de colisiones de 16.384 configuraciones uniformemente distribuidas, utilizando sólo el máximo grado de precisión frente a utilizar varios grados de precisión, alcanzando reducciones medias de más del 90%.

Este planteamiento resulta muy útil, pero sólo para determinar que no existe colisión dada una configuración fija de un robot, ya que en casos más generales puede acarrear la realización de una gran cantidad de cálculos que sean innecesarios. Por ejemplo, para un caso de colisión, se habrá recorrido toda la estructura en todos sus niveles y grados de precisión hasta detectarlo.

Una posible solución para asegurar la existencia de colisión en una configuración del robot, consiste en disponer una estructura jerárquica (extendida con varios grados de precisión o no) paralela con los modelos envueltos. Es evidente que si dos modelos envueltos colisionan se asegura la existencia de colisiones.

Otra posible solución, que no consume tanta memoria (no necesita una estructura jerárquica paralela) ni tiempo de cálculo (no requiere la distancia entre objetos envolvente y envueltos), es una extensión de las mejoras de actualización de distancias y jerarquías estáticas y consiste en acotar el error realizado por cada modelo envolvente. Para ello, en la estructura jerárquica se puede guardar junto a cada modelo una cota máxima del error que se comete al sustituir el objeto real por su correspondiente modelo. Representando por la dupla *nivel-grado* (N,g) al grado de precisión g del nivel N, llámese $E^{M}_{(N,g)}$ al mayor error cometido y $E^{m}_{(N,g)}$ al menor error cometido en el modelado del sistema robotizado completo para ese nivel-grado (máxima y mínima distancia entre los modelos envolventes y sus correspondientes modelos exactos). Si para este nivel-grado

existe colisión, el procedimiento de cálculo de distancias devolverá la intensidad de penetración, $d^c_{(N,g)}$. Es evidente que cuando $|d^c_{(N,g)}|>E^M_{(N,g)}$ existe colisión, mientras que cuando se cumpla $|d^c_{(N,g)}|<E^M_{(N,g)}$ no existe colisión, por lo que resulta innecesario recorrer el resto de la estructura jerárquica en ambos casos. Si no se cumple ninguna de estas condiciones hay que continuar la búsqueda en el árbol hasta que se encuentre un nivel-grado que cumpla alguna de estas condiciones o se llegue al último nivel-grado (las hojas del árbol con el modelo exacto del robot), donde se determina si hay o no hay colisión.

Por otra parte, si el procedimiento se quiere aplicar a la detección de colisiones de un camino fijado para un robot se debería aplicar el procedimiento anterior para cada configuración alcanzada del camino. Supóngase que para una de estas configuraciones el procedimiento devuelve colisión en todos los grados de precisión de los niveles sistema y sub-sistema, pero no para el nivel elemento. Un mínimo movimiento del robot implicaría calcular nuevamente todas las distancias en los niveles sistema y sub-sistema, aunque el movimiento haya sido tan pequeño que resulte imposible para dichos modelos haber salido fuera de la zona de colisión.

Sería conveniente poder determinar cuál es el movimiento mínimo que debe haber realizado el robot para que modelos que hayan producido colisión puedan haber salido, en el mejor de los casos, de la situación de colisión (hayan salido realmente o no) y vuelvan a tenerse en cuenta dentro de la estructura jerárquica.

Este nuevo planteamiento consiste en detectar colisiones para la trayectoria de un robot en ciertos instantes de tiempo, $\{t_0,..., t_n\}$. Representando la configuración en que se encuentra un robot como un vector $\mathbf{q}$ de las variables de articulación del robot, la trayectoria vendrá definida mediante una sucesión de configuraciones, $\{\mathbf{q}_0,...,\mathbf{q}_n\}$, generadas por el módulo cinemático del sistema a partir de una configuración inicial $\mathbf{q}_0$, una configuración final $\mathbf{q}_n$ y una velocidad de movimiento, realizando una interpolación lineal en el espacio de articulaciones, de forma que en el tiempo $t=t_i$ el robot se encuentra en la configuración $\mathbf{q}_i$. Para cada intervalo de tiempo (t_i,t_{i+1}), de duración $\Delta t_i=t_{i+1}-t_i$, el robot habrá realizado cierto movimiento $\Delta\mathbf{q}_i$ a una velocidad predefinida. Como existe un límite de velocidad, la velocidad programada de movimiento del robot, el desplazamiento realizado será acotado. Se define $\Delta a(\mathbf{q}_i)$ como el máximo recorrido posible realizado por un punto del robot en el intervalo de tiempo (t_i,t_{i+1}) al moverse de la configuración $\mathbf{q}_i$ a la $\mathbf{q}_{i+1}$. Este valor es fácil de obtener sabiendo para cada elemento del robot cuál es el punto más alejado del eje de rotación respecto al que se realiza el giro en las articulaciones de revolución. Para articulaciones prismáticas es igual al desplazamiento realizado.

Supóngase que en un cierto instante t_i, se ha detectado colisión para el nivel-grado (N,g). El cálculo de distancia habrá devuelto la intensidad de penetración $d^c_{(N,g)}$ como el menor valor (el más negativo) de las distancias obtenidas entre los modelos de este nivel y el obstáculo. Sin embargo, para el siguiente grado de precisión considerado, g+1, no se produce colisión, obteniéndose la distancia al obstáculo $d^c_{(N,g+1)}$ como la menor distancia entre los nuevos modelos de este nivel y el obstáculo. Si no existe el grado de precisión

g+1 se plantea de forma similar un cambio al nivel N+1 con la menor precisión disponible (g=1). Las siguientes cuestiones se pueden plantear: En el instante de tiempo $t=t_{i+1}$, ¿es necesario detectar colisiones?. Si lo es, ¿a partir de qué nivel y grado de precisión se empieza la comprobación?. Dos respuestas posibles son las siguientes:

1. Si $\Delta a(\mathbf{q}_i)<d^c_{(N,g+1)}$ resulta imposible que el modelo envolvente del nivel-grado (N,g+1) haya colisionado en este movimiento. Por tanto, no es necesario comprobar colisiones para $t=t_{i+1}$. Evidentemente, no es necesario comprobar colisiones en el nivel-grado (N,g+1) hasta el instante de tiempo $t=t_{i+j}$ tal que

$$\sum_{k=i}^{i+j} \Delta a(\mathbf{q}_k) \geq d_{N,g+1}$$

   con lo que se evitan cálculos en los instantes de tiempo $t=t_{i+1}$,..., t_{i+j-1}.

2. Si $\Delta a(\mathbf{q}_i)<|d^c_{(N,g)}|$ resulta imposible que este modelo del robot haya evitado la colisión en este movimiento. El procedimiento de detección de colisiones para el grado-nivel (N,g) devolverá colisión, por lo que resulta innecesario su comprobación. Por tanto, no es necesario comprobar colisiones para el nivel-grado (N,g) hasta el instante de tiempo $t=t_{i+j}$ tal que

$$\sum_{k=i}^{i+j} \Delta a(\mathbf{q}_k) \geq \left|d^c_{N,g}\right|$$

   con lo que se evitan cálculos en los instantes de tiempo $t=t_{i+1}$,..., t_{i+j-1}.

Hasta ahora se ha considerado un árbol jerárquico como el considerado en [TM+92], donde se desciende de un nivel al nivel inferior con una única descomposición posible y de forma horizontal: si los modelos a nivel sistema de dos robot colisionan, se descende a nivel elemento y se comprueba la existencia de colisión entre todos los pares elemento-elemento formados por un elemento de cada robot. Si dos elementos colisionan, se desciende a nivel primitivas y se realizan las comprobaciones involucrando a todas las primitivas de los dos robots.

La novedad de la técnica que se presenta a continuación es que también se puede aplicar al árbol jerárquico dinámico extendido de la Sección 3.4 que permitía que un nodo se descomponga de diferentes maneras en paralelo. Además, la estructura permite descomponer sólo un nodo de forma vertical, sin tener que descomponer todos los nodos de un cierto nivel para pasar al nivel inferior. Como el árbol guarda para cada nodo diferentes grados de precisión con modelos esféricos envolventes, se utilizarán los métodos del cálculo de distancia explicados en las Secciones 4.3 y 4.4.

Por otra parte, la detección de colisiones se realiza entre ramas de un mismo árbol, es decir, que si el árbol representa una célula, hay un sub-árbol por robot, otro por cada obstáculo, otro para un AGV, etcétera. También puede aparecer un nivel intermedio que agrupe dos o más de estos sistemas, para permitir, por ejemplo, que el módulo de detección de colisiones de un robot vea inicialmente al resto del árbol como a un único

obstáculo. Este planteamiento convierte la detección de colisiones en un problema de detección de auto-colisiones (*self-collision detection problem*).

Para no tener que replicar la estructura jerárquica para cada dispositivo, se dispone de una base de datos común de la célula que guarda el árbol completo. Cada sistema controlable tiene un módulo cinemático que actualiza la configuración en que se encuentra ese sistema a partir del programa que gobierna sus movimientos. Esta actualización se realiza sólo sobre los nodos del nivel elemento, propagándose automáticamente al resto de nodos del árbol. El árbol se ha inicializado con los modelos envolventes de los nodos rígidos ya calculados off-line en todos sus grados de precisión, mientras que para aquellos nodos articulados se dispone de un módulo que calcula la envolvente según el grado de precisión requerido.

Para la célula debe haber un coordinador que se encarga de determinar cuál de los sistemas controlables debe realizar una detección de colisiones, así como de cuándo los módulos cinemáticos tienen que actualizar el árbol y qué movimientos se pueden realizar en cada momento.

Para cada sistema controlable (robots, AGVs, ...), su detector de colisiones se compondrá de tantos módulos como sistemas con los que se deba realizar la detección de colisiones. No se deben considerar aquellos obstáculos que se encuentren fuera de su zona de alcance, pero sí que es posible que exista un módulo para detectar la auto-colisón de ese sistema. Un gestor global de este sistema se encarga de decidir cuándo se debe arrancar cada uno de los módulos según el resultado de la última comprobación, asegurándose que sólo se ejecuten los módulos necesarios. Cada módulo es un gestor del árbol con una lista que almacenará pares de nodos (uno del sistema controlable y uno del sistema obstáculo), grados de precisión de cada nodo y el menor tiempo en el que se puede producir colisión.

La Figura 4.4. muestra el árbol de una célula compuesta de cuatro sistemas, de los cuales tres, al ser controlables (dos brazos-robot y una AGV) deben tener un detector de colisiones, mientras que el otro sistema es un obstáculo fijo (una mesa) que no requiere detector de colisiones. Para los tres detectores de colisiones se representa la situación inicial con que se debe partir en cada módulo, es decir, aquellos pares de nodos considerados. Obsérvese que el brazo-robot del primer sistema puede colisionar con su plataforma base, por lo que su detector de colisiones debe tener un módulo para este caso. Por otra parte, el vehículo autoguiado del cuarto sistema de la célula puede colisionar con el brazo-robot del tercer sistema de dos formas distintas, por lo que la situación de partida es con dos pares de nodos en la lista.

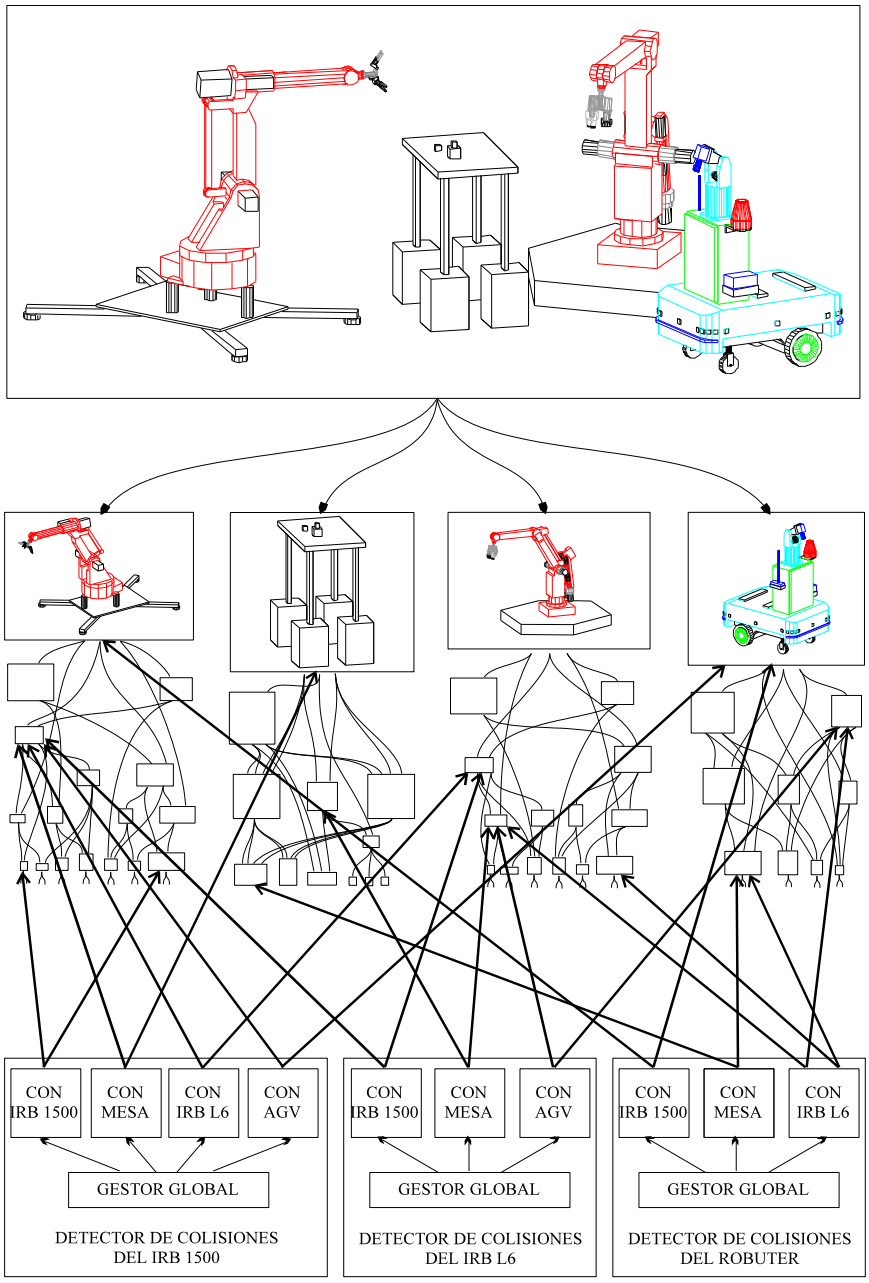


Figura 4.4. Estructura Jerárquica de una Célula y Detectores de Colisión Necesarios. *La célula se compone de cuatro sistemas, dos brazos-robot y un AGV (que deben disponer de un detector de colisiones) y una mesa (que no lo requiere). Los detectores de colisiones tienen tantos módulos como sistemas con los que puedan colisionar. El IRB 1500 de ABB debe tener un módulo para detectar colisiones con su plataforma, mientras que el AGV Robuter tiene dos posibles formas de colisionar con el IRB L6 de ABB. Las flechas indican los nodos con que parte cada módulo.*

El coordinador de la célula se encarga de lanzar los detectores de colisión de los sistemas según el tiempo previsto en que se puede producir colisión, de forma que se dispare primero aquel que antes pueda provocar una colisión. El coordinador maneja dos tiempo: un tiempo t_m para el que se conoce que no hay colisión y a todos los sistemas se les ha permitido que se muevan hasta la configuración alcanzable en ese momento y un tiempo t_i ($t_i>t_m$) que es el mínimo tiempo en que un sistema puede colisionar. El coordinador se debe asegurar (mediante llamadas a los módulos cinemáticos) que el árbol represente la situación del instante t_i y lanzar aquellos detectores que hubieran devuelto este tiempo. Cuando se obtenga un nuevo mínimo tiempo de colisión t_i, se deben mandar movimientos a los sistemas de la célula hasta el antiguo tiempo t_i y actualizar el árbol para el nuevo tiempo t_s.

Los datos de entrada al coordinador de la célula y el proceso que debe realizar el coordinador de la célula se resumen en el siguiente algoritmo:

Datos de Entrada del Proceso:

Configuraciones Programadas de los Sistemas $\mathbf{q}_{i,n}$
Velocidades Programadas V_i

Proceso del Coordinador:

Hacer t=0, t_j=0, $\mathbf{T}=\{\varnothing\}$, $\mathbf{q}_i=\mathbf{q}_{i,0}$
Para cada Detector de Colisiones de un Sistema Controlable:
t_i=*Próxima Detección de Colisiones del Detector* i
$\mathbf{T}=\mathbf{T}\cup\{t_i\}$
Repetir
$t_m=t_j$, $t_j=\min\{t_i\in\mathbf{T}\}$, $t=t_j$, $\mathbf{T}=\mathbf{T}\sim\{t_j\}$
$\mathbf{q}_i=\mathbf{q}_{i,0}+t_j(\mathbf{q}_{i,n}-\mathbf{q}_{i,0})$
Permitir Movimiento de los Sistemas hasta $\mathbf{q}_i$ *a velocidad* V_i *para Tiempo* t_m
Ordenar Módulos Cinemáticos que Actualicen Árbol para Tiempo t_j
t_j=*Próxima Detección de Colisiones del Detector j*
Si hay Colisión
Detener Movimientos de la Célula y Mostrar Mensaje de Colisión
$\mathbf{T}=\mathbf{T}\cup\{t_j\}$
Hasta que $\mathbf{q}_i=\mathbf{q}_{i,n}$

Un gestor global de un detector de colisiones de un sistema controlable de la célula será invocado por el coordinador de la célula para un tiempo t para el que existe al menos un módulo que puede producir colisión. Para todos estos módulos se debe calcular el nuevo tiempo en que se produce colisión y devolverlo al coordinador superior.

El proceso que debe realizar el gestor global del detector de colisiones de un sistema se resume en el siguiente algoritmo:

Proceso del Gestor Global del Detector de Colisiones:
Mientras Exista un Módulo con Tiempo de Colisión t
Considerar el Módulo i
t_i=*Próxima Detección de Colisiones del Módulo* i
Si hay Colisión
Devolver Mensaje de Colisión
Devolver t_j *el Mínimo Tiempo de Colisión de un Módulo*

Cada módulo de un detector de colisiones dispone de dos listas: en una primera lista (*Lista_Nodos*) guarda todos los pares de nodos, uno por sistema, que no colisionan, junto a sus grados de precisión y el tiempo mínimo en que se puede producir una colisión. En otra lista (*Lista_Colisión*) se guardan todos los pares de nodos, uno por sistema, que han producido colisión, junto a sus grados de precisión y el tiempo mínimo en que se puede evitar la colisión. Cuando el gestor global invoca al módulo con un tiempo t, no existe ningún par de nodos en la *Lista_Nodos* que pueda haber dado colisión antes de ese tiempo. Esta condición de inicio se debe conseguir asegurando al final del proceso que se devuelve el mínimo tiempo en que un par de nodos pueden producir colisión, por lo que el gestor global solo vuelve a invocar a este módulo para este tiempo.

El módulo debe calcular la distancia entre todos los pares de nodos que tienen como mínimo tiempo de colisión el tiempo actual en que se invoca al módulo. Para todos los pares de nodos que produzcan colisión, se intenta aumentar el grado de precisión de uno de los nodos o, en su defecto, descomponerlo. En ambos casos se debe guardar en la *Lista_Colisión* los casos de colisión producidos, con el mínimo tiempo en que podría evitarse la colisión. Si para un caso de colisión, no es posible aumentar el grado de precisión o descomponer alguno de los nodos, debe terminarse el proceso con un mensaje de colisión, ya que se ha producido colisión para el máximo grado de precisión de las hojas de los árboles. Cuando dos nodos no dan colisión, se actualiza el mínimo tiempo en que se puede producir colisión.

Para asegurar la posibilidad de ascender de nivel, si durante la ejecución del módulo no se realiza en ningún caso un aumento del grado de precisión ni ninguna descomposición de un nodo, se comprueba si algún nodo padre guardado en la *Lista_Colisión* puede haber salido de la colisión. Para ello, se consideran sólo los que tienen tiempo menor que el tiempo actual, calculándose las distancias. Si se produce colisión, se actualiza el mínimo tiempo en que se puede evitar la colisión. Si no hay colisión, se elimina este par de nodos de la *Lista_Colisión*, se guarda en la *Lista_Nodos*, y se elimina de esta última todos los pares de nodos que estén formados por sucesores de estos nodos. Se entiende por sucesor de un nodo, no sólo a aquellos nodos hijos, sino a aquellos con mayor grado de precisión.

El proceso que debe realizar cada módulo del detector de colisiones de un sistema se resume en el siguiente algoritmo:

Proceso del Módulo j:

Mientras Exista $\{(Nodo_{Sis},g_{Sis},Nodo_{Obs},g_{Obs}),t_i\}$ *con* t_i=t *en la Lista_Nodos*
Considerar un $\{(Nodo_{Sis},g_{Sis},Nodo_{Obs},g_{Obs}),t_i\}$
Calcular distancia entre $(Nodo_{Sis},g_{Sis},Nodo_{Obs},g_{Obs})$
Si hay Colisión Entonces
Eliminar $\{(Nodo_{Sis},g_{Sis},Nodo_{Obs},g_{Obs}),t_i\}$ *de Lista_Nodos*
t_c=*Mínimo tiempo en que se puede evitar la colisión*
Guardar $\{(Nodo_{Sis},g_{Sis},Nodo_{Obs},g_{Obs}),t_c\}$ *en Lista_Colisión*
Si se puede aumenter el grado de precisión de un nodo
Decidir para cuál nodo se aumenta la precisión
Aumentar grado de precisión del nodo
Guardar el nuevo $\{(Nodo_{Sis},g_{Sis},Nodo_{Obs},g_{Obs}),t_i\}$ *en Lista_Nodos*
Si no, si se puede descomponer algún nodo
Decidir cuál nodo se descompone
Decidir qué descomposición se realiza
Descomponer el nodo en nodos hijos
Guardar los nuevos $\{(Nodo_{Sis},g_{Sis},Nodo_{Obs},g_{Obs}),t_i\}$ *en Lista_Nodos*
En otro caso
Devolver Mensaje de Colisión
Si no hay Colisión
t_k=*Mínimo tiempo en que se puede producir colisión*
Sustituir t_k *en vez de* t_i *en* $\{(Nodo_{Sis},g_{Sis},Nodo_{Obs},g_{Obs}),t_i\}$ *de Lista_Nodos*
Si no ha habido ningún caso de Colisión
Mientras Exista $\{(Nodo_{Sis},g_{Sis},Nodo_{Obs},g_{Obs}),t_i\}$ *con* t_i<t *en la Lista_Colisión*
Considerar el $\{(Nodo_{Sis},g_{Sis},Nodo_{Obs},g_{Obs}),t_i\}$ *con menor* t_i
Calcular distancia entre $(Nodo_{Sis},g_{Sis},Nodo_{Obs},g_{Obs})$
Si Hay Colisión Entonces
t_c=*Mínimo tiempo en que se puede evitar la colisión*
$t_c=\max\{t_c,t\}$
Actualizar con t_c *en* $\{(Nodo_{Sis},g_{Sis},Nodo_{Obs},g_{Obs}),t_i\}$*de Lista_Colisión*
Si no hay Colisión
Eliminar $\{(Nodo_{Sis},g_{Sis},Nodo_{Obs},g_{Obs}),t_i\}$ *de Lista_Colisión*
t_k=*Mínimo tiempo en que se puede producir colisión*
Guardar $\{(Nodo_{Sis},g_{Sis},Nodo_{Obs},g_{Obs}),t_k\}$ *en Lista_Nodos*
Eliminar de Lista_Nodos elementos con sucesores de $Nodo_{Sis}$
Eliminar de Lista_Nodos elementos con sucesores de $Nodo_{Obs}$
Obtener t_j=*Mínimo* t_i *de Lista_Nodos*
Devolver t_j *como el mínimo tiempo sin colisión para el módulo* j

Cuando en este algoritmo se calcula la distancia entre dos nodos con cierto grado de precisión hay que recordar que puede ser necesario calcular los modelos envolventes de los nodos si no se dispone de ellos (son sub-sistemas articulados) a partir de sus puntos característicos.

La situación de partida con que debe inicializarse todos los procesos, mediante un módulo de inicialización es la siguiente:

Situación de Partida (Inicialización):

Árbol de la Célula para Configuración Inicial
Próxima Detección de Colisiones de los Detectores = 0
Próxima Detección de Colisiones de los Módulos de los Detectores = 0
Lista_Nodos de los Módulos = Pares $\{(\text{Nodo}_{\text{Sis}},1,\text{Nodo}_{\text{Obs}},1),0\}$ *Considerados*
Lista_Colisión de los Módulos = Vacía
$\mathbf{q}_{i,0}$ *= Configuraciones Iniciales de los Sistemas*

La Figura 4.5. muestra el proceso de un módulo del detector de colisiones de un sistema controlable con otro fijo. En la gráfica se muestra la situación inicial con la parte de los árboles visible por el módulo y tres pasos del proceso con las descomposiciones realizadas. En el primer paso, se produce una colisión y se descompone un nodo del sistema 2. En el segundo paso se producen dos colisiones y para ambos casos se descompone un nodo del sistema 1. En el tercer paso, se produce una colisión, y se descompone un nodo del sistema 1 sólo para ese par de nodos. En todo momento, la *Lista_Nodos* tiene almacenados todos los pares de nodos con sus grados de precisión entre los que se está realizando la detección de colisiones actualmente y la *Lista_Colisión* guarda aquellos pares de nodos con su precisión que han producido colisión en algún momento. Por simplicidad, la gráfica no repite pares de nodos para diferentes grados de precisión, como en realidad ocurre.

En el algoritmo anterior no se refleja ningún cambio en la velocidad de movimiento de los sistemas. En principio parece lógico, como se comentó anteriormente, que la velocidad de un sistema sea proporcional a las distancias obtenidas entre ese sistema y sus obstáculos. Sin embargo, al utilizar una estructura jerárquica con modelos envolventes según diferentes grados de precisión, esta solución no parece adecuada. Por ejemplo, si la envolvente de un sistema está cercano a un obstáculo, no representa que el sistema lo esté, ya que al descender de nivel en el árbol, las distancias pueden ser mucho mayores. Por contra, parece adecuado que cuanto más bajo se encuentren los modelos del sistema menor sea la velocidad. Se podría plantear una relación entre la velocidad de movimiento con el nivel más bajo del modelo que se esté utilizando y la distancias que se obtengan con ese modelo.

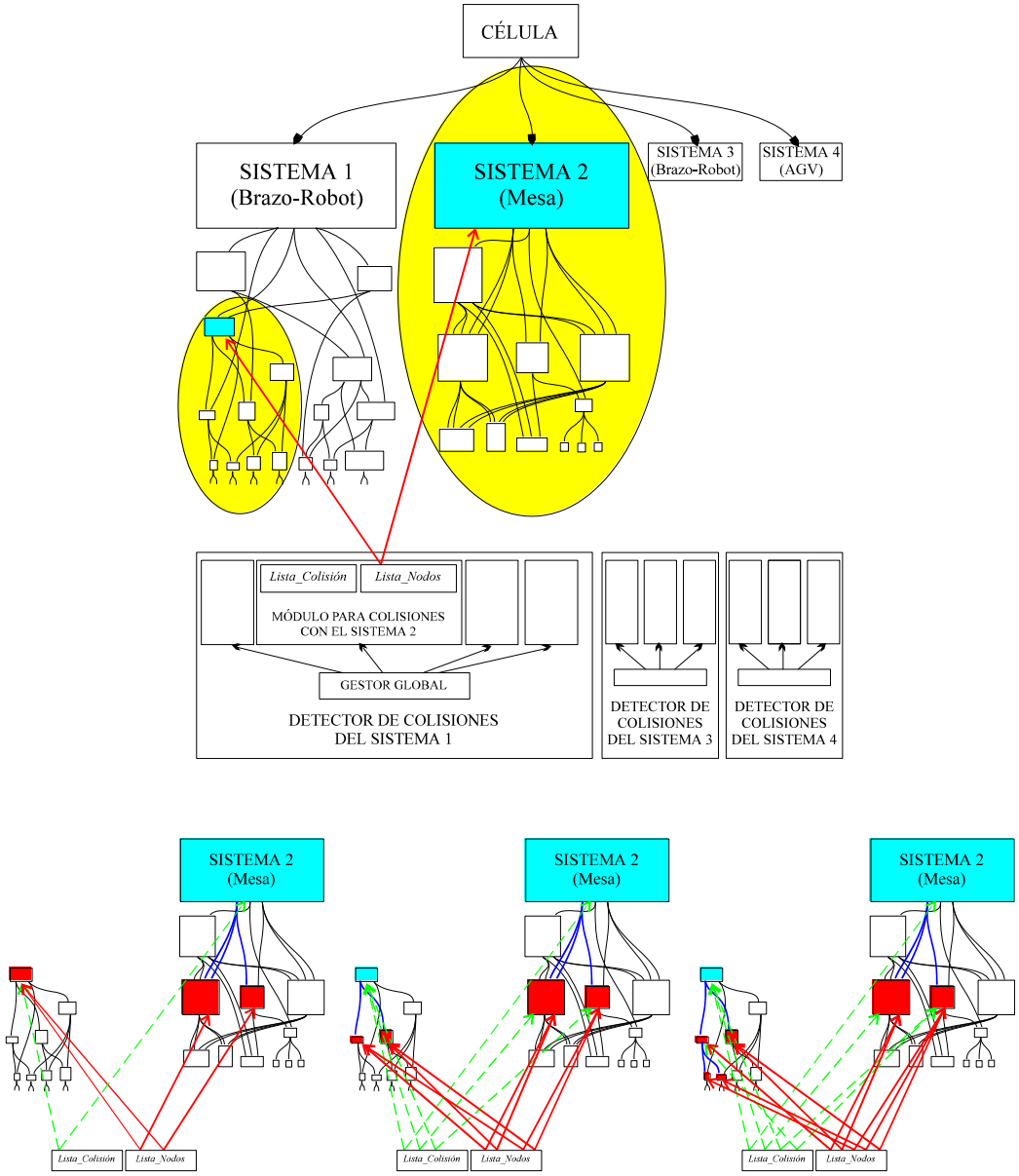


Figura 4.5. Proceso de un Módulo del Detector de Colisiones. *En la gráfica superior se muestra la situación de partida del módulo (con los nodos guardados en Lista_Nodos) remarcándose la parte del árbol visible. En las tres gráficas inferiores se muestran los pasos que produce el algoritmo: primero se produce una colisión y se descompone el árbol del sistema 2; después se producen dos colisiones y para ambos casos se descompone el árbol del sistema 1; en el tercer paso, se produce una colisión, y se descompone el nodo del sistema 1 sólo para ese par de nodos. La Lista_Colisión tiene almacenados todos los pares de nodos que han producido colisión. La Lista_Nodos guarda aquellos nodos con los que actualmente está realizando la detección de colisiones. Por simplicidad, no se repiten nodos para diferentes grados de precisión, como en realidad ocurre.*

En el algoritmo anterior ha quedado otro punto pendiente: cuando hay que aumentar la precisión de un nodo o descomponer un nodo, ¿cómo se decide qué nodo se selecciona? Existen varias formas de realizar esta selección:

- Probar todos los casos posibles y quedarse con el que devuelva mayor distancia entre los nodos. Esta opción dará siempre los mejores resultados, pero es una búsqueda exhaustiva y consume más tiempo.
- Frente a la opción anterior, se podría tomar siempre un caso predeterminado (por ejemplo, seleccionar siempre el nodo del sistema controlable), pero esto evita la posibilidad de tomar diferentes alternativas.
- Mejor que la segunda opción es seleccionar el caso en que se puedan calcular las distancias más rápidamente, pero tampoco producirá diferentes caminos para recorrer los árboles.
- Otra opción es seleccionar el nodo aleatoriamente, lo que producirá diferentes alternativas con poco coste.

Una buena opción de realizar la selección del nodo para el que se aumenta la precisión o se realiza la descomposición consiste en una mezcla de las dos últimas: realizar la selección aleatoriamente pero con unas probabilidades proporcionales a la rapidez con que se puedan calcular las distancias. Esta opción favorece a los casos rápidos pero permite diferentes variantes de forma aleatoria.

El algoritmo se podría mejorar si la ascensión de nivel también se puede realizar de diferentes formas, y no sólo como una vuelta atrás de la descomposición realizada anteriormente. Para ello habría que seleccionar qué nodos se agrupan, de una forma similar a como se seleccionan los nodos de descomposición.

√

CAPÍTULO 5. PLANIFICACIÓN DE MOVIMIENTOS MEDIANTE CAMPOS POTENCIALES ARTIFICIALES

En este quinto capítulo se analiza la técnica de planificación de movimientos basada en campos potenciales, con especial interés en su aplicación a los modelos esféricos. La formalización de esta técnica para el caso de un móvil con rotación y el estudio y análisis de la utilización de técnicas de optimización clásicas para resolver la planificación de movimientos con campos potenciales artificiales son los temas tratados en este capítulo.

—¡Correcto! ¿Y dónde estamos ahora?
—Exactamente en la explicación. El conflicto entre las varias reglas es subsanado por los diferentes potenciales positrónicos en el cerebro. Supongamos que un robot se dirige hacia un peligro y lo sabe. El potencial automático implantado por la Regla Tres le hace volver atrás. Pero supongamos que tú le ordenas que se dirija hacia ese peligro. En ese caso, la Regla Dos establece un contrapotencial más elevado que el anterior, y el robot sigue las órdenes con riesgo de su existencia.

"Círculo Vicioso". Isaac Asimov, 1942.

5.1. La Técnica de Campos Potenciales Artificiales

En comparación con otros métodos, la técnica de planificación de movimientos mediante campos potenciales artificiales [Kha86], ya introducida en la Sección 1.3.3, puede ser muy eficiente, pese a tener un gran inconveniente: al ser un método de minimización local puede quedar atrapado en mínimos locales que no sean la configuración destino. Una solución es diseñar funciones potenciales que no tengan mínimos locales distintos a la configuración destino o bien sean pequeños en número y tamaño, como los intentos de [V&K87], [K&V88] y [K&K91]. Otra opción consiste en combinar campos potenciales con mecanismos potentes de escape de mínimos locales, como los movimientos aleatorios de [B&L89a], [B&L89b] y [B&L90], el seguimiento de obstáculos de [War89], el relleno de mínimos locales con valores infinitos del potencial de [P&L88], el aprendizaje de caminos mediante diferentes técnicas de [F&T89], o el conexionado de los mínimos y máximos locales de la función potencial en el espacio de configuraciones de [SSK92].

Una de las funciones propuestas es la de los campos potenciales generalizados [Kro84]. Se trata de una función que depende tanto de la configuración del robot como de su velocidad: el robot será repelido por un obstáculo sólo si está cercano al mismo y su velocidad apunta hacia el obstáculo; si el robot se mueve paralelo al obstáculo, no tendrá que ser repelido. Una idea similar utilizando campos potenciales vectoriales se presenta en [M&B93].

Una idea similar [F&T87a] describe un método consistente en minimizar un criterio cuadrático bajo unas restricciones lineales. La minimización provoca la atracción del robot hacia el destino, mientras que cada restricción actúa a modo de amortiguador, apartando el robot de los obstáculos cuando está cerca y se dirige hacia ellos. Estas funciones potenciales, si bien son interesantes, provocan, como efecto secundario, la presencia de mínimos locales, puede que incluso más difíciles de tratar que el caso general.

Existen funciones potenciales con ningún mínimo local, pocos o pequeños mínimos locales. La "calidad" de una función potencial se puede medir con el número de mínimos locales, pero existen otros factores importantes, como el "tamaño", el "poder" o la "potencia" atractiva de estos mínimos. Es decir, una técnica de búsqueda puede quizá escapar de varios mínimos locales pequeños sin dificultad y, sin embargo, fallar en el intento de escapar de un único pero potente mínimo local.

[Til90] considera dos tipos de campos locales: clásicos y generalizados. El campo potencial repulsivo clásico depende sólo de la posición relativa del robot y los obstáculos, de forma que se ejercen fuerzas repulsivas en puntos cercanos a obstáculos,

atenuando la fuerza con la potencia k-ésima de la distancia al obstáculo más próximo, incluso si no existe ninguna componente de la velocidad del robot en la dirección del obstáculo. Éste fue el hecho que motivó, al menos en parte, a Krogh [Kro84] a investigar sobre un campo potencial generalizado que fuese función de la velocidad, no sólo de la posición. Con esta idea, un obstáculo no ejerce grandes fuerzas repulsivas en puntos vecinos que están estacionarios o alejándose del mismo. Este campo potencial generalizado es sensible al tiempo de impacto, más que a la distancia. Krogh define el potencial generalizado como la inversa del "tiempo reserva de evitación", que es la diferencia entre el tiempo que costaría decelerar con la menor velocidad constante permitiendo al robot detenerse cuando llegue al obstáculo más próximo en una dirección dada y el tiempo que permitiría la detención con la máxima deceleración. El potencial generalizado se aproxima a infinito cuando el tiempo reserva se acerca a cero, lo que corresponde al caso de tolerancia nula para evitar una colisión. A medida que el robot se aproxima al obstáculo el tiempo de reserva disminuye y, eventualmente, la fuerza total apuntará hacia fuera del obstáculo.

Es de destacar que no es posible afirmar nada sobre la "bondad" de un campo potencial de forma independiente al algoritmo usado para la generación de la trayectoria. Por eso Tilove [Til90] estudia cuatro posibilidades, las que resultan de utilizar campos clásicos o generalizados con dos algoritmos de generación de caminos: *hill climbing* (en cada paso se mueve en la dirección de la fuerza) y *force control* (se trata al móvil como una unidad de masa y al campo como una fuerza de entrada, definiendo así la dirección de aceleración, más que la siguiente posición del robot). La ventaja de éste último es que tiene en cuenta la dinámica del vehículo. Los ejemplos muestran que no deberían usarse campos generalizados junto con *hill climbing*. Oscilaciones indeseables se experimentan al utilizar algoritmos *force control*, resultantes de interacciones entre la aceleración del robot y los límites de velocidad. Para mitigar el efecto de estas oscilaciones se añade una fuerza amortiguadora viscosa en la dirección normal a la dirección hacia el destino y cuya magnitud depende de la componente de la velocidad en la dirección normal.

Los mínimos locales diferentes del mínimo global son la causa de la ineficiencia, casi inherente, de los métodos de campos potenciales. La idea de las funciones de navegación es la siguiente: puesto que la función potencial no es algo que venga impuesto en la formulación original del problema básico de planificación, surge la siguiente pregunta:

> *¿Es posible construir una función potencial U con un único mínimo localizado en la configuración final cuyo dominio de atracción incluya el subconjunto del espacio libre (no ocupado por obstáculos) que está conectado a la configuración final?*

Si tal función existe se llama *función de navegación global*. Y si se pudiese construir, con una búsqueda *depth-first* se garantizaría la llegada al punto destino para cualesquiera que fuesen las configuraciones inicial y final, ambas en el mismo subconjunto del espacio libre, y la indicación de fallo en caso contrario.

Sin embargo es posible demostrar, que en general no existe una función de navegación global válida para todos los casos. Pero de todos modos no parece haber ninguna objeción topológica para la existencia de una función potencial con las características arriba mencionadas aunque con un conjunto de puntos de silla de la función potencial. Tales puntos serían configuraciones en equilibrio inestable y, con una ligera desviación aleatoria, sería posible evadirlos. Entonces se tendría una función *cuasi* de navegación global.

Aunque parece difícil construir una función de navegación de forma analítica, el problema puede volverse mucho más sencillo si se calcula una función de navegación numérica sobre la representación del espacio de configuraciones discreto, en forma de rejilla o *grid*. En [B&L89a] y [B&L89b] se proponen algoritmos para hallar dos de estas funciones de navegación, que son eficientes cuando la dimensión del espacio de configuraciones es pequeña (2 ó 3) y su complejidad es independiente de la geometría del espacio libre. También trabajan en espacio discreto [CLZ95] y [Jan95].

Los potenciales elípticos son otra aproximación para reducir el número y el tamaño de los mínimos locales de la función potencial. Consiste en definir la función potencial U como una combinación de un potencial atractivo con una simetría alrededor de la configuración objetivo (una parábola, por ejemplo) y un potencial repulsivo elíptico. La idea básica es sencilla: definir el potencial repulsivo alrededor del obstáculo de tal modo que las líneas o las superficies equipotenciales converjan hacia la frontera del obstáculo cuando la distancia al mismo tiende a cero y hacia superficies esféricas cuando la distancia aumenta [V&K87]. Se desarrollan ecuaciones que representan n-elipses (con $n \geq 1$). Esta expresión se modifica progresivamente para que vaya cumpliendo las restricciones deseadas. Además se puede actuar sobre ciertos parámetros que participan en las ecuaciones resultantes de forma que eligiendo los valores apropiados, se ajuste el "poder" de los mínimos locales.

No obstante las funciones elípticas tienen varios inconvenientes que dificultan su aplicación práctica:

- Sólo se pueden generalizar a formas convexas simples en espacios bi- y tri-dimensionales usando formulaciones supercuadráticas con factores de escala no constantes.
- Se requiere que los obstáculos estén bastante separados unos de otros; ningún obstáculo debe estar en la zona de influencia de otro obstáculo.
- Su definición no proporciona un modo directo de calcular el potencial en determinadas configuraciones y requiere interpolación entre valores precalculados en configuraciones vecinas.

La formulación supercuadrática es una generalización del método de funciones potenciales elípticas, y por tanto es viable para una clase mucho más amplia de objetos [K&V88].

[CBW90] y [K&K91] son dos ejemplos en los que se discute el uso de funciones harmónicas, que son funciones que satisfacen la ecuación de Laplace. Con estas funciones, la búsqueda de caminos puede realizarse de forma muy rápida en el espacio de configuraciones del robot, y lo más importante es que estas funciones no presentan mínimos locales. El cómputo de la ecuación de Laplace puede realizarse de forma muy adecuada sobre arquitecturas masivamente paralelas (*MPP*). Tras una presentación bastante formal de las funciones harmónicas, en estos artículos se analizan los problemas que presenta la superposición de funciones harmónicas, entre otros aspectos. Se concluye que la prevención de mínimos locales se consigue a costa de una menor velocidad, aunque esto no lo consideran gran inconveniente si se recurre a *MPPs*.

La técnica de planificación *Depth-First* [Lat91] consiste en construir una trayectoria como concatenación de segmentos sucesivos de trayectoria donde cada uno de estos segmentos se orienta en la dirección del gradiente negativo de la función potencial. Su aplicación se basa en un proceso iterativo donde para cada configuración, según la dirección del gradiente, se determina la siguiente configuración a alcanzar. La longitud de cada segmento se acota para asegurar que el movimiento del móvil a lo largo del segmento no interseccione con una zona ocupada por los obstáculos. Resulta aconsejable utilizar longitudes de segmentos pequeños, para que el móvil no colisione con los obstáculos por un lado, y por otro que el segmento de la trayectoria no sea tan largo que el móvil sobrepase la configuración destino, pues, de lo contrario, podría producirse alguna oscilación del móvil (incluso indefinidamente) en las cercanías de la configuración destino. Por otra parte, la longitud del segmento puede ser un valor variable durante la trayectoria, función de la distancia al destino y/o a los obstáculos. Esto es particularmente interesante, pues permite que en zonas sin obstáculos cercanos, el móvil pueda aprovechar esta situación y avanzar con segmentos de longitudes mayores. También tiene sentido que el móvil avance a tramos mayores si se encuentra lejos de la configuración destino que si se encuentra cerca de ella.

La técnica anterior sigue la filosofía de seguir siempre la mayor pendiente descendente de la función potencial hasta conseguir la configuración final (de ahí su nombre, primero en profundidad). Esta técnica tiene como inconveniente la facilidad con que puede quedar atrapado el móvil en un mínimo de la función potencial diferente de la configuración destino, no resultando fácil tratar de solucionar los mínimos locales en la planificación Depth-First. Ni tan siquiera descubrir que el móvil ha llegado a uno de estos mínimos resulta directo, ya que, los movimientos son discretizados y no es probable que el móvil se pare exactamente en una configuración donde la suma de las fuerzas se anule. En cambio, será usual que se genere un bucle de segmentos intermedios alrededor de esta configuración de mínimo local.

Para detectar que se alcanza alguno de estos mínimos habrá que comprobar que varias configuraciones sucesivas están bastante cercanas unas de otras. Una vez detectado un mínimo local, el planificador debería escapar de él. Una posibilidad es moverse a lo largo de cierta dirección una cierta distancia. La dirección de este movimiento podría ser tangente a la superficie equipotencial repulsiva en el mínimo local, lo cual equivale a moverse alrededor de la combinación de obstáculos que originan este mínimo problemático.

Por tanto, en la técnica Depth-First, ni la detección de los mínimos locales ni el escape de los mismos se basan en un mecanismo sistemático y seguro. Una aproximación más metódica que la anterior es la técnica *Best-First* [Lat91], donde se construye una rejilla regular de configuraciones tras discretizar todas las dimensiones que constituyen el espacio y se utiliza el concepto de *p-vecinos*, con $1 \leq p \leq m$, siendo m el número de dimensiones de la rejilla. Los p-vecinos de una configuración se definen como las configuraciones de la rejilla que tienen como máximo p coordenadas diferentes. La idea de Best-First es construir iterativamente un árbol cuyos nodos son configuraciones en la rejilla, siendo la raíz del árbol la configuración inicial. En cada iteración, el algoritmo examina los p-vecinos de la hoja del árbol que tiene el menor valor de la función potencial, se queda con los vecinos que todavía no están en el árbol en los que la función potencial es menor que un cierto umbral relativamente grande, e instala los vecinos con los que se queda en el árbol como sucesores de la hoja actualmente considerada. El algoritmo termina cuando se alcanza la configuración final (éxito) o cuando el subconjunto de la rejilla accesible desde la configuración inicial ha sido completamente explorado (fallo). Cada nodo en el árbol tiene un puntero hacia su padre. Si se alcanza la configuración destino se puede generar la trayectoria recuperando los punteros desde la configuración inicial a la configuración destino.

El algoritmo presentado garantiza que se devuelve una trayectoria libre de colisiones siempre que ésta exista y que informa del fallo en otro caso. Sin entrar en consideraciones temporales o de memoria, se puede usar un árbol balanceado para tener las configuraciones en el árbol ordenadas según las coordenadas. Esta modificación incrementa ligeramente el tiempo de ejecución, pero reduce la cantidad de memoria necesaria. Sin embargo, esta técnica presenta dos inconvenientes: es una planificación totalmente global y, por tanto, off-line y se basa en un espacio (cartesiano o de configuraciones) discretizado y no continuo.

Finalmente, otra técnica comentada en [Lat91] es el movimiento según planteamiento óptimo, construyendo una función de la trayectoria que se optimiza sobre todas las posibles trayectorias. Si la función potencial está definida sobre todo el espacio de configuraciones, con valores grandes en los obstáculos y valores pequeños en el espacio libre, una posible forma de definir la función trayectoria es como suma de un término para hacer que la optimización produzca una trayectoria libre de colisiones y un segundo término (opcional) para poder obtener trayectorias más cortas. Además, otros objetivos/criterios pueden codificarse de forma similar en la función trayectoria. Esta elegante posibilidad es la mayor ventaja de esta técnica. El mayor inconveniente es similar al de Depth-First, los mínimos locales, junto al hecho de optimizar un índice sobre un espacio de más dimensiones (el número de configuraciones en el camino), lo que puede y suele ser muy costoso.

5.2. Definición de Campos Potenciales

Las propiedades que debería tener una función potencial para modelar obstáculos [K&V88] son:

- El potencial debería tener simetría esférica (principalmente para grandes distancias), para evitar la creación de mínimos locales cuando este potencial se suma con otros.
- El contorno del potencial cerca de la superficie de un objeto debería seguir el contorno de la superficie, de forma que no se eliminen grandes porciones del área de trabajo.
- El potencial repulsivo de un obstáculo debería tener un rango de influencia limitado.
- Tanto el potencial como su gradiente deben ser funciones continuas.

Sin embargo, la mayoría de técnicas basadas en campos potenciales comentadas en la sección anterior representan los objetos (obstáculos, móvil, ...) como entes poligonales, lo que dificulta la definición de una función potencial que cumpla las características anteriores. El modelado de objetos con volúmenes esféricos permite definir unas funciones potenciales que cumplan las características anteriores, ya que las dos primeras características se cumplen directamente, por lo que sólo hay que exigir las dos últimas, fácilmente conseguibles. Además la utilización de envolventes esféricas reduce la complejidad del problema de la planificación de movimientos, por la rapidez con que se pueden calcular las distancias entre objetos, y presenta grandes ventajas para su utilización en la técnica de campos potenciales, ya que ésta se basa fundamentalmente en el cálculo de distancias.

Aunque el robot o móvil se represente geométricamente mediante poli-esferas, su situación en el espacio de trabajo se puede definir mediante su configuración. Si el móvil tiene m grados de libertad, se trabaja en un espacio $\Re^m$, donde la configuración del robot se puede representar como un vector $\mathbf{q}=(q_1,...,q_m)$. Cualquier cambio de la situación en que se encuentre el robot es un cambio en su configuración. Para gobernar los movimientos del robot, se define una función potencial $U(\mathbf{q}):\Re^m \rightarrow \Re$ derivable con un mínimo global en el destino a alcanzar y con valores elevados donde se encuentren los obstáculos.

En este entorno, el movimiento del móvil se puede obtener como aplicación de una fuerza en la dirección negativa (para alcanzar el mínimo) del gradiente del potencial, $\mathbf{F}(\mathbf{q}) = -\nabla U(\mathbf{q})$, definido éste según:

$$\nabla U(\mathbf{q}) = \begin{pmatrix} \dfrac{\partial U(\mathbf{q})}{\partial q_1} \\ \vdots \\ \dfrac{\partial U(\mathbf{q})}{\partial q_m} \end{pmatrix}$$

Para conseguir que el robot sea atraído hacia el destino y repelido por los obstáculos, $U(\mathbf{q})$ se construye como la suma de dos funciones potenciales más elementales:

$$U(\mathbf{q}) = U_{atr}(\mathbf{q}) + U_{rep}(\mathbf{q})$$

donde $U_{atr}(\mathbf{q})$ es el potencial atractivo asociado con la configuración destino $\mathbf{q}_d$, y $U_{rep}(\mathbf{q})$ es el potencial repulsivo asociado a los obstáculos. $U_{atr}(\mathbf{q})$ es independiente de los obstáculos; mientras que $U_{rep}(\mathbf{q})$ es independiente de la configuración final.

Con esta nomenclatura, $\mathbf{F}(\mathbf{q})$ es la suma de dos vectores:

$$\mathbf{F}(\mathbf{q}) = -\nabla U(\mathbf{q}) = -\nabla U_{atr}(\mathbf{q}) - \nabla U_{rep}(\mathbf{q}) = \mathbf{F}_{atr}(\mathbf{q}) + \mathbf{F}_{rep}(\mathbf{q})$$

llamados fuerza atractiva $\mathbf{F}_{atr}(\mathbf{q})$ y fuerza repulsiva $\mathbf{F}_{rep}(\mathbf{q})$. Las formas de definir los potenciales atractivos y repulsivos, y de obtener las fuerzas atractivas y repulsivas se comenta en las dos secciones siguientes.

5.2.1. Potenciales Atractivos

El campo potencial atractivo se puede definir de varias formas, por ejemplo como una función parabólica, esto es:

$$U_{atr}^{P}(\mathbf{q}) = \frac{\varepsilon_p}{2} \left\| \mathbf{q} - \mathbf{q}_d \right\|^2$$

donde ε_p es un factor de escala positivo, que actúa como factor de atracción, indicando con qué intensidad la fuerza atractiva tenderá a llevar al móvil hacia su configuración final. El potencial $U_{atr}^{P}(\mathbf{q})$ es siempre positivo excepto en la configuración destino, $\mathbf{q}_d$, donde existe un mínimo global con $U_{atr}^{P}(\mathbf{q}_d) = 0$.

La fuerza atractiva $\mathbf{F}_{atr}(\mathbf{q})$ se obtiene derivando $U_{atr}^{P}(\mathbf{q})$, lo que resulta:

$$\mathbf{F}_{atr}(\mathbf{q}) = -\nabla U_{atr}^{P}(\mathbf{q}) = -\varepsilon_p \left(\mathbf{q} - \mathbf{q}_d \right)$$

El módulo de esta fuerza atractiva converge linealmente hacia 0 a medida que la configuración del robot se aproxima a la configuración final. Sin embargo, genera una fuerza que aumenta con la distancia a la configuración destino y tiende a infinito

cuando dicha distancia tiende a infinito. Esto produce que los efectos de este potencial atractivo parabólico sean buenos "cerca" del destino pero excesivos "lejos" de él.

Para evitar este problema se puede definir el campo atractivo como un "pozo" cónico:

$$U_{atr}^{C}(\mathbf{q}) = \varepsilon_c \left\| \mathbf{q} - \mathbf{q}_d \right\|$$

El factor de escala ε_C tiene el mismo significado que en el caso anterior. Para este caso la fuerza atractiva $\mathbf{F}_{atr}(\mathbf{q})$ resultante es:

$$\mathbf{F}_{atr}(\mathbf{q}) = -\nabla U_{atr}^{C}(\mathbf{q}) = -\varepsilon_c \frac{\left(\mathbf{q} - \mathbf{q}_d\right)}{\left\| \mathbf{q} - \mathbf{q}_d \right\|}$$

El módulo de esta fuerza atractiva cónica es constante, por lo que este potencial es bueno "lejos" del destino, excepto en $\mathbf{q}_d$, donde la función potencial es singular. Ahora la amplitud de la fuerza no tiende a 0 cuando $\mathbf{q} \rightarrow \mathbf{q}_d$, por lo que "cerca" de el destino no tiene las características de estabilidad de la función parabólica.

Una forma de combinar las ventajas de ambos tipos de funciones es definir un potencial parabólico cerca de destino, y otro cónico para distancias más alejadas de la configuración objetivo. El campo resultante sería el siguiente:

$$U_{atr}^{PC}(\mathbf{q}) = \begin{cases} U_{atr}^{P}(\mathbf{q}) & \text{si } \left\| \mathbf{q} - \mathbf{q}_d \right\| \leq 2\dfrac{\varepsilon_C}{\varepsilon_P} \\ U_{atr}^{C}(\mathbf{q}) & \text{si } \left\| \mathbf{q} - \mathbf{q}_d \right\| \geq 2\dfrac{\varepsilon_C}{\varepsilon_P} \end{cases}$$

La distancia umbral de cambio de potencial debe ser $2\varepsilon_C/\varepsilon_P$ para que el campo no pierda su propiedad de continuidad. Sin embargo, pese a que en los dos casos anteriores el gradiente era continuo, en este caso el gradiente no es continuo, por lo que este potencial no cumple la cuarta condición exigible a una buena definición de potencial.

La Figura 5.1 intenta ilustrar estas ideas: el potencial atractivo cónico cumple que el módulo de su fuerza atractiva se mantiene constante con la distancia al destino (este comportamiento no es deseable cerca del destino, puesto que podría producir inestabilidades en el móvil); para el potencial atractivo parabólico, el módulo de su fuerza atractiva decrece con la proximidad al destino (de esta forma se puede producir la necesaria convergencia), pero su magnitud aumenta con la distancia, no siendo deseable este tipo de fuerzas tan elevadas lejos del destino; la combinación de las ventajas de ambos, produce un campo potencial del que se derivan unas fuerzas cuyo módulo es constante cuando la distancia es mayor que un umbral y se reduce con la distancia al destino cuando la distancia es menor que esa distancia umbral. Sin embargo, aunque el potencial que resulta es continuo, no lo es el módulo de su fuerza.

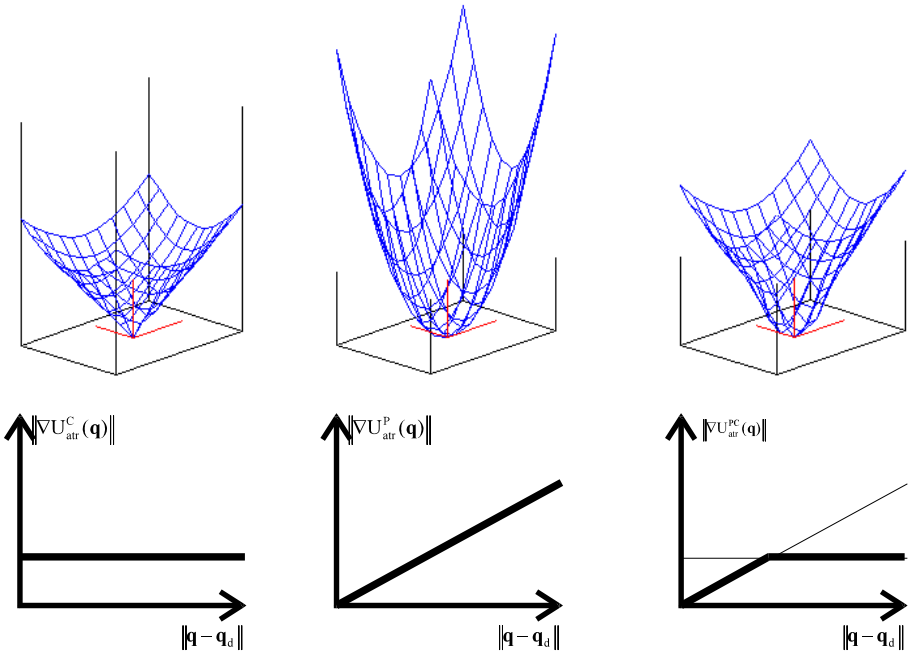


Figura 5.1. Campos Potenciales Atractivos Cónico, Parabólico y Combinado. *En la gráfica de la izquierda se muestra un potencial atractivo cónico, donde el módulo de la fuerza atractiva es constante, independientemente de la distancia al destino. En la gráfica central se muestra un potencial atractivo parabólico, con el módulo de la fuerza proporcional a la distancia al destino. En la gráfica de la derecha se muestra un potencial combinado, con el problema de la discontinuidad en el módulo de la fuerza. En todas las gráficas, el avance en el sentido positivo del eje de abcisas supone el alejamiento del destino (punto medio de un espacio de trabajo 2D).*

Una posible alternativa consiste en utilizar, en lugar de una distancia umbral, un campo potencial atractivo parabólico-cónico ponderado, de forma que para cualquier distancia al destino se tuviese en cuenta los dos campos, pero ponderados: un factor proporcional a la distancia multiplicaría al potencial cónico (de esta forma, este potencial pesaría mucho a distancias elevadas), mientras que un factor inversamente proporcional a la distancia multiplicaría al potencial atractivo (así contribuiría más al potencial resultante a pequeñas distancias).

5.2.2. Potenciales Repulsivos

La definición de un potencial repulsivo pretende crear una especie de barrera potencial alrededor de cada obstáculo impidiendo así que pueda ser atravesado por el robot. Además es interesante que el potencial repulsivo no afecte al movimiento del robot si éste está suficientemente alejado de los obstáculos, y por tanto no existe peligro de colisión (tercera característica deseable de la definición de un potencial artificial). Estas dos condiciones se tienen en cuenta en la siguiente función de potencial repulsivo para un obstáculo:

$$U_{rep}^{obs}(\mathbf{q}) = \begin{cases} 0 & \text{si } d_{inf}^{obs} \le \left\| \mathbf{q} - \mathbf{q}_{obs} \right\| \\ \frac{\eta_{obs}}{2}\left(\frac{1}{\left\| \mathbf{q} - \mathbf{q}_{obs} \right\|} - \frac{1}{d_{inf}^{obs}} \right)^2 & \text{si } d_{seg}^{obs} \le \left\| \mathbf{q} - \mathbf{q}_{obs} \right\| \le d_{inf}^{obs} \\ -K_{seg}\left(\frac{\left(\mathbf{q} - \mathbf{q}_{obs} \right)^2}{2} - \frac{\left(\mathbf{q}_{seg} - \mathbf{q} \right)^2}{2} \right) & \text{si } \left\| \mathbf{q} - \mathbf{q}_{obs} \right\| \le d_{seg}^{obs} \end{cases}$$

donde:

- $\left\| \mathbf{q} - \mathbf{q}_{obs} \right\|$ denota la mínima distancia del móvil en la configuración $\mathbf{q}$ al obstáculo.
- d_{inf}^{obs} es un factor de escala positivo que representa la *zona de influencia* que determina el espacio alrededor del obstáculo dentro del cual la acción repulsiva del obstáculo influirá sobre el móvil. De esta forma, el móvil se hace insensible al potencial repulsivo del obstáculo cuando les separa una distancia grande (mayor que la distancia de la zona de influencia).
- El *factor de repulsión* η_{obs} es un factor de escala que permite ponderar en mayor o menor medida el grado de repulsión que el obstáculo ejerce sobre el móvil cuando se encuentra bajo su área de influencia.
- La constante positiva no nula d_{seg}^{obs} se llama *distancia de seguridad* del obstáculo y se utiliza como un margen de seguridad del obstáculo. Esta distancia previene que el móvil se acerque demasiado al obstáculo, definiendo una especie de envoltura que el móvil no debe sobrepasar. Sin este parámetro, el móvil no llegaría a colisionar, pero podría rozar el obstáculo, lo que, en general, no será deseable.
- $\mathbf{q}_{seg}$ es el obstáculo con un desplazamiento exterior de valor d_{seg}^{obs} (en el caso de utilizar modelos esféricos, equivale a aumentar los radios que definen la poli-esfera con un incremento d_{seg}^{obs}).
- La constante K_{seg} sirve para conseguir que el potencial y su gradiente sean continuos y debe cumplir:

$$K_{seg} = \frac{\eta_{obs}}{\left(d_{seg}^{obs} \right)^3}\left(\frac{1}{d_{seg}^{obs}} - \frac{1}{d_{inf}^{obs}} \right)$$

Estos parámetros permiten definir unos obstáculos más sensibles o amenazantes a la colisión que otros, es decir, si por cualquier causa, es especialmente importante que el móvil no colisione con él o no se aproxime tanto. En general, los obstáculos que resulten más peligrosos para una colisión, tendrán los valores d_{inf}^{obs}, η_{obs} y d_{seg}^{obs} más altos que el resto de obstáculos.

La función potencial repulsiva, tal como se ha definido, es positiva o cero, tendiendo a infinito a medida que **q** se aproxima al obstáculo y es nula cuando la distancia de la configuración del robot al obstáculo es mayor que la distancia de influencia.

La fuerza que ejercerá el obstáculo sobre el móvil viene dada por el gradiente del potencial repulsivo. Para ello hay que tener en cuenta que el gradiente de la distancia del móvil en la configuración **q** al obstáculo $\mathbf{q}_{obs}$ es el vector unitario de colisión expresado como $\frac{(\mathbf{q}-\mathbf{q}_{obs})}{\|\mathbf{q}-\mathbf{q}_{obs}\|}$. Por tanto, la fuerza repulsiva ejercida por un obstáculo es:

$$-\nabla U_{rep}^{obs}(\mathbf{q}) = \begin{cases} \mathbf{0} & \text{si } d_{inf}^{obs} \le \|\mathbf{q}-\mathbf{q}_{obs}\| \\ \eta_{obs}\left(\frac{1}{\|\mathbf{q}-\mathbf{q}_{obs}\|}-\frac{1}{d_{inf}^{obs}}\right)\frac{1}{\|\mathbf{q}-\mathbf{q}_{obs}\|^2}\frac{(\mathbf{q}-\mathbf{q}_{obs})}{\|\mathbf{q}-\mathbf{q}_{obs}\|} & \text{si } d_{seg}^{obs} \le \|\mathbf{q}-\mathbf{q}_{obs}\| \le d_{inf}^{obs} \\ K_{seg}\left\{(\mathbf{q}-\mathbf{q}_{obs})+(\mathbf{q}_{seg}-\mathbf{q})\right\} & \text{si } \|\mathbf{q}-\mathbf{q}_{obs}\| \le d_{seg}^{obs} \end{cases}$$

El tercer término representa el vector $K_{seg}(\mathbf{q}_{seg}-\mathbf{q}_{obs})$ pero considerado en la dirección que va del obstáculo al móvil. El campo repulsivo total ejercido por todos los obstáculos es la suma del potencial repulsivo debido a cada uno de los obstáculos. Si existen nobs obstáculos, el campo repulsivo total es:

$$U_{rep}(\mathbf{q}) = \sum_{obs=1}^{nobs} U_{rep}^{obs}(\mathbf{q})$$

Análogamente, la fuerza repulsiva que se ejerce sobre el móvil se obtiene como suma de todas las fuerzas repulsivas ejercidas por cada obstáculo:

$$\mathbf{F}_{rep}(\mathbf{q}) = -\nabla U_{rep}(\mathbf{q}) = -\sum_{obs=1}^{nobs} \nabla U_{rep}^{obs}(\mathbf{q})$$

Un problema típico de la utilización de estos potenciales es que la configuración destino $\mathbf{q}_d$ a la que se quiere llevar el móvil se encuentre cerca de un obstáculo, quedando englobada por la zona de influencia de ese obstáculo. En este caso, no se puede garantizar que la configuración destino sea un mínimo global de la función potencial total, ya que aunque sí lo es para el potencial atractivo, no lo es para el potencial repulsivo. Esta situación puede provocar que el robot sea incapaz de llegar a $\mathbf{q}_d$, dado el fuerte potencial repulsivo que el obstáculo cercano provoca sobre él.

Una forma de resolver este problema es acotar la distancia de influencia de los obstáculos a un valor menor que la distancia que existe entre el obstáculo y el destino. Sin embargo, esta solución limita la posibilidad de definir obstáculos muy sensibles a la colisión con el móvil cerca de la configuración destino.

Sin embargo existe una mejor forma de tratar este caso, consistente en definir una fuerza atractiva mínima $\mathbf{F}_{min}=\nabla U_{rep}(\mathbf{q}_d)$, es decir, igual en módulo a la fuerza repulsiva que existe en el destino, pero de sentido opuesto. Esta fuerza mínima se superpondría, en cada punto alcanzado por el móvil, a la fuerza resultante, dando una fuerza total:

$$\mathbf{F}(\mathbf{q})=\mathbf{F}_{rep}(\mathbf{q})+\mathbf{F}_{atr}(\mathbf{q})+\mathbf{F}_{min}$$

Por otra parte, para evitar oscilaciones cuando el móvil está cercano al destino, se puede definir un umbral suficientemente pequeño que permita llevar al móvil directamente al destino.

Otro problema que puede plantearse es que el móvil se encuentre en la zona de influencia de más de un obstáculo, con lo que se suman las fuerzas repulsivas resultando una fuerza más grande que si hubiera un único obstáculo de mayor tamaño. Por ello se pueden ponderar los potenciales según el tamaño de los obstáculos, realizando, por ejemplo, el cálculo automático del factor de repulsión η_{obs} en función de los tamaños de los obstáculos.

5.2.3. Planificación Depth-First

En la técnica Depth-First se calcula la siguiente configuración del móvil, a partir de la configuración actual, tomando la dirección de la fuerza resultante total (el gradiente negativo de la función potencial).

Sea $\mathbf{q}_i$ la configuración actual del móvil y $\mathbf{q}_{i+1}$ la nueva configuración a obtener en el segmento i-ésimo de la trayectoria. Sea $\mathbf{F}(\mathbf{q}_i) = -\nabla U(\mathbf{q}_i)$ la fuerza resultante total (suma de la atractiva, las repulsivas y la mínima) ejercida sobre el móvil en la configuración actual. Considérese el vector unitario de dicha fuerza $\hat{\mathbf{F}}(\mathbf{q}_i) = \mathbf{F}(\mathbf{q}_i) / \left\|\mathbf{F}(\mathbf{q}_i)\right\|$. Por tanto el valor de $\mathbf{q}_{i+1}$ puede obtenerse como:

$$\mathbf{q}_{i+1} = \mathbf{q}_i + \delta_i(\mathbf{q}_i)\hat{\mathbf{F}}(\mathbf{q}_i)$$

Esta expresión indica que, al calcular la trayectoria, el cálculo de una configuración en el paso i se basa en el valor calculado para la configuración anterior (en el paso i-1); se trata de un proceso iterativo en el que:

1. Se calcula la fuerza resultante y se halla el vector unitario de dicha fuerza.
2. Se calcula $\mathbf{q}_{i+1}$ en función de $\mathbf{q}_i$.
3. Se vuelve al punto 1, repitiéndose el proceso hasta que se consigue llegar a la configuración destino, o a una vecindad de la misma.

El valor de $\delta_i(\mathbf{q}_i)$ puede ser una constante pequeña a lo largo de la trayectoria, por un lado para que el móvil no colisione con los obstáculos, y por otro para que el segmento de la trayectoria no sea tan largo que el móvil sobrepase la configuración destino, pues, de lo contrario, podría producirse alguna oscilación del móvil (incluso indefinidamente)

en las cercanías de la configuración destino. Por tanto, la longitud de cada segmento se debe acotar por la mínima distancia a un obstáculo o al destino.

Por otra parte, puede ser un valor variable durante la trayectoria, función de la distancia al destino y/o a los obstáculos. Esto es particularmente interesante, pues permite que en zonas sin obstáculos cercanos, el móvil pueda aprovechar esta situación y avanzar con segmentos de longitudes mayores. También tiene sentido que el móvil avance a tramos mayores si se encuentra lejos de la configuración destino que si se encuentra cerca de ella. Esta posibilidad es factible usando una función $\delta_i(\mathbf{q}_i)$ apropiada.

Otra forma alternativa de calcular la nueva configuración $\mathbf{q}_{i+1}$ a partir de la anterior $\mathbf{q}_i$ es:

$$\mathbf{q}_{i+1} = \mathbf{q}_i + \delta_i(\mathbf{q}_i)\mathbf{F}(\mathbf{q}_i)$$

con lo cual no se utiliza el vector unitario, sino directamente el vector fuerza en la dirección más prometedora. En este caso la elección de $\delta_i(\mathbf{q}_i)$ es más peligrosa, o sea, es más difícil tener una función que cumpla las restricciones anteriores (no colisión, no sobrepasamiento del destino), ya que la fuerza resultante puede ser muy grande lejos del destino o cerca de los obstáculos.

En cualquier caso se puede intuir que, a partir de estas dos ecuaciones anteriores, es posible realizar combinaciones interesantes de la función $\delta_i(\mathbf{q}_i)$ consiguiendo diferentes propiedades de las trayectorias resultantes.

La Figura 5.2. muestra tres ejemplos de planificación depth-first en un entorno de trabajo bi-dimensional, dos utilizando el vector unitario de la fuerza pero con diferentes constantes y otro utilizando directamente el vector de la fuerza acotado siempre por la mínima distancia a los obstáculos y el destino. El potencial atractivo en ambos casos es cónico, y tanto el móvil como los obstáculos se han modelado mediante poli-esferas (esfera para el móvil y bi-esfera y tri-esfera para los obstáculos). De las gráficas se desprende que esta técnica de planificación depende enormemente de las constantes seleccionadas, ya que si el incremento es demasiado grande, se producen oscilaciones cerca de los obstáculos.

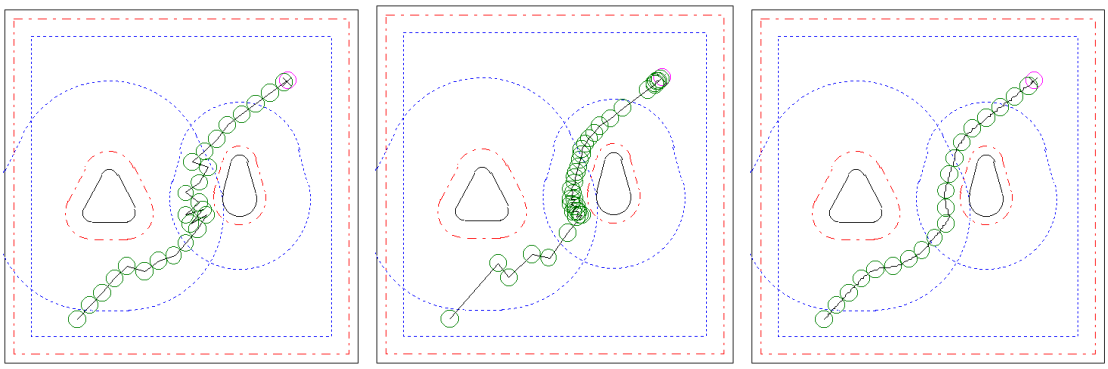


Figura 5.2. Planificación Depth-First con Objetos Modelados con Poli-Esferas. *En los tres casos se utiliza la dirección del vector fuerza resultante para obtener la siguiente configuración del móvil, pero en con diferentes incrementos. En la gráfica de la izquierda, el vector unitario resulta demasiado grande, produciéndose oscilaciones en los entornos del obstáculo. En la gráfica central, el avance en cada segmento está acotado por la mínima distancia a los obstáculo y al destino, lo que produce movimientos largos en zonas despejadas y cortos en las proximidades de los obstáculos y el destino. En la gráfica de la derecha, donde el móvil se dibuja cada 10 iteraciones, se ha considerado un factor pequeño del vector unitario (0.1) con lo que el resultado es mucho más suave. En todos los casos se utiliza el mismo espacio de trabajo, con un móvil representado por una esfera, dos obstáculos con una bi-esfera y una tri-esfera, la posición inicial en la zona inferior izquierda y la posición final en la superior derecha. Los obstáculos (y las paredes de la zona de trabajo) se representan con su distancia de seguridad (marcada a trazo-punto) y su zona de influencia (dibujada a puntos)* [(*)] *. El obstáculo tri-esfera tiene un factor de repulsión y una zona de influencia mayor que el obstáculo representado con una bi-esfera, por lo que el móvil se acerca mucho menos al primer obstáculo que al segundo. En todos los casos, el potencial atractivo es cónico.*

5.2.4. Simplificación del Cálculo de Potenciales Repulsivos

Sea cual sea la técnica que se utilice en la planificación de movimientos, resulta evidente que es deseable obtener un resultado tan eficiente como sea posible, siendo el tiempo de cálculo uno de los parámetros de eficiencia que se puede considerar. La obtención del camino libre de colisiones basada en potenciales es un proceso iterativo; en cada iteración hay que calcular diferentes fuerzas, obtener la resultante de todas ellas y aplicar un método de planificación que permita obtener la configuración siguiente a partir del actual. La cantidad de fuerzas repulsivas que se calcule en cada paso depende del número de obstáculos. La siguiente idea pretende introducir una mejora temporal reduciendo el número de obstáculos a considerar durante la planificación de movimientos y, por tanto, reduciendo el número de fuerzas repulsivas que habría que

(*) Esta forma de representar la distancia de seguridad y la zona de influencia de los obstáculos y las paredes se mantiene en todas las gráficas de este capítulo (generadas todas con el mismo programa) mientras no se indique lo contrario.

calcular. La idea se basa en los algoritmos de cálculo de distancias incrementales, realizándose una poda en los obstáculos a considerar.

Para reducir el número de obstáculos a tener en cuenta en cada iteración, se tiene un array en el que, para cada obstáculo, se guarde la distancia que necesita recorrer el móvil para entrar en la zona de influencia de ese obstáculo. De esta forma el obstáculo puede no tenerse en cuenta mientras el móvil no haya realizado un desplazamiento similar a esta distancia. El array se inicializa cuando el móvil se encuentra en su posición inicial, almacenando para cada obstáculo la distancia que le separa del móvil y substrayendo a esta cantidad la distancia de influencia del obstáculo.

Al aplicar un método de planificación, se obtendrá una nueva configuración del móvil. Sabiendo el máximo desplazamiento realizado por un punto del móvil, se resta esta distancia a todos los elementos del array. En la próxima iteración sólo habrá que considerar aquellos obstáculos que tengan en la posición asociada en el array un valor negativo en el vector. En este caso hay que reinicializar el valor de este componente del array, como se hizo en principio. De esta forma se asegura que sólo se considera aquellos obstáculos en cuya zona de influencia puede haber entrado el móvil.

Como mejora de esta técnica, se puede ampliar la información que se guarda para cada obstáculo en el array. Si se introducen los vectores de colisión, se podrá conocer si el móvil se dirige hacia el obstáculo o se separa del mismo. El producto escalar del vector desplazamiento (en una iteración) por el vector de colisión unitario es la proyección del avance del móvil en el vector de colisión, es decir, la cantidad que se ha alejado (o acercado si resulta negativo) el móvil al obstáculo en la dirección del vector de colisión. Si en cada paso del algoritmo se suma esta cantidad a la almacenada en el array, de nuevo sólo habrá que considerar los obstáculos con valor negativo en el array, y en este caso, recalcular la nueva distancia y el nuevo vector de colisión. Así se acumulan acercamientos/alejamientos a cada obstáculo.

Sin embargo, debido a la rapidez con que se calculan las distancias entre modelos esféricos envolventes, la aplicación de esta técnica sólo ha presentado mejoras significativas para entornos con un número elevado de obstáculos (del orden de 50) con cierta complejidad en sus modelos (poli-esferas con un número alto de vértices esféricos, tanto para el móvil como para los obstáculos).

5.3. Potenciales con Rotación en el Espacio Cartesiano

La definición de campos potenciales atractivos y repulsivos de la Sección anterior implica la utilización de un espacio $\Re^m$ donde se mueve el robot o móvil considerado. Cuando los grados de libertad del robot son traslacionales, la definición anterior sirve para aplicar correctamente una planificación de sus movimientos, tanto en el espacio de configuraciones como en el espacio cartesiano.

Sin embargo, al considerar la rotación de un robot que se mueve en un espacio tridimensional(*) se suele trabajar en el espacio de configuraciones ([Z&H94]) donde se define la configuración en que se encuentra el robot mediante un vector de seis componentes, $\mathbf{q}=(q_x,q_y,q_z,q_\alpha,q_\beta,q_\gamma)$, donde las tres primeras representan la posición del origen de un sistema de referencia local al robot y las tres últimas suelen ser tres valores angulares para representar la orientación del sistema de referencia local respecto al fijo. Una solución extendida en robótica es utilizar los ángulos de Euler RYP (*roll-yaw-pitch*) que representan los tres ángulos de giro del sistema de referencia local OUVW respecto a los ejes principales X,Y,Z (en este orden) del sistema de referencia OXYZ del espacio de trabajo, pero cualquier otra representación de la orientación se puede calcular a partir de estos tres ángulos. Los tres giros de Euler RYP son equivalentes a tres giros respecto a los ejes principales del sistema de referencia local W,V,U (en este orden), ya que la matriz de rotación obtenida es:

$$R = R_{Z,\gamma} R_{Y,\beta} R_{X,\alpha} = R_{W,\gamma} R_{V,\beta} R_{U,\alpha}$$

Trabajar en este nuevo espacio de configuraciones implica convertir el móvil en un punto 6D y obtener C-obstáculos de los obstáculos, que se suelen representar por *slices* para rotaciones discretas [L-P83b]. Así es posible aplicar potenciales de traslación en el nuevo espacio, teniendo precaución con las distancias, puesto que los C-obstáculos no son convexos en general. El problema principal es obtener el gradiente de la distancia, es decir, el vector de colisión en el caso de traslación en el nuevo espacio.

Este problema se puede evitar trabajando directamente en el espacio cartesiano. Sin embargo, la existencia de grados de libertad rotacionales implica que se debe realizar una adaptación de la técnica de los campos potenciales. Una extensión natural se presentó en [HBC90], que contemplaba el efecto de giro debido a la acción de un potencial sobre el móvil mediante la utilización de momentos de giro. Su idea básica es muy intuitiva: si la planificación de movimientos con potenciales determina un camino hacia el destino usando la fuerza resultante para dirigir los movimientos traslacionales, también debería ser posible determinar los movimientos rotacionales considerando momentos de giro. Sobre esta base proponen un algoritmo para un espacio 2D donde se

(*) Para un espacio de trabajo bi-dimensional, se consideran sólo dos componentes de traslación y una de orientación, siendo el vector que representa una configuración del robot $\mathbf{q}=(q_x,q_y,q_\theta)$.

consideran dos términos independientes: por una parte, los movimientos traslacionales se fijan mediante fuerzas atractivas y repulsivas como si el móvil no tubiera rotación; por otra parte, el obstáculo más cercano al móvil genera una serie de momentos sobre un conjunto de puntos distribuidos uniformemente a lo largo del contorno del móvil cuya suma se utiliza para determinar el ángulo de giro del móvil. El momento de giro producido en cada uno de estos puntos se obtiene mediante la tercera componente del producto vectorial entre el vector de posición de ese punto respecto al centroide del móvil y el vector de colisión entre el móvil y el obstáculo más cercano.

Sin embargo, el algoritmo propuesto presenta los siguientes problemas:

- No es general para trabajar en 3D.
- La configuración destino no genera ningún momento de giro, con lo que resulta imposible forzar una orientación final a alcanzar.
- El momento generado en cualquier par de puntos equidistantes del centroide es igual en valor absoluto, independientemente de la mayor o menor cercanía de ese punto al obstáculo.
- La suma de momentos generados en puntos del contorno se cancela fácilmente, anulando posibles giros.
- El ángulo de giro obtenido en cada paso es proporcional únicamente al momento calculado, lo que hace difícil determinar una amplitud adecuada.
- El ángulo de giro no está acotado, por lo que un giro puede provocar un acercamiento del móvil al obstáculo.
- Sólo considera el obstáculo más cercano, sin tener en cuenta que una traslación o rotación podría producir una colisión con otro obstáculo.
- Es difícil acotar el movimiento del móvil de forma homogénea en traslación y rotación.
- No determina ninguna distancia de seguridad entre móvil y obstáculo.
- No determina una zona de influencia del obstáculo en el giro del móvil, por lo que éste siempre girará por efecto de algún obstáculo.

Para resolver estos problemas, a continuación se introduce una nueva definición de potenciales [M&T96] que utiliza las distancias calculadas directamente en el espacio cartesiano para definir las traslaciones y rotaciones en tres dimensiones provocadas por el destino o los obstáculos y acotar los desplazamientos realizados por el móvil mediante una representación homogénea.

5.3.1. Representación Homogénea de Posición y Orientación

Se desea disponer de una representación de la posición y orientación del móvil para la que sea cómoda acotar los desplazamientos realizados por una transformación que incluya traslación y rotación. Para ello, sería aconsejable disponer de dimensiones homogéneas y utilizar, por ejemplo, una norma euclídea. Además, debe permitir trabajar de forma vectorial donde un incremento en cada una de sus componentes equivalga al desplazamiento producido por el cambio de un grado de libertad.

En el espacio cartesiano 3D donde se mueve el móvil, éste dispone de un sistema de referencia local OUVW (no necesariamente dentro de él) cuyo origen fija su posición **q** en el espacio y sobre cuyos ejes normales se aplican las rotaciones. La forma del móvil vendrá definida mediante una poli-esfera de vértices esféricos $s_0,...,s_n$, cuyas componentes se expresan respecto al sistema de referencia local.

Es evidente que si se aplica una traslación al móvil determinada por un vector **v**, el máximo desplazamiento de un punto del móvil (de todos ellos) producido por esta traslación es $\|\mathbf{v}\|$.

Cuando se aplica una rotación sobre el eje U del sistema de referencia local del móvil, el máximo desplazamiento producido por este giro viene dado por el punto del móvil más alejado del eje U. Este punto, o más concretamente, la distancia de este punto al eje se puede obtener según:

$$d_U = \max_{j=0,...,n}\left\{\sqrt{\left(s_j.y - \mathbf{q}.y\right)^2 + \left(s_j.z - \mathbf{q}.z\right)^2} + s_j.r\right\}$$

Si el giro producido sobre el eje U es un ángulo α, el máximo desplazamiento lineal provocado en un punto del móvil es αd_U.

Igualmente, las máximas distancias de un punto del móvil a los ejes V y W son:

$$d_V = \max_{j=0,...,n}\left\{\sqrt{\left(s_j.x - \mathbf{q}.x\right)^2 + \left(s_j.z - \mathbf{q}.z\right)^2} + s_j.r\right\}$$
$$d_W = \max_{j=0,...,n}\left\{\sqrt{\left(s_j.x - \mathbf{q}.x\right)^2 + \left(s_j.y - \mathbf{q}.y\right)^2} + s_j.r\right\}$$

dando βd_V y γd_W como los máximos desplazamientos lineales provocados por un punto del móvil al girar ángulos β y γ sobre los ejes V y W respectivamente.

Una representación del estado en que se encuentre el móvil según los objetivos deseados es la siguiente:

$$\mathbf{Q} = \begin{pmatrix}\mathbf{q} & d_U\alpha & d_V\beta & d_W\gamma\end{pmatrix}^T$$

La ventaja de introducir estas tres nuevas componentes en vez de considerar directamente los ángulos de rotación como suele ser habitual, es que al representar el máximo desplazamiento realizado por un punto del móvil producido por un giro del mismo, se puede normalizar con las componentes de desplazamiento traslacional del móvil, es decir, se puede acotar el máximo desplazamiento realizado en el móvil por efecto de una transformación (cambio de posición y orientación).

Ahora el estado destino que se desea alcanzar será:

$$\mathbf{Q} = \begin{pmatrix} \mathbf{q}_d & d_U\alpha_d & d_V\beta_d & d_W\gamma_d \end{pmatrix}^T$$

Se define la resta entre dos vectores en este nuevo espacio homogéneo de configuraciones mediante la operación:

$$\mathbf{Q}_1 - \mathbf{Q}_2 = \begin{pmatrix} \mathbf{q}_1 - \mathbf{q}_2 & d_U(\alpha_1 - \alpha_2) & d_V(\beta_1 - \beta_2) & d_W(\gamma_1 - \gamma_2) \end{pmatrix}^T$$

con una resta angular definida para devolver siempre el menor ángulo dentro del intervalo $(-\pi,\pi)$:

$$\theta_1 - \theta_2 = \begin{cases} \theta_1 - \theta_2 + 2\pi & \text{si } \theta_1 - \theta_2 < -\pi \\ \theta_1 - \theta_2 - 2\pi & \text{si } \theta_1 - \theta_2 > \pi \\ \theta_1 - \theta_2 & \text{en otro caso} \end{cases}$$

Se define la norma de un vector en este espacio como:

$$\|\mathbf{Q}\| = \sqrt{\|\mathbf{q}\|^2 + (d_U\alpha)^2 + (d_V\beta)^2 + (d_W\gamma)^2}$$

Esta nueva representación homogénea del móvil permite definir potenciales atractivos y repulsivos con rotación directamente en el espacio cartesiano. Es importante resaltar que la información necesaria para realizar la planificaicón es el gradiente del potencial y no la propia función potencial, lo que va a simplificar su cálculo sobre todo en el caso de la rotación. Definiendo $-\nabla W_{atr}(\mathbf{Q})$ y $-\nabla W_{rep}(\mathbf{Q})$ como los gradientes de los potenciales atractivos y repulsivos generados por el destino y los obstáculos respectivamente, así como $-\nabla W(\mathbf{Q})$ como la suma de ambos, se puede realizar una planificación depth-first que permita obtener $\mathbf{Q}_{i+1}$ a partir de $\mathbf{Q}_i$ según:

$$\mathbf{Q}_{i+1} = \mathbf{Q}_i - \frac{\nabla W(\mathbf{Q})}{\|\nabla W(\mathbf{Q})\|}$$

La definición de $\nabla W_{atr}(\mathbf{Q})$ y $\nabla W_{rep}(\mathbf{Q})$, con especial atención al término de rotación en el caso repulsivo, se muestra en las dos secciones siguientes.

5.3.2. Potenciales Atractivos con Rotación

Con la utilización de esta nueva representación homogénea, el gradiente atractivo se puede definir con la misma filosofía cuando se considera la rotación que como se hizo cuando sólo se consideró la traslación, es decir, proporcional a la diferencia entre los vectores que representa las configuraciones actual y destino para el caso parabólico, dividiéndose por la norma en el caso cónico:

$$-\nabla W_{atr}^{P}(\mathbf{Q}) = -\varepsilon_P\left(\mathbf{Q}-\mathbf{Q}_d\right) \text{ y } -\nabla W_{atr}^{C}(\mathbf{Q}) = -\varepsilon_C \frac{\left(\mathbf{Q}-\mathbf{Q}_d\right)}{\left\|\mathbf{Q}-\mathbf{Q}_d\right\|}$$

Aunque su cálculo no resulte necesario, es fácil comprobar que para ambos casos se puede definir un potencial con el respectivo gradiente anterior, obteniendo:

$$W_{atr}^{P}(\mathbf{Q}) = \frac{\varepsilon_P}{2}\left\|\mathbf{Q}-\mathbf{Q}_d\right\|^2 \text{ y } W_{atr}^{C}(\mathbf{Q}) = \varepsilon_C\left\|\mathbf{Q}-\mathbf{Q}_d\right\|$$

La Figura 5.3. muestra dos ejemplos de planificación depth-first para un móvil con dos grados de libertad de traslación y uno de rotación en un espacio de trabajo 2D sin obstáculos en el que se ha definido un potencial atractivo parabólico con rotación.

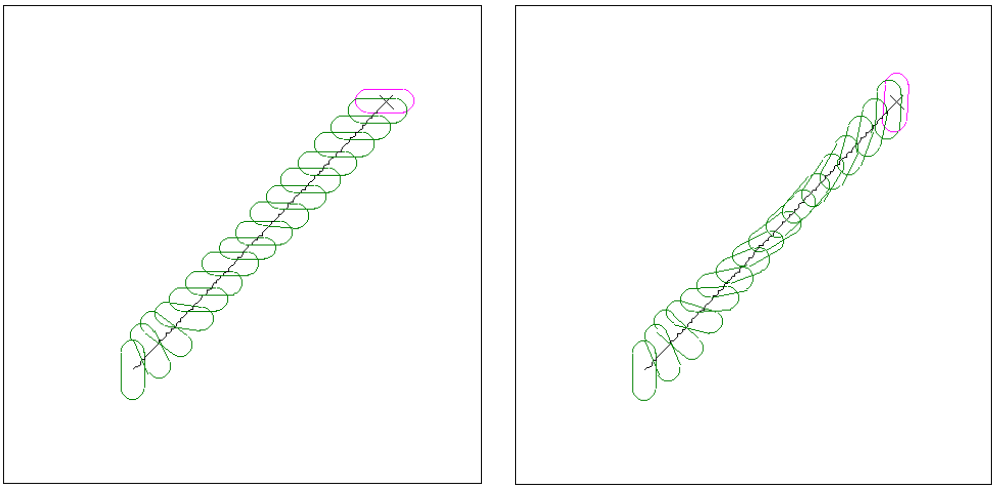


Figura 5.3. Aplicación del Potencial Atractivo con Rotación. *En un espacio de trabajo 2D sin obstáculos con un móvil representado por una bi-esfera que puede girar respecto a su centro, se ha definido un potencial atractivo parabólico con rotación. La configuración inicial se encuentra en la zona inferior izquierda y la destino en la superior derecha. En la gráfica de la izquierda, la posición destino está girada 90º respecto a la inicial y en la gráfica de la derecha, lo está 180º. El móvil se dibuja cada 5 iteraciones.*

5.3.3. Potenciales Repulsivos con Rotación

El gradiente del potencial repulsivo generado por nobs obstáculos será la suma de gradientes de potencial debido a cada uno de ellos:

$$-\nabla W_{rep}(\mathbf{Q}) = \sum_{obs=1}^{nobs} -\nabla W_{rep}^{obs}(\mathbf{Q})$$

La fuerza repulsiva producida por los obstáculos será la suma de las fuerzas que producen cada uno de ellos. La fuerza que produce un obstáculo está formada por dos sub-vectores:

$$-\nabla W_{rep}^{obs}(\mathbf{Q}) = \begin{pmatrix} -\nabla W_{rep,des}^{obs}(\mathbf{Q}) \\ -\nabla W_{rep,rot}^{obs}(\mathbf{Q}) \end{pmatrix}$$

donde $-\nabla W_{rep,des}^{obs}(\mathbf{Q})$ representa el desplazamiento provocado por el obstáculo mientras que $-\nabla W_{rep,rot}^{obs}(\mathbf{Q})$ va a suponer las rotaciones sobre los ejes del sistema de referencia local del móvil que genera el obstáculo.

El término $-\nabla W_{rep,des}^{obs}(\mathbf{Q})$ es la fuerza repulsiva ejercida por el obstáculo como si no hubiese rotación, siendo por tanto:

$$-\nabla W_{rep,des}^{obs}(\mathbf{Q}) = -\nabla U_{rep}^{obs}(\mathbf{q})$$

El término $-\nabla W_{rep,rot}^{obs}(\mathbf{Q})$ debe producir un giro respecto al punto de referencia del móvil tal que aleje el móvil del obstáculo, pero no siempre es posible definir este giro. Sean s_m y s_{obs} las esferas del móvil y obstáculos más cercanas (pueden no ser únicas) y el vector de colisión desde el obstáculo al móvil $\mathbf{v}_c=(s_m.c-s_{obs}.c)$. Estos valores son devueltos por el módulo del cálculo de distancias entre móvil y obstáculo. En los siguientes tres casos, cualquier giro del móvil disminuye la distancia entre móvil y obstáculo (Figura 5.4):

- La esfera s_m es un vértice esférico de la poli-esfera que representa al móvil, definiendo una línea que une su centro y el centro de s_{obs}, y el punto de referencia $\mathbf{q}_0$ cumple que el vector $(s_m.c-\mathbf{q}_0)$ es paralelo con el mismo sentido que $\mathbf{v}_c$.

- Existe una arista esférica del móvil paralela al obstáculo (a uno de sus vértices, aristas o planos esféricos), con lo que s_m, que puede no ser única, pertenece a una bi-esfera S_{01} de eje $\mathbf{v}_{01}$ y se cumple que la proyección en la dirección $\mathbf{v}_c$ (en su sentido o

el opuesto) del punto de referencia $\mathbf{q}_0$ sobre la línea definida por $\mathbf{v}_{01}$ cae dentro del eje de la bi-esfera.

- Existe una cara esférica del móvil paralela al obstáculo, con lo que s_m, que puede no ser única, pertenece a una tri-esfera y se cumple que la proyección en la dirección $\mathbf{v}_c$ (en su sentido o el opuesto) del punto de referencia $\mathbf{q}_0$ sobre el plano de centros de la tri-esfera cae dentro del triángulo definido por los centros de las esferas vértices de la tri-esfera.

Para estos tres casos, en los que no se debe producir ningún giro del móvil, se dice que el móvil se encuentra en *equilibrio de orientación* respecto al obstáculo considerado, siendo los casos ideales de orientación entre móvil y obstáculo, es decir, aquellos a los que debe tender el giro producido en el móvil por el obstáculo.

Cuando el móvil no se encuentra en equilibrio de orientación con un obstáculo, se puede definir, a partir de $\mathbf{v}_c$ y $\mathbf{v}_r=(s_m.c-\mathbf{q}_0)$, considerando la esfera s_m más cercana a $\mathbf{q}_0$ cuando hay más de una, el eje del giro $\mathbf{r}_\mathbf{Q}$ según:

$$\mathbf{r}_\mathbf{Q} = \mathbf{v}_r \wedge \mathbf{v}_c$$

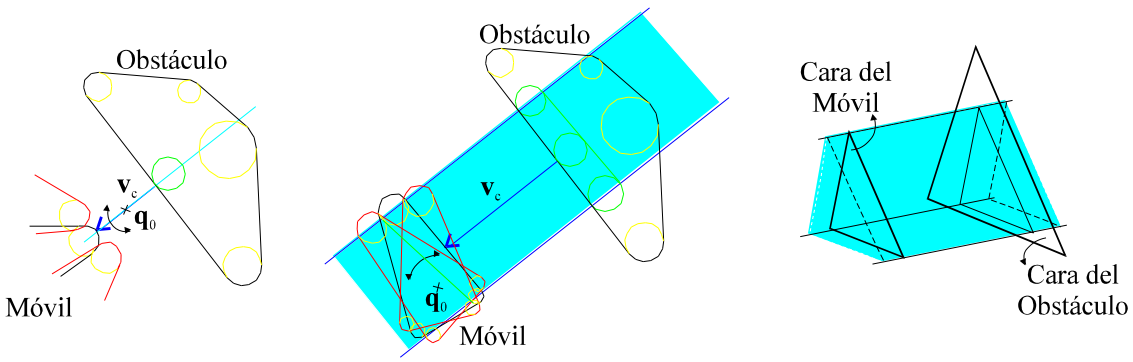


Figura 5.4. Situaciones en las que un Giro Acerca el Móvil al Obstáculo. *En la gráfica de la izquierda, el móvil se acerca al obstáculo si se produce un giro respecto a* $\mathbf{q}_0$*, situado sobre la línea que define la mínima distancia entre móvil y obstáculo. En la gráfica central, donde el móvil y el obstáculo tienen aristas esféricas paralelas, se muestra sombreada la zona en que si se encuentra* $\mathbf{q}_0$*, el giro produce un acercamiento del móvil al obstáculo, ilustrándose un ejemplo. En la gráfica de la derecha se resalta la misma zona cuando el móvil y el obstáculo tienen caras esféricas paralelas.*

El plano de giro será aquél que tiene $\mathbf{r}_\mathbf{Q}$ como vector normal y pasa por $\mathbf{q}_0$. El máximo giro permitido es aquél que lleva el móvil a una situación de equilibrio de orientación con el obstáculo. Para determinar este máximo giro, se realiza una sección del móvil por el plano de giro y se calcula, para todas las aristas esféricas, la esfera cuyo centro es intersección del eje de la arista con el plano de giro. Todas estas esferas, incluida la

esferas s_m, se ordenan en el sentido de las agujas del reloj visto desde el eje $\mathbf{r_Q}$ y a partir de un punto del interior del móvil (por ejemplo, s_m.c). Se considera la esfera s_a anterior según esta ordenación a la esfera s_m. De esta forma se define la bi-esfera S_{am} con vértices esféricos s_a y s_m que tiene como ángulo de convergencia γ_a y como eje $\mathbf{v}_a$. El ángulo que forma este eje con el vector de colisión es $\beta = \cos^{-1}\left(\hat{\mathbf{v}}_c \cdot \hat{\mathbf{v}}_a\right)$. Entonces el ángulo máximo de giro se define como $\theta_{max}=\beta+\gamma_a-\pi/2$. Este ángulo es el que fuerza a la bi-esfera S_{am} a ser perpendicular al vector esférico $\mathbf{v}_c$ (Figura 5.5).

Hay una situación, complementaria al primer caso de equilibrio en orientación, donde el eje de giro no está definido, al ser los ejes $\mathbf{v}_r$ y $\mathbf{v}_c$ paralelos y con el mismo sentido. Para este caso se debe tomar como eje de giro, aquél perpendicular a $\mathbf{v}_c$ que permita obtener el mayor θ_{max} posible.

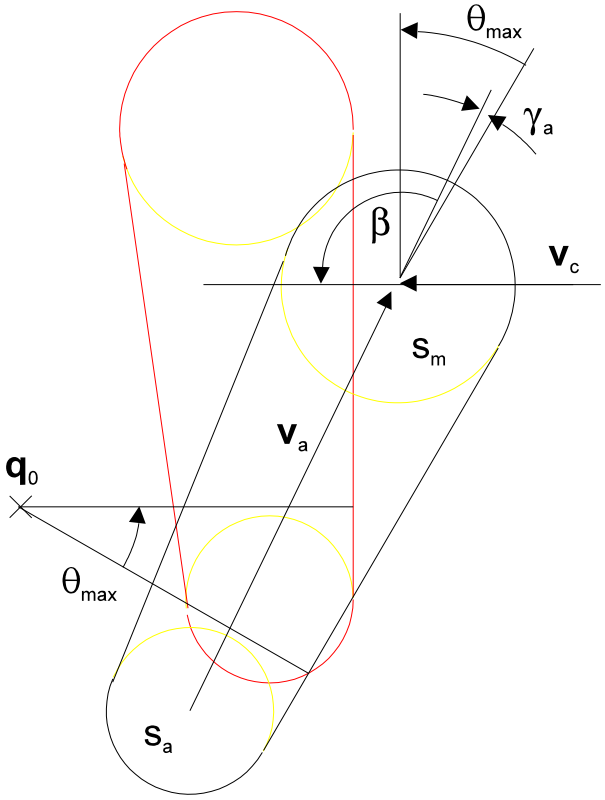


Figura 5.5. Determinación del Máximo Ángulo de Giro. *Tras realizar la sección del móvil con el plano de giro y obtener y ordenar las esferas con centro en este plano, se determina la esfera* s_a *anterior a la esfera* s_m*, definiéndose la bi-esfera* S_{am}*. A partir de esta bi-esfera es fácil calcular el máximo ángulo de giro como aquel ángulo de giro respecto a* $\mathbf{q}_0$ *que situa a la bi-esfera perpendicular al vector de colisión* $\mathbf{v}_c$*.*

Para la definición de un buen potencial, el gradiente, es decir, el giro producido, debe ser nulo cuando el móvil se encuentre fuera de la zona de influencia del obstáculo, para ir incrementando según el móvil esté más cerca al obstáculo. El giro máximo θ_{max} debe ocurrir sólo cuando el móvil se encuentre tocando la zona de seguridad del obstáculo. Para casos intermedios, se puede definir cualquier tipo de interpolación (lineal, cuadrática, cúbica, ...). Por ejemplo, utilizando una interpolación lineal, el ángulo de giro que debe aplicarse al móvil se define según:

$$\theta(\mathbf{Q}) = \begin{cases} 0 & \text{si } d_{inf}^{obs} \leq \left\| \mathbf{q} - \mathbf{q}_{obs} \right\| \\ \dfrac{d_{inf}^{obs} - \left\| \mathbf{q} - \mathbf{q}_{obs} \right\|}{d_{inf}^{obs} - d_{seg}^{obs}} \theta_{max} & \text{si } d_{seg}^{obs} \leq \left\| \mathbf{q} - \mathbf{q}_{obs} \right\| \leq d_{inf}^{obs} \\ \theta_{max} & \text{si } \left\| \mathbf{q} - \mathbf{q}_{obs} \right\| \leq d_{seg}^{obs} \end{cases}$$

El ángulo de giro es positivo, ya que el sentido de giro viene dado por el sentido del eje $\mathbf{r_Q}$. Este giro respecto a un eje arbitrario $\mathbf{r_Q}$ expresado respecto al sistema de referencia local del móvil, se puede representar mediante tres ángulos de Euler (α_r,β_r,γ_r), obteniendo finalmente:

$$-\nabla W_{rep,rot}^{obs}(\mathbf{Q}) = \begin{pmatrix} d_U \alpha_r \\ d_V \beta_r \\ d_W \gamma_r \end{pmatrix}$$

La Figura 5.6 muestra cinco ejemplos de la aplicación del potencial repulsivo con rotación definido de esta forma en un espacio de trabajo 2D con un móvil bi-esfera que dispone de un grado de libertad de rotación y dos de traslación. En los dos últimos ejemplos, se puede observar la buena respuesta que presenta esta técnica para entrar en el pasillo formado por dos obstáculos y para bordear obstáculos, ya que el giro pretende siempre que el móvil sea paralelo a los obstáculos.

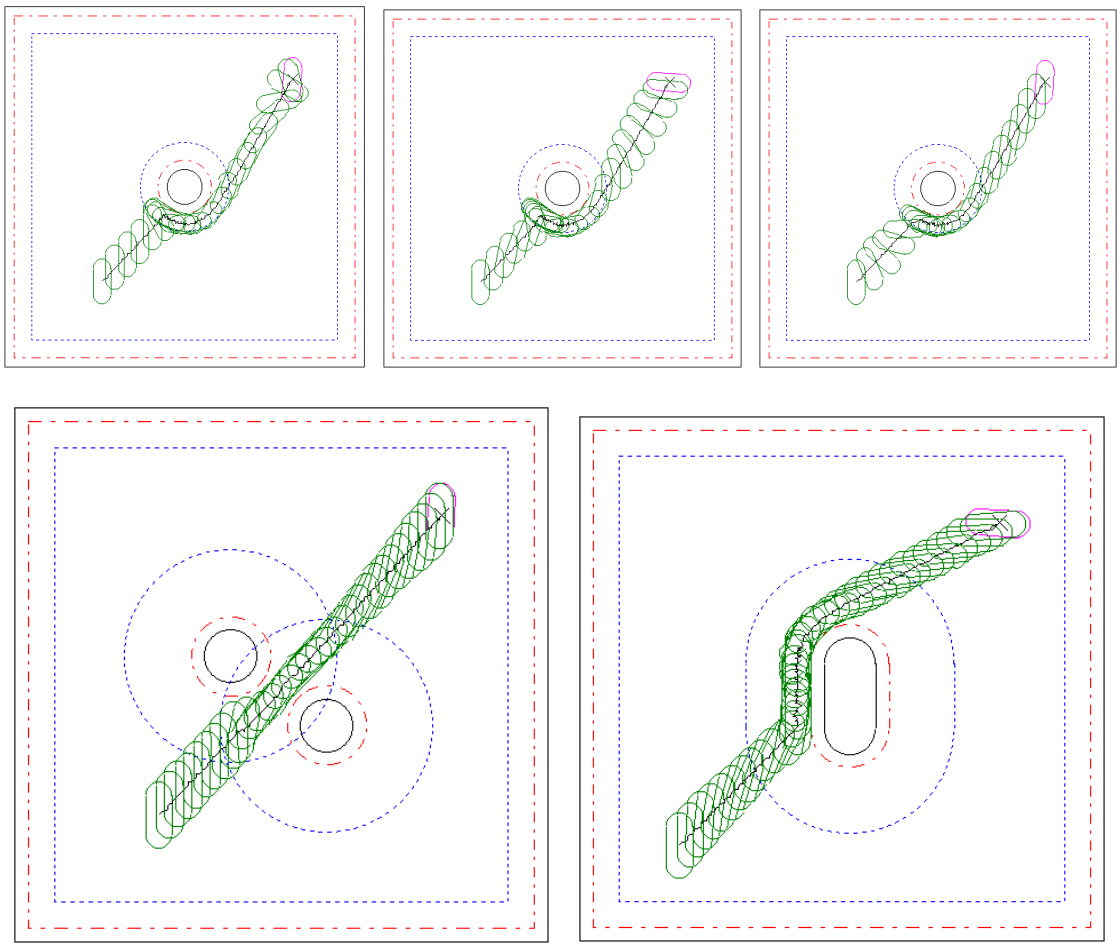


Figura 5.6. Aplicación del Potencial Repulsivo con Rotación. *En un espacio de trabajo 2D con un móvil representado por una bi-esfera que puede girar respecto a su centro, se ha definido un potencial atractivo parabólico con rotación y un potencial repulsivo con rotación. La configuración inicial se encuentra en la zona inferior izquierda y la destino en la superior derecha. En la gráfica superior izquierda, la posición destino tiene la misma orientación que la inicial, en la gráfica superior central está girada 90º respecto a la inicial y en la gráfica superior derecha, lo está 180º. En la gráfica inferior izquierda se puede observar la respuesta que presenta para entrar a pasar entre dos obstáculos esféricos que se encuentran en su camino. En la gráfica inferior derecha, se puede observar cómo el bordeo o seguimiento del contorno de un obstáculo de tipo bi-esfera presenta características muy adecuadas, ya que el efecto del potencial repulsivo de giro fuerza al móvil a desplazarse paralelamente al obstáculo. En todos los casos, cuando el móvil está fuera de la zona de influencia de los obstáculos, los giros producidos, si los hay, son debidos al potencial atractivo con rotación debido al cambio de orientación del destino. En las gráficas superiores el móvil se dibuja cada 10 iteraciones, cada 5 en las inferiores.*

5.4. Planificación Mediante Técnicas de Optimización Clásicas

La planificación de movimientos mediante campos potenciales artificiales es un problema de optimización donde la configuración del móvil debe buscar un mínimo. Además se debe aprovechar que el gradiente está disponible, por lo que los métodos de optimización tradicionales basados en el gradiente se pueden aplicar para resolver este problema. En concreto, a continuación se van a introducir la aplicación de los métodos de Davis, Swan y Campey, basado en el gradiente como técnica depth-first o *steepest descendent*, de Newton, basado en el Hessiano y de Davidon que conjuga ambos métodos [M&T96]. Estas técnicas van a mejorar uno de los problemas típicos de los campos potenciales: la oscilación que se produce al recorrer pasillos entre obstáculos [K&B91].

5.4.1. Aplicación del Método de Davis, Swan y Campey

El método de Davis, Swan y Campey (DSC) [Bur74] es un método de optimización en la dirección de la fuerza resultante. Considerando el incremento de movimiento variable:

$$\mathbf{Q}'(\delta) = \mathbf{Q} - \delta \frac{\nabla U(\mathbf{Q})}{\left\| \nabla U(\mathbf{Q}) \right\|}$$

se pretende encontrar un mínimo del potencial $U(\mathbf{Q})$ en la dirección $-\nabla U(\mathbf{Q})$. Para ello se prueban valores incrementales de δ hasta que se obtengan tres valores δ_1, δ_2 y δ_3 de forma que se garantice que incluye un mínimo, es decir:

$$U(\mathbf{Q}'(\delta_1)) > U(\mathbf{Q}'(\delta_2)) < U(\mathbf{Q}'(\delta_3))$$

Para conseguir estos valores δ_1, δ_2 y δ_3, teniendo en cuenta que la fuerza resultante apunta en una dirección decreciente en el potencial, se puede utilizar el algoritmo siguiente, con el que se consigue acotar la zona en que se encuentra el mínimo del potencial en la dirección de la fuerza resultante en un intervalo de longitud $2^i H$. El δ óptimo a considerar para el movimiento en la dirección de la fuerza resultante, teniendo en cuenta que los puntos generados por el algoritmo son equidistantes, es:

$$\delta_{\min} = \tfrac{1}{2}\left(\delta_1 + \delta_2 + \delta_3\right) - \tfrac{1}{2}\left(\frac{\delta_1 U(\mathbf{Q}'(\delta_1)) + \delta_2 U(\mathbf{Q}'(\delta_2)) + \delta_3 U(\mathbf{Q}'(\delta_3))}{U(\mathbf{Q}'(\delta_1)) + U(\mathbf{Q}'(\delta_2)) + U(\mathbf{Q}'(\delta_3))} \right)$$

Un algoritmo que encuentra el valor de este δ_{min} óptimo es el siguiente:

$\lambda_0 = 0$
$U_0 = U(\mathbf{Q})$
repetir
$\lambda_{i+1} = \lambda_i + 2^i H$
$U_{i+1} = U(\mathbf{Q}'(\lambda_{i+1}))$
hasta $U_{i+1} \geq U_i$
$\lambda'_{i+1} = \lambda_{i+1} - 2^{i-1} H$
$U'_{i+1} = U(\mathbf{Q}'(\lambda'_{i+1}))$
si i=0 *(caso a)*

$$\delta_{min} = 3H - \frac{1}{4}\frac{HU'_1 + 2HU_1}{U_0 + U'_1 + U_1}$$

sino, si $U'_{i+1} < U_i$ *(caso b)*

$$\delta_{min} = \frac{3\lambda'_{i+1}}{2} - \frac{1}{2}\frac{\lambda_i U_i + \lambda'_{i+1} U'_{i+1} + \lambda_{i+1} U_{i+1}}{U_i + U'_{i+1} + U_{i+1}}$$

sino *(caso c)*

$$\delta_{min} = \frac{3\lambda_i}{2} - \frac{1}{2}\frac{\lambda_{i-1} U_{i-1} + \lambda_i U_i + \lambda'_{i+1} U'_{i+1}}{U_{i-1} + U_i + U'_{i+1}}$$

Para ilustrar esta técnica, la gráfica de la Figura 5.7 muestra lo que podría ser una sección transversal de la superficie que representa el campo potencial total. La sección se tomaría en base al vector que marca la fuerza resultante, esto es, la dirección de la máxima pendiente. Un problema de esta técnica, ilustrado en esta Figura, es que se puede pasar por alto un máximo local (obstáculo) buscando un mínimo local. Ello se produce si $U_i < U_{i+1}$ cuando existe un obstáculo entre $\mathbf{Q}(\lambda_i)$ y $\mathbf{Q}(\lambda_{i+1})$, ya que los valores obtenidos son discretos. Para resolver esta situación hay que acotar el máximo desplazamiento en la dirección resultante mediante la distancia al obstáculo más próximo, es decir, acotar el incremento teniendo en cuenta la desigualdad:

$$\sum_{j=0}^{i} 2^j H < \left\| \mathbf{Q} - \mathbf{Q}_{obs} \right\| - d_{seg}^{obs}$$

Si en ese tramo no se ha garantizado la existencia de un mínimo, se toma el máximo avance posible en esa dirección (distancia al obstáculo más próximo). Para evitar aquellos obstáculos que no se encuentran en la dirección de la fuerza resultante, una posible mejora consiste en considerar la mínima distancia a los obstáculos que se encuentran en esa dirección.

La elección del valor de H puede resultar crítica: debe ser función de la distancia al destino o al obstáculo más próximo (en este caso, cuando se esté cerca de un obstáculo, los movimientos serán pequeños).

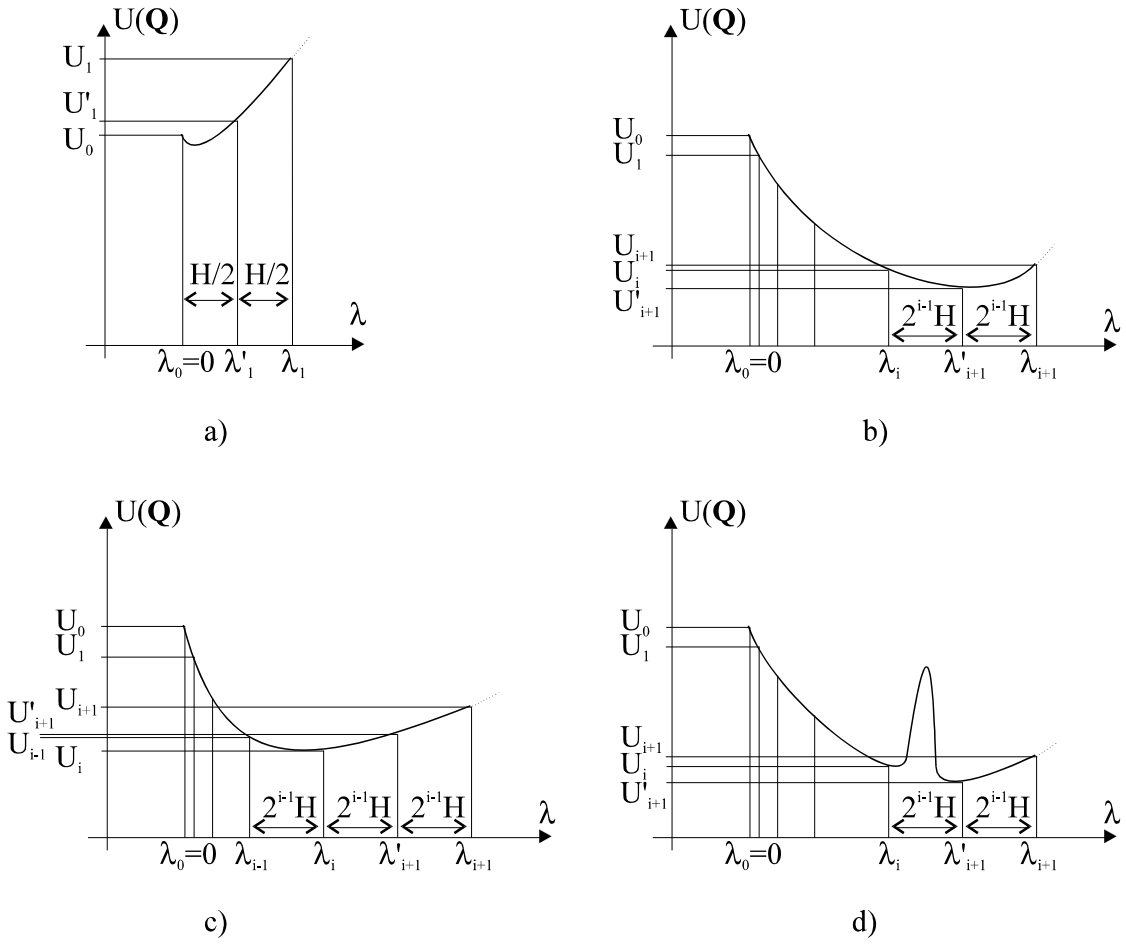


Figura 5.7. Acotación del Mínimo en la Dirección del Gradiente. *Las gráficas a, b y c muestran los tres casos posibles representados en el algoritmo. En a se obtiene un primer valor para* λ=H *mayor que el potencial inicial, estando el mínimo ya acotado. En b tras sucesivos valores descendentes, se acota el mínimo entre* λ_i *y* λ_{i+1}*. En el tercer caso, mostrado en c, resulta necesario volver un paso atrás para tener el mínimo acotado. En d se muestra la posibilidad de pasarse por alto un máximo local, que representa un obstáculo, ya que los valores* U_i *obtenidos son los mismos que en el caso b, por lo que se corre el peligro de suponer que hay un mínimo dentro del obstáculo.*

Esta técnica de optimización basada en el gradiente resulta adecuada *lejos* del destino y los obstáculos (se pueden realizar grandes avances), pero suele ser muy lenta, ya que para cada paso del algoritmo hay que evaluar varias veces la función potencial. La Figura 5.8 compara esta técnica con la técnica de planificación depth-first. En el segundo ejemplo se puede observar la reducción de oscilaciones producidas en el móvil al recorrer un pasillo estrecho.

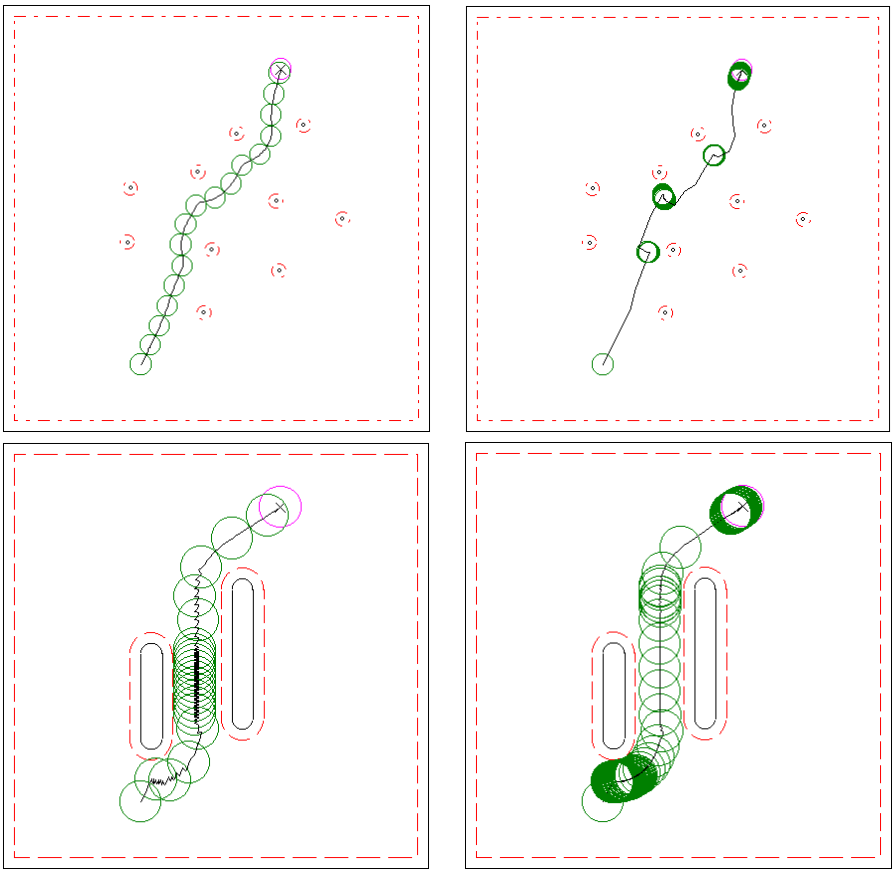


Figura 5.8. Planificación Mediante la Optimización de Davis, Swan y Campey. *El espacio de trabajo es 2D con un móvil modelado con una esfera que debe ir de su configuración inicial (posición inferior) a una destino (posición superior). En el ejemplo superior, donde hay 10 esferas de obstáculos, se compara la técnica depth-first (gráfica izquierda) y la optimización DSC (gráfica derecha), pudiendo observarse que esta última realiza grandes avances en zonas despejadas y pequeños desplazamientos cerca de los obstáculos o el destino. En el ejemplo inferior, donde dos bi-esferas forman un pasillo, la aplicación de la optimización DSC (derecha) no presenta oscilaciones en el pasillo, a diferencia de la técnica depth-first (izquierda). En ambos casos, el número de iteraciones del proceso se ha multiplicado por 10 y el tiempo de cálculo por 100 al utilizar la técnica DSC. En el caso superior, la longitud del camino encontrado es un 10% mayor para la técnica DSC, pero en el segundo ejemplo, es casi el 45% que el de la técnica depth-first al evitarse las oscilaciones. En todos las gráficas el móvil se dibuja cada 10 configuraciones calculadas.*

5.4.2. Aplicación del Método de Newton

El método de Newton [Bur74] se trata de otro método de optimización en el que la dirección en la que se moverá el móvil depende del gradiente y del inverso del Hessiano, según la siguiente expresión:

$$\mathbf{q}_{i+1} = \mathbf{q}_i + \delta \frac{\mathbf{J}^{-1}(\mathbf{q}_i)\mathbf{F}(\mathbf{q}_i)}{\left\|\mathbf{J}^{-1}(\mathbf{q}_i)\mathbf{F}(\mathbf{q}_i)\right\|}$$

con δ una constante de proporcionalidad. Como se puede observar, el movimiento no siempre se realizará en la dirección de la fuerza resultante.

Para un móvil con traslación, el Hessiano del vector fuerza se define como:

$$\mathbf{J}(\mathbf{q}_i) = \begin{pmatrix} \partial \mathbf{F}_x(\mathbf{q}_i) / \partial x & \partial \mathbf{F}_x(\mathbf{q}_i) / \partial y & \partial \mathbf{F}_x(\mathbf{q}_i) / \partial z \\ \partial \mathbf{F}_y(\mathbf{q}_i) / \partial x & \partial \mathbf{F}_y(\mathbf{q}_i) / \partial y & \partial \mathbf{F}_y(\mathbf{q}_i) / \partial z \\ \partial \mathbf{F}_z(\mathbf{q}_i) / \partial x & \partial \mathbf{F}_z(\mathbf{q}_i) / \partial y & \partial \mathbf{F}_z(\mathbf{q}_i) / \partial z \end{pmatrix}$$

Puesto que se utiliza el Hessiano, hay que calcular las derivadas parciales del vector fuerza. Ya que se tienen fuerzas atractivas y repulsivas, el Hessiano resultante será la suma de Hessianos:

$$\mathbf{J}(\mathbf{q}) = \mathbf{J}_{atr}(\mathbf{q}) + \sum_{obs=1}^{nobs} \mathbf{J}_{rep}^{obs}(\mathbf{q}) + \mathbf{J}_{min}$$

El Hessiano atractivo depende del potencial atractivo utilizado. Para un potencial atractivo cónico, el Hessiano tiene determinante nulo, por lo que no se puede obtener la inversa. Para un potencial atractivo parabólico, la fuerza atractiva es:

$$\mathbf{F}_{atr}(\mathbf{q}) = -\varepsilon_p \left(\mathbf{q} - \mathbf{q}_d\right)$$

Calculando las derivadas parciales:

$$\frac{\partial \mathbf{F}_{atr,q_x}(\mathbf{q})}{\partial \mathbf{q}_x} = \frac{\partial \mathbf{F}_{atr,q_x}(\mathbf{q})}{\partial \mathbf{q}_y} = \frac{\partial \mathbf{F}_{atr,q_z}(\mathbf{q})}{\partial \mathbf{q}_z} = -\varepsilon_p$$

$$\frac{\partial \mathbf{F}_{atr,q_x}(\mathbf{q})}{\partial \mathbf{q}_y} = \frac{\partial \mathbf{F}_{atr,q_x}(\mathbf{q})}{\partial \mathbf{q}_z} = \frac{\partial \mathbf{F}_{atr,q_y}(\mathbf{q})}{\partial \mathbf{q}_z} = 0$$

Por tanto, el Hessiano es:

$$\mathbf{J}_{\text{atr}}(\mathbf{q}) = \begin{pmatrix} -\varepsilon_{\text{P}} & 0 & 0 \\ 0 & -\varepsilon_{\text{P}} & 0 \\ 0 & 0 & -\varepsilon_{\text{P}} \end{pmatrix} \text{ con } \left|\mathbf{J}_{\text{atr}}(\mathbf{q})\right| = -\varepsilon_{\text{P}}^3$$

La fuerza repulsiva generada por un obstáculo sobre el móvil cuando éste se encuentra en su zona de influencia es:

$$\mathbf{F}_{\text{rep}}^{\text{obs}}(\mathbf{q}) = \eta_{\text{obs}}\left(\frac{1}{\|\mathbf{q}-\mathbf{q}_{\text{obs}}\|} - \frac{1}{\text{d}_{\text{inf}}^{\text{obs}}}\right)\frac{1}{\|\mathbf{q}-\mathbf{q}_{\text{obs}}\|^2}\frac{(\mathbf{q}-\mathbf{q}_{\text{obs}})}{\|\mathbf{q}-\mathbf{q}_{\text{obs}}\|} = \eta_{\text{obs}}\left(\frac{\mathbf{v}_{\text{c}}}{\|\mathbf{v}_{\text{c}}\|^4} - \frac{\mathbf{v}_{\text{c}}}{\text{d}_{\text{inf}}^{\text{obs}}\|\mathbf{v}_{\text{c}}\|^3}\right)$$

con $\mathbf{v}_{\text{c}}$ el vector de colisión que va del obstáculo al móvil. Las derivadas parciales son:

$$\frac{\partial \mathbf{F}_{\text{rep},q_x}^{\text{obs}}(\mathbf{q})}{\partial \mathbf{q}_{\text{x}}} = \eta_{\text{obs}}\left(\left(\mathbf{v}_{\text{c,x}}\right)^2 \text{A}(\mathbf{q}) + \text{B}(\mathbf{q})\right)$$

$$\frac{\partial \mathbf{F}_{\text{rep},q_y}^{\text{obs}}(\mathbf{q})}{\partial \mathbf{q}_{\text{y}}} = \eta_{\text{obs}}\left(\left(\mathbf{v}_{\text{c,y}}\right)^2 \text{A}(\mathbf{q}) + \text{B}(\mathbf{q})\right)$$

$$\frac{\partial \mathbf{F}_{\text{rep},q_z}^{\text{obs}}(\mathbf{q})}{\partial \mathbf{q}_{\text{z}}} = \eta_{\text{obs}}\left(\left(\mathbf{v}_{\text{c,z}}\right)^2 \text{A}(\mathbf{q}) + \text{B}(\mathbf{q})\right)$$

$$\frac{\partial \mathbf{F}_{\text{rep},q_x}^{\text{obs}}(\mathbf{q})}{\partial \mathbf{q}_{\text{y}}} = \frac{\partial \mathbf{F}_{\text{rep},q_x}^{\text{obs}}(\mathbf{q})}{\partial \mathbf{q}_{\text{z}}} = \frac{\partial \mathbf{F}_{\text{rep},q_y}^{\text{obs}}(\mathbf{q})}{\partial \mathbf{q}_{\text{z}}} = -\eta_{\text{obs}}\text{A}(\mathbf{q})\text{C}(\mathbf{q})$$

donde

$$\text{A}(\mathbf{q}) = \frac{3\|\mathbf{v}_{\text{c}}\| - 4\text{d}_{\text{inf}}^{\text{obs}}}{\text{d}_{\text{inf}}^{\text{obs}}\|\mathbf{v}_{\text{c}}\|^6}; \quad \text{B}(\mathbf{q}) = \frac{\text{d}_{\text{inf}}^{\text{obs}} - \|\mathbf{v}_{\text{c}}\|}{\text{d}_{\text{inf}}^{\text{obs}}\|\mathbf{v}_{\text{c}}\|^4}; \quad \text{C}(\mathbf{q}) = \mathbf{v}_{\text{c,x}}\mathbf{v}_{\text{c,y}}\mathbf{v}_{\text{c,z}}$$

El Hessiano repulsivo producido por un obstáculo en su zona de influencia es:

$$\mathbf{J}_{\text{inf}}^{\text{obs}}(\mathbf{q}) = \eta_{\text{obs}}\begin{pmatrix} \left(\mathbf{v}_{\text{c,x}}\right)^2 \text{A}(\mathbf{q}) + \text{B}(\mathbf{q}) & -\text{A}(\mathbf{q})\text{C}(\mathbf{q}) & -\text{A}(\mathbf{q})\text{C}(\mathbf{q}) \\ -\text{A}(\mathbf{q})\text{C}(\mathbf{q}) & \left(\mathbf{v}_{\text{c,y}}\right)^2 \text{A}(\mathbf{q}) + \text{B}(\mathbf{q}) & -\text{A}(\mathbf{q})\text{C}(\mathbf{q}) \\ -\text{A}(\mathbf{q})\text{C}(\mathbf{q}) & -\text{A}(\mathbf{q})\text{C}(\mathbf{q}) & \left(\mathbf{v}_{\text{c,z}}\right)^2 \text{A}(\mathbf{q}) + \text{B}(\mathbf{q}) \end{pmatrix}$$

Para que fuera de la zona de influencia no intervenga el obstáculo, el Hessiano repulsivo total del obstáculo se define como:

$$\mathbf{J}_{rep}^{obs}(\mathbf{q}) = \begin{cases} I_{3\times3} & si\ d_{inf}^{obs} < \left\|\mathbf{q}-\mathbf{q}_{obs}\right\| \\ \mathbf{J}_{inf}^{obs}(\mathbf{q}) & si\ d_{seg}^{obs} \le \left\|\mathbf{q}-\mathbf{q}_{obs}\right\| \le d_{inf}^{obs} \\ I_{3\times3} & si \left\|\mathbf{q}-\mathbf{q}_{obs}\right\| < d_{seg}^{obs} \end{cases}$$

Esta técnica de optimización basada en el Hessiano resulta adecuada cerca del destino y los obstáculos. En general, esta técnica presenta resultados muy similares a la técnica depth-first, pero el tiempo de cálculo aumenta al necesitar calcular el Hessiano y su inverso. La Figura 5.9 compara mediante dos ejemplos esta técnica con la técnica de planificación depth-first. En el segundo caso se puede observar la reducción de oscilaciones producidas en el móvil al recorrer un pasillo estrecho.

5.4.3. Aplicación del Método de Davidon

A partir de una idea original de Davidon se han desarrollado una serie de métodos que son los más utilizados como métodos del gradiente [PF+88]. Su planteamiento básico es muy simple, intentar aprovechar lo mejor de las dos técnicas anteriores y evitar lo peor de ellas. La técnica de DSC es lenta pero siempre decrece el valor de la función, sólo requiere la primera derivada y es bueno lejos del destino. La técnica de Newton converge rápidamente, es buena cerca del destino pero requiere las segundas derivadas y la inversa de este Hessiano. El método de Davidon busca comenzar como el DSC y terminar como el de Newton, pero utilizando sólo las primeras derivadas, de forma que la dirección de búsqueda varíe suavemente de un método a otro.

La actualización de la configuración se realiza en cada método según:

DSC: $\mathbf{Q}_{i+1} = \mathbf{Q}_i + \delta_{min}\mathbf{IW}(\mathbf{Q}_i)$

Newton: $\mathbf{Q}_{i+1} = \mathbf{Q}_i + \mathbf{J}^{-1}(\mathbf{Q}_i)\mathbf{W}(\mathbf{Q}_i)$

El método de Davidon considera un compromiso entre $\delta_{min}\mathbf{IW}(\mathbf{Q}_i)$ y $\mathbf{J}^{-1}(\mathbf{Q}_i)\mathbf{W}(\mathbf{Q}_i)$:

$$\mathbf{Q}_{i+1} = \mathbf{Q}_i + \delta_{min}\mathbf{H}(\mathbf{Q}_i)\mathbf{W}(\mathbf{Q}_i)$$

Se debe construir una secuencia de matrices $\mathbf{H}_i=\mathbf{H}(\mathbf{Q}_i)$ que sean simétricas, definidas positivamente y que tiendan a $\mathbf{J}^{-1}$ según $i\to\infty$. Para conseguir esta secuencia de matrices, se estima $\mathbf{H}_{i+1}$ a partir de la anterior con un cierto error $\mathbf{E}_i$:

$$\mathbf{H}_{i+1} = \mathbf{H}_i + \mathbf{E}_i$$

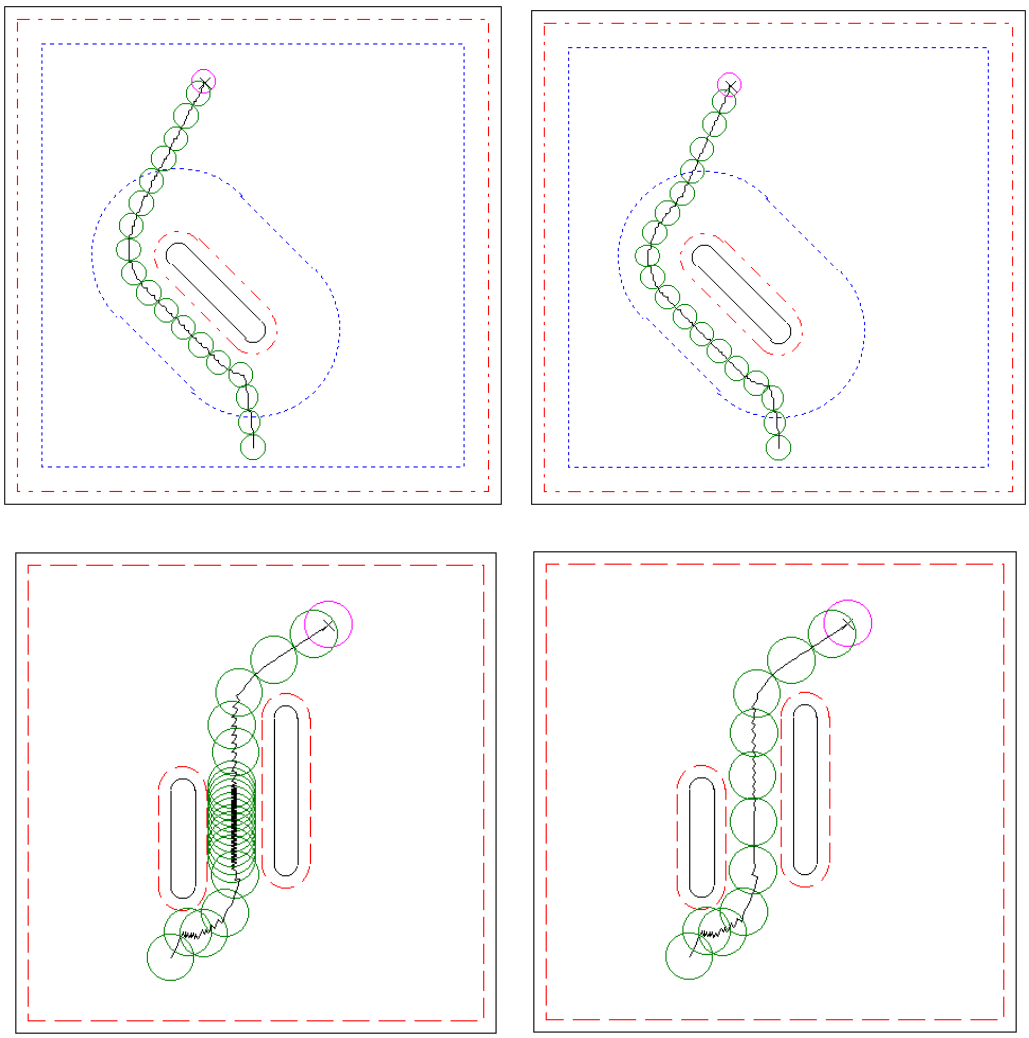


Figura 5.9. Planificación Mediante la Optimización de Newton. *El espacio de trabajo es 2D con un móvil modelado con una esfera que debe ir de su configuración inicial (posición inferior) a una destino (posición superior). En el ejemplo superior, donde hay una bi-esfera de obstáculos y el móvil se dibuja cada 10 configuraciones calculadas, se comprara la técnica depth-first (gráfica izquierda) y la optimización de Newton (gráfica derecha), pudiendo observarse que esta última reacciona antes ante la presencia de un obstáculo. En el ejemplo inferior, donde hay dos bi-esferas de obstáculos que forman un pasillo, y el móvil se dibuja cada 10 configuraciones calculadas, se compraran las técnicas depth-first (izquierda) y Newton (derecha), siendo resaltable que esta última técnica no presenta casi oscilaciones en el pasillo, a diferencia de la primera. En ambos casos, el número de iteraciones del proceso y la longitud del camino se mantienen similares, pero el tiempo de cálculo aumenta un 50 por 100 al utilizar la técnica de Newton, ya que se requiere calcular el Hessiano y su inverso.*

Una clase de soluciones a este problema viene dada por ([PF+88]):

$$\mathbf{E}_i(\alpha,\beta,\gamma) = -\frac{\mathbf{H}_i\mathbf{y}_i\mathbf{u}_i^T(\alpha)}{\mathbf{u}_i^T(\alpha)\mathbf{y}_i} + \gamma\frac{\mathbf{p}_i\mathbf{v}_i^T(\beta)}{\mathbf{v}_i^T(\beta)\mathbf{y}_i}$$

donde

$$\mathbf{y}_i = \mathbf{W}_{i+1} - \mathbf{W}_i, \quad \mathbf{p}_i = \mathbf{Q}_{i+1} - \mathbf{Q}_i$$
$$\mathbf{u}_i(\alpha) = \mathbf{p}_i + \alpha\mathbf{H}_i\mathbf{y}_i, \quad \mathbf{v}_i(\beta) = \mathbf{p}_i + \beta\mathbf{H}_i\mathbf{y}_i$$

por lo que se dispone de una familia de tres parámetros (α,β,γ). Se puede comprobar que para una función cuadrática de n variables se alcanza la convergencia al mínimo en n pasos siempre que las búsquedas lineales se realicen exactamente. Entre todas las posibles aproximaciones destacan los siguientes tres casos:

a) $\gamma = 1, \alpha = \infty, \beta = 0$:

$$\mathbf{E}_i^a = \frac{\mathbf{p}_i\mathbf{p}_i^T}{\mathbf{p}_i^T\mathbf{y}_i} - \frac{\mathbf{H}_i\mathbf{y}_i\mathbf{y}_i^T\mathbf{H}_i}{\mathbf{y}_i^T\mathbf{H}_i\mathbf{y}_i}$$

b) $\gamma = 1, \alpha = -1, \beta = -1$:

$$\mathbf{E}_i^b = \frac{(\mathbf{p}_i - \mathbf{H}_i\mathbf{y}_i)(\mathbf{p}_i - \mathbf{H}_i\mathbf{y}_i)^T}{(\mathbf{p}_i - \mathbf{H}_i\mathbf{y}_i)^T\mathbf{y}_i}$$

c) $\gamma = 1, \alpha = 0, \beta = -\left(1 + \frac{\mathbf{y}_i^T\mathbf{H}_i\mathbf{y}_i}{\mathbf{p}_i^T\mathbf{y}_i}\right)^{-1}$:

$$\mathbf{E}_i^c = \frac{1}{\mathbf{p}_i^T\mathbf{y}_i}\left[\left(1 + \frac{\mathbf{y}_i^T\mathbf{H}_i\mathbf{y}_i}{\mathbf{p}_i^T\mathbf{y}_i}\right)\mathbf{p}_i\mathbf{p}_i^T - \mathbf{p}_i\mathbf{y}_i^T\mathbf{H}_i - \mathbf{H}_i\mathbf{y}_i\mathbf{p}_i^T\right]$$

Un algoritmo para aplicar el método de Davidon es el siguiente:

$\mathbf{W}_0=\mathbf{W}(\mathbf{Q}_0)$; $\mathbf{H}_0=\mathbf{I}$
repetir
    Calcular δ_{min} *que minimiza* $U(\mathbf{Q}_i+\delta\mathbf{H}_i\mathbf{W}_i)$
    Calcular $\mathbf{Q}_{i+1}=\mathbf{Q}_i+\delta\mathbf{H}_i\mathbf{W}_i$ y $\mathbf{W}_{i+1}=\mathbf{W}(\mathbf{Q}_{i+1})$
    Calcular $\mathbf{y}_i=\mathbf{W}_{i+1}-\mathbf{W}_i$ y $\mathbf{p}_i=\mathbf{Q}_{i+1}-\mathbf{Q}_i$
    Actualizar $\mathbf{H}_{i+1}=\mathbf{H}_i+\mathbf{E}_i$ *según a, b o c*
hasta Convergencia

El problema que tiene este método es el cálculo de la minimización lineal para obtener δ_{min} en el primer paso del bucle. Se puede utilizar una técnica similar a la del método DSC, pero suele resultar bastante costoso para un paso intermedio del algoritmo. Una técnica relativamente sencilla que da buenos resultados es considerar en cada paso la secuencia δ_0=H y δ_{i+1}=0.1δ_i, hasta que se consiga una reducción significante en el potencial, y entonces realizar una interpolación cúbica para determinar una aproximación al mínimo δ_{min}. De nuevo la elección de H dependerá de la distancia al destino y al obstáculo más cercano.

En los ejemplos que se muestran en la Figura 5.10 se puede comparar las tres variantes de Davidon con las técnicas anteriores DSC y Newton. Para los tres casos el inicio es similar, actuando bien lejos del obstáculo (como DSC) y cerca del mismo (como Newton). Sin embargo, siempre presentan la misma respuesta al salir de la zona de influencia del obstáculo: mantiene la misma dirección de movimiento, tardando en reaccionar hacia el destino. Esto es debido a que para calcular la nueva dirección se utiliza la dirección anterior, realizando un filtrado o suavizado. De esta forma se puede concluir que la técnica de optimización de Davidon, pese a comportarse según lo esperado, no mejora las características de las técnicas DSC y principalmente de Newton, cuyo comportamiento es el más adecuado de los tres.

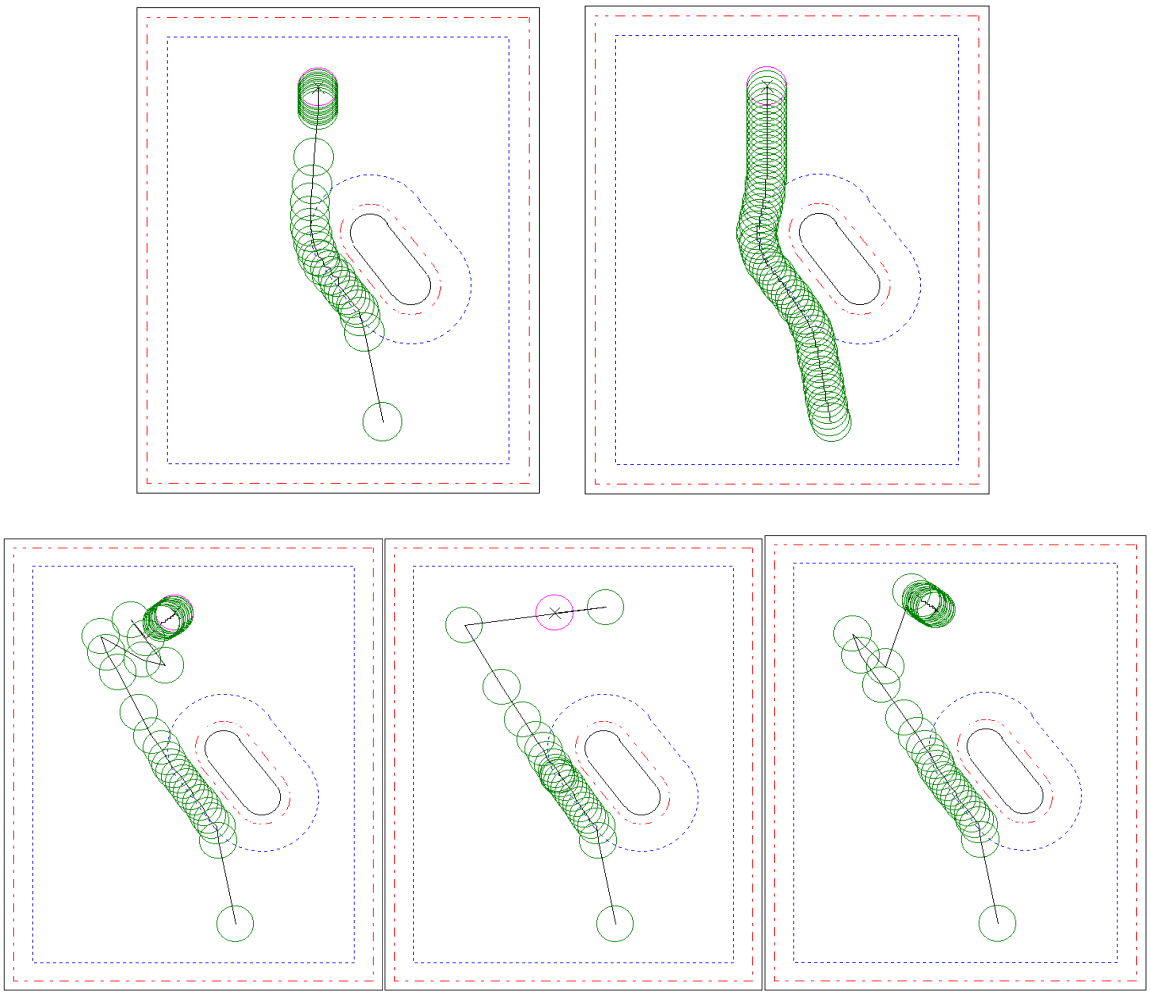


Figura 5.10. Planificación Mediante las Tres Variantes de Optimización de Davidon. *El espacio de trabajo es 2D con una bi-esfera como obstáculos y el móvil, modelado con una esfera, debe ir de su configuración inicial (posición inferior) a una destino (posición superior). En los dos ejemplos superiores, se muestra el resultado de aplicar la optimización con DSC (gráfica izquierda) y Newton (gráfica derecha). En los tres ejemplos de la parte inferior se muestran, de izquierda a derecha, las tres variantes de la optimización de Davidon. En los tres casos el inicio es similar, actuando bien lejos del obstáculo (como DSC) y cerca del mismo (como Newton). Sin embargo, siempre presentan la misma respuesta al salir de la zona de influencia del obstáculo: mantiene la misma dirección de movimiento, tardando en reaccionar hacia el destino. El móvil se dibuja en los cinco casos para todas las configuraciones calculadas.*

√

EPÍLOGO

En este epílogo se muestran diversos sistemas experimentales realizados con el objetivo de corroborar el trabajo desarrollado mediante unas aplicaciones prototipo, se presentan las conclusiones obtenidas en este trabajo y las principales aportaciones realizadas y, finalmente, se contempla una autocrítica al trabajo y ampliaciones futuras a realizar.

—¿Entonces ya no fabrican robots flexibles y adaptables como yo? —preguntó Andrew.
—No.
—Las investigaciones que he efectuado con motivo de mi libro —continuó diciendo Andrew— indican que soy el robot más antiguo actualmente en activo.
—El más antiguo ahora —dijo Smythe-Robertson— y el más antiguo que jamás existirá. Ningún robot es de utilidad alguna después de cumplidos los veinticinco años de vida. Los recuperamos y los sustituimos por modelos más modernos.

"El Hombre del Bicentenario". Isaac Asimov, 1976.

I. Plataformas Experimentales y Aplicaciones

Con el fin de poder aplicar la investigación realizada en el campo de la planificación de movimientos de sistemas robotizados, resulta necesario integrar los métodos desarrollados en sistemas de programación de robots para comprobar sus resultados. Si bien en los últimos años se empieza a considerar seriamente la necesidad de una estandarización en la programación de robots, hoy en día los robots industriales se programan en diferentes lenguajes, desarrollados por las propias compañías fabricantes de los robots. De hecho, la mayoría de fabricantes disponen de su propio lenguaje de programación, ejecutable directamente sobre la unidad de control. Estos sistemas suelen ser cerrados, dificultando enormemente la posibilidad de integrar módulos de detección de colisiones o de planificación de movimientos. Esto ha hecho que muchos de los trabajos realizados en el campo de la planificación de movimientos se queden en meras simulaciones, sin verificación sobre los sistemas reales.

Como consecuencia, en paralelo al trabajo de investigación expuesto en los capítulos anteriores, ha resultado necesario invertir un gran esfuerzo en el desarrollo de sistemas orientados a la programación (y simulación) de los robots disponibles en el Laboratorio de Robótica(*) del DISCA para utilizarse como plataformas experimentales. Todas las aplicaciones software han sido implementadas en C++ para entornos Windows. A continuación se realiza una breve explicación de los sistemas desarrollados, si bien una visión global con las características principales de la mayoría de ellos se puede encontrar en varias publicaciones ([M&V94], [M&V95] y [V&M95]). Tras la revisión de estos sistemas, se comentará su integración con los conceptos teóricos de planificación de movimientos desarrollados.

El primer paso para disponer de un sistema de programación abierto, con la posibilidad de integrar módulos específicos, consiste en evitar la programación del robot desde la unidad de control, permitiendo programarlo desde un ordenador externo. Para ello se ha desarrollado el sistema denominada WinArla [VMV93] que emula gráficamente el interfaz de las unidades de control S2 y S3E de los sistemas robotizados IRB de ABB, así como de sus unidades de programación. Con esta aplicación software se puede programar el sistema sobre un modelo virtual del robot en un PC, con una metodología similar a la que dispone la unidad de programación del sistema de control del robot real y utilizando el mismo lenguaje de programación, el ARLA, aunque actualmente se están desarrollando sistemas análogos que permitan programar los robots con otros lenguajes, tal como el VAL II, el RAPL o el formato propuesto como estándar internacional IRL.

(*) Robots industriales IRB L6 (cinco grados de libertad, sistema de control S2, motores DC, lenguaje de programación ARLA, herramienta múltiple con dos pinzas) e IRB 1500 (seis grados de libertad, sistema de control S3E, motores AC, lenguaje de programación ARLA, herramienta múltiple con tres pinzas) de ABB, vehículo autoguiado Robuter de Robosoft.

El programa dispone de los menús desplegables, iconos y botones necesarios que reemplazan a los dispositivos físicos reales. Las tres ventanas principales de la aplicación (Figuras E.1a-c) están diseñadas para que ofrezcan un aspecto lo más parecido posible a los dispositivos reales, esto es, a partir de iconos, marcos, botones, texto integrado, mapas de bits, etcétera. WinArla tiene su propio simulador gráfico del robot para poder moverlo con un interfaz semejante a la palanca de mando real y verificar los movimientos en la pantalla del robot. Esta filosofía de trabajo es comparable a la que utiliza ABB en su nuevo sistema S4 con programación desde PC, sacado al mercado el año 1995 (cabe resaltar que la primera versión de WinArla se desarrolló dos años antes).

El sistema ofrece la posibilidad de dos modos de trabajo: sin conexión con el robot y conectado al robot. La primera posibilidad permite una programación y simulación off-line, trabajando sobre un modelo del robot en pantalla. Por otro lado, la segunda posibilidad permite comunicar con el robot con el fin de llevar a cabo tareas como la creación, edición y corrección de programas, el aprendizaje por parte del robot de movimientos programados, la ejecución de programas, etcétera.

Otro sistema desarrollado en esta línea, denominado IntArla, es un intérprete de programas robot en lenguaje ARLA con un interfaz sencillo (Figura E.1d) que incluye un simulador 3D de los robots de la serie IRB de ABB. La aplicación toma como entrada dos ficheros ASCII, uno con el código fuente que puede incluir más de un programa ARLA con llamadas entre ellos y otro que define las localizaciones (posiciones y orientaciones) por las que debe pasar el robot. El código fuente es interpretado y ejecutado, avisando de cualquier error sintáctico que tenga el programa. Entre sus características principales se encuentran la posibilidad de simular la ejecución de un programa pasa o paso para la detección y corrección de errores y la ejecución de forma continua, tanto en simulación como en conexión con el robot.

Como segundo paso, de cara a evitar trabajar de una forma dependiente de la sintaxis de un lenguaje, se ha desarrollado un sistema, denominado SIPRAC (Sistema Integrado de Programación de Robots Asistida por Computador), que permite programar un robot en formato neutro a partir de la definición gráfica en un sistema CAD de la trayectoria que debe seguir la herramienta del robot [SMV96]. Además, con el interfaz desarrollado (Figura E.1.e) para este programa, se puede acceder al resto de aplicaciones ya comentadas, permitiendo una integración del software a través de los ficheros de programas robot con un formato neutro, TRY, similar a los estándares internacionales propuestos. Este formato incluye una cabecera de inicialización propia para cada robot y un conjunto de acciones y movimientos que se pueden ejecutar sobre el robot real seleccionado.

La generación de la trayectoria de la herramienta se puede realizar con cualquier programa CAD que permita salida a formato DXF, utilizando sus posibilidades gráficas con líneas, polilíneas, arcos, círculos, elipses, curvas *Bezier* y *splines*, etcétera. El fichero se interpreta en SIPRAC y se digitaliza, entendiendo por este proceso el cambio de curvas a polilíneas controlado por una cierta precisión. Para poder ejecutar esta trayectoria, se deben seleccionar parámetros de movimiento en cada tramo de la trayectoria, tales como velocidad, tipo de movimiento (libre, lineal, circular, ...),

precisión, etcétera. Un aspecto importante es la definición de la orientación de la herramienta, no incluida en la trayectoria inicial. Igualmente, después de cada movimiento básico, opcionalmente se pueden incluir acciones del tipo abrir/cerrar pinzas, esperas temporales, esperas a la activación de una entrada digital, tratamiento de salidas digitales, control de registros, ...

Para disponer de una herramienta que permita la programación y simulación desde un entorno gráfico se desarrolló el sistema RCad ([VMT91], [MTV92], [VMT92a] y [VMT92b]). Esta aplicación, desarrollada bajo el programa de Diseño Asistido por Ordenador *AutoCAD*, permite programar interactivamente y simular los movimientos de varios robots sobre un modelo gráfico 3D de los mismos en la pantalla de un ordenador (Figura E.1.f). El sistema realiza automáticamente las verificaciones oportunas (rangos, sintaxis, etcétera) para garantizar que los programas generados por dicha herramienta sean válidos a la hora de su posterior ejecución. Cuando resulte necesario, se puede verificar la programación sobre los robots reales. El sistema desarrollado resulta ser una herramienta de manejo y aprendizaje fácil de cara a la programación de robots industriales, ya que se apoya en las características gráficas propias de un sistema CAD, a la vez que permite disponer de un entorno agradable de trabajo para el programador.

Por último, la cinemática de los robots se resuelve mediante el sistema SimCef, orientado al modelado y la simulación geométrica y cinemática de sistemas robotizados. SimCef (Figura E.1.g) es una completa herramienta CAD especializada en el modelado de sistemas robotizados que permite la representación gráfica de objetos tridimensionales. Para ello implementa una serie de primitivas muy utilizadas a la hora de modelar estos sistemas, cómo son el prisma rectangular, la pirámide truncada, el cilindro, el cono truncado, la semiesfera, un elemento unión de un cilindro y una pirámide y un elemento compuesto de dos cilindros unidos. Los elementos suponen un nivel más en el proceso de modelado de un sistema robotizado: son modelados partiendo de primitivas y constituyen las partes lógicas (separadas mediante articulaciones) en las que podríamos dividir un robot. SimCef está pensado de forma que se pueda generar incluso una librería de elementos que puedan ser utilizados posteriormente en el modelado de sistemas completos. Finalmente, un sistema es modelado partiendo de una serie de elementos. Entre cada par de elementos se debe definir un eje de articulación, pudiendo ser ésta de revolución o prismática.

A partir del modelo geométrico del robot y de la definición de los ejes de articulación, SimCef calcula el modelo cinemático del sistema mediante la conocida representación de *Denavit-Hartenberg*. Una vez efectuado este cálculo, se pueden consultar y modificar las variables de cada articulación, definiendo movimientos mediante el encadenamiento de diferentes configuraciones. Cada vez que se ejecute la trayectoria del robot, ésta irá simulándose gráficamente en pantalla, pasando por todas las configuraciones almacenadas.

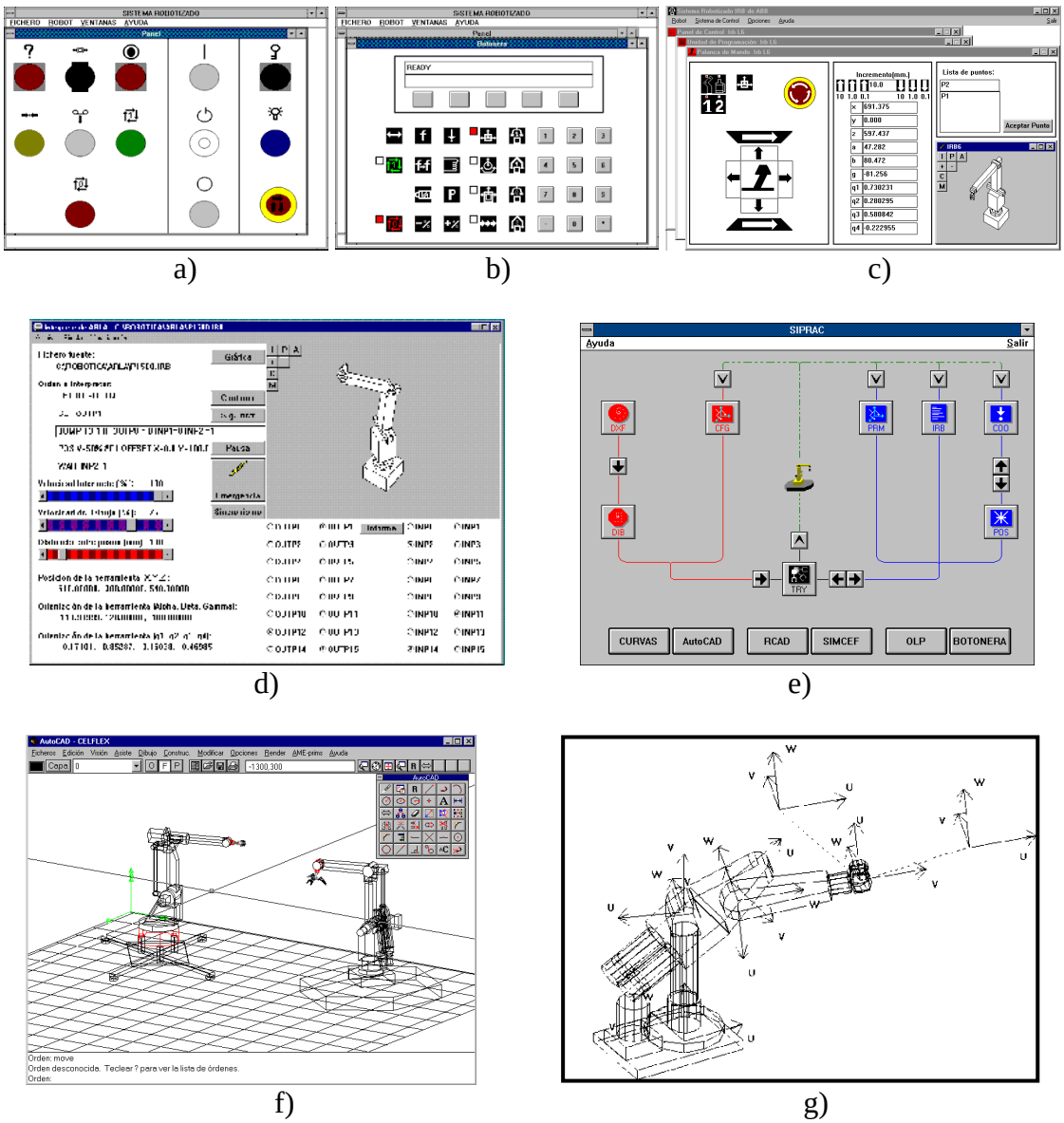


Figura E.1. Ejemplos de Plataformas Experimentales Desarrolladas. *En las gráficas* a, b *y* c *se muestran las tres ventanas principales de la aplicación WinArla, que permite realizar una programación del robot similar a como se realiza en la unidad de control. En* d *se muestra el interfaz del sistema IntArla, que interpreta, simula y ejecuta un programa robot en código ARLA. SIPRAC permite, mediante el interfaz mostrado en* e*, generar programas robot a partir de la definición gráfica de la tarea a realizar. En la gráfica* f *se muestra la utilización de la aplicación RCad a la programación y simulación de una célula multi-robot. Finalmente, en* g *se muestra un ejemplo del modelo geométrico y cinemático de un robot tipo Puma obtenido mediante el sistema SimCef.*

Para resolver la comunicación entre el ordenador y el robot, se ha desarrollado una librería de comunicaciones PC-robot, denominada CToolWin [ND+96]. La librería CToolWin, desarrollada para los robots de la serie IRB de ABB, es un librería de funciones que pueden ser llamadas por el usuario desde un programa C o C++ para facilitar la transferencia de datos entre el PC y la unidad de control del robot. Esta librería permite al usuario escribir aplicaciones en un lenguaje de alto nivel, accediendo a la unidad de control del robot a través del puerto serie del PC vía RS-232C.

El protocolo utilizado para el nivel de enlace de datos entre el PC y la unidad de control del robot es el *ADLP-10* (*ASEA Data Link Protocol*) cuya misión es garantizar la integridad de los datos que se transfieren en ambos sentidos. El protocolo del nivel de aplicación usado es el *ARAP* (*ASEA Robot Application Protocol*) cuyo propósito es definir los tipos y el formato de los mensajes que son intercambiados entre el PC y la unidad de control del robot durante la comunicación. Ambos protocolos son absolutamente transparentes al usuario, el cual simplemente dispone de una serie de funciones para la comunicación, obviando el funcionamiento a bajo nivel de la mismas.

Por encima de la librería se ha implementado un sistema que monitoriza y controla la comunicación del PC con los robots, a la vez que permite compartir y administrar su uso en una red de área local con más de 10 PCs. La administración, basada en la implementación *WinSockets* del protocolo TCP/IP, se realiza de forma transparente al usuario, con un acceso desde cualquier puesto de trabajo de la red a cualquiera de los dos robots existentes. Los robots son compartidos con la misma filosofía que se comparten recursos informáticos en la red, es decir, impresoras, directorios de discos, etcétera, pero, evidentemente, con especial tratamiento al tema de seguridad.

Con el uso de todas estos sistemas, se ha obtenido un entorno completo de modelado, programación y simulación de sistemas robotizados que permite integrar y validar de forma cómoda y eficiente las técnicas desarrolladas en este trabajo de investigación de planificación de movimientos con detección de colisiones. De hecho, para modelar los sistemas robotizados con volúmenes esféricos, se pueden utilizar tanto SimCef como RCad para obtener los puntos originales del sistema robotizado real, calculándose así mismo el modelo cinemático con la primera aplicación. Como entrada a la planificación de movimientos, la programación de los robots puede ser realizada con diferentes sistemas, tal como SIPRAC, WinArla, IntArla y RCad, siendo este último especialmente interesante para simular el comportamiento del robot.

La filosofía de integración dentro de estas aplicaciones de un módulo de planificación de movimiento con detección de colisiones para sistemas robotizados se refleja en la arquitectura de la Figura E.2. En el esquema se muestran las relaciones que existen entre los diferentes módulos así como los formatos de ficheros utilizados por cada uno.

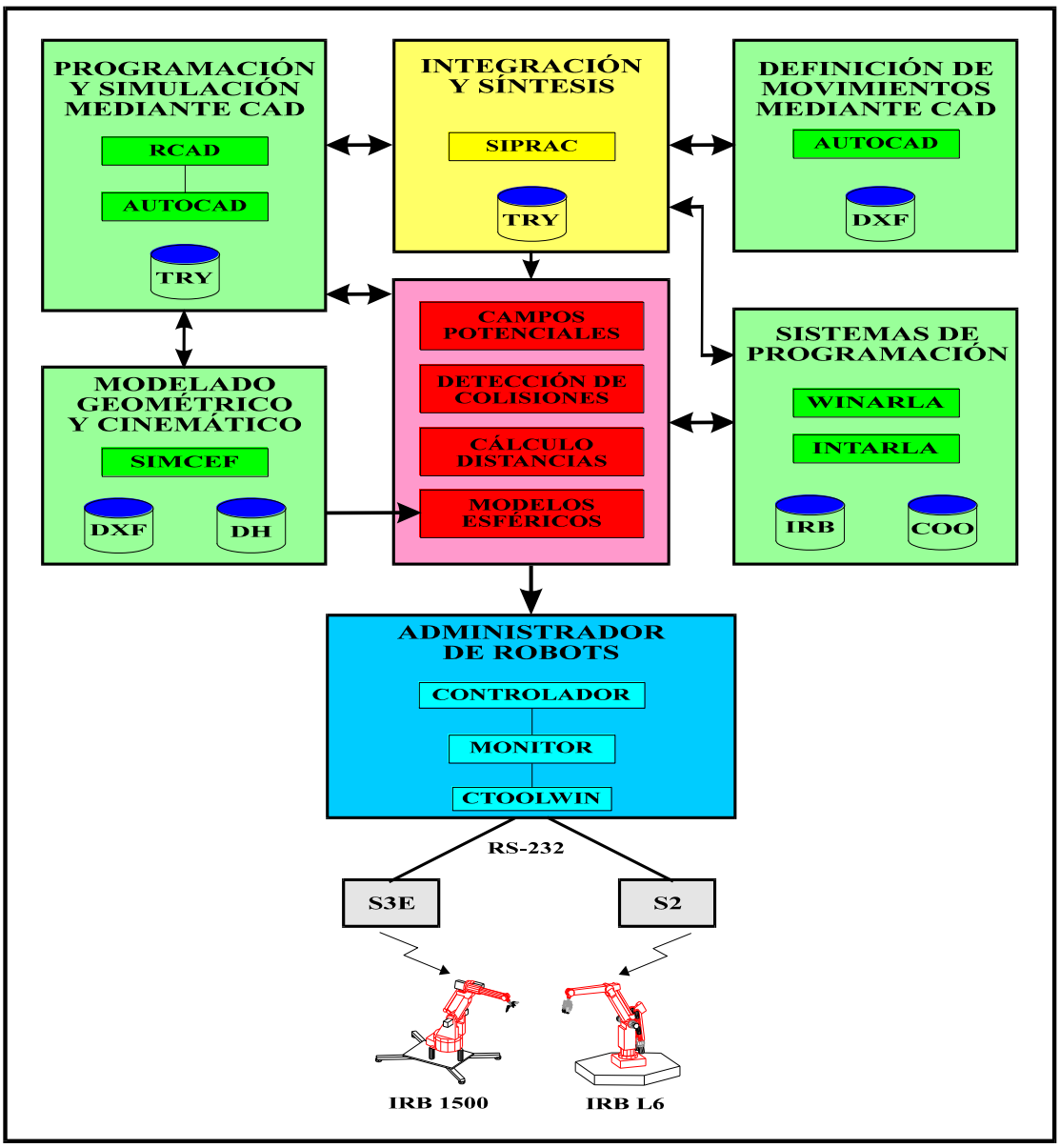


Figura E.2. Integración de la Planificación de Movimientos con los Diferentes Sistemas de Programación de Robots. *En la arquitectura propuesta se dispone de un módulo para la planificación de movimientos con detección de colisiones basada en el cálculo de distancias entre volúmenes esféricos. La entrada al módulo puede ser tomada a partir de diferentes sistemas de programación, como SIPRAC, WinArla, IntArla y RCad, siendo este último de gran utilidad para la simulación del comportamiento de los robots. Los modelos geométricos y cinemáticos de los robots se pueden obtener como salida del sistema SimCef. Una vez el módulo de planificación de movimientos determina los caminos libres de colisión, éstos se pueden ejecutar sobre los robots reales mediante el módulo administrador de los robot.*

Con la utilización de todas estos sistemas se ha definido una serie de aplicaciones prototipo sobre las que se han aplicado los conceptos teóricos desarrollados en este trabajo. Estas aplicaciones pueden cubrir desde la detección de colisiones de un robot o en una célula multi-robot a la planificación automática del movimiento de un robot o de un AGV. Algunas de estas aplicaciones se comentan a continuación.

La detección de colisiones basada en la estructura jerárquica con diferentes grados de precisión y para el modelado con volúmenes esféricos se ha utilizado para verificar los movimientos realizados por dos robots (con variación en todas sus articulaciones) cuyas herramientas se cruzan en el espacio (Figura E.3a). Este ejemplo se ha programado y simulado mediante RCad, si bien los programas se podrían haber generado con WinArla o SIPRAC. El movimiento de una posición a otra para cada robot se realiza con una interpolación lineal en las variables de articulación de los robots, generada con SimCef.

Por la situación en que se encuentran los robots, sólo pueden colisionar los elementos desde el brazo a la herramienta, por lo que la plataforma, la base y el cuerpo no se consideran en la estructura. En la secuencia de gráficas para las situaciones inicial y final (Figuras E.3b y E.3f), así como para tres pasos intermedios (Figuras E.3c-e), puede observarse cómo actúa el procedimiento desarrollado. Inicialmente, los modelos considerados para cada robot son dos esferoides cuadráticas, entre los que calcular la distancias se realiza en 2.66ms. Esta representación se mantiene mientras no se produzca colisión entre ellas, en cuyo caso se pasa a utilizar otros modelos o a descender de nivel, como ocurre en este ejemplo. En este ejemplo, el mayor nivel de precisión requerido ha consistido en modelar un robot con seis bi-esferas y el otro con una esferoide cuadrática, una tri-esfera y dos bi-esferas. En esta situación, la detección de colisiones se puede realizar en 14milisegundos. En el siguiente paso se representa un robot con 2 bi-esferas y el otro con una spline esférica con cinco esferas de control, calculándose la distancia entre ellos en 6.4milisegundos. En el último paso mostrado, se consideran una esferoide cuadrática y una cúbica como modelos, con un tiempo de cálculo de distancias de 4ms.

Para verificar completamente el movimiento de los robots han sido necesario realizar 8 cálculos de distancia (dos para el primer paso, tres para el segundo, uno para el tercero y dos para el último) realizados en un total de 57.72ms. Al utilizar la estructura jerárquica extendida, cuando se desciende de nivel (o aumenta el grado de precisión), es porque se ha producido colisión en un nivel superior, habiendo sido infructuoso este cálculo. Para este ejemplo, se han invertido 36.26ms en cálculos de distancias no provechosos. El tiempo necesario de calcular los modelos, descartando aquellos que se pueden obtener mediante un preproceso off-line, es de 24ms. Por tanto, la verificación del movimiento realizado por los dos robots se realiza en un tiempo total de 118milisegundos. Este resultado se debe considerar como muy bueno, teniendo en cuenta la dificultad del problema, ya que en el caso más conflictivo, la distancia entre las herramientas de los robots es menor a 10cm. Sin utilizar el modelo jerárquico ni las envolventes esféricas, es decir, considerando la representación exacta de los elementos considerados, se puede estimar que el tiempo total para la detección de colisiones es

aproximadamente de 24 segundos (3segundos para cada uno de los 8 cálculos de distancia realizados).

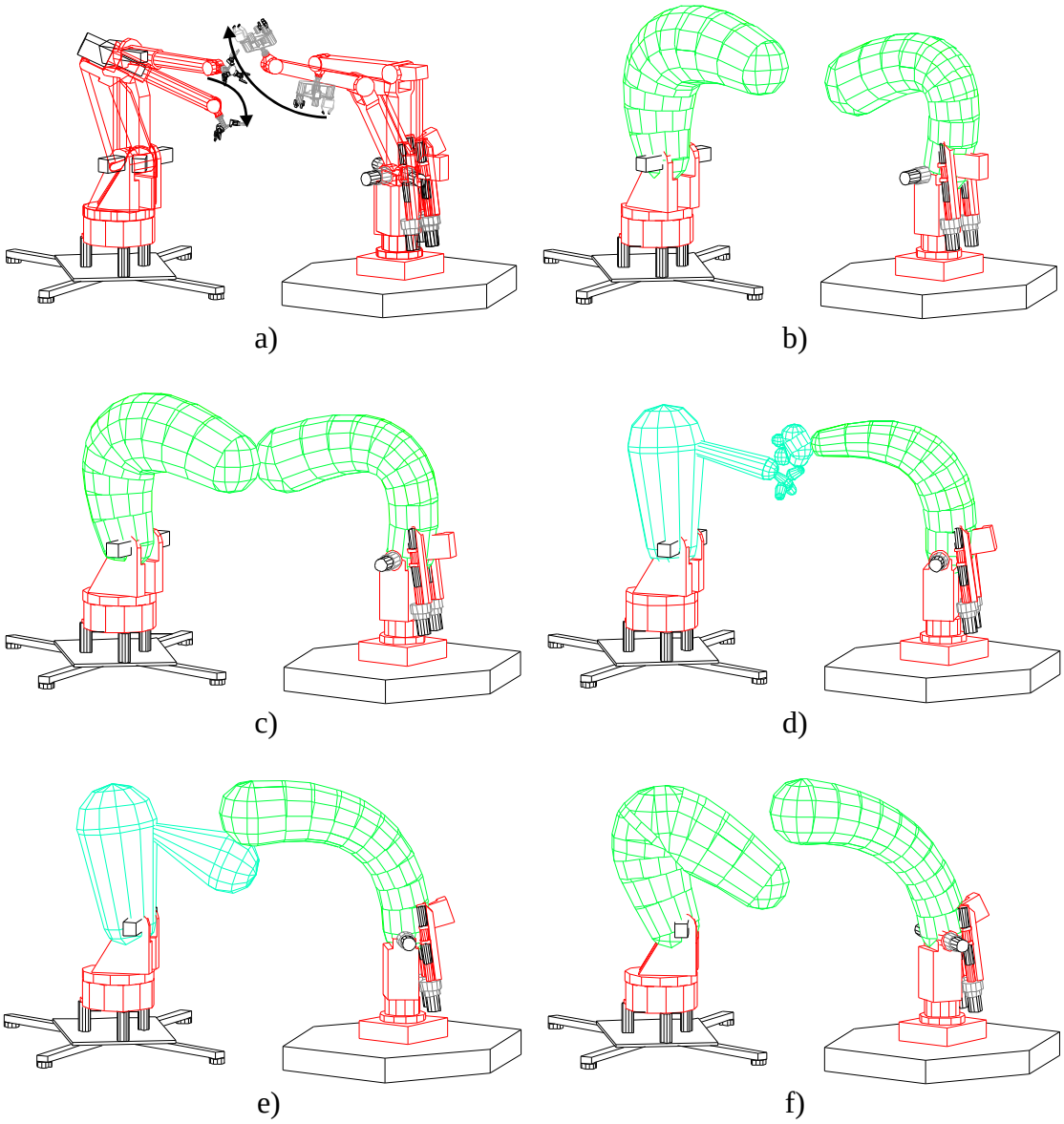


Figura E.3. Aplicación Prototipo para la Detección de Colisiones entre Dos Robots. *En la gráfica* a *se muestra el ejemplo al que se aplica la detección de colisiones, con dos robots que realizan un movimiento con variación en todas sus articulaciones, con una distancia mínima de diez centímetros entre sus pinzas. La secuencia de gráficas* b-f *muestra los diferentes modelos que se utilizan para la detección de colisiones en la estructura jerárquica extendida. La verificación de los movimientos se realiza con un tiempo total de 118ms, incluyendo todos los cálculos de distancias y la generación de modelos necesaria según la configuración de los robots.*

Otra aplicación posible sería aquella en que se supone una programación en tiempo real de un brazo-robot donde el programador mueve el brazo con la palanca de mandos para definir una instrucción de movimiento. El movimiento se realiza en diversos pasos, implicando cada paso el movimiento independiente de una articulación del robot. Estas acciones se pueden introducir mediante WinArla. Todos estos movimientos básicos se combinan para producir el movimiento de la configuración inicial a otra final, que corresponden al comando de movimiento del robot. El efecto de la ejecución de esta instrucción no será repetir la secuencia de movimientos que el programador ha guiado al robot con la palanca de mandos, sino realizar una interpolación lineal para cada una de las variables de articulación del robot (si el movimiento es rectilíneo o circular, se interpolará lineal o circularmente en el espacio cartesiano para un punto característico de la herramienta). Esta interpolación se puede obtener mediante SimCef, ya que dispone del modelo cinemático del robot. El problema se plantea porque el programador no sabe si el movimiento introducido presenta un problema de colisión, debiendo ser el sistema el que le responde de una forma rápida. El mismo caso se plantea si la programación es off-line, realizada por ejemplo con IntArla o SIPRAC, aunque el requisito temporal no es tan importante.

La planificación de movimientos con campos potenciales se puede aplicar para determinar los movimientos de un brazo-robot en el espacio cartesiano de dos formas diferentes. En la primera se utiliza una filosofía de generación-y-prueba: si el robot debe transportar una carga, se planifica el movimiento de la carga desde su posición actual a la posición destino como si fuera un objeto de vuelo libre. Para cada una de las posiciones intermedias de la pieza a transportar se resuelve el problema cinemático inverso para calcular los valores de las variables de articulación del robot. Realizando una detección de colisiones del robot con los obstáculos se podrá verificar este movimiento. En caso contrario, para la posición de la carga donde se ha producido colisión se genera ficticiamente un obstáculo, probándose un nuevo camino. Esta filosofía ha sido utilizada, por ejemplo, por [Z&H94].

La otra posibilidad de aplicar los campos potenciales en el espacio cartesiano a un brazo-robot se basa en la aplicación de los campos potenciales con rotación. Sobre los diferentes elementos se considera el potencial atractivo generado por el destino y el potencial repulsivo generado por los obstáculos. En ambos casos se considera la rotación (traslación) que se produce sobre el eje de articulación de revolución (prismática). A diferencia del caso general planteado en el Capítulo 5, cuando el eje es de revolución, el momento de giro y la rotación vienen dadas respecto a un eje fijo.

La planificación de movimientos por campos potenciales también se puede aplicar a un vehículo autoguiado en un entorno modelado con volúmenes esféricos. Un ejemplo en un entorno real se muestra en la Figura E.4 para el vehículo autoguiado disponible en el Laboratorio de Robótica del DISCA. El entorno se modela con diferentes poli-esferas, mientras que para el móvil se han considerado dos envolventes, una esfera (Figura E.4c) y una bi-esfera (Figura E.4d). En el primer caso se ha utilizado la optimización de Newton, mientras que para la bi-esfera se utiliza la rotación. En ambos casos, la planificación obtenida ha permitido dirigir el AGV hasta la posición destino. Las

pequeñas oscilaciones que se producen en el camino no son preocupantes, ya que son suavizadas directamente por la unidad de control del AGV mediante clotoides.

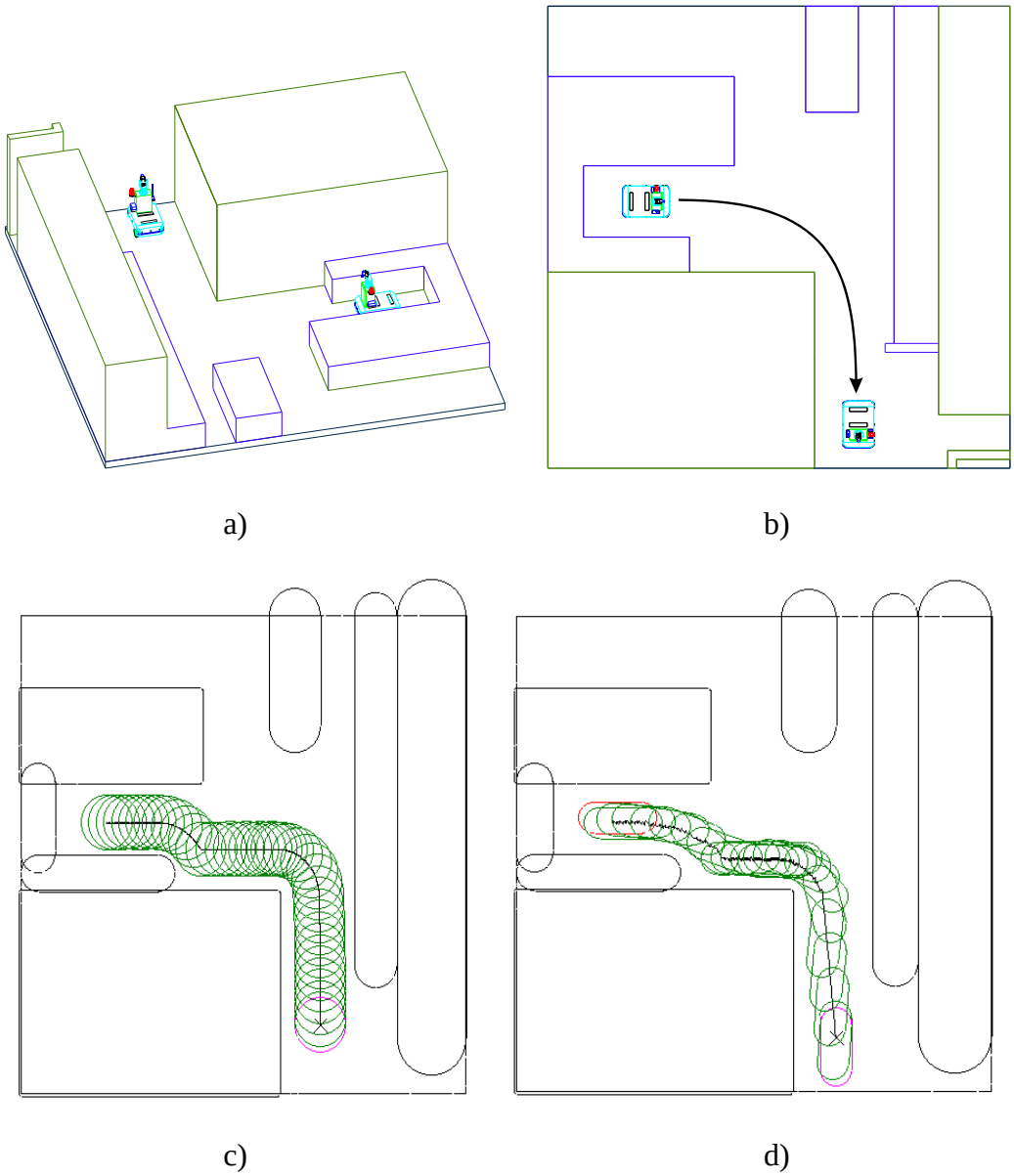


Figura E.4. Aplicación Prototipo para la Planificación de Movimientos de un AGV. *En las gráficas* a *y* b *se muestran una perspectiva y la planta del laboratorio donde se desea dirigir el AGV desde su configuración actual a otra destino. Tras modelar el entorno con poli-esferas, en la gráfica* c *se muestra el resultado de aplicar la optimización de Newton a la esfera modelo del AGV. En la gráfica* d *el móvil se modela con una bi-esfera y se realiza la planificación considerando la rotación del móvil.*

II. Conclusiones y Principales Aportaciones

El trabajo de investigación reflejado en esta Tesis Doctoral aporta un nuevo enfoque al problema de la planificación de movimientos que permite trabajar directamente en el espacio cartesiano y que ha sido aplicado con buenos resultados a diferentes aplicaciones industriales prototipo para sistemas multi-robot en entornos complejos. Su desarrollo se ha basado en la aplicación de varios métodos de optimización nuevos en el contexto de la planificación de movimientos con campos potenciales y en la especificación de procedimientos rápidos de detección de colisiones sobre una estructura jerárquica extendida que representa, mediante diferentes niveles y grados de precisión, una célula flexible con diferentes sistemas. Ambos procesos se han acelerado mediante la utilización de unos modelos basados en esferas.

Más específicamente, se han definido unos modelos esféricos que pueden entenderse como continuos, ya que para cualquier nivel de aproximación deseado, formado por un número finito de esferas de control, el modelo real está compuesto de infinitas esferas. El dominio de objetos representables es casi ilimitado, al poder aplicarse tanto relaciones lineales (poli-esferas) como polinomios en uno o varios grados de libertad (las esferoides y sus extensiones) a las esferas de control.

Frente a los modelos esféricos discreto-jerárquicos, presentan la ventaja de su sencillez, la facilidad de su cálculo y la rapidez con que se pueden generar a partir de la representación real del objeto (alcanzable en un número de pasos de refinamiento finito), lo que hace de estos modelos una potente herramienta para el modelado de sistemas robotizados.

Su utilización, como volumen envolvente representativo de un margen de seguridad para diferentes grados de precisión, es la base del desarrollo de una estructura jerárquica extendida, representable mediante un grafo AND-OR, para la detección rápida de colisiones en una célula multi-robot. Para ello se ha desarrollado, en el caso de las poli-esferas, un procedimiento de detección de colisiones rápido que generaliza y amplía las características de los algoritmos utilizados para los poliedros convencionales.

Con los modelos esféricos como cimiento y el procedimiento rápido de detección de colisiones como soporte, se ha podido aplicar, directamente en el espacio cartesiano, la planificación de movimientos mediante campos potenciales artificiales a diferentes sistemas robotizados, así como presentar alternativas a las técnicas de búsqueda de caminos óptimos.

Es de resaltar que la investigación realizada ha sido desarrollada principalmente dentro del marco de dos proyectos de investigación subvencionados por la Comisión Interministerial de Ciencia y Tecnología (C.I.C.Y.T.). Por una parte, la definición de modelos esféricos y el desarrollo de procedimientos rápidos de detección de colisiones

ha venido motivada por la participación en el Proyecto "Inspección y Manipulación Inteligente de Productos Deformables: Automatización Agroalimentaria" (Proyecto Número ROB91-0362, 1992-1994). Así mismo, la generación automática de modelos y la planificación de movimientos mediante campos potenciales es una labor realizada dentro del Proyecto(*), todavía en fase de desarrollo, "Planificación de Movimientos para Vehículos Autónomos en Entornos Industriales Basados en Percepción Sensorial" (Proyecto Número TAP95-1086-C02-01, 1995-1998).

Con estas aportaciones se han abordado dos aspectos fundamentales, la representación y modelado del robot y los obstáculos que se encuentren en su región de trabajo y la posibilidad de disponer de procedimientos rápidos para la detección de colisiones entre ellos, abriendo una nueva vía de impulso para la implantación de la planificación de movimientos en el campo de la robótica.

III. Autocrítica y Trabajos Futuros

A la hora de realizar un balance sobre el trabajo realizado, no sería lógico ni justo olvidar aquellos aspectos en los que no se ha alcanzado el objetivo deseado o que no se han contemplado en el trabajo. De hecho, esta reflexión resulta necesaria para poder reforzar una labor que nunca debe darse por terminada para que una línea de investigación siga abierta.

La formalización de los modelos esféricos debe servir de sustento a futuros trabajos en el campo de la planificación de movimientos que se beneficien de la sencillez que estos modelos aportan. Las posibilidades de algunos de ellos no ha sido investigada en profundidad, tal como las extensiones bi-paramétricas de las esferoides, potencialmente muy útiles para determinar el volumen barrido en el movimiento realizado por un brazo-robot en aplicaciones de detección de colisiones.

Los modelos presentados son suficientemente generales para ser aplicados tanto en el espacio cartesiano como en el de configuraciones. A pesar de las claras ventajas que trae consigo trabajar en el espacio cartesiano, muchos autores prefieren utilizar una representación en el espacio de configuraciones ([Val90], [BR+92], [K&S95], [E&C95] y [BB+95]). Consideraciones y simplificaciones especiales que se puedan presentar al utilizar esta metodología de modelado no han sido analizadas, ni sus posibilidades al aplicarse a los métodos de planificación de movimientos diferentes a los campos potenciales, tal como la descomposición celular o los mapas de carretera.

Respecto a la generación automática de los modelos, si bien se ha determinado una forma rápida y eficiente de calcularlos, se puede echar en falta cuestiones como el

(*) En colaboración con la Universidad de Murcia y la Universidad de Valencia.

análisis de las precisiones utilizadas o de la dispersión de los errores cometidos por la aproximación. Igualmente, se puede evaluar que el esfuerzo invertido en la simplificación y reducción de los modelos no ha dado los frutos apetecidos, si bien es cierto, que ello es debido a la rapidez de convergencia que presenta el procedimiento del cálculo de distancias con las poli-esferas.

El cálculo de distancias cuando intervienen esferoides se ha resuelto mediante un problema de optimización que presenta resultados satisfactorios. Sin embargo, resulta bastante probable que se pueda mejorar utilizando un planteamiento geométrico como extensión al algoritmo que normalmente se utiliza del cálculo de distancias entre dos curvas tri-dimensionales ([Gla90]).

La detección de colisiones basada en la estructura jerárquica extendida debe ser ampliada con un mayor nivel de inteligencia, que decida las descomposiciones o agrupaciones de nodos a realizar cada vez que se desciende o asciende de nivel, así como el grado de precisión a utilizar en cada caso. Algunas opciones ya se presentaron en la sección correspondiente, si bien no han sido estudiadas con profundidad.

Respecto a la utilización de métodos clásicos de optimización para la búsqueda del camino óptimo en la planificación de movimientos basada en campo potenciales, la vía ha quedado abierto a nuevas perspectivas con la utilización de otros métodos diferentes. Tanto los métodos introducidos como los nuevos por considerar deben buscar incluir la posición y orientación para permitir trabajar cómodamente en el espacio cartesiano.

Evidentemente, aparte de estos aspectos concretos, se pueden considerar campos más amplios en los que el autor de esta Tesis Doctoral se plantea como objetivos inmediatos y que quedan reflejados a continuación:

- A nadie puede escaparse que este trabajo no ha tenido como objetivo el resolver el principal problema de los campos potenciales: los mínimos locales. Muchas técnicas se han desarrollado para generar potenciales con un único mínimo local (el destino) o para escapar de los diferentes mínimos en que pueda quedar atrapado el móvil. Sin embargo, parece que el camino a seguir debe consistir en una predicción de estas situaciones para la determinación de una búsqueda alternativa de caminos. Esta predicción debe actuar en un nivel global, por encima de comportamientos reactivos de funcionamiento del robot. Este debe ser uno de los campos a cubrir en el Proyecto ya reseñado sobre planificación de movimientos en vehículos autónomos que se encuentra en fase de desarrollo.

- Las posibilidades de la integración sensorial han provocado hoy en día un auge en la robótica móvil como plataformas de investigación. Los vehículos autoguiados, al disponer de menos grados de libertad, no son tan complejos como los brazos-robot articulados, resultando además más cómodo, por su estructura mecánica, acoplarles dispositivos sensoriales. La utilización de AGVs ha abierto de nuevo las puertas a temas de investigación prácticamente cerrados en la robótica tradicional, como la calibración y auto-localización ([FBE94], [PJU95] y [CFG95]) o el suavizado de trayectorias cumpliendo las restricciones cinemáticas de los vehículos no-

holonómicos ([Z&M92], [L&C93], [ACB95] y [CBA96]). Estos campos están siendo tratados en el marco del Proyecto subvencionado por la Comisión Europea dentro del Programa de Tecnologías de la Información "Use of the Expert System Platform REAKT for Transport Robot Guidance", RETRARO(*) , (Número 20.778, 1995-1997).

- La integración completa de todo el software desarrollado con las plataformas experimentales existentes debe completarse con los nuevos aspectos a considerar en los dos apartados anteriores, permitiendo estudiar casos hasta ahora no implementados como la planificación de un robot con campos potenciales artificiales en el espacio cartesiano con rotación en tres dimensiones.

√

(*) En consorcio con GMV S.A. (España), TZ RS, FH Weingarten y Albert Weber (Alemania).

REFERENCIAS

Powell alcanzó el Manual de Robótica que gravitaba precariamente en una esquina de su escritorio y lo abrió con reverencia. En una ocasión había saltado por la ventana de una casa en llamas vestido tan sólo con calzoncillos y el manual. En caso de necesidad, se hubiera desprendido de los calzoncillos.

"Atrapa esa Liebre". Isaac Asimov, 1944.

[Abr88] S. Abramowski
"Collision Avoidance For Nonrigid Objects"
Proc. Int. Workshop On Computational Geometry & Its Applications, Berlin (Germany) 1988.

[ABF88] F. Avnaim, J.D. Boissonnat & B. Faverjon
"A Practical Exact Motion Planning Algorithm For Polygonal Objects Amidst Polygonal Obstacles"
Technical Report No. 890, INRIA, France, 1988.

[ACB95] B. d'Andréa-Novel, G. Campion & G. Bastin
"Control Of Nonholonomic Wheeled Mobile Robots By State Feedback Linearization"
The International Journal Of Robotics Research, Vol. 14, No. 6, December 1996.

[Agb95] E.I. Agba
"Seamaster: An ROV-Manipulator System Simulator"
IEEE Computer Graphics And Applications, January, 1995.

[A&P75] A.P. Ambler & R.J. Popplestone
"Inferring The Positions Of Bodies From Specified Relationships"
Artifical Intelligence, 6(2), 157-174, 1975.

[ASL89] R. Alami, T. Siméon & J.P. Laumond
"A Geometric Approach To Planning Manipulation Tasks. The Case Of Discrete Placements & Grasps"
Preprints Of The Fifth Int. Symp. Of Robotics Research, Tokyo (Japan), 1989.

[Bar89] D.A. Barford
"Fast Detection Of Collisions Between Moving Bodies"
Proc. SIAM Conf. On Geometric Design, 1989.

[BB+95] C. Balaguer, A. Barrientos, F.J. Rofríguez, R. Aracil, E.A. Puente & U. Peter
"Reduction Of Free-Space-Loss For Good & Rapid 3D Path Pklanning Of 6DOF Robots"
Journal F Intelligent And Robotic Systems, 13, pp 263-278, 1995.

[BBB87] R.H. Bartels, J.C. Beatty & B.A. Barsky
"An Introduction To Splines For Use In Computer Graphics & Geometric Modeling"
Morgan Kaufmann Publishers Inc., 1987.

[BC+91] J.D. Boissonnat, A. Cérézo, O. Devillers & M. Teillaud
"Output Sensitive Construction Of 3D Delaunay Triangulation Of Constrained Set Of Points"
Canadian Conf. On Computational Geometry, August 1991 (INRIA T-R 1415)

[BC+92] J.D. Boissonnat, A. Cérézo, O. Devillers, J. Duquesne & M. Yvinec
"An Algorithm For Constructing The Convex Hull Of A Set Of Spheres In Dimension D"
4th Canadian Conf. On Computational Geometry, April 1992 (INRIA T-R 2080)

[B&D83] M.C. Bonney, N.K. Dooner Et Al.
"Verifying Robot Programs For Collision Free Tasks"
Developments In Robotics, IFS Publication LTD, 1983.

[Bic95] A. Bicchi
"On The Closure Properties Of Robotic Grasping"
The International Journal Of Robotics Research, Vol.14, No. 4, August 1995.

[B&K88] S. Bonner & R.B. Kelley
"A Representation Scheme For Rapid 3-D Collision Detection"
Third Int. Symp. On Intelligent Control, Arlington, VA, August 1988.

[B&K89a] S. Bonner & R.B. Kelley
"Planning 3-D Collision-Free Paths Using Spheres"
NASA Conf. On Space Telerobotics, Pasadena, California, January 1989.

[B&K89b] S. Bonner & R.B. Kelley
"Planning 3-D Collision-Free Paths"
Fourth Int. Symp. On Intelligent Control, Albany, N.Y., September 1989.

[B&K89c] S. Bonner & R.B. Kelley
"A Novel Representation For Planning 3-D Collision-Free Paths"
IEEE Trans. On Systems, Man And Cybernetics, Vol. SMC-19, No. 6, December 1989.

[B&L89a] J. Barraquand & J.C. Latombe
"Robot Motion Planning: A Distributed Representation Approach"
Report No. STAN-CS-89-1257, Department Of Computer Science, Standford University, 1989.

[B&L89b] J. Barraquand & J.C. Latombe
"Robot Motion Planning: A Distributed Representation Approach"
Report No. STAN-CS-89-1285, Department Of Computer Science, Standford University, 1989.

[B&L90] J. Barraquand & J.C. Latombe
"A Monte-Carlo Algorithm For Path Planning With Many Degrees Of Freedom"
Proc. IEEE Intl. Conf. On Robotics And Automation, Cincinnati, OH, 1712-1717, 1990.

[BLL89] J. Barraquand, B. Langlois & J.C. Latombe
"Robot Motion Planning With Many Degrees Of Freedom And Dynamic Constraints"
Preprints Of The Fifth Int. Symp. Of Robotics Research, Tokyo (Japan), 1989.

[Bob89] J.E. Bobrow
"A Direct Minimization Approach For Obtaining The Distance Between Convex Polyhedra"
Int. Journal Of Robotics Research, Vol.8, No.3, 1989.

[Bon89] S. Bonner
"Successive Spherical Approximations For Planning Collision-Free Paths In A Three-Dimensional Robot Workcell Environment"
Ph.D. Dissertation, Rensselaer Polytechnic Institute, Troy, N.Y., December 1989.

[Boy79] J.W. Boyse
"Interference Detection Among Solids And Surfaces"
Comm. ACM, Vol 22, No 1, 1979.

[BR+92] C. Balaguer, F.J. Rodríguez, J.M. Pastor, E. Gambao, R. Aracil & A. Barrientos
"A Rule-Based Selection Of C-Space Representation For Rapid 3D Robot Path Planning"
IFAC-INCOM'92, Ontario, Canada, 1992.

[Bro82] R.A. Brooks
"Symbolic Error Analysis And Robot Planning"
International Journal Of Robotics Research, 1(4), 29-68, 1982.

[Bro83a] R.A. Brooks
"Solving The Find-Path Problem By Representing Free Space As Generalized Cones"
Artificial Intelligence Lab., AI Memo No. 674, Massachusetts Inst. Of Technology, 1983.

[Bro83b] R.A. Brooks
"Solving The Find-Path Problem By Good Representation Of Free Space"
IEEE Transactions On Systems, Man And Cybernetics, SMC-13, 190-197, 1983.

[B&T95a] E.J. Bernabeu & J. Tornero
"An Optimal And Automatic Geometric Modeler Of Real Objects In Robotic Systems Using The Hough Transform Method"
II Int. Symp. On Method & Models In Automations & Robotics, Miedzyzdroje, Polonia, 1995

[B&T95b] E.J. Bernabeu & J. Tornero
"An Automatic Method For The Geometric Modeling Of Robotic Systems Based On The Hough Transform"
11th Int. Conf. On CAD/CAM, Robotics & Factories Of The Future, Pereira, Colombia, 1995

[BTM96] E.J. Bernabeu, J. Tornero & M. Mellado
"A Generalized Method For Optimal Modeling In Robotic Applications"
Second World Automation Congress, Montpellier, France, 1996

[Bur74] D.M. Burley
"Studies In Optimization"
Intertex Book, 1974

[Bur88] J.W. Burdick
"Kinematic Analysis And Design Of Redundant Robot Manipulators"
Ph.D. Dissertation, Report No. STAN-CS-88-1207, D.C.S., Stanford University, 1988.

[Cam85] S. A. Cameron
"A Study Of The Clash Detection Problem In Robotics"
Proc. Of The Int. Conf. On Robotics And Automation, 1985.

[Cam89] S.A. Cameron
"Efficient Intersection Tests For Objects Defined Constructively"
Intl. Journal Of Robotics Research, Vol. 8, 1989.

[Cam90] S.A. Cameron
"Collision Detection By Four-Dimensional Intersection Testing"
IEEE Transactions On Robotics & Automation, Vol.6, No.3, June 1990.

[Cam91] S.A. Cameron
"Efficient Bounds In Constructive Solid Geometry"
IEEE Computer Graphics And Applications, May 1991.

[Can84] J.F. Canny
"Collision Detection For Moving Polyhedra"
Technical Report Artificial Intelligence Lab., MIT, No. 806, 1984.

[Can86] J.F. Canny
"Collision Detection For Moving Polyhedra"
IEEE Trans. On Pattern Analysis And Machine Intelligence, Vol. 8 No2, March 1986.

[Can88] J.F. Canny
"The Complexity Of Robot Motion Planning"
MIT Press, Cambridge, MA, 1988.

[Can89] J.F. Canny
"On Computability Of Fine Motion Plans"
IEEE Int. Conf. On Robotics And Automation, Scottsdale, AZ, 1989

[CBA95] G. Campion, G. Bastin & B. d'Andréa-Novel
"Structural Properties And Classification Of Kinematic And Dynamic Models Of Wheeled Mobile Robots"
Ieee Transactions On Robotics And Automation, Vol.12, No. 1, February 1996.

[CBW90] C.I. Connolly, J.B. Burns & R. Weiss
"Path Planning Using Laplace' S Equation"
IEEE Int. Conf. On Robotics And Automation, Cincinnati (Ohio), 2102-2106, 1990.

[C&C86] S. A. Cameron & R.K. Culley
"Determinig The Minimum Translational Distance Between Two Convex Polyhedra"
Proc. Of The Int. Conf. On Robotics And Automation, 1986.

[CFG95] R. Caracciolo, F. Fanton & A. Gasparetto
"A Low Cost Laser Based System for Mobile Robot Localization"
Int. Symp. on Measurement & Control in Robotics, Smolenice C., Slovakia, June 1995.

[Cha82] R. Chatila
"Path Planning And Environment Learning In A Mobile Robot System"
European Conf. On Artificial Intelligence, Orsay, 1982.

[C&K86] R.K. Culley & K.G. Kempf
"A Collision Detection Algorithm Based On Velocity And Distance Bounds"
Proc. Of The Int. Conf. On Robotics And Automation, 1986.

[C&L93] J.F. Canny & M.C. Lin
"An Opportunistic Global Path Planner"
Algorithmica, Special Issue On Computational Robotics, Vol. 10, No. 2-4, 1993.

[CL+94] J.D. Cohen, M.C. Lin, D. Manocha & M.K. Ponamgi
"Interactive And Exact Collision Detection For Large-Scaled Environments"
TR94-005, Dept. Of Computer Science, University Of North Carolina, Chapel Hill, 1994

[CL+95] J.D. Cohen, M.C. Lin, D. Manocha & M.K. Ponamgi
"I-COLLIDE: An Interactive And Exact Collision Detection System For Large-Scale Environments"
Proc. Of ACM Interactive 3D Graphics Conference, Monterrey, CA, 1995.

[CLZ95] G. Conte, S. Longhi & R. Zulli
"An Algorithm For Non-Holonomic Motion Planning"
Int. Symp. On Measurement & Control In Robotics, Smolenice C., Slovakia, June 1995.

[Cra88] J.J. Craig
"Introduction To Robotics. Mechanics & Control"
Addison-Wesley, 1988.

[Dai89] F. Dai
"Collision-Free Motion Of An Articulated Kinematic Chain In A Dynamic Environment"
IEEE Computer Graphics & Applications, January 1989.

[dBo78] C. De Boor
"A Practical Guide To Splines"
Springer-Verlag, 1978.

[D&G92] A. Delchambre & P. Gaspart
"KBAP: An Industrial Prototype Of Knowledge-Based Assembly Planner"
Proceedings Of The IEEE Int. Conf. On Robotics And Automation, Nice, France, 1992.

[D&G94] O. Devillers & M. Golin
"Incremental Algorithms For Finding The Convex Hulls Of Circles And The Lower Envelopes Of Parabolas"
RIA Technical-Report 2280, Juny 1994.

[D&H55] J. Denavit & R.S. Hartenberg
"A Kinematic Notation Of Lower-Pair Mechanisms Based On Matrices"
J. App. Mech. Vol. 77, pp. 215-221, 1955.

[D&K85] D.P. Dobkin & D.G. Kirkpatrick
"A Linear Algorithm For Determining The Separation Of Convex Polyhedra"
Journal Of Algorithms, 6, 1985.

[D&K90] D.P. Dobkin & D.G. Kirkpatrick
"Determining The Separation Of Preprocessed Polyhedra - Unified Approach"
ICALP-90, 443, pp. 400-413, 1990.

[D&L84] B. Dufay & J.C. Latombe
"An Approach To Automatic Robot Programming Based On Inductive Learning"
International Journal Of Robotics Research, 3(4), 3-20, 1984.

[DMT92] O. Devillers, S. Meiser & M. Teillaud
"The Space Of Spheres, A Geometric Tool To Unify Duality Results On Voronoi Diagrams"
INRIA Technical Report 1620. February 1992.

[Don84] B.R. Donald
"Local And Global Techniques For Motion Planning"
Msc Thesis, MIT, May 1984.

[dPE93] A.P. Del Pobil, M.T. Escrig & J.A. Jaen
"An Attempt Towards A General Representation Paradigm For Spatial Reasoning"
Proc. IEEE Int. Conf. On Systems, Man And Cybernetics, Le Touquet (France), 1993.

[dPo91] A.P. Del Pobil
"Planificación De Movimientos De Robots Basada En Técnicas De Inteligencia Artificial: Un Modelo Esférico"
Tesis Doctoral. Universidad De Navarra (Spain), 1991.

[dPS92a] A.P. Del Pobil & M.A. Serna
"Solving The Find-Path Problem By A Simple Object Model"
Proc. 10th European Conference On Artificial Intelligence, Vienna (Austria), 1992.

[dPS92b] A.P. Del Pobil, M.A. Serna & J. Llovet
"A New Representation For Collision Avoidance And Detection"
Proc. IEEE Int. Conf. On Robotics And Automation, Nice (France), 1992.

[dPS94a] A.P. Del Pobil & M.A. Serna
"A New Object Representation For Robotics And Artificial Intelligence Applications"
Int. Journal Of Robotics & Automation, Vol.9, No.1, 1994.

[dPS94b] A.P. Del Pobil & M.A. Serna
"A Simple Algorithm For Intelligent Manipulator Collision-Free Motion"
Journal Of Applied Intelligence, Vol.4, 1994.

[E&C95] J. Eliás & A. Cervenan
"Robot Collision Free Path Optimisation in Configuration Space"
Int. Symp. on Measurement & Control in Robotics, Smolenice C., Slovakia, June 1995.

[ELP86] M. Erdmann & T. Lozano-Pérez
" On Multiple Moving Objects"
Artificial Intelligence Lab., AI Memo No. 883, Massachusetts Inst. Of Technology, 1986.

[Ern61] H.A. Ernst
"A Computer-Operated Mechanical Hand"
Scd. Thesis Dissertation, Massachussets Institute Of Technology, 1961.

[ESZ93] A. Efrat, M. Sharir & A. Ziv
"Computing The Smallest *K*-Enclosing Circle And Related Problems"
Technical Report CIS-9309, Faculty Of Computer Science, Technion - IIT, Haifa, Israel, 1993.

[Eve95] H.R. Everett
"Sensors For Mobile Robots"
A.K. Peters Ltd., Wellesley, 1995.

[Fav84] B. Faverjon
"Obstacle Avoidance Using An Octree In The Configuration Space Of A Manipulator"
IEEE Int. Conf. On Robotics, Atlanta (Georgia), 504-512, 1984.

[Fav86] B. Faverjon
"Object Level Programming Of Industrial Robots"
Proc. Of The Int. Conf. On Robotics And Automation, 1986.

[Fav89] B. Faverjon
"Hierarchical Object Models For Efficient Anti-Collision Algorithms"
Proc. Of The Int. Conf. On Robotics And Automation, 1989.

[FBE94] L. Feng, J. Borenstein & H.R. Everett
"Where Am I? Sensors And Methods For Autonomous Mobile Robot Positioning"
Technical Report UM-MEAM-94-21, The University Of Michigan, December 1994.

[F&C92] C. Ferrari & J. Canny
"Planning Optimal Grasps"
Proceedings Of The IEEE Int. Conf. On Robotics And Automation, Nice, France, 1992.

[FGL85] K.S. Fu, R.C. González & C:S.G. Lee
"Robotics. Control, Sensing, Vision And Intelligence"
Mcgrawhill, 1985.

[F&H93] A. Foisy & V. Hayward
"A Safe Swept Volume Method For Collision Detection"
6th Int. Symp. On Robotics Research, 1993.

[FHA90] A. Foisy, V. Hayward & S. Aubry
"The Use Of 'Awareness' In Collision Prediction"
Proc. Of The Int. Conf. On Robotics And Automation, 1990.

[Fio95] P. Fiorini
"Robot Motion Planning Among Moving Obstacles"
Phd. Dissertation, Dep. Of Mechanical, Nuclear & Aerospace Engineering, UCLA 1995.

[FS+95] S. Fleury, P. Souères, J.P. Laumond & R. Chatila
"Primitives For Smoothing Mobile Robot Trajectories"
IEEE Transactions On Robotics And Automation, Vol. 11, No. 3, June 1995.

[F&T87] B. Faverjon & P. Tournassoud
"A Local Based Approach For Path Planning Of Manipulators With A High Number Of Degrees Of Freedom"
IEEE Int. Conf. On Robotics And Automation, Raleigh (North Carolina), 1987.

[F&T89] B. Faverjon & P. Tournassoud
"A Local Based Approach For Path Planning Of Manipulators With Many Degrees Of Freedom"
Fifth International Symposium Of Robotics Research, Tokyo, 65-73, 1989.

[GC+95] K. Glass, R. Colbaugh, D. Lim & H. Seraji
"Real-Time Collision Avoidance For Redundant Manipulators"
IEEE Transactions On Robotics And Automation, Vol. 11, No. 3, June 1995.

[G&F90] E.M. Gilbert & C.P. Foo
"Computing The Distance Between General Convex Objects In Three-Dimensional Space"
IEEE Trans. On Robotics And Automation, Vol.6, No.1, February, 1990.

[G&H89] E.M. Gilbert & S.M. Hong
"A New Algorithm For Detecting The Collision Of Moving Objects"
Proc. Of The Int. Conf. On Robotics And Automation, 1989.

[G&J85] E.M. Gilbert & D.W. Johnson
"Distance Functions And Their Application To Robot Path Planning In The Presence Of Obstacles"
EEE Journal Of Robotics And Automation, Vol RA-1, No 1, March 1985.

[GJK88] E.M. Gilbert, D.W. Johnson & S.S. Keerthi
"A Fast Procedure For Computing The Distance Between Complex Objects In Three-Dimensional Space"
IEEE Journal Of Robotics And Automation, Vol.4, No.2 April 1988.

[Gla90] A. Glassner
"Solving The Nearest Point On The Curve Problem"
In Graphics GEMS I, Ed. Academic Press, 1990.

[G&T95] V. Graefe & Q.H. Ta
"An Approach To Self-Learning Manipulator Control Based On Vision"
Int. Symp. On Measurement & Control In Robotics, Smolenice C., Slovakia, June 1995.

[Gou84] L. Gouzènes
"Strategies For Solving Collision-Free Trajectories Problems For Mobile And Manipulator Robots"
International Journal Of Robotics Research, 3(4), 51-65, 1984.

[GSF94] A. García Alonso, N.Serrano & J. Flaquer
"Solving The Collision Detection Problem"
IEEE Computer Graphics And Applications, May 1994.

[H&A88] Y.K. Hwang & N. Ahuja
"Path Planning Using A Potential Field Representation"
Coordinated Science Laboratory, University Of Illinois, UILU-ENG-88-2251, 1988.

[HA+95] V. Hayward, S. Aubry, A. Foisy & Y. Ghallab
"Efficient Collision Prediction Among Many Moving Objects"
The International Journal Of Robotics Research, Vol.14, No2, April 1995.

[HBC90] T. Hague, M. Brady & S. Cameron
"Using Moments to Plan Paths for the Oxford AGV"
Proc. IEEE Intl. Conf. On Robotics And Automation, Cincinnati, OH, 1990.

[HC+92] D. Henrich, X.Cheng, U. Rembold & R. Dillman
"Fast Distance Computation For On-Line Collision-Detection With Multi-Arm Robots"
Proc. Of The Int. Conf. On Robotics And Automation, Nice (France), May 1992.

[Her86] M. Herman
"Fast, Three-Dimensional, Collision-Free Motion Planning"
IEEE Int. Conf. On Robotics And Automation, San Francisco (California), 1986.

[HKT92] G.J. Hamlin, R.B. Kelley & J.Tornero
"Efficient Distance Calculation Using The Spherically-Extended Polytope (S-Tope) Model"
IEEE Int. Conf. On Robotics And Automation, Nice (France), Vol. 3, 2502-2507, 1992.

[HMS91] L.S. Homem De Mello & A.C. Sanderson
"A Correct And Complete Algorithm For The Generation Of Mechanical Assembly Sequences"
IEEE Trans. On Robotics And Automation, Vol. 7, No. 2, 1991.

[Hof89] C.M. Hoffmann
"Geometric & Solid Modeling. An Introduction"
Morgan Kaufmann Publishers Inc., 1989.

[HSS83] J.E. Hopcroft, J. Schwartz & M. Sharir
"Efficient Detection Of Intersections Among Spheres"
Intl. Journal Of Robotics Research, Vol 2. No. 4, Winter 1983.

[H&S88] G. Hurteau & N.F. Stewart
"Distance Calculation For Imminent Collision Indicationing A Robot System Simulation"
Robotica, No. 6, pp. 47-51, 1988.

[Hub93] P. Hubbard
"Interactive Collision Detection"
Proc. IEEE Symp. On Research Frontiers In Virtual Reality, October, 1993.

[Hub94] P. Hubbard
"Collision Detection For Interactive Graphics Applications"
PhD Thesis, Dept. Of Computer Science, Brown Univ. October 1994.

[Hub95] P. Hubbard
"Collision Detection For Interactive Graphics Applications"
IEEE Trans. On Visualization & Computer Graphics, Vol. 1, No.3, September, 1995.

[H&Z94] E. S. H. Hou & D. Zheng
"Mobile Robot Path Planning Based On Hierarchical Hexagonal Decomposition And Artificial Potential Fields"
Journal Of Robotics Systems, 11(7), 1994.

[Ike93] K. Ikeuchi
"Assembly Plan From Observation"
The Sixth Int. Symp. On Robotics Research, pp 111-120, 1993.

[Ina93] M. Inaba
"Remote-Brained Robotics: Interfacing AI With Real World Behaviours"
The Sixth Int. Symp. On Robotics Research, pp 335-344, 1993.

[Ino74] H. Inoue
"Force Feedback In Precise Assembly Tasks"
Technical Report No. AIM-308, AI Laboratory, Massachussets Inst. Of Technology, 1974.

[Jan95] D. Janglová
"Collision-Free Path Of Autonomous Mobile Vehicle"
Int. Symp. On Measurement & Control In Robotics, Smolenice C., Slovakia, June 1995.

[J&D88] A.K. Jain & R.C. Dubes
"Algorithms For Clustering Data"
Prentice Hall, 1988.

[J&G85] D.W. Johnson & E.M. Gilbert
"Minimum Time Robot Path Planning In The Presence Of Obstacles"
Proceedings Of 24th Conference On Decision And Control, December 1985

[J&P87] D.A. Jospeh & W.H. Plantinga
"Efficient Algorithms For Polyhedron Collision Detection"
Technical Report: UWMADISONCS CS-TR-87-738, 1987.

[J&T95] P. Jiménez & C. Torras
"Interferencia Entre Poliedros: Pares Arista-Cara Susceptibles De Entrar En Contacto"
Proc. Of VI Encuentros De Geometría Computacional, Barcelona (Spain), July 1995.

[Kav95] L.E. Kavraki
"Computation Of Configuration-Space Obstacles Using The Fast Fourier Transform"
IEEE Transactions On Robotics And Automation, Vol. 11, No. 3, June 1995.

[K&B91] Y. Koren & J. Borenstein
"Potential Field Methods And Their Inherent Limitations For Mobile Robot Navigation"
IEEE Int. Conf. On Robotics And Automation, Sacramento, California - April, 1991.

[Kha80] O. Khatib
"Commande Dynamique Dans l'Espace Opérationnel Des Robots Manipulateurs En Présence d'Obstacles"
Ph.D. Dissertation, Ecole Nat. Supérieure De l'Aéronautique Et De l'Espace, Toulouse, 1980.

[Kha86] O. Khatib
"Real-Time Obstacle Avoidance On Manipulators And Mobile Robots"
The Int. Journal Of Robotics Research, Vol 5, No 1, 1986.

[K&I93] S.B. Kang & K. Ikeuchi
"Toward Automatic Robot Instruction From Perception-Recognizing A Grasp From Observation"
IEEE Transactions On Robotics And Automation, Vol. 9, No. 4, August 1993.

[KII94] Y. Kuniyoshi, M. Inaba & H. Inoue
"Learning By Watching: Extracting Reusable Task Knowledge From Visual Observation Of Human Performance"
IEEE Transactions On Robotics And Automation, Vol. 10, No. 6, December 1994.

[K&K91] J.O. Kim & P. Khosla
"Real-Time Obstacle Avoidance Using Harmonic Potential Functions"
IEEE Int. Conf. On Robotics And Automation, Sacramento, California - April, 1991.

[K&K95] L. Kleeman & R. Kuc
"Mobile Robot Sonar For Target Localization And Classification"
The Int. Journal Of Robotics Research, Vol.14, No. 4, August 1995.

[K&N90] R. Kashipati & R. Nevatia
"Computing Volume Descriptions For Sparse 3-D Data"
Advances In Spatial Reasoning, Editor S.S. Chen, Ablex Pub. Corporation, Vol.2, 1990

[Kod87] D.E. Koditschek
"Exact Robot Navigation By Means Of Potential Functions: Some Topological Considerations"
IEEE Int. Conf. On Robotics And Automation, Raleigh (North Carolina), 1-6, 1987.

[Kro84] B.H. Krogh
"A Generalized Potential Field Approach To Obstacle Avoidance Control"
Proc. Of The SME Conf. On Robotics Research: The Next Five Years An Beyond, Bethlehem, PA, August 1984.

[K&S89a] K.J. Kyriakopoulus & G.N. Saridis
"An Efficient Minimum Distance And Collision Estimation Technique For On-Line Motion Planning Of Robotic Manipulation"
Tech. Rep. CIRSSE-TR-89-19, Rensselaer Polytechnic Institute, Troy (New York), 1989.

[K&S89b] K.J. Kyriakopoulus & G.N. Saridis
"Minimum Distance Estimation And Collision Prediction Under Undertainty For On-Line Robotic Motion Planning"
Tech. Rep. CIRSSE-TR-89-25, Rensselaer Polytechnic Institute, Troy (New York), 1989.

[K&S95] M. Kohler & M. Spreng
"Fast Computation Of The C-Space For Convex 2D Algebraic Objects"
Int. Journal Of Robotics Research, Vol. 14, No. 6, December 1995.

[K&V88] P Khosla & R. Volpe
"Superquadric Artificial Potentials For Obstacle Avoidance And Approach"
IEEE Int. Conf. On Robotics And Automation, Philadelphia, Pennsylvania, April, 1988.

[K&Z86] K. Kant & S.W. Zucker
"Toward Efficient Trajectory Planning: Path Velocity Decomposition"
International Journal Of Robotics Research, 5, 72-89, 1986.

[KZB85] D.T. Kuan, J.C. Zamiska & R.A. Brooks
"Natural Decomposition Of Free Space For Path Planning"
IEEE Int. Conf. On Robotics And Automation, St. Louis (Missouri), 1985.

[L&A89] J. P. Laumond & R. Alami
"A Geometrical Approach To Planning Manipulation Tasks In Robotics"
Technical Report No. 89261, LAAS, Toulouse, 1989.

[L&A94] Y.H. Liu & S. Arimoto
"Computation Of The Tangent Graph Of Polygonal Obstacles By Moving-Line Processing"
IEEE Transactions On Robotics And Automation, Vol. 10(6), December, 1994.

[Lat91] J.C. Latombe
"Robot Motion Planning"
Kluwer Academic Publisher, 1991.

[Lau86] J.P. Laumond
"Feasible Trajectories For Mobile Robots With Kinematic And Environment Constraints"
Int. Conf. On Intelligent Autonomous Systems, Elsevier Science Pub., Amsterdam, 1986.

[Lau87] J.P. Laumond
"Finding Collision-Free Smooth Trajectories For A Non-Holonomic Mobile Robot"
10th Int. Joint Conf. On Artificial Intelligence, Milán (Italy), 1120-1123, 1987.

[L&C89] Z. Li & J.F. Canny
"Robot Motion Planning With Non-Holonomic Constraints"
Memo UCB/ERL M89/13, Electronics Research Lab., U. Of California, Berkeley, 1989.

[L&C91] M.C. Lin & J.F. Canny
"A Fast Algorithm For Incremental Distance Calculation"
Proc. Of The Int. Conf. On Robotics And Automation, 1991.

[L&C92] M.C. Lin & J.F. Canny
"Efficient Collision Detection For Animation"
Eurographics Workshop On Animation And Simulation, Cambridge, (England), 1992.

[L&C93] Z. Li & J.F. Canny (Editors)
"Nonholonomic Motion Planning"
Kluwer Academic Pub., 1993.

[L&C94] C.G. Lee & H. Chung
"Global Path Planning For Mobile Robot"
Robotics & Computer-Integrated Manufacturing. An International Journal, V.11(1), 1994.

[Lin94] M.C. Lin
"Efficient Collision Detection For Animation And Robotics"
PhD. Dissertation, Dep. Electr. Eng. & Computer Sc., U. Of California, Berkeley, 1994.

[L&M95] M.C. Lin & D. Manocha
"Efficient Contact Determination Between Geometric Models"
Available In Ftp:Cs.Unc.Edu//Pub/Users/Manocha/PAPERS/COLLISION

[LMP95] M.C. Lin, D. Manocha & M.K. Ponamgi
"Fast Algortihms For Penetration And Contact Determination Between Non-Convex Polyhedral Models"
Available In Ftp:Cs.Unc.Edu//Pub/Users/Manocha/PAPERS/COLLISION

[LMT84] T. Lozano-Pérez, M.T. Mason & R.H. Taylor
"Authomatic Synthesis Of Fine-Motion Strategies For Robots"
International Journal Of Robotics Research, 3(1), 3-24, 1984.

[L&P83] C. Laugier & J. Pertin
"Automatic Grasping: A Case Study In Accesibility Analysis"
Advanced Sofware In Robotics, Ed. By Danthine &Géradin, North-Holland, NY., 1983.

[L-P76] T. Lozano-Pérez
"The Design Of A Mechanical Assembly System"
Technical Report AI-TR 397, Artificial Intelligence Laboratory, MIT, 1976.

[L-P81] T. Lozano-Pérez
"Automatic Planning Of Manipulator Transfer Movements"
IEEE Transactions On Systems, Man And Cybernetics, SMC-11(10), 681-698, 1981.

[L-P83a] T. Lozano-Pérez
"Robot Programming"
Proceedings Of The IEEE, Vol. 71, No. 7, pp 455-475, 1983.

[L-P83b] T. Lozano-Pérez
"Spatial Planning: A Configuration Space Approach"
IEEE Transactions On Computers, V. C-32(2), pp. 108-120, 1983.

[L-P87] T. Lozano-Pérez Et Al.
"Handey: A Robot System That Recognizes, Plans, And Manipulates"
IEEE Int. Conf. On Robotics And Automation, Raleigh (North Carolina), 843-849, 1987.

[L-P83b] T. Lozano-Pérez
"Some Non-Robotics Applications Of Robot Motion Planning"
The Sixth Int. Symp. On Robotics Research, pp. 71-75, 1993.

[LPW79] T. Lozano-Pérez & M.A. Wesley
"An Algorithm For Planning Collision-Free Paths Among Polyhedral Obstacles"
Communications Of The ACM, 22(10), 560-570, 1979.

[L&W77] L.I. Liebermann & M.A. Wesley
"AUTOPASS: An Automatic Programming System For Computer Controlled Mechanical Assembly"
IBM Journal Of Research And Development, 21(4), 321-333, 1977.

[Mad86] S.R. Maddila
"Decomposition Algorithm For Moving A Ladder Among Rectangular Obstacles"
IEEE Int. Conf. On Robotics And Automation, San Francisco (California), 1986.

[Mas82] M.T. Mason
"Manipulator Grasping And Pushing Operations"
Report No. TR-690, Artificial Intelligence Laboratory, MIT, 1982.

[M&B93] A.A. Masoud & M.M. Bayoumi
"Robot Navigation Using The Vector Potential Approach"
IEEE Int. Conf. On Robotics And Automation, Atlanta, Georgia, 1993.

[Mdp95a] B. Martínez & A.P. Del Pobil
"Recubrimiento Óptimo De Polígonos No Convexos Con Círculos"
Proc. VI Encuentros De Geometría Computacional, Barcelona, Spain, July 1995.

[Mdp95b] B. Martínez & A.P. Del Pobil
"Recubrimiento De Polígonos Generalizados Mediante Círculos Para La Detección De Colisiones"
Proc. VI Encuentros De Geometría Computacional, Barcelona, Spain, July 1995.

[MDW93] J.M. Manyika & H.F. Durrant-Whyte
"A Tracking Sonar Sensor For Vehicle Guidance"
IEEE Int. Conf. On Robotics And Automation, Atlanta, Georgia, 1993.

[Meg89] N. Meggido
"On The Ball Spanned By Balls"
Journal Of Discrete Computational Geometry, Vol.4, pp 605-610, 1989.

[Mel85] R.C. Melville
"An Implementation Study Of Two Algorithms For The Minimum Spanning Circle Problem"
In "Computational Geometry", Editor G.T. Toussaint, North-Holland Pub., 1985.

[Mey86] W. Meyer
"Distances Between Boxes: Applications To Collision Detection And Clipping"
Proc. Of The Int. Conf. On Robotics And Automation, 1986.

[M&T91] M. Mellado & J. Tornero
"Object Contour Surface Generation For Robotic Applications"
Symposium On Robot Control (SYROCO), Vienna, Austria, 1991.

[M&T93] M. Mellado & J. Tornero
"On The Spherical Splines For Robot Modeling"
Working Conf. On Geometric Modeling In Computer Graphics, Genove, Italy, July 1993.

[M&T96] M. Mellado & J. Tornero
"Robot Motion Planning With Artificial Potential Fields In Cartesian Space"
Int. Conf. On CAD/CAM, Robotics, & Factories Of The Future, London, UK, August 1996.

[MTV92] M. Mellado, J. Tornero & E. Vendrell
"Sistema CAD Para Programar Robots"
Revista Proyecto 2000, Nº 82. Ed. Pulsar, 1992.

[M&V94] M. Mellado & E. Vendrell
"Software Tools For Teaching In Robotics"
TEMPUS-IMPACT Workshop Teaching Control Engineering And Process Control For Engineers In Mining, Metallurgy, Mechanical Engineering, Miskolc, Hungary, 1994.

[M&V95] M. Mellado & E. Vendrell
"Software Específico Para La Docencia Práctica En Robótica"
XVI Jornadas De Automática, San Sebastián, España, 1995.

[Mye81] J.K. Myers
"A Supervisory Collision-Avoidance System For Robot Controllers"
Msc Thesis. Carnegie Mellon University, 1981.

[Nil69] N.J. Nilsson
"A Mobile Automaton: An Application Of Artificial Intelligence Techniques"
1st Int. Joint Conf. On Artificial Intelligence, Washington D.C., 509-520, 1969.

[ND+96] S.F. Navarro, J.R. Diaz, E. Vendrell & M. Mellado
"Management And Monitoring Of A Robotic Systems In A Windows Based Network"
AMPST'96, Bradford (United Kingdom), 1996.

[N&M65] Nelder & Mead (1965)
Computer Journal. Vol. 7, pp. 308, 1965.

[N&Y92] Y. Nagashima & S. Yuta
"Ultrasonic Sensing For A Mobile Robot To Recognize An Environment - Measuring The Normal Direction Of Walls"
Proc. Of Int. Conf. On Intelligent Robots And Systems, Raleigh, North Carolina, 1992.

[O&B79] J. O'Rourke & N. Badler
"Decomposition Of Three-Dimensional Objects Into Spheres"
IEEE Trans. Pattern Analysis And Machine Intelligence, Vol. 1, No. 3, July 1979.

[OLP89] P.A. O'Donnell & T. Lozano-Pérez
"Deadlock-Free And Collision-Free Coordination Of Two Robots Manipulators"
IEEE Int. Conf. On Robotics And Automation, Scottsdale, AZ, 1989

[O&Y82] C. O'Dúnlaing & C.K. Yap
"A Retraction Method For Planning The Motion Of A Disc"
Journal Of Algorithms, 6, 104-111, 1982.

[PAB80] R.J. Popplestone, A.P. Ambler & I.M. Bellos
"An Interpreter For A Language For Describing Assemblies"
Artificial Intelligence, 14(1), 79-107, 1980.

[Pas89] M. Pasquier
"Planification De Trajectories Pour Un Robot Manipulateur"
Phd. Dissertation, Institut National Polytechnique, Grenoble (France) (In French), 1989.

[PBB83] A. De Pennington, M.S. Bloor & M. Balila
"Geometric Modelling: A Contribution Towards Intelligence Robots"
Proc. ISIR, Chicago, 1983.

[PC+93] R. Pérez-Caballero, J. García-Tejedor & M. Dorronsoro
"Diseño, Simulación Y Programación Off-Line De Aplicaciones Robotizadas"
3er Congreso De La Asociación Española De Robótica, Zaragoza, Noviembre, 1993.

[Ped88] D. Pedoe
"Geometry. A Comprehensive Course"
Ed. Dover, 1988.

[PF+88] W.H. Press, B.P. Flannery, S.A. Teukolsky & W.T. Vetterling
"Numerical Recipes In C. The Art Of Scientific Computing"
Cambridge University Press, 1988.

[PJU95] I. Povazan, D. Janglová & L. Uher
"Odometry in Determination of the Position of an Autonomous Mobile Vehicle"
Int. Symp. on Measurement & Control in Robotics, Smolenice C., Slovakia, June 1995.

[P&L88] J.K. Poon & P.D. Lawrence
"Path Planning With A Potential Surface In Configuration Space"
Department Of Elecrtical Engineering, University Of British Columbia, 1988.

[PML94] M.K. Ponamgi, D. Manocha & M.C. Lin
"Incremental Algorithms For Collision Detection Between Solid Models"
TR94-061, Dept. Of Computer Science, University Of North Carolina, Chapel Hill, 1994.

[PLP90] N.S. Pollard & T. Lozano-Pérez
"Grasp Stability And Feasibility For An Arm With An Articulated Hand"
Proceedings Of The IEEE Int. Conf. On Robotics And Automation, Cincinnati, OH, 1990.

[P&R91] A. Pruski & S. Rohmer
"Multivalue Coding: Application To Autonomous Robot Path Planning With Rotations"
IEEE Int. Conf. On Robotics And Automation, Sacramento (California), 1991.

[PSF93] J. Ponce, D. Stam & B. Faverjon
"On Computing Two-Finger Force-Closure Grasps Of Curved 2D Objects"
International Journal Of Robotics Research, Vol. 12, No. 3, 1993.

[Qui94] S. Quinlan
"Efficient Distance Computation Between Non-Convex Objects"
Proc. Of The Int. Conf. On Robotics And Automation, San Diego 1994.

[Rim91] E. Rimon
"A Navigation Function For A Simple Rigid Body"
IEEE Int. Conf. On Robotics And Automation, Sacramento (California), 1991.

[R&K88] E. Rimon & D.E. Koditschek
"Exact Robot Navigation Using Cost Functions: The Case Of Spherical Boundaries In E^n"
IEEE Int. Conf. On Robotics And Automation, Philadelphia (PA), 1791-1796, 1988.

[R&K89] E. Rimon & D.E. Koditschek
"The Construction Of Analytic Diffeomorphisms For Exact Robot Navigation On Star Worlds"
IEEE Int. Conf. On Robotics And Automation, Scottsdale (AZ), 21-26, 1989.

[R&K90] E. Rimon & D.E. Koditschek
"Exact Robot Navigation In Geometrically Complicated But Topologically Simple Spaces"
IEEE Int. Conf. On Robotics And Automation, Cincinnati (Ohio), 1937-1943, 1990.

[R&T57] H. Rademacher & O. Toeplitz
"The Spanning Circle Of A Finite Set Of Points"
In "The Enjoyment Of Mathematics" (Chapter 16), Princenton University Press, 1957.

[S&K92] N.K. Sancheti & S.S. Keerthi
"Computation Of Certain Measures Of Proximity Between Convex Polytopes: A Complexity Viewpoint"
Proc. Of The Int. Conf. On Robotics And Automation, Nice, France, 1992.

[Sky91] S. Skyum
"A Single Algorithm For Computing The Smallest Enclosing Circle"
Journal Of Inform. Process. Lett., Vol 37. pp 121-125, 1991.

[S&M89] A. Sciomachen & P.G. Magnani
"A Collision Avoidance System For A Spaceplane Manipulator Arm"
Proc. Of The NASA Conf. On Space Telerobotics, Pasadena, California, 1989.

[SMV96] A. Sánchez, M. Mellado & E. Vendrell
"Integrated System For Computer Aided Robot Programming"
AMPST'96, Bradford (United Kingdom), 1996.

[S&S83a] J.T. Schwartz & M. Sharir
"On The Piano Mover' S Problem: I. The Case If A Two-Dimensional Rigid Polygonal Body Moving Amidst Polygonal Barriers"
Communications On Pure And Applied Mathematics, 36, 345-398, 1983.

[S&S83b] J.T. Schwartz & M. Sharir
"On The Piano Mover' S Problem: II. General Techniques For Computing Topological Properties Of Real Algebraic Manifolds"
Advances In Applied Mathematics, Academic Press 4, 298-351, 1983.

[S&S83c] J. Schwartz & M. Sharir
"On The Piano Movers' Problem: III. Coordinating The Motion Of Several Independent Bodies: The Special Case Of Circular Bodies Moving Amidst Polygonal Barriers"
Int. Journal Of Robotics Research, Vol. 2, No. 3, 1983.

[SSK92] H. Shashikala, , N.K. Sancheti & S.S. Keerthi
"Path Planning: An Approach Based On Connecting All The Minimizers And Maximizers Of A Potential Function"
Proc. Of The Int. Conf. On Robotics And Automation, Nice (France), May 1992

[SSK93] K. Sridharan, H.E. Stephanou & S.S. Keerthi
"On Computing A Distnace Measure For Path Planning"
Proc. Of The Int. Conf. On Robotics And Automation, 1993.

[S&W86] S. Singh & M.D. Wagh
"Robot Path Planning Using Intersecting Convex Shapes"
IEEE Int. Conf. On Robotics And Automation, San Francisco (California), 1986.

[Sud95] M. Sudolsky
"World Model Generation From Sensor Data"
Int. Symp. On Measurement & Control In Robotics, Smolenice C., Slovakia, June 1995.

[Tay76] R.H. Taylor
"Synthesis Of Manipulator Control Programs From Task-Level Specifications"
Ph.D. Dissertation, Department Of Computer Science, Stanford University, 1976.

[T&B91] L. Tarassenko & A. Blake
"Analogue Computation Of Collision-Free Paths"
IEEE Int. Conf. On Robotics And Automation, Sacramento (California), 1991.

[T&B92] J. Tornero & E.J. Bernabeu
"An Intelligent Procedure For Collision-Detection In Robotic Systems"
IFAC Symp. On Intelligent Comp. & Instruments For Control App., Málaga, Spain, 1992

[Tha88] B.K. Thakur
"Automatic Path-Planning Of Industrial Robots"
Ph.D. Dissertation, Cornell University (USA), 1988.

[THK90] J. Tornero, J. Hamlin & R.B. Kelley
"Efficient Distance Functions Using Spherical-Objects And Their Application To The Two-Puma Platform System"
Tech. Rep. CIRSSE-TR-90-64, Rensselaer Polytechnic Institute, Troy (New York), 1990.

[THK91] J. Tornero, J. Hamlin & R.B. Kelley
"Spherical-Object Representation And Fast Distance Computation For Robotic Applications"
Proc. IEEE Intl. Conf. On Robotics And Automation, Sacramento (California), 1991.

[Til90] R.B. Tilove
"Local Obstacle Avoidance For Mobile Robots Based On The Method Of Artificial Potentials"
Proc. IEEE Intl. Conf. On Robotics And Automation, Cincinnati, OH, 1990.

[TM+92] J. Tornero, M. Mellado, J. Hamlin & R.B. Kelley
"An Extended Hierarchical Robotic System Representation For Industrial Applications"
23rd Int. Symp. On Industrial Robots, ISIR92, Barcelona, Spain, October 1992

[T&M94] J. Tornero & M. Mellado
"Collision Detection And Avoidance In Robotic Systems"
ESPRIT Advanced Summer Institute (ASI94) In CIMIA, Patras, Greece, 1994

[Tor95] C. Torras
"Detección De Colisiones En 3D"
Proc. Of VI Encuentros De Geometría Computacional, Barcelona (Spain), July 1995.

[T&S89] O. Takahashi & R.J. Schilling
"Motion Planning In A Plane Using Generalized Voronoi Diagrams"
IEEE Transactions Robotics And Automation, Vol. 5, (2), 143-150, 1989.

[T&T94] F. Thomas & C. Torras
"Interference Detection Between Non-Convex Polyhedra Revisited With A Practical Aim"
Proc. Of The Int. Conf. On Robotics And Automation, San Diego, 1994.

[Udu77] S. Udupa
"Collision Detection And Avoidance In Computer Controlled Manipulators"
Ph.D. Dissertation, Dep. Of Electrical Engineering, California Institute Of Technology, 1977

[UOT83] T. Uchiki, T. Ohashi & M. Tokoro
"Collision Detection In Motion Simulation"
Computer And Graphics, Vol.7 No.3-4, 1983.

[Val90] F.J. Valero
"Planificación de Trayectorias Libres de Obstáculos para un manipulador plano"
Tesis Doctoral, Dept. Ing. Mecánica y de Materiales, Univ. Politécnica de Valencia, 1990.

[V&K87] R. Volpe & P. Khosla
"Artificial Potentials With Elliptical Isopotential Contours For Obstacle Avoidance"
Proc. Of The 26 IEEE Conf. On Decision And Control, Los Angeles, CA, 1987.

[V&M95] E. Vendrell & M. Mellado
"Robotics Tools And Experiences In Manufacturing Systems"
Journal Of Studies In Informatics And Control, Vol. 4, No. 3, September 1995.

[VMT91] E. Vendrell, M. Mellado y J. Tornero
"Programación De Robots Mediante Una Herramienta CAD Comercial"
Segundo Congreso De La Asociación Española De Robótica, Zaragoza, Spain, 1991.

[VMT92a] E. Vendrell, M. Mellado & J. Tornero
"Robot Programming Based On Commercial CAD Systems"
Third IFAC Symposium On Low Cost Automation (LCA92), Vienna, Austria, 1992.

[VMT92b] E. Vendrell, M. Mellado y J. Tornero
"Programación De Robots. Desarrollo De Un Entorno Basado En Un Sistema CAD"
Revista Automatización Integrada & Robótica, N 72. Ed. Pulsar, 1992.

[VMV93] E. Vendrell, M. Mellado y E. Vidal
"Sistema Educacional De Programación De Robots Basado En PC"
3er Congreso De La Asociación Española De Robótica, Zaragoza, España, 1993.

[VTM92] A. Valera, J. Tornero & M. Mellado
"An Experience In Robotics Education"
Europ. Sem. On Automation & Control Technology Education, Dresden, Germany, 1992.

[War89] C.W. Warren
"Global Path Planning Using Artificial Potential Fields"
IEEE Int. Conf. On Robotics And Automation, Scottsdale, AZ, 1989

[War90] C.W. Warren
"Multiple Robot Path Coordination Using Artificial Potential Fields"
Proc. IEEE Intl. Conf. On Robotics And Automation, Cincinnati, OH, 1990.

[Wel91] E. Welzl
"Smallest Enclosing Disks (Balls And Ellipsoids)"
In "New Results And New Trends In Computer Science", H. Maurer, Springer Verlag, 1991.

[Wid74] C. Widdoes
"A Heuristic Collision Avoider For The Stanford Robot Arm"
Stanford C.S. Memo 227, June 1974

[W&J88] C.H. Wu & C.C. Jou
"Design Of A Controlled Spatial Curve Trajectory For Robot Manipulators"
27th Conf. On Decision And Control, Austin, Texas, 1988.

[WK+95] R.H. Wilson, L. Kavraki, J.C. Latombe & T. Lozano-Pérez
"Two-Handed Assembly Sequencing"
The International Of Robotics Research, Vol.14, No. 4, August 1995.

[WLL92] R.H. Wilson, J.C. Latombe & T. Lozano-Pérez
"On The Complexity Of Partitioning An Assembly"
Technical Report No. STAN-CS-92-1458, Dep. Of Computer Science, Stanford U., 1992.

[Wol76] P. Wolfe
"Fnding The Nearest Point In A Polytope"
Mathematical Programming Study, Vol 11, No2, Balinski Ed. Amsterdam, Oct. 1976.

[Z&B92] Y. Zhao & S.L. BeMent
"Kinematics, Dynamics And Control Of Wheeled Mobile Robots"
Proc. Of The Int. Conf. On Robotics And Automation, Nice (France), May 1992

[Zel94] A. Zelinsky
"Using Path Transforms To Guide The Search For Findpath In 2D"
The International Journal Of Robotics Research, Vol.13, No.4, August 1994.

[Z&H94] E. Zussman & T. Horsch
"A Planning Approach For Robot-Assisted Multiple-Bent Profile Handling"
Robotics & Computer-Integrated Manufacturing. An International Journal, Vol. 11 (1), 1994.

[ZRL92] S. Zeghloul, P. Rambeaud & J.P. Lallemand
"A Fast Distance Calculation Between Convex Objects By Optimization Approach"
Proc. Of The Int. Conf. On Robotics And Automation, Nice (France), May 1992

√