Planificación basada en Percepción Activa para la Navegación de un Robot Móvil

Miguel Schneider Fontán
Instituto de Automática Industrial
km. 22,800 N III, La Poveda, Arganda del Rey
Madrid 28500, España

5 Septiembre 1996

Agradecimientos

Tras casi 5 años de trabajo* ha llegado la hora de mostrar mi agradecimiento a las numerosas personas que de alguna u otra forma han colaborado en este trabajo.

Quiero comenzar agradeciendo a mi director de tesis **Ricardo García Rosa** su confianza y apoyo, mostrados durante todos estos años, en mi trabajo. Asimismo, a **Maria Teresa de Pedro**, por su paciencia, y sabios consejos a la hora de la redacción final de esta tesis. Sin su aportación, el resultado final no sería el mismo.

Tengo numerosas razones para incluir en los agradecimientos a mis compañeros de fatigas de estos años en el IAI: A **Lissette Lemus**, por tener siempre una historia asombrosa que contar de su Cuba natal con una sonrisa de oreja a oreja; a **Marco Tulio Andrade**, por mantenerme informado de las desgracias de nuestro Atleti cuando me encontraba lejos; a **Pablo Bustos** por las numerosas conversaciones "profundas" y que tantas ideas me aportaron en las horas que seguían al cafecito de después de la comida; a **Jose Luis Pedraza**, por haberme servido de guía, y por haberme maravillado con sus extensos (¿infinitos?) conocimientos y su humildad, generosidad y modestia; a **Adelaida Agote** por su simpatía, y por alegrar el dia con su mera presencia; y a **Jorge Gasós**, por sus valiosos consejos (y no sólo los que me daba jugando al DOOM).

A mis amigos de los tiempos de la facultad: **Ana, Marisa, Moche, Katzalin y Salva**. Me aguantaron durante la carrera y lo siguen haciendo. ¡Eso sí que tiene mérito!

A lo largo de estos años he tenido la oportunidad de permanecer durante cortos periodos de tiempo en centros de investigación en el extranjero, donde he hecho numerosos amigos. Especialmente quiero agradecer a **Tomás Lozano-Pérez**, quien me dio la oportunidad de permanecer durante tres meses en el laboratorio de Inteligencia Artificial del MIT que nunca olvidaré. Tampoco olvidaré su predisposición, amabilidad y generosidad mostrada entonces y que aún me muestra hoy en día cuando tiene ocasión. A **Maja Mataric** va dedicada una parte muy importante de este trabajo: Por su entusiasmo, motivación, paciencia y "useful hints" que tanto me han ayudado en la composición final de esta tesis. Tambien querría hacer una especial mención a mis amigos de Boston: **Vicente Linares**, por sacarme de marcha los Jueves por la noche y, sobre todo, por ser todo un amigo; **Karina Baum** y **Pablo Funes** por cuidar los sábados por la mañana de mí. A **Dani Goldberg** y **Barry Werger** por hacer del Interaction Lab. un lugar entretenido y divertido (para aquellos que hablan español, claro).

La tesis la he conseguido terminar de redactar por capricho de la metereología, que ha permitido que los fines de semana pudiese hacer lo que más me gusta: Esquiar. En concreto, **La Pinilla** ha sido, sin lugar a dudas, un factor importantísimo en la consecución final de este trabajo, al igual que las noches de Riofrío y el posterior Infierno. Gracias a **Tomi** y **Laura** (por cuidar de mí en momentos de "perjudicamento"), **Jose, Nadia, Javi** y a todo el mundo.

*Más de 30.000 líneas de código, unos 125.000 kilometros recorridos en coche desde mi casa al IAI, que son unos 8.750 litros de gasolina y, a razón de unas 100 pts por litro (tirando por lo bajo), son un gasto de 875.000 pts.

Y como los buenos toreros, dejo para el final lo más importante: Gracias a **Lourdes** por todo. La verdad es que tendría que escribir una nueva tesis para poder expresar todo mi agradecimiento: Por tus consejos, por aguantarme, por tu bondad, por escuchar los rollos sobre arquitecturas de robots, por tranquilizarme en los peores momentos. En definitiva: Gracias por tu amistad, que siempre me has demostrado.

Finalmente quiero agradecer a las personas más importates de mi vida, y a las que va dedicada esta obra y que siempre han estado a mi lado, han confiado en mí, me han animado y me han apoyado en todo lo que he hecho: **Mis padres**. Gracias, Vielen Dank.

Índice General

Índice de Tablas

Índice de Figuras

Capítulo 1

Robots Móviles Autónomos

1.1 Introducción

Uno de los principales objetivos de la Inteligencia Artificial es el de profundizar en el conocimiento de la inteligencia natural mediante la generación y el análisis de comportamientos artificales [75]. Los robots móviles presentan la plataforma ideal para investigar sobre los procesos de percepción y acción autónomos e inteligentes, y por tanto son una herramienta muy útil para validar las teorías planteadas por la Inteligencia Artificial.

Los primeros desarrollos de la robótica móvil se produjeron para su aplicación en entornos industriales, perfectamente estructurados, de tal forma que la incertidumbre sobre el conocimiento del mundo se reducía a un nivel mínimo. Así, cuanto mayor es el conocimiento sobre el entorno de trabajo, menos necesarias han de ser las capacidades sensoriales y de razonamiento del robot. De hecho, la gran mayoría de robots industriales trabajan en entornos tan estructurados que apenas requieren sensores externos ni necesitan tomar decisiones, limitándose a repetir una determinada secuencia de movimientos.

Sin embargo, no todos los entornos en los que puede trabajar un robot móvil están completamente estructurados. Es muy difícil conocer a-priori la localización exacta de todos los elementos presentes en el entorno y planificar todas las posibles acciones que deba tomar el robot [2]. Por tanto, es necesario que el robot sea capaz de tomar decisiones propias, mientras tiene en cuenta las directrices de un plan general. Es decir, el robot ideal es aquel que realiza las acciones propuestas en un plan proporcionado por un usuario y, además, es capaz de actuar de forma autónoma ante eventos imprevistos.

1.1.1 Evolución de las Arquitecturas para Robots Móviles Autónomos

Propuesta Tradicional

Inicialmente, los investigadores en robótica móvil trataron de conectar los ingenios manipuladores de símbolos que se habían desarrollado en el campo de la Inteligencia Artificial a los sensores y actuadores de los robots móviles [88]. Uno de los principales objetivos era el de generar un modelo simbólico del mundo (escenario cartesiano) a partir de la información sensorial, de tal forma que la máquina de inferencia pudiera razonar y planificar a-priori los movimientos del robot [42]. Primero se especificaban el formato de la base simbólica de conocimiento y la máquina de inferencia, y más tarde se desarrollaban los módulos de percepción que deberían proveer los símbolos necesarios: Por ejemplo, el ingeniero de sistemas que estaba desarrollando la base de conocimiento y la máquina de inferencia del robot suponía que el ingeniero de "visión artificial" realizaría un módulo que transformase las imágenes percibidas en símbolos.

En definitiva, la propuesta de la Inteligencia Artificial clásica era la de realizar una **planificación a largo plazo** basándose en el modelo del mundo desarrollado en un instante dado. Si el robot se encontraba ante un evento imprevisto, el sistema de planificación realizaba un nuevo plan de actuación.

Sistemas Distribuidos

A mediados de los años ochenta, Brooks [26], en respuesta a esta visión tradicional de la Inteligencia Artificial aplicada a la robótica móvil, introdujo una nueva filosofía de diseño que él definió como *Sistemas Reactivos*. Reemplazó la modularización funcional por módulos generadores de comportamientos, o agentes. Una jerarquía de procesos se hacía cargo del sistema; cada uno de estos procesos era extremadamente simple y dotado de una funcionalidad completa (evitar obstáculos, bordear una pared, etc). El corto camino entre los sensores y los actuadores (desde un punto de vista computacional) proporcionaba una respuesta en tiempo real del robot ante cambios en el entorno. La implementación del sistema se realizaba de abajo a arriba, donde primero se trataba de asegurar la supervivencia del robot, mediante comportamientos básicos como evitar obstáculos, y después se le iban asignando nuevas y más complejas aptitudes, como la confección de mapas del entorno [74], o la manipulación de objetos [29].

Tanto las arquitecturas reactivas como las basadas en comportamientos [73] se fundamentan en la **acción inmediata** ante el estímulo percibido. Los objetivos del sistema son múltiples, y las condiciones del entorno dictan cual de ellos es el prioritario.

Los robots diseñados mediante este tipo de técnicas se asemejan en su comportamiento a los insectos [92], y son utilizados como herramienta de estudio en el campo de la *Vida Artificial* [62], que trata de emular los procesos básicos de la vida en ordenadores. Asimismo, las arquitecturas multi–agente se han utilizado en otros campos de la inteligencia artificial y en entornos complejos [44], como buscadores de información en la Internet [40] o la cooperación de un grupo de robots móviles [75][85].

1.2 Objetivo de la Tesis

El objetivo del presente trabajo ha sido la integración de las capacidades de acción inmediata ante los estímulos percibidos, con la consecución de un plan propuesto por un usuario. Es decir, el robot móvil es capaz de llevar a cabo un plan propuesto por un usuario, mientras responde a los eventos no previstos en dicho plan de forma autónoma y en tiempo real.

De esta manera, se ha conseguido especificar planes de la misma manera a como se le indica a una persona cómo llegar de un lugar a otro basándose en marcas del entorno y una serie de consejos:

> "*Para llegar a la sala de fotocopiadoras, sigue la pared por la derecha y busca la puerta a tu izquierda ...*"

En la especificación de un plan a largo plazo, el usuario ha de tener en cuenta las habilidades de percepción y de acción del robot móvil, como, por ejemplo, la capacidad de detectar una serie de marcas en el entorno (balizas), y de dirigirse a ellas. Por otra parte, el robot posee un cierto "sentido común", que permite que el usuario no deba indicarle de forma explícita acciones y comportamientos a seguir que son obvios. Por ejemplo, no hay que indicar al robot que evite los obstáculos mientras se dirige a una posición determinada, o que pare de seguir una pared si ya ha localizado el objetivo.

1.2.1 Estructura del Sistema

En la Figura 1.1 se puede observar la composición del sistema de control del robot móvil. Se divide en tres partes: Los agentes motores de navegación, los agentes motores de visión y el plan. A continuación realizaremos un breve repaso de cada uno de estos subsistemas:

Agentes Motores de Navegación

Los agentes motores del robot son capaces de interpretar la información sensorial suministrada por los sensores de ultrasonido con los que va equipado para realizar las tareas de navegación de forma autónoma. Debido a los numerosos errores cometidos en las lecturas de los sensores, se ha generado una representación local del entorno que se construye basándose en la información

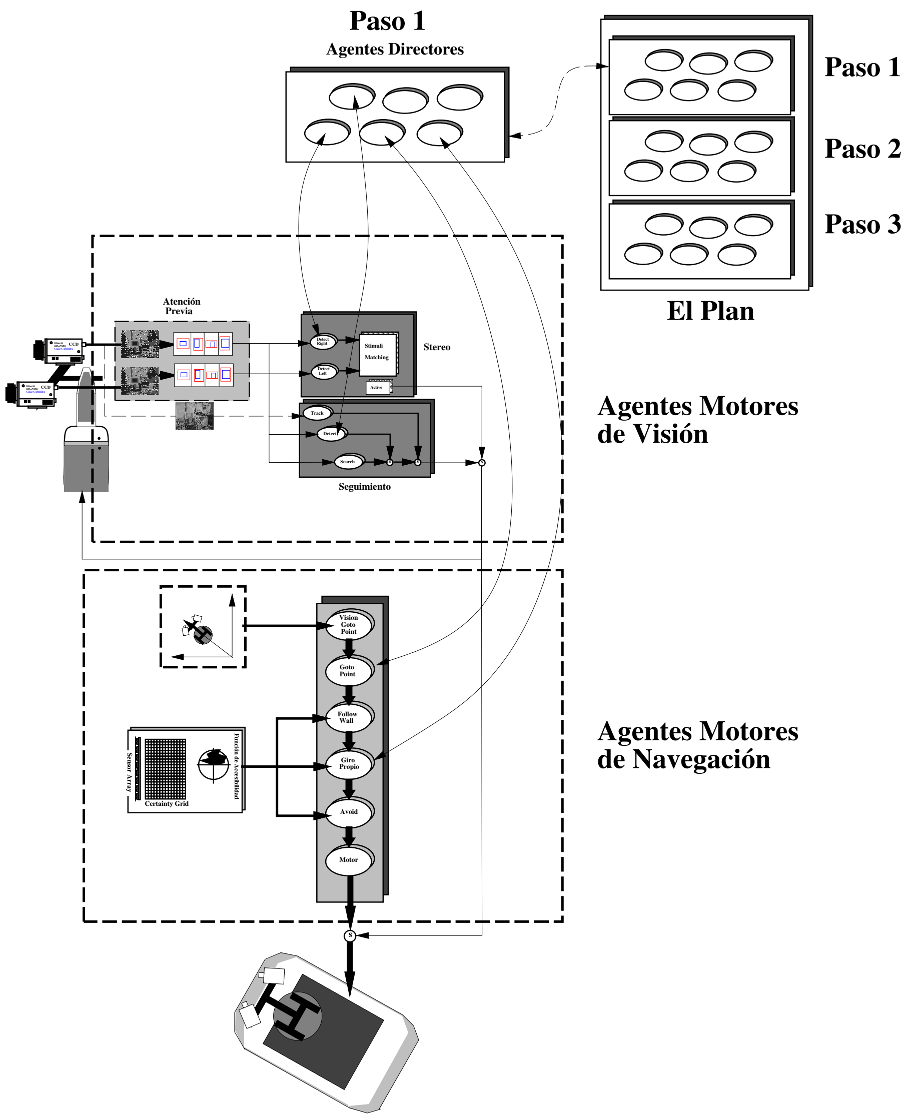


Figura 1.1: Sistema Completo.

proporcionada por éstos para eliminar el ruido, la imprecisión en las medidas y la información superflua para la navegación del robot.

La arquitectura multi–agente que hemos implementado sirve de soporte formal para la implementación del conjunto de comportamientos motores de navegación que evitan obstáculos, se dirigen a localizaciones del entorno, siguen paredes, etc.

Agentes Motores de Visión

En los últimos años se ha optado por considerar la visión artificial como un proceso que está íntimamente relacionado con el entorno de trabajo del robot y la tarea a realizar [10] [52] [1]. En lugar de considerar la visión como un proceso estático, diseñando sofisticados algoritmos que traten de reconstruir el entorno, se ha optado por explotar las características del entorno y trabajar más con términos cualitativos que cuantitativos [3]. La naturaleza es el mejor ejemplo de estos sistemas: Los organismos vivos están especialmente "diseñados" para sobrevivir en su habitat [33], tanto por sus aptitudes sensoriales como motoras. Por ejemplo, el sistema visual de una rana está especializado en la detección de objetos en movimiento. En principio, en el habitat de la rana los objetos pequeños en movimiento son insectos que le servirán como alimento, por lo que lanzará su lengua como acto reflejo en cuanto los detecte. Sin embargo, una rana rodeada de insectos muertos suspendidos de un hilo, se muere de hambre al no ser capaz de detectar el estímulo que apuntaba al alimento.

Por tanto, y para simplificar los procesos de búsqueda y reconocimiento de los hitos del entorno que constituirán las metas del robot, se ha optado por señalizar dichos lugares utilizando balizas compuestas por combinaciones de colores que no se den habitualmente en entornos interiores. De esta manera, una baliza compuesta por un cuadrado azul inscrito dentro de uno rojo indicaría la presencia de una puerta, siendo posible asignar diferentes balizas a cada hito del entorno (puertas, pasillos, mesas, etc.)

Los agentes de visión de nuestro sistema, están especialmente diseñados para participar en las tareas especificas del robot. Es decir, se han diseñado con un *propósito* muy determinado: Buscar, detectar, seguir y conocer la localización de un conjunto de balizas compuestas por una combinación de colores que no se dan habitualmente en entornos interiores.

El sistema de visión artificial funciona en paralelo con el sistema de navegación. Es decir, el sistema es capaz de seguir un objeto determinado a la vez que se evitan los obstáculos imprevistos en la trayectoria del robot.

Planificación

La planificación de las actividades de un robot móvil implica la necesidad de indicar explícitamente, de forma flexible, un conjunto de metas al robot. Ahora bien, partiendo de la base de que la actuación del robot móvil se ha generado como un conjunto de agentes, pretendemos conseguir una transición suave entre dichos agentes (de bajo nivel) y la estructura que los controla. Es decir, hemos diseñado la comunicación con el robot una vez que se ha establecido su modelo cognitivo (agentes, atributos y símbolos asociados a ellos), en lugar de introducir un *sistema de procesamiento simbólico tradicional* independiente*, que planifique las acciones del robot [4].

Un **plan** está constituido por un conjunto de pasos, donde cada paso está formado por una meta y conjunto de **consejos** de cómo llegar a ella. Tanto las metas como los consejos de cada paso están basados en las **capacidades de acción y percepción** del robot móvil. En este sentido, el robot es capaz de evitar obstáculos, seguir paredes e ir a una posición determinada. Asimismo, es capaz de buscar, reconocer, seguir y estimar la posición 3D de un conjunto de balizas visuales. Todas estas habilidades del robot están implementadas como un conjunto de agentes independientes que se ejecutan en paralelo. Es decir, en nuestro sistema la planificación de las acciones del robot a largo plazo no se considera como un proceso independiente de la percepción y la acción inmediata.

El sistema es una extensión de los sistemas basados en comportamientos, en el que además se ha logrado establecer un lenguaje de comunicación entre el robot y un usuario, que permite definir un conjunto de metas diferentes a alcanzar por el robot. Se han definido nuevos agentes de más alto nivel (agentes directores), los cuales inspeccionan y modifican los atributos de los agentes motores (de navegación y de visión). Los agentes directores no recibirán entradas de los sensores, ni sus salidas afectarán directamente a los actuadores del robot.

1.2.2 Un Recorrido del Robot

La Figura 1.2 muestra la trayectoria seguida por el robot móvil durante el transcurso de la ejecución de un plan que le fue propuesto, así como la localización de las balizas y de los obstáculos que debía sortear. El plan consistía en llegar secuencialmente a cada una de las cuatro balizas. Durante la realización del plan, el robot tuvo que evitar obstáculos, seguir paredes a derecha e izquierda, detectar un conjunto de balizas diferentes, e inferir su localización en el espacio mediante el sistema de estereovisión.

El usuario propuso un plan basándose en el conocimiento que tenía del entorno (orientación relativa entre balizas, paredes, etc.), y le propuso un conjunto de metas al robot junto con una serie de consejos para su consecución.

1.3 Robot Móvil

El robot móvil utilizado para la implementación práctica de la presente tesis, es un robot comercial de la fábrica *RoboSoft*, modelo *Robuter*. Posee un cinturón compuesto por 12 sensores de ultrasonido *Polaroid* alrededor del robot, además de sensores de odometría para estimar

*Arquitecturas Híbridas

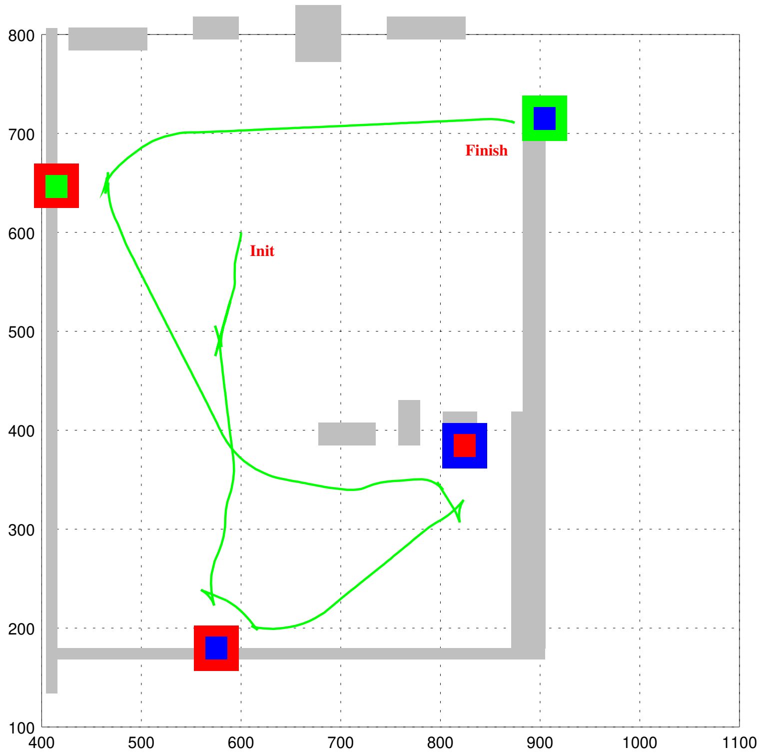


Figura 1.2: Realización de la tarea

Figura 1.3: Robot Móvil *Robuter*

los desplazamientos realizados. La forma de comunicación con el robot móvil se ha realizado mediante una conexión con un cable serie desde una estación de trabajo UNIX, donde se ejecutan todos los procesos.

El sistema de visión artificial está montado sobre una torreta con dos grados de libertad (aunque en la implementación realizada sólo se realizan movimientos azimutales) y está compuesto de dos cámaras de color conectadas a una tarjeta digitalizadora *S2200* de *Imaging Technology*. Para implementar los agentes motores de visión, se desarrolló un entorno de programación basado en la programación orientada a objetos para el tratamiento de imágenes [84].

1.4 Estructura de la Tesis

En el Capítulo 2 se realizará un repaso del concepto de las arquitecturas multi–agente, las implementaciones más destacadas de robots móviles realizadas hasta la actualidad, y se expondrán los diferentes problemas que provocan su utilización. Finalmente, se expondrá la arquitectura realizada para el control de los agentes motores de navegación.

En el Capítulo 3 se detallarán los problemas de la utilización de los sensores de ultrasonido para la navegación autónoma de robots móviles. Se propone utilizar un mapa del entorno para reducir la incertidumbre en el conocimiento del entorno, que servirá como base para generar la *Función de Accesibilidad*, cuya definición se expone en el Capítulo 4.

La *Función de Accesibilidad* será utilizada por los agentes motores de navegación, de cuya implementación trata el Capítulo 5.

El sistema de visión artificial del que va equipado el robot móvil se describirá en el Capítulo 6. En este capítulo se describe el sistema de búsqueda, detección y seguimiento de las balizas, así como el sistema de estereovisión que determina la localización 3–D de los objetos.

Finalmente, el Capítulo 7 tratará sobre la planificación de alto nivel de las tareas del robot. Se introducirá el concepto de agentes directores, y cómo éstos actúan sobre los atributos de los agentes motores. Finalmente se describirá como los agentes motores y sus atributos actúan como símbolos bien enraizados, y su utilización, por parte de un usuario, para expecificar los planes que debe acometer el robot.

Capítulo 2

Arquitectura de Control

Una arquitectura de control provee al diseñador de una serie de principios y reglas que seguir a la hora de implementar un sistema. En nuestro caso, estamos interesados en implementar un sistema que sea capaz de interpretar la información sensorial suministrada por los sensores con los que vaya equipado el robot móvil, y poder realizar una serie de tareas básicas. La tendencia y la filosofía de la construcción de sistemas de navegación de robots móviles han variado de forma sustancial en los últimos años. En el presente capítulo se realizará un resumen de las diferentes alternativas que se han propuesto a lo largo de la historia, centrándonos principalmente en las llamadas arquitecturas reactivas y en las basadas en comportamientos. Finalmente describiremos de un modo formal la arquitectura multi-agente **MAgDA**, que servirá de soporte formal a la implementación del conjunto de comportamientos motores de navegación. Este capítulo se complementa con los que le siguen, que se centran con mayor detalle en la generación de un mapa del entorno, así como en la creación de una representación local del entorno y la implementación física de los comportamientos de navegación.

2.1 Introducción

Desde un principio, el principal objetivo que se impuso la comunidad científica que investigaba en el área de la robótica móvil era el de construir ingenios que, a partir de la información proporcionada por un sistema sensorial, como sensores de ultrasonidos, infrarojos, odométricos, visión artificial, etc., fueran capaces de moverse inteligentemente por el entorno y realizar una serie de tareas de forma autónoma. Un robot móvil debe ser capaz de procesar la información sensorial suministrada por los sensores y adoptar una acción de control que le lleve a un estado definido como meta.

Realizando un repaso histórico, se puede asegurar que gran parte del entusiasmo inicial por la Inteligencia Artificial en sus orígenes fue debido al éxito con que se resolvieron ciertos problemas, como jugar al ajedrez o la resolución automática de teoremas. En la *Inteligencia Artificial*, que podemos definir como *Clásica* o *Inteligencia Artificial Basada en el Conocimiento*, todos los esfuerzos se concentraron en definir una serie de estructuras simbólicas formales y métodos de inferencia que permitiesen un razonamiento sobre estas estructuras. Es decir, se trataba de definir una base de conocimiento y un sistema inteligente (máquina de inferencia) que fuese capaz de interpretar esa información simbólica y obtener resultados (en forma de símbolos) de forma automática. Siguiendo ese tipo de filosofía, los investigadores de la rama de la robótica móvil trataron de conectar los ingenios manipuladores de símbolos a los sensores y actuadores de los robots móviles. En principio, los módulos sensoriales deberían proveer a la base de conocimiento de un conjunto de símbolos que identificasen cada uno de los elementos que estuviesen percibiendo. Uno de los principales objetivos era el de generar un modelo simbólico del mundo (escenario cartesiano) a partir de la información sensorial, de tal forma que la máquina de inferencia pudiera razonar y planificar a-priori los movimientos del robot. Los requerimientos de este tipo de sistemas son de arriba a abajo. Es decir, primero se especificaban el formato de la base simbólica y la máquina de inferencia y, más tarde, se desarrollaban los módulos de percepción que deberían proveer los símbolos necesarios: Por ejemplo, el ingeniero de sistemas que estaba desarrollando la base de conocimiento y la máquina de inferencia del robot suponía que el ingeniero de "visión artificial" realizaría un módulo capaz de transformar las imágenes percibidas en símbolos de ese tipo.

A mediados de los años ochenta, Rod Brooks, en respuesta a la visión tradicional de la Inteligencia Artificial, introdujo una nueva filosofía de diseño aplicada a la robótica móvil que él definió como *Arquitecturas Reactivas*. Observando que *"los robots desarrollados hasta entonces eran excesivamente lentos y en absoluto robustos"* [62] reemplazó la modularización funcional por módulos generadores de comportamientos. Una jerarquía de procesos se hacía cargo del sistema; cada uno de estos procesos era extremadamente simple, y el corto camino entre los sensores y los actuadores (desde un punto de vista computacional) proporcionaba una respuesta en tiempo real del robot ante cambios en el entorno. La implementación del sistema se realizaba de abajo a arriba, de modo que primero se trataba de asegurar la supervivencia del robot, y después se le iban asignando nuevas y más complejas aptitudes. Por otro lado, las *Arquitecturas Basadas en Comportamientos**, al igual que las arquitecturas reactivas, están distribuidas y se componen de una colección de comportamientos que se ejecutan concurrentemente, sin necesidad de un planificador o árbitro central. En claro contraste con las arquitecturas reactivas, los comportamientos son más "poderosos" que las reglas puramente reactivas, puesto que incorporan

*Del inglés **Behavior-based architectures.**

memoria y un preprocesamiento de la señal sensorial.

Recientemente, las llamadas *Arquitecturas Híbridas* han ganado popularidad. Estas arquitecturas ofrecen un compromiso entre las arquitecturas puramente reactivas y las arquitecturas tradicionales basadas en un planificador. Se componen de un sistema reactivo para ejecutar tareas de bajo nivel, y de un planificador para establecer tareas de más alto nivel. Estas arquitecturas separan el sistema en dos o más partes independientes, que se comunican. En la mayor parte de los casos, los procesos reactivos de bajo nivel se encargan de la integridad del robot en cada instante, mientras que el planificador selecciona toda una serie de acciones que ejecutar en el futuro.

En los siguientes apartados se realizará un resumen de las diferentes alternativas que se han detallado en los párrafos anteriores, enumerando los objetivos seguidos y los resultados obtenidos en cada uno de los casos. A su vez se repasarán algunas de las implementaciones físicas más importantes. No nos centraremos en resultados teóricos ni en sistemas que hayan sido implementados en simulación ya que, a nuestro parecer, este tipo de sistemas no pueden ser comparados con aquellos que han sido implementados y comprobados en el "mundo real". Los simuladores ofrecen muchas ayudas de forma preliminar a los diseñadores de robots móviles, pero su construcción implica una simplificación del mundo que en numerosas ocasiones lleva a resultados engañosos. Es cierto que se pueden utilizar simuladores en entornos bien conocidos, como, por ejemplo, en el caso de simuladores aéreos. Sin embargo, en este caso la dinámica y las características físicas de los aviones es bien conocida, no así la de los robots móviles. Por otro lado, y de modo mucho más importante y fundamental, el entorno en el que se han de mover (en este caso, volar) los aviones es evidentemente más simple que un entorno no estructurado como por el que se tiene que mover un robot móvil. Simular los sensores de un robot no es una tarea en sí excesivamente compleja, e incluso se pueden encontrar sistemas que reproducen las características de numerosos sensores con gran fiabilidad. Lo realmente complicado es simular un complejo "mundo real", lleno de incertidumbres, en donde esos sensores han de realizar sus medidas y el robot ha de actuar.

2.2 Arquitecturas Tradicionales

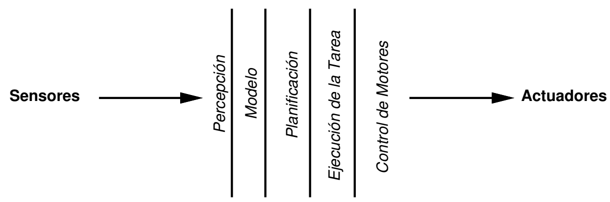


Figura 2.1: Descomposición tradicional de la arquitectura de control de un robot móvil en módulos funcionales.

Las arquitecturas tradicionales se caracterizan por dos fundamentos muy determinados:

1. La descomposición funcional de la arquitectura.

2. La generación de un modelo del mundo.

2.2.1 Descomposición funcional

El método tradicional de diseño de un sistema de navegación se basa en que las tareas a realizar por el robot móvil se encuentran en un nivel de abstracción más elevado que el conjunto de variables de control del robot, por lo que no se utilizan directamente estas variables en la resolución de las tareas. La solución que se aporta es la de descomponer la tarea del robot en un refinamiento progresivo de la solución, de forma que se definan una serie de funciones que reduzcan progresivamente el nivel de abstracción del problema para acercarse al nivel de las variables de control del robot. Desde el punto de vista del flujo de la información, esta metodología realiza una descomposición en una serie de unidades funcionales como la percepción, la generación de un modelo de lo percibido, la planificación de una trayectoria sobre este modelo, la ejecución de la tarea, y finalmente el control de los motores del robot móvil (ver Figura 2.1). Este flujo es cíclico, con lo que se repite para cada una de las acciones del robot móvil.

2.2.2 Modelo del Mundo

Los sistemas construidos siguiendo las líneas marcadas por la Inteligencia Artificial clásica tratan de modelar el dominio y pueden responder a cuestiones relacionadas con él. Por lo general son sistemas "cerrados", en el sentido de que no hay una interacción directa con el dominio problema del que se ha pre-codificado un cierto conocimiento. El tiempo de respuesta del sistema no es una de las prioridades, ya que se supone que el entorno no va a cambiar mientras se está calculando la solución. Por otro lado, los defensores de este tipo de filosofía siempre han abogado por un incremento en las prestaciones de los ordenadores y sistemas sensoriales en un futuro próximo. Pero, en definitiva, lo que realmente caracteriza a este tipo de sistemas es la necesidad de generar un modelo simbólico del mundo a partir de la información suministrada por los sensores. Este modelo del mundo está lleno de incertidumbres, sujeto a errores, y su cómputo requiere una gran cantidad de tiempo y de recursos. Independientemente de la tarea que se vaya a realizar, el modelo del mundo se construye **siempre**, a pesar de que gran parte de la información contenida en él no sea necesaria para el objetivo que trata de alcanzar el sistema.

2.2.3 Implementaciones

Existen innumerables trabajos y sistemas que han seguido esta línea teórica, pero no es el fin del presente trabajo presentar un exhaustivo examen de los diferentes sistemas sino presentar los trabajos más representativos.

Shakey

En 1969 se terminó en el Instituto de Investigación de Stanford la construcción de *Shakey* [77], que fue el primer y único robot móvil hasta la fecha controlado fundamentalmente por medio de programas de razonamiento. *Shakey* estaba equipado con un sistema remoto de TV y su control le permitía moverse en el entorno basándose únicamente en la información suministrada por las imágenes que proveían las cámaras con las que iba equipado. Utilizaba toda una serie de algoritmos de estereo-visión que le permitían localizar objetos y representarlos en tres dimensiones.

El sistema únicamente funcionaba cuando los objetos eran simples, planos y de un solo color, así que se construyó para el robot un entorno especial que constaba de varias habitaciones con paredes despejadas, en el que había un cierto número de bloques y cuñas pintadas de un solo color. Tanto el problema como el entorno eran inventados, pero proporcionaron una motivación y comprobación del programa razonamiento STRIPS† [42]. El funcionamiento del sistema permitía pedir al robot que realizara una tarea, y STRIPS elaboraba un plan basado en los limitados movimientos que podía efectuar el robot, cada uno de los cuales tenía condiciones previas y consecuencias probables. Sin embargo, el resultado de una acción no siempre era el que se esperaba. La capacidad de *Shakey* para enfrentarse a estas irregularidades era limitada, ya que tenía que ajustar de nuevo sus descripciones internas y volver a plantearse sus movimientos si esas condiciones no eran las que había supuesto.

Tal y como relata Hans Moraveck, uno de los investigadores que integró el equipo que diseño y construyó a *Shakey*, en su libro *Mind Children* [79]: *"Shakey era impresionante como concepto, pero digno de compasión cuando se movía. Cada uno de los movimientos del robot, cada ojeada que daba con su cámara, consumía aproximadamente una hora de tiempo de ordenador y entrañaba una alta probabilidad de fracaso"*

HILARE

HILARE [45] es un excelente ejemplo de un sistema que utiliza una metodología clásica de control. Este robot estaba equipado con un cinturón formado por 14 sensores de ultrasonido, un sistema de triangularización infra-rojo y una cámara dotada de un haz láser. Partiendo de la base de que el entorno de un robot móvil es complejo, este sistema trataba de construir un modelo del mundo lo más preciso posible para poder localizar al robot dentro de él. Este modelo no era global y único, sino que estaba constituido por tres niveles de representación diferentes:

1. En el primer nivel, los datos que provenían de los sensores eran plasmados en un modelo geométrico del mundo bidimensional, y se utilizaban aproximaciones poligonales para describir los objetos percibidos.
2. En el segundo nivel, el modelo geométrico era utilizado para deducir las propiedades

†**S**tanford **R**esearch **I**nstitute **P**roblem **S**olver

topológicas del entorno. El espacio se estructuraba generando polígonos convexos que se correspondían con los bordes y esquinas de los objetos percibidos.

3. En un tercer y último nivel, se construía un modelo semántico del mundo etiquetando los objetos del nivel anterior.

La gran ventaja que ofrecía el modelo del mundo en diferentes niveles, era que diferentes módulos de control del robot móvil podían acceder a la diferentes partes de la información y comprobar su consistencia. Sin embargo, la gran desventaja era la complejidad creciente en la actualización de los modelos, así como su baja tolerancia a fallos. Otro de los grandes problemas era el de la localización del robot móvil dentro del modelo del mundo. La solución aportada por los autores incluía la utilización de balizas infra-rojas para la localización en coordenadas absolutas del robot móvil, el uso de sensores odométricos e inerciales así como la navegación mediante el uso de elementos del entorno que sirviesen como referencias relativas.

2.3 Arquitecturas Reactivas

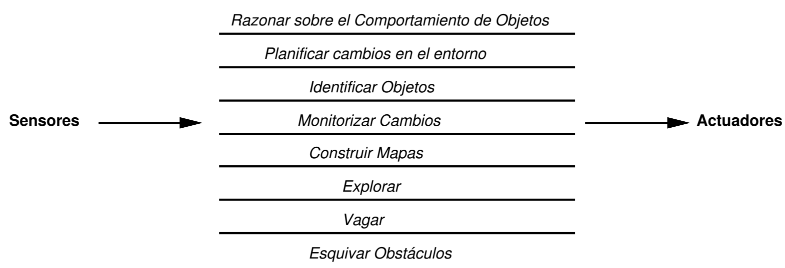


Figura 2.2: Descomposición basada en módulos generadores de comportamiento.

La idea básica en la que se sustentan las arquitecturas reactivas es la descomposición jerárquica del problema en niveles de competencia horizontales. Se construyen niveles del control para cada uno de los niveles de competencia, añadiendo nuevas capas a las ya existentes. Se comienza construyendo un robot completo que logre acometer una competencia de grado cero, y depurando este control se consigue un robot que realiza a la perfección las especificaciones encomendadas a ese nivel. Tal y como se puede observar en la Figura 2.2, un nivel cero de competencia sería aquel que asegurase la supervivencia del robot en el entorno. Es decir, un nivel que reaccione a la señal sensorial de tal forma que cuando el robot se encuentra muy cercano a un obstáculo, lo evite. Sobre esta capa de control se puede añadir una segunda capa, en la que el objetivo de la misma sea la de vagar por el entorno, sin tener en principio un objetivo claro. De esta forma, el control del robot está estratificado de modo que los niveles inferiores toman el control del robot cuando eventos del entorno así lo dicten. El sistema se puede cortar en cualquiera de los niveles implementados, y los niveles inferiores formarán un sistema de control operativo completo.

Concretamente, la arquitectura *subsumption* [26], concebida por Rod Brooks, utiliza máquinas de estados finitos añadiendo temporizadores (AFSMs)‡ para construir reglas de control simples. Los AFSMs se comunican mediante un protocolo de paso de mensajes y de supresión e inhibición de mensajes. Mediante la supresión un AFSM puede suprimir las entradas que recibe otro AFSM durante un periodo fijo de tiempo, mientras que mediante la inhibición se anulan las salidas. La combinación de diversos AFSMs forman *comportamientos*, que son los constituyentes de la arquitectura subsumption [23], y sirven como niveles de abstracción. Los comportamientos son combinados en niveles de competencia, o capas, que se corresponden con cada una de las habilidades del robot (esquivar obstáculos, vagar por el entorno, seguir paredes, etc.).

Mediante esta técnica de ir añadiendo niveles a los ya pre-existentes se consigue que cada una de las capas pueda estar trabajando en diferentes metas concurrentemente. Por otro lado, se obvia por completo el problema de la fusión sensorial, ya que cada uno de los módulos extraerá de los sensores la información que le sea necesaria para su objetivo particular: No hay ninguna necesidad de generar un modelo del mundo donde se fusione la información proporcionada por los diferentes sub-sistemas sensoriales. La robustez del sistema se garantiza por el desarrollo del diseño del sistema, donde se implementan y depuran los niveles de competencia más bajos antes de que se añada una nueva capa. Finalmente, la arquitectura permite que se vayan añadiendo cada vez más capas de competencia, aumentando por tanto las habilidades y prestaciones del robot.

2.3.1 Implementaciones

Los sistemas basados en arquitecturas reactivas tratan de modelar y construir ingenios que se "comporten" en un determinado entorno muy definido. Por lo general, estos sistemas tienen integrados en su repertorio múltiples competencias de bajo nivel. A su vez, están "situados" en su entorno, es decir, están directamente conectados al dominio problema a través de sus sensores y actuadores. En este tipo de sistemas se pone un especial énfasis en la autonomía del sistema: ha de observar el entorno y decidir por sí mismo cómo se ha de resolver el problema y cómo ha de elegir entre un número elevado de metas simultáneas. Sus estructuras internas son "activas" [74] y generan un "comportamiento" determinado, al contrario que las estructuras estáticas que utilizan los sistemas basados en el conocimiento. En particular, el sistema se considera integrado en su entorno y está situado en un espacio determinado. Esto implica que hay una necesidad menor de tener que modelar el entorno porque el *mundo es su mejor modelo* [25], y se puede utilizar como una memoria externa [29]. Por lo general, el entorno tiene toda una serie de características que le permiten al sistema explotarlas en su propio beneficio a la hora de razonar sobre él [52], como, por ejemplo, suponer que en un laboratorio las paredes son verticales respecto al suelo, o que las puertas tienen unas determinadas dimensiones fijas.

El Laboratorio de Inteligencia Artificial del Massachusetts Institute of Technology (M.I.T.) fue el pionero en el diseño e implementación de robots móviles que utilizaban una arquitectura reactiva como sistema de control. Estos modelos se caracterizaban por ser capaces de realizar

‡Siglas que provienen de **A**ugmented **F**inite **S**tate **M**achines.

toda una serie de tareas de una forma robusta y fiable en tiempo real. A continuación se revisará una serie de implementaciones realizadas en dicho laboratorio en los últimos años, que ejemplifican el uso y las posibilidades de este tipo de arquitecturas.

Allen

Allen [20] fue el primer robot móvil que se diseño y se construyó en el Laboratorio de Inteligencia Artificial del MIT. Poseía sensores de ultrasonido y un sistema de odometría que permitía la localización del robot en el entorno. El control se realizaba desde una máquina LISP que simulaba las características de la arquitectura *subsumption*.

La primera capa de control permitía al robot evitar obstáculos dinámicos y estáticos. En un principio, *Allen* esperaba a que algún tipo de objeto se le aproximase (como, alguna persona) y se alejaba de él evitando los obstáculos que se encontrasen en su camino. La forma de evitar los obstáculos se basaba en considerar fuerzas repulsivas inversamente proporcionales a la distancia que indicasen las lecturas de los sensores de ultrasonido. Mediante la suma vectorial de estas fuerzas repulsivas, el robot sabía a qué dirección dirigirse. Un reflejo adicional permitía al robot parar en cuanto detectase que existía algo justo en frente de él.

La segunda capa de control permitía al robot vagar por el entorno de forma aleatoria. Una tercera capa buscaba lugares distantes a los que el robot se podía dirigir. Esta última capa monitorizaba la trayectoria realizada por el robot mediante el estudio de los datos que provenían de los sensores odométricos, generando una dirección de movimiento que suprimía la salida del comportamiento de vagar por el entorno.

Allen se puede considerar como la primera implementación de un robot en la historia que era capaz de responder a los estímulos (lecturas de los sensores de ultrasonido) de forma “inteligente” en tiempo real.

Herbert

Herbert [29] se puede considerar como el robot reactivo arquetípico. El sistema de control estaba compuesto por un conjunto de 24 procesadores distribuidos que se comunicaban mediante el uso de paso de mensajes. Poseía 30 sensores infrarrojos de proximidad y un sistema láser que permitía estimar la forma tridimensional de los objetos sobre los cuales recaía la luz del láser. *Herbert* estaba programado para vagar por el entorno, entrar en las oficinas del edificio y recoger los botes de refresco que alguien hubiese olvidado sobre su mesa. Entre su repertorio de habilidades se podía contar con comportamientos para evitar obstáculos, seguir paredes, reconocer botes de refresco en tiempo real, más un conjunto de 15 comportamientos que conducían el brazo para recoger un bote que reposase sobre una mesa.

La ventaja de este sistema era la de no necesitar ningún tipo de representación del mundo para planificar acciones posteriores; el sistema era completamente reactivo y podía responder fácilmente a circunstancias cambiantes, como la de evitar un objeto que se acercase peligrosamente.

Toto

Maja J. Matarić [74] demostró con el robot *Toto*, la posibilidad de crear y utilizar un mapa del entorno sin la necesidad de generar una representación simbólica (en el sentido clásico de la palabra). *Toto* contaba con 12 sensores de ultrasonido equiespaciados alrededor de la base circular móvil, y un compás. La arquitectura de control de *Toto* se basaba en:

1. Un sistema de bajo nivel reactivo que le permitía ejecutar las tareas básicas de forma robusta, como vagar por el entorno evitando colisionar contra otros objetos y seguir las paredes de los corredores de forma satisfactoria, mientras exploraba el mundo explícitamente

2. Un nivel intermedio de comportamientos trataba de reconocer determinados tipos de hitos como paredes, corredores o esquinas.

3. En paralelo funcionaba otra red de control compuesta de comportamientos idénticos que esperaban a que un tipo de hito fuese reconocido. Cada vez que se reconocía un elemento, se le asignaba un comportamiento a la posición de esa pared, corredor o esquina. Entre los comportamientos que correspondían a localizaciones vecinas se establecía una relación de vecindad (mediante una conexión entre ellos). Así se podía generar una estructura de control tipo grafo, aunque en este caso los nodos eran elementos de computación activos, al contrario que las estructuras de datos estáticas.

El funcionamiento del sistema de control era el siguiente: Mientras el robot se movía por el entorno, el sistema averiguaba en dónde se encontraba el robot en cada momento, y se activaba el nodo correspondiente a la localización del robot. A su vez, un usuario podía activar un determinado nodo, y *Toto* era capaz de ir desde el lugar en donde se encontraba a la localización indicada por el usuario.

En definitiva, *Toto* demostró que con una arquitectura reactiva era posible realizar tareas que siempre se habían considerado exclusivas de las arquitecturas clásicas, como generar mapas del entorno y utilizarlos para navegar.

Allen, Herbert y *Toto* son los ejemplos mas significativos de arquitecturas reactivas realizadas en el Laboratorio de Inteligencia Artificial del M.I.T. Permiten observar una evolución en las capacidades y habilidades de los robots. Otra serie de robots han seguido este tipo de filosofía, incluyendo robots con patas [22], robots de dimensiones reducidas (un volumen de 5/4 pulgadas cúbicas) [43] o robots guiados por visión artificial [53], entre muchos otros.

2.4 Arquitecturas Híbridas

Las arquitecturas híbridas aparecieron como un compromiso entre las arquitecturas tradicionales y las arquitecturas reactivas. Este tipo de arquitecturas comparten con las arquitecturas tradicionales el concepto fundamental de la necesidad de una planificación en un modelo del mundo generado a partir de la información sensorial. Por otro lado, reconocen la necesidad de que el robot "reaccione" y sea robusto ante cambios en tiempo real en el entorno. Por tanto, este tipo de arquitecturas se suelen componer de un subsistema reactivo que se encarga de la supervivencia de robot en todo momento, y un sistema de planificación de alto nivel.

Uno de los más claros ejemplos se puede encontrar en el sistema *SSS* [30], donde se combinan tres técnicas de control diferentes que se pueden caracterizar por su tratamiento del tiempo y del espacio. Por su parte, *Arkin* [4] propone una metodología a base de lo que él define como *esquemas* (de una supuesta base biológica), como control del sistema. Este tipo de *esquemas* se pueden interpretar como comportamientos que actúan en un modelo del mundo (basado en métodos de potenciales atractivos y repulsivos) generado a partir de los datos de los sensores. En un extremo, se puede encontrar el trabajo de *Payton* [82], donde el sistema realiza una planificación a-priori de todas las posibilidades que pueda ofrecer el entorno.

En este capítulo no insistiremos en las arquitecturas híbridas, ya que al fin y al cabo vienen a unir y utilizar las metodologías anteriormente expuestas, y a la hora de realizar el control del robot no añaden ningún concepto nuevo a los ya estudiados. Sin embargo, se hará un estudio más detallado de las mismas en el capítulo dedicado a la planificación e interfase de comunicación hombre-máquina (véase Capítulo 7).

2.5 Discusión

Aunque está comúnmente aceptado el hecho de que sistemas puramente reactivos son estrictamente menos poderosos que los sistemas que contienen un planificador de tareas, es difícil demostrarlo de forma analítica. La demostración depende enteramente de la comparación de ambas filosofías bajo el marco de una tarea determinada, del entorno y de las capacidades computacionales y sensoriales del robot [73].

En principio, las arquitecturas tradicionales pueden parecer más propicias para realizar un estudio analítico de sus capacidades y su respuesta a diversas configuraciones del entorno. Sin embargo, la existencia de condiciones imprecisas e impredecibles en el entorno, hace que estos sistemas sean tan difíciles de validar como aquellos que se basan en una arquitectura reactiva o de comportamientos.

Por otro lado, un sistema de control basado en una arquitectura reactiva o de comportamientos, permite construir robots móviles cuya respuesta sea en tiempo real, e incorporar nuevas funcionalidades al sistema. Sin embargo, este tipo de arquitecturas presentan una serie de inconvenientes que pasaremos a repasar a continuación:

2.5.1 Arbitrar comportamientos

El gran problema de este tipo de sistemas no se encuentra en el nivel de integración sensorial, sino más bien en el nivel en el que se arbitran los diferentes comportamientos y se selecciona uno de ellos [65]. En efecto, el problema de arbitrar los diversos comportamientos que componen una arquitectura reactiva es uno de los aspectos centrales del diseño de un robot cuya función sea la trabajar de una forma robusta en un entorno complejo y dinámico. Más en concreto, se trata de que el robot elija, de su repertorio de comportamientos, el conjunto más apropiado de acciones motoras para satisfacer las metas que deba alcanzar y responder a estímulos externos. En la mayoría de estos sistemas, la forma de arbitrar los diferentes comportamientos suele ser una precodificación del protocolo que impone una jerarquía en el orden de actuación de los comportamientos. Este protocolo asegura que sólo uno de los comportamientos del repertorio del robot tome el control sobre los actuadores, y que este control sea siempre el apropiado respecto al comportamiento con la mayor prioridad dentro del contexto en el que se encuentre el robot.

Por otro lado, el modo de selección de comportamientos que utiliza una arquitectura *subsumption* clásica, puede producir problemas adicionales: En numerosas ocasiones robots móviles basados en una arquitectura de procesos reactivos se quedan estancados en zonas del espacio conflictivas, debido a que el protocolo de activación/inhibición lleva al robot móvil a situaciones en las que los diferentes agentes toman de forma alternativa el control de los actuadores, deshaciendo un determinado agente la acción previa de otro. En realidad lo que ocurre es que al precodificar el protocolo de activación de los diferentes comportamientos de forma apriorística, se codifica a su vez un cierto conocimiento del entorno. Es decir, a pesar de que la filosofía de este tipo de sistemas se basan en que el robot "reaccione" a lo que está percibiendo en cada instante, en realidad no suele ser así. Por ejemplo [29], el comportamiento de *evitar obstáculos* toma el control de los actuadores del robot cuando percibe la inminencia de una colisión, e inhibe las salidas del resto de los comportamientos durante el periodo de tiempo durante el cual el comportamiento está percibiendo el obstáculo, más otro instante adicional. Durante este periodo de tiempo adicional, el comportamiento de *evitar obstáculos* no está realmente percibiendo la presencia de un obstáculo en los alrededores del robot, pero "presupone" su existencia, y por motivos de seguridad sigue esquivando el obstáculo aunque no lo perciba. Este hecho conduce a que en cuanto otros comportamientos de más alto nivel vuelvan a tomar el control de los actuadores, puedan "deshacer" las acciones del comportamiento de menor nivel. En resumen, no hay nada que impida que las acciones de control de niveles superiores dirijan al robot a la zona de donde provenía, ya que el comportamiento de *evitar obstáculos* no está realmente percibiendo el obstáculo y no tiene necesidad de tomar el control de los actuadores.

2.5.2 Incorporación de nuevos comportamientos

A pesar de que la arquitectura *subsumption* está especialmente pensada y diseñada para incorporar nuevas capas con capacidades "superiores" a las ya existentes, su incorporación presenta numerosos problemas prácticos. A medida que se van incorporando nuevos niveles de competencia, la complejidad del sistema va aumentado. Esto implica que el diseñador deberá tener en cuenta cada vez más módulos (comportamientos) de bajo nivel a la hora de incorporar una nueva capa. Es decir, el diseño de una nueva capa deberá tener en cuenta tanto las acciones de control que realizan los niveles inferiores como las salidas que producen sobre los actuadores. En la práctica no se puede incorporar un nuevo módulo sin conocer perfectamente las especificaciones de los niveles inferiores y de cómo se comporta el sistema en el entorno.

Incluso la utilización de herramientas especialmente diseñadas para este tipo de sistemas, como el *Behavior Language* [23], no reduce la complejidad de la incorporación de nuevos comportamientos. Con este lenguaje de programación, se especifican explícitamente los diversos comportamientos de los que está compuesto el sistema y las conexiones entre unos y otros. La selección de los diferentes comportamientos no se realiza por un sistema central, sino que son los propios comportamientos los que inhiben y suprimen las acciones de control de una forma directa. Esto se traduce en que la codificación de un determinado comportamiento requiere conocer explícitamente los canales de comunicación con los restantes, y los efectos de suprimir e inhibir sus acciones de control. En efecto, las inhibiciones y supresiones se realizan mediante el paso de un mensaje al comportamiento cuyas salidas se quieran inhibir (o cuyas entradas se deseen suprimir). La recepción de dicho mensaje implica la activación de un monoestable incorporado al propio comportamiento, que inhibe o suprime las acciones de control por un tiempo fijo y predeterminado. Por tanto, a efectos prácticos, a la hora de diseñar un nuevo comportamiento, se deberá conocer a la perfección el resto de los comportamientos ya implementados y sus variables internas, ya que el método de arbitrar el control entre ellos depende de los diferentes tiempos de activación de los monoestables que incidirán en el tiempo de inhibición y supresión de las acciones de control.

2.5.3 Especificación de objetivos

De los robots móviles que siguen una filosofía reactiva se ha dicho que son *"insectos que vagan por el entorno de forma desamparada sin realizar una tarea realmente útil"* [80]. Es cierto que en las primeras implementaciones de arquitecturas reactivas, los robots no realizaban tareas que se pudiesen catalogar como "productivas". Su "inteligencia" se concentraba en "sobrevivir" en el entorno, en el sentido de no colisionar contra obstáculos u otros objetos. Sin embargo, robots como *Herbert* o *Toto* demuestran que pueden realizar tareas de alto nivel, como navegar por el entorno creando mapas o recogiendo una serie de botes de refresco. Sí es cierto que estas arquitecturas son cerradas en el sentido de que todas las funciones del robot están, en cierta manera, pre-programadas. Es decir, el diseñador ha codificado y diseñado el sistema de tal forma que se hace difícil a un usuario definir de una manera flexible la serie de tareas que deba realizar el robot. Sobre este asunto se insistirá con mayor concrección en el Capítulo 7.

2.6 Arquitectura Propuesta

Una arquitectura determina cómo organizar la estructura de control de un sistema, e impone una serie de ligaduras a las que que se tiene que atener el diseñador de un sistema para resolver un problema de control. Los objetivos que nos hemos planteado al diseñar la arquitectura de control para el robot móvil son:

1. Respuesta en tiempo real a situaciones cambiantes en el entorno.
2. Posibilidad de incorporar nuevas funcionalidades al sistema, mientras que se mantienen las antiguas.
3. Capacidad de arbitrar los diferentes módulos del sistema, evitando al robot móvil situaciones de estancamiento.

Por tanto, siguiendo las líneas teóricas marcadas por las arquitecturas reactivas y las basadas en comportamientos, se consigue, mediante la división del sistema de control en sub-módulos generadores de comportamiento, una respuesta del sistema en tiempo real a los cambios que se puedan producir en el entorno de trabajo del robot móvil. La idea básica en la que se sustenta dicha filosofía es en la de *"divide y vencerás"*, que implica un corto camino entre los sensores y los actuadores, permitiendo por tanto una reactividad a cambios bruscos en el entorno.

Sin embargo, la aplicación *"dogmática"* [24] de este tipo de filosofía lleva a los inconvenientes ya mencionados anteriormente. Resumiendo, las razones que llevan a este tipo de problemas son:

1. La ausencia total de algún tipo de representación explícita del mundo, que lleva al diseñador a codificar su propio modelo del mundo de una forma implícita en el sistema.
2. El método mediante el cual se arbitran los comportamientos, que a su vez incide en dos puntos muy importantes:
   (a) La facilidad en la incorporación de nuevos módulos de control a la estructura general del sistema.
   (b) El "estancamiento"del robot en determinadas zonas del espacio.

A continuación analizaremos los puntos esenciales en la definición de una arquitectura basada en comportamientos, y las diferentes propuestas que hacemos para la implementación de una arquitectura de control.

2.6.1 Representación del Mundo

Gran parte del fracaso de la *Inteligencia Artificial Clásica* al construir robots móviles, se debe al empeño de los diseñadores en generar modelos simbólicos del mundo en donde un sistema "inteligente" pueda planificar las acciones de control. Por un lado, estos modelos están sujetos a errores de construcción que implican una incertidumbre al realizar una acción de control. Por otro, el generar un modelo simbólico y completo del mundo retarda el tiempo de respuesta del módulo de control, dándose el caso de que cuando se realiza la acción de control, las condiciones iniciales sobre las que se sustentaba el razonamiento de control ya no se dan más.

Los robots móviles actuales no se pueden comparar en cuanto a densidad de sensores con los organismos vivos. Los sistemas sensoriales con los que van equipados los robots no pueden percibir el entorno en un 100%. Como ejemplo, se puede tomar el robot móvil con el cual se ha desarrollado la parte experimental de esta tesis: Centrándonos en el sistema de ultrasonidos, el robot posee un anillo de 12 sensores, con los que se puede percibir gran parte del entorno, pero no en su totalidad. Por otra parte, los sensores de ultrasonido tienen una gran cantidad de fuentes de incertidumbre y cometen numerosos errores (ver en el Capítulo 3 una relación de los mismos). Por ejemplo, puede darse el caso de que un sensor esté captando la presencia de un determinado objeto durante una serie de muestras consecutivas, pero que cuando el robot se acerque a este objeto el sensor ya no lo perciba. Una arquitectura puramente reactiva sólo tendría en cuenta la última medida de los sensores, no tendría una memoria previa de lo percibido, con lo que el riesgo de colisionar con el objeto en cuestión sería grande. Por tanto, si se mantuviese una **pequeña representación local y temporal del entorno** se podrían tener en cuenta situaciones en las cuales, debido a la idiosincrasia del sistema sensorial, no se puedan percibir elementos previamente detectados.

2.6.2 Arbitrar módulos

Es importante, a la hora de arbitrar las acciones de control de los diversos módulos del sistema, que la acción de un determinado módulo no deshaga las acciones anteriores. Como ya se ha mencionado en párrafos anteriores, en el caso de las arquitecturas puramente reactivas este tipo de situaciones son intrínsecas a la ausencia de una memoria explícita de lo percibido y al método de activación de los comportamientos del sistema. Por tanto, es necesario que una arquitectura de control no "presuponga" la existencia de una determinada entrada sensorial cuando no se percibe realmente. El sistema debe reaccionar a lo que **percibe** y a lo que ha **percibido** en instantes anteriores al actual.

Por otro lado, es sumamente importante que un comportamiento del sistema no trate de realizar una acción de control que sólo sea función de lo que está percibiendo en un instante determinado, sino que también tenga en cuenta las acciones de otros comportamientos a la hora de calcular su acción de control. Mediante la observación de las acciones de control de otros comportamientos, se puede evitar que el robot se quede en situaciones de estancamiento. Por tanto, la acción de control de un comportamiento deberá ser función tanto de las acciones de control que deseen ejecutar comportamientos de niveles superiores, como del estado del entorno.

2.7 MAgDA

La arquitectura de control que proponemos para los Agentes Motores de Navegación, definida como **MAgDA**[§], es una arquitectura multi-agente con las siguientes propiedades:

- Arquitectura jerárquica y modular.
- Respuesta en tiempo real.
- Facilita la incorporación de nuevos agentes a la estructura del sistema.
- Adapta las acciones de control de los agentes de bajo nivel a los *deseos* de control de agentes superiores, *modulándolos* y tratando de ajustarse los más posible a ellos.
- Incorpora una memoria/representación externa de un entorno *local* al robot, que permite:
  - Un control más fiable.
  - La utilización de un conocimiento *explícito* del entorno.

En los siguientes apartados se definen el concepto de agente, la estructura general de la arquitectura **MAgDA** y las diversas reglas de implementación que se han de seguir a la hora de construir un sistema de control para realizar tareas de navegación básicas de un robot móvil.

2.7.1 Agente

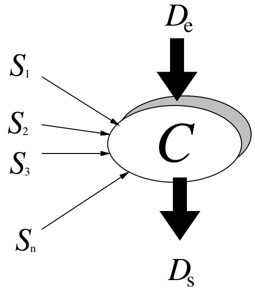


Figura 2.3: Definición de agente.

Un agente es un sistema que trata de llevar a cabo una serie de objetivos en un entorno complejo y dinámico. Un agente está situado en su entorno; puede muestrear el entorno con sus sensores y modificarlo con sus actuadores. Los objetivos de un agente pueden tomar diferentes formas: Pueden ser "objetivos de meta" o estados particulares que el propio agente trata de lograr. Un

[§]Modulated **A**gent **D**esire **A**rchitecture agente se define como autónomo si opera de forma completamente autónoma, es decir, si decide por sí mismo cómo correlacionar los datos recibidos por sus sensores con las órdenes que le manda a sus actuadores, de forma que los objetivos se cumplen satisfactoriamente [67].

En la arquitectura empleada en el presente trabajo, un agente se define como un ente que recibe una serie de entradas sensoriales $\{S_1, S_2, \ldots, S_n\}$, un deseo de control de entrada D_e y genera una acción de control de salida, D_s. A su vez, el agente se puede considerar como una abstracción de alto nivel a la que se le asigna un *comportamiento* determinado, como pueda ser el de esquivar obstáculos, seguir paredes, etc.

En general, la acción de control que generará un determinado agente será dependiente de la señal sensorial que reciba y del deseo de control de entrada:

$$D_s = f(D_e, \{S_1, S_2, \ldots, S_n\})$$

Sin embargo, el agente puede estar en dos estados diferentes: activo o inactivo. Cuando el agente está *activo*, su salida será función del deseo de entrada y de la señal sensorial. Cuando el agente está *inactivo*, su salida será exactamente el deseo de control de entrada.

$$\begin{array}{ll} D_s = f(D_e, \{S_1, S_2, \ldots, S_n\}), & \text{activo;} \\ D_s = D_e, & \text{inactivo;} \end{array}$$

2.7.2 Arquitectura Multi-agente

Puesto que **MAgDA** se caracteriza por ser una arquitectura jerárquica multi-agente, formalmente se puede expresar como un conjunto ordenado de agentes:

$$\{C_1, C_2, \ldots, C_n\}$$

Cada uno de estos agentes vendrá caracterizado por aceptar un deseo de control de entrada D_e^i y generar un deseo de control de salida D_s^i.

El orden de un agente en este conjunto establece su jerarquía respecto al resto. Un agente de más bajo nivel modulará los deseos del agente que le anteceda en la jerarquía de control. Por tanto, un agente C_i tendrá mayor prioridad, $P(C_i)$, que un agente C_j siempre que $i < j$. Es decir, la jerarquía de prioridades de los agentes que componen el sistema vendrá dada por:

$$P(C_1) > P(C_2) > \ldots > P(C_n)$$

La arquitectura tendrá por tanto una configuración lineal, conectándose los agentes uno a uno; es decir, la salida de control de un agente será el deseo de entrada del agente de nivel inferior (véase Figura 2.4). El caso en el que dos o más agentes se encuentren en un mismo nivel dentro de la arquitectura se estudiará en detalle en el apartado referido a las múltiples entradas.

2.7.3 Composición de Agentes

Para implementar nuevos agentes e integrarlos a la estructura general de la arquitectura de control **MAgDA**, se debe satisfacer una serie de reglas y requerimientos, relativos a la entrada y salida de control, o a la composición funcional de los agentes. En los siguientes sub-apartados se definirán y explicarán las reglas de implementación.

Entrada Salida

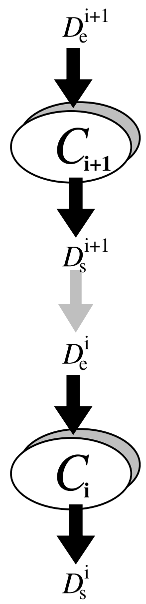


Figura 2.4: Composición de agentes.

La entrada/salida de un agente debe ser "compatible" con la entrada/salida de los agentes con los que esté conectado.

Así, si un nuevo agente C_i tiene como deseos de entrada y salida a D_e^i y D_s^i respectivamente, el formato del deseo de entrada tiene que ser compatible con el formato del deseo de salida D_s^{i+1} del agente de más alto nivel C_{i+1} (ver Figura 2.4). Del mismo modo, el formato del deseo de control de salida D_s^i deberá ser compatible con el del deseo de entrada D_e^{i-1} del agente C_{i-1}

En cuanto a la implementación del nuevo agente, el diseñador deberá concentrarse únicamente en los detalles del comportamiento que desea implementar y del interfaz de entrada y salida. Por ejemplo, si el diseñador desea implementar un comportamiento de *ir a un punto*, deberá concentrarse en lo que concierne a una trayectoria que le acerque a dicho punto y en que el deseo de control de salida sea compatible con el deseo de entrada del agente de más bajo nivel (como, por ejemplo, del comportamiento que trata de evitar obstáculos). El diseñador no tendrá que preocuparse de evitar los obstáculos que el robot se pueda encontrar en su entorno a la hora de implementar el agente de ir a *un punto*, ya que el comportamiento de *evitar obstáculos* recibirá la acción de control de este nuevo agente y modulará la salida dependiendo de la relación del robot con el entorno.

Múltiples Entradas

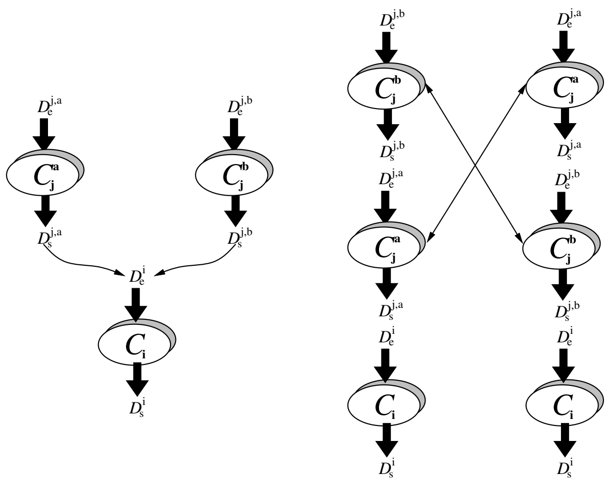


Figura 2.5: Agente con múltiples entradas.

Desde un punto de vista semántico, puede ocurrir que dos agentes se encuentren en un mismo nivel en la jerarquía de la estructura de control. Por ejemplo, un agente cuya misión fuese la de *seguir paredes* estaría, en principio, en un mismo nivel en la jerarquía de procesos que el agente de ir *a un punto*. A priori, no existe ninguna razón por la cual uno de estos dos comportamientos deba tener prioridad sobre el otro. Es decir, sin conocer la tarea que realizar por el robot, ni el estado del entorno, es imposible decidir cuál de los dos agentes ha de tomar el control. Ambos comportamientos son excluyentes, pues no tiene "mucho sentido" dirigirse a un determinado punto del espacio mientras se están bordeando las paredes. Por tanto, tanto la situación del entorno como la tarea que realizar serán los que determinen la prioridad de uno u otro agente. En este caso, el dar la prioridad no significa que uno de los agente module las acciones de control del otro, sino que uno de los dos agentes no estará activo cuando el otro lo esté. Por tanto, no es importante asignar una prioridad determinada a cada uno de estos agentes, al no poder estar activos ambos al mismo tiempo.

A efectos de mayor claridad de exposición, se puede reformular la definición de arquitectura multi-agente como el conjunto ordenado de agentes:

$$\{C_1^1 \dots C_1^i; C_2^1 \dots C_2^j; \dots; C_n^1 \dots C_n^k\}$$

donde para cada uno de los agentes C_m^n, m indica el nivel en la jerarquía general, y n actúa como un factor de distinción entre los agentes de un mismo nivel semántico.

En principio, la jerarquía de prioridades de los agentes vendrá dada por:

$$P(C_1^1) = \ldots = P(C_1^i) > P(C_2^1) = \ldots = P(C_2^j) > \ldots > P(C_n^1) = \ldots P(C_n^k)$$

Formalmente, se dice que los comportamientos C_i^a, C_i^b $\ldots C_i^z$ son **incompatibles** desde el punto de vista de la jerarquía de control, cuando no existe una referencia a-priori clara de cuál de los dos es de nivel superior, y se designará tal relación como:

$$C_i^a \odot C_i^b \odot \ldots \odot C_i^z$$

Al estar solamente activo uno de los agentes de un mismo nivel de jerarquía, es indiferente la relación de orden que se les dé en la jerarquía. Por tanto, concluimos que si dos comportamientos son **incompatibles**:

$$C_i^a \odot C_i^b$$

su prioridad en la jerarquía de control será indiferente:

$$P(C_i^a) > P(C_i^b) \rightleftharpoons P(C_i^a) < P(C_i^b)$$

Por último, si en la arquitectura del sistema hay dos agentes en el mismo nivel de jerarquía, C_j^a, C_j^b, podrá existir un comportamiento C_i, tal que $i > j$. Por tanto, está claro que en cualquiera de los casos, el agente C_i modulará las salidas tanto de C_j^a como las de C_j^b.

Composición de Funciones

Al estar constituida la arquitectura de control del sistema por un conjunto jerárquico de agentes, $\{C_1, C_2, \ldots, C_n\}$, donde el deseo de salida de un agente es el deseo de entrada del agente que le sigue en la jerarquía, el comportamiento global del sistema se puede considerar como una composición de las funciones que definen a cada uno de los comportamientos.

Efectivamente, si consideramos a cada uno de los agentes del sistema como una función que depende de su entrada sensorial y del deseo de control de entrada, se puede expresar el deseo de salida de un agente C_i como:

$$D_s^i = C_i(D_e^i) \equiv f(D_e^i, \{S_1, S_2, \ldots, S_n\}) \tag{2.1}$$

Y teniendo en cuenta que:

$$D_e^i \equiv D_s^{i+1}$$

La salida final del sistema se puede expresar como:

$$D_s^1 = C_1(C_2(\ldots(C_n(D_e^n)))) \tag{2.2}$$

Este tipo de notación será especialmente útil a la hora de describir el desarrollo experimental y las características de implementación de los diferentes agentes motores de navegación, tal y como se verá en el Capítulo 5.

Capítulo 3

Generación del Mapa

3.1 Introducción

Tanto en la actualidad como en el pasado, numerosos robots móviles [39] [11] [13] [12] [38] [18] han utilizado sensores de ultrasonidos para realizar tareas de navegación por el entorno como, por ejemplo, evitar obstáculos, seguir paredes, atravesar puertas, etc. Las razones de su utilización son numerosas y entre ellas cabe destacar su bajo coste, velocidad de procesamiento y una más que aceptable precisión. Sin embargo, la navegación de un robot móvil basada en la información suministrada por un conjunto de sensores de ultrasonidos presenta una serie de inconvenientes debidos, sobre todo, a los numerosos errores que cometen estos sensores en sus medidas.

En efecto, una única medida proporciona poca información acerca del entorno, y mediante la acumulación de la evidencia de una serie de medidas, se incrementa la precisión de la localización de los diversos obstáculos que hay en el entorno. Por tanto, una forma de reducir la incertidumbre intrínseca a los sensores de ultrasonido es la utilización de mapas del entorno donde sucesivamente se vayan plasmando y actualizando las medidas realizadas. Por otro lado, al moverse el robot por el entorno, la incertidumbre en la generación del mapa y su actualización se va acrecentando con el tiempo, debido a que el error en la estimación de la posición del robot respecto a un sistema de referencia general es acumulativo y va creciendo con el tiempo. Por tanto, se debe contar con un procedimiento que permita reducir la incertidumbre de la posición del robot para mantener la coherencia espacial de las diferentes medidas. En definitiva, en la generación y actualización de un mapa basado en la información suministrada por los sensores de ultrasonido embarcados en un robot móvil podemos hablar de dos fuentes de incertidumbre [61]:

1. Incertidumbre determinación de la posición de robot, e
2. incertidumbre en las propias medidas.

En el presente capítulo haremos en primer lugar una breve descripción de los diferentes errores asociados a los sensores de ultrasonido, pasando a continuación a realizar un estudio más detallado de la incertidumbre asociada a la estimación de la posición del robot. En el apartado siguiente se hará un resumen de las diferentes técnicas utilizadas hasta hoy a la hora de minimizar la incertidumbre de estas dos fuentes. Finalmente, se abordará la descripción del sistema desarrollado incidiendo en el concepto definido en este trabajo como *decremento temporal* en la certeza de la presencia de obstáculo. Demostraremos cómo con este sencillo principio se puede abordar de una manera fiable las dos fuentes de incertidumbre y se repasarán las diferentes ventajas que se derivan de aplicar este principio a la generación y mantenimiento del mapa de ultrasonidos.

3.2 Sensores de Ultrasonido

La mayoría de los sensores de ultrasonido de bajo coste se basan en la emisión de un pulso de ultrasonido cuyo lóbulo, o campo de acción, es de forma cónica. Midiendo el tiempo que transcurre entre la emisión del sonido y la percepción del eco se puede establecer la distancia a la que se encuentra el obstáculo que ha producido la reflexión de la onda sonora, mediante la fórmula:

$$d = \frac{1}{2}Vt \tag{3.1}$$

donde V es la velocidad del sonido en el aire y t es el tiempo transcurrido entre la emisión y recepción del pulso. Sin embargo, factores inherentes tanto a los ultrasonidos como al mundo real, influyen de una forma determinante en las medidas realizadas. Por tanto, es necesario un conocimiento de las diversas fuentes de incertidumbre que afectan a las medidas para poder tratarlas de forma adecuada, minimizando su efecto en el conocimiento del entorno que se desea adquirir. Entre los diversos factores que alteran las lecturas que se realizan con los sensores de ultrasonido cabe destacar:

1. El campo de actuación del pulso que se emite desde un transductor de ultrasonido tiene forma cónica. El eco que se recibe como respuesta a la reflexión del sonido indica la presencia del objeto más cercano que se encuentra dentro del cono acústico y no especifica en ningún momento la localización angular del mismo. Aunque la máxima probabilidad es que el objeto detectado esté sobre el eje central del cono acústico, la probabilidad de

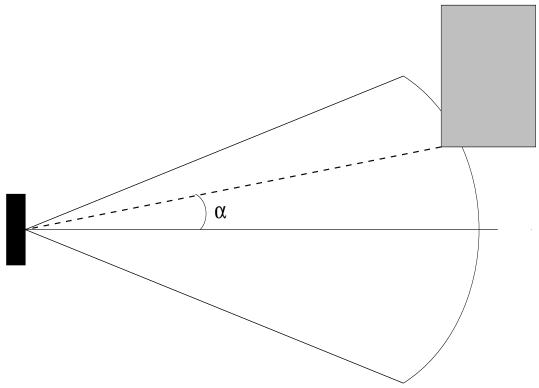


Figura 3.1: Incertidumbre angular en la medida de un ultrasonido

que el eco se haya producido por un objeto presente en la periferia del eje central no es en absoluto despreciable y ha de ser tenida en cuenta y tratada convenientemente.

2. La cantidad de energía acústica reflejada por el obstáculo depende en gran medida de la estructura de su superficie. Para obtener una reflexión altamente difusa del obstáculo, el tamaño de las irregularidades sobre la superficie reflectora debe ser comparable a la longitud de onda de la onda de ultrasonido incidente [27].

3. En los sensores de ultrasonido de bajo coste se utiliza el mismo transductor como emisor y receptor. Tras la emisión del ultrasonido se espera un determinado tiempo a que las vibraciones en el sensor desaparezcan y esté preparado para recibir el eco producido por el obstáculo. Esto implica que existe una distancia mínima (d) (proporcional al tiempo de relajación del transductor) a partir de la cual el sensor mide con precisión. Por lo general, todos los objetos que se encuentren por debajo de esta distancia, d, serán interpretados por el sistema como que están a una distancia igual a la distancia mínima.

4. Los factores ambientales tiene una gran repercusión sobre las medidas: Las ondas de ultrasonido se mueven por un medio material que es el aire. La densidad del aire depende de la temperatura, influyendo este factor sobre la velocidad de propagación de la onda según la expresión:
$$V_s = V_{so}\sqrt{1 + \frac{T}{273}} \tag{3.2}$$
siendo V_{so} la velocidad de propagación de la onda sonora a 0^o C, y T la temperatura absoluta (grados Kelvin).

   Por otro lado, los sensores de ultrasonido móviles (como, por ejemplo, los que vayan embarcados en robot móvil) experimentarán un efecto perturbador debido a las pequeñas turbulencias de aire que se producen delante del transductor. En este caso, y al contrario del efecto de la temperatura, la influencia de las turbulencias sobre la señal ultrasónica es muy difícil de ser cuantificada.

5. Un factor de error muy común es el conocido como *falsos ecos*. Estos falsos ecos se pueden producir por razones diferentes: Puede darse el caso en que la onda emitida por el

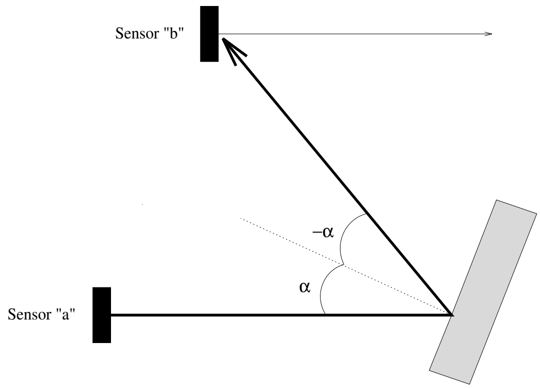


Figura 3.2: Crosstalk: El sensor "a" emite el pulso que recibe el sensor "b"

transductor se refleje varias veces en diversas superficies antes de que vuelva a incidir en el transductor (si es que incide). Este fenómeno, conocido como reflexiones múltiples, implica que la lectura del sensor evidencia la presencia de un obstáculo a una distancia proporcional al tiempo transcurrido en el viaje de la onda; es decir, una distancia mucho mayor que a la que está en realidad el obstáculo más cercano, que pudo producir la primera reflexión de la onda. Otra fuente más común de *falsos ecos*, conocida como *crosstalk*, se produce cuando se emplea un cinturón de ultrasonidos donde una serie de sensores están trabajando al mismo tiempo. En este caso puede ocurrir (y ocurre con una frecuencia relativamente alta) que un sensor emita un pulso y sea recibido por otro sensor que estuviese esperando el eco del pulso que él había enviado con anterioridad (o viceversa).

6. Las ondas de ultrasonido obedecen a las leyes de reflexión de las ondas, por lo que una onda de ultrasonido tiene el mismo ángulo de incidencia y reflexión respecto a la normal a la superficie. Esto implica que si la orientación relativa de la superficie reflectora con respecto al eje del sensor de ultrasonido es mayor que un cierto umbral, el sensor nunca reciba el pulso de sonido que emitió [9].

7. Para emitir un pulso de ultrasonido hay que excitar la membrana del transductor con una señal en forma de delta de Dirac. Sin embargo, en los sensores de bajo coste, la señal excitadora es en la práctica un pulso cuadrado por lo que el efecto resultante es el de la emisión de todo un tren de ondas de ultrasonidos que emergen de la membrana del sensor. El momento t_i en el que se emite la onda se suele considerar como el momento en el que se emitió la onda excitadora. La dificultad estriba en determinar el momento t_f en que se recibe el eco de la señal emitida: Tanto la forma la envolvente de tren de ondas de la señal emitida como la recibida, crece desde cero hasta alcanzar un valor máximo y vuelve a decrecer hasta cero. En definitiva, la incertidumbre en la determinación del tiempo transcurrido entre la emisión y la recepción se traduce en un error $\pm\epsilon$ en la medida.

A la vista de la relación de los diferentes factores que provocan errores o incertidumbre en la medida de un sensor de ultrasonido, se puede realizar una clasificación de los mismos en errores o incertidumbres intratables, tratables directamente y tratables indirectamente. Como errores intratables podemos considerar la no reflexión de la señal por superficies poco rugosas. Afortunadamente las superficies lisas que se encuentran en entornos reales suelen tener alguna que otra pequeña irregularidad que en un momento dado hacen que la señal se refleje en la superficie. Como se verá más adelante, mediante la generación de un mapa del entorno este problema se podría resolver parcialmente. Otras fuentes de incertidumbre en la lectura de la medida realizada, como el tiempo de relajamiento o la resolución del sensor, se pueden tratar directamente modificando y mejorando la tecnología del transductor. En el próximo apartado se demostrará cómo mediante la generación y utilización de un mapa del entorno, a partir de la información de distancia proporcionada por los sensores de ultrasonido, se pueden tratar errores e incertidumbres debidos a la localización angular del objeto detectado, el *crosstalk*, y la percepción de falsos ecos.

3.3 Mapas de Acumulación

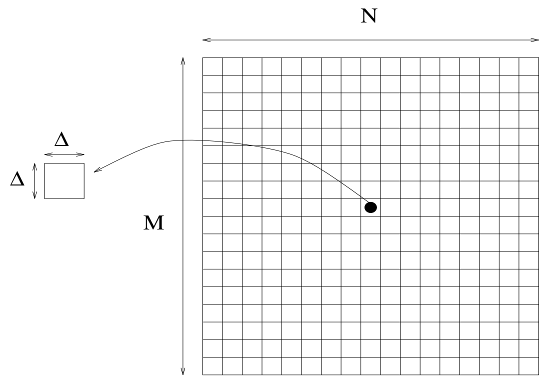


Figura 3.3: Mapa de Acumulación

A la vista de los problemas asociados a los sensores de ultrasonido, podemos asegurar que la navegación reactiva [20] de un robot móvil basada en las informaciones que proveen los sensores en un único instante de tiempo es altamente peligroso, pues zonas consideradas como ocupadas hasta ese momento pueden *aparecer* como libres para los ultrasonidos, con el consiguiente riesgo de colisión. Sin embargo, si se combina la evidencia de muchas lecturas hechas mientras el robot se mueve por el entorno, y se genera un modelo del mundo con la información adquirida hasta ese momento, la incertidumbre en las medidas y en el conocimiento de las zonas libres y ocupadas se reduce drásticamente.

Numerosos sistemas de navegación de robots móviles [38] [78] [39] generan un modelo del mundo basado en el concepto de Mapas de Acumulación (*certainty grid*) (Figura 3.3) elaborado por Moraveck [78]. Este sistema, que se ha demostrado como un método eficiente para representar el espacio ocupado y vacío, se basa en la construcción de un mapa cartesiano bidimensional dividido en un conjunto de MxN celdas cuadradas de tamaño ΔxΔ. Cada celda tiene asociado un valor que indica el grado de certeza de la presencia de un obstáculo en esa localización.

Una medida de distancia se actualiza sobre el mapa aumentando la certeza de presencia de un objeto en el área que encierra la celda. Los diferentes sistemas realizados hasta ahora se basan en que el efecto global sobre el mapa, según se van añadiendo nuevas medidas, es un aumento gradual en la precisión de la localización de los diferentes objetos que rodean al robot: Medidas sucesivas de una misma posición del espacio se reflejan en un aumento de la certeza de presencia de obstáculo de la celda que se corresponde con esa posición. Por el contrario, las celdas que hayan sido actualizadas con un medida falsa tendrán un valor bajo, ya que la probabilidad de que se produzca una serie de medidas falsas en la misma posición del espacio es, en principio, muy baja.

En cuanto a la ley de actualización del valor de las celdas, se pueden diferenciar de forma muy clara dos tendencias:

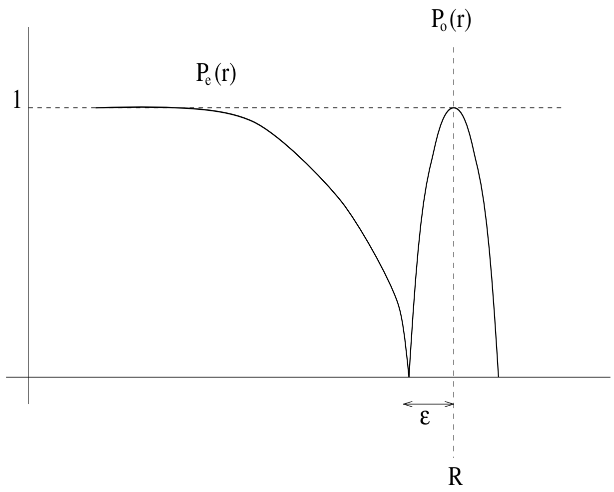


Figura 3.4: Relación entre la distancia de la medida y la distribución de probabilidad.

1. **Actualización Probabilística** (Elfes [38]): Cada medida realizada por un sensor provoca una variación de la probabilidad de presencia o ausencia de obstáculo de un conjunto de celdas del mapa. Por cada medida que se realiza, se calculan dos funciones de probabilidad diferentes (véase la Figura 3.4):

   - La probabilidad de que las celdas que se encuentran en el eje del sensor entre la posición del sensor y la de la medida realizada (R) estén vacías: $P_e(r)$. Esta función de probabilidad toma su valor máximo (1) en la celda en la que está el sensor, y disminiye gradualmente hasta llegar a cero en la celda indicada por la medida.
   - La "probabilidad de que las celdas estén ocupadas" ($P_o(r)$). Esta función tiene su valor máximo (1) en la celda indicada por la medida (R) y disminuye gradualmente hasta 0, para las celdas situadas en el intervalo $[R - \epsilon, R + \epsilon]$, siendo ϵ la resolución del sensor.

   Por otra parte, la región que ocupa el cono acústico a la distancia de medida (R) también se actualiza en el mapa de probabilidades siguiendo la ley expresada por la curva de la

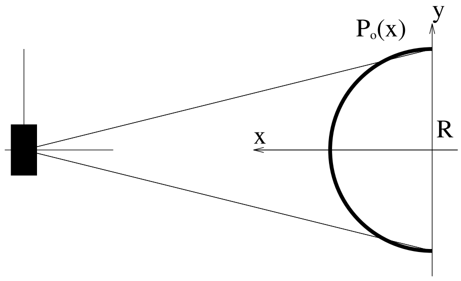


Figura 3.5: Distribución de probabilidad en ls región que ocupa el cono acústico de la medida.

Figura 3.5; las celdas más cercanas al eje del cono acústico tendrán una mayor probabilidad de estar ocupadas por un objeto que las que están a los extremos.

2. **Histogram Grid** (Kohren, Bohrenstein [11] [13] [12]): Cada celda en el mapa contiene un valor de certeza (certainty value) que representa la confianza que el algoritmo de actualización tiene sobre la existencia de un obstáculo en esa localización. En este método y mediante el uso del movimiento del robot y a la altísima frecuencia de muestreo, solamente se actualiza una única celda del mapa: aquella que se encuentra sobre el eje del sensor a la distancia R medida. De esta forma se obtiene una distribución de probabilidad (likelihood) mediante una rápida y continua medida de los ultrasonidos mientras el robot se está moviendo.

A pesar de las evidentes diferencias en la elaboración y actualización de los mapas basados en medidas realizadas con sensores de ultrasonidos que se han expuesto, se pueden apreciar una serie de problemas común a ambos.

3.3.1 Distinción entre Medidas Falsas y Medidas Recientes

Si no se tiene en cuenta el instante en que se realizan las medidas, es posible confundir celdas ocupadas con celdas libres o viceversa. En efecto, si la probabilidad o grado de certeza de que una celda esté ocupada es baja, puede ser debido a una medida falsa o a una medida reciente; en este último caso, la acumulación de evidencias no ha sobrepasado cierto umbral porque no ha habido tiempo de realizar suficientes medidas.

Como se verá con mayor detalle en el apartado 3.4, trabajar con el conocimiento del instante en el que se actualizó una celda por última vez, aumenta el grado de confianza y de certidumbre en el conocimiento de si una zona está ocupada o no.

3.3.2 Incertidumbre en la Posición del Robot

Tal y como se ha detallado en los apartados anteriores, la acumulación de las medidas realizadas por los sensores de ultrasonido aumenta, de forma considerable, la certidumbre de la distribución del espacio ocupado y del espacio libre del entorno del robot. Sin embargo, existe una dificultad a la hora de mantener la relación espacial entre las diferentes medidas realizadas y actualizadas sobre el mapa, que estriba en la forma en la que se lleva a cabo la actualización de la posición del robot sobre el mapa.

La posición del robot móvil debe estar referida a un sistema de coordenadas fijo mediante un vector de posición $\vec{d_i}$

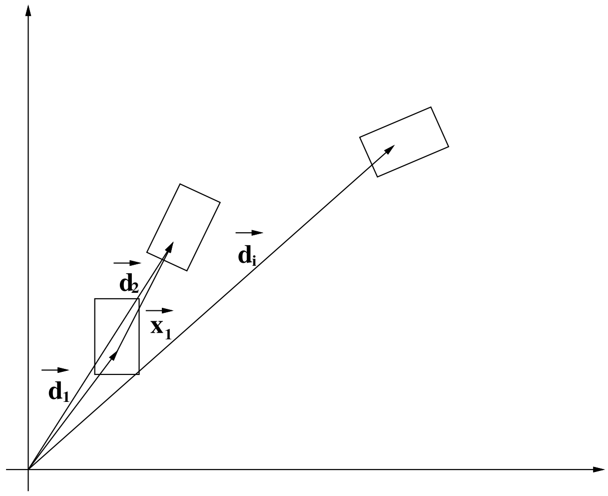


Figura 3.6: Posición del robot móvil en coordenadas absolutas.

Una de las formas más comunes, y utilizada en el presente trabajo, de actualización del vector de posición se basa en las medidas de los sensores odométricos del robot móvil. Mediante los datos que proveen los sensores odométricos, se puede establecer el cambio de posición relativa del robot que se ha producido en un lapso de tiempo determinado, expresado por el vector de desplazamiento $\vec{x_i}$ (ver Figura 3.6):

$$\vec{d_i} = \vec{d_1} + \sum_{j=1}^{j=i-1} \vec{x_j} \tag{3.3}$$

Sin embargo, en la medida de $\vec{x_i}$ siempre hay un error $\vec{\epsilon_i}$ difícilmente cuantificable, debido a factores como, por ejemplo, la inercia del coche, deslizamientos o la existencia de suelos no perfectamente planos. Si se expresa el vector de desplazamiento $\vec{x_i}$ como una suma de dos vectores:

$$\vec{x_i} = \vec{x_{mi}} + \vec{\epsilon_i} \tag{3.4}$$

donde $\vec{x_{mi}}$ indica el desplazamiento real que ha realizado el robot móvil y $\vec{\epsilon_i}$ como el error asociado a esa medida.

La posición actual del robot respecto al sistema de referencia general resulta:

$$\vec{d_i} = \vec{d_1} + \sum_{j=1}^{j=i-1} \vec{x_{mj}} + \sum_{j=1}^{j=i-1} \vec{\epsilon_j} \tag{3.5}$$

Es decir, la posición del robot en cada instante viene fijada por una primera estimación de la posición del mismo ($\vec{d_1}$), más el término:

$$\sum_{j=1}^{j=i-1} \vec{x_{mi}} \tag{3.6}$$

que indican el desplazamiento real del robot, más el sumatorio

$$\sum_{j=1}^{j=i-1} \vec{\epsilon_i} \tag{3.7}$$

que expresa el *error total* cometido en la estimación de la posición actual . Este error, como se puede observar en la expresión, es acumulativo y va creciendo con el tiempo, con lo que la posición del robot respecto al eje de referencia general es cada vez más imprecisa. Esta incertidumbre en el conocimiento de la posición del robot afecta a la actualización de las medidas de los sensores de ultrasonido de la siguiente manera:

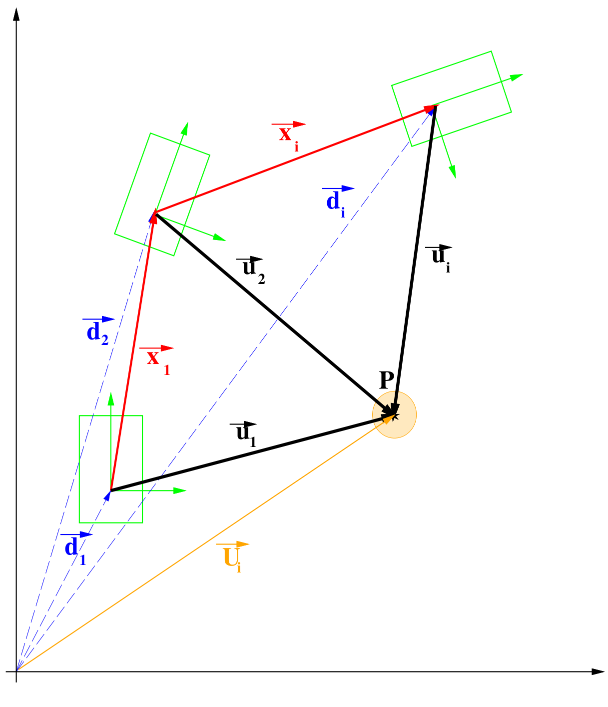


Figura 3.7: Error en la estimación de la posición de la medida debido a los sensores odométricos

Consideremos el caso de un robot en movimiento con un sensor de ultrasonido que está midiendo una serie de distancias, $\{\vec{u_1}, \vec{u_2}, \ldots, \vec{u_i}\}$, en instantes sucesivos a un punto fijo del entorno, P (ver Figura 3.7). Si en un instante t_i se expresa el punto (P) respecto al sistema de referencia general mediante los vectores que definen la posición del robot $\vec{d_i}$ y el correspondiente a la medida realizada $\vec{u_i}$, tenemos:

$$\vec{U_i} = \vec{d_i} + \vec{u_i} \tag{3.8}$$

y sustituyendo la expresión (3.5) en (3.8):

$$\vec{U_i} = \vec{d_1} + \sum_{j=1}^{j=i-1} \vec{x_{mj}} + \sum_{j=1}^{j=i-1} \vec{\epsilon_i} + \vec{u_i} \tag{3.9}$$

Se observa que la estimación de la posición de P está afectada por el error acumulativo en el tiempo: $\sum_{j=1}^{j=i-1} \vec{\epsilon_i}$. Por consiguiente, cuanto más alejadas en el tiempo sean las medidas, más imprecisa será la relación espacial entre ellas. Uno de los efectos que se pueden producir sobre el mapa que se está generando es un "emborronamiento" de los diferentes elementos detectados; en efecto, si tras una serie de medidas de un mismo punto no se obtiene la misma posición respecto al origen de referencia general, se produce un aumento en la superficie ocupada del mapa. Por tanto, zonas que deberían ser consideradas como libres serían consideradas erróneamente como ocupadas. En la literatura se puede encontrar numerosas referencias a este problema y sus posibles soluciones: La utilización de filtros de Kalman [5], o elementos fijos exteriores [61] [36], cuya localización respecto al sistema de referencia general sea conocida de forma precisa, sirven como elementos de reubicación del robot.

3.4 Decremento Temporal de la Certeza de Presencia de Obstáculo

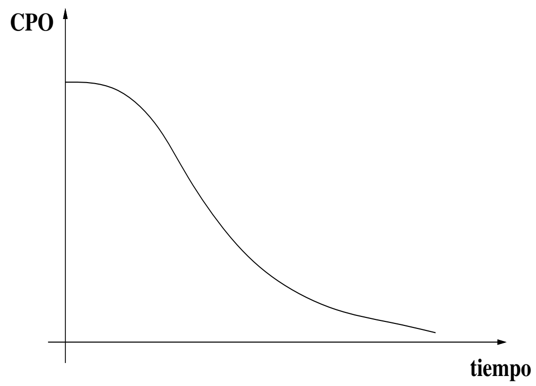


Figura 3.8: Evolución de la **CPO** de una celda durante el tiempo transcurrido entre dos medidas sucesivas que afectan a dicha celda.

Basándonos en las observaciones expuestas en los anteriores apartados, podemos introducir el concepto de *decremento temporal* del factor de certeza de presencia de obstáculo. Cada celda del espacio tiene dos valores asignados:

1. La certeza de presencia de obstáculo (**CPO**).

2. El último instante en que fué actualizada la **CPO** de la celda, que definiremos como instante de adquisición (T_a).

Así, si en un instante $t = t_i$, una medida realizada por un sensor de ultrasonido recae sobre la zona ocupada por la celda, se incrementará el valor de la certeza de presencia de obstáculo y se actualizará el instante de adquisición: $T_a = t_i$. A partir de ese instante, el valor de la certeza de presencia de obstáculo de la celda irá decrementándose con el paso del tiempo.

Con la forma de proceder propuesta, se consiguen los efectos siguientes:

1. **Disminución de la incertidumbre de ocupación de celdas con CPO baja:** Una celda siempre se considera ocupada si su **CPO** es mayor que cero. Si esa celda ha sido afectada por una medida falsa, con el paso del tiempo el valor de la certeza de presencia de obstáculo ira decrementándose hasta llegar a cero (ausencia de obstáculo). Sin embargo, si una celda ha sido actualizada por un obstáculo presente en esa zona, las sucesivas medidas que se realicen sobre esa zona aumentarán la **CPO**.

   En definitiva, se elimina la incertidumbre de si una celda con una baja **CPO** está realmente ocupada o no: Solo el **tiempo** puede determinar si una medida con una baja tasa de certeza de presencia de obstáculo es una medida falsa o reciente.

2. **Disminución de la incertidumbre de la relación espacial entre medidas:** Como ya se ha detallado en parrafos anteriores, la relación en el espacio entre las diferentes medidas realizadas mientras el robot móvil se mueve por el entorno, va degradándose con el tiempo. Es decir, el error en la estimación de la posición entre la serie de medidas, $\{u_{ti}, u_{ti+1}, \ldots, u_{ti+\Delta t}\}$, que han sido realizadas en un intervalo de tiempo $[t_i, t_{i+\Delta t}]$, están afectadas por un error acumulativo en la estimación de la posición del robot. Cuanto mayor sea el periodo de tiempo Δt considerado en la toma de medidas, peor será la relación espacial entre ellas.

   Sin embargo, si desde la última actualización la **CPO** de una celda comienza a decrecer, las celdas que han sido actualizadas en un periodo de tiempo “lejano” y su posición relativa, debido a los errores en la estimación de la posición con el robot móvil, ya no se conservan, irán desapareciendo gradualmente (ya que no se vuelve a medir sobre esas posiciones del espacio y no se actualizan las celdas). En su lugar serán actualizadas celdas que sí mantienen una relación espacial relativa al robot correcta. En definitiva, el efecto global es el del mantenimiento de las posiciones relativas de los obstáculos respecto al robot en el mapa, y el paso del tiempo no degrada esta relación de una forma gradual y acumulativa.

3. **Objetos dinámicos:** En numerosas aplicaciones los robots móviles han de actuar en entornos dinámicos, lugares donde puedan circular otros robots o personas. En principio, el sistema sensorial del robot deberá detectar este tipo de objetos y así actualizará el mapa con las medidas realizadas, pues al cambiar la localización de los objetos, el mapa

del espacio ocupado en un instante pasado puede ser distinto al actual. Sin embargo, si se va decrementando la **CPO** de las celdas del mapa que no son actualizadas por medidas, tras un determinado lapso de tiempo, la zona se considerará libre, ya que sobre este espacio no se vuelve a detectar objeto alguno.

4. **Reducción de la incertidumbre en la localización angular:** La incertidumbre en la localización angular del punto emisor del eco se reduce. En efecto, si se actualizan todas las celdas que cubre el cono acústico a la distancia R a la que se detecta un objeto P, una amplia zona del mapa será considerada como ocupada cuando no lo esté. Sin embargo, según se acerque el robot al objeto P, el área que barre el cono acústico a la distancia de medida será más pequeña, y la **CPO** de las celdas consideradas como ocupadas irá disminuyendo por no ser actualizadas por una nueva medida: Mediante este método se consigue una mayor definición de las fronteras de los diversos objetos según se acerque el robot a ellos.

En definitiva, se puede concluir que mediante el uso de un mapa del entorno constituido por celdas que indican la **CPO** y el instante de adquisición (T_a), y mediante el empleo del *decremento temporal* de la **CPO**, se consigue una mejora notable en la disminución de los errores y comprensión de las diversas fuentes de incertidumbre que influyen en el proceso de adquisición y generación del mapa del entorno, que pasamos a resumir a continuación:

- Las celdas afectadas por falsas medidas (debidas a reflexiones múltiples, falsos ecos, crosstalk, etc.) irán desapareciendo con el tiempo.
- No existe ambigüedad entre medidas recientes y falsas medidas.
- La relación espacial entre las medidas realizadas y la posición del robot móvil se conserva con un error fijo y no acumulativo.
- El error en la resolución de sensor ($\pm\epsilon$) se tiene implícitamente en cuenta al estar constituido el mapa por celdas de determinadas dimensiones (x).
- Indirectamente se consigue una reducción de la incertidumbre de la localización angular del objeto detectado.
- Mediante el uso de *mapas acumulativos* las superficies lisas dejan de ser un elemento peligroso y prácticamente invisible para un sistema de ultrasonidos: Por un lado, este tipo de superficies tienen por lo general algún tipo de irregularidad que hace que sea posible su localización. Por otro lado, si el eje del transductor es perpendicular a la superficie, ésta será detectada sin problemas. Es decir, a pesar de que en un número de lecturas de los sensores no se detecte este tipo de elementos, algunas medidas sí reflejarán sobre el mapa su posición.

  En conclusión, una navegación del robot basada en la información suministrada por el mapa del entorno será más robusta y segura que una navegación reactiva, basada únicamente en las últimas lecturas realizadas por los sensores.

3.5 Sistema

Una vez definidos los conceptos teóricos de la generación y mantenimiento del mapa basado en la información suministrada por los sensores de ultrasonido, se pasará a describir los diferentes aspectos involucrados en la implementación práctica de un mapa de acumulación, su evolución a medida que se van realizando nuevas medidas y el mantenimiento de las mismas a lo largo del tiempo.

3.5.1 Incremento de la CPO de una celda.

Tal y como se ha descrito en el apartado 3.3, se pueden distinguir dos tendencias claramente diferenciadas a la hora de actualizar el valor de la **CPO** de una celda por una medida realizada por un sensor de ultrasonido: La actualización probabilística de un conjunto de celdas, o el incremento de la **CPO** de una única celda (la que se encuentre a la distancia medida sobre el eje del sensor).

Por un lado, la actualización probabilística de un conjunto de celdas afectadas por una medida conlleva un costo computacional muy alto (en términos de tiempo de procesamiento). Por otra parte, una actualización teóricamente muy precisa, pierde parte de su sentido a la hora de realizar tareas de navegación. En el caso del presente trabajo, una estimación cuantitativa muy precisa de la presencia de un obstáculo en una determinada región del espacio no incrementa la fiabilidad de las acciones de control que debe realizar el robot móvil. A los algoritmos de navegación les es suficiente un conocimiento "cualitativo" de la presencia de obstáculos en su camino hacia la meta. En este sentido se dirige la filosofía de actualización del incremento del la **CPO** de Kohren y Bohrenstein. Estos sistemas se fundamentan en la adquisición de señales ultrasónicas de muestreo muy alto. De esta forma se puede obviar el problema de la incertidumbre angular en la medida, ya que se realizarán un número muy alto de medidas sobre cada celda.

Sin embargo, el robot móvil sobre el que se ha realizado todo el desarrollo experimental del presente trabajo, no disfruta de un sistema de adquisición tan sofisticado como el descrito. En nuestro caso hay que contar con un tiempo de latencia de 0.7 segundos entre medidas de un mismo sensor. Por tanto, la **CPO** de una celda se incrementará siempre que la celda sea afectada por la medida de un sensor de ultrasonido. Además, se actualizarán también las celdas vecinas a la misma que estén dentro del campo de acción del lóbulo acústico.

El valor del incremento (Δ_{CPO}) de la **CPO** de una celda dependerá de:

- La distancia (d) entre el sensor y la celda.
- El ángulo (θ) entre el eje principal del sensor y la celda.

Distancia entre Sensor y la Celda

Como se describió en el apartado 3.2, la probabilidad de que un obstáculo se encuentre realmente en la posición estimada por el sensor es inversamente proporcional a la distancia medida. Por tanto, partiendo de un valor máximo de incremento (K_d) de la **CPO**, definido como de **Máxima Actualización**, el valor de incremento de la **CPO** estará definido por la función que relaciona el incremento de la **CPO** con la distancia (d) a la que se encuentra la celda del sensor:

$$y(d) = K_d \cdot e^{-d/\alpha} \tag{3.10}$$

donde:

- K_d es una constante, que marca el valor máximo de actualización de una celda ($y(0) = K_d$)
- α marca la pendiente de la función, y por tanto indica como decrece la posibilidad de que la lectura del sensor sea correcta.

Mediante la asignación de la constante α, se especifica el valor de la actualización de la **CPO** (I_{max}) para la máxima distancia (d_{max}) que puede detectar el sensor:

$$\alpha = \frac{-d_{max}}{ln(I_{max}/K_d)}$$

Ángulo entre el Eje del Sensor y la Celda

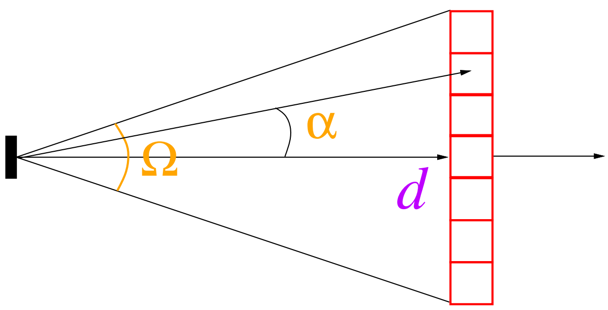


Figura 3.9: Variables en la actualización de la **CPO** de una celda.

Como ya se describió en la sección 3.2, el campo de actuación del pulso que se emite desde un transductor tiene forma cónica y, por tanto, la probabilidad de presencia de obstáculo en la zona que abarca el lóbulo acústico no es del todo despreciable. La certeza de que el obstáculo se encuentre sobre el eje central del cono acústico es máxima, y por tanto el incremento (I) de la **CPO** será máxima en esa zona.

Por otro lado, la certeza irá disminuyendo a medida que nos alejemos del eje central de sensor.

La función que establece el valor del incremento de las celdas que se encuentran sobre el lóbulo del sensor de ultrasonido a la distancia de medida, viene dada por:

$$y(\theta) = 1 - (2\theta/\Omega)^2; \quad \forall\theta \in [-\frac{\Omega}{2}, \frac{\Omega}{2}] \tag{3.11}$$

donde:

- θ es el ángulo que forma la celda con el eje principal del sensor.
- Ω es el ángulo sólido del sensor.

Resumiendo, el incremento que experimentará la **CPO** de una celda será función tanto de la distancia a la que se encuentre la misma del sensor que realizó la medida (ecuación 3.8) como de la distancia que la separa del eje central del sensor (ecuación 3.11):

$$\Delta_{CPO} = y(d) \cdot y(\theta) = K_d \cdot e^{-d/\alpha}(1 - (2\theta/\Omega)^2); \quad \forall\theta \in [-\frac{\Omega}{2}, \frac{\Omega}{2}],\ d \in [0, d_{max}] \tag{3.12}$$

3.5.2 Decremento Temporal de la CPO

Por los principios detallados en el apartado 3.4, la **CPO** de una celda se decrementará con el tiempo, si no se vuelven a realizar medidas sobre ella. Para decrementar el valor de la **CPO** de una celda se tendrán en cuenta los siguientes factores:

Como ya se ha comentado, medidas recientes y falsas medidas tendrán una **CPO** baja. El sistema de navegación siempre supondrá que toda celda con un valor de **CPO** mayor que cero está ocupada. Sin embargo, resulta necesario que la **CPO** de aquellas celdas que hayan sido actualizadas de una forma incorrecta tienda a cero lo más rápido posible, a la par que las celdas cuya **CPO** sea elevada mantengan su valor durante el mayor periodo de tiempo posible. Esto se expresa en un *Tiempo de Mantenimiento (T_m)* diferente que depende del rango en el que se encuntre la **CPO**. La Tabla 3.1 expresa la relación entre el *Tiempo de Mantenimiento (T_m)* y la **CPO** de una celda.

CPO	T_m
$0 < CPO < 10$	$T_m = 2$ seg
$10 \leq CPO \leq 30$	$T_m = 10$ seg
$CPO > 30$	$T_m = 15$ seg

Tabla 3.1: Relación entre el Tiempo de Mantenimiento y la **CPO**

El *Tiempo de Mantenimiento* está relacionado con el ritmo de decrecimiento de la **CPO**, y la consecuencia es un ritmo de decrecimiento de la **CPO** diferente para cada rango. En el caso de celdas con **CPO** baja, su ritmo de decrecimiento será más rápido que en celdas con **CPO** alta.

Sin embargo, es muy importante tener en cuenta el hecho de que una medida reciente pueda no ser una falsa medida. Por tanto, no sería muy conveniente hacer decrecer todas las celdas con **CPO** baja a un ritmo muy elevado, ya que podría ocurrir que debido a la configuración de los sensores y a la frecuencia de muestreo de los mismos, ninguna celda consiguiese alcanzar un valor de **CPO** mayor que el valor de Máxima Actualización. Por tanto, es necesario mantener durante un margen de tiempo mínimo la **CPO** de una celda para poder "comprobar" mediante la evidencia de otras medidas sobre la zona ocupada por la celda, si se trata una medida sobre un espacio ocupado o una falsa medida. Empíricamente hemos calculado que el *Tiempo de Mantenimiento* necesario para valores de **CPO** bajos es de 2 segundos (véase Tabla 3.1).

3.6 Resultados Experimentales

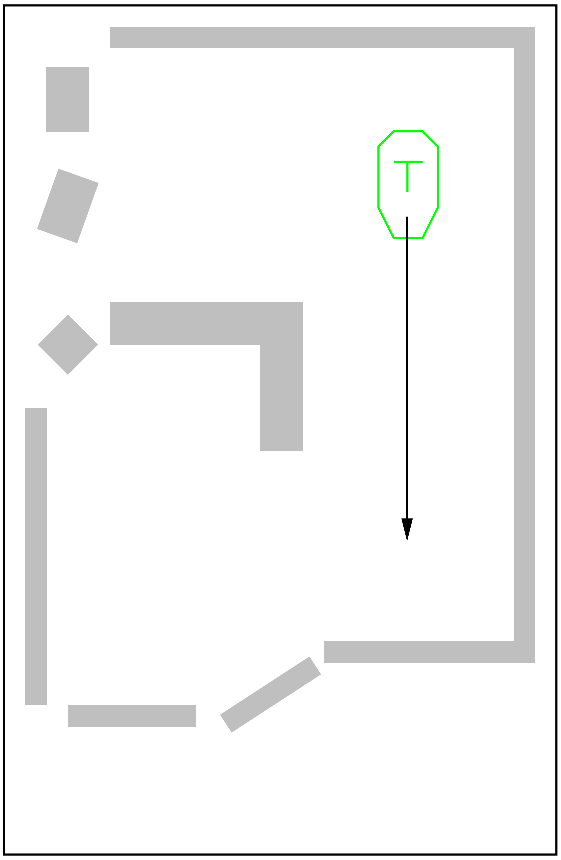


Figura 3.10: Escenario donde se realizaron las pruebas experimentales

Los resultados experimentales que se van a describir a continuación fueron realizados en un escenario como el que se describe en la Figura 3.10. El robot siguió un movimiento en línea recta, a una velocidad de 10 cm/s en el sentido que indica la flecha de la Figura 3.10. La

Figura 3.11 indica, por una parte, el recorrido del robot (línea a trazos) y todas las lecturas realizadas por los sensores de ultrasonido a lo largo de la trayectoria. En esta Figura, se puede observar cierta regularidad en las medidas que corresponden a objetos que se encuentran en el entorno. A su vez, las falsas medidas son fácilmente identificables por estar dispersas en el espacio y fuera de los márgenes delimitados por las paredes y demás obstáculos (por ejemplo, las medidas que están comprendidas en el rango $740 < x < 1000$).

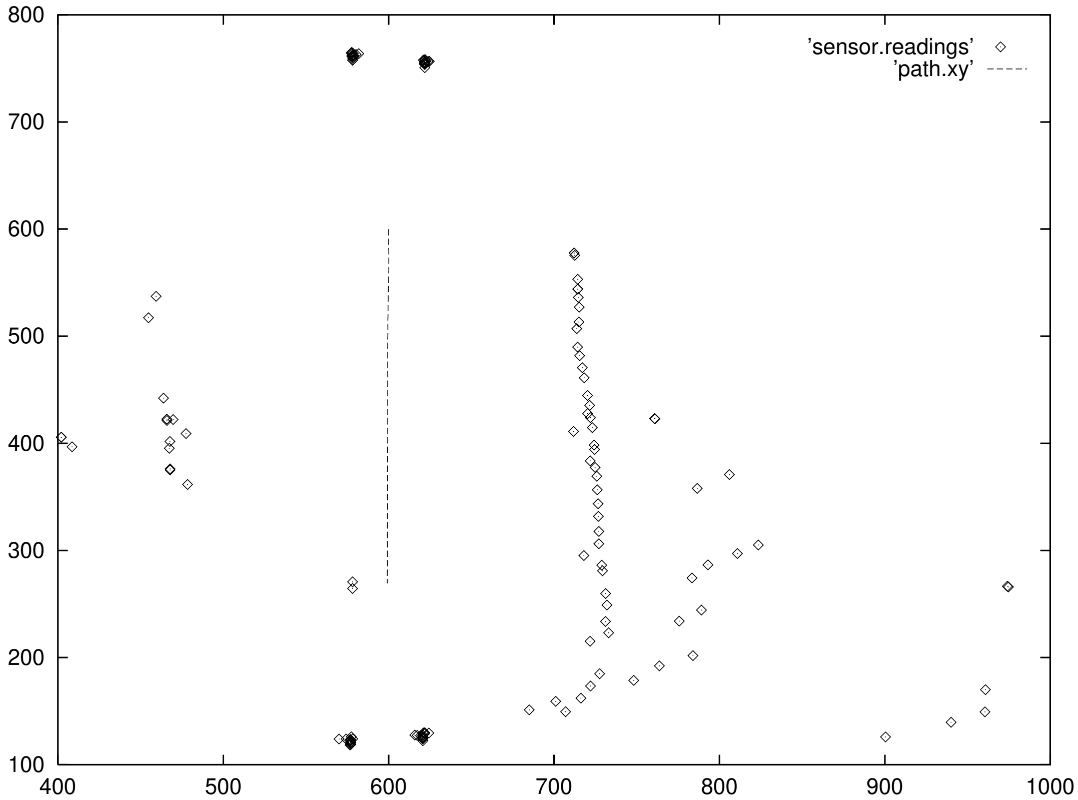


Figura 3.11: Representación de todas las medidas realizadas por los sensores de ultrasonidos y la trayectoria del robot

La Figura 3.12 muestra como se va generando el mapa de acumulación en los últimos instantes del recorrido del robot. La posición del robot, en el instante en el que fue tomado el estado del mapa, está representada por un rectángulo negro, mientras que las posiciones anteriores están marcadas con rectángulos de color mas claro. Las celdas tienen asignados un color y un número. El número representa el valor de la **CPO** y el color la relación entre el *Tiempo de mantenimiento* y la **CPO**, expresada en la Tabla 3.1. Los colores más oscuros indican un rango de **CPO** más elevado, por tanto, un *Tiempo de Mantenimiento* también más elevado. El color más claro indica celdas con una **CPO** con un valor comprendido en un rango $1 \leq$ **CPO** ≤ 10.

En la Figura 3.12 se puede observar cómo se van delimitando los diferentes objetos presentes en el entorno en la representación de celdas. A su vez, las celdas que se corresponden con objetos ven cómo va aumentando su **CPO** a medida que se van realizando más medidas sobre la zona del espacio que ocupan. Las falsas medidas desaparecen del mapa con el paso del tiempo, y las correctas inducen un aumento en la **CPO** de la celda correspondiente. Es muy importante

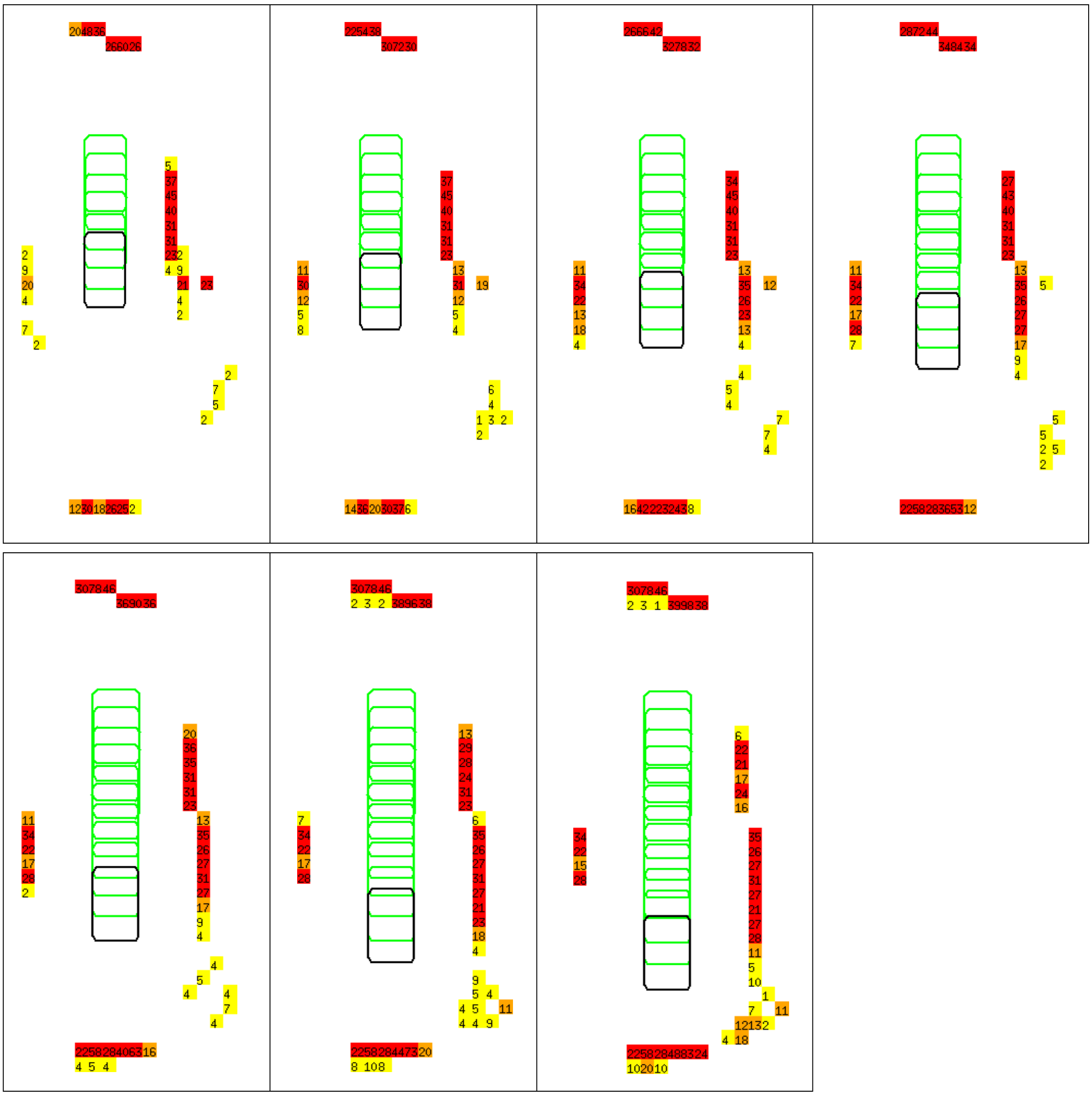


Figura 3.12: Generación del Mapa y variación de la **CPO** en los últimos instantes.

destacar que las zonas a las que se acerca el robot van definiéndose con mayor nitidez. Es decir, las celdas que se corresponden con objetos va aumentando de forma significativa, y que celdas cercanas a los objetos van disminuyendo su **CPO** hasta cero, al no haber sido confirmada la presencia de obstáculo por la evidencia de nuevas medidas.

Es muy importante resaltar el hecho de que en la generación del Mapa se pretende tener un conocimiento más detallado de las zonas cercanas al robot que de las lejanas, puesto que los diferentes módulos de navegación que hagan uso de esta representación serán locales. Es decir, la respuesta de estos módulos, en términos de acciones de control de los actuadores del robot móvil, será debida a una representación de un entorno cercano y centrado en el robot*. Por tanto, a la hora de realizar una representación del entorno es mucho más importante que la representación de los obstáculos cercanos al robot se ajusten más a la realidad que aquellos que se encuentren lejos del mismo. Sin embargo, no por este hecho se desecharan las medidas que indiquen presencia de objetos alejados del robot: Estas medidas se actualizarán también en el Mapa de Acumulación, y nuevas medidas y el tiempo decidirán si pertenecen realmente a objetos físicos de entorno.

No obstante, siempre es posible realizar un Mapa detallado del entorno, si la aplicación así lo requiere, variando ligeramente los diferentes *Tiempos de Mantenimiento*. Un resultado como el que se puede observar en la Figura 3.13 se obtiene con los *Tiempos de Mantenimiento* de la Tabla 3.2:

CPO	T_m
$0 < CPO < 10$	$T_m = 2$ seg
$10 \leq CPO \leq 30$	$T_m = 50$ seg
$CPO > 30$	$T_m = 300$ seg

Tabla 3.2: Relación entre el Tiempo de Mantenimiento y la **CPO** para el caso de querer obtener un Mapa del entorno detallado.

La generación del Mapa que se puede observar en la Figura 3.13 fue realizada autónomamente por el robot móvil, mediante la activación de los agentes de evitar obstáculos y seguir paredes. En la Figura 3.14 se pueden observar todas las medidas realizadas por el robot móvil y la trayectoria seguida por el mismo. El número de medidas falsas fue muy elevado, pero no fue óbice para que el mapa generado por el robot se ajustase a la realidad del entorno, tal y como se puede observar en la Figura 3.13.

*Tal y como se describirá en el apartado dedicado a la definición de la **Función de Accesibilidad.**

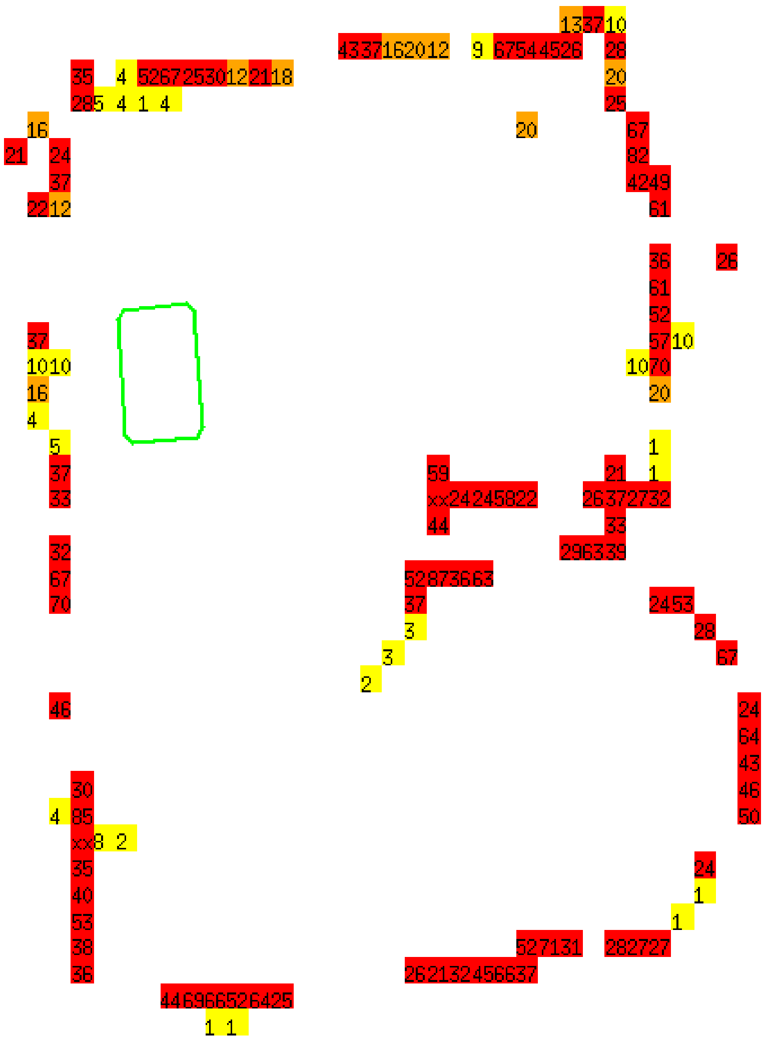


Figura 3.13: Mapa generado

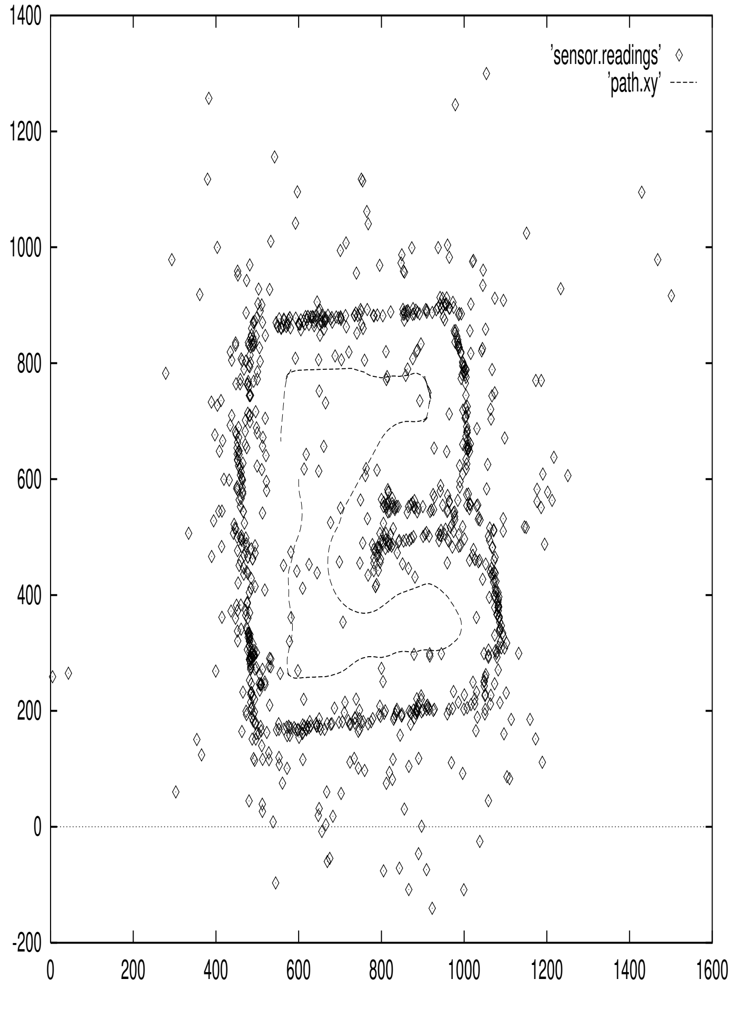


Figura 3.14: Representación de todas las medidas realizadas por los sensores de ultrasonidos (*sensor.readings*) y el camino del robot para realizar el mapa del entorno (*path.xy*)

Capítulo 4

Función de Accesibilidad

4.1 Introducción

Todos los robots móviles requieren algún tipo de sistema para evitar obstáculos; desde algoritmos primitivos que detectan un obstáculo y simplemente paran el robot, hasta sofisticados algoritmos que permiten al robot desviarse para evitarlo. Estos últimos son mucho más complejos, ya que no involucran únicamente la detección del obstáculo, sino que requieren un cierto conocimiento cuantitativo del entorno y de las dimensiones del mismo.

La navegación autónoma de un robot móvil presenta un nivel de complejidad más elevado, ya que se ha de evitar el obstáculo al mismo tiempo que se ha de tener en cuenta la posición del objetivo al que se desea llegar. En este sentido, se puede hacer una clasificación de los algoritmos en aquellos que realizan una planificación de alto nivel sobre una representación simbólica del entorno y los algoritmos de navegación local.

4.1.1 Navegación Global

Los métodos de navegación global se basan en la elaboración de la trayectoria a seguir por el robot móvil, sobre una representación del espacio. En este sentido, una de las alternativas más utilizadas ha sido la creación de un mundo simbólico que representa los obstáculos como poliedros [64]. En este tipo de representación, los obstáculos se agrandan de tal forma que representan posiciones prohibidas un robot reducido a un punto. Así, mediante la utilización de grafos de visibilidad, una trayectoria que evita las regiones prohibidas es una trayectoria libre de colisiones. Esta trayectoria se puede optimizar atendiendo a ciertos criterios, como recorrer la mínima distancia posible, mediante la utilización de algoritmos sub-óptimos de búsqueda, tipo A^* [50].

También destaca la utilización del espacio de configuraciones [63], donde el robot está caracterizado por su posición y orientación como un punto en un espacio de configuraciones, en el cual cada coordenada representa un grado de libertad del robot. Así, los obstáculos están marcados como coordenadas prohibidas de este espacio. El algoritmo de búsqueda simplemente ha de encontrar la trayectoria desde la posición actual hasta la configuración deseada.

La lista de métodos de navegación global es muy extensa, y el libro *Autonomous Mobile Robots: Perception, Mapping and Navigation (Volume 1)* [54] muestra una visión muy amplia y profunda del tema.

4.1.2 Navegación Local

Algunos métodos de navegación local, como los basados en campos de potencial [56], se han utilizado comúnmente para la planificación de movimientos muy precisos por un robot manipulador. Estos métodos calculan una función a partir de los datos obtenidos de los sensores del robot, y su resultado es un vector que guiará el movimiento siguiente del robot.

Sin embargo, la característica principal que diferencia estos métodos de los de navegación global es la ausencia de un modelo simbólico del entorno. En este sentido, algunos sistemas [74] [29] se basan en las lecturas que proveen los sensores en un único instante. Tal y como ya se ha comentado en el apartado dedicado a los errores asociados a los sensores de ultrasonido, este tipo de estrategia puede ser peligroso para la integridad del robot y la de los objetos del entorno.

Por otro lado, se pueden destacar aquellos sistemas [17] [16] [15] [39] que utilizan un modelo del entorno (principalmente mapas de acumulación) para tratar los errores e incertidumbres asociados a los sensores. Sobre este modelo se calcula una función local que determina la siguiente acción del robot.

El sistema descrito por *Bohrenstein* [17] tiene grandes similitudes con la *Función de Accesibilidad* que presentaremos a continuación. Utiliza un mapa de acumulación para tratar los errores en los sensores de ultrasonido. Sin embargo, al calcular la función local (denominada *Histograma Polar*) utiliza el grado de la certeza de presencia de obstáculos en cada una de las celdas del mapa.

En resumen: El modelo del entorno proporcionado por el mapa de acumulación servirá para reducir los errores e incertidumbres de los sensores de ultrasonido. Sin embargo, este modelo es incompleto y el conocimiento del entorno en lugares alejados del robot es impreciso. Por tanto, hemos definido una función local que permite a los agentes motores de navegación conocer el estado del entorno y generar acciones de control basadas en esta información.

4.2 Función de Accesibilidad

La Función de Accesibilidad (F_A) proporciona una estimación cuantitativa de las zonas del espacio que rodean al robot en un instante determinado t_i. A partir de esta función, se puede interpretar en términos cualitativos de la proximidad de los obstáculos al robot. En la definición y cálculo de la F_A se tendrá en cuenta tanto la estructura geométrica como la cinemática del robot.

El fin último de la elaboración de la F_A es proporcionar una representación del entorno local, de forma que los comportamientos motores de navegación puedan establecer una acción de control de una manera robusta y fiable. Mediante la definición de esta función se podrán realizar tareas de navegación, como evitar obstáculos, seguir paredes, etc.

4.2.1 Generación

Antes de explicar cómo se genera la función de accesibilidad, interesa hacer ciertas consideraciones y definiciones. La generación de la F_A se basa en la información contenida en el mapa de acumulación, que se interpreta como sigue: Se considera que una celda del entorno del robot está "ocupada", siempre que la **CPO** de la celda tenga un valor mayor que cero.

Por otra parte, la F_A depende de las características geométrico-cinemáticas del robot móvil. En este sentido, el robot móvil del que disponemos tiene una anchura D y una longitud L (ver Figura 4.1). Cabe destacar que se trata de un robot móvil que se puede mover omnidireccionalmente respecto al punto central del eje que une las dos ruedas motrices.

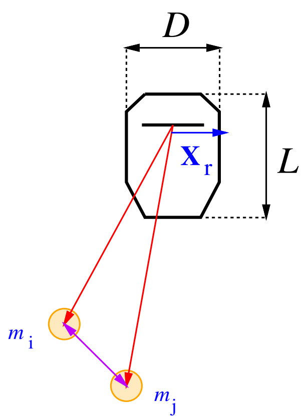


Figura 4.1: Definición de sector

Sistema de Referencia: La F_A es una representación del entorno local que rodea al robot móvil y, por tanto, interesa definirla respecto a un sistema de referencia solidario al propio robot. Para mantener una perfecta simetría en el cálculo de la F_A, se elegirá como el origen de este sistema al centro del eje que une las dos ruedas motrices del robot $\vec{X}_r$, alrededor del cual se pueden realizar giros ominidireccionales. El eje de las abscisas es paralelo al eje de las ruedas motrices, mientras que el eje de ordenadas tendrá la misma dirección y sentido que el de avance del robot. Finalmente, los ángulos en la representación polar de la F_A seguirán un sentido anti-horario.

Sector: Un *sector* S_{ij} es la zona definida por los centros geométricos de dos celdas ocupadas del mapa, $m_i = (m_x^i, m_y^i)$, $m_j = (m_x^j, m_y^j)$, y el origen del sistema de referencia. (ver Figura 4.1).

Sector Accesible: Un sector accesible será todo aquel que satisfaga la relación:

$$\sqrt{(m_x^i - m_x^j)^2 + (m_y^i - m_y^j)^2} \; > \; D \tag{4.1}$$

Sector No Accesible: Un sector no accesible será todo aquel que satisfaga la relación:

$$\sqrt{(m_x^i - m_x^j)^2 + (m_y^i - m_y^j)^2} \; \leq \; D \tag{4.2}$$

Siendo en ambos casos D la anchura del robot móvil.

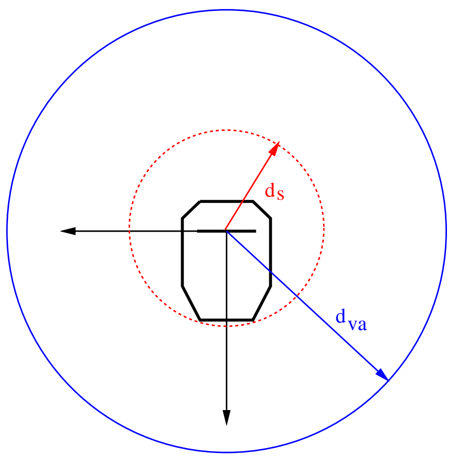


Figura 4.2: Ventana Activa y Zona de Seguridad

Ventana Activa: Dada la posición actual del robot, interesa conocer las zonas accesibles en un entorno local al robot, y por ello se define la Ventana Activa como un un círculo, centrado en el origen del sistema de referencia, y de radio d_{va} (ver Figura 4.2), que dependerá de las dimensiones del entorno en el que trabaje el robot. Los sectores no accesibles alrededor del robot **sólo** se calculan en este espacio restringido de la ventana activa.

Zona de Seguridad: La Zona de Seguridad es un círculo en el que el robot puede girar libremente sin colisionar con obstáculo alguno. En el caso de nuestro robot móvil, éste se puede mover omnidireccionalmente en torno al punto central del eje que pasa por sus dos ruedas motrices. Por tanto, la zona de seguridad estará delimitada como el círculo centrado en el origen del eje del robot y de radio d_s, siendo d_s la distancia entre el origen y el robot móvil del extremo más alejado del mismo (ver Figura 4.2).

Distancia $d(\theta_l)$: es la longitud del segmento cuyo origen es el del sistema de referencia y cuyo extremo es la intersección de una recta que pasa por el origen y forma un ángulo θ_l con el eje x, con el segmento $\overline{m_i m_j}$.

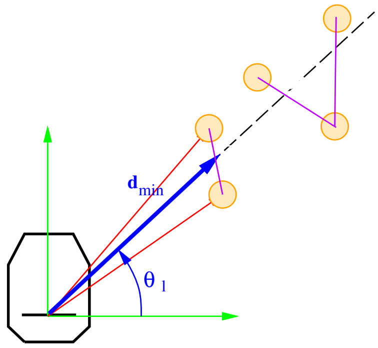


Figura 4.3: Distancia mínima.

Distancia mínima $d_{min}(\theta_l)$: La orientación θ_l puede estar contenida en un elevado número de sectores no accesibles. La distancia mínima $d_{min}(\theta_l)$ estará definida como la mínima de las distancias $d(\theta_l)$ en la orientación θ_l (véase Figura 4.3).

Por tanto, el conjunto de sectores no accesibles alrededor del eje de referencia relativo del robot móvil, calculados dentro de la ventana activa, dará lugar a un conjunto de distancias mínimas que permitirán definir la **Función de Accesibilidad** (F_A) como una función del tipo:

$$f_a(\theta_i) = d_{min}(\theta_i) = d_i \tag{4.3}$$

Esta función polar barrerá todo el espacio que rodea al robot dentro de la ventana activa. θ_i tomará valores en el dominio $[0, 360[$, y d_i será la distancia mínima $d_{min}(\theta_i)$ en cada dirección θ_i.

4.2.2 Normalización de la Función de Accesibilidad

Generalmente resulta mucho más fácil, al realizar algoritmos de control basados en información sensorial, trabajar en términos relativos que en absolutos. En este caso, mediante la función expresada en (4.3) obtenemos una medida de la distancia que separa al centro del coche de los obstáculos dentro de la ventana activa. Con este tipo de función, se debería conocer explícitamente las dimensiones del robot para saber cómo de *peligrosa* o *cercana* es esa distancia respecto al robot. Por tanto, se ha normalizado la función (4.3) mediante:

$$\Gamma(d_i) = 1 - \frac{d_i - d_s}{d_{va} - d_s} \tag{4.4}$$

Con las condiciones:

$$\Gamma(d_i) \quad = 1.1; \quad ; d_i < ds \tag{4.5}$$

Siendo:

- d_s: El radio de la zona de seguridad.
- d_{va}: El radio de la ventana activa.

En definitiva, la **Función de Accesibilidad** (F_A) estará definida como:

$$F_A(\theta_i) = \Gamma(f_a(\theta_i)) \tag{4.6}$$

e indica la cercanía de un obstáculo en una dirección θ_i dentro de la zona restringida por la **Ventana Activa**, estando definida en un dominio $\theta_i \in [0, 360[$. El resultado de la F_A está definido en un rango $[0, 1.1]$, indicando el valor 0 la ausencia de obstáculo dentro de la ventana activa y el valor 1.1 la presencia de obstáculo dentro de la **Zona de Seguridad**.

4.2.3 Resultados Experimentales

En la definición de la F_A existe una serie parámetros que dependen de la geometría del robot, y otros que puede especificar el usuario.

Especificación de Parámetros

El radio de la *Zona de Seguridad* no es d_s (tal y como se muestra en la Figura 4.2), sino que se incrementa en una cantidad mayor que depende de:

- Las dimensiones de las celdas: Debido a que en el cálculo de los sectores accesibles se han considerado las celdas puntuales como su centro geométrico, una porción del espacio ocupado por la celda que no se ha tenido en cuenta en el cálculo de la F_A.
- Los agentes motores de navegación que utilizan un radio de la zona de supervivencia un poco mayor al definido, simplificándose los algoritmos de control en gran medida (véase el Capítulo 5).

En cuanto a la *Ventana Activa*, su tamaño dependerá de las dimensiones del entorno y del robot. No existe un criterio analítico que establezca el tamaño óptimo de esta zona, sino que la experiencia del ingeniero y las necesidades de los agentes motores de navegación establecerán su tamaño.

En nuestro caso, los radios de la zona de seguridad y la ventana activa, han sido de 90 cm y 130 cm, respectivamente.

Ejemplos

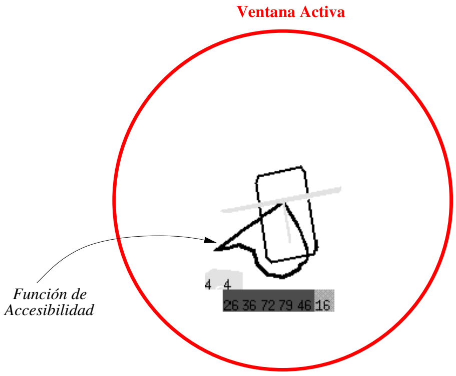


Figura 4.4: Ejemplo de la F_A

En la Figura 4.4 se puede observar la forma de la F_A calculada dentro de la zona delimitada por la ventana activa. En esta figura, se han incluido a modo de referencia, el círculo que define la ventana activa y el sistema de referencia asociado al robot móvil. Se puede apreciar cómo la F_A toma valores más altos en las direcciones en las que el robot está más cercano al obstáculo (marcado por las celdas ocupadas del mapa de acumulación).

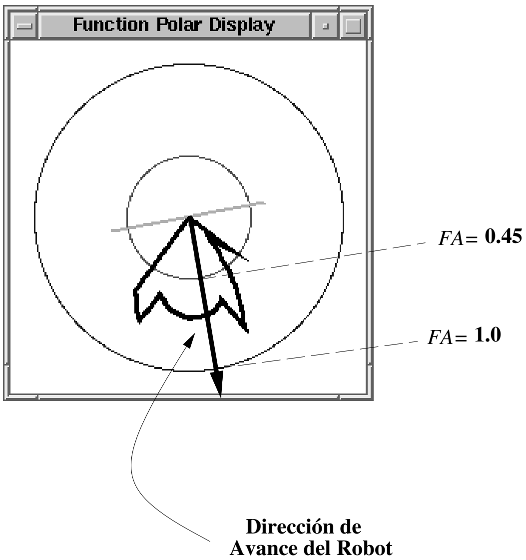


Figura 4.5: Ventana de F_A.

La Figura 4.5 muestra con mayor detalle la representación de la F_A utilizada en las Figuras 4.6 y 4.7. La circunferencia más externa marca el valor 1.0 de la F_A, y la circunferencia interior simplemente sirve como referencia, marcando el valor 0.45 de la F_A. Asimismo, la orientación del robot móvil está indicada por medio del sistema de referencia asociado al robot móvil.

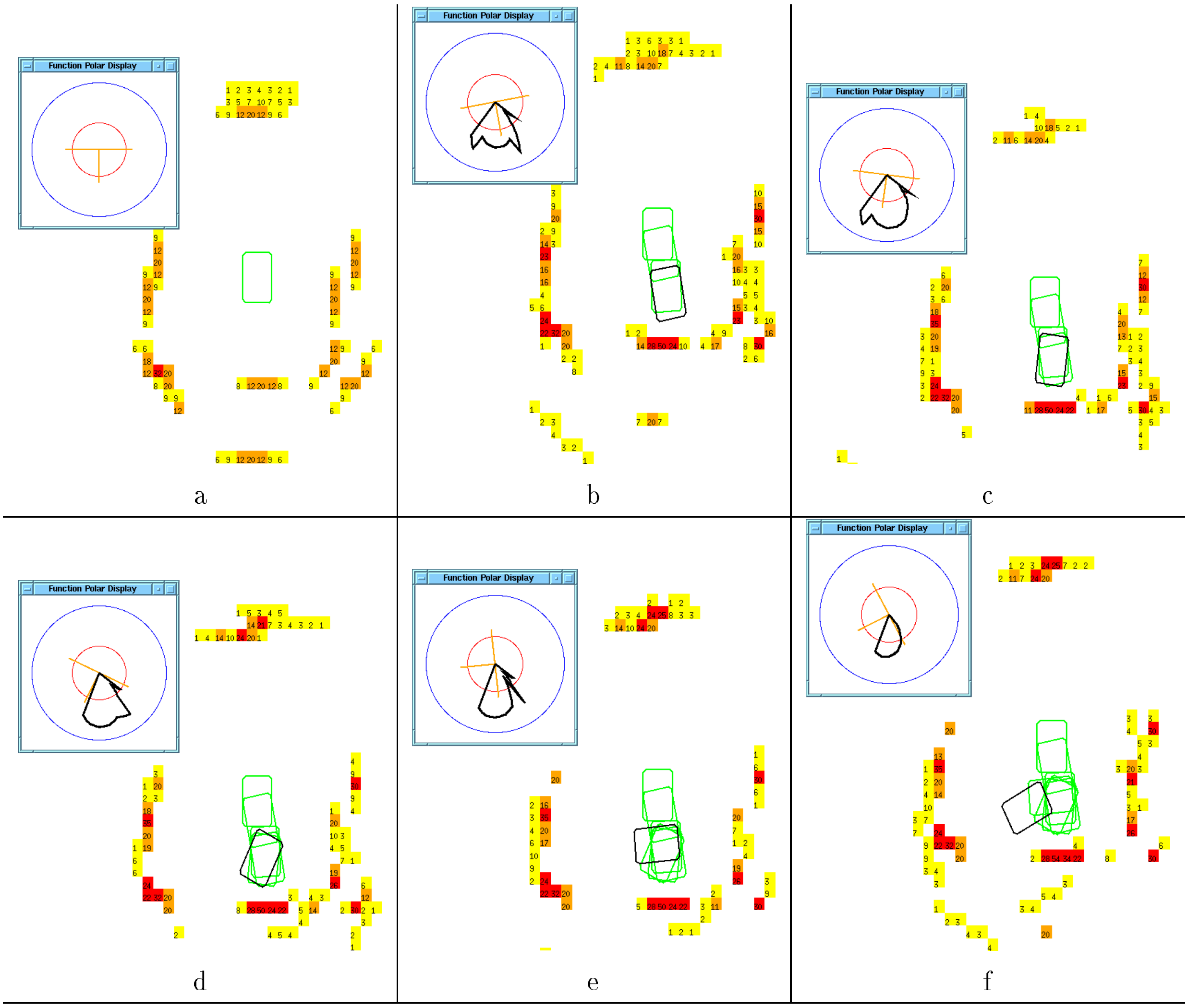


Figura 4.6: Ejemplo de la F_A (A)

Las Figuras 4.6 y 4.7 muestran las imágenes de una secuencia de movimientos del robot. En el recuadro del extremo superior izquierdo de cada una de las imágenes, se puede observar la Función de Accesibilidad calculada en cada momento. Asimismo, se ha incluido la localización del robot dentro del mapa de acumulación para poder observar con mayor detalle la relación entre la F_A y el mapa de acumulación. Por último, cabe destacar que la secuencia temporal en ambas figuras es de izquierda a derecha y de arriba a abajo.

La Figura 4.6-a) muestra el robot en un entorno local sin obstáculos. Las Figuras 4.6-b) y 4.6-c) muestran al robot en una situación en la que se encuentra "en frente" de un obstáculo, dentro de la ventana activa. Finalmente, las Figuras 4.6-d) hasta 4.6-f), muestran los valores que toma la F_A cuando el robot tiene los obstáculos a su "izquierda", dentro de la ventana activa.

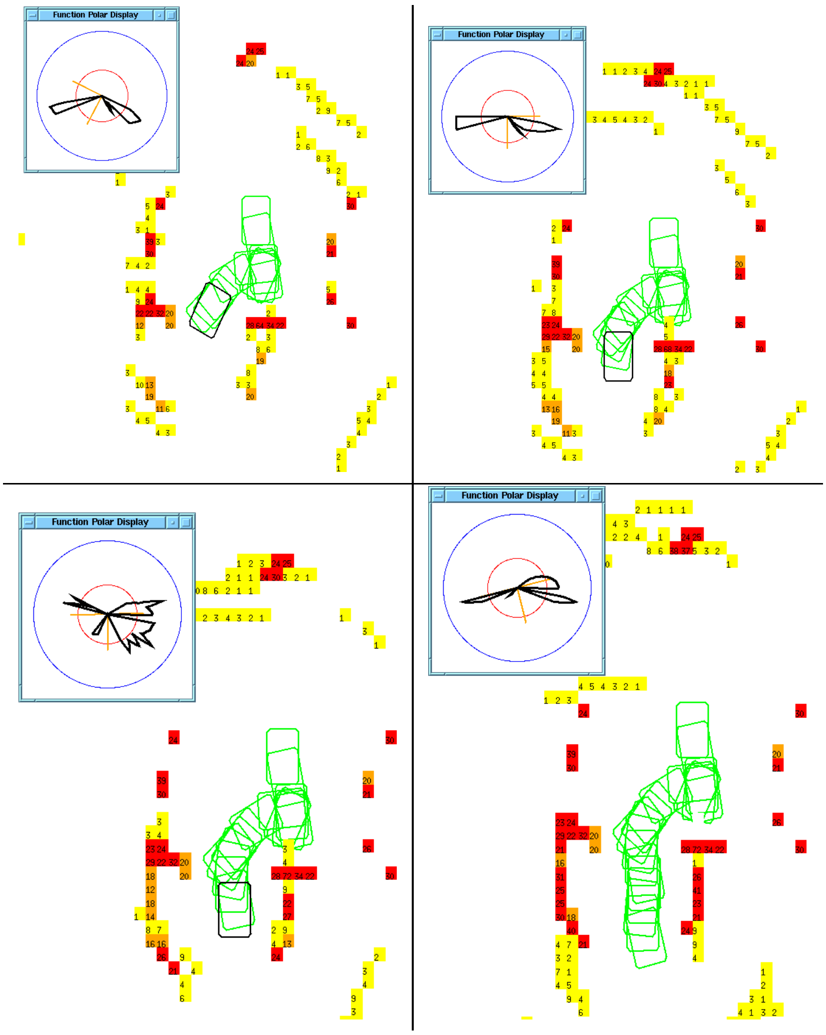


Figura 4.7: Ejemplo de la F_A (B)

Conviene destacar que numerosos algoritmos locales de navegación utilizan el concepto de campos de potencial para determinar la dirección de movimiento [12][56][20]. Con este tipo de algoritmos, las zonas angostas o pasillos delimitadas por obstáculos, producen "fuerzas repulsivas” que hacen moverse al robot de un lado para otro.

Sin embargo, mediante la utilización de la F_A, se conoce la presencia de los obstáculos en todas las direcciones que rodean al robot. Por tanto, los algoritmos de navegación pueden hacer un uso eficiente de esta información, traduciéndose en una conducción suave sin giros bruscos en zonas angostas. Las imágenes en la Figura 4.7 muestran la respuesta de los algoritmos motores de navegación del robot móvil, en una situación en la que los obstáculos dejan un paso angosto.

4.3 Conclusiones

Como conclusión, podemos decir que el cálculo de la función de accesibilidad sobre la representación del entorno como un mapa de acumulación, presenta las siguientes ventajas:

1. Se provee una representación cuantitativa del entorno local del robot móvil que indica el grado de “accesibilidad” en cada una de las direcciones.

2. No es necesario poseer grandes recursos computacionales, y su cálculo se realiza en tiempo real. Es decir, al contrario que en otros sistemas [32][14], no es necesario parar al robot para calcular la F_A.

Capítulo 5

Comportamientos Motores de Navegación

5.1 Introducción

Los comportamientos motores de navegación interpretan la información sensorial y dan como resultado una acción de control. En el sistema implementado en el presente trabajo, la información sensorial que recibirán parte de estos agentes no serán los datos crudos de los sensores de ultrasonido, sino una estimación de los alrededores del robot proporcionada por la *Función de Accesibilidad*. Por tanto, es necesario que el diseñador de los agentes dedicados a las tareas de navegación tenga un conocimiento detallado del tipo de información que reciben los agentes tanto de los sensores como de los módulos de tratamiento de la señal. En este sentido, pasaremos a hacer un breve resumen del significado de la *Función de Accesibilidad* y de los diferentes aspectos que la caracterizan en su relación con los agentes que van a realizar las tareas de navegación.

5.1.1 Función de Accesibilidad

La Función de Accesibilidad (F_A) proporciona una estimación del espacio ocupado que rodea al robot en un instante determinado, y se puede interpretar en términos cualitativos como una medida de la proximidad de los obstáculos al robot. La F_A es una función definida en coordenadas polares del tipo $F_A(\theta_i) = d_i$ que barre un determinado espacio que rodea al robot, definido como la *ventana activa*. La F_A está definida en el dominio $\theta_i \in [0, 360[$, y los valores d_i toman valores desde 0 hasta 1.1. El valor "0" representa la ausencia total de obstáculos en el entorno de la ventana activa, y el valor "1.1" indica que en la dirección θ_i se encuentra un obstáculo en la zona conocida como *zona de seguridad* con el que el robot podría colisionar. Valores de la F_A comprendidos en el rango $[0, 1]$, indican los diferentes grados de peligrosidad de los obstáculos respecto al robot.

Sistema de Coordenadas

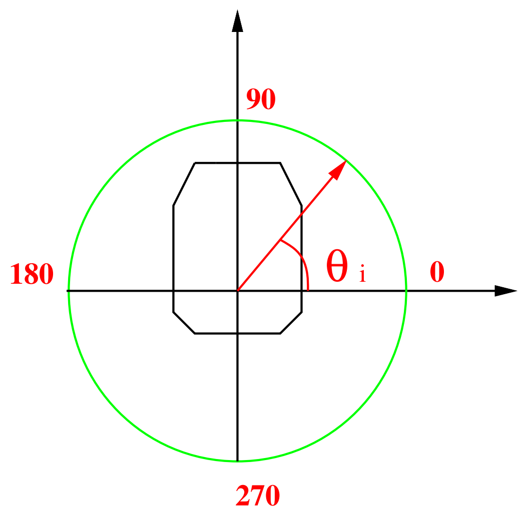


Figura 5.1: Sistema de Referencia centrado en el Robot Móvil y los valores de θ_i

Tal como se puede observar en la Figura 5.1, la F_A está referida a un sistema de coordenadas centrado en el centro del eje que une las ruedas motrices del robot móvil. Este sistema de coordenadas es solidario al robot, en el sentido de que el eje de ordenadas siempre tiene la misma dirección y sentido que el movimiento del robot, siendo la dirección $\theta_i = 90$ la dirección frontal del coche.

Etiquetas e Intervalos

Algunos agentes hacen abstracciones de alto nivel mediante la definición de etiquetas como **delante, detrás, izquierda o derecha**. Cada una de estas etiquetas corresponde a un intervalo en el dominio de la F_A. La Tabla 5.1 muestra un ejemplo de la relación entre los intervalos del dominio de la F_A y la etiqueta asociada a cada intervalo.

Intervalo del Dominio	Etiqueta
$\theta_i \in [90, 270]$	Izquierda
$\theta_i \in [270, 90]$	Derecha
$\theta_i \in [0, 180]$	Delante
$\theta_i \in [180, 0]$	Detrás

Tabla 5.1: Ejemplo de las diferentes etiquetas que se pueden asignar en función de los intervalos en el dominio de la F_A

5.1.2 Diseño e Implementación

En el proceso de diseño de una arquitectura basada en comportamientos, se especifican inicialmente un conjunto de aptitudes genéricas que debe tener el robot móvil*: Ir de una localización a otra, esquivar de forma autónoma los obstáculos que se encuentre en su trayectoria, seguir paredes, etc. Estas aptitudes se traducen en un conjunto de comportamientos implementables como agentes de una arquitectura.

El proceso de diseño e implementación de estos agentes es un proceso de abajo a arriba. Es decir, se comienza implementando y comprobando el buen funcionamiento de los agentes de más bajo nivel, y se añaden de forma gradual nuevos agentes de más alto nivel.

Este proceso requiere, sin embargo, nuevos refinamientos debido a la aparición de comportamientos emergentes no funcionales [86] al interactuar varios agentes. Es decir, el resultado de la interacción de varios agentes puede dar lugar a un comportamiento del robot que no se ajuste a las especificaciones iniciales.

La estructura general de la arquitectura de control y, más concretamente, la estrategia elegida para arbitrar los agentes imponen el procedimiento a seguir para resolver los conflictos entre los agentes. En una arquitectura *subsumption clásica* [26], el problema se resolvería mediante una cuidadosa elección de los tiempos de inhibición y supresión de las entradas y salidas de los comportamientos (véase Capítulo 2).

Sin embargo, en nuestra arquitectura se pueden añadir agentes que estén especialmente diseñados para detectar y resolver los problemas de la interacción entre agentes. La presencia de estos agentes se justifica por la necesidad de arbitrar al resto de los agentes del sistema e intentar no re-diseñar y re-implementar los agentes funcionales. En efecto, una alternativa hubiese sido volver a diseñar e implementar los agentes "conflictivos" hasta que el sistema global funcionase. Sin embargo, esto no sería lógico, ya que la funcionalidad de cada uno de los agentes ya había sido verificada con anterioridad.

*Véase en el Capítulo 7 una detallada descripción de las razones por las cuales se han implementado los agentes motores de navegación descritos en el presente capítulo.

5.1.3 Estructura del Capítulo

En el presente capítulo se realizará una descripción detallada de los diversos aspectos de implementación de los agentes motores de navegación. El orden de descripción de los agentes se hace atendiendo a su prioridad (de mayor a menor) dentro de la arquitectura de control. Sin embargo, esto no ha sido siempre posible, como en el caso del agente *Giro Propio* (sección 5.4), que ha sido descrito a continuación del agente *Ir a un Punto* por cuestiones de claridad. El agente *Giro Propio* fue implementado como resultado de la interacción conflictiva entre el comportamiento de "evitar obstáculos" y el de "ir a un punto".

Por otro lado, el número de trabajos y de artículos publicados sobre implementaciones de robots con una arquitectura basada en comportamientos es muy elevado (véanse [24] [26] [25] [43] [29] [66] [52] [74] [80], entre otros muchos). Como consecuencia directa de ello, se ha definido una serie de términos comunes, que ha dado lugar a denominar determinados comportamientos por sus nombres en inglés. Nosotros nos hemos adaptado a estos convencionalismos de la comunidad científica; así, por ejemplo, al comportamiento de "evitar obstáculos", lo definiremos indistintamente como *Avoid*, y al comportamiento de "ir a un punto" como *Goto Point*.

5.2 Evitar Obstáculos (Avoid)

El comportamiento de evitar obstáculos (*Avoid*) recibirá instrucciones de control (deseos) de los niveles superiores y mediante el estudio de la distancia que separa al robot de los obstáculos que puedan rodearle (evaluando la F_A), modulará los deseos recibidos y dará como salida un deseo propio en términos de acciones de control a tomar por los actuadores del robot móvil. *Avoid* intentará aproximarse lo más posible a los deseos de los niveles superiores. El deseo que reciba *Avoid* será un vector *(velocidad, giro)*, donde *velocidad* es la velocidad lineal (en cm/sg) que deba seguir el robot, y *giro* el giro a realizar (en rad/sg). El deseo de salida de Avoid también es un vector *(velocidad, giro)* que podrá alimentar a agentes de más bajo nivel o directamente al agente **Actuador**, que es el que recibe vectores en forma de deseo y los convierte en instrucciones que los controladores del robot móvil puedan entender.

5.2.1 Entrada y Salida

Figura 5.2: Acciones de control y deseos de entrada y salida en **Avoid**

En términos computacionales el agente *Avoid* recibirá un vector de entrada, $d_e = (velocidad, giro)$, y mediante la evaluación de la F_A dará como resultado un vector de salida, $d_s = (velocidad, giro)$. Por tanto, el agente de evitar obstáculos se puede definir como la función A:

$$\begin{aligned} d_s &= A(d_e) \\ d_s, d_e &: (\text{velocidad, giro}) \end{aligned} \tag{5.1}$$

Las propias características del robot móvil influirán a la hora de desarrollar los algoritmos de control del agente de evitar obstáculos. Sería deseable que los controles de robot admitiesen un rango de velocidades continuo, pero en el caso del robot del que disponemos no es así. En nuestro caso, el rango de velocidades a las que se pueden mover las ruedas motrices del robot es discreto y a saltos de $\pm$ 10 cm/s. Por tanto, los deseos de control de los diferentes módulos se verán restringidos a unos valores discretos muy determinados. En un robot con un rango continuo y suave de velocidades se podría implementar un control de giro y velocidad que fuese una función lineal dependiente de la F_A. Sin embargo, en nuestro caso deberemos discretizar el resultado de la F_A y actuar en consonancia.

5.2.2 Direcciones

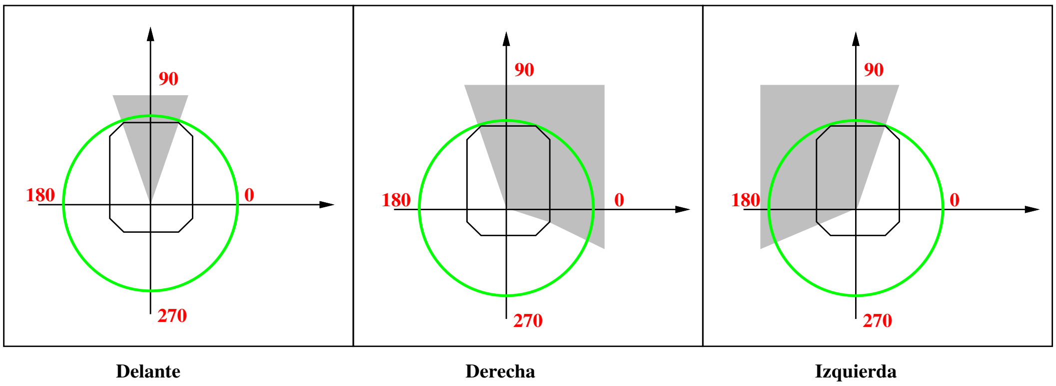


Figura 5.3: Direcciones: Las zonas sombreadas indican el intervalo de valores, θ_i, para cada una de las etiquetas lingüísticas (delante, izquierda, derecha), empleadas por el agente de evitar obstáculos (Avoid).

El comportamiento de evitar obstáculos asignará una serie de etiquetas lingüísticas a diferentes intervalos del dominio de la F_A para definir las direcciones izquierda, derecha y delante, tal y como se puede observar en la Figura 5.3.

- La etiqueta **delante** se asocia al intervalo $[90 - \alpha_f, 90 + \alpha_f]$ del domino de la F_A.

  α_f es el módulo del ángulo que forman el vector de dirección de movimiento del robot móvil y el vector que une el origen de coordenadas con el extremo superior izquierdo o derecho (indistintamente) del robot.

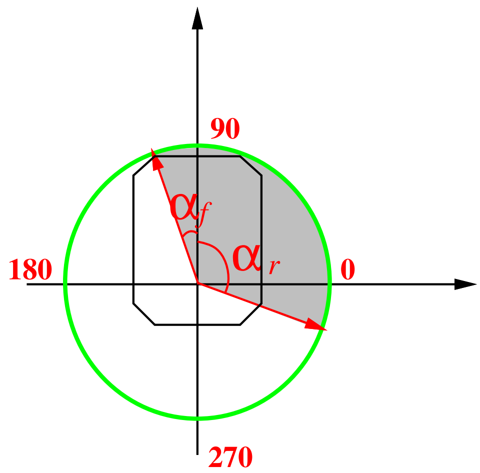


Figura 5.4: Etiqueta **Derecha**.

- La etiqueta **derecha** se asocia al intervalo $[90 - \alpha_r, 90 + \alpha_f]$ del dominio de la F_A.

  α_r es el módulo del ángulo que forman el vector que une el origen del sistema de referencia asociado al robot móvil y el extremo inferior derecho con el vector de movimiento (véase Figura 5.4).

- Análogamente, la etiqueta **izquierda** se asocia al intervalo $[90 - \alpha_f, 90 + \alpha_r]$ del dominio de la F_A.

5.2.3 Funciones de Control

El procedimiento de decisión de la acción de control a tomar por el agente *Avoid*, se puede esquematizar como sigue:

1. Comprobación de si el giro en el sentido de giro del deseo de entrada es posible.
2. En el caso en que esta acción de giro no sea posible, buscar el giro más cercano al deseado.
3. Asignación de la velocidad de giro y velocidad lineal, dependiendo de la cercanía de los obstáculos que se encuentren delante del robot.

Función de Supervivencia

La función *Sobrevive* tiene como misión evaluar la F_A dentro de la zona de seguridad y comprobar si el giro que se desea realizar es posible. *Sobrevive* comprobará si en la zona de giro se encuentra un obstáculo, que en términos de la $F_A(\theta_i)$ se traduce en que en el intervalo $\theta_i \in [\alpha_f, \alpha_r]$, la función tome un valor igual a "1.1".

Si la F_A no toma ningún valor igual a "1.1" en el rango $[\alpha_f, \alpha_r]$, la función *Sobrevive* asegura que se puede realizar un giro omnidireccional ese sentido (izquierda o derecha).

El Algoritmo 1 expresa en pseudo-código los valores que toman α_f y α_r y la forma en que se evalúa la F_A. Debido a las características geométricas del robot móvil, los valores de α_f y α_r son de 34 y 143 grados respectivamente. La F_A se evaluará en un rango comprendido entre $[AnguloInicial, AnguloFinal]$ (véase Algoritmo 1). Como ya se describió con anterioridad, la F_A está definida en un sistema de referencia solidario al robot móvil y, por tanto, a la hora de evaluarla, el $AnguloActual$ del Robot Móvil siempre será igual a 90 grados (ver Figura 5.1).

```
Boolean Sobrevive(Giro) {
   if (Giro == Derecha) {
       AnguloInicial = AnguloActualCoche+34;
       AnguloFinal   = AnguloActualCoche-143;
   } else {
       AnguloInicial = AnguloActualCoche-34;
       AnguloFinal   = AnguloActualCoche+143;
     }
   for (t=AnguloInicial; t != AnguloFinal; t++)
       if (FuncionAccesibilidad[t] == 1.1) return False;
   return True;
}
```

Algoritmo 1: *Sobrevive(Giro)*

Por lo general, se incrementa un poco el radio de la *zona de seguridad*, de tal forma que se pueda utilizar la función *Sobrevive* para evaluar si un giro puede realizarse en una dirección, aun no siendo omnidireccional.

Acciones de Control

El Algoritmo 2 muestra en pseudo-código las reglas de control para realizar la acción de evitar obstáculos.

El agente *Avoid* es una función que recibe un deseo de entrada (*Desired*) y tiene como salida una acción de control (*Output*). El valor máximo de la F_A en el intervalo etiquetado como **delante**, será la variable que establezca la acción de control a tomar por el agente *Avoid*:

```
max = Maximum(alfa_valley, -alfa_valley)
```

siendo `alfa_valley` el ángulo α_f (véase Figura 5.4).

Para controlar el giro y velocidad se han definido tres valores umbrales (*MinValue*,

```
Output Avoid (Desired) {
  max = Maximum(alfa_valley, -alfa_valley);

  if (max > MaxValue) {
    Output.speed    = -10;
    Output.steering = Desired.steering;
  } else {
    if (max > MinValue) {
      if (max > MediumValue) {
        if (Sobrevive(Desired.steering)) {
          Output.steering=Desired.steering;
          Output.speed = 0;
        } else {
          Output.steering=NILL;
          Output.speed = 10;
        }
      } else {
        if (Sobrevive(Desired.steering)) {
          Output.speed= 10;
          Output.steering = Desired.steering;
        } else {
          Output.steering=NILL;
          Output.speed = 10;
        }}}
    else {
      if (Sobrevive(Desired.steering))
        Output = Desired;
      else {
        Output.steering=NILL;
        Output.speed = 10;
      }} }
  if (max > 1.0) {
    Output.steering = NILL;
    Output.speed= -10;
  }}
```

Algoritmo 2: *Avoid*

$MediumValue$, $MaxValue$) del rango de la F_A, tal que:

$$MinValue < MediumValue < MaxValue$$

Dependiendo del intervalo en el que se encuentre el valor max, el giro y la velocidad que tomará el robot móvil viene definido por las siguientes reglas:

- $max < MinValue$: Si el robot puede realizar el giro "deseado", entonces la acción de giro será la misma que la especificada en el deseo de entrada. Si no se puede realizar el giro deseado, entonces se seguirá en línea recta, y con la velocidad que indica el deseo de entrada.
- $MinValue < max < MediumValue$: Si el robot puede realizar el giro "deseado", entonces la acción de giro será la misma que la especificada en el deseo de entrada, pero la velocidad del robot se verá reducida a $10cm/s$. En el caso en que no se pueda realizar el giro deseado, el robot irá en línea recta y con una velocidad de $10cm/s$.
- $MediumValue < max < MaxValue$: Si el robot puede realizar el giro deseado, entonces se realizará un giro omnidireccional en el sentido del giro deseado. Si no es así el caso, el robot irá en línea recta y con una velocidad de $10cm/s$.
- $max > MaxValue$: En este caso el robot dará marcha atrás mientras realiza un giro en la dirección deseada.
- Finalmente, si $max = 1.1$ indicará que un obstáculo se encuentra dentro de la zona de seguridad del robot móvil en su zona delantera. Por tanto, se dará marcha atrás.

5.3 Ir a un Punto (Goto Point)

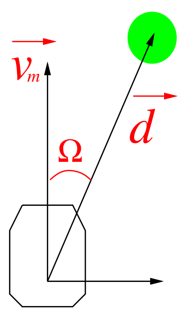


Figura 5.5: Ir a un Punto

El agente *Ir a un Punto* tiene como misión la de acercarse a un determinado punto del espacio, definido en coordenadas cartesianas como $P = (x, y)$. De forma cualitativa, las acciones de este agente se pueden describir de la siguiente manera:

"Si la meta se encuentra a la derecha del robot móvil, éste girará a la derecha. Si la meta se encuentra a la izquierda girará a la izquierda."

5.3.1 Control

El agente tratará de minimizar el ángulo $\Omega = \langle \vec{v_m}, \vec{d} \rangle$, donde $\vec{v_m}$ es el vector de dirección de robot móvil, y $\vec{d}$ es el vector que une el origen del sistema de referencia centrado en el robot móvil y el punto destino (ver Figura 5.5). El giro a realizar será de tal forma que se cumpla la condición:

$$\frac{d\Omega}{dt} < 0$$

Por tanto, la acción de giro a tomar, dependiendo del ángulo Ω, se resume en las siguientes reglas:

$$\begin{array}{lcl} \Omega & > & 0 \;\Rightarrow Giro = Derecha \\ \Omega & < & 0 \;\Rightarrow Giro = Izquierda \end{array} \tag{5.2}$$

La velocidad de movimiento del robot móvil será constante y podrá modificarse libremente (por el usuario o por otro agente). El agente considerará que ha llegado a su objetivo cuando la distancia que le separe de la meta esté por debajo de un valor umbral Γ determinado.

La velocidad del agente vendrá determinada por las reglas:

$$\begin{array}{lcl} |\vec{d}| & > & \Gamma \;\Rightarrow Velocidad = cte. \\ |\vec{d}| & \leq & \Gamma \;\Rightarrow Velocidad = 0 \end{array} \tag{5.3}$$

5.3.2 Entrada y Salida

El módulo ir a un punto recibirá como entrada las coordenadas cartesianas de un punto, $P = (x, y)$, y proporcionará un deseo de salida, $d_s = (velocidad, giro)$ que podrá alimentar a agentes de más bajo nivel.

El deseo de entrada podrá provenir de agentes de más alto nivel, como, por ejemplo, de los agentes motores de visión. En este caso, el sistema de estereovisión podrá determinar el punto $P = (x, y)$, en el que se encuentra el objetivo. El agente asociado a la visión deberá, por tanto, establecer su deseo de salida en términos de puntos cartesianos del espacio del tipo $p_i = (x_i, y_i)$.

En definitiva, Goto Point recibirá un vector de entrada, $d_e = (x, y)$, y mediante la evaluación del ángulo Ω dará como resultado un vector de salida, $d_s = (velocidad, giro)$. Por tanto, el agente de *Ir a un Punto* se puede definir como una función, GP:

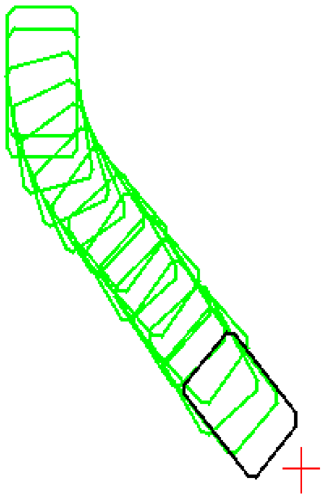


Figura 5.6: Ir a un Punto (Ejemplo Real)

Figura 5.7: Acciones de control y deseos de entrada y salida en **Goto Point**

$$
\begin{aligned}
d_s = \quad & GP(d_e) \\
d_e : \quad & (x, y) \\
d_s : \quad & (velocidad, giro)
\end{aligned}
\tag{5.4}
$$

5.4 Giro Propio

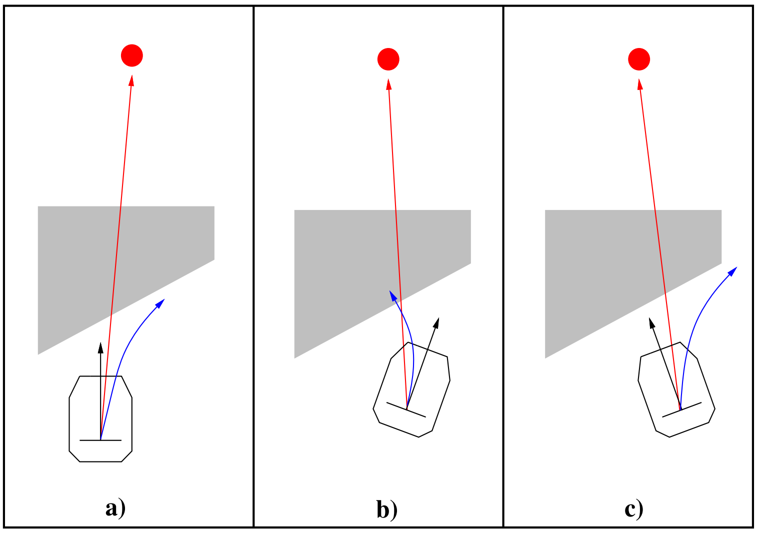


Figura 5.8: Dead Lock

En numerosas ocasiones puede ocurrir lo que se muestra de forma gráfica en la Figura 5.8: Dos módulos independientes pueden interferir de tal forma que el robot se quede "estancado" en una zona y no pueda salir de ella. En el ejemplo mostrado en la Figura 5.8, podemos observar como el robot se dirige a su objetivo (marcado por un círculo). En un primer momento (a) el objetivo se encuentra a la derecha del robot. Por tanto, *Avoid* cumpliendo los deseos del agente *Goto Point* evita el obstáculo girando a la derecha. Sin embargo, y tal como se observa en (b), tras unos breves instantes, el objetivo se encuentra a la izquierda del robot. Por tanto, el robot debería girar a la izquierda. Este ciclo se repite indefinidamente (movimientos de izquierda a derecha) sin que el robot consiga avanzar. Para evitar este problema, conocido en la literatura como *dead-lock*, y muy frecuente en los sistemas reactivos, el agente *Giro Propio* incorpora una función que, evaluando la orientación relativa entre el obstáculo y el robot, estima cómo se ha de evitar un obstáculo. La finalidad de este módulo es la de establecer una estrategia que decida por qué lado se ha de *bordear* el obstáculo, procurando minimizar el giro a realizar para alinearse con el mismo.

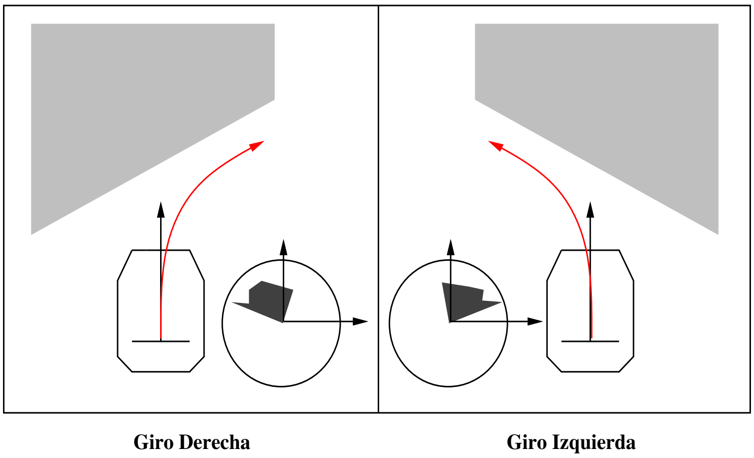


Figura 5.9: Evaluación del Giro: Se puede observar el robot móvil ante el obstáculo y a su lado la F_A calculada en ese momento. El giro a realizar está indicado con una flecha.

Intervalo del Dominio	**Etiqueta**
$\theta_i \in [90, 180]$	Izquierda
$\theta_i \in [0, 90]$	Derecha

Tabla 5.2: Asignación de etiquetas a intervalos de la F_A por el agente *Giro Propio*

5.4.1 Control

Mediante el cálculo del área que encierra la F_A en diversos intervalos, resulta muy fácil comprobar en que orientación se encuentra el robot respecto a un obstáculo que le esté obstruyendo el paso hacia la meta.

El área que engloba la F_A en un intervalo $[\theta_i, \theta_f]$ se define como:

$$Area(\theta_i, \theta_f) = \sum_{i=\theta_i}^{\theta_f} F_A(i) \tag{5.5}$$

Luego, las áreas designadas con las etiquetas izquierda y derecha (tal y como está expresado en la Tabla 5.2) se calculan como:

$$Area(izquierda) = \sum_{i=90}^{180} F_A(i)$$

$$Area(derecha) = \sum_{i=0}^{90} F_A(i)$$

Comparando el área que encierra la F_A a la izquierda y a la derecha del robot, se puede establecer una simple regla de control: Si el área que engloba la F_A a la izquierda del robot es mayor que el área a la derecha, el obstáculo está más cercano al lado izquierdo del robot, debiéndose realizar un giro a la derecha. Por tanto, el giro realizado para alinearse con el obstáculo ha sido menor que si se hubiese decidido girar hacia el lado contrario (izquierda). En la Figura 5.9 se puede observar la acción de control tomada en cada caso. Las reglas que se acaban de definir están expresadas, en pseudo-código, como:

```
if (Area(Izquierda) >= Area(Derecha))
   Giro = Derecha;
if (Area(Izquierda) < Area(Derecha))
   Giro = Izquierda;
if (Area(Izquierda)== 0 && Area(Derecha)==0)
   Giro = UltimoGiroRealizado;
```

Algoritmo 3: Propio Giro

5.4.2 Entrada y Salida

Siguiendo la filosofía empleada en los agentes de navegación del robot móvil, *Giro Propio* aceptará deseos de entrada y proporcionará un deseo de salida dependiente del contexto. Este

Figura 5.10: Acciones de control y deseos de entrada y salida en **Giro Propio**

agente solo proporcionará un deseo de salida diferente al deseo de entrada cuando perciba, mediante la evaluación de la F_A, que el robot se encuentra muy próximo a un obstáculo. Evaluando la Función de Accesibilidad del intervalo *delante* (ver Figura 5.3), se puede establecer que, si sobrepasa un determinado umbral, τ, el robot se encuentra cerca de un obstáculo. Es decir, el valor de τ indicará el valor de la F_A a partir del cual el agente *Giro Propio* toma el control del robot.

En definitiva, *Giro Propio* recibirá un vector de entrada, $d_e = (velocidad, giro)$, y mediante la evaluación de la F_A dará como resultado un vector de salida, $d_s = (velocidad, giro)$. Por tanto, el módulo *Giro Propio* se puede definir como una función G:

$$\begin{aligned} &d_s = G(d_e) \\ &d_s, d_e : (\text{velocidad, giro}) \end{aligned} \tag{5.6}$$

5.5 Resultados Experimentales

En esta sección se mostrarán los resultados experimentales relativos a los comportamientos que se han detallado y expuesto en las anteriores secciones, como los agentes *Avoid*, *Giro Propio* y *Goto Point*.

Las figuras que representan los resultados experimentales muestran la trayectoria seguida por el robot sobre la representación del entorno proporcionada por el mapa de acumulación. Tal y como ya se ha detallado en secciones anteriores, los comportamientos motores de navegación reaccionan a la información sensorial suministrada por la *Función de Accesibilidad* y no realizan ningún tipo de planificación sobre el modelo del mundo generado por el mapa de acumulación. Las figuras se han mostrado en este formato para que el lector tenga como referencia los obstáculos con que se encontró el robot a la hora de realizar su tarea.

En la Figura 5.11 se puede observar la trayectoria trazada por el robot móvil cuando trataba de llegar a un punto del espacio evitando los obstáculos que hubiese entre él y el objetivo. Los comportamientos que estaban activos en esta situación eran *Avoid* y *Goto Point*, con lo que el deseo resultante de la interacción entre ambos se podría expresar como:

$$d_s = A(GP(d_e))$$

Es decir, en este caso no estaba activo el agente *Giro Propio*, que asegura que el robot no se va a quedar en una situación de dead-lock. El resultado de la interacción entre ambos

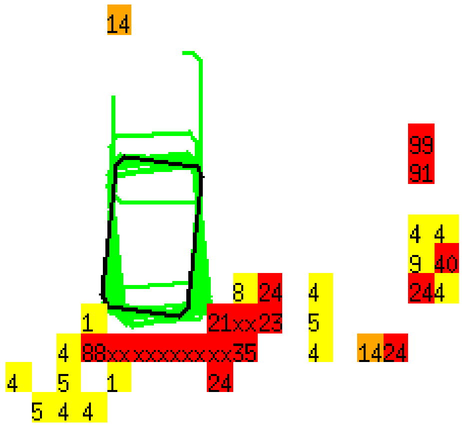


Figura 5.11: Dead Lock

comportamientos queda claramente plasmado en la Figura 5.11, donde se puede observar cómo el robot móvil se queda prácticamente estático en una determinada posición, cabeceando a derecha e izquierda sin ser capaz de moverse del sitio. Los deseos del comportamiento *Goto Point* se ven anulados por la presencia de un obstáculo en la trayectoria y de la acción de control tomada por el agente *Avoid*.

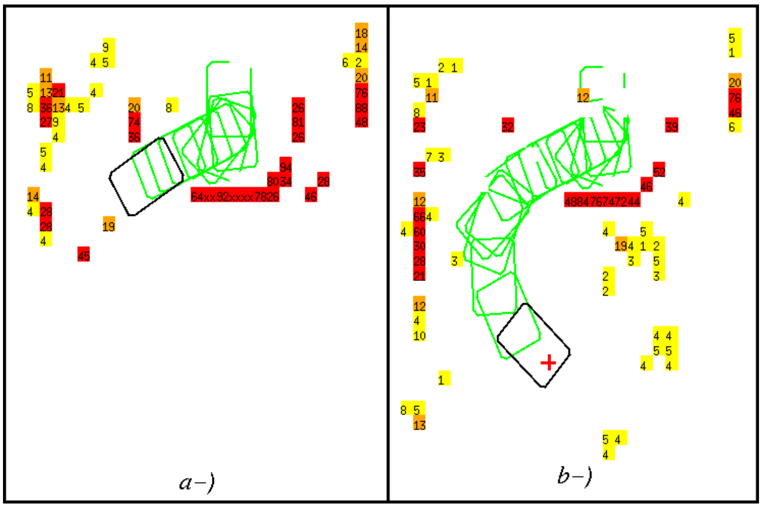


Figura 5.12: Con agente *Giro Propio*

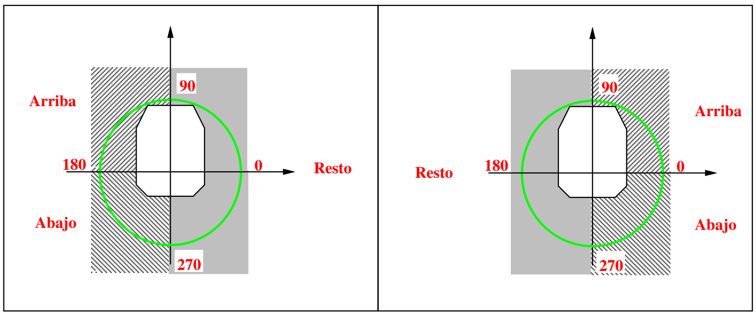
Sin embargo, y mediante la modulación de los deseos de control por parte del agente *Giro Propio*, la acción de control se puede expresar como:

$$d_s = A(G(GP(d_e)))$$

En la Figura 5.12 se puede observar cómo el sistema es capaz de bordear el obstáculo y llegar a la meta. El agente *Giro Propio* actuó en el momento de enfrentarse el robot al obstáculo de tal manera que su acción de control hizo que el robot alinease su lado izquierdo con el obstáculo. Una vez que el obstáculo se encontraba del lado izquierdo del robot, éste comenzó a acercarse a la meta bordeando el obstáculo (ver Figura 5.12-a). En cuanto el camino entre la meta y el robot quedó expedito, y ningún obstáculo se encontraba entre la meta y el robot, la trayectoria que siguió el sistema fue directa hacia el objetivo, tal y como se puede observar en la Figura 5.12-b.

5.6 Seguir Paredes (Follow Wall)

5.6.1 Direcciones



Izquierda **Derecha**

Figura 5.13: Direcciones: Las zonas sombreadas indican el intervalo de valores, θ_i, para cada una de las etiquetas lingüísticas (arriba, abajo, resto), empleadas por el agente de seguir paredes (Follow Wall) en cada una de las posibles alineaciones con la pared (Izquierda, Derecha).

Al igual que el resto de comportamientos motores de navegación, el comportamiento de seguir paredes asigna una serie de etiquetas a diversos intervalos de la F_A (véase la Figura 5.13). Estos intervalos variarán en función del lado por el que el robot se vaya a alinear con la pared.

5.6.2 Control

Una vez definidas las etiquetas que se asignan, el control de este comportamiento se basa en la comparación de las áreas que engloba la F_A en los diversos intervalos (arriba, abajo, resto).

Por ejemplo, en el caso en que el robot deba alinearse con la pared por la izquierda, las áreas de los intervalos etiquetados como arriba, abajo y resto se calculan como:

$$Area(arriba) = \sum_{i=90}^{180} F_A(i)$$

$$Area(abajo) = \sum_{i=180}^{270} F_A(i)$$

$$Area(resto) = \sum_{i=270}^{360} F_A(i) + \sum_{i=0}^{90} F_A(i)$$

Control del Giro

Si el objetivo es que el robot siga una pared alineándose por la derecha, entonces se comprobará si el área de la F_A *arriba* es mayor que el área *abajo*. Si éste es el caso, el control tratará de que el robot gire hacia la izquierda. Intuitivamente, se puede ver fácilmente que en este caso la parte superior izquierda del robot se encuentra más cerca de la pared, por tanto, tratará de separase ligeramente de ella. En caso contrario, el agente tratará que el robot gire hacia la derecha, es decir, que se acerque a la pared.

```
SeguirPared(Alinear) {
    if (Alinear == Derecha) {
       if (Area(Arriba) > Area(Abajo))
          Giro = Izquierda;
       else
          Giro = Derecha;}
    if (Alinear == Izquierda) {
       if (Area(Arriba) > Area(Abajo))
          Giro = Derecha;
       else
          Giro = Izquierda;}
}
```

Algoritmo 4: *Seguir Pared*

Buscar Pared

Puede darse el caso de que en los alrededores del robot no se encuentre ninguna pared, o incluso que la pared se encuentre del lado contrario a aquél por el que se desea seguirla. Debido a esto, el agente de seguir paredes evaluará el área que se encuentra del lado contrario del que se desea seguir (expresado como *resto* en la Figura 5.13). Así, si esta área es mayor que la suma de las áreas de los intervalos denominados como *arriba* y *abajo*, el agente tratará de "buscar" la pared. Buscar la pared simplemente consistirá en girar en la dirección contraria a aquella con la que se desea que se alinee el robot para seguir la pared. Por tanto, se consigue una gran flexibilidad con este comportamiento, ya que independientemente de la orientación en la que se encuentre el robot respecto de la pared que se quiera seguir, el agente orientará automáticamente al robot en la dirección adecuada.

Control de la Velocidad

El comportamiento de seguir paredes intenta mantener siempre una velocidad lineal constante. Sin embargo, y de forma parecida a como actúa el agente *Avoid*, la velocidad será función de la F_A evaluada "delante" del robot.

El rango de la F_A se dividirá en tres intervalos, definidos por los valores d_{min}, d_{med} y d_{max}, tales que:

$$d_{max} > d_{med} > d_{min}$$

La Tabla 5.3. muestra la correspondencia entre el valor máximo (Max) de la F_A en el intervalo etiquetado como delante[†], y la velocidad.

Max=Máximo(F_A(Delante))	**Velocidad**
$Max < d_{min}$	Deseo de Entrada
$d_{min} < Max < d_{med}$	velocidad = 10
$d_{med} < Max < d_{max}$	velocidad = 0
$d_{max} < Max \leq 1.1$	velocidad = -10

Tabla 5.3: Correspondencia entre el máximo valor de la F_A, evaluada en la dirección frontal del robot (delante), y la velocidad.

La condición de que el robot no choque contra ningún obstáculo vendrá asegurada por la acción que el agente *Avoid* realice sobre el deseo de salida del agente *Follow Wall*. En principio no será necesario que *Avoid* module el deseo de *Follow Wall*, ya que la acción de seguir una pared viene a ser como la de evitar un obstáculo bordeándolo. Sin embargo, situaciones en las que el robot se pueda encontrar en un entorno complicado, como una esquina convexa o multitud de obstáculos, como sillas, mesas, etc. pueden obligar al comportamiento de evitar obstáculos a modular el deseo de salida del agente de seguir paredes para garantizar la seguridad del robot. La forma en la que está definida y generada la arquitectura de control asegura que la acción del comportamiento de evitar obstáculos no va a deshacer las acciones del comportamiento de seguir paredes.

5.6.3 Entrada y Salida

El agente de seguir paredes (*Follow Wall*) recibirá un vector de entrada, $d_e = (velocidad, giro)$, y dará como resultado un vector de salida, $d_s = (velocidad, giro)$. Por tanto, el módulo *Follow Wall* se puede definir como una función FW:

$$\begin{aligned} & d_s = FW(d_e) \\ & d_s, d_e : \text{(velocidad, giro)} \end{aligned} \tag{5.7}$$

[†]Véase la Figura 5.3.

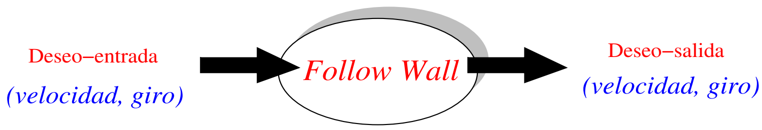


Figura 5.14: Acciones de control y deseos de entrada y salida en el agente de seguir paredes (**Follow Wall**)

El vector de entrada es del tipo $(velocidad, giro)$ por motivos de compatibilidad con el resto de los agentes motores de navegación. En efecto, el objetivo del comportamiento de seguir paredes es alinearse con una pared a partir de la información suministrada por los módulos sensoriales (en este caso la F_A). La acción de control será generalmente una acción de giro a realizar dependiendo de esta información. Por tanto, no se tendrá en cuenta el deseo de "giro" de comportamientos de más alto nivel, ya que la acción de giro de este comportamiento sólo tiene sentido dependiendo de cuál sea la relación espacial del robot con su entorno.

5.6.4 Resultados Experimentales

Como ya se ha indicado en la sección 5.6.2, la acción de control resultante de seguir una pared es debida a la actuación del agente *Follow Wall* y la modulación de su deseo de salida por el comportamiento *Avoid*. Así, el deseo de salida resultante viene determinado por la composición:

$$d_s = A(FW(d_e))$$

En la Figura 5.15 se puede observar la trayectoria "real" que ha seguido el robot cuando estaba activo el comportamiento *Follow Wall*. El término "real" lo utilizamos para expresar el hecho de que la trayectoria que se muestra en la figura ha sido obtenida de las mediciones realizadas por los sensores odométricos. La trayectoria está enmarcada en cuatro zonas bien diferenciadas (1,2,3 y 4), mostrándose las imágenes de los diferentes lugares por donde pasó el robot.

En la zona marcada como "1", podemos observar cómo el robot realizó un giro a la izquierda para encontrar la pared que tenía que seguir alineándose con ella por la derecha. Tal y como se puede observar en la imagen de esta zona, el elemento del entorno a seguir no era una pared propiamente dicha, sino un conjunto de cajas. Las zonas marcadas como "2" y "3" tampoco son "paredes", sino un conjunto de elementos del entorno, como mesas sillas e incluso personas que se encontraban trabajando en el momento de realizar las pruebas experimentales. Las zonas "2" y "3" demuestran claramente como el robot trata de seguir las "paredes" en zonas convexas, adaptándose a la forma del entorno. La zona etiquetada como "4" es un claro ejemplo del seguimiento de una pared.

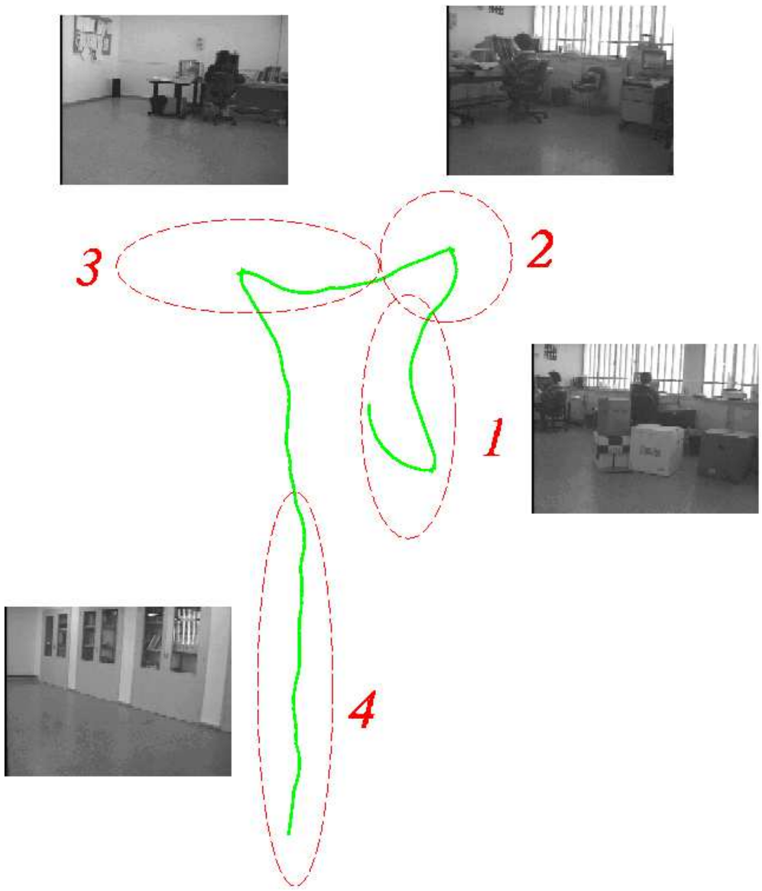


Figura 5.15: Resultados Experimentales.

5.7 Ir a un Punto Visual (Vision Goto Point)

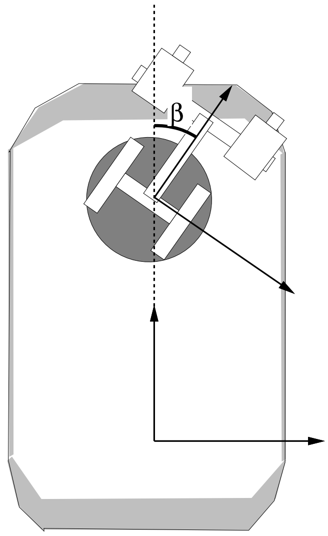


Figura 5.16: Ángulo β entre el sistema de referencia de la torreta y la plataforma del robot móvil.

Tal y como se detallará en el capítulo dedicado a los agentes motores visuales, el sistema de visión del robot será capaz de detectar y seguir toda una serie de elementos visuales del entorno. El sistema de seguimiento visual recibirá su señal sensorial de las cámaras que van montadas sobre la torreta del robot, y actuarán sobre los motores de la misma. Los agentes visuales sólo actuarán sobre los controles de la torreta y nunca (directamente) sobre los del robot móvil.

Por tanto, es necesario definir y crear un agente que realice la tarea de interfaz entre los agentes visuales y los de navegación. Es decir, un agente capaz de comprender los "deseos" de los agentes visuales y traducirlos en deseos que muevan a la plataforma del robot móvil. El agente de ir a una localización visual (*Vision Goto Point*) realizará esta tarea partiendo de la premisa de que el sistema de visión está realmente siguiendo un objeto. Si tal es el caso, las cámaras estarán siempre apuntando al objeto que están siguiendo, ya que los comportamientos de detección y seguimiento de un objeto tratarán en todo momento, y mediante la actuación sobre los controladores de la torreta, de tener el objetivo visual en el centro de la imagen. De tal manera, y mediante una comunicación a traves del entorno, el agente motor *Vision Goto Point* puede interpretar el deseo de los agentes visuales de seguimiento mediante la observación del ángulo de giro β de la torreta.

Para comprender el funcionamiento del agente cuya tarea es la de dirigirse a una localización visual (*Vision Goto Point*), es necesario conocer diversos detalles de la implementación física de los elementos del robot, y más en concreto los que conciernen a la torreta sobre la que va montado el sistema de visión. En este sentido cabe destacar que la torreta del robot móvil posee dos motores independientes que le permite realizar movimientos de inclinación y azimutales. Sin embargo, en el presente trabajo se ha reducido la movilidad de la torreta a únicamente movimientos azimutales.

La torreta tiene un sistema de referencia independiente del robot móvil (ver Figura 5.16). Para conocer en cada instante el ángulo β de rotación entre el sistema de referencia asociado a la torreta y el sistema de referencia asociado al robot, la torreta posee un sistema de posicionamiento (encoders). Este sistema de posicionamiento tiene una resolución de $1.4 \cdot 10^{-2}$ grados, con un rango de movimientos comprendidos entre los -164.79 y los 164.79 grados.

5.7.1 Control

Al igual que en el caso de el comportamiento *Goto Point*, cuyo objetivo es ir a una localización (x,y) del espacio, el agente Vision Goto Point tratará de minimizar el ángulo β, que forma la torreta con el eje de referencia del robot móvil (ver Figura 5.16). El giro a realizar será de tal forma que se cumpla la condición:

$$\frac{d\beta}{dt} < 0$$

Por tanto, la acción de giro a tomar, dependiendo del ángulo β, se resume en las siguientes reglas:

$$\begin{array}{ccll} \beta & > & 0 & \Rightarrow Giro = Derecha \\ \beta & < & 0 & \Rightarrow Giro = Izquierda \end{array} \tag{5.8}$$

La velocidad de movimiento del robot móvil será constante y podrá modificarse libremente (por el usuario o por otro agente). El propio agente nunca podrá considerar que ha llegado al objetivo, ya que carece por completo de la información relativa a la distancia a la que se encuentra del mismo. Como se verá, otros agentes serán los encargados de determinar cuando se ha llegado al objetivo. El comportamiento *Vision Goto Point* actúa de una forma refleja y asegura que el robot móvil se mueva en la dirección en la que se encuentra el objetivo visual, dando la oportunidad a otros agentes de calcular la distancia a la que se encuentra el robot de la meta.

5.7.2 Entrada y Salida

El agente *Vision Goto Point* recibirá un valor escalar de entrada, $d_e = \beta$, y dará como resultado un vector de salida, $d_s = (velocidad, giro)$. Por tanto, el módulo *Vision Goto Point* se puede definir como una función VGP:

$$\begin{array}{l} d_s = VGP(d_e) \\ d_e : \mathrm{R} \\ d_s : (\mathrm{velocidad,\ giro}) \end{array} \tag{5.9}$$

Figura 5.17: Acciones de control y deseos de entrada y salida en el agente **Vision Goto Point**

5.8 Agente Actuador

La arquitectura de control del robot móvil está definida por una serie de comportamientos jerarquizados, donde cada uno de ellos emite un deseo de control que será modulado por los comportamientos de más bajo nivel. Los deseos de entrada y salida de los comportamientos motores de navegación siguen una sintaxis y una semántica propia que entienden e interpretan los comportamientos mismos. Sin embargo, la sintaxis de estos deseos no es la apropiada para que los controladores de la plataforma móvil (Robuter) sean capaces de interpretarlos correctamente. Por tanto, hace falta un comportamiento que sirva de interfaz entre los comportamientos motores y los actuadores del robot móvil. Esta es precisamente la labor que tiene encomendado el agente *Actuador*: Interpretará los deseos de los agentes motores de navegación y proporcionará una acción de control que el robot sea capaz de comprender.

5.8.1 Control en Velocidad

La plataforma móvil es capaz de aceptar comandos de movimiento tanto en control en posición como en velocidad. Cuando se especifica una acción de control en posición, el comando de control debe especificar a la posición (absoluta o relativa a la posición actual) a la que debe llegar el robot. Una vez que la plataforma haya llegado a su destino se podrá volver a mandar una nueva acción de control para especificar una nueva coordenada espacial a la que deberá dirigirse el robot. Sin embargo, durante la ejecución de una acción de control en posición, ésta no se podrá abortar ni cambiar por otra. Este tipo de control del robot móvil no permite una reacción en tiempo real a cambios en el entorno del robot, resultando por tanto una forma de controlarlo excesivamente arriesgada.

Mediante el control en velocidad del robot móvil, se puede especificar en cada instante la velocidad que deberá tener cada una de las dos ruedas motrices con las que está equipada la plataforma móvil. La velocidad de las ruedas, especificada en el comando de control, se mantendrá fija hasta que se reciba una nueva acción de control que modifique la anterior. Por tanto, este tipo de control resulta ser el más adecuado para asegurar la reactividad del sistema a cambios en el entorno y no depender del lapso de tiempo que transcurre entre que se manda una acción y se cumple la misma. Sin embargo, el inconveniente que presenta este tipo de control del robot móvil es la resolución que permiten los controladores de las ruedas motrices en el modo de control en velocidad, que es a saltos de $\pm 10 cm/s$.

En definitiva, el agente *Actuador* recibirá deseos de la forma $d_e = (velocidad, giro)$ y mandará una acción de control a la plataforma del robot móvil mediante el comando:

$$MOVE\ VAC = v_i, v_d$$

donde v_i y v_d indican la velocidad (en cm/s) que deberán tomar las ruedas motrices izquierda y derecha, respectivamente.

La transformación entre la velocidad y el giro del deseo de entrada del agente *Actuador*, y la velocidad que se transmitirá a las ruedas motrices vendrá determinada por:

$$\begin{aligned} &v_d = \text{velocidad};\ v_i = \text{velocidad}; \\ &\forall\ \text{giro} > 0,\ v_i = v_d + \text{giro}; \\ &\forall\ \text{giro} < 0,\ v_d = v_i - \text{giro}; \end{aligned} \tag{5.10}$$

A parte de realizar el interfaz entre los agentes motores de navegación y los controladores de bajo nivel de la plataforma móvil, este agente también tratará de realizar funciones de control suaves. De esa forma se evita que el robot se mueva a "trompicones" y la transición de velocidades y giros se realice de una forma suave.

5.9 Estructura del Sistema

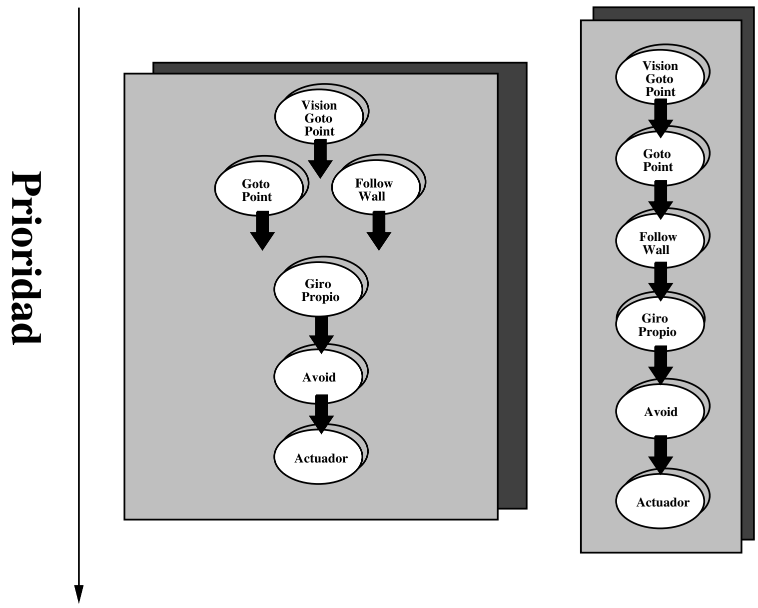


Figura 5.18: Jerarquía de los agentes

La Figura 5.18 muestra la jerarquía de los agentes atendiendo a su prioridad dentro del sistema. En el nivel de prioridad más alto, se encuentra el comportamiento de evitar obstáculos, que garantiza la supervivencia del robot. El agente *Giro Propio* le sigue en el orden jerárquico dentro del sistema.

Los agentes *Goto Point* y *Follow Wall* se encuentran en un mismo nivel dentro de la jerarquía, y por tanto son incompatibles:

$$GP \odot FW$$

A priori, no existe ninguna razón por la cual uno de estos dos comportamientos deba tener prioridad sobre el otro. Es decir, sin conocer la tarea que realizar por el robot, ni el estado del entorno, es imposible decidir cuál de los dos agentes ha de tomar el control. Ambos comportamientos son excluyentes, pues no tiene "mucho sentido" dirigirse a un determinado punto del espacio mientras se están bordeando las paredes.

La estructura lineal de la arquitectura **MAgDA**, requiere que todos los agentes se encuentren en niveles de prioridad diferentes. Por tanto, se ha decidido otorgar una prioridad mayor al agente *Follow Wall* que al agente *Goto Point*. A efectos prácticos esto no tiene ninguna repercusión en el comportamiento global del sistema, ya que ambos comportamientos no estarán activos a la vez en el mismo instante de tiempo.

Sin embargo, la decisión de otorgar al agente *Goto Point* una prioridad menor tiene otro motivo: El agente *Vision Goto Point* estará únicamente activo durante el periodo de tiempo en que se esté siguiendo un objeto por el sistema de seguimiento visual, y el sistema de estereovisión no haya calculado la localización del objetivo. Durante este periodo de tiempo, el agente *Goto Point* no conocerá la posición (x, y) a la cual debe dirigirse, periodo durante el cual el sistema está gobernado por las acciones de control del agente *Vision Goto Point*. Este agente estará siempre activo, pero sus salidas serán "inhibidas" por el agente *Goto Point* en cuanto éste conozca la localización del objetivo.

Finalmente, la jerarquía del sistema vendrá dada por la relación:

$$P(A) > P(G) > P(FW) > P(GP) > P(VGP)$$

Y la salida de control de los agentes de navegación se puede expresar como la composición de funciones:

$$Salida = A(G(FW(GP(VGP))))$$

Capítulo 6

Comportamientos Motores de Visión

6.1 Introducción

En los últimos años se ha producido, en el ámbito de la visión artificial, una tendencia a considerar el sistema visual en el contexto del comportamiento de un robot interaccionando con su entorno. Esta nueva filosofía en el concepto y utilización de la visión artificial como recurso para la acción del robot se ha denominado *visión activa**. En este sentido, *Ballard* [8] considera que la percepción visual es uno de los muchos comportamientos utilizados por un robot para lograr sus objetivos en un entorno dinámico. Por su parte, *Aloimonos* [3] hace un especial énfasis del carácter cualitativo y de propósito del sistema visual. Es decir, la *visión activa* representa un claro giro en la concepción de los procesos tradicionales de la visión artificial en dos sentidos:

- La operación activa de los procesos visuales en el entorno, de forma que modifiquen las imágenes que se están captando, para optimizar la eficiencia de la tarea.
- El propósito para el cual se quiere construir un sistema visual reduce el tiempo de cálculo y aumenta la velocidad de respuesta del sistema.

Por tanto, el sistema visual de un robot no debe generar complejas representaciones del entorno, sino que debe explotar las características del mismo y tener en cuenta únicamente

*No se debe confundir el término de *visión activa* con el que tradicionalmente se utilizó para designar a aquellos sistemas que iluminaban la escena con un patrón de luz determinado.

aquellas aptitudes que le ayuden a realizar la tarea encomendada. Como ejemplo, *Ian Horswill* [52] implementó un robot móvil dotado de un sistema visual activo, que permitía evitar obstáculos sin la necesidad de hacer una reconstrucción simbólica de lo percibido. El sistema se basaba en explotar las características del entorno de trabajo del robot. Así, en vez de realizar una reconstrucción tridimensional de la escena, el robot consideraba como obstáculos aquellas zonas donde hubiese "textura", ya que el suelo por el que se movía el robot era una moqueta sin ningún tipo de textura.

El sistema de seguimiento visual implementado en el presente trabajo, está concebido para seguir un conjunto de balizas de colores. En el desarrollo del presente capítulo se hace un especial énfasis en la etapa de atención previa del sistema, donde se extraen las características más sobresalientes de la imagen. A su vez, el sistema está concebido como un conjunto de agentes que cooperan para buscar, detectar y seguir el objeto visual. En las siguientes secciones se desarrollarán los conceptos y la implementación del sistema.

6.2 Atención Previa

El guiado de la atención visual en escenas reales ha sido el objeto de estudio de numerosos trabajos [89] [90] [91]. Estos sistemas dirigen su atención a elementos sobresalientes de la escena, pero no son parte de un agente que utilice la visión para realizar una tarea determinada. En su lugar, estos sistemas imitan a un observador estático cuyos ojos se fijan en partes de la escena sin una razón particular. En estos trabajos, los elementos sobresalientes se definen como elementos que tienen alguna propiedad única, como la luminosidad, bordes con una curvatura grande, colores determinados, e incluso la combinación de una serie de ellos. Experimentos de orden psicológico [89] demuestran que el tiempo requerido para detectar la presencia de un elemento de la imagen es constante si hay una única característica que se pueda utilizar para diferenciarla del resto de elementos. En este tipo de escenas se dice que el elemento "resalta". En la Figura 6.1, se puede observar cómo la línea vertical de mayor grosor resalta sobre las demás. El grosor de la línea diferencia al objetivo del resto de los elementos que distraen la atención.

Numerosos sistemas extraen características (líneas, colores, texturas, etc.) de las imágenes para guiar la atención del sistema. La etapa en la que se extraen estas características se define como fase de *atención previa*†, en la que se genera una serie de *mapas de características*. Estos mapas de características se crean en paralelo sobre toda la imagen, y generalmente se ha considerado la creación de estos mapas como un proceso de abajo a arriba, por lo que la extracción de las características de la imagen no está influida por el conocimiento del agente y su objetivo.

Por otra parte, es muy importante determinar cómo la atención puede ser guiada para ayudar al agente a alcanzar sus metas. Este problema fue estudiado por *Chapman* [28], mediante la implementación de un agente cuyo sistema visual operaba en escenas bidimensionales de un juego de vídeo. El agente utilizaba la atención para detectar objetos importantes, atacar a sus

†Del ingles **pre-attentive**.

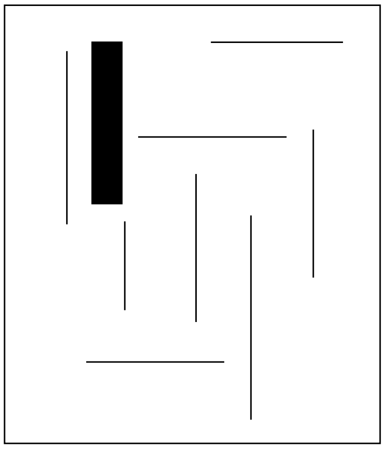


Figura 6.1: La linea vertical resalta por su grosor.

enemigos, evitar ser matado y salir de situaciones peligrosas. Sin embargo, el sistema de *Chapman* obvió por completo todos los problemas que suscita operar con imágenes "reales", a pesar de lo cual proporcionó numerosas respuestas a aspectos abstractos de cómo se seleccionaban las características en la imagen para guiar la atención.

Finalmente, en la fase de atención previa, no es necesario generar un mapa de características de la imagen excesivamente prolijo. Por ejemplo, una información que indique "groso modo" si un elemento de la imagen es más rojo que azul y verde puede ser suficiente para localizar el objetivo buscado. Un procesamiento posterior de los diferentes elementos del mapa de características que hayan cumplido unas ciertas condiciones reduce en gran medida el costo computacional final de la búsqueda del objetivo.

6.2.1 Segmentación en Color

Los elementos del entorno que han de ser detectados por el sistema visual determinan las características básicas que se van a extraer de las imágenes en la fase de atención previa. Las balizas visuales están compuestas por una composición de colores que no se dan habitualmente en entornos interiores. Es decir, en lugar de tratar de explotar las características del entorno [52] se ha decidido modelarlo para nuestros intereses. A pesar de modificar el entorno para que sea más "amigable" para el robot, no se pierde ninguna generalidad en cuanto a la validez del sistema visual propuesto. El mismo sistema básico se ha utilizado satisfactoriamente para detectar hitos naturales, como sillas, monitores y personas dentro de un laboratorio, generando diferentes mapas de características (movimiento, color, texturas, etc.) como elementos de atención previa.

El color ha resultado ser una gran fuente de información para procesar e interpretar la información visual. La utilización del color permite encontrar elementos que resalten en las imágenes [8] [68], que reducen el espacio de búsqueda del sistema, o sistemas que traten de detectar el objeto de interés. Sin embargo, los numerosos algoritmos propuestos en la literatura [51] [58] [83] requieren un elevado tiempo de cálculo y son poco robustos a los cambios de iluminación en el entorno.

En un primer paso, únicamente resulta necesario extraer una información cualitativa del componente de rojo, azul y verde de cada uno de los pixels de la imagen. El sistema de adquisición de imágenes permite que por cada pixel se codifique un valor RGB, que indica la componente roja(R), verde(G) y azul(B). Aplicando un predicado lógico a cada pixel, tal y como se expresa en la Tabla 6.1, se obtiene una representación en la que cada pixel codifica la componente principal de su color.

Condición	**Codificación**
$R > G, R > B$	Rojo
$G > R, G > B$	Verde
$B > R, B > G$	Azul
else	Desconocido

Tabla 6.1: Codificación del Color.

Las dos principales ventajas de este tipo de segmentación de la imagen son la velocidad de cálculo y la robustez ante cambios en la iluminación de la escena. Esta última viene impuesta por el amplio abanico de colores que abarca cada una de las codificaciones. Sin embargo, con este método es imposible discernir si un determinado pixel es de color naranja o rojo bermellón, pero la aplicación tampoco lo requiere. La composición de las balizas (alternancia de colores rojo, verde y azul) no requiere un conocimiento exhaustivo de las componentes de la luminancia, crominancia y saturación de cada uno de los pixels de la imagen.

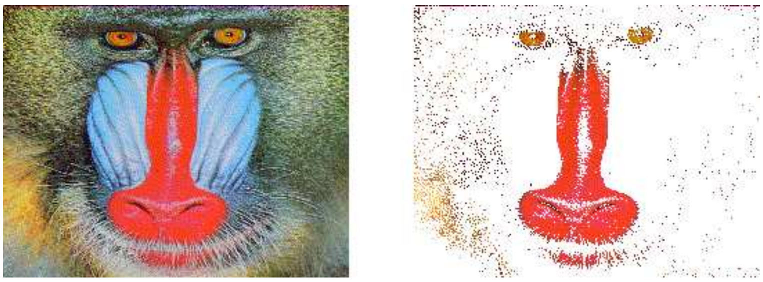


Figura 6.2: Segmentación de la nariz y los ojos del mandril.

A pesar de su sencillez, este método puede utilizarse en numerosas aplicaciones. Por ejemplo, en la Figura 6.2 se puede observar la imagen de un mandril, en la que mediante la utilización de técnicas tradicionales de visión artificial basadas en imágenes de niveles de gris resultaría complejo segmentar los ojos y la nariz del resto. Sin embargo, mediante la utilización del color, y aplicando un predicado lógico como:

$$P(x,y) = \begin{cases} R, & R > G,\, R > B; \\ 0, & resto; \end{cases}$$

se pueden resaltar tanto los ojos como la nariz. Estos rasgos de la imagen son fácilmente identificables mediante una técnica basada en el color. Sin embargo, en la Figura 6.2 se puede observar cómo ciertos elementos de la imagen también han pasado el filtro inicial del predicado lógico; Un nuevo procesamiento sobre los ellos, como, por ejemplo, un agrupamiento de los pixels en regiones con características similares, proporcionaría una nueve fuente de información (como el área y el perímetro de la región conectada) para poder determinar de qué tipo de elemento se trata (ojo, nariz, desconocido). Es decir, una extracción jerarquizada de características basadas en un algoritmo de segmentación sencillo de colores, permite reducir el espacio de búsqueda del elemento de interés de una manera sustancial.

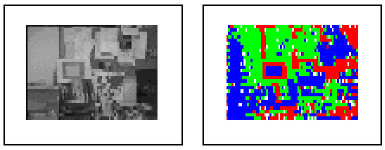


Figura 6.3: Imagen en niveles de gris y su correspondiente segmentación basada en el color.

En la parte izquierda de la Figura 6.3, se puede observar la imagen en niveles de gris del entorno en que se debía mover el robot. En esta imagen resulta difícil detectar y reconocer la baliza compuesta por un cuadrado azul dentro de uno rojo. A la derecha de esta imagen se puede observar el resultado de la segmentación de la imagen basada en la codificación expresada en la Tabla 6.1. Por ejemplo, en azul aparecen los pixels cuya componente de azul fue mayor que la de rojo y verde, mientras que en blanco aparecen los pixels etiquetados como *resto*.

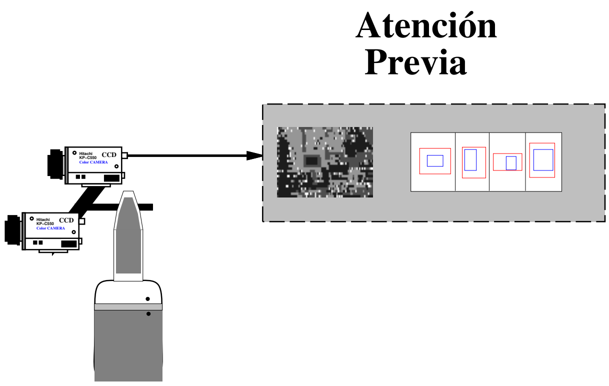


Figura 6.4: Fase de atención previa del sistema visual: Extracción de características basadas en el el color.

La información que se obtiene en la fase de atención previa no establece de una forma definitiva la existencia y localización del objeto de interés, pero sí una valiosa información que podrán utilizar los agentes de visión para detectar, reconocer, localizar y seguir el objeto.

6.2.2 Etiquetado de Imágenes

Nuestro sistema trata de evaluar la forma y las relaciones entre los diferentes objetos de interés a partir del mapa de características de la imagen, en lugar de procesar la información sobre la localización y forma de los diferentes objetos que componen la escena de forma independiente [59]. Para ello hay que conocer los diferentes objetos que componen la escena en términos de grupos conectados de pixels.

Es decir, una vez que los pixels de la imagen estén etiquetados como rojos, azules, verdes o desconocidos, éstos son agrupados con el fin de extraer las regiones de tamaño mayor de las que se pueda calcular sus características y las relaciones entre ellas.

Para ello se etiquetan las imágenes, con una técnica que utiliza la similitud entre los pixels vecinos como criterio para formar las diferentes regiones que existen en la escena [48]. Se trata de un procedimiento que agrupa pixels o sub-regiones similares en regiones más grandes.

Si llamamos $\Re$ a la región que incluye a la imagen completa, la segmentación es el proceso que divide $\Re$ en n regiones $\Re_1, \Re_2, \ldots, \Re_n$ de forma que:

1. La segmentación sea total; es decir, todos los pixels deben pertenecer a alguna región:

$$\bigcup_{i=1}^{n} \Re_i = \Re. \tag{6.1}$$

2. $\Re_i$ es una región conectada. Todos los pixels de una región deben estar conectados. Cuando solo se consideran los vecinos al norte, sur, este y oeste de un pixel, las regiones resultantes se dice que están 4-conectadas.

3. Las regiones deben de ser disjuntas:

$$\Re_i \cap \Re_j = \emptyset; \quad \forall i, j. \tag{6.2}$$

4. Todos los puntos de una región deben cumplir un predicado lógico, satisfaciendo una determinada condición.

$$P(\Re_i) = verdadero. \tag{6.3}$$

   Esta propiedad puede ser, por ejemplo, que el color de los pixels que componen una región sea el mismo.

6.2.3 Extracción de Características

Como resultado de etiquetar una imagen se obtiene una imagen simbólica, donde cada uno de sus elementos es una etiqueta que vincula el pixel a la región en la cual está incluida. A partir de esta información se pueden extraer características de las regiones, como sus puntos extremos, el área y el centroide.

En las siguientes definiciones designaremos el conjunto de pixels pertenecientes a una región como $\Re$. Asumiendo que los pixels de una imagen son cuadrados, una región se caracteriza por:

- **Área:**
$$A = \sum_{(x,y)\in\Re} 1$$
- **Centroide:**
$$\overline{X} = \frac{1}{A} \sum_{(x,y)\in\Re} x$$
$$\overline{Y} = \frac{1}{A} \sum_{(x,y)\in\Re} y$$
- **Puntos extremos:**

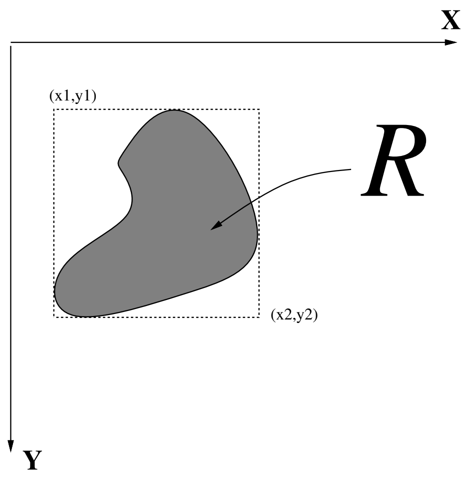


Figura 6.5: Puntos extremos de una región.

$$x_1 = \min\{x \mid (x,y) \in \Re\}$$
$$y_1 = \min\{y \mid (x,y) \in \Re\}$$
$$x_2 = \max\{x \mid (x,y) \in \Re\}$$
$$y_2 = \max\{y \mid (x,y) \in \Re\}$$

Determinadas características de las regiones segmentadas se calcularán siempre, como el área, sus puntos extremos, el centroide y el color. Sin embargo, hay otro tipo de características que serán calculadas en función del objeto que se desee detectar en la imagen. Es decir, el objeto que se desee detectar impondrá las características que se deben extraer. Por ejemplo, si se desea

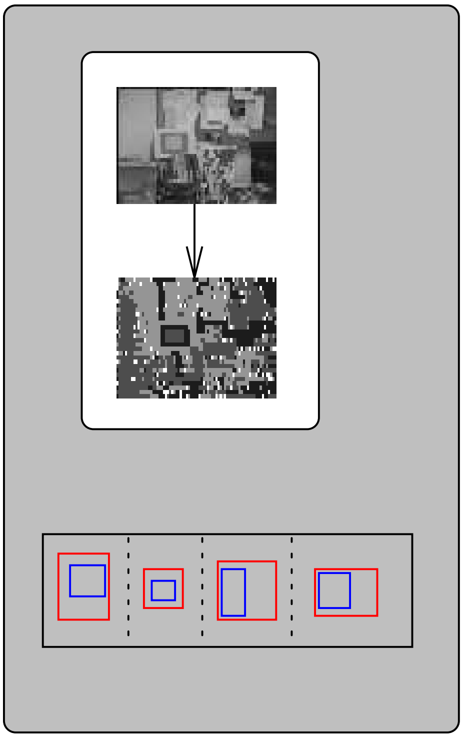


Figura 6.6: Segmentación y extracción de características de las regiones resultantes.

detectar una baliza compuesta por un cuadrado azul inscrito dentro de uno rojo, se calcularán solo las regiones azules que están inscritas dentro de regiones rojas.

Para determinar si la región $\Re_1$ está dentro de $\Re_2$ se ha optado por seguir el siguiente criterio:

Sea d_i la distancia entre el centroide $C_i \equiv (X_i, Y_i)$ de la región $\Re_i$ y su extremo más alejado. Si se cumplen las condiciones:

$$d_1 < d_2 \tag{6.4}$$

$$\sqrt{(X_1 - X_2)^2 + (Y_1 - Y2)^2} < (d_2 - d_1) \tag{6.5}$$

$$(d_2 - d_1) < d_1 \tag{6.6}$$

se considera que la región $\Re_1$ se encuentra incluida dentro de $\Re_2$.

6.3 Agentes de Seguimiento Visual

El sistema de seguimiento visual está compuesto de tres agentes que tratan de buscar (*Search*), detectar (*Detect*) y seguir (*Track*) la baliza. En la Figura 6.7 se puede observar la jerarquía de control del sistema, que básicamente sigue las directrices de una arquitectura *subsumption* [20]. Así, cuando un agente de mayor prioridad toma el control de los actuadores de la torreta del robot, inhibe las salidas de los restantes agentes del sistema.

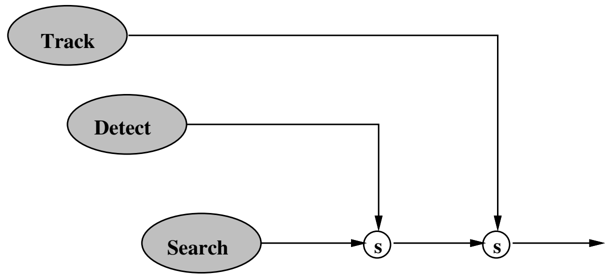


Figura 6.7: Jerarquía de control de los agentes motores de visión.

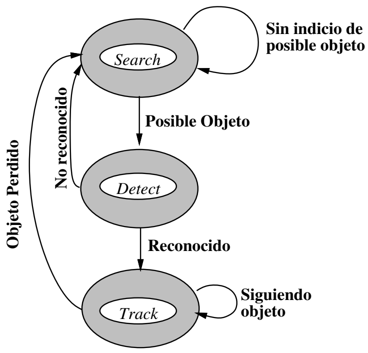


Figura 6.8: Flujo de control.

En la Figura 6.8 se puede observar el flujo de control entre los diferentes agentes de sistema. Cuando el agente *Search* está activo, trata de localizar la baliza en la imagen, moviendo la torreta hasta que encuentra un objeto que pueda ser una baliza. El agente *Detect* tratará de confirmar si ese objeto es la baliza que se está buscando, mediante la comprobación de un conjunto de características que deben cumplir las balizas. Si el objeto no fuese una baliza, el control de los motores de la torreta lo volvería a tomar el agente *Search*. Si *Detect* hubiese reconocido la baliza, el agente *Track* tomaría el control del sistema hasta que perdiese la baliza, en cuyo caso volvería a repetirse el ciclo, mediante la búsqueda de la baliza por el agente *Search*.

Sin embargo, y al contrario que en una concepción pura de arquitectura reactiva, los agentes comparten información acerca del estado del sistema. Así, el agente *Detect* le pasará al agente *Track* la posición de la baliza para su seguimiento. Por su parte, el agente *Search* recibirá continuamente información del agente *Track* de la posición de la baliza durante su seguimiento. Así, si la baliza se "pierde", el agente *Search* podrá hacer uso de la historia pasada para minimizar el espacio de búsqueda.

En las siguientes secciones se hará un detallado análisis de los diferentes agentes del sistema de seguimiento visual, atendiendo a su jerarquía de control.

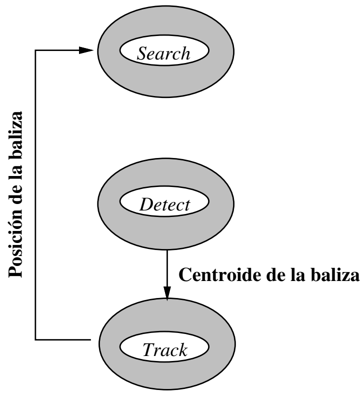


Figura 6.9: Flujo de información entre los agentes.

6.3.1 Search

El cometido principal del agente *Search* será el de mover la torreta de tal forma que se encuentre el objetivo visual. Para ello analiza la información proveniente de la fase de atención previa y actúa sobre los motores de la torreta.

El agente *Search* simplemente observa si existe un "indicio" de la presencia de la baliza en la imagen. Este "indicio" vendrá determinado por la existencia de algún elemento en la lista que provee el sistema de atención previa que satisfaga las condiciones atribuidas a las balizas. En caso afirmativo, *Search* parará el movimiento de la torreta, dando así la oportunidad al resto de agentes del sistema de determinar si el estímulo detectado por este agente se corresponde con la baliza busacada. Si es así, *Search* pasará el control de los actuadores de la torreta a los agentes *Detect* y *Track*.

Para evitar que la torreta se quede parada en una zona donde puedan darse un elevado número de estímulos erróneos (regiones incluidas dentro de otras pero que no satisfacen las condiciones que deben cumplir las balizas), el agente tiene un modo de auto-inhibición. Mediante la activación de un monoestable de auto-inhibición se obliga que pase un breve periodo de tiempo antes de que el agente pueda volver a parar la torreta. En este tiempo la zona del entorno que capten las cámaras se habrá modificado.

Zona de Búsqueda

Tal y como se verá en el Capítulo 7, el agente *Search* tiene un atributo en el que se puede indicar, mediante la asignación de una etiqueta simbólica (véase Tabla 6.2), la zona de búsqueda del objetivo. La zona de búsqueda se traduce en límitar los ángulos de giro de la torreta. En este sentido, la torreta es capaz de moverse en un rango comprendido entre los -164.79 y los 164.79 grados. La dirección 0 grados se corresponde con la dirección frontal del robot móvil, los ángulos negativos al lado izquierdo del robot, y los positivos al lado derecho (véase la Figura 6.10)

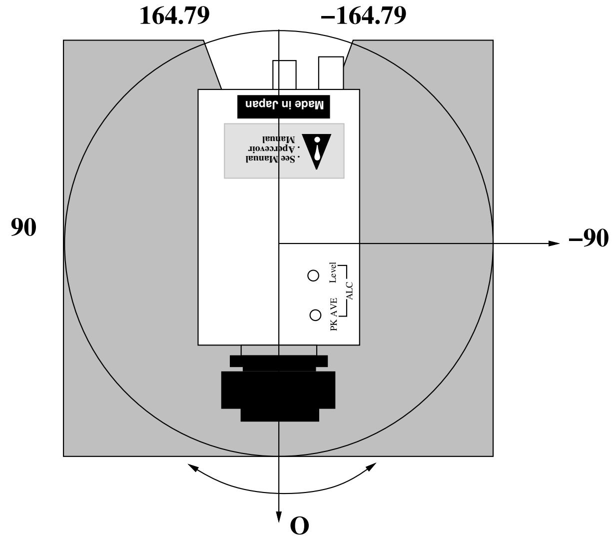


Figura 6.10: Rangos de movimiento de la torreta.

Etiqueta	Rango de Búsqueda
Forward	$[-100, 100]$
Left	$[-164.79, 0]$
Right	$[0, 164.79]$
Everywhere	$[-164.79, 164.79]$

Tabla 6.2: Etiquetas asignadas a los rangos de búsqueda del agente *Seach*.

Las acciones del agente *Search* dependen de la historia anterior del sistema, y se pueden describir dos estados bien diferenciados:

1. En un primer momento, cuando se inicia el sistema, y tal como se ha comentado en el párrafo anterior, un atributo del agente puede indicar la zona de búsqueda de la baliza. En este caso, el agente restringirá el rango de movimientos de la torreta al rango determinado por el atributo.

2. Una vez que la baliza haya sido detectada y seguida, el agente *Search* seguirá atento a la dinámica del sistema para saber por donde buscar la baliza si esta se ha perdido. En el caso en que el sistema de seguimiento (*Track*) no haya sido capaz de seguir la baliza, *Search* recordará la zona en donde se encontraba ésta por última vez. Así, si la baliza se encontraba a la izquierda del centro de la imagen, el rango de búsqueda se centrará en esa zona.

6.3.2 Detect

El agente *Detect* será el encargado de determinar si existe algún elemento de la lista que provee el sistema de atención previa que satisface todas las restricciones que debe cumplir la baliza que se está buscando. Las restricciones que deben cumplir las balizas son de dos tipos: geométricas y espacio-temporales.

Restricciones Geométricas

La lista de regiones que provee el sistema de atención previa solo indica la existencia de determinadas regiones incluidas dentro otras. Por ejemplo, si se desea reconocer y localizar una baliza compuesta por un cuadrado azul dentro de uno rojo, el sistema de atención previa proveerá una lista de todas las regiones azules de la imagen, que se encuentran incluidas de regiones rojas. Sin embargo, este tipo de información no explicita ninguna otra relación entre las regiones extraídas, y hará falta calcular nuevas características.

En el caso concreto del reconocimiento de las balizas, éstas deben cumplir dos restricciones geométricas:

- El cociente entre el área del cuadrado exterior $A_{exterior}$ y el del cuadrado interior $A_{interior}$ se mantenga por debajo de un valor umbral constante:

$$\frac{A_{exterior}}{A_{interior}} < \delta \tag{6.7}$$

- Tanto el centroide de la región exterior $(\overline{X_e}, \overline{Y_e})$ como el de la interior $(\overline{X_i}, \overline{Y_i})$, que componen la baliza, deben estar a una distancia menor de un cierto valor ϵ:

$$\sqrt{(\overline{X_e} - \overline{X_i})^2 + (\overline{Y_e} - \overline{Y_i})^2} < \epsilon \tag{6.8}$$

Las balizas siempre cumplirán estos dos criterios, independientemente del ángulo que forme el plano de la baliza con el de la lente. Es decir, se trata de criterios invariantes ante cambios en la proyección de la imagen de la baliza.

Restricciones Espacio-Temporales

Debido, sobre todo, a errores en la digitalización de la imagen, puede haber determinados elementos de la lista de objetos que provee el sistema de atención previa que cumplan la restricción geométrica, pero que no apunten a la baliza que se está buscando.

Este tipo de errores se dan con una frecuencia inversamente proporcional a las dimensiones de la imagen que se este capturando. Es decir, cuanto más pequeña es la imagen que se esta capturando, los pequeños errores se hacen más notorios al procesar dicha información. Sin embargo, la aleatoriedad con que se producen dichos errores, hace muy difícil que se produzca el mismo error en la misma posición de la imagen en instantes sucesivos.

Para establecer de una forma fidedigna que un determinado estímulo visual es una baliza, éste se tiene que mantener en el espacio durante un periodo de tiempo fijo. Por tanto, si un elemento pasa los criterios indicados en las ecuaciones (6.7) y (6.8), *Detect* comprobará si éste se vuelve a dar en la misma posición y con las mismas características geométricas durante la adquisición de las dos imágenes siguientes.

Salidas y Acciones de Control

El agente *Detect* no actúa directamente sobre los controles de la torreta del robot móvil. En el caso de que *Detect* reconozca una determinada baliza, le indicará al agente de seguimiento visual (*Track*), la posición (x, y), en coordenadas de la imagen, del centroide de la baliza. Con esa información, el agente de seguimiento visual tomará el control de los motores de la torreta y seguirá al objeto. En el caso en que *Search* pare el movimiento de la torreta, y *Detect* no encuentre entre el conjunto de estímulos uno que satisfaga tanto las restricciones geométricas como las espacio-temporales, *Search* volverá a tomar el control de los actuadores y seguirá la búsqueda de la baliza.

6.3.3 Track

En breves palabras, el sistema de seguimiento tratará de predecir la posición del objeto que esté siguiendo en coordenadas de la imagen, y actuará sobre los motores de la torreta de manera que mantenga al objeto en el centro de la imagen.

El agente *Track* opera únicamente sobre los datos recibidos directamente de la cámara, sin realizar por tanto un procesamiento de la señal de alto nivel. En este sentido, este sistema es de los que se basan en una técnica de correlación para seguir un objetivo visual. En el extremo contrario se encuentran los sistemas [93] [57] [19] que realizan un pre-procesamiento previo de la imagen y siguen determinados elementos, como esquinas, líneas, etc. Estos sistemas siguen la filosofía clásica en visión artificial [72], donde la visión se considera un proceso rígido con unas etapas de captación, pre-procesamiento, segmentación, reconocimiento y seguimiento que se ejecutan cíclicamente.

Nuestro sistema se basa fundamentalmente en la interacción del sistema con el entorno y el uso de lo ya percibido como elemento de optimización. En efecto, una vez que se ha localizado el estímulo de interés, el sistema de seguimiento no necesita pasar de nuevo por las etapas de reconocimiento del estímulo. Dos son las razones:

1. La velocidad de procesamiento y de respuesta del sistema aumenta en gran medida utilizando un mecanismo de seguimiento visual diferente al reconocimiento continuo del objetivo. A pesar de la sencillez y velocidad con la que se ejecutan los algoritmos de búsqueda, detección y reconocimiento de objetos, el seguimiento de un objeto basado en una técnica de correlación es mucho más rápido que la repetición cíclica del proceso de reconocimiento y seguimiento del objeto.

2. El estímulo que activó los agentes *Search* y *Detect* puede desaparecer durante el seguimiento del objeto. En efecto, puede darse el caso en que el objeto que se esté siguiendo esté semi-ocluido, o que las condiciones de iluminación ambiental no sean las idóneas para su reconocimiento. Sin embargo, una vez localizado el objeto de interés, no es necesario volver a reconocerlo mientras el sistema de seguimiento no lo pierda.

Seguimiento

El agente de seguimiento de la baliza está basado en una técnica de correlación entre una sub-imagen ($\mathbf{M}$), centrada en el centroide del objeto detectado y reconocido por el agente *Detect*. Esta sub-imagen, o máscara, se correlará en un espacio de búsqueda reducido $\{x_i \le x \le x_f,\ y_i \le y \le y_f\}$ de la imagen en niveles de gris ($\mathbf{I}$), que se obtiene de la tarjeta digitalizadora.

La posición de la máscara indicará la posición en la que se encuentra el objeto de interés en coordenadas de la imagen. Para conocer la posición de la baliza dentro de la imagen, la máscara extraída en el paso $t-1$ será correlada con la imagen tomada en el paso t.

Definamos $I(u,v,t)$ como el valor de la intensidad de un elemento de la imagen en la posición (u,v) en el instante t, y $M(u,v,t-1)$ como el valor de la intesidad del elemento de la máscara en la posición (u,v) en el instante $t-1$. (x,y) se define como la traslación en x y en y en una región acotada y rectangular $x_i \le u \le x_f$ y $y_i \le v \le y_f$. Los extremos superior izquierda (L_{x_i}, L_{y_i}) e inferior derecha (L_{x_f}, L_{y_f}), definen el tamaño de la máscara.

La función de correlación entre $M(t-1)$ e $I(t)$ se define formalmente como:

$$\sigma(x,y,t) = \frac{\int_{L_{y_i}}^{L_{y_f}} \int_{L_{x_i}}^{L_{x_f}} |I(u+x, v+y, t) - M(u,v,t-1)|\, du\, dv}{\int_{L_{y_i}}^{L_{y_f}} \int_{L_{x_i}}^{L_{x_f}} du\, dv} \tag{6.9}$$

La mejor correspondencia entre la máscara y la imagen vendrá dada por:

$$\sigma(t) = min(\sigma(\mathbf{x}, \mathbf{y}, \mathbf{t})); \quad x_i \leq x \leq x_f,\ y_i \leq y \leq y_f \tag{6.10}$$

$\sigma(t)$ determina la posición (x, y) a donde se ha movido el objeto, así como el mejor factor de correspondencia entre $M(t-1)$ e $I(t)$.

Pérdida del Objeto

El sistema de seguimiento visual ha de ser capaz de detectar cuando se ha perdido el objeto de forma autónoma. Debido a que el agente *Track* no realiza un reconocimiento simbólico de alto nivel del objeto para su seguimiento, dicha detección se basará en el factor de correspondencia entre la máscara que apunta al objeto, y la imagen en niveles de gris de la escena.

Por tanto, para detectar si el objeto se ha perdido, el factor

$$\vartheta = |\frac{\delta\sigma}{\delta t}| \tag{6.11}$$

establecerá el cambio en el tiempo de la mejor correspondencia entre la máscara y la imagen. Si ϑ es mayor que un umbral pre-determinado, *Track* se desactivará y pasará el control del sistema a los agentes *Search* y *Detect*.

La variación temporal del factor σ establecerá el cambio temporal que se ha producido entre los valores de los pixels de la máscara y los valores de los pixels de la zona de la imagen donde se supone que se encuentra el objeto. Ante cambios bruscos en el tiempo del factor σ, el sistema considera que el objetivo se ha perdido.

Selección de la Nueva Máscara

Existen numerosas alternativas para seleccionar una nueva máscara para ser utilizada como imagen de correlación en el paso $t + 1$:

1. Utilizar la máscara original extraída en la localización indicada por el agente *Detect*. El mayor inconveniente de esta alternativa es la propia dinámica del robot, que por lo general trata de acercarse al objetivo. Por tanto, a medida que se acerca el robot al objeto visual, éste cambiará de forma y el agente *Track* lo consideraría como "perdido", según el factor ϑ, con demasiada frecuencia.

2. Utilizar la nueva posición (x, y) como centro de la nueva máscara. Mediante esta estrategia de actualización se permite una cierta flexibilidad en los cambios que se producen en la imagen. Si el periodo de muestreo del sistema no es excesivamente bajo, la variación del factor σ, medido por ϑ, se mantendrá en el rango de valores en los que se considerará que el objeto no se ha perdido.

Por tanto, en cada instante t se extraerá una nueva máscara (M'), centrada en la posición (x, y), que servirá como la nueva máscara de correlación en el paso $t+1$.

Acciones de Control

El objetivo del sistema de seguimiento es mantener el objeto centrado dentro de la imagen. Otros sistemas [31], mantienen centrado el objeto en el centro de la imagen como método para estimar la distancia a la que está. Por su parte, otro tipo de sistemas [87] hacen una simulación de las propiedades del ojo humano, por lo que tratan de mantener el objeto dentro de la zona de la fóvea.

Sin embargo, en nuestro sistema no hacemos uso de una implementación que intente simular la fóvea de un ojo, ni las distorsiones radiales de la lente obligan a tener centrado el objeto en la imagen. La razón principal es que el robot puede seguir un movimiento independiente al de la cámara, como en el caso de que los agentes motores de navegación tomen el control de los actuadores para evitar un obstáculo. En este caso, el sistema de seguimiento visual estará siguiendo la baliza en paralelo a la acción de los agentes de navegación. Así, cuando los agentes de visión vuelvan a tomar el control pleno del robot, el objetivo visual no se habrá perdido.

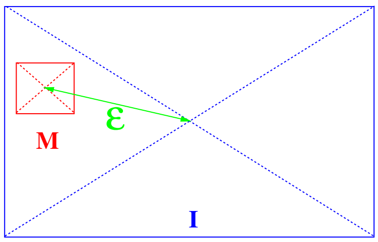


Figura 6.11: Error ϵ entre la posición de la máscara y el centro de la imagen.

Por tanto, las acciones de control del agente *Track* tratarán de mantener el centro de la máscara siempre en el centro de la imagen, ya que la probabilidad de perder el objetivo del campo de visión de las lentes es menor si el objeto está centrado en el centro de la imagen que en un extremo.

Si designamos como ϵ (véase Figura 6.11) a la distancia entre el centro de la máscara $(\overline{X_M}, \overline{Y_M})$ y el centro de la imagen $(\overline{X_I}, \overline{Y_I})$:

$$\epsilon = \sqrt{(\overline{X_M} - \overline{X_I})^2 + (\overline{Y_M} - \overline{Y_I})^2} \tag{6.12}$$

La acción de control que el agente *Track* mandará a los actuadores de la torreta será proporcional a ϵ:

$$y(t) = \begin{cases} K \cdot \epsilon & \epsilon > \tau \\ 0 & \epsilon \leq \tau \end{cases}$$

$y(t)$ especificará una acción de control que indicará la velocidad con la que se deberán mover los motores de la torreta. Para que el control de la torreta sea estable cerca del origen (donde la baliza está prácticamente centrada en la imagen), se optó por establecer un umbral τ a partir del cual se consideraba que la baliza estaba centrada y la torreta se debía parar.

La constante K dependerá tanto de las dimensiones de la imagen como de la velocidad con la que se esté muestreando la tarjeta de adquisición de imágenes. Para un tamaño de la imagen de 64×47 pixels, y una velocidad media de muestro de 35 frames por segundo, el valor de K calculado empíricamente es de 1.24.

Otro tipo de alternativa al modo de control de la torreta hubiese sido realizar un control "en posición"; es decir, conociendo la posición de giro de la torreta en un sistema de referencia determinado, indicarle la nueva posición a la que se debería mover, dependiendo de la señal de error ϵ. Las razones por no haber utilizado esta alternativa son principalmente dos:

1. Habría que realizar una costosa transformación entre las coordenadas de la imagen y las de la torreta. Para conocer las matrices de transformación entre las coordenadas de la imagen y de la torreta habría que realizar una calibración muy precisa de las cámaras.

2. El control en posición de la torreta de la que disponíamos para realizar el desarrollo experimental del trabajo, no permitía interrumpir la ejecución de una instrucción ya iniciada. Es decir, si se le indicaba a la torreta que se dirigiese a una determinada posición, hasta que no se terminase la ejecución de dicha instrucción no se podía volver a mandar una nueva señal de control a la torreta. Sin embargo, al mandar órdenes de control basados en la velocidad que debían mantener los motores de la torreta, permitía un control inmediato de los mismos.

6.3.4 Resultados Experimentales

El objetivo del experimento era que el sistema buscase, detectase y siguiese una baliza compuesta por un cuadrado azul dentro de uno rojo. En la Figura 6.12 se puede observar la secuencia de imágenes, tomadas a intervalos de aproximadamente 0.1 segundos, durante el periodo de búsqueda y detección de la baliza. En esta figura, se puede observar cómo las imágenes etiquetadas desde 1 a 13 muestran el resultado de la segmentación en colores generada por el sistema de atención previa. Lo que se trata de ilustrar con este tipo de representación es la secuencia de la activación de los diferentes comportamientos, ya que los agentes *Search* y *Detect* se alimentan de la información que suministra el sistema de atención previa, y el agente *Track* de la imagen en niveles de gris obtenida directamente de la cámara.

Inicialmente se le indicó al agente *Search* que la baliza se debía encontrar a la izquierda. La secuencia de imágenes de la 1 a la 10 se tomaron en el rastreo del entorno para detectar la baliza. En la imagen 11, el agente *Search* paró los motores de la torreta, para que el agente *Detect* determinase si el estímulo satisfacía las restricciones geométricas del objeto baliza. Una vez reconocido el estímulo como la baliza buscada, *Detect* le "indicó" al agente *Track* la posición

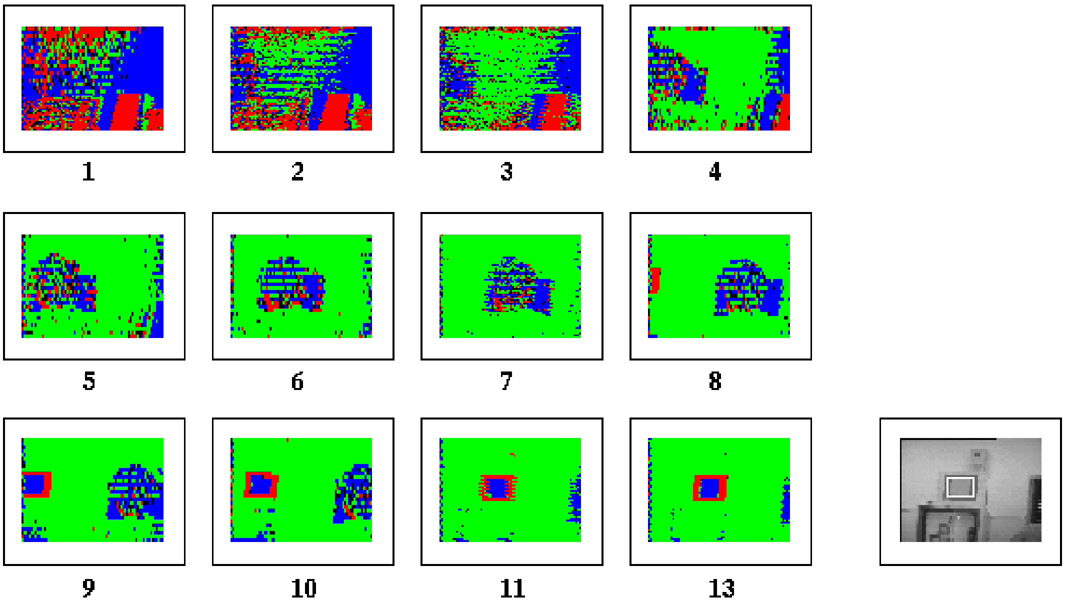


Figura 6.12: Fase de búsqueda y detección.

del centroide de la baliza, y éste tomó el control de la torreta para su posterior seguimiento. La imagen situada en la esquina inferior derecha de la Figura 6.12, muestra la imagen (en niveles de gris) adquirida por el agente *Track*, y recuadrada la zona que se utilizaría como imagen de correlación, o máscara.

La Figura 6.13 muestra la secuencia de imágenes tomadas durante el seguimiento de la baliza. El recuadro más claro indica la posición y tamaño de la máscara de correlación de la que hace uso el agente *Track* para el seguimiento de la baliza. La secuencia temporal va de izquierda a derecha y de arriba a abajo.

Tal y como se puede observar en las imágenes, se tomó la baliza por un extremo y se realizó un movimiento de izquierda a derecha (secuencia desde (a-1) hasta (d-4)). En la imagen (d-4) comienza el movimiento de derecha a izquierda, y el sistema es capaz de reaccionar ante estos bruscos cambios en la trayectoria del objeto.

En la fila 7 de la Figura 6.13, se puede observar cómo el agente *Track* pierde momentáneamente la baliza (secuencia (7-b) hasta (7-e)). El agente *Search* obtuvo en todo momento la información de dónde se encontraba la baliza (izquierda o derecha del centro de la imagen), por lo que movió la cámara hacia la izquierda en cuanto *Track* perdió la baliza. Tal y como se puede observar en las imágenes, la baliza seguía en movimiento durante el periodo de tiempo que *Track* la perdió, y sin embargo los agentes *Search* y *Detect* fueron capaces de encontrarla y reconocerla rápidamente (0.4 segundos), tomando de nuevo el agente *Track* el control del sistema (imagen 8-a).

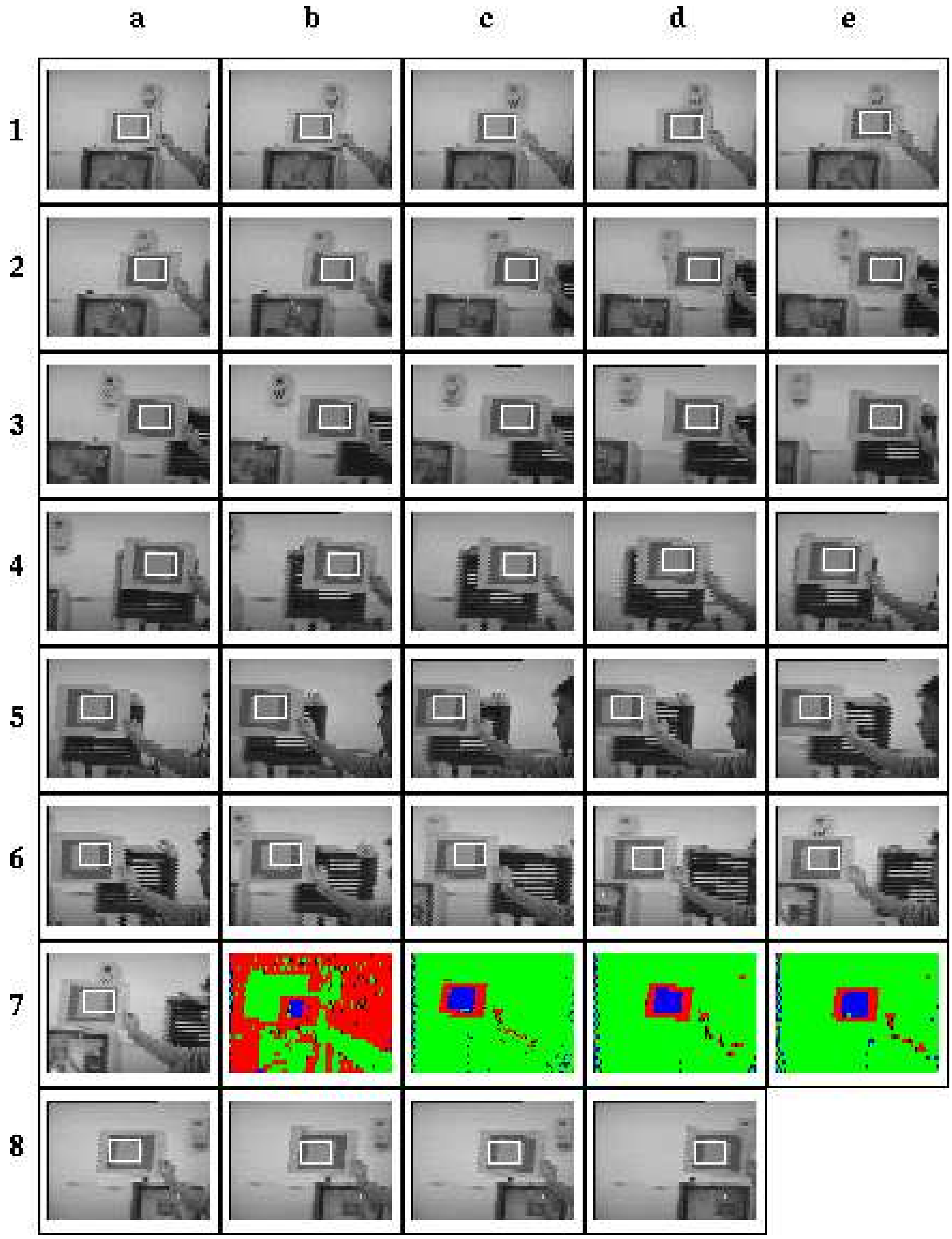


Figura 6.13: Seguimiento de la baliza.

6.4 Sistema de Visión Estereoscópica

Las tareas que debe realizar el robot móvil requieren que éste conozca la distancia que le separa de las balizas. Existen numerosas alternativas para conocer la distancia que separa al objeto del robot. Por ejemplo, conociendo las dimensiones y características geométricas de la baliza, se podría inferir la distancia mediante el estudio de su proyección en el plano imagen; o mediante un sistema binocular montado sobre una plataforma activada por motores [31], se podría mantener la proyección del objeto en el centro de las dos imágenes: El ángulo de convergencia entre las dos cámaras indicaría la distancia que separa al sistema visual del objeto.

En nuestro caso, hemos decidido implementar un sistema de medición 3-D que no dependa de la geometría del elemento a detectar. Como las características del sistema no nos permiten una estrategia *activa* a la hora de calcular distancias, hemos optado por utilizar una técnica pasiva de cálculo de la profundidad, como es un sistema estereoscópico binocular, que calcula la distancia entre el objeto visual y las cámaras basándose en una simple técnica de triangulación [41].

Los pasos en todo sistema de estereovisión pasivo se reducen a tres [35]:

- Extracción de características de las imágenes.
- Correspondencia entre características.
- Cálculo de disparidad entre las características.

6.4.1 Selección de Características

El grado de información que contengan las características extraídas para su posterior correspondencia influye significativamente en el tiempo de proceso. En este sentido, se puede realizar una clasificación de las características extraídas en función de su grado de información:

1. Puntos de alto contraste: Basándose en estudios de origen psicológico, *Marr et al.* [71] propusieron extraer puntos de las imágenes que perteneciesen a zonas de alto contraste, como bordes o esquinas. El bajo contenido en información implica que en la fase de correspondencia se utilicen elaborados algoritmos heurísticos [46], o de programación dinámica [81], para reducir la incertidumbre.
2. Segmentos: Mediante la utilización de segmentos [7][6][76] como elementos de correspondencia, se reduce de una forma sustancial el espacio de búsqueda en la fase de correspondencia.
3. Regiones: Mediante la segmentación de la imagen en diferentes regiones [70] se consigue la mayor reducción posible del espacio de búsqueda, ya que las propiedades de las regiones (área, perímetro, nivel medio de gris, etc.) poseen un alto valor discriminatorio. Desgraciadamente, mediante esta técnica la resolución del mapa de disparidades es muy

gruesa, no permitiendo en principio su aplicación a sistemas que requieran de una precisión muy alta.

La velocidad de respuesta es la restricción crítica de nuestro sistema, y por tanto, una segmentación inicial en regiones permite una rápida correspondencia entre las características de las dos imágenes.

Todas estas necesidades y restricciones ya habían sido contempladas en el sistema de seguimiento visual, por lo que se pueden re-utilizar agentes de este sistema para calcular la distancia entre el robot y la baliza. El agente *Detect* está diseñado para reconocer un conjunto de balizas, y como salida proporciona una lista de objetos de la escena que cumplen las condiciones geométricas de las balizas. Por tanto, asignando un agente *Detect* tanto a la imagen izquierda como a la derecha, capturadas por las dos cámaras del sistema, la fase de correspondencia se reduce a establecer el grado de similitud entre la información de las dos imágenes.

6.4.2 Fase de Correspondencia

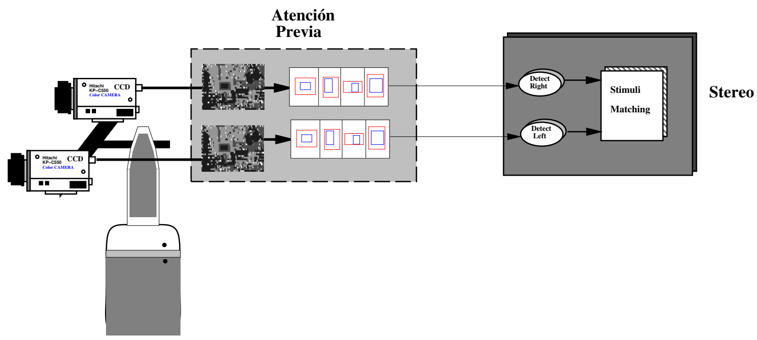


Figura 6.14: Fase de Correspondencia

De forma genérica, en la fase de correspondencia el sistema de estereovisión debe establecer qué elementos de la imagen izquierda $\{i_1, i_2, \ldots, i_n\}$ se corresponden con los de la imagen derecha $\{d_1, d_2, \ldots, d_n\}$. Los elementos de ambas imágenes son extraídos por dos agentes *Detect*, que procesan la información de cada una de las cámaras (véase Figura 6.14).

Sin embargo, al realizar el seguimiento visual con una de las cámaras, la lista de posibles balizas de esta cámara se reduce a un elemento, aprovechándo el hecho de que el sistema de seguimiento visual está siguiendo al objeto de interés. Por tanto, se deberá establecer cual de los elementos de la otra imagen se corresponde con la baliza detectada y seguida.

Sea O el elemento de la imagen derecha‡ detectada por el agente *Detect Derecha*. Por su parte, el agente *Detect Izquierda* proveerá un conjunto $\{i_1, i_2, \ldots, i_n\}$ de posibles balizas. En el caso de que el número de elementos de la lista seleccionados por el agente *Detect Izquierda* sea mayor

‡En este caso se parte del hecho de que el sistema de seguimiento visual ha realizado el seguimiento de la baliza basándose en las imágenes provistas por la cámara derecha.

que uno, se hará uso de la restricción epipolar y la de orden para establecer la correspondencia entre O y los elementos de la lista $\{i_1, i_2, \ldots, i_n\}$:

Restricción Epipolar

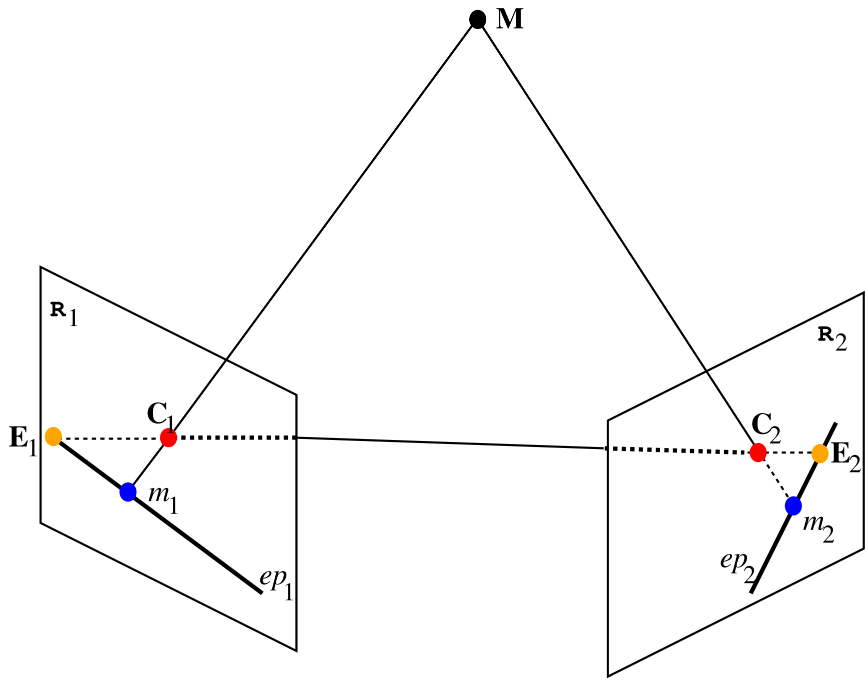


Figura 6.15: Geometría Epipolar.

Sean C_1 y C_2 los centros ópticos de dos cámaras cuyos planos imagen R_1 y R_2 se encuentran a la distancia f_1 y f_2 de sus respectivos centros ópticos. Sea M un punto en el espacio, siendo m_1 y m_2 sus proyecciones en los planos imagen R_1 y R_2 respectivamente.

En la Figura 6.15 se puede observar que, dado el punto m_1 del plano imagen R_1, todos los posibles puntos M que pueden haberse proyectado en él se encuentran sobre la recta $< m_1, C_1 >$. Como consecuencia directa, todas las posibles correspondencias del punto m_1 en el plano R_2 se encuentran sobre la proyección en el plano imagen R_2, de dicha recta. La proyección de $< m_1, C_1 >$ en el plano R_2 es la línea infinita ep_2, denominándose línea epipolar del punto m_1 en la imagen R_2. Todas las líneas epipolares de la imagen R_2 pasan por el punto E_2, llamado epipolo, que es la intersección de la recta que une los centros ópticos de ambas cámaras $< C_1, C_2 >$, con el plano R_2.

Por tanto, la restricción epipolar establece que, dado un punto m_1 del plano R_1, sus posibles correspondencias en el plano R_2 se encuentran sobre la línea epipolar correspondiente. De tal forma se reduce el espacio de búsqueda bidimensional a una búsqueda unidimensional. La restricción epipolar es simétrica y, por tanto, para un punto m_2 en el plano R_2, las posibles correspondencias en el plano imagen R_1 se encuentran sobre la línea ep_1 que pasa por el epipolo

E_1, que es la intersección de la línea $< C_1, C_2 >$ con el plano R_1. Las líneas ep_1 y ep_2 son las intersecciones del plano C_1MC_2, denominado el *plano epipolar* definido por M, con los planos imagen R_1 y R_2, respectivamente.

Cuando los planos R_1 y R_2 son paralelos a la línea $< C_1, C_2 >$ los epipolos se encuentran en el infinito, y las líneas epipolares son paralelas en los planos R_1 y R_2. Esta situación, en la que los planos imagen son paralelos a la línea $< C_1, C_2 >$, es la que se da en nuestro sistema binocular: ambas cámaras reposan sobre el mismo plano y sus planos imagen son paralelos.

Restricción de Orden

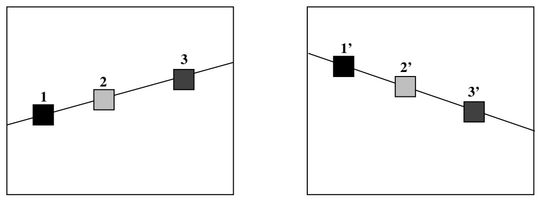


Figura 6.16: Restricción de Orden.

Esta restricción presupone que el orden entre los puntos homólogos sobre la línea epipolar se mantiene, Tal y como se puede observar en la Figura 6.16, el orden entre los puntos de la imagen izquierda 1, 2 y 3 se mantiene en su proyección en la imagen derecha: 1', 2' y 3'.

En situaciones límite, puede ocurrir que esta relación de orden no se satisfaga [41], pero las características geométricas de los objetos presentes en la escena nunca violarán dicha restricción.

6.4.3 Cálculo de la Distancia

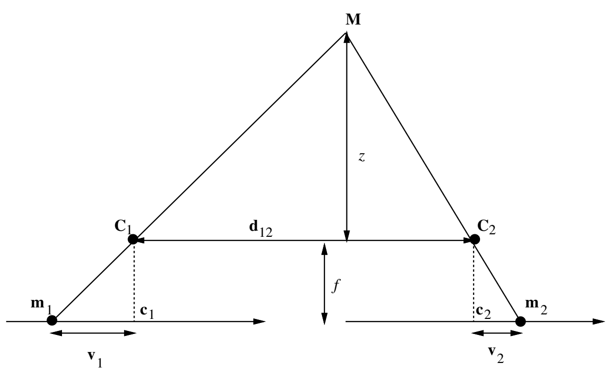


Figura 6.17: Relación entre la distancia y la disparidad.

Dado un pixel m_1 con coordenadas (u_1, v_1) en el plano imagen izquierdo, y su correspondiente pixel m_2 con coordenadas (u_2, v_2) en el plano imagen derecho, la *disparidad* se define como la diferencia $v_2 - v_1$. Esta definición implica que los planos imagen de las cámaras coincidan, tal y como lo ocurre en nuestro sistema de cámaras. Un valor de disparidad 0 implica que el punto M se encuentra en el infinito. Si se acerca el punto M al centro óptico C_1 a lo largo de la línea $< m_1, C_1 >$, la disparidad aumentará desde 0 hasta $+\infty$.

Observando en la Figura 6.17 que los triángulos $m_1c_1C_1$, m_1m_2M, y $m_2c_2C_2$ son similares, la relación entre la disparidad y la distancia (z) a la que se encuentra el punto M de la línea que une los centros ópticos de las dos cámaras, se calcula mediante la expresión:

$$d = v_2 - v_1 = \frac{d_{12}f}{z} \tag{6.13}$$

Siendo d_{12} la distancia que separa a los centros ópticos de las dos cámaras, y f la focal de las lentes.

Las unidades de medida en las que debe expresarse el valor de la disparidad (d), serán las mismas que se utilizan para la focal y la distancia d_{12}. Por tanto, el valor de la disparidad debe transformarse mediante la expresión:

$$d = (v_2 - v_1) * p \tag{6.14}$$

donde p es el tamaño de un pixel del CCD de la cámara de vídeo.

Finalmente, los puntos m_1 y m_2 que se utilizan para el cálculo de la profundidad a la que se encuentra la baliza serán los centroides de las balizas que se corresponden en la imagen izquierda y derecha.

6.4.4 Transformación de Coordenadas

La información de la distancia a la que se encuentra la baliza del robot móvil será utilizada por los agentes motores de navegación. En concreto, el agente de ir a un punto utilizará esta información para acercarse a la baliza. Genéricamente, este agente recibe como entrada las coordenadas de un punto del espacio referido a un sistema de referencia general. Por tanto, el valor de distancia que el sistema de estereovisión calcula se ha de referir a este sistema de referencia.

En la Figura 6.18, se representan los sistemas de referencia que se utilizan:

- Sistema de referencia general: X_oY_o.
- Sistema de referencia de la plataforma móvil: X_cY_c.
- Sistema de referencia de la torreta: X_tY_t.

Los sensores odométricos proporcionan la traslación (T_{x_c}, T_{y_c}) y la rotación (α_c) del sistema de referencia asociado a la plataforma móvil respecto al sistema de referencia general (X_oY_o). De tal forma, para cualquier (x_c, y_c) punto referido en el sistema de referencia X_cY_c, su localización (x_o, y_o) en el sistema de referencia $X_oY_o$viene dado por la expresión:

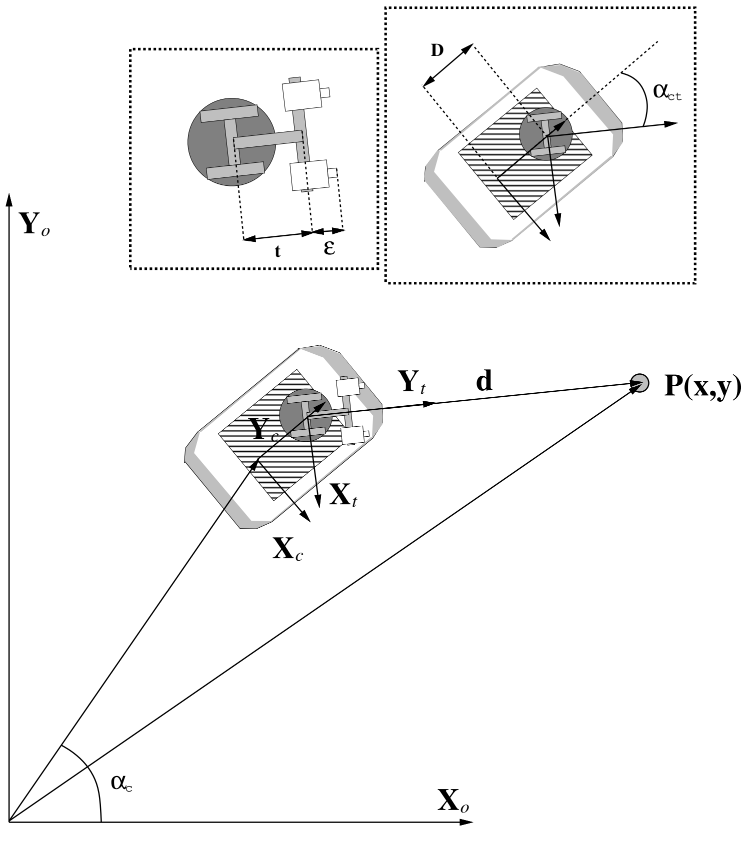


Figura 6.18: Transformación de coordenadas.

$$\underbrace{\begin{pmatrix} 1 \\ x_o \\ y_o \end{pmatrix}}_{\vec{X}_o} = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ T_{x_c} & \cos\alpha_c & \sin\alpha_c \\ T_{y_c} & -\sin\alpha_c & \cos\alpha_c \end{pmatrix}}_{\mathbf{M_c}} \begin{pmatrix} 1 \\ x_c \\ y_c \end{pmatrix} \tag{6.15}$$

Por otra parte, la torreta tiene unos sensores (encoders) que permiten conocer en todo momento su ángulo de giro (α_{ct}), encontrándose la torreta fija en la plataforma móvil a una distancia D del origen del sistema de referencia asociado a la plataforma móvil (X_cY_c). Cualquier punto (x_t, y_t) referido al sistema de referencia X_tY_t se transforma como un punto (x_c, y_c) en el sistema de referencia X_cY_c mediante la expresión:

$$\begin{pmatrix} 1 \\ x_c \\ y_c \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha_{ct} & \sin\alpha_{ct} \\ D & -\sin\alpha_{ct} & \cos\alpha_{ct} \end{pmatrix}}_{\mathbf{M_t}} \begin{pmatrix} 1 \\ x_t \\ y_t \end{pmatrix} \tag{6.16}$$

El sistema de estereovisión calcula una distancia d. Trabajando con las cámaras situadas en el mismo plano, y al estar sus epipolos situados en el infinito, la distancia d estará calculada respecto al plano focal de las dos cámaras (véase la Figura 6.17). Tal y como está configurado el sistema de estereoscópico, el plano focal de las cámaras y el centro del sistema de referencia de la torreta (X_tY_t) están separados por una distancia $t + \epsilon$. Por tanto, la baliza se localizará en el sistema de referencia asociado a la torreta (X_tY_t) mediante la expresión:

$$\begin{pmatrix} 1 \\ x_t \\ y_t \end{pmatrix} = \underbrace{\begin{pmatrix} 1 \\ 0 \\ d + \epsilon + t \end{pmatrix}}_{\vec{X}_t} \tag{6.17}$$

Por tanto, para conocer la posición de la baliza en el sistema de referencia general X_oY_o, se ha de utilizar la expresión:

$$\boxed{\vec{X}_o = \mathbf{M_c} \cdot \mathbf{M_t} \cdot \vec{X}_t} \tag{6.18}$$

6.4.5 Toma de Medidas

La ecuación 6.18 permite mantener la localización de la baliza independiente de la posición del robot móvil. Según se mueva el robot, el sistema de estereovisión realizará varias medidas de la localización de la baliza. Las razones son:

1. Aumentar la certeza de la localización por medio de repetidas medidas.
2. Descartar falsas medidas.

Para realizar tareas de navegación basándose en la información provista por el sistema de estereovisión, es importante conocer la precisión del mismo. Estudios analíticos de las características de los sistemas estereoscópicos binoculares [60] hacen un detallado análisis de los errores inherentes al proceso del cálculo de la posición. De forma analítica, se demuestra que el error en la estimación de la distancia es inversamente proporcional a la profundidad a la que se encuentra el objeto de las cámaras.

Así, en cuanto se realice una medida 3D de la localización de la baliza, se considerará como **buena** si esta se encuentra a una distancia menor de 4 metros del robot móvil.

Si denotamos por $M = \{m_1, m_2, \ldots, m_n\}$ el conjunto de medidas buenas de la localización de la baliza, la media del conjunto M fijará la posición de la baliza en el sistema de referencia general X_oY_o.

6.4.6 Resultados Experimentales

Los estudios analíticos, sobre la precisión de los sistemas estereoscópicos binoculares [60], se realizan considerando que las condiciones son ideales, obviando ciertos problemas del "mundo real". Por tanto, hemos optado por realizar una serie de pruebas experimentales que ilustren de forma cuantitativa la precisión del sistema de estereovisión implementado.

Las Figuras 6.19 y 6.20 muestran dos escenarios diferentes. En la primera figura se muestran las medidas realizadas por el sistema de estereovisión, cuando ni la plataforma móvil ni la torreta se movían. La Figura 6.20, sin embargo, muestra el caso más general en el que tanto la plataforma como la torreta se movían.

En el primer caso (robot estático) se trató de estimar los errores asociados únicamente al sistema de estereovisión, y en el segundo (robot en movimiento), la resultante de la combinación de todos los errores.

Robot Estático

En la Figura 6.19 el origen del sistema de referencia asociado al robot móvil X_cY_c se encontraba a una distancia de 3 metros de la baliza. El sistema de estereovisión tomó 11 medidas de la posición a la que encontraba ésta, y la figura muestra los datos de las mediciones respecto al sistema de referencia general X_oY_o. Durante la toma de medidas, tanto la plataforma móvil como la torreta no se movieron, tratándose de estimar, por tanto, el error asociado únicamente al sistema de estereovisión.

La media de todas las medidas se encontraba en la posición $(608cm, 294cm)$ en el sistema de referencia general, con un error estimado, mediante la desviación estándar, de $6\,cm$ en el eje x y de $4\,cm$ en el eje y.

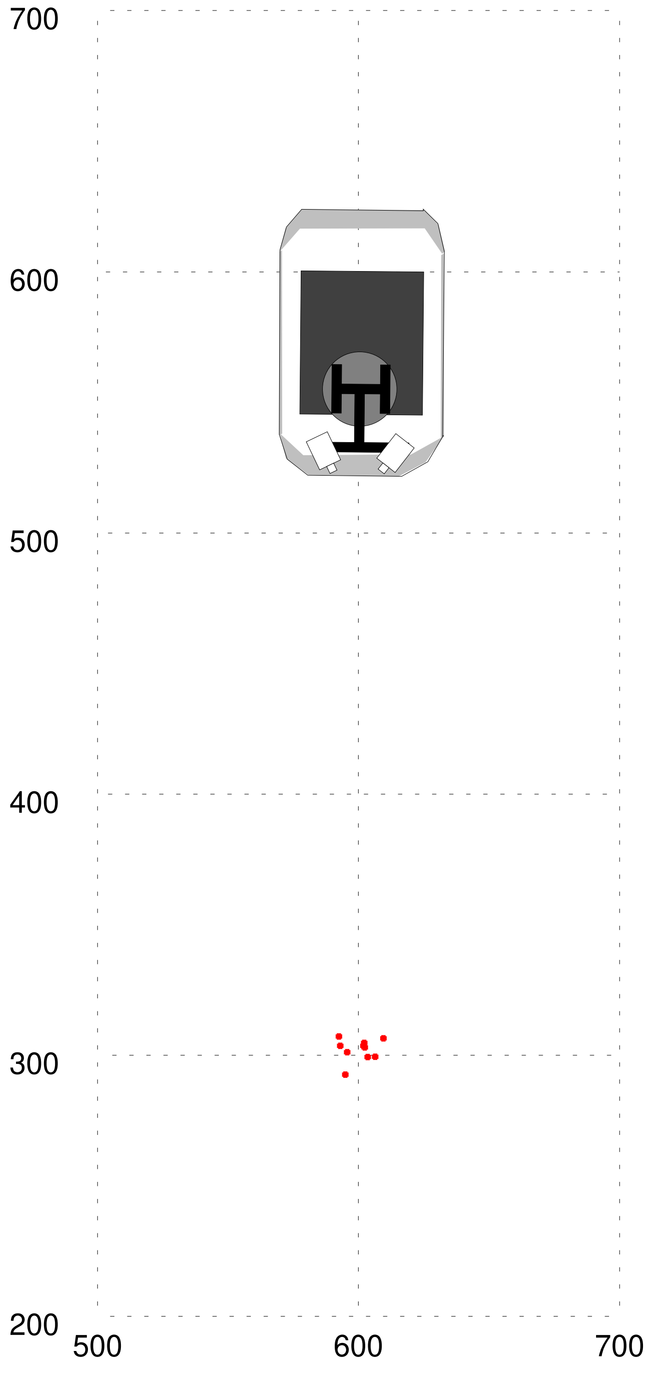


Figura 6.19: Medidas sin movimiento.

$$\overline{X} = 608 \pm 6\,cm$$

$$\overline{Y} = 294 \pm 4\,cm$$

Robot en Movimiento

La Figura 6.20 muestra la trayectoria seguida por el robot móvil, y las medidas realizas por el sistema de estereovisión de la posición de la baliza.

Los datos de la trayectoria del robot móvil fueron obtenidos por los sensores odométricos de la plataforma móvil. Estos datos estiman las posiciones del origen del sistema de referencia asociado a la plataforma móvil (X_cY_c), respecto al origen del sistema de referencia general (X_oY_o). La orientación de la plataforma se infiere partiendo del hecho de que el eje de ordenadas del sistema de referencia X_cY_c es tangente a la trayectoria (véase Figura 6.18).

La trayectoria muestra cómo la orientación del robot móvil respecto a la baliza cambiaba mientras éste se acercaba y alejaba de ella. Simultáneamente al movimiento de la plataforma, el sistema de seguimiento visual seguía a la baliza, moviendo la torreta para mantener la baliza en el centro de la imagen de la cámara derecha.

Por otro lado, y en periodos de tiempo fijos, se calculaba la distancia a la que se encontraba la baliza. Los datos de las mediciones mostraron un error estimado por la desviación estándar de $\pm 7\,cm$ en el eje x y de $\pm 9\,cm$ en el eje y.

$$\overline{X} = 611 \pm 7\,cm$$

$$\overline{Y} = 297 \pm 9\,cm$$

Los errores en la estimación de la posición de la baliza se deben fundamentalmente a:

1. Los errores en la estimación de la posición y orientación de los orígenes de cada sistema de referencia respecto del sistema de referencia general, medidos por los sensores odométricos correspondientes.
2. El ruido presente en las imágenes que afectan al cálculo de la disparidad entre las proyecciones de la baliza en las dos imágenes.

Sin embargo, la precisión del cálculo de la distancia por el sistema de estereovisión, entra dentro de los límites admisibles de una planificación de alto nivel (véase el Capítulo 7).

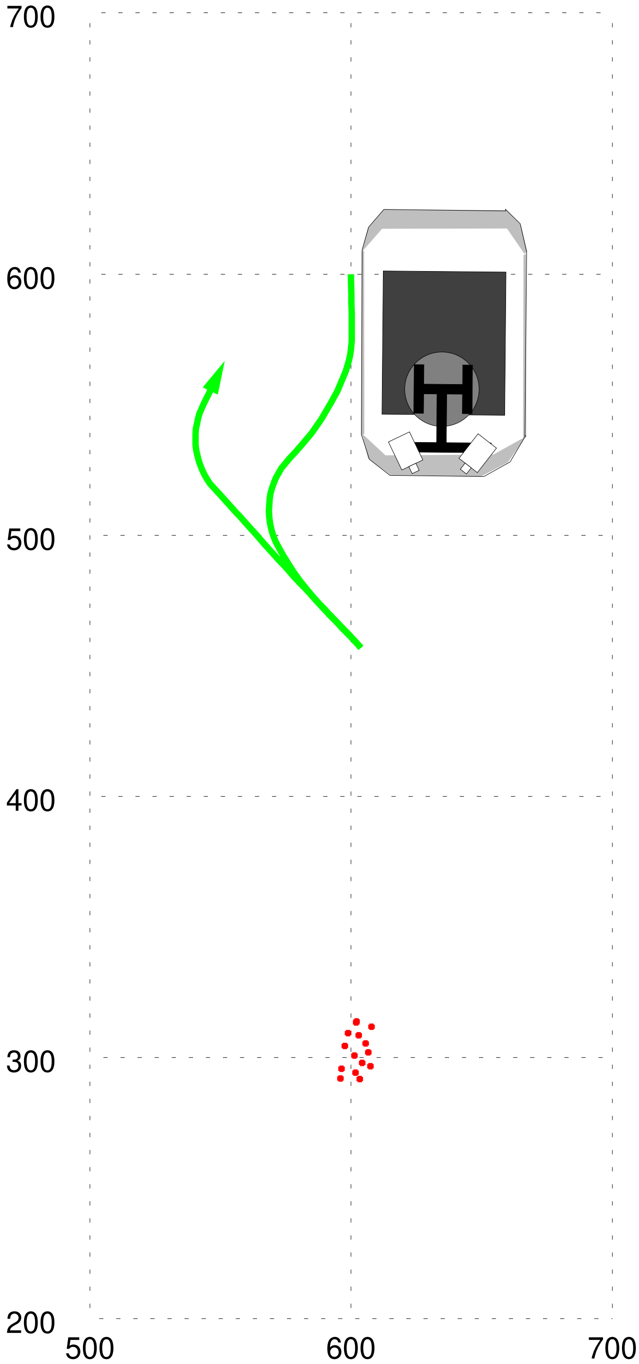
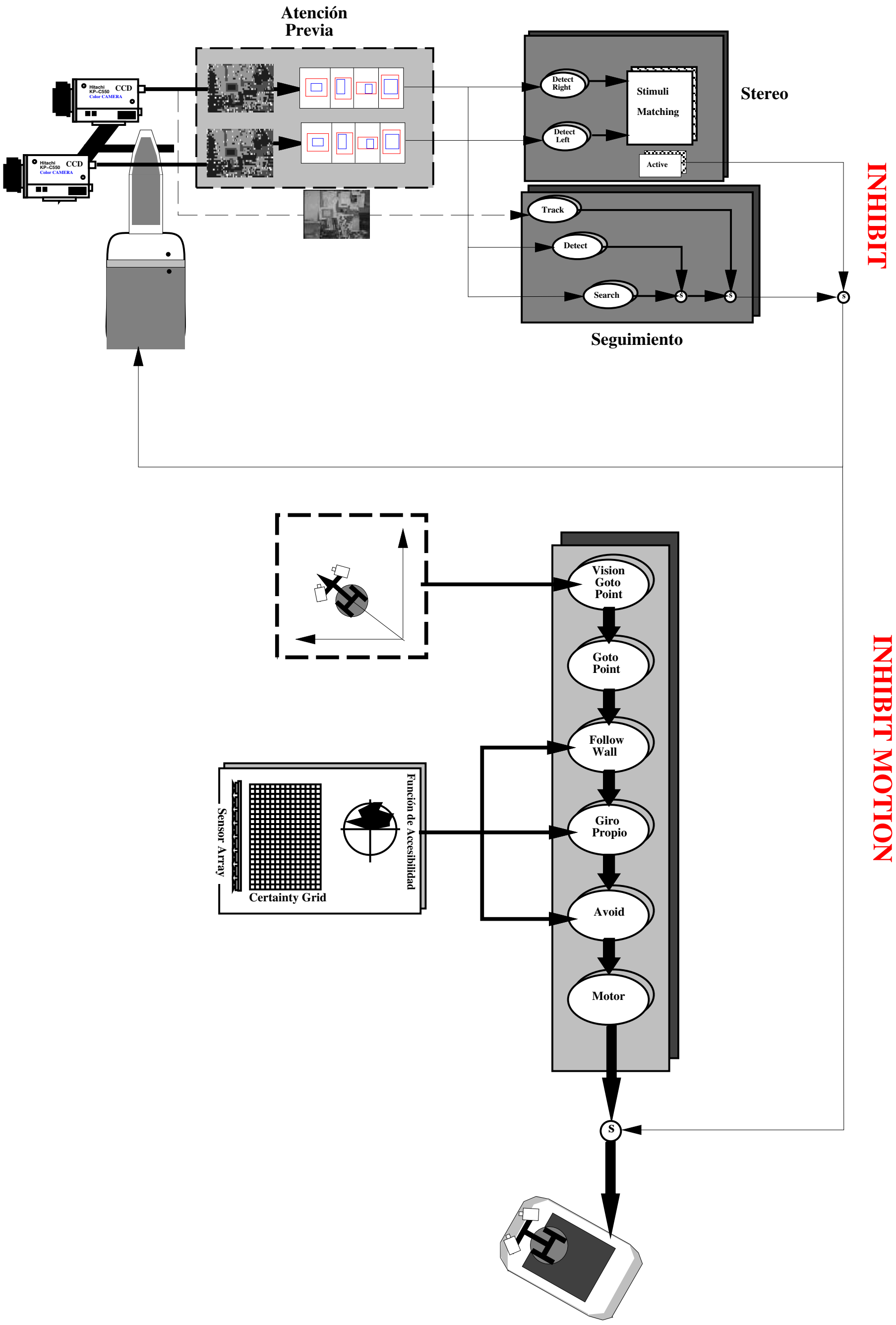


Figura 6.20: Medidas en movimiento.


Atención
Previa
CCD
CCD
Detect
Right
Detect
Left
Stimuli
Matching
Active
Stereo
Track
Detect
Search
Seguimiento
INHIBIT
Vision
Goto
Point
Goto
Point
Follow
Wall
Giro
Propio
Avoid
Motor
Función de Accesibilidad
Sensor Array
Certainty Grid
S
INHIBIT MOTION

6.5 Interacción entre Sub-Sistemas

A pesar de que los agentes motores de navegación operan en paralelo con los de visión, hay ocasiones en las que deben interactuar directamente. En este sentido, la interacción se limita a:

1. Los datos que el sistema de estereovisión proporciona al agente *Goto Point* de la localización de la baliza le permiten a este último dirigirse a la posición donde se encuentra la baliza.
2. La inhibición de las salidas de los agentes motores de navegación, por parte de los agentes de visión.

En este apartado nos centraremos en las contingencias que motivan que los agentes de visión inhiban las acciones de control de otros agentes. En este sentido, existen dos tipos de inhibición:

- **Inhibición total:** Se inhiben los movimientos de la plataforma y de la torreta.
- **Inhibición de movimiento:** Se inhibe el movimiento de la plataforma, pero no el de la torreta.

Como ya se ha descrito con anterioridad, el sistema de visiíon artificicial se compone de los agentes de seguimiento visual y los de estereovisón. En los siguientes sub-apartados nos centraremos en cada uno de ellos, y de cómo interaccionan entre si y los agentes motores de navegación.

6.5.1 Seguimiento Visual de la Baliza

Si no se conoce la localización de la baliza, de forma que el agente *Goto Point* se pueda dirigir a ella, pero sí que se está siguiendo el objetivo por medio de los agentes motores de seguimiento visual, el agente *Vision Goto Point* tratará de acercarse a la baliza.

Si en esta situación se pierde la baliza, los agentes motores de seguimiento visual inhibirán las salidas de los agentes motores de navegación de forma que la plataforma móvil no se mueva. Es decir, los agentes de seguimiento se encargarán de volver a buscar, reconocer y seguir a la baliza, y únicamente se pararán los movimientos de la plataforma, pero la torreta seguirá atendiendo a los comandos de control de estos agentes.

Una vez que el agente *Track* esté siguiendo la baliza, las condiciones para que se desinhiban las salidas de los agentes motores de navegación son:

1. Que el centroide de la baliza (C_x, C_y) esté cerca del centro de la imagen (X, Y).

$$\sqrt{(X - C_x)^2 + (Y - C_y)^2} < \xi_1 \tag{6.19}$$

2. Que el último control de movimiento azimutal (*pan*) de la torreta sea pequeño:

$$|pan| < \xi_2 \tag{6.20}$$

6.5.2 Sistema de Estereovisión

La tarjeta de adquisición de imágenes no es capaz de adquirir dos imágenes en color simultáneamente. Sin embargo, posee un multiplexor que permite la adquisición de dos fuentes de vídeo de forma secuencial. Hay un periodo de estabilización entre el cambio de fuente de vídeo y la adquisición de la nueva imagen. Por esta razón es por la cual el sistema de estereovisión para el movimiento de la plataforma y de la torreta siempre que vaya a realizar una medición de la localización de la baliza.

La inhibición de la plataforma y de la torreta se realiza de forma secuencial. Primero se inhibe el movimiento de la plataforma, dando así oportunidad a los agentes de seguimiento de tener la baliza en el centro de la imagen. En cuanto se cumplan las condiciones expresadas en las ecuaciones (6.19) y (6.20), el sistema de estereovisión estará preparado para realizar una medición de la localización de la baliza.

Una vez realizada la medición 3-D, el sistema de estereovisión desinhibirá los controles de la plataforma y la torreta.

Activación del Sistema de Estereovisión

El sistema de estereovisión se auto-activa en periodos de fijos de tiempo que dependen de la historia del sistema, diferenciándose dos estados:

1. Cuando el sistema de estereovisión aún no ha conseguido establecer la localización de la baliza, se auto-activará en periodos cortos de tiempo.

2. En el caso en el que ya se haya realizado una medida "buena", la auto-activación se realizará en periodos de tiempo más alejados, con lo que el robot se dirigirá a su meta más rápidamente.

Capítulo 7

Interacción con el Usuario

7.1 Introducción

El principal objetivo de la Inteligencia Artificial en general, y de la robótica móvil en particular, es desarrollar sistemas con un alto grado de autonomía que sean capaces de operar en entornos no estructurados y con poca información *a priori*. Para lograr este grado de independencia, el robot debe ser capaz de interpretar la información que proveen los sensores con los que va equipado y actuar en función de ella en tiempo real.

El objetivo de todo robot móvil autónomo debe ser la ejecución de un plan propuesto, y la forma de llevarlo a cabo dependerá de la filosofía de diseño con la que se haya construido la arquitectura de control del robot móvil; es decir, la metodología empleada al tratar los procesos de percepción y acción del robot.

Podemos clasificar las diferentes alternativas de la planificación de las tareas del robot atendiendo a las fases de percepción y acción del robot. Así, la percepción y acción se pueden considerar en conjunto, siguiendo un esquema de agentes cooperativos, o de forma separada, tal y como propone la Inteligencia Artificial clásica.

7.1.1 Planificación Tradicional

Los sistemas clásicos [45] [37] proponen una descomposición de la arquitectura en un conjunto de niveles de procesamiento conceptual: La interpretación sensorial, la integración sensorial, la generación de un modelo del mundo, la navegación, y el control.

Estos sistemas parten de la base de que la información proporcionada por los sensores es incompleta, pero la integración de las diferentes fuentes sensoriales en un modelo único reduce la incertidumbre. Sin embargo, tanto la incertidumbre en las medidas realizadas por los sensores como sus errores, no permiten que un módulo de percepción pasivo genere un modelo del mundo tan preciso como requieren los planificadores simbólicos.

Por otra parte, existe una imposibilidad teórica de prever todas las situaciones posibles, resultando la planificación sobre un modelo del mundo un problema NP-completo [47].

7.1.2 Planificación Reactiva

El desarrollo de robots inteligentes capaces de interactuar con un entorno dinámico ha dado lugar a una nueva alternativa a la metodología basada en la planificación sobre un modelo simbólico del mundo, que se ajusta mejor a las demandas de respuesta en tiempo real. Por ejemplo, el trabajo de *Brooks* [21] se centra en la obtención de una acción con un propósito determinado sin la necesidad de ningún tipo de planes. En su lugar, la acción inteligente se obtiene como una simple manifestación de la interacción coordinada de un conjunto de procesos simples. Aunque los planes no están representados explícitamente en estos sistemas, están, sin embargo, implícitos en la interacción pre–establecida entre los comportamientos.

Agre y *Chapman* [2] demostraron cómo un sistema que determina sus acciones mediante una continua evaluación de su situación en cada instante podía realizar tareas complejas que en otro caso hubiesen requerido de la necesidad de un planificador. *Rosenshein* y *Kaebling* [55] utilizaron un enfoque similar, con la excepción de que ellos compilaban automáticamente sus redes de decisión basándose en un formalismo lógico del conocimiento del mundo. Por su parte, *Pattie Maes* [65] implementó un sistema que trataba de emular a los organismos vivientes, con variables que indicaban su necesidad de comer, de recargar baterias, etc. Se trataba de un sistema con multitud de metas internas, activadas tanto por el estado interno del robot como por el entorno.

En todos estos sistemas, el énfasis se pone en que la acción se establece mediante el conocimiento de qué hacer en cada momento. Sin embargo, las acciones del robot están precodificadas de antemano y los objetivos no se pueden cambiar de una forma flexible por parte de un usuario.

7.1.3 Arquitecturas Híbridas

Atendiendo al intervalo temporal que abarcan los dos tipos de planificación que acabamos de descibir, se puede realizar la siguiente clasificación:

- La planificación inmediata: En un sistema reactivo las acciones a tomar por el robot están pre-codificadas: El sistema está compuesto por un conjunto de agentes, cada uno de los cuales tiene una meta diferente, a la que trata de llegar en cada instante. El entorno, las metas de cada agente, junto con el arbitraje del control pre-codificado, establecen el comportamiento global del sistema.
- Los sistema tradicionales realizan una planificación a largo plazo de un conjunto de acciones del robot, basándose en un modelo simbólico del mundo completo y preciso.

Las arquitecturas híbridas han intentado aunar los dos tipos de planificación en un único sistema jerárquico. De tal forma, el sub–sistema reactivo se encarga de las tareas básicas de supervivencia y de navegación, mientras que un sub–sistema simbólico de alto nivel planifica las tareas y trayectorias de un futuro a más largo plazo.

Un claro ejemplo de este tipo de arquitecturas es la propuesta por *Connel* [30]. Su sistema es un sistema híbrido estratificado en tres niveles diferentes. En el nivel más bajo, se generan las acciones que deben realizar los servo-controladores de los que está equipado el robot móvil. En un nivel superior, el sistema reactivo lleva a cabo las acciones generadas por el sistema de planificación de más alto nivel. En cuanto una “contingencia” (evento no previsto en el plan) se detectaba, el sistema de planificación volvía a generar un plan que debía llevar a cabo el robot, mediante la utilización de “tablas de contingencia”.

Sin embargo, las arquitecturas híbridas presentan dos problemas en su utilización, tanto de índole práctica como conceptual:

1. Desde un punto de vista práctico, en el momento en el que el plan inicial no sea correcto, se debe volver a planificar un nuevo conjunto de acciones. Por tanto, el robot (en principio) se debe parar hasta que se genere el nuevo plan. Es decir, utilizar un control reactivo para que el robot reaccione en tiempo real a eventos imprevistos en el entorno, y un sistema simbólico estático que planifique las tareas, es un contrasentido.
2. Desde un punto de vista conceptual, la transición entre el sistema reactivo y el de planificación es brusco y no natural. Es decir, en este tipo de arquitecturas se ha tratado de dar un salto cualitativo en las prestaciones de las arquitecturas reactivas dotándolas de un componente simbólico tradicional. Sin embargo, no se ha tratado de aumentar las prestaciones de los robots reactivos tratando de seguir la misma filosofía empleada en las primeras “capas” cognitivas del sistema [25].

Malcolm y *Smithers* han realizado importantes aportaciones, tanto conceptuales como prácticas, en el desarrollo de las arquitecturas híbridas [69]. Sus ideas, de especial interés por su relación con numerosos aspectos teóricos del presente trabajo, se plasman en el sistema *SOMASS*. Se trata de un robot manipulador para el ensamblado de piezas, que utiliza una arquitectura híbrida de control y que incluye un sistema de planificación PROLOG y un agente "ejecutor". El agente "ejecutor" está compuesto por un conjunto de módulos generadores de comportamientos, a los que se le asocian unos símbolos que el planificador utiliza para generar sus planes. De tal forma, el planificador se simplifica debido a una mayor competencia de los módulos generadores de comportamientos, que sirven como símbolos bien enraizados en el mundo [49].

7.2 Objetivos

Los objetivos que nos hemos propuesto para establecer una metodología en la planificación de las tareas del robot han sido:

1. Que se puedan especificar planes complejos por parte de un usuario que el robot pueda comprender sin ningún tipo de ambigüedad.
2. Que los planes no sean rígidos [82], ni que sean un conjunto de órdenes taxativas que el robot debe seguir ciegamente, sino que consistan en una serie de directrices y metas.
3. Ser coherentes con la estrategia de diseño modular empleada: Si se han implementado una serie de agentes que se encargan tanto de la navegación como de la percepción visual de entorno, tratar de seguir la misma filosofía y realizar tareas más "inteligentes" con estructuras similares.

7.2.1 Respuestas y Estructura del Capítulo

Está comúnmente asumido que es extremadamente difícil, si no imposible, que un usuario o sistema pueda producir un plan detallado en dominios de un complejidad realista. Al igual que *Agre* y *Chapman* [2], pensamos que no es necesario indicarle explícitamente al robot qué debe hacer en cada situación.

Los agentes de bajo nivel (de navegación y de visión) forman en sí mismos un sistema de planificación inmediata. El robot es capaz de detectar una serie de balizas, seguirlas visualmente y dirigirse a ellas mientras se esquivan los obstáculos que el robot encuentra en su trayectoria hacia la meta.

Sin embargo, la planificación a largo plazo de las actividades de un robot móvil implica la necesidad de indicar explícitamente, de forma flexible, un conjunto de metas a alcanzar por el robot. En nuestro caso, las metas están constituidas por un conjunto de balizas de colores a las que debe llegar el robot móvil.

Para alcanzar cada una de las metas del plan, el usuario debe proveer al robot de un conjunto de consejos (sección 7.6.1), basados en las capacidades de acción y percepción del robot (sección 7.6.2) y del conocimiento que el usuario tenga de la localización de las metas y el entorno.

Por tanto, partiendo de la base de que la actuación del robot móvil se ha generado como un conjunto de agentes, pretendemos conseguir una transición suave entre dichos agentes (de bajo nivel) y la estructura que los controla. Los agentes motores tienen asociados un conjunto de atributos (sección 7.4) que permiten a un usuario utilizar símbolos bien enraizados en el mundo real, para especificar planes complejos. Estos planes se basan en la secuenciación lineal de un conjunto de nuevos agentes, llamados agentes directores (sección 7.5), de una estructura similar a los agentes motores, pero que inspeccionan y modifican a los atributos de los agentes motores.

7.3 Atributos

Los agentes del robot móvil, tanto de navegación como de visión, poseen un elemento de control crucial: **Atributos**. Los atributos se pueden definir como variables locales que influyen en el comportamiento de un agente. En efecto, la salida de un agente depende no sólo de las lecturas de los sensores, la historia pasada y el estado actual, sino también de los atributos del agente, que desempeñan un papel fundamental en su comportamiento dinámico. Los atributos afectan a la interpretación de las medidas de los sensores, a cómo se adquiere la experiencia, y a cómo y cuándo cambia el estado del agente. En general, modificando los atributos del agente, cambiará la salida, y por consiguiente su comportamiento se modificará.

Los atributos han sido tradicionalmente considerados como constantes, y no se consideraba la posibilidad de una modificación dinámica de los mismos. Por ejemplo, umbrales para obtener e interpretar los datos que proveen los sensores, patrones de acción fija e incluso tablas de equivalencia entre entradas y la salidas son claros ejemplos de atributos en sistemas reactivos y basados en comportamientos. Los atributos han sido generalmente prefijados (o pre-cableados) en los sistemas reactivos. Sin embargo, mediante la modificación dinámica de los atributos se puede conseguir un nuevo conjunto de capacidades por parte de los agentes.

El comportamiento de un agente se parametriza por medio de los atributos. Un agente está caracterizado por la función que realiza (como por ejemplo, esquivar obstáculos o seguir una pared), y un conjunto de atributos que modifican este comportamiento. Así, el agente motor de navegación de *seguir paredes* tiene un atributo “A” en la función: *Alinearse por la “A” con la pared*, donde “A” puede ser izquierda o derecha. El comportamiento intrínseco del agente no se modifica (seguir la pared), pero el atributo determina por qué lado se alineará el robot con la pared.

La inspección y modificación de los atributos de los agentes desempeñan un papel fundamental en las prestaciones generales del robot móvil, y en la forma en la que el usuario se comunica con él. Veremos que los comportamientos y sus atributos sirven como símbolos bien enraizados con la realidad, que el usuario podrá utilizar para expresar sus planes al robot.

7.4 Comportamientos y sus atributos como símbolos

Entre las relaciones que se plantean al investigar la capacidad que tienen las lenguas para transmitir mensajes cargados de significación existe una que ha apasionado a los lingüistas y filósofos del siglo XX: se trata de averiguar qué relación existe entre la palabra como combinación de elementos fonéticos provistos de significación y el objeto de la realidad al que se refiere. El signo lingüístico es una *entidad psíquica* compuesta por dos elementos: el *significante* y el *significado:*

El significante o imagen acústica es la huella psíquica que se produce en el cerebro al oír, por ejemplo, la palabra "árbol", que se une inmediatamente con el concepto (el significado) que todos los seres humanos tienen de lo que es un árbol. Ambos elementos están íntimamente unidos y, por tanto, los signos no están aislados, sino que forman un sistema, un conjunto de relaciones, que son las que se definen los signos y las reglas que al lingüista le interesa investigar.

El problema de dar un significado a los símbolos que utiliza un robot se está convirtiendo en el objetivo de numerosos investigadores [34] [49]. La representación simbólica en un ordenador puede estar conectada al mundo mediante sistemas de entrada y salida, como los sensores y actuadores de un robot, sin embargo existen grandes dificultades para mantener una adecuada correspondencia entre el mundo del usuario (su imaginación) y el mundo real que el robot percibe y donde actúa.

Los sistemas operativos son una interfase de entrada y salida entre el programador y el hardware del sistema, sin los cuales no se podría diseñar un software dependiente de una máquina concreta. Este ejemplo es análogo a la situación que se da en el escenario cognitivo/simbólico: Sistemas simbólicos, como los sistemas tradicionales e incluso numerosas arquitecturas híbridas, no están generalmente basados en símbolos conectados con el mundo real. Aún así, los programas de aplicaciones que se ejecutan en un ordenador, tienen que tratar con un entorno más reducido y simple que aquel con el que se enfrenta un robot móvil. Este ejemplo resalta la importancia de un buen diseño inicial de los módulos de más bajo nivel que deben capturar todas las esencias de la percepción y acción en el mundo real. El desarrollo posterior de módulos de más alto nivel los utilizan como recursos de entrada y salida para interactuar con el mundo real.

Los agentes motores y sus atributos no sirven únicamente como módulos con capacidad de generar un determinado comportamiento, sino también como elementos de abstracción que permiten la existencia de una representación del mundo interno del robot móvil idealizada y manejable.

Para que la comunicación entre el usuario y el robot esté sólidamente fundamentada en el mundo real, las capacidades de estos módulos generadores de comportamiento deben ser efectivas y fiables. En nuestra opinión, igual que en una concepción puramente de abajo a arriba [26], los agentes de bajo nivel deben ser implementados en primer lugar, y basándose en cómo estos trabajan e interaccionan con el entorno, se podrán implementar nuevas aptitudes de más alto nivel.

Finalmente, hasta que no se determine cómo se "comporta" el sistema en el entorno, y cómo funcionan los diferentes módulos que componen el sistema, es harto difícil comunicarse con el robot de una forma fiable. El conjunto de tareas que el robot puede realizar determina la forma en la que los planes pueden ser expresados por parte de un usuario, y estas tareas (o, en otras palabras: comportamientos y sus atributos) dan un significado semántico al plan. Por tanto, es absolutamente necesario que el usuario tenga un conocimiento claro de los símbolos que va a utilizar a la hora de comunicar sus planes al robot.

7.4.1 Selección de Atributos para la Comunicación

Un agente del repertorio de la arquitectura de control de un robot móvil puede contener un elevado número de atributos que lo caracterizan. Se puede hacer una clasificación de los atributos dependiendo de su granularidad o amplitud de su significado.

En el grado de granularidad más bajo se encuentran los atributos escalares: $\{\alpha_1, \alpha_2, \ldots \alpha_n\}$. Por ejemplo, valores que marquen unos umbrales de las señales sensoriales, a partir de los cuales el comportamiento del agente varía. En el caso del agente de seguir paredes, unos atributos podrían ser los rangos de la Función de Accesibilidad que se asignan a las diferentes etiquetas lingüísticas.

En un nivel intermedio puede haber atributos que engloben otros atributos escalares: $\beta_k \supset \{\alpha_1, \alpha_2, \ldots, \alpha_j\}$. Es decir, la modificación de un determinado atributo de este nivel implica la modificación de toda una serie de atributos de más bajo nivel.

En el último nivel se pueden clasificar aquellos atributos que tienen asignada una etiqueta lingüística: S. El atributo del agente de seguir paredes que indica por qué lado se alinea el robot con la pared (*izquierda* o *derecha*), es un claro ejemplo. En este caso, al modificar un único atributo, el comportamiento del robot se modificará de un modo sustancial, al igual que toda una serie de atributos de nivel más bajo. Siguiendo con el ejemplo del agente de *seguir paredes*, al cambiar el atributo que indica por qué lado se debe alinear el robot con la pared, cambia toda una serie de atributos de granularidad mas fina, cómo los rangos de la Función de Accesibilidad respecto a los cuales se definen las etiquetas lingüísticas arriba, abajo ... que, a su vez, inciden en el control final.

Por tanto, para comunicarse con el robot se ha de decidir la granularidad del mensaje que se le va a enviar, es decir, si se opta por una comunicación basada en la definición de unos valores escalares que modificarán el comportamiento del agente, o se prefiere una granularidad más gruesa en la que los parámetros engloban a otros de menor nivel.

La forma y filosofía que se pretenda seguir a la hora de planificar la tarea del robot establece en cierta medida la granularidad de la comunicación. Hay autores que prefieren establecer a-priori todos los parámetros de más bajo nivel de los diferentes agentes [80]. En este caso, la comunicación se hace difícil y engorrosa, pues el usuario debería conocer toda la serie de parámetros y variables de bajo nivel del robot móvil, y cómo se modifica su comportamiento al modificar estos atributos. Sin embargo, si se establece una granularidad gruesa la comunicación con el robot se hace sensiblemente más sencilla y se asemeja a los mensajes entre personas. La comunicación gruesa establece una serie de comportamientos generales mediante etiquetas asignadas a los agentes y los modula a través de la modificación de sus atributos simbólicos de más alto nivel. El diseñador del sistema debe establecer los valores óptimos de los parámetros de control; éstos serán los valores que tomen por defecto y pueden modificarse dependiendo a lo máximo de indicaciones generales.

Por tanto el usuario, a la hora de comunicar sus deseos al robot, podrá hacer uso de un conjunto de símbolos que tienen significado en el mundo real. Esto se garantiza por un diseño del sistema de abajo a arriba: Los agentes motores de navegación se diseñan, implementan y comprueban de una manera jerárquica. Cada agente se comporta de forma muy determinada (evitar obstáculos, seguir paredes, seguir objetos visuales, etc.), y se le asigna un símbolo, que describa su comportamiento y sea comprensible para el usuario, y unos atributos de alto nivel, que incrementan el repertorio simbólico mediante el cual el usuario puede definir sus planes.

7.4.2 Comportamientos Básicos de Comunicación

Para establecer el conjunto de símbolos y atributos que se asignarán a los agentes motores, se deben tener en cuenta los criterios siguientes:

1. Debe existir una base canónica de símbolos que permita la elaboración de planes sofisticados por parte del usuario.

2. El significado de los símbolos debe ser el mismo para el robot que para el usuario.

Algunos autores [75] han propuesto una serie de criterios para definir una base canónica de comportamientos elementales del robot. En cualquier caso, en la implementacion de todo sistema existe una fase de diseño previa, donde se especifican los objetivos y cualidades del sistema; es decir, existe una imposición de arriba a abajo por parte del diseñador que influye directamente en el conjunto de agentes con los que va equipado el robot. En este sentido, en el presente trabajo, se decidió dotar al robot de la habilidades siguientes:

- evitar obstáculos.
- detectar una serie de objetos.
- ir a las posiciones en las que se encuentran estos objetos.

Para realizar este tipo de tareas es necesario un conjunto de agentes que generan dichos comportamientos. Parte de estos agentes servirán como símbolos que el usuario pueda entender y manipular, y otros no (un ejemplo de estos últimos es el agente motor de navegación *Giro Propio*). Tampoco se asignará un símbolo a los comportamientos emergentes del sistema [86], ya que no fueron diseñados e implementados explícitamente, sino que surgen de la interacción de los comportamientos motores.

Por otra parte, elementos del entorno, como paredes o las balizas que debe detectar el robot, tienen que ser caracterizados sin ningúna ambigüedad; si se indica al robot que detecte, siga y se dirija a una determinada baliza, éste debe localizar ese hito y no otro.

7.4.3 Selección de Comportamientos

Agentes Motores de Navegación		Agentes Motores de Visión	
Comportamiento	**Atributo(s)**	**Comportamiento**	**Atributo(s)**
Avoid		Track	
Follow Wall	Left	Detect	Blue in Red
	Right		Blue in Green
Goto Point	Reached		Red in Blue
			Red in Green
			Green in Red
			Green in Blue
		Search	Forward
			Left
			Right
			Everywhere

Tabla 7.1: Agentes Motores y sus Atributos

En la Tabla 7.1 se puede observar la selección de agentes motores y de sus atributos que forman una base canónica para la comunicación usuario-robot.

Todos los agentes motores tienen un conjunto de atributos comunes que no están especificados en la Tabla 7.1. Como se especificó en el capítulo 2, los agentes motores de navegación pueden estar *activos* o *inactivos*. La actividad de un agente motor de navegación influye en cómo éste modula la información de agentes superiores, y en su participación activa en la respuesta del robot. Por su parte, la actividad o inactividad de los agentes motores de visión indica el estado del sistema. Por ejemplo, si el agente de *seguimiento* de un objeto está activo, indica que una determinada baliza ha sido detectada y reconocida y que en ese preciso momento el sistema está haciendo un seguimiento visual de la misma.

A continuación se repasarán los diferentes símbolos asociados a los agentes y sus atributos, presentes en la Tabla 7.1.

Agentes Motores de Navegación

Los agentes motores de navegación que se han elegido para la comunicación entre el usuario y el robot son:

- **Avoid**: Esquivar los obstáculos que el robot encuentre en su trayectoria.
- **Goto Point**: Ir a una posición del entorno. Un atributo booleano (*Reached*) indicará si el agente ha conseguido llegar a la meta.
- **Follow Wall**: Seguir paredes. A la hora de seguir las paredes el robot se podrá alinear con la misma por la izquierda (*Left*) o la derecha (*Right*).

Otros agentes, como *Giro Propio* o el definido como *Ir a un Punto Visual*, no están incluidos en la base de símbolos de comunicación con el usuario. El agente *Giro Propio* existe únicamente para evitar que el robot se quede en situaciones de *dead-lock*, es decir, "estancado". El usuario no necesita conocer este tipo de mecanismos de escape de los que va equipado el sistema para proponer un plan al robot, aunque éste debe ser capaz de reconocer tales circunstancias y reaccionar ante ellas.

La acción del agente *Ir a un Punto Visual* no difiere de la acción de ir a una posición determinada. Su existencia se justifica en la necesidad de proveer una interfase entre los agentes motores de navegación y los de visión. Este agente actúa mientras el agente *Goto Point* no conozca explícitamente la posición donde debe dirigirse, localización que, por otra parte, debe proveer el agente de visión *Stereo*, que calcula la distancia a la que se encuentra el objetivo del robot.

Agentes Motores de Visión

La comunicación entre el usuario y el robot puede hacer referencia a los símbolos asignados a los agentes motores de visión siguientes:

- **Track**: Comprobando si el agente está *activo*, se puede saber si el sistema visual ha detectado y reconocido un determinado objeto visual y lo está siguiendo.
- **Detect**: Detectar una baliza concreta, determinada por el atributo.
- **Search**: Dirigir la búsqueda visual en cierta dirección (*Forward, Left, Right, Everywhere*).

El agente de visión, *Stereo*, no tiene ningún símbolo asociado ya que su activación es permanente. Este agente determina la distancia a la que se encuentra el objetivo, y proporciona dicha información al agente *Goto Point*.

7.5 Agentes Directores

Los agentes directores inspeccionan y modifican los atributos de los agentes motores, tanto de visión como de navegación. Por tanto, no necesitan entradas de los sensores, y sus salidas no afectan directamente a los actuadores del robot móvil. Los agentes directores son simples reglas de control que invocan a los agentes motores y sus atributos como símbolos, monitorizan los atributos de los agentes motores y los modifican cuando se satisface una serie de condiciones previas.

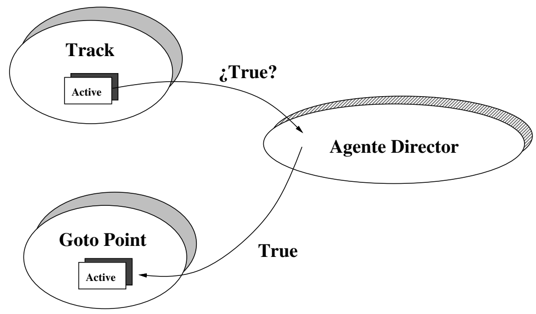


Figura 7.1: Ejemplo de Agente Director.

En la Figura 7.1 se puede observar como un agente director monitoriza si el agente *Track* está activo. En cuanto lo esté, el agente director activará al agente *Goto Point*.

Un agente director se describe formalmente como una regla (AD_i), con un conjunto de antecedentes ($\{A_n^i\}$) y un conjunto de consecuentes ($\{P_m^i\}$):

$$AD_i : A_0^i, A_1^i, \ldots, A_n^i \rightarrow P_0^i, P_1^i, \ldots, P_m^i \quad (7.1)$$

Los antecedentes son estados del sistema, descritos en términos de los atributos de los agentes motores. Los consecuentes son acciones que modifican los valores de los atributos de los agentes motores.

Los agentes directores pueden ser de dos tipos: **Perpetuos** o de **contingencia**. Los agentes directores perpetuos estarán siempre activos, monitorizando las precondiciones y ejecutando las pos-condiciones siempre que se satisfagan los antecedentes. Los agentes directores de contingencia aplican las pos-condiciones cuando las pre-condiciones han sido satisfechas una única vez; en otras palabras, la primera vez que las pre-condiciones de un agente conductor de contingencia han sido satisfechas, las pos-condiciones se ejecutarán y el agente motor ya no estará activo.

7.6 Especificación del Plan

7.6.1 Planes como "Consejos"

Nuestro sistema permite dar al robot una serie de *consejos* útiles para realizar la tarea encomendada, teniendo en mente (por parte del usuario) las habilidades de percepción y acción

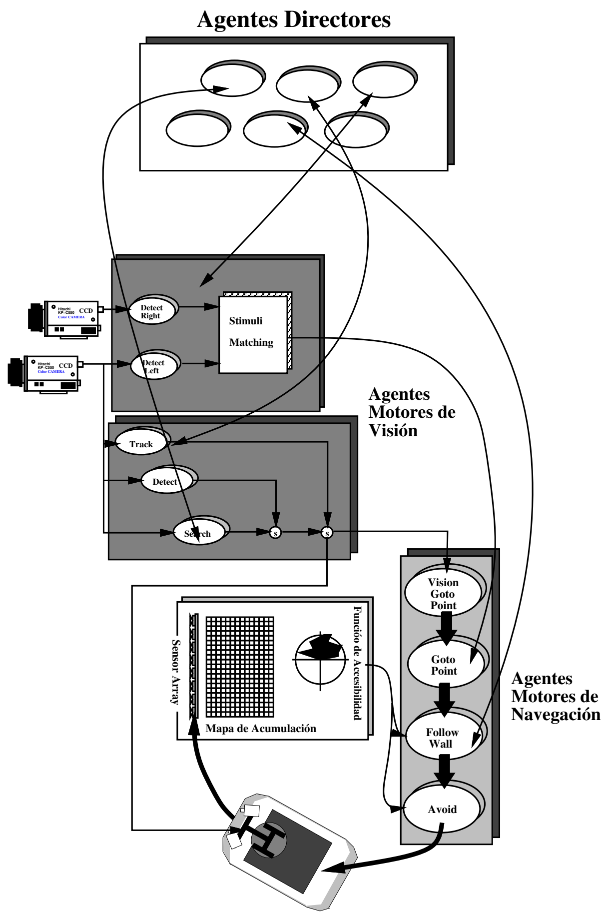


Figura 7.2: Agentes Directores: Los agentes directores inspeccionan los atributos de los agentes motores y los modifican.

del robot, como, por ejemplo, seguir paredes, evitar obstáculos o detectar un determinado estímulo visual. De esta forma, la comunicación entre el usuario y el robot se parece a la forma en que una persona le explica a otra cómo llegar desde un lugar a otro: *"Una vez que hayas cruzado la puerta, sigue el pasillo hasta que encuentres la entrada principal ..."*. Por tanto, el usuario debe conocer lo que el robot es capaz de hacer y de percibir para elaborar el plan de la misión. Este es uno de los puntos en los que trata de incidir esta tesis: que el usuario sepa "cómo" comunicarse con el robot, en lugar de imponer una serie de habilidades de arriba a abajo al robot móvil. Es decir, no construir inicialmente el sistema mediante el cual se comunicarán los planes al robot (el planificador simbólico en sistemas tradicionales), sino plantearse la comunicación con el robot una vez que se conoce su modelo cognitivo (agentes, atributos y símbolos asociados a ellos).

Los consejos son un recurso para que el robot mejore su rendimiento global. Los consejos son válidos durante un periodo de tiempo: Por ejemplo, se puede indicar al robot que un objetivo señalado por una baliza está a su derecha si está siguiendo una pared con la cual está alineado a la izquierda. El robot utilizará este consejo para dirigir su atención en esa dirección, pero en cuanto el objetivo haya sido localizado, este consejo dejará de ser útil para decisiones futuras, de manera que si el objetivo, después de ser localizado, se pierde momentánemente, los agentes motores de visión procesarán la información sensorial y sabrán dónde buscar haciendo uso de la historia pasada, pero no del consejo dado por el usuario.

Introduciendo consejos en la elaboración de un plan, el usuario no ha de concentrase en detalles de bajo nivel, ni en las estrategias para resolver eventos imprevistos, como qué hacer cuando se ha perdido el objetivo o cómo evitar los obstáculos que pueda haber en el camino a la meta. Los símbolos asociados a los agentes motores y sus atributos servirán como base simbólica para introducir los consejos del usuario; el repertorio de agentes motores del robot debe ser capaz de seguir estos consejos así como de resolver todo tipo de imprevistos.

7.6.2 Referencias Perceptibles

En general, un sistema de planificación descompone un plan en diferentes sub-tareas. En concreto, nuestro sistema utiliza los consejos aportados por un usuario para llegar a un conjunto de metas intermedias. Estos consejos se basarán en referencias a elementos del entorno que tanto el robot como el usuario pueden percibir a través de sus sensores. Estas referencias son el nexo de unión entre el intelecto del usuario y las capacidades perceptivas y de comportamiento del robot, y deberán ser estables en el tiempo, robustas, no ambiguas y fáciles de detectar por el robot.

Referencias como "izquierda", "derecha" o "detrás" pueden ser ambiguas para el robot. Por ejemplo, en un supuesto plan, se le puede indicar al robot que tras finalizar uno de los pasos, se dirija a la izquierda; sin embargo, es difícil que el usuario pueda anticipar de antemano la posición relativa entre el robot y la sub-meta de una forma tan específica como para que la zona considerada como izquierda sea la que se pensó antes de que llegase el robot a esa localización.

Por tanto, las referencias que unen los pasos (o sub-metas) deben estar basadas en referencias que se puedan percibir y que no dependan de la orientación relativa entre el robot y dicha sub-meta.

Las paredes en entornos interiores cumplen todas las condiciones que las referencias han de satisfacer: Son estables en el tiempo, pueden ser detectadas de una forma robusta y no son ambiguas ni para el usuario ni para el robot. Las paredes sirven como referencias robustas entre el usuario y el robot. De esta forma, la posición de objetivos se pueden proporcionar haciendo referencia a la pared. Por ejemplo, se puede indicar al robot que *"el siguiente objetivo está a la derecha si te alineas con la pared a la izquierda"*.

7.6.3 El Plan

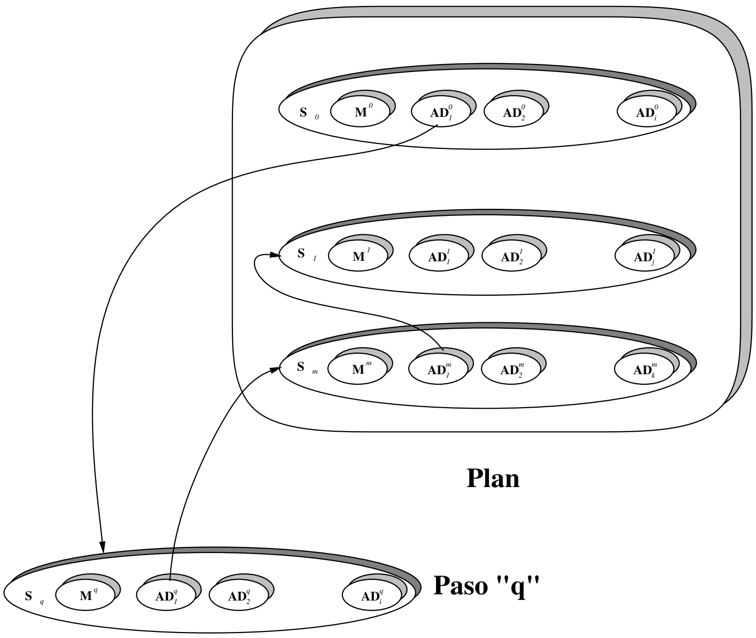


Figura 7.3: Plan: El Paso S_q toma el control de los agentes motores. Las flechas indican las transiciones entre pasos dentro del plan.

Hasta el momento se ha descrito un conjunto de agentes motores (de navegación y de visión), a los cuales se puede asociar una serie de símbolos bien fundamentados en el mundo real, que servirán para que el usuario pueda establecer planes como un conjunto de consejos. En la presente sección se aúnan los conceptos descritos en secciones anteriores y se definen estructuras computacionales mediante las cuales un usuario define los planes. Por tanto, se comenzará definiendo lo que es un plan en el contexto de este trabajo, y se describirá la metodología que deberá seguir un usuario a la hora de comunicarse con el robot móvil.

Formalmente, un plan P está compuesto por una serie de pasos, o sub-tareas, a seguir:

$$P = \{S_0, S_1, \ldots, S_n\}$$

A su vez, cada uno de los pasos está caracterizado por una **meta** que debe lograr, y un conjunto de "consejos" para llegar a ella. En un mismo paso no deben coexistir dos metas que puedan dar lugar a planes alternativos.

$$S_i = \{M^i, AD_0^i, \ldots, AD_m^i\}$$

Tanto la meta (M^i) como los consejos (AD_j^i) estarán "codificados" como agentes directores, que inspeccionan y modifican los atributos de los agentes motores del sistema. El usuario tiene toda una serie de símbolos asociados a los agentes motores y sus atributos, con los cuales puede expresar las metas y los consejos de cada uno de los pasos de los que está compuesto el plan.

Concretamente en el sistema implementado, un paso suele ser llegar a la posición de una baliza. Además, existirá un determinado agente director que inspeccionará si se ha llegado a la meta propuesta en el paso. Al estar la condición de meta del paso codificada como un agente director M^i, éste se encargará de emitir la señal de activación de un nuevo paso del plan. Es decir, los agentes directores monitorizan si el robot ha llegado a la meta propuesta en cada uno de los pasos y se encargan de la secuenciación entre los diferentes pasos del plan.

Finalmente, en la transición de un paso S_i a otro S_j, todos los agentes motores del sistema se desactivan, siendo los agentes directores del paso S_j, que toman el control del sistema los encargados de activar y particularizar los agentes motores que sean necesarios para la consecución de la meta asociada a ese paso.

Resulta interesante resaltar el hecho de que cabría la posibilidad de incluir un conjunto de metas dentro de un mismo paso que diesen lugar a planes alternativos. En este caso se trataría de que el sistema pudiese aprovechar en cada instante la situación para llegar antes al objetivo final. Sin embargo, y debido a como está constituido el sistema de visión artificial que sólo sigue un único estímulo, este tipo de alternativas no se pueden llevar a cabo. Una posible extensión de este trabajo podría ser la activación simultánea de toda una serie de agentes motores de visión que estuviesen atentos a diversos estímulos del entorno (un conjunto de balizas) y, dependiendo del plan expresado por el usuario, pudiesen "aprovechar" la situación y optimizar el desarrollo del plan para alcanzar antes la meta.

7.6.4 La Gramática

Una vez que se ha descrito formalmente un plan como una consecución de pasos, cada uno de los mismos compuesto por un conjunto de consejos y de metas expresados como agentes directores, se debe proveer al usuario de una herramienta mediante la cual pueda generar dichos planes. Para ello se ha desarrollado una gramática muy simple (véase la Figura 7.4), expresada mediante una sintaxis normal de Bakus (BNF), que le permite al usuario escribir planes que el robot sea capaz de comprender y llevar a cabo.

Plan `<Plan>`**:** El plan está compuesto por un conjunto de pasos (`<Step>`), teniendo que estar especificado al menos un paso.

```
<Plan>             = <Step>{<Step>}.
<Step>             = "BEGIN" "n" {<C-Behavior>} "END".
<C-Behavior>       = <C-Behavior Type> "{" "i" <Cond>{<Cond>}
                                          "j" <Cons>{<Cons>} "}".
<Type>             = "ONCE" | "IF".
<Cond>             = "TRUE" | <Flag> | <Attribute Quest>
<Flag>             = "REACHED" | "TRACKING".
<Attribute Quest> = <Quest><Behavior>.
<Quest>            = "ENABLED" | "NOT_ENABLED".
<Cons>             = {<Mod><Behavior> "n" {<Attribute>}}
                     | {"NEXTSTATE =" "m"}.
<Mod>              = "ENABLE"| "DISABLE".
```

Figura 7.4: Especificación de la gramática del lenguaje mediante una sintaxis BNF

Pasos <Step>**:** Cada paso del plan está delimitado por las palabras clave BEGIN y END. Un número n, a la derecha de la palabra clave BEGIN indica el número de orden del paso dentro del plan. Dentro de cada paso se engloban los agentes directores que lo forman.

Agentes Directores <C-Behavior>**:** La gramática permite la utilización tanto de agentes directores perpetuos (IF) como de contingencia (ONCE). Cada agente director puede tener i pre-condiciones y j pos-condiciones.

- La parte de la pre-condición (<Cond>), del agente director, puede ser TRUE, un indicador <Flag>, o bien la inspección de un atributo de un agente motor (<Attribute Quest>).
  - Si la condición es TRUE significa la ausencia de condiciones reales para proceder con las acciones del consecuente. Será especialmente útil a la hora de iniciar a los agentes motores en el inicio de un nuevo paso.
  - Los "Flags" indican el estado de determinados agentes y de sus atributos. Así, si la condición REACHED es verdadera, indicará que el el agente de ir a un punto (*Goto Point*) ha alcanzado su meta. Si la condición REACHED es falsa indicará, por tanto, que el agente aun no ha alcanzado la meta.
  - TRACKING indicará si el sistema de seguimiento visual está siguiendo un objeto.
  - La inspección (<Quest>) de un atributo de un agente (<Behavior>) se reduce a la posibilidad de ver si éste está activo (ENABLED) o inactivo (NOT_ENABLED).
- El consecuente (<Cons>) del agente director puede activar (ENABLE), o desactivar (DISABLE) un determinado agente motor.
  - Si se activa un agente motor mediante la utilización de la palabra clave ENABLE, se debe especificar el número n de atributos que se desean modificar, y el símbolo

`<Attribute>` asociado al mismo. Por ejemplo, si se desea especificar un consejo mediante el cual se le dice al robot que la próxima baliza debería encontrarse a la izquierda del mismo, se debe definir una pos-condición del tipo:

```
ENABLE SEARCH 1 LEFT
```

Uno de los agentes directores será el encargado de verificar si se ha llegado a la meta encomendada al paso en el que está incluido y de pasar el control al siguiente paso del plan. Esto último se hace mediante la palabra clave `NEXTSTATE`, donde se indica cuál es el siguiente paso (`m`) del plan que se debe ejecutar. Si se especifica un número de paso que no está en la definición del plan, el robot se quedará quieto y no reaccionará ante ningún estímulo exterior, ya que la condición previa a la ejecución de un nuevo paso es la desactivación de todos los agentes motores del sistema.

Evaluación de las condiciones

Hemos visto que los "consejos" del plan se implementan mendiante agentes directores para que el robot alcance la meta especificada. Para que se ejecuten las pos-condiciones de un agente director se deben satisfacer **todas** las pre-condiciones del agente en cuestión.

Ejemplo de un agente director

```
ONCE { 1 TRUE
    3
    ENABLE FOLLOW_WALL 1 RIGHT
    ENABLE DETECT 1 RED_IN_BLUE
    ENABLE SEARCH 1 FORWARD
}
```

- La palabra clave "`ONCE`" indica que este agente director es de contingencia. Solo hay una pre-condición (`1 TRUE`), que indica que el agente director ejecuta de inmediato las acciones de la parte consecuente.
- El agente director contiene `3` consecuentes:
- `ENABLE FOLLOW_WALL 1 RIGHT`, Inicia el agente motor de navegación *Follow Wall*, de tal forma que el robot se alinee con la pared por la derecha (`RIGHT`).
- `ENABLE DETECT 1 RED_IN_BLUE`, Inicia el agente motor de visión *Detect*, de tal forma que el sistema visual del robot trate de detectar la baliza compuesta por un cuadrado azul inscrito en uno rojo (`RED_IN_BLUE`).
- `ENABLE SEARCH 1 FORWARD`, Indica al robot que busque preferentemente en la dirección frontal del robot (`FORWARD`).

7.7 Sentido Común

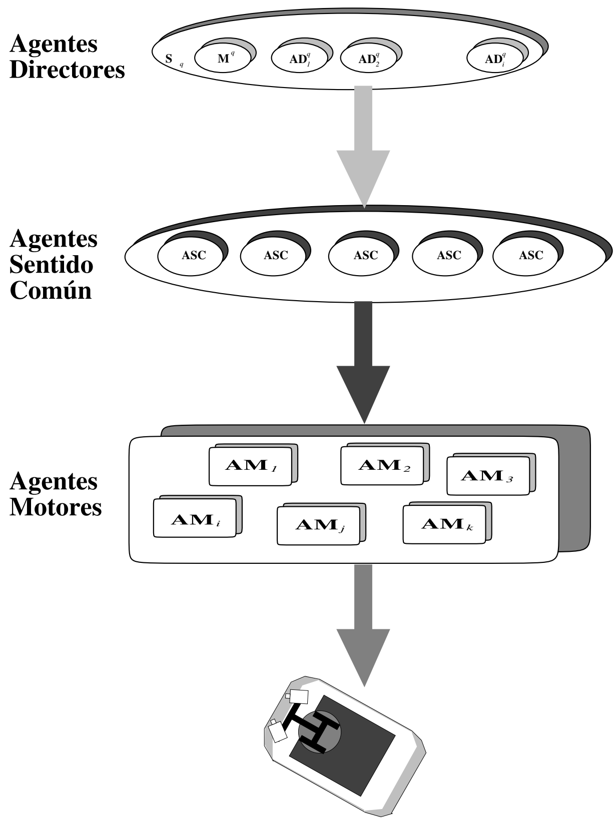


Figura 7.5: Jerarquía de los diferentes agentes (motores, sentido común y directores) del sistema.

En principio, además del conocimiento sobre los agentes, los atributos y los símbolos asociados a ellos, el usuario debería conocer cuestiones relacionadas con el funcionamiento interno del sistema. Así, debería ser necesario indicarle al robot que dejase de seguir una pared mientras estuviese dirigiéndose a un punto, pues en la arquitectura de control los agentes motores de navegación *Follow Wall* y *Goto Point* están en un mismo nivel de jerarquía, y la activación de un determinado agente implica la desactivación del otro. Esto habría que indicarlo explícitamente en el plan que el usuario provee al robot, y dificulta, en cierta medida, la sencillez en la comunicación entre el usuario y el robot. Para conservar la sencillez en la especificación de los planes, se ha implementado una serie de agentes directores, denominados **agentes de sentido común**. estos agentes, aunque no están explícitos en los planes que escribe el usuario, están siempre controlando qué agentes motores deben estar activos en cada momento. En el ejemplo citado, un agente director perpetuo desactiva el agente de seguir paredes cuando el agente de ir a una localización esté activo. Así, el usuario escribe un consejo como:

```
IF { 1 TRACKING
1
ENABLE GOTO_POINT 0
}
```

en vez de:

```
IF { 1 TRACKING
4
ENABLE   AVOID 0
ENABLE   GOTO_POINT 0
ENABLE   VISION_GOTO_POINT 0
DISABLE  FOLLOW_WALL 0
}
```

Los agentes de sentido común se denominan así porque tratan de emular los pensamientos implícitos de un usuario. Un ser humano considera que no es necesario que cuando le indica a alguien cómo llegar de un lugar a otro, le tenga que decir que vaya evitando los obstáculos. De la misma manera, agentes conductores perpetuos monitorizarán los agentes motores para activarlos o desactivarlos según sea el caso. Por ejemplo, un agente conductor será el responsable de monitorizar si el agente de ir a una localización está activo, en cuyo caso el comportamiento de evitar obstáculos se activaría automáticamente.

Los agentes de sentido común actúan como un árbitro entre los agentes motores que permiten que el usuario asuma que el robot tiene cierto “sentido común” a la hora de comunicarse con él. Cuestiones obvias, como evitar los obstáculos mientras el robot se mueve hacia la meta, o desactivar el comportamiento de seguir paredes cuando el robot se dirige a un punto del espacio, están controladas por el sentido común del robot móvil. Por tanto, los agentes conductores de sentido común permiten una interacción más fácil y flexible entre el usuario y el robot. El usuario no tiene que incluir una serie compleja de agentes directores que tengan en cuenta todos los aspectos involucrados en la arquitectura del robot móvil.

A pesar de todo, el usuario puede hacer un pleno uso de su libertad personal e ir en contra del “sentido común” del robot móvil. El usuario es el dueño y señor de las actividades del robot móvil, y los deseos expresados en sus planes son órdenes directas para el robot móvil. Por tanto, desde un punto de vista de la jerarquía general del sistema, los deseos del usuario estarán por encima de los agentes de sentido común del robot móvil, y el usuario podrá expresar planes tan absurdos como él quiera.

7.8 Comportamiento del Sistema

Para observar el comportamiento de los agentes motores del sistema en los diferentes pasos de un plan, se definió un plan muy simple (véase Plan 1), que se puede resumir en las siguientes frases:

Detectar e ir a la baliza que se compone de un cuadrado azul incluido dentro de uno rojo, que probablemente esté en la dirección de avance. Desde baliza, seguir la pared

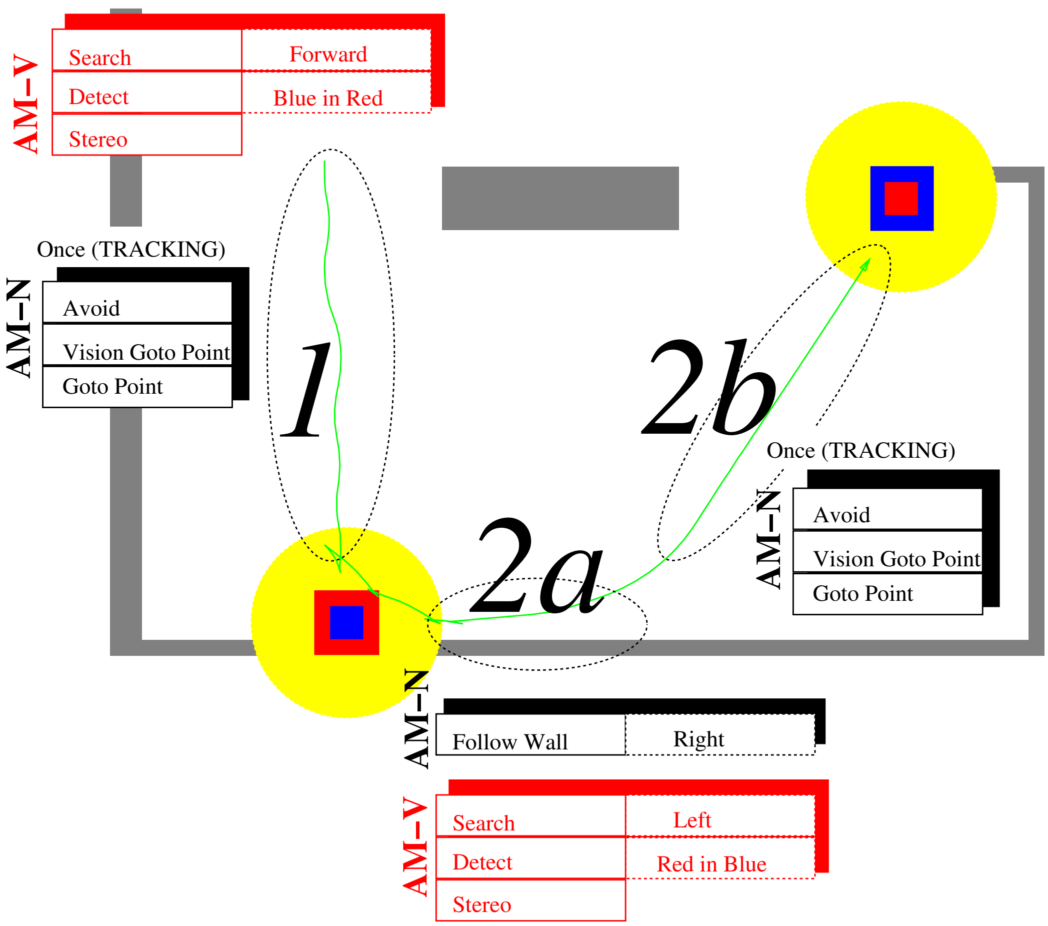


Figura 7.6: Desarrollo dinámico.

```
BEGIN 1
        ONCE {
        1 TRUE 2
        ENABLE SEARCH 1 FORWARD
        ENABLE DETECT 1 BLUE_IN_RED
        }
        ONCE {
        1 TRACKING 1
        ENABLE GOTO_POINT 0
        }
        IF {
        1 REACHED 1
        NEXTSTATE 2
        }
END

BEGIN 2
        ONCE {
        1 TRUE 3
        ENABLE FOLLOW_WALL 1 RIGHT
        ENABLE SEARCH 1 LEFT
        ENABLE DETECT 1 RED_IN_BLUE
        }
        IF {
        1 TRACKING 1
        ENABLE GOTO_POINT 0
        }
        IF {
        1 REACHED 1
        NEXTSTATE 3
        }
END

BEGIN 3
END
```

Plan 1: Plan para realizar la tarea de la Figura 7.6

por la derecha y buscar a la izquierda la baliza del cuadrado rojo incluido dentro del azul. Al llegar a esta baliza, parar.

En definitiva, el plan consta de dos pasos claramente diferenciados por las dos balizas, que marcan la meta de cada uno. En los siguientes apartados se detallarán la manera en que se ha de codificar el plan y la reacción del sistema.

La trayectoria seguida por el robot móvil (véase Figura 7.6) se obtuvo de los datos de los sensores odométricos. La situación de las paredes y balizas se superpuso posteriormente, pero trata de reproducir fidedignamente el entorno de experimentación.

En la Figura 7.6 se pueden observar además los agentes motores activos en cada instante de la consecución de los diferentes pasos del plan y sus respectivos atributos. Los agentes motores de navegación están encuadrados y etiquetados como **AM-N** y los de visión como **AM-V**. También se han incluido las condiciones que monitorizan los agentes directores (en este caso: Once(Tracking)), sobre el conjunto de agentes motores. Finalmente, las circunferencias centradas sobre las balizas indican la zona a partir de la cual el agente de ir a un punto determina que se ha llegado a la meta.

7.8.1 Primer paso

La trayectoria seguida por el robot móvil en el primer paso del plan está enmarcada en la Figura 7.6 dentro del óvalo 1. En este paso, el sistema debe detectar la baliza compuesta por un cuadrado azul dentro de un cuadrado rojo, que mediante la sintaxis definida se expresa como:

`ENABLE DETECT 1 BLUE_IN_RED`

Inicialmente, el usuario que especificó el plan conocía la posición relativa entre el robot y la primera baliza. Por tanto, le indicó al robot que debía buscar hacia adelante (`SEARCH 1 FORWARD`). En el caso de que se hubiese desconocido por completo la orientación relativa entre el robot y el primer objetivo, se debería de aconsejar al robot que buscase en todas direcciones, mediante el atributo: `SEARCH 1 EVERYWHERE`.

La detección de una baliza determinada, en la dirección de búsqueda se debe realizar inmediatamente. Por tanto, se utilizó un agente director de contingencia, cuya única precondición se satisface siempre: `ONCE 1 TRUE`.

En definitiva, el primer consejo del plan se expresa mediante el siguiente agente director:

```
ONCE {
1 TRUE 2
ENABLE SEARCH 1 FORWARD
ENABLE DETECT 1 BLUE_IN_RED
}
```

En cuanto el sistema visual detecte la baliza, el robot se debe dirigir a la posición donde se encuentra. Por tanto, se incluye en el paso el agente director:

```
ONCE {
1 TRACKING 1
ENABLE GOTO_POINT 0
}
```

que activa al agente motor *Goto Point* en cuanto se activa el agente de seguimiento visual *Track*. En este sentido, es importante destacar el detalle de que es completamente análogo inspeccionar si el sistema de visión está siguiendo un objeto mediante la expresión `ENABLED 1 TRACK` que mediante la inspección del flag `TRACKING`.

Finalmente, la meta de este paso del plan es llegar a la baliza. Por tanto, en cuanto el agente motor de ir a una localización detecte que ha llegado a la posición donde se encuentra la baliza, se pasaría al siguiente paso del plan (el número 2):

```
IF {
1 REACHED 1
NEXTSTATE 2
}
```

7.8.2 Segundo Paso

Para observar con detalle el resultado de la activación de los agentes directores que codifican los consejos del paso, en la Figura 7.6 la trayectoria del segundo paso se ha dividido en dos zonas bien diferenciadas (2a y 2b).

En la primera parte (2a), se aconseja al robot que siga la pared por la parte derecha `FOLLOW_WALL 1 RIGHT`, tratando de detectar la baliza `DETECT 1 RED_IN_BLUE` a la izquierda `SEARCH 1 LEFT`. Es interesante observar en la trayectoria del robot móvil, cómo cerca de la baliza, al comenzar el segundo paso, da marcha atrás para tratar de alinearse con la pared.

En cuanto el sistema visual detecta la baliza, el robot deja de seguir la pared y se dirige a la baliza (señalado como la zona 2b).

7.8.3 Tercer Paso

En la definición de todo plan se puede incluir un último paso, definido como el *paso vacío*. Tal y como ya se describió previamente (véase sección 7.6.3), cuando se inica un nuevo paso de un plan todos los agentes motores del sistema se desactivan. Por tanto, si tras la consecución del último paso del plan se desea que el robot se quede parado y sin ningún tipo de actividad, se puede incluir un paso desprovisto de agentes directores:

```
BEGIN 3
END
```

7.8.4 Agentes de Sentido Común

Aunque en el segundo paso del plan (véase Plan 1) no hay ninguna orden explícita de dejar de seguir la pared, al detectarse la baliza el robot deja de seguir la pared. Los agentes directores de sentido común desactivarán al agente de seguir paredes al observar que el agente de ir a una localización (*Goto Point*) está activo. Los agentes directores de sentido común permiten que el usuario no tenga que escribir pasos excesivamente prolijos. En este sentido, tanto en el primero como en le segundo paso del plan, se omitió que el robot evitase los obstáculos que encontrase en su camino hacia la meta, pues la mera activación del agente motor *Goto Point* implica la activación automática del agente *Avoid*. A su vez, y en el segundo paso del plan, los agentes de sentido común desactivan automáticamente al agente de seguir paredes en el momento en que el robot se dirige a la meta,

7.9 Otros Ejemplos

Las Figuras 7.7, 7.8, 7.9 y 7.10 muestran la trayectoria seguida por el robot móvil en la consecución de una serie de planes.

Por otra parte, en las Figuras 7.7, 7.8, y 7.10 se ha incluido un sistema de referencia métrico (expresado en centímetros) que permite inferir las distancias recorridas por el robot, así como la localización espacial de las balizas.

La Figura 7.7 muestra la trayectoria seguida por el robot, que cumplía un plan que le indicaba que se dirigiese a dos balizas. En este caso, la pared en la cual se encontraban estas dos balizas sirvió como elemento de referencia al robot para localizar los objetivos finales.

La Figura 7.8 muestra la trayectoria de un plan complejo, en el que el robot debía dirigirse a la posición de cuatro balizas, utilizando consejos de por qué lado alinearse con la pared, y por dónde buscar el siguiente objetivo.

La condición final de un plan no ha de ser siempre que el robot se quede parado en una determinada baliza. El plan puede ser cíclico, con lo que se repiten determinados pasos del mismo. Las Figuras 7.9 y 7.10 muestran el resultado de planes “infinitos”, en los que el robot realiza de forma secuencial todos los pasos del plan.

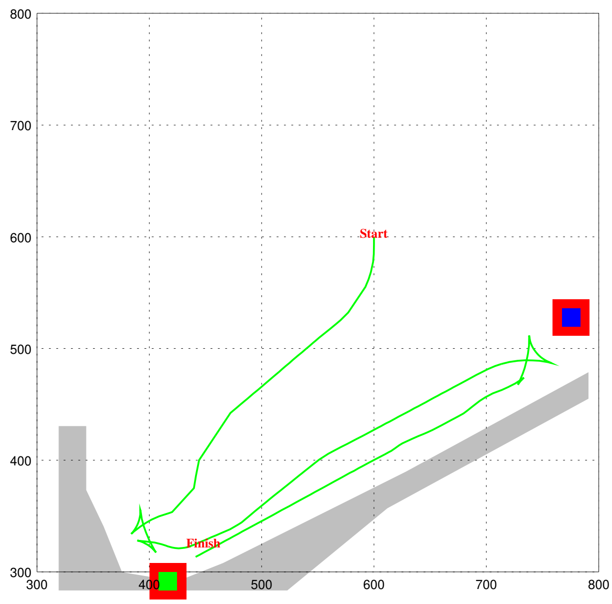


Figura 7.7: Dos balizas de ida y vuelta.

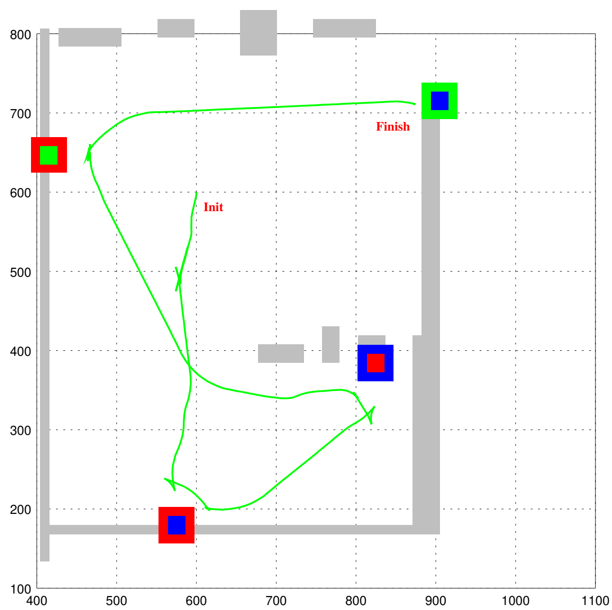


Figura 7.8: Cuatro balizas.

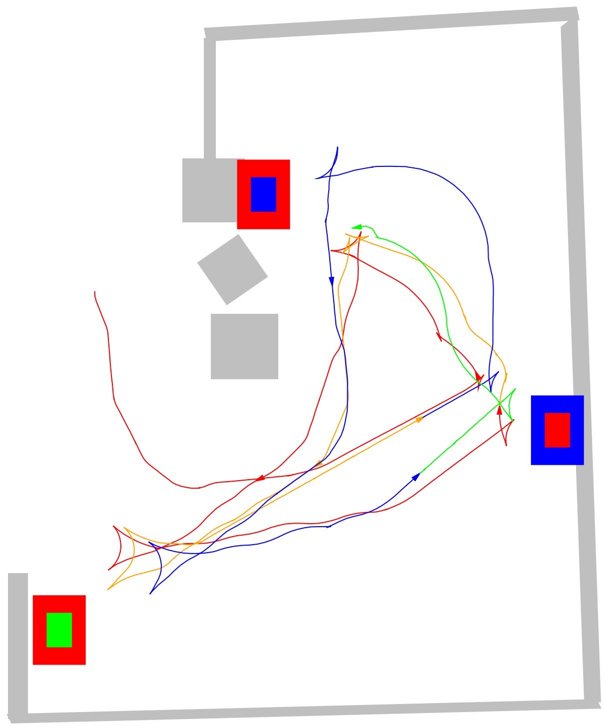


Figura 7.9: Tres balizas cíclico

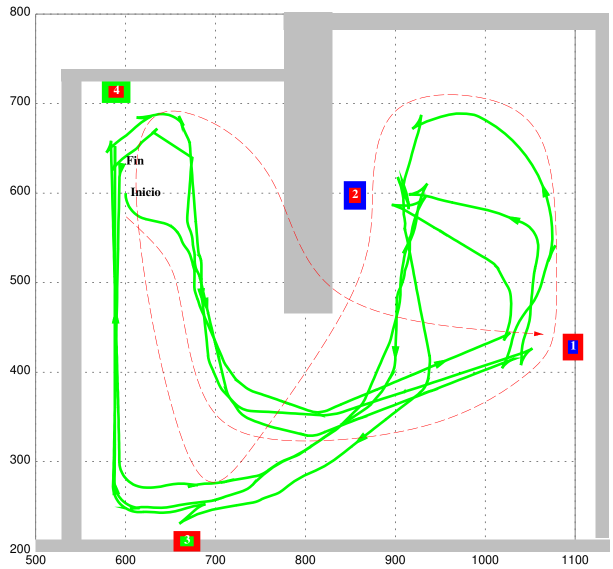


Figura 7.10: Cuatro balizas cíclico.

Capítulo 8

Conclusiones

8.1 Aportaciones

Las aportaciones de esta tesis han sido:

1. El diseño y la implementación de la arquitectura completa de un robot móvil. Se han tenido en cuenta tanto los sistemas de bajo nivel (navegación, visión), así como la planificación de alto nivel.

2. Se ha definido e implementado una arquitectura multi–agente para los agentes motores de navegación, con las siguientes propiedades:

   - Arquitectura jerárquica y modular.
   - Respuesta en tiempo real.
   - Facilita la incorporación de nuevos agentes a la estructura del sistema.
   - Adapta las acciones de control de los agentes de bajo nivel a los *deseos* de control de agentes superiores, *modulándolos* y tratando de ajustarse los más posible a ellos.

3. Se ha implementado un sistema que genera un mapa del entorno basado en las lecturas de los ultrasonidos, que permite:

   - Reducir la incertidumbre en las lecturas de los sensores de ultrasonido.

- Mediante la utilización del decremento temporal de la certeza de presencia de obstáculos se reduce la incertidumbre existente entre medidas falsas y medidas recientes. Asimismo se reducen los errores en la localización del robot dentro del mapa de acumulación debidos a los sensores odométricos.

4. Se ha definido e implementado la *Función de Accesibilidad*, que permite a los agentes motores de navegación conocer el estado del entorno y generar acciones de control basadas en esta información.

5. Se ha desarrollado un conjunto de agentes motores de navegación que permiten al robot evitar obstáculos, seguir paredes, e ir a localizaciones específicas del entorno.

6. Se ha desarrollado un conjunto de agentes motores de de visión, sobre una arquitectura reactiva, que permiten al robot buscar, detectar y seguir un conjunto de balizas de colores. La utilización de un sistema de estereovisión ha permitido que el robot conozca las posiciones en el espacio de sus objetivos.

7. Se ha establecido la comunicación con el robot, a través de la especificación de planes. Los planes se definen como un conjunto de pasos a seguir, estando cada uno de estos pasos especificados como una meta y una serie de "consejos" útiles para la consecución de la meta final. Asimismo, las metas y los consejos se plasman en agentes directores (perpetuos o de contingencia) que monitorizan la consecución del plan propuesto por el usuario.

8.2 Líneas Futuras

Ningún trabajo se puede considerar terminado: Siempre hay nuevas alternativas, mejoras y nuevas ideas sobre lo ya realizado. Este trabajo no es ninguna excepción, y abre un amplio abanico de nuevas posibilidades.

Una de mejoras, sobre la que ya estamos trabajando es el incremento de símbolos con los cuales el usuario puede especificar los planes. Estos nuevos símbolos no estarían biunívocamente relacionados con los agentes motores y sus atributos, sino que emergerían de una combinación de ellos. Por ejemplo, se podría especificar un símbolo "cruzar puerta" que estaría relacionado con los atributos del agente *Avoid* y el agente *Goto Point*.

También se pretende utilizar la noción del tiempo en la especificación de los planes. Por ejemplo, se le podría indicar al robot que se parase durante un cierto periodo de tiempo en una localización.

Asimismo pretendemos incorporar un mecanismo para que el robot decida por sí mismo si está acometiendo el plan de forma correcta. Por ejemplo, el robot podría inspeccionar el tiempo y el espacio recorrido en la consecución de un determinado paso. En el caso de que considerase que no está acometiendo las metas del plan de forma correcta, trataría de pedir consejo al usuario para continuar con el plan.

Anexo A

Comunicación entre Procesos

A.1 Comunicación entre Procesos

Para conseguir una mayor velocidad y eficiencia, los agentes motores de navegación y de visión se ejecutan en máquinas separadas, conectadas a través de una red "ethernet" (véase Figura A.1). A pesar de que estos agentes motores están diseñados para trabajar independientemente y en paralelo, en numerosas ocasiones es necesario que los procesos se comuniquen entre sí.

Los **agentes motores de navegación** se ejecutan en la máquina que está conectada a través de una conexión serie (bien por radio o directamente a través de un cable) con la plataforma móvil. La plataforma móvil acepta comandos de control a través de una interfaz serie por parte de un ordenador. Estos comandos de control pueden ser acciones de control que deben tomar los controladores, o bien la encuesta sobre el estado de los sensores (sensores de ultrasonido, o los encoders asociados a la plataforma y la torreta).

Los **agentes directores** se ejecutarán en la misma máquina en donde se ejecuten los agentes motores de navegación (véase Figura A.1), por lo que se deberán proveer una serie de mensajes para que estos agentes conozcan y modifiquen los atributos de los agentes motores de visión.

Por su parte, los **agentes motores de visión** se ejecutan en la máquina en la que se encuentra la tarjeta de adquisición de imágenes. Por tanto, si estos agentes desean mandar acciones de control a la torreta, o desean conocer la posición de esta, se tendrán que comunicar con la máquina a la que esté conectada la plataforma móvil.

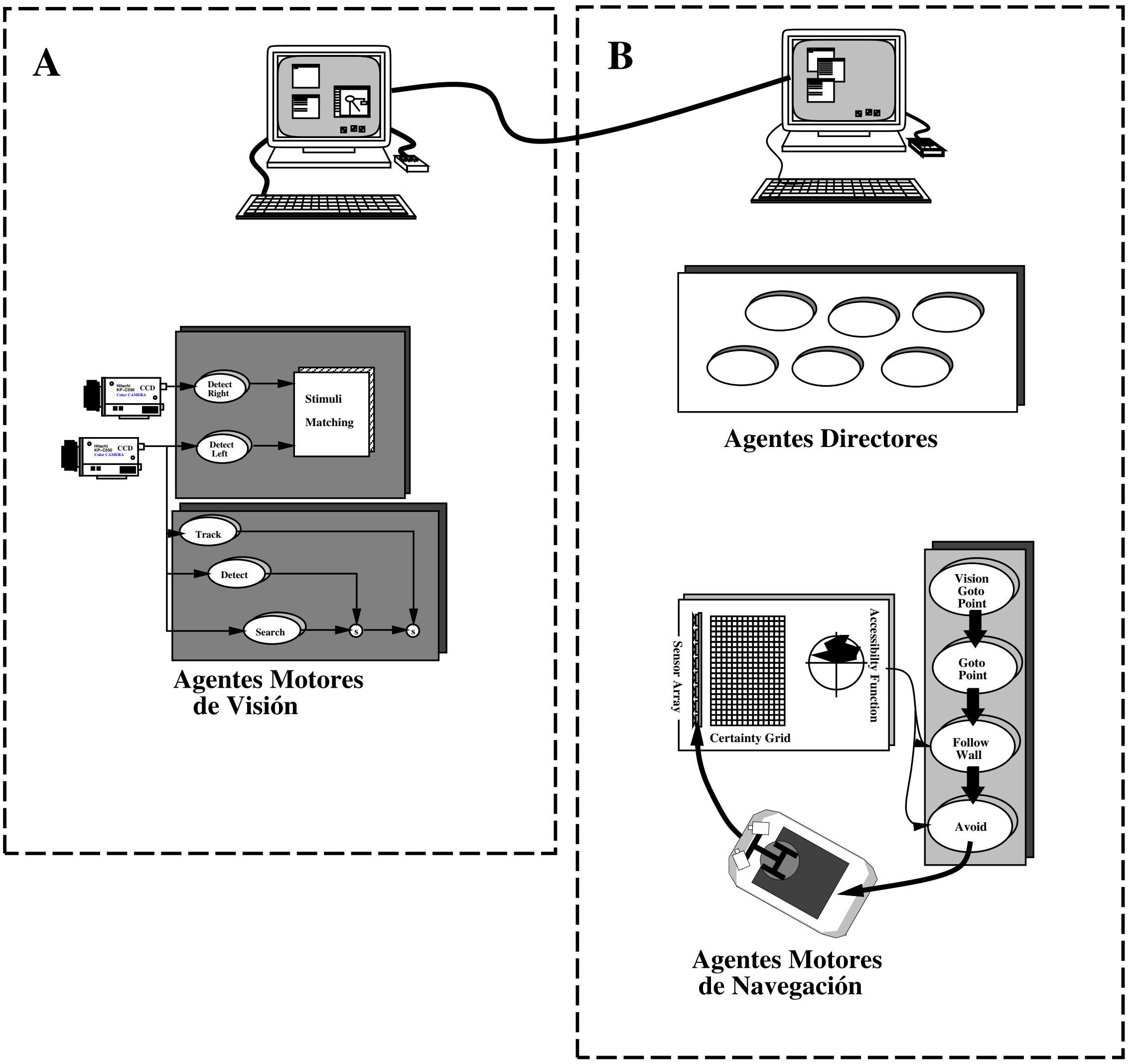


Figura A.1: Comunicación entre sub-sistemas.

El tipo de mensajes que deben intercambiarse los diferentes sub-sistemas son:

1. Estado del sistema: Los agentes motores de visión deben conocer la posición de la torreta en cada instante.

2. Comandos de control: Los agentes motores de visión deben poder actuar sobre los controles de la torreta para moverla.

3. Acciones de inhibición: Los agentes motores de visión pueden inhibir las salidas de los agentes motores de navegación.

4. Estado de los agentes: Los agentes directores deben conocer el estado de los diferentes agentes motores que componen el sistema, así como asignar diferentes valores a los atributos de los agentes motores.

A.2 Protocolo de Comunicación

Para realizar la comunicación entre procesos, se ha optado por la utilización de un interfaz de programación por medio de "sockets".

Un "socket" establece un nodo de comunicación por el cual un proceso puede emitir o recibir información. El dominio de un "socket" determina el formato de las direcciones que se le podrán dar y los diferentes protocolos soportados por las comunicaciones a través de los "sockets" de este dominio. En este sentido, el dominio empleado en la comunicación ha sido el dominio "Internet", al asegurar una mayor conectabilidad (al contrario que otros dominios, como "Unix", "Appletalk", etc.).

La comunicación entre procesos tiene dos modos principales: la comunicación en modo conectado y en modo no conectado:

- En modo no conectado, un proceso que desee enviar un mensaje con destino a otro, debe conocer la dirección de su interlocutor. El proceso remitente supone que el destinatario dispone de un "socket" conectado a esa dirección, ya que este tipo de interconexión no permite estar seguro de ello, y el mensaje debe incluir la dirección del "socket" de destino. Así, al no haber ninguna detección de que un socket está enlazado a la dirección dada, el mensaje se puede perder si el receptor no está activo o, simplemente, no ha abierto una conexión.

- La comunicación en modo conectado tiene que establecer una "conexión virtual" entre dos puntos. De esta forma, mientras que en el método sin conexión podemos decir que la comunicación es simétrica en el sentido de que el indicador del diálogo puede ser cualquiera de los dos que intervienen, el establecimiento previo de una conexión en el modo conectado determina la asimetría entre los dos procesos implicados: Uno de los dos procesos (el cliente) pregunta al otro (el servidor) si acepta esta comunicación. En el caso en que el servidor no esté activo, el cliente se quedará a la espera de que lo esté.

En la implementación del sistema, se ha optado por una comunicación en modo no conectado ("sockets" tipo datagrama) para asegurar una total independencia de los procesos: El modelo de comunicación no conectada no requiere que un proceso dependa de otro, o en otras palabras, no requiere de un modelo cliente-servidor. Por ejemplo, se pueden ejecutar los agentes motores de navegación sin la necesidad de que los agentes de visión estén activos. Así, se podrán comprobar y validar las acciones de estos agentes sin la participación de los agentes de visión. Los mensajes que se manden podrán ser interceptados por otro proceso, o simplemente se perderán.

A.3 Mensajes

Los mensajes que manden los procesos tienen que tener una estructura prefijada, de modo que el proceso que los lea los pueda comprender. En este sentido, todos los mensajes enviados tendrán una longitud de 1024 bytes. En principio, gran parte de esta cadena estará vacía, pero se reserva un espacio suficiente para futuras implementaciones y nuevos mensajes.

La cabecera del mensaje indicará el formato del resto del mensaje y el número de argumentos. Por ejemplo, si los agentes motores de visión envían un mensaje en cuya cabecera aparezca el comando `PANTILT`, los dos datos que le siguen serán dos valores enteros que indican la velocidad que deben tomar los motores de la torreta.

Si designamos como "A" al conjunto de agentes motores de visión que se ejecutan en la misma máquina, y como "B" al conjunto de agentes motores de navegación y agentes directores que se ejecutan en la misma máquina (véase Figura A.1), se puede hacer una clasificación basada en los mensajes que se envían desde cada una de estas dos máquinas. Los mensajes enviados por "A" deben ser recibidos e interpretados por "B" y viceversa.

A.3.1 Mensajes Enviados por "A"

- `PANTILT arg1 arg2` arg1 indica la velocidad del motor de rotación (pan) y arg2 la velocidad del motor de cabeceo (tilt).
- `STOP` Inhibe las acciones de los agentes motores de navegación, mientras mantiene parado al robot móvil.
- `GO` Desactiva la acción de `STOP`
- `TRACKING arg1` Si el valor de arg1 es 1, indica que el agente *Track* está activo, y el valor 0 indica que no está activo.
- `DISPARITY arg1` arg1 indica el valor de la disparidad calculada por el sistema de estereovisión.

A.3.2 Mensajes Enviados por "B"

- `PANTILT arg1 arg2` Los argumentos (arg1 y arg2) indican la posición de la torreta (pan, til).
- `DETECT arg1` El argumento (arg1) indica el tipo de baliza que se debe detectar: `BLUE_IN RED`, `RED_IN BLUE`, `BLUE_IN GREEN`, `RED_IN GREEN`, `GREEN_IN_BLUE`, `GREEN_IN_RED`.
- `SEARCH arg1` El argumento indica la zona de búsqueda del agente *Search*: `FORWARD`, `LEFT`, `RIGHT`, `EVERYWHERE`.

A.3.3 Envío y Recepción de Mensajes

Para garantizar el adecuado comportamiento del sistema, un mensaje enviado por un proceso debe ser leído e interpretado por el receptor en el menor tiempo posible.

La primera alternativa que se implementó consistió en la elaboración de unas subrutinas de aceptación de mensajes basadas en la intercepción de interrupciones del sistema. Es decir, se asoció una subrutina a la activación de la señal de entrada/salida del descriptor del socket de comunicación. En cuanto el sistema detectase que un mensaje había llegado, se activaba una interrupción de entrada/salida que hacía que el proceso de lectura de mensajes leyese el mensaje recibido.

Sin embargo, y debido a la multitud de procesos de entrada y salida más prioritarios (comunicaciones a través de canal serie, y adquisición de imágenes), las subrutinas asociadas a estas interrupciones no se activaban de forma inmediata, con lo que la lectura del mensaje se retardaba en exceso.

Por tanto, se optó por diseñar unas rutinas de lectura de mensajes que comprobaban cada $25ms$ si había algún mensaje que leer. En tal caso, se leía el mensaje y se continuaba con la ejecución de los diferentes procesos.

Bibliografía

[1] *Workshop on Visual Behaviors.* IEEE Computer Society Press, June 19 1993.

[2] Philip E. Agre and David Chapman. What are plans for? *Robotics and Autonomous Systems*, 6:17–34, 1990.

[3] John Aloimonos. Purposive and qualitative active vision. In *DARPA Image Understanding Workshop*, pages 816–828, 1990.

[4] Ronald C. Arkin. Integrating behavioral, perceptual and world knowledge in reactive navigation. In Pattie Maes, editor, *Designing Autonomous Agents*, March 1989.

[5] N. Ayache and Faugueras. Mantaining representations of the environment of a mobile robot. *IEEE Transactions on Robotics and Automation*, 5(6):804–819, 1989.

[6] Nicholas Ayache. *Artificial Vision for Mobile Robotics; Stereo Vision and Multisensory Perception.* MIT Press, 1991.

[7] H.H. Baker and T.O Binford. Depth from edge and intensity-based stereo. In *Proceedings of the 7th International Joint Conference*, pages 631–636, 1981.

[8] D. H. Ballard. Animate vision. *Artificial Intelligence*, 48(1):57–86, Feb 1991.

[9] Billur Barshan and Roman Kuc. Differentiating sonar reflections from corners and planes by employing an intelligent sensor. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):560–569, June 1990.

[10] Andrew Blake and Alan Yuille. *Active Vision.* MIT Press, 1992.

[11] J. Bohrenstein and Y. Khoren. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, June 1991.

[12] J. Bohrenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):1179–1187, September/October 1989.

[13] J. Bohrenstein and Y. Koren. Histogramic in-motion for mobile robot obstacle avoidance. *IEEE Transactions on Robotics and Automation*, 7(4):535–539, August 1991.

[14] Johann Bohrenstein and Yoram Koren. Obstacle avoidance with ultrasonic sensors. *IEEE J. Robotics and Automation*, RA-4(2):213–218, 1988.

[15] Johann Bohrenstein and Yoram Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):1179–1187, September/October 1989.

[16] Johann Bohrenstein and Yoram Koren. Histogramic in-motion mapping for mobile robot obstacle ovoidance. *IEEE Transactions on Robotics and Automation*, 7(4):535–539, August 1991.

[17] Johann Bohrenstein and Yoram Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, June 1991.

[18] Nikolaos G. Bourbakis. Real-time path planning of autonomous robots in a two-dimensional unknown dynamic navigation environment. *Journal of Inteligent and Robotic Systems*, 4:333–362, 1991.

[19] S.A. Brock-Gunn, G.R. Dowling, and T.J. Ellis. Tracking using colour information. Technical Report TCU/CS/1994/7, City University London, 1994.

[20] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automatition*, RA-2(1):14–23, March 1986.

[21] Rodney A. Brooks. Planning is just a way of avoiding figuring out what to do next. MIT Working Paper 303, September 1987.

[22] Rodney A. Brooks. A robot that walks; emergent behavior from a carefully evolved network. *Neural Computation*, 1(2):253–262, Summer 1989.

[23] Rodney A. Brooks. The behavior language; user's guide. A.I. Memo 1227, M.I.T., April 1990.

[24] Rodney A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.

[25] Rodney A. Brooks. Intelligence without reason. A. I. Memo 1293, M. I. T. Artificial Intelligence Laboratory, April 1991.

[26] Rodney A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[27] Michael K. Brown. Feature extraction techniques for recognizing solid objects with an ultrasonic range sensor. *IEEE Journal of Robotics and Automation*, RA-1(4):191–205, December 1985.

[28] David Chapman. Vision, instruction and action. A. I. Technical Report 1204, M. I. T. Artificial Intelligence Laboratory, April 1990.

[29] Jonathan H. Connell. A colony architecture for an artificial creature. M.I.T. A.I. Lab Tech Report 1151, M. I. T. Artificial Intelligence Laboratory, June 1989.

[30] Jonathan H. Connell. SSS: A hybrid architecture applied to robot navigation. Technical report, IBM Research Division, 1991.

[31] D. J. Coombs, T. J. Olson, and C. M. Brown. Gaze Control And Segmentation. In *AAAI-90 Workshop on Qualitative Vision*, pages 26–30, 29 Jul 1990.

[32] J. L. Crowley. Dynamic world modelling for an intelligent mobile robot. In *IEEE Seventh Int. Conf. Pattern Recognition*, pages 59–66, August 1984.

[33] C. Darwin. *On the Origin of Species by Means of Natural Selection.* London: Murray, 1859.

[34] D. Dennet. *Consciousness Explained.* Back Bay Books, 1991.

[35] Umesh R. Dhond and J. K. Aggarwal. Structure from stereo – a review. *IEEE Transactions on Systems, Man and Cybernetics*, 19(6):1489–1510, 1989.

[36] Michael Drumheller. Mobile robot localization using sonar. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(2):325–332, March 1987.

[37] Alberto Elfes. A distributed control architecture for an autonomous mobile robot. *Artificial Intelligence*, 1(2):135–144, 1986.

[38] Alberto Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, RA-3(3):249–265, June 1987.

[39] David Kortkamp et al. Integrated mobile-robot design. Winning the AAAI'92 competition. *IEEE Expert*, pages 61–73, August 1993.

[40] Oren Etzioni and Daniel Weld. A softbot-based interface to internet. *Communications of the ACM*, 37(4):72–76, July 1994.

[41] Olivier Faugeras. *Three-dimensional Computer Vision: A Geometric Viewpoint.* MIT Press, 1993.

[42] Richard E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[43] Anita M. Flynn, Rodney A. Brooks, William M. Wells, and David S. Barret. The world's largest one cubic inch robot. In *Proceedings IEEE Micro Electro Mechanical Systems*, pages 98–101. IEEE, February 1989.

[44] Michael R. Genesereth and Steven P. Ketchpel. Software agents. *Communications of the ACM*, 37(4):48–54, July 1994.

[45] G. Giralt, R. Chatila, and M. Vaisset. An integrated navigation and motion control system for autonomous multisensory mobile robots. In M. Brady and R. Paul, editors, *Firts International Symposium on Robotics Research*, 1983.

[46] W.E.L Grimson. Computational experiments with a feature-based stereo algorithm. *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-7:17–34, January 1985.

[47] N. Gupta and D. Nau. Optimal block's world solution ar NP-hard. Technical report, Computer Science Department, University of Maryland, 1990.

[48] Robert M. Haralick and Linda G. Shapiro. *Computer and Robot Vision*, volume 1. Addison Wesley, 1992.

[49] S. Harnard. Grounding symbolic capacity in robotic capacity. In L. Steels and R. Brooks, editors, *The "artificial life" route ro "artificial intelligence"*. New Haven: Lawrence Erlbaum, 1995.

[50] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Man, and Cybernetics*, SSC-4(2):100–107, July 1968.

[51] Glenn Healey. Segmenting images using normalized color. *IEEE Transactions On Systems, Man and Cybernetics*, 22(1):64–73, January/February 1992.

[52] Ian Horswill. Characterizing adaptation by constraint. In *Toward a practice of autonomous systems. Proceedings of the First European Conference on Artificial Life*, pages 58–63. MIT Press/Bradford Books, 1992.

[53] Ian Douglas Horswill and Rodney Allen Brooks. Situated vision in a dynamic world: Chasing objects. *AAAI-88*, pages 796–800, 1988.

[54] S. Stharama Iyengar and Alberto Elfes. *Autonomous Mobile Robots: Perception, Mapping, and Navigation*, volume 1. IEEE Computer Society Press, 1991.

[55] Leslie Pack Keabling and Stanley J. Rosenschein. Action and planning in embedded agents. *Robotics and Autonomous Systems*, 6:35–48, 1990.

[56] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *IEEE International Conference on Robotics and Automation*, pages 500–505, March 1985.

[57] H. Kimura and J.J.E. Slotine. Adaptive visual tracking and caussian network algorithms for robotic catching. In *Advances In Robust and Non-Linear Control Systems*, pages 67–74, November 1992.

[58] Gudrun J. Klinker, Steven A. Shafer, and Takeo Kanade. The measurement of highlights in color images. *International Journal of Computer Vision*, 2:7–32, 1988.

[59] Stephen M. Kosslyn and Olivier Koenig. *Wet Mind: The New Cognitive Neuroscience*. The Free Press, New York, 1992.

[60] David J. Kriegman, Ernst Triendl, and Thomas O. Binford. Stereo vision and navigation in buildings for mobile robots. *IEEE Transactions on Robotics and Automation*, 5(6):xxx–xxx, December 1989.

[61] John J. Leonard and Hugh F. Durrant-Whyte. Dynamic map building for an autonomous mobile robot. *The International Journal of Robotics Research*, 11(4):286–298, August 1992.

[62] Steven Levy. *Artificial Life; A Report From The Frontier Where Computers Meet Biology*. Pantheon Books, 1992.

[63] Tomás Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C-32(2):395–407, February 1983.

[64] Tomás Lozano-Pérez and Michael A. Wesley. An algorithm for planning collision-free paths among polyedral obstacles. *Comunications of the ACM*, 22(10):560–570, 1979.

[65] Pattie Maes. The dynamics of action selection. In *IJCAI-89*, pages 991–997, 1989.

[66] Pattie Maes. Behavior-based artificial intelligence. In *Prooceedings of the second Conference on Adaptative Behavior*. MIT Press, 1993.

[67] Pattie Maes. Modeling adaptive autonomous agents. In *Artificial Life 1*, pages 135–162. MIT Press, 1994.

[68] S. Tanveer F. Mahmood. Attentional selection in object recognition. AI-Technical Report-1420, MIT, 1993.

[69] Chris Malcolm and Tim Smithers. Symbol grounding via a hybrid architecture in an autonomous assembly system. *Robotics and Autonomous Systems*, 6:123–144, 1990.

[70] S.B. Maraphane and M.M. Trivedi. Region-based stereo analysis for robotic applications. *IEEE Transactions on Systems, Man and Cybernetics*, 19(6):1447–1464, November/December 1989.

[71] D. Marr and T. Poggio. A computational theory of human stereo vision. *Proc. Royal Soc. London*, B 204:301–328, 1979.

[72] D. C. Marr. *Vision*. Freeman San Francisco CA, 1982.

[73] Maja Mataric. Behavior-based architectures for intelligent control. *Workshop on Intelligent Autonomous Control Systems*, November 1992.

[74] Maja Mataric. A distributed model for mobile robot environment-learning and navigation. Technical Report AI-TR-1228, MIT Artificial Intelligence Laboratory, 1992.

[75] Maja Mataric. Interaction and intelligent behavior. Technical Report AI-TR-1495, MIT Artificial Intelligence Laboratory, 1994.

[76] G. Medioni and R. Nevatia. Segment-based stereo matching. *Comput. Vision, Graphics Image Processing*, 31:2–18, July 1985.

[77] H. P. Moraveck. Towards automatic visual obstacle avoidance. *Proceedings of the 5th International Joint Conference on Artificial Intelligence (IJCAI)*, page 584, August 1977.

[78] H. P. Moraveck. Sensor fusion in certainty grids for mobile robots. *AI Mag.*, pages 61–74, Summer 1988.

[79] Hans Moraveck. *Mind Children. The Future of Robot, and Human Intelligence.* Harvard University Press, 1988.

[80] Fabrice R. Noreils and Raja G. Chatila. Plan execution monitoring and control architecture for mobile robots. *IEEE Journal of Robotics and Automatition*, 11(2):255–266, April 1995.

[81] Y. Ohta and T. Kanade. Stereo by Intra- and Inter-scanline Search Using Dynamic Programming. *IEEE Trans. Pattern Anal. Machine Intell.*, 7(2):139–154, March 1985.

[82] David W. Payton. Internalized plans: A representation for action resources. *Robotics and Autonomous Systems*, 6:89–103, 1990.

[83] J. Ross, Joey Griffith, Ralf R. Kohler, Allen R. Hanson, and Edward M. Riseman. Segmenting images using localized histograms and region merging. *International Journal of Computer Vision*, 2:311–347, 1989.

[84] Miguel Schneider-Fontán and Lourdes Agapito. An image processing environment based on object oriented programming. In *Proceedings of the Second International Congress on Signal Processing Applications and Tecnology*, pages 497–502. DSP Associates, 1993.

[85] Miguel Schneider-Fontán and Maja J Mataric. A study of territoriality: The role of critical mass in adaptive task division. In *From Animals to Animats 4, Fourth International Conference on Simulation of Adaptive Behavior.* MIT Press, 1996.

[86] Luc Steels. The artificial roots of artificial intelligence. *Artificial Life 1*, pages 75–100, 1994.

[87] I. Thomas, E. Simonelli, and R. Bajcsy. Spherical retinal flow for a fixating observer. In *Workshop On Visual Behaviors: Computer Vision and Pattern Recognition*, pages 37–44, June 19 1994.

[88] Alan M. Thomson. The navigation system of the JPL robot. In *Proceedings of the International Joint Conference on Artificial Inteligence*, pages 749–757, 1977.

[89] A. Treisman. The perception of features and objects. In A. Baddeley and L. Weiskrants, editors, *Attention: Selection, Awareness and Control*, pages 5–35. Clarendon Press, 1993.

[90] J. K. Tsotsos. Analyzing vision at the complexity level. *Behavioral and Brain Sciences*, 13:423–469, 1990.

[91] W. Y. K. Wai and J. K. Tsotsos. Directing attention to onset and offset of image events for eye-head movement control. In *Workshop On Visual Behaviors: Computer Vision and Pattern Recognition*, pages 79–84, 1994.

[92] Barbara Webb. *Perception in Real and Artificial Insects: A Robotic Investigation in Cricket Photonaxis.* PhD thesis, University of Edimburgh, 1993.

[93] Anne Wright. A high speed low lattency portable vision sensing system. In David P. Casasent, editor, *Intelligent Robots and Computer Vision XII: Algorithms and Techniques*, volume 2055, pages 263–270, August 1993.