

Índice

1 Introducción.....	1
1.1 Motivación.....	2
1.2 Antecedentes	3
1.3 Objetivos y condicionantes a la implementación	4
1.4 Organización de la tesis	6
2 Sistemas de navegación.....	9
2.1 Introducción.....	10
2.2 Navegación	11
2.3 Tareas incluidas en la navegación.....	12
2.4 Arquitecturas de navegación	16
2.5 Clasificación de las arquitecturas de navegación.....	18
2.5.1 Reactivas frente a deliberativas	19
2.5.2 Arquitecturas basadas en comportamientos	24
2.5.3 Arquitecturas jerárquicas.....	26
2.5.4 Arquitecturas jerárquicas basadas en comportamientos.....	30
2.5.5 Arquitecturas basadas en sistemas de pizarra (<i>blackboard</i>)	34
2.6 Control de alto nivel e interconexión entre módulos.....	37
3 Detección, diagnóstico y recuperación de fallos	39
3.1 Introducción.....	39
3.2 Fases en el tratamiento de errores.....	42
3.3 Generación de síntomas	45
3.3.1 Generación analítica de síntomas.....	45
3.3.1.1 Modelado del sistema.....	46
3.3.1.2 Extracción de datos	47
3.3.1.3 Obtención de síntomas	52
3.3.2 Generación heurística de síntomas.....	53
3.4 Diagnóstico.....	53
3.4.1 Métodos de clasificación.....	54
3.4.1.1 Clasificadores geométricos	55
3.4.1.2 Clasificadores estadísticos	56
3.4.1.3 Redes neuronales	60
3.4.2 Sistemas basados en conocimiento.....	62
3.4.2.1 Sistemas expertos	63
3.4.2.2 Sistemas basados en conocimiento con razonamiento aproximado.....	64
3.4.3 Sistemas mixtos (redes neuronales - lógica borrosa)	68

3.4.3.1 Arquitectura de redes neuronales en la que se integra la lógica borrosa.....	69
3.4.3.2 Arquitectura de lógica borrosa en la que se integran redes neuronales.....	70
4 Detección, diagnosis y recuperación de fallos en robótica móvil.....	73
4.1 Introducción.....	73
4.2 Niveles de tratamiento de fallos.....	76
4.3 Tratamiento de fallos en el sistema sensorial.....	78
4.3.1 Modelos y tipos de fallos a detectar.....	78
4.3.2 Extracción de síntomas.....	79
4.3.3 Detección y diagnosis.....	82
4.3.4 Recuperación.....	82
4.3.5 Ventajas e inconvenientes.....	83
4.4 Control de errores al nivel de navegación.....	84
4.4.1 Modelos y tipos de fallos a detectar.....	84
4.4.2 Extracción de síntomas.....	87
4.4.3 Detección y diagnosis.....	88
4.4.4 Recuperación.....	89
5 Arquitecturas y sistema de navegación.....	91
5.1 Xavier.....	92
5.1.1 Base.....	93
5.1.2 Sistema sensorial.....	95
5.1.2.1 Sonar.....	95
5.1.2.2 Láser.....	96
5.1.2.3 Sistema de visión.....	101
5.1.3 Ordenadores de a bordo.....	101
5.1.4 Sistema de voz.....	101
5.2 Amelia y Rato.....	102
5.3 Arquitectura de navegación.....	103
5.3.1 Flujo de información.....	105
5.3.2 Descripción de las capas.....	107
5.4 Procesos y comunicación.....	111
5.4.1 Arquitectura de control de tareas.....	112
6 Modelo determinista.....	115
6.1 Introducción.....	115
6.2 Situaciones de excepción.....	116
6.2.1 Espacio de estados.....	118
6.3 Extracción de síntomas (monitores).....	122
6.3.1 Monitorización del tiempo.....	123

6.3.2 Seguimiento de la posición.....	124
6.3.3 Seguimiento de vueltas	130
6.3.4 Detección negativa de intersecciones	130
6.3.5 Carga de la batería	131
6.4 Diagnóstico y recuperación.....	133
6.4.1 Nivel de batería bajo	134
6.4.2 Demasiadas vueltas	135
6.4.3 Error de posición	136
6.4.4 Otras excepciones	138
 7 Modelo estocástico	 141
7.1 Introducción.....	141
7.2 Características del sistema.....	143
7.2.1 Información de estado.....	143
7.2.2 Acciones.....	144
7.2.3 Objetivos	145
7.3 Modelos de diagnóstico aplicables.	146
7.3.1 Redes de Bayes en supervisión, diagnóstico y recuperación.....	146
7.3.2 Procesos de Markov	146
7.3.2.1 COMDP	147
7.3.2.2 POMDP.....	148
7.3.2.3 Aplicación en supervisión, diagnóstico y recuperación.....	149
7.4 Descripción del modelo	151
7.4.1 Estados.....	151
7.4.1.1 Descripción de los estados (componentes).....	155
7.4.1.2 Independencia de los componentes (reducción de estados).....	157
7.4.2 Acciones.....	158
7.4.3 Observaciones	164
7.4.3.1 Monitores	164
7.4.3.2 Observaciones provenientes de las acciones	166
7.4.4 Probabilidades de transición	166
7.4.5 Probabilidades de observación.....	168
7.4.6 Bonificación	171
7.5 El supervisor.....	172
7.5.1 Actualización de la información de estado	175
7.5.2 Planificación y toma de decisiones	177
7.5.2.1 Política local	179
7.5.2.2 Política global	181
7.5.2.3 Estrategias de decisión heurísticas.....	184
7.5.2.4 Estrategias de decisión heurísticas con predicción	190

8 Validación del sistema de supervisión	197
8.1 Técnicas de validación	197
8.2 Descripción de los tests de evaluación	199
8.3 Evaluación comparativa.....	201
8.3.1 Comparativa sobre el modelo	201
8.3.1.1 Realización de los experimentos	202
8.3.1.2 Resultados	204
8.3.2 Análisis sobre un modelo más sencillo.....	208
8.3.3 Comparaciones sobre problemas de referencia	215
8.3.3.1 Problemas simples con solución óptima disponible.....	216
8.3.3.2 Problemas de mayor dimensión con solución óptima no disponible	221
8.4 Pruebas en el robot (tests de hipótesis)	226
9 Conclusiones.....	233
9.1 Contribuciones.....	233
9.2 Modelo determinista.....	234
9.3 Modelo estocástico.....	235
9.4 Sistemas de decisión.....	237
9.5 Líneas futuras	238
A Procesos de decisión de Markov	241
A.1 COMDP	241
A.1.1 Clasificación.....	242
A.1.2 Políticas de decisión. Planes de actuación	243
A.1.3 Funciones Valor.....	244
A.1.4 Cálculo de la política óptima	246
A.1.4.1 Iteración de la función valor.....	246
A.1.4.2 Iteración de la política	248
A.2 POMDP.....	249
A.2.1 Modelo	249
A.2.2 Transformación de POMDP en COMDP.....	251
A.2.3 Función valor.....	253
A.2.4 Cálculo de la política óptima	255
A.2.4.1 Iteración de la política	255
A.2.4.2 Iteración de la función valor.....	255

B Ejemplo sencillo de supervisión usando POMDP	259
<i>B.1 Definición de la situación</i>	<i>259</i>
<i>B.2 Plan local.....</i>	<i>260</i>
B.2.1 Incertidumbre en las observaciones.....	265
<i>B.3 Plan global.....</i>	<i>267</i>
C Definición del modelo.....	277
<i>C.1 Sintaxis del archivo de definición del modelo.</i>	<i>277</i>
C.1.1 Conceptos generales.....	278
C.1.2 Probabilidades de transición	279
C.1.3 Probabilidades de observación.....	280
C.1.4 Bonificación	281
<i>C.2 Listado del archivo.....</i>	<i>283</i>
D Solución al COMDP asociado al modelo.....	291
Referencias Bibliográficas	295

1 Introducción

Desde los años 70 se han venido utilizando industrialmente vehículos autónomos para el transporte de distintos elementos en la cadena productiva, especialmente en la industria del automóvil. La gran mayoría de los vehículos en servicio en la actualidad son filoguiados. Éstos circulan siguiendo caminos fijos predispuestos en la planta. El recorrido de este tipo de vehículos está restringido a trayectorias preestablecidas, lo que hace que sean poco flexibles tanto para la elección de caminos alternativos como para la evitación de posibles obstáculos.

Como contrapartida a los vehículos aloguidados, se está investigando la posibilidad de realizar vehículos con más autonomía, capaces de desplazarse desde un punto (origen) a otro (destino) evitando los obstáculos que encuentre a su paso. Este proceso es el que se conoce con el nombre de navegación. A lo largo de esta tesis se utilizarán indistintamente los términos robot móvil, agente móvil y vehículo móvil para referirse a vehículos con esta capacidad.

Los robots móviles que operan en entornos dinámicos necesitan un sistema de navegación muy fiable para actuar de forma autónoma durante largos períodos de tiempo. El sistema de navegación dispone de información ruidosa e imprecisa obtenida de los sensores en un entorno dinámico y parcialmente conocido. Además, el robot tiene que responder a eventos externos en tiempo real independientemente de la situación en la que se encuentre. Debido a que es muy difícil para el programador tener en cuenta todas las situaciones posibles, se hace necesario un mecanismo de supervisión. Mientras que existen muchas técnicas para gestionar diversas partes del problema de navegación (evitación de obstáculos, planificación de trayectorias, etc.), se han dedicado menos esfuerzos, comparativamente, al problema de supervisión y tratamiento de situaciones de excepción.

En esta tesis se presenta una arquitectura para tratar las situaciones de excepción. Para ello se usa un conjunto de monitores que obtienen información acerca de la situación en la que se encuentra el robot. Por monitor aquí se entiende una tarea continua en tiempo real que determina las condiciones del sistema, recopilando

información del mismo, reconociendo y notificando un determinado tipo de anomalías para las que fue diseñado.

Se han realizado dos soluciones basadas en la información que éstos proveen. En la primera (determinista) se organizan los monitores de forma jerárquica. En la parte más alta de esta arquitectura se sitúan los monitores más generales que detectan un gran número de excepciones, mientras que en las capas inferiores se encuentran los monitores más específicos que detectan situaciones concretas. El disparo, o lo que es lo mismo, la verificación de condiciones de un monitor de bajo nivel provee más información que el disparo de los de alto nivel, pues el número de posibilidades en cuanto al estado del robot es menor. Además, los monitores específicos son mucho más rápidos al verificar parámetros bien determinados. El esquema desarrollado permite la fácil adición de nuevos monitores a medida que se van identificando nuevas excepciones.

En la segunda solución se usa un modelo estocástico que tiene en cuenta la incertidumbre acerca del estado del modelo proveniente del ruido asociado a las observaciones y la fluctuación en el resultado de las acciones. Para ello, utilizando como observadores los monitores ya desarrollados, se ha construido un modelo basado en procesos de Markov parcialmente observables.

1.1 Motivación

Esta tesis se enmarca dentro de la línea de investigación sobre robots móviles del Departamento de Ingeniería de Sistemas y Automática de la Universidad de Vigo, si bien gran parte del desarrollo se ha llevado a cabo en colaboración con el laboratorio *Robot Learning Laboratory* de la Universidad de Carnegie Mellon, Pittsburgh (USA). En este laboratorio se ha desarrollado un robot móvil para interiores apodado *Xavier*. Entre otras, *Xavier* realiza tareas de navegación entre dos posiciones de la 5ª planta del edificio donde se ubica el laboratorio arriba mencionado, de captura y envío de imágenes a través de correo electrónico y de articulación de mensajes. Estas misiones son realizadas por el robot de forma autónoma, sin intervención humana, con un alto porcentaje de éxito. Además, las tareas pueden ser activadas a través de Internet [Simmons 1999]. Sin embargo, en algunas ocasiones se hace inevitable la intervención de un operador.

Una de las expresiones de inteligencia en un robot móvil es que éste pueda operar de forma autónoma ante cualquier situación, incluso responder ante situaciones

imprevistas con anticipación. El objetivo anhelado en esta tesis es, precisamente, aumentar la autonomía del robot mediante la elaboración de un módulo de supervisión y tratamiento de situaciones de excepción.

Xavier lleva funcionando desde diciembre de 1995 [Simmons 1997b]. En julio de 1996 había completado unas 1800 peticiones y viajado más de 75 km. La tasa de consecución con éxito de tareas es de un 95%, lo cual supone todo un avance en este campo. Con objeto de mejorar este porcentaje y reducir la presencia del operador, se ha planteado el desarrollo de un módulo de tratamiento de situaciones de excepción. En la elaboración de este supervisor se ha aprovechado la experiencia de personas que han trabajado con el robot para catalogar una serie de situaciones problemáticas. Este método también está siendo aplicado al robot adquirido por el Departamento de Ingeniería de Sistemas y Automática de la Universidad de Vigo (apodado *Rato*) utilizando la arquitectura de navegación de *Xavier*.

1.2 Antecedentes

Aunque al principio de este capítulo se han mencionado las aplicaciones industriales de los robots móviles, no es en este campo donde más se han empleado por el momento. Aparte de su elevado coste, la falta de robustez de los sistemas de navegación existentes en la actualidad está frenando su implantación industrial.

Uno de los principales motivos del desarrollo del campo de la robótica móvil es su potencial para reducir la necesidad de presencia humana en entornos peligrosos. Aplicaciones como limpieza de residuos tóxicos, trabajo en plantas nucleares, exploración planetaria, misiones de rescate, vigilancia y reconocimiento tienen elementos de peligro para los operadores humanos. En todas esas aplicaciones es deseable reducir el riesgo para las personas e introducir robots autónomos aunque el coste de éstos sea elevado. En cualquier caso, sigue siendo necesario el aumento de la robustez del sistema, dado que la intervención externa no es posible o está muy limitada.

El objetivo del incremento de la autonomía de los robots móviles se ha enfocado en algunos casos como una tarea de aumentar la tolerancia a fallos. En la mayor parte de los casos, se realiza un tratamiento de fallos del sistema sensorial, ya sea a nivel hardware [Stergios 1998b] o incluyendo también fallos software [Murphy 1996] [Ferrell 1994]. En algunos sistemas [Noreils 1995] [Sankaran 1977] [Stuck 1992] [Stuck 1995] también se tienen en cuenta los fallos en el sistema de navegación.

El tratamiento de fallos, en este último grupo, está siempre condicionado por la arquitectura de navegación usada, dando lugar a diferentes sistemas de supervisión, detección y recuperación ante fallos.

El término fallo, usado con distintos matices a lo largo de la bibliografía, es considerado, en esta tesis, como cualquier desviación no permitida de al menos una característica, propiedad o parámetro del sistema con respecto a su valor aceptable, nominal o estándar.

El enfoque aquí presentado es más amplio, al tratar con *excepciones* en lugar de fallos. Una *excepción* en el contexto de esta tesis es cualquier situación que impida la consecución con éxito de una determinada tarea. La diferencia radica en que pueden existir situaciones en las que, aún funcionando todo el sistema de acuerdo con las especificaciones, el robot no pueda conseguir su objetivo porque se encuentra en unas condiciones que no habrían sido contempladas en la fase de diseño. Por ejemplo, se sabe que algunos obstáculos no pueden ser detectados por cierto tipo de sensores y así aparece indicado en las especificaciones del fabricante. Si esto no se tiene en cuenta, se puede llegar a una situación donde el robot no pueda continuar por no detectar un obstáculo que tiene delante, aún cuando su sistema sensorial esté funcionando de acuerdo a las especificaciones.

Otra diferencia fundamental entre el modelo de supervisión y tratamiento de situaciones de excepción aquí presentado, con respecto a los descritos por otros autores se refiere a la forma de tratar la incertidumbre en las observaciones, la probabilidad de fallo de las acciones y las consideraciones acerca del coste de éstas para distintas situaciones. Asimismo, con la excepción de aplicaciones puntuales [Kleer 1987], por lo general, no se tienen en cuenta múltiples fallos simultáneos, sino que se considera que sólo se puede producir un fallo en cada instante. Esta restricción no existe en el sistema que aquí se presenta.

1.3 Objetivos y condicionantes a la implementación

Debido a que es muy difícil para el programador tener en cuenta todas las situaciones con las que el robot se puede encontrar, el planteamiento que aquí se propone consiste en desarrollar métodos de detección según se van descubriendo las situaciones de excepción, en lugar de incluir el tratamiento de éstas dentro del sistema de navegación.

Dada la importancia del tema de la fiabilidad y seguridad en sistemas dinámicos, éste ha sido tratado en multitud de congresos y artículos. El tratamiento de fallos se suele dividir en una fase de generación de residuos y otra de diagnóstico. Los residuos son parámetros basados en una desviación entre medidas del sistema y cálculos obtenidos del modelo del mismo. A partir de estos parámetros, en la fase de diagnóstico se determina el tipo y características del fallo.

La mayoría de los esfuerzos en este campo se ha enfocado al problema de la generación de residuos, argumentando que la parte de decisión basada en residuos que representan fielmente el fenómeno que se pretende detectar es relativamente fácil [Chen 1998]. En el caso de la robótica móvil, uno de los elementos diferenciados es el del tratamiento de la incertidumbre en distintos niveles: modelado, actuación y sensorial. El tratamiento de fallos con modelado perfecto es, a todas luces, mucho más sencillo que el tratamiento de los mismos cuando hay que tener en cuenta factores derivados de la incertidumbre en distintas partes del proceso. Esto, unido a la complejidad del sistema (es muy difícil modelar todas las situaciones con las que el robot se puede encontrar), hace prácticamente imposible la generación de residuos completamente fiables.

El enfoque aquí presentado no se limita a la generación de residuos o síntomas (indicadores de fallo) para identificar las posibles situaciones de excepción. Partiendo de la base que los síntomas no son completamente fiables, el siguiente paso es optimizar las decisiones del sistema para que la probabilidad de éxito de las distintas tareas encomendadas al robot sea lo más alta posible. Para ello, hay que tener en cuenta el coste de las falsas detecciones, así como de la probabilidad de las mismas. Estas consideraciones sugieren el planteamiento de un sistema que comparte bastantes puntos en común con los procesos de Markov parcialmente observables:

- ✓ El proceso puede ser descrito usando un conjunto finito de estados.
- ✓ El sistema de supervisión puede actuar sobre el proceso a través de un conjunto de acciones.
- ✓ La dinámica del sistema se describe mediante un esquema de probabilidades de transición entre estados.
- ✓ El supervisor posee información acerca del estado del proceso a través de un conjunto finito de observadores.
- ✓ La calidad de la supervisión viene dada por medio de un sistema de bonificación o costes asociados a estados o transiciones.
- ✓ El objetivo del supervisor es maximizar una función que combina los costes y bonificaciones obtenidas a lo largo del tiempo.

Este planteamiento constituye una estructura muy completa para modelar procesos dinámicos parcialmente observables, así como sistemas de decisión basados en dicha estructura. Sin embargo, actualmente no es posible encontrar soluciones exactas del cálculo de la política óptima para sistemas complejos como el que aquí se trata.

Otro problema de estos procesos está causado por la enorme cantidad de parámetros a determinar para definir el modelo del sistema. La mayor parte de estos parámetros viene dado en forma de probabilidades y se hace necesaria la reducción o automatización en la obtención de dichos parámetros.

1.4 Organización de la tesis

La descripción de las técnicas involucradas en el desarrollo de esta tesis, contribuciones y resultados se ha organizado en ocho capítulos y cuatro apéndices.

En el primer capítulo, en el cual nos encontramos, se pretende dar una visión general sobre el contenido de la tesis.

Los tres siguientes capítulos (segundo, tercero y cuarto) están dedicados a la descripción del estado de la técnica sobre los temas de arquitecturas de navegación en robots móviles y sistemas de supervisión, detección e identificación de fallos. Éstos se tratan primero desde un punto de vista global y más tarde, en particular, en el campo de los agentes móviles.

El segundo capítulo describe, de forma resumida, los conceptos fundamentales que definen la realización de tareas de un robot móvil. Se presentan algunas soluciones adoptadas por diversos autores en el diseño del sistema de navegación, sin olvidar que el objetivo principal de esta tesis es la parte de supervisión y tratamiento de fallos. Dicha descripción se hace necesaria debido a la dependencia del sistema de supervisión con la arquitectura de navegación. Desde el punto de vista de supervisión, detección y recuperación de errores, lo ideal sería que estas funciones se implementaran de forma independiente al sistema de navegación y la plataforma usada. En la práctica, las soluciones utilizadas suelen estar extremadamente ligadas a la arquitectura del sistema de navegación [Sankaran 1977] [Ferrell 1993].

La detección, identificación y diagnosis para sistemas en general se estudian en el capítulo tres. Algunas de las técnicas descritas son directamente aplicables al caso de

la robótica móvil, si bien existen elementos característicos que centran nuestro interés en determinados problemas como son los de diagnóstico.

En el capítulo cuarto se aborda la aplicación de los conceptos del capítulo anterior a la robótica móvil. Se describen las aproximaciones presentadas por distintos autores, tanto a nivel detección de fallos hardware como software.

En el capítulo quinto se presentan las plataformas sobre las que se llevó a cabo este trabajo. En primer lugar los componentes funcionales (sistema sensorial y motriz) de los distintos robots; a continuación, la arquitectura de navegación y los distintos procesos que forman parte del sistema.

Los capítulos siguientes constituyen el núcleo de las aportaciones de la tesis donde se presentan dos aproximaciones a la solución del problema. La primera de tipo general y sencilla, pensada sobre todo para los sistemas que poseen información precisa sobre el estado del mismo; mientras que la segunda es más compleja. Dicha complejidad es rentable para sistemas que poseen incertidumbre acerca de la información de estado, como es el caso de los robots móviles.

En el capítulo sexto se describe la primera aproximación para tratar la supervisión de situaciones de excepción o errores utilizando un sistema jerárquico de monitores.

Debido a la dificultad para obtener síntomas que permitan discernir de forma fiable entre las situaciones en las que puede encontrarse el agente móvil, se ha elaborado una aproximación estocástica presentada en el capítulo séptimo.

En el capítulo octavo se describen los resultados obtenidos en las distintas evaluaciones del sistema de decisión. En primer lugar, se comparan los resultados obtenidos mediante simulación para las distintas variaciones del sistema de decisión descritas en el capítulo anterior sobre el modelo de supervisión cuyos parámetros se definen en el apéndice C. Como no es posible obtener la solución exacta al POMDP para dicho modelo, también se presentan los resultados obtenidos para un ejemplo más sencillo (apéndice B). De igual forma, se muestran los resultados obtenidos sobre distintos problemas que vienen siendo utilizados como referencia en la bibliografía sobre procesos de Markov para ver en qué ocasiones es conveniente usar variantes *anytime* de los algoritmos heurísticos propuestos en la tesis. Asimismo, se presentan algunas de las respuestas del robot ante distintos escenarios donde existen situaciones de excepción. Por último, las conclusiones se incluyen en el capítulo noveno.

Los apéndices incluyen un breve resumen de los procesos de Markov, un simple ejemplo del funcionamiento del sistema de decisión desde dos puntos de vista diferentes (local y global), una copia del archivo donde se define el modelo utilizado y la solución COMDP del modelo.

2 Sistemas de navegación

Desde muy antiguo el ser humano ha perseguido el sueño de construir un robot autónomo capaz de realizar tareas inteligentes similares a las de los humanos para hacer la vida de estos más sencilla y placentera. Este tema suscitó un sinnúmero de obras de ciencia-ficción y una de las primeras referencias que se conocen al término robot se debe al novelista Karel Capek quien, en 1921 usó este término “Rossum’s Universal Robots”.

Desde entonces, este tema ha servido de inspiración para una cantidad considerable de obras, algunas de las cuales se están convirtiendo en grandes superproducciones de cine. Como siempre, la imaginación humana no tiene límites y la realidad que hoy vivimos dista mucho de la ficción de las novelas. Lo cual no quiere decir que la producción de mecanismos parecidos no sea posible en el futuro. Prueba de ello son algunos de los ingenios que aparecen en algunas de las novelas que antaño escribía el novelista Julio Verne, como el submarino que describe en la novela “Veinte mil leguas de viaje submarino”, han dejado de ser ficción y forman ahora parte de nuestra vida. No obstante, la ficción siempre va bastante por delante de la realidad y los robots actuales, que es lo que aquí nos interesa, se parecen muy poco (al menos funcionalmente) a los de la ficción.

Movidos por el afán de construir un agente autónomo, los primeros proyectos en robots móviles datan de los años 70 si bien las técnicas disponibles en aquellos momentos no permitían alcanzar las perspectivas creadas inicialmente. Fue a comienzos de los 80 cuando la investigación en este campo volvió a tomar especial auge.

Hoy, sin haber alcanzado el objetivo de construir robots móviles personales, se han logrado grandes avances en este campo. En la mayor parte se trata de prototipos elaborados por universidades que a veces calan en la industria para su posterior comercialización, la mayor parte de las veces destinados a la investigación. Algunas compañías como Honda se están ya aventurando en este campo y destinando grandes sumas a la investigación.

2.1 Introducción

Dejando a parte el contexto novelístico, en este capítulo se describen sucintamente algunas soluciones aplicadas al sistema de navegación (proceso que permite a un agente móvil moverse de forma autónoma por un determinado entorno). Nuestro interés se centra en aquellos puntos que puedan influir en los módulos de supervisión y tratamiento de fallos.

Desde el punto de vista de supervisión, detección y recuperación de errores, lo ideal sería que estas funciones se implementaran de forma independiente al sistema de navegación y la plataforma usada. En la práctica, las soluciones utilizadas están extremadamente ligadas a la arquitectura del sistema de navegación [Sankaran 1977] [Ferrell 1993]. Así, en sistemas reactivos, rara vez se usa recuperación de errores en navegación dado que estos sistemas básicamente reaccionan a eventos.

En el contexto de navegación en robots móviles, existen múltiples arquitecturas que tratan con excepciones (errores) de diferentes formas [Noreils 1990] [Stuck 1992] y a distintos niveles. Algunos autores tratan los errores en el nivel más bajo posible de la estructura jerárquica [Ferrell 1993]. Estos desarrollos se realizan sobre todo cuando se manejan fallos en el sistema sensorial.

Los procesos de monitorización y recuperación ante errores son responsables de que el robot ejecute correctamente las tareas encomendadas, detectando cuando esto no es así y ejecutando las acciones de recuperación oportunas.

En algunas arquitecturas de tipo deliberativo [Sankaran 1977] [Stuck 1992] la monitorización se limita a verificar la correcta finalización de cada uno de los pasos trazados en la planificación. Como se detallará más adelante (capítulo 5), la arquitectura sobre la cual se han llevado a cabo las investigaciones de esta tesis, combina comportamientos reactivos y planificados. Incluso en sistemas reactivos es posible extraer observaciones acerca del comportamiento del robot y determinar si se ajusta al esperado.

Aunque algunas etapas en el tratamiento de errores puede realizarse de forma independiente del sistema de navegación, otras, como la extracción de síntomas, tienen que estar vinculadas al navegador de donde obtienen la información.

A continuación se describe el estado de la técnica en sistemas de navegación dejando para próximos capítulos la parte de detección de errores.

2.2 Navegación

Navegación es el proceso mediante el cual un agente móvil sigue un camino a través de un determinado entorno con el objeto de alcanzar la posición destino. Para que esto sea posible, el agente debe poseer capacidad de movimiento, percepción de su entorno, memoria, razonamiento y reacción ante eventos no previstos. Las aplicaciones donde se están usando robots móviles van aumentando a medida que se optimizan los sistemas que los componen. Sin pretensión de ser exhaustivos, se pueden mencionar entre otras:

- ✓ Realización de tareas agrícolas en invernaderos [Gómez 1996]
- ✓ Ayuda a minusválidos y personas mayores [Matos 1999]
- ✓ Exploración de entornos hostiles como desiertos [Wettergreen 1999], volcanes [Boehmke 1995], etc.
- ✓ Realización de tareas de forma remota a través de Internet [Simmons 1999].
- ✓ Guía en un museo [Burgard 1998].
- ✓ Exploración del entorno y construcción de mapas [Thrun 1998]. En algunos casos interesa explorar un entorno desconocido ya sea para la creación de mapas, añadir información o ajustar parámetros sobre mapas ya existentes.
- ✓ Exploración planetaria [Chatila 1996] [Krotov 1992]. De todos son conocidos los esfuerzos de organismos como la NASA para elaborar robots que se puedan utilizar en tareas de exploración de otros planetas.

La consecución de objetivos se puede medir de distinta forma según las tareas encomendadas en cada caso. Mientras que en la exploración del entorno el éxito se evaluará de acuerdo a la cantidad de información acumulada así como la precisión y veracidad de la misma, en la entrega de correo podría cuantificarse por la cantidad de correo entregado y el tiempo de retardo en la entrega. Si bien todos estos objetivos son distintos, existe un fin común a todos ellos que es ir de un punto a otro del entorno evitando obstáculos y demás posibles situaciones excepcionales tales como pasillos bloqueados, que no se caiga por las escaleras, etc. El proceso destinado a la consecución de este fin es lo que se conoce por navegación.

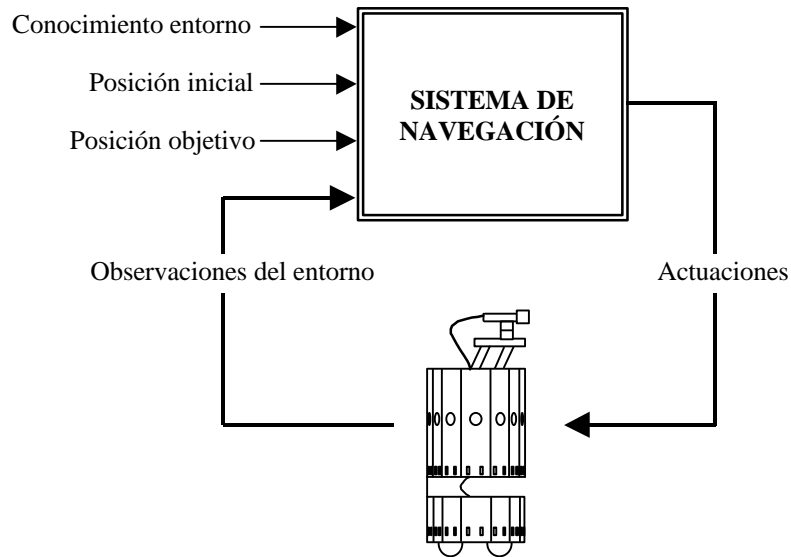


Figura 2.1

Por tanto, en líneas generales se puede considerar el proceso de navegación como aquél cuyas entradas son el conocimiento específico del entorno, descripción de la posición actual, descripción del destino y observaciones del agente sobre el entorno, tal y como se muestra en la figura 2.1. Las salidas producidas son las órdenes de movimiento adecuadas para alcanzar la posición destino evitando obstáculos y otros imprevistos que puedan surgir.

2.3 Tareas incluidas en la navegación

Debido a la complejidad que presenta el proceso de navegación, se suele dividir en subtarefas cuya solución es más fácil de abordar. Esta modularidad facilita al mismo tiempo la adición de otras tareas, así como la detección y recuperación ante posibles fallos, reutilización de los módulos y mantenimiento del sistema. En la mayor

parte de los casos, cada subtarea puede estar formada por uno o varios módulos que se implementan como distintos procesos que intercambian datos a través de diversos mecanismos.

La mayoría las arquitecturas presentan de forma explícita o implícita las siguientes subtarear:

- ✓ Planificación de trayectorias.
- ✓ Evitación de obstáculos.
- ✓ Estimación de la posición.
- ✓ Supervisión, detección y recuperación de errores.

Planificación de trayectorias

El planificador de trayectorias es el encargado de crear una secuencia ordenada de posiciones (puntos, lugares, direcciones o cualquier otro tipo de referencias) por las que tiene que pasar el robot hasta alcanzar la posición destino. Para la determinación de esa trayectoria, se suele utilizar el mapa del entorno en el cual no se incluyen los objetos móviles. La trayectoria calculada puede venir dada de diversas formas según la arquitectura y algoritmos de navegación. En la mayor parte de los sistemas, la planificación de trayectorias se realiza a dos niveles: la *planificación global* y la *planificación local*.

La primera se basa normalmente en un modelo topológico del entorno: entidades o nodos (habitaciones, pasillos, etc.) unidos por arcos (conexiones entre pasillos, puertas, etc.).

La planificación local suele tener un mapa más detallado del área en donde se encuentra el robot con lo cual puede elaborar una trayectoria más precisa en un trozo del camino global. En algunos casos se realiza la actualización dinámica del mapa del entorno al mismo tiempo que el robot sigue la trayectoria planificada. De esta forma, se introducen nuevos obstáculos a medida que el sistema sensorial los va detectando. Bajo tales circunstancias, el planificador local puede replanificar la ruta local cada vez que se encuentran nuevos obstáculos realizándose de esta forma la evitación de los mismos, tarea que, en la mayor parte de los casos, se asigna al control reactivo.

La división entre estos dos tipos de planificación no es siempre clara dependiendo en gran medida del modelo.

En algunos sistemas de navegación basada en visión [Stuck 1995] la descripción de una ruta viene dada por una serie de instrucciones de dos tipos: instrucciones que especifican movimientos a ejecutar en términos de distancias y direcciones, y relación de entidades que el robot ha de ir viendo a lo largo del camino.

Los algoritmos empleados para la construcción de las rutas dependen, en gran medida, de la forma de representación de estas. Cuando la trayectoria se describe de forma topológica o mediante un sistema de celdas, se suelen utilizar algoritmos de búsqueda [Koenig 1996] [Korf 1990] como el A^* , $LRTA^*$, $Min-LRTA^*$, " $1-Step LRTA^*$ ", " $edge counting$ ", " $Node counting$ ", $BETA$, etc. Si el plano es sólo parcialmente conocido se usan otras aproximaciones como la de planificar suponiendo espacio libre a menos que se especifique lo contrario [Koenig 1997] y replanificando de forma dinámica mediante algoritmos como el A^* *dinámico*. Cuando la trayectoria se especifica de una forma más detallada (por ejemplo secuencia de líneas o puntos) se suelen usar otros algoritmos basados en grafos de visibilidad, diagramas de Voronoi, etc.

Evitación de obstáculos

Si los obstáculos con los que se encuentra el robot están modelados en el mapa utilizado por el planificador de trayectorias, éste ha de indicar la ruta correspondiente bordeando dichos obstáculos. Sin embargo, por regla general es poco realista considerar que el robot se mueve en un entorno donde todos los objetos están modelados. En efecto, en una planta de oficinas, por ejemplo, existe un sinnúmero de objetos como sillas, mesas, cajas, etc. que en el transcurso de un día laboral pueden cambiar de posición varias veces. También hay que tener en cuenta la existencia de personas que están constantemente moviéndose de un lugar a otro. Es evidente la necesidad de un mecanismo que permita evitar los obstáculos, tanto estáticos como dinámicos, con los que se puede encontrar el robot.

Si bien algunos sistemas [Murphy 1999] utilizan una planificación local que incluye replanificar cada vez que un objeto nuevo es detectado e integrado en el mapa, en la mayor parte de los casos el camino a seguir permanece inalterado y existe un proceso que se encarga de evitar obstáculos, ajustándose al camino planeado en la medida que sea posible. La razón de esta división radica sobre todo en el tiempo necesario para la replanificación dinámica que la convierte, en la mayor parte de los casos en una solución inviable.

Entre las soluciones presentadas se pueden citar las de campos potenciales [Barraquand 1992], el algoritmo VFH [Borenstein 1991], el método de curvatura de la

velocidad (*CVM*) [Simmons 1996] o su variante el método de curvatura de velocidad usando carriles (*lane-curvature method*) [N.Y. Ko 1998]. En otros casos se emplean métodos basados en lógica borrosa para la realización de esta tarea [Reigner 1994] [Vázquez 1994] [Lin 1997] [Xu 1998].

Estimación de la posición

Para navegar de un sitio a otro, el robot necesita saber, con mayor o menor precisión, la posición en la que está en cada momento o, cuando menos, su posición relativa con respecto al objetivo.

La situación del robot en el plano viene dada normalmente por sus coordenadas cartesianas (x,y) y la orientación (j) o ángulo de giro. El conjunto de estos datos se suele denominar pose. Suponiendo la pose inicial conocida, la actualización de la posición a medida que el robot se mueve se lleva a cabo de muy diversas formas. El agente móvil puede usar herramientas internas como sistemas odométricos, giroscopios, compases magnéticos o bien marcas de referencia externas como el reconocimiento de ciertas características del entorno. En algunos casos se modifica el entorno situando balizas para obtener la pose del robot por triangulación o GPS cuando se trata de entornos abiertos. Estos últimos (GPS) no sirven para interiores y los sistemas de balizas tienen el inconveniente de introducir modificaciones en el entorno.

Los sistemas de odometría determinan la posición del robot integrando el desplazamiento de las ruedas mediante las relaciones dadas por el modelo cinemático del vehículo. Como en todos los procesos de integración, el problema es que los errores, por pequeños que sean, se van acumulando y pueden llegar a ser considerables en grandes desplazamientos. Por esta razón se suelen acompañar de información suministrada por otro tipo de sensores que perciben características del entorno y permiten actualizar la posición del robot. Esta información puede venir de un sistema de visión [Domonte 1997] [Stuck 1995] o bien de otros tipos de sensores tales como de sensores sonar o de un sensor láser de barrido. La integración de la información proporcionada por los distintos sistemas se suele realizar utilizando filtros de Kalman [Jetto 1999], cadenas de Markov [Simmons 1995] [Fox 1998], teorema de Bayes, etc.

Supervisión, detección y recuperación de errores

En actividades como exploración planetaria, trabajo en plantas nucleares, o intervención en ambientes peligrosos para el ser humano, la dinámica del entorno junto

con las restricciones en las comunicaciones exigen robots con un alto nivel de autonomía.

El objetivo final es que el robot realice su tarea de forma autónoma durante largos períodos de tiempo sin necesidad de intervención humana, o que esta sea la menor posible. Debido a que es casi imposible para el programador predecir todas las situaciones en las que se puede encontrar el robot, es necesario establecer un mecanismo para tratar situaciones distintas de las *nominales*. Se denominan situaciones *nominales* aquellas contempladas al realizar el sistema de navegación y situaciones de excepción o excepciones al resto incluyendo, entre otras, fallos de sensores y situaciones del entorno no contempladas. Por otra parte, si se desea lograr mayor autonomía, es necesario que el sistema sea capaz de funcionar incluso ante posibles fallos en alguno de sus componentes.

El funcionamiento de este módulo no debe depender de otros módulos, pero sí obtiene la información acerca del estado del sistema, monitorizando al resto de los módulos y obteniendo “síntomas” acerca de situaciones de excepción [Fernández 1998]. La obtención de algunos síntomas es, por tanto, dependiente del sistema de navegación.

En el resto de este capítulo se tratará la arquitectura del sistema de navegación, dejando para un capítulo posterior la tarea de detección, clasificación y recuperación ante excepciones.

2.4 Arquitecturas de navegación

Del apartado anterior se puede concluir que la navegación es un proceso complejo, que normalmente se divide en tareas para poder tratarlas de forma individual y luego integrarlas formando el sistema de navegación. La arquitectura de navegación define la descomposición y organización de las tareas incluidas en el sistema, así como las estructuras compartidas entre los distintos módulos de control.

Para realizar el proceso de detección de excepciones es necesario conocer la arquitectura de navegación, ya que la forma de obtener la información sobre el estado del sistema depende de la arquitectura y la recuperación de las excepciones debe hacerse en función de los recursos disponibles.

En estos últimos años, como fruto de la investigación sobre robots móviles, se han venido desarrollando distintas arquitecturas para la navegación en agentes móviles.

Al construirse los distintos robots en laboratorios de forma aislada en la mayor parte de los casos, las plataformas son distintas y esto repercute también en la arquitectura de navegación. No obstante, a medida que las plataformas son construidas por compañías (*rwii*, *nomadics*, *labmate*, etc) que proveen una estructura básica para programar las distintas tareas de control, como puede ser *beesoft* o *mobility* para los robots contruidos por *RWI* se van adaptando prototipos que permiten una mayor integración entre trabajos de distintos investigadores.

A pesar de esta disparidad, existen unos objetivos comunes en el diseño de estas arquitecturas. En cuanto a su comportamiento, los atributos más característicos son:

- ✓ **Reactividad:** Debe reaccionar a cambios en el entorno dentro de un período de tiempo razonable. El sistema tiene que ser capaz de controlar el robot moviéndose a velocidades relativamente altas. Esto quiere decir que ciertos procesos como el de evitación de obstáculos deben poseer respuestas en tiempo real y la arquitectura debe facilitar dichos desarrollos.
- ✓ **Robustez:** Un sistema se dice robusto si es capaz de funcionar ante eventos inesperados, ruido en la entrada e incluso algunos fallos.
- ✓ **Fiabilidad:** La fiabilidad se mide por la capacidad del sistema para funcionar sin fallos ni degradación de las prestaciones durante un determinado período de tiempo.

Objetivos adicionales:

- ✓ **Objetivos múltiples.** Debe ser capaz de planificar múltiples objetivos de forma simultánea a la realización de los mismos.
- ✓ **Disponibilidad de aprender:** Es una idea bastante extendida el hecho de que el aprendizaje es una pieza clave en el desarrollo de sistemas inteligentes. La arquitectura debe facilitar el aprendizaje de forma que las prestaciones del robot puedan mejorar con el tiempo y aprenda de sus errores y éxitos.
- ✓ **Facilidad de interacción:** La interfaz de usuario debe ser expresiva y de fácil manejo.
- ✓ **Eficiencia:** No basta con que la plataforma no tropiece con los objetos sino que ha de dirigirse a su objetivo al mismo tiempo. Dicho de otra forma, sus acciones tienen que estar orientadas a la consecución del objetivo y hacerlo de forma eficiente aprovechando al máximo todos sus recursos.

En cuanto al diseño, se desea que el sistema de navegación sea:

- ✓ Modular. Se busca que el sistema sea modular pues ello simplifica el diseño, mantenimiento y detección de fallos. Si se dividen las tareas, algunas de estas sub tareas pueden ser utilizadas por diferentes procesos. Esto, junto a una buena planificación, permite aumentar la eficiencia, además de realizar objetivos parciales que pueden ser interesantes en caso que la tarea global sea inalcanzable.
- ✓ Flexible. Debido a la multitud de situaciones y detalles a tener en cuenta en el desarrollo y puesta a punto, las modificaciones posteriores son inevitables por lo cual se requiere flexibilidad para llevarlas a cabo.
- ✓ Ampliable. Una arquitectura ampliable permite ser usada por investigaciones posteriores, así como añadir nuevos sensores y otros dispositivos. También permite incluir nuevos módulos para incrementar las prestaciones del sistema.
- ✓ General. La arquitectura debe ser capaz de dar soporte a diferentes tareas como construcción de mapas, búsqueda, aprendizaje, etc.
- ✓ Versátil: Debe ser lo suficiente versátil como para poder incluir tanto comportamientos reactivos como deliberativos. También ha de soportar diferentes tipos de solución de problemas a distintos niveles si se trata de una estructura jerárquica.
- ✓ Escalable: Es deseable que el sistema sea escalable y su complejidad aumente con el tiempo haciendo modificaciones mínimas en la estructura inicial.

2.5 Clasificación de las arquitecturas de navegación

Dependiendo del uso de la planificación y observaciones sobre el entorno los sistemas de navegación se pueden clasificar en reactivos y deliberativos. No obstante, existen otras muchas clasificaciones atendiendo a distintos factores:

- ✓ Uso del modelo del entorno (planos CAD, planos topológicos, mapa de celdas, etc.)
- ✓ Distribución del conocimiento (centralizados frente a distribuidos)
- ✓ Grado de escalabilidad

- ✓ Organización de los bucles de control (secuencial, paralelo o mixtos)

Se puede realizar otra clasificación en función de la distribución de los procesos existentes en el sistema de navegación y su interconexión. Si bien esta estructura varía según la arquitectura en particular e incluso la funcionalidad de los procesos puede variar ligeramente según el autor, se pueden reconocer tres grandes grupos: arquitecturas basadas en comportamientos, arquitecturas modulares jerárquicas y arquitecturas basadas en sistemas de pizarra (*blackboard*). A continuación se describen algunos ejemplos y características de las clasificaciones más usadas.

2.5.1 Reactivas frente a deliberativas

Los comportamientos reactivos y deliberativos se distinguen fundamentalmente en tres aspectos:

- ✓ Tiempo disponible para responder a un evento.
- ✓ Nivel de abstracción de la información tratada.
- ✓ Disponibilidad de proyección en el futuro.

Por lo general, los comportamientos reactivos disponen de menos tiempo para responder a los eventos (normalmente se trabaja en tiempo real), trabaja con un nivel de abstracción menor (casi siempre con la información directa de los sensores) y no hace proyección en el futuro. La parte deliberativa se refiere a la planificación como la acción de proyectar la situación actual en el futuro para determinar una serie de acciones que lleven el sistema a la posición destino.

Las arquitecturas clásicas son básicamente reactivas al tratarse de un sistema de control con un único bucle donde la entrada sensorial es filtrada e integrada con el plan para calcular la salida de control a los actuadores. Este esquema se usa con bastante frecuencia en procesos sencillos, pero el problema con plataformas móviles radica en que la velocidad a la que los eventos ocurren (cambios en el entorno) es demasiado alta para que los cálculos computacionales se puedan hacer en un tiempo razonable. Sus realizaciones se han basado sobre todo en comportamientos reactivos en los que la planificación es mínima.

Mientras que en la navegación reactiva la planificación es casi inexistente y el agente se limita a reaccionar a las observaciones sobre el entorno, en la planificación deliberativa pura ocurre todo lo contrario, el agente apenas usa información de las observaciones y sus actuaciones se rigen por un plan trazado de antemano. Cuando se le encomienda una tarea de navegación, el agente genera un plan como una secuencia de acciones para alcanzar el objetivo. Esta secuencia de acciones se crea a partir del conocimiento previo del entorno y de las posiciones inicial y final.

Un método de navegación exclusivamente deliberativo fallará cuando el entorno real difiera del conocimiento previo o expectativas al no poseer capacidad de responder ante eventos u obstáculos no modelados.

Los robots que se basan en sistemas puramente reactivos tienen una serie de limitaciones como son:

- ✓ Falta de flexibilidad. Normalmente modificar su comportamiento requiere un rediseño completo del sistema de control.
- ✓ Son demasiado locales al no hacer previsiones en el futuro y no se obtienen buenas prestaciones donde no hay información local relevante.
- ✓ Ineficiencia al no poseer un plan de actuación, pudiendo incluso darse la posibilidad que el agente no alcance el objetivo.

Estas limitaciones de los sistemas reactivos son precisamente los puntos fuertes de los sistemas deliberativos, de ahí el empeño en combinar ambos sistemas de forma que se complementen. Sin embargo, para el caso de un sistema de planificación automática, el tiempo necesario para producir un plan es, desde el punto de vista comparativo, considerablemente largo, lo cual impide utilizar esta solución dentro del lazo reactivo del sistema de control. El problema es, ¿cómo incorporar cálculos deliberativos relativamente lentos en un sistema que tiene que ser capaz de reaccionar en tiempo real? Se han presentado diversas formas de abordar este problema:

- ✓ **Algoritmos Anytime:** Obtienen una solución factible en un tiempo mínimo y refinan luego la solución si disponen de más tiempo. La calidad de la solución es función del tiempo. Sin embargo, no se puede aplicar este tipo de algoritmo en todos los casos puesto que algunos métodos no se pueden plantear de esta forma.
- ✓ **Planificación probabilística:** El objetivo de esta aproximación es planear una secuencia de acciones las cuales, independientemente del estado inicial y de su evolución, conduzcan al sistema a su objetivo con probabilidad suficientemente alta. La ventaja de esta alternativa radica en que el sistema de control no depende de la incertidumbre en la percepción del entorno por parte del robot ni en sus actuaciones. El problema, en este caso, reside en la complejidad de la construcción del plan, además de que, no siempre está garantizada la existencia de éste.
- ✓ **Planificación reactiva:** Se basa en prever los cambios en el entorno y producir las acciones adecuadas. Los inconvenientes de este sistema vienen de las hipótesis acerca del conocimiento completo del entorno y su

evolución. Aunque dicha hipótesis fuera cierta, no es seguro que la previsión se pueda hacer en tiempo real.

Normalmente, la mayoría de las arquitecturas más avanzadas comparten características reactivas y deliberativas. Así, en [Perret 1996] se mantiene que la parte reactiva y deliberativa son dos actividades que deben permanecer distintas y cooperar. De este modo, su solución se asemeja más a la planificación reactiva. Esta aproximación se fundamenta en el desarrollo de un plan que luego será seguido por el sistema reactivo tal y como se muestra en la figura 2.2.

Hay que resaltar que para que la ejecución de una tarea se realice de forma segura, los planes tienen que ser:

- ✓ Seguros: No deben guiar el robot a situaciones de peligro.
- ✓ Completos: Tiene que tener en cuenta todos los eventos posibles.
- ✓ Orientados al objetivo: Significa que los planes deben conducir el robot al punto de destino fijado.

No obstante, en [Perret 1996] se suavizan estas hipótesis para que el problema sea tratable en complejidad y tiempo. Los sistemas a los que se puede aplicar dicha solución han de suponer que el robot posee un conocimiento exacto de aquellas características de su entorno que sean relevantes a sus tareas o bien las puede obtener a través de una serie de acciones. Estas características incluyen el estado del entorno y su evolución en relación con la actividad del robot lo cual es, por lo general, una visión poco real del problema. También tiene que poseer un modelo de todas las acciones que puede ejecutar y sus posibles consecuencias.

Los operadores usados en la planificación son variantes de los operadores STRIPS con precondiciones y postcondiciones. Se distingue entre *acciones* como operadores sobre las cuales el robot tiene control completo y *eventos*, que no pueden ser controlados por el robot. Cada acción tiene dos listas de efectos: los primeros son los resultados normales o nominales y los segundos los posibles fallos.

El supervisor (figura 2.2) es el encargado de llevar a cabo los planes obtenidos por el planificador actuando sobre el control del robot en función de las observaciones. La estructura de un plan se presenta en forma de árbol como en WARPLAN-C [Warren 1976].

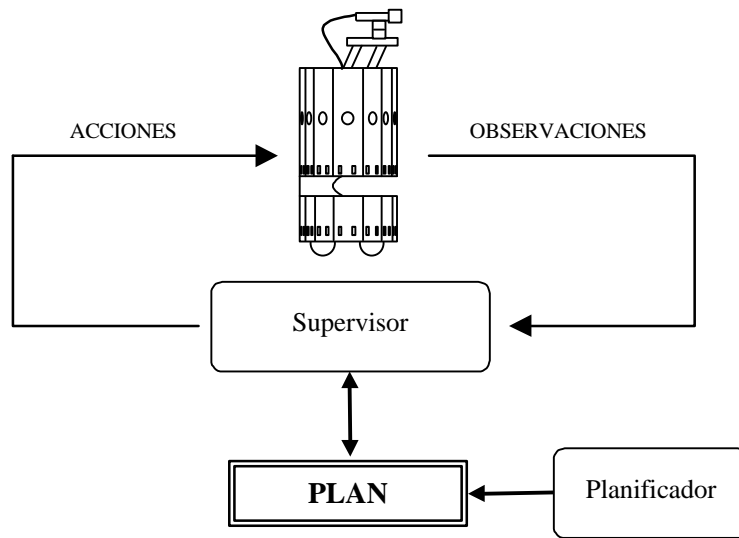


Figura 2.2. Paradigma planificador-supervisor.

Con el mismo objetivo de obtener las ventajas de los sistemas reactivos y deliberativos, se desarrolló la arquitectura TOA (Time Oriented Architecture) [Lewis 1991], donde el control se divide en capas basándose en el tiempo de respuesta requerido y el nivel abstracción de la información.

Como puede apreciarse en la figura 2.3, se trata de un mecanismo de múltiples lazos donde cada problema es tratado en una capa distinta con diferentes métodos. Aquellos situados en la parte inferior (más próximos al hardware) representan los comportamientos reactivos, dado que disponen de menos tiempo para calcular las actuaciones y trabajan básicamente con la información de los sensores. Los lazos situados en la parte más alta trabajan con niveles de abstracción superiores, provistos por las capas inferiores, y disponen de más tiempo para calcular su respuesta, siendo posible la planificación utilizando proyecciones a más largo plazo.

Cada capa de la arquitectura TOA se organiza en una serie de agentes que se comunican a través de mensajes.

En la figura 2.3 se representan los agentes de una capa junto con el flujo de información entre éstos. Se pueden observar tres agentes: uno de *coordinación*, uno de *abstracción* y otro de *descomposición*. Este último se encarga de descomponer las tareas en otras realizables por el nivel inferior y enviárselas. El de abstracción parte de la información sensorial disponible, la procesa para obtener un nivel de abstracción mayor y se la envía a la capa superior. Por último, el de coordinación obtiene las tareas de la capa superior y la información sensorial de la inferior, y se la pasa, posiblemente modificada, a los agentes de descomposición y abstracción respectivamente.

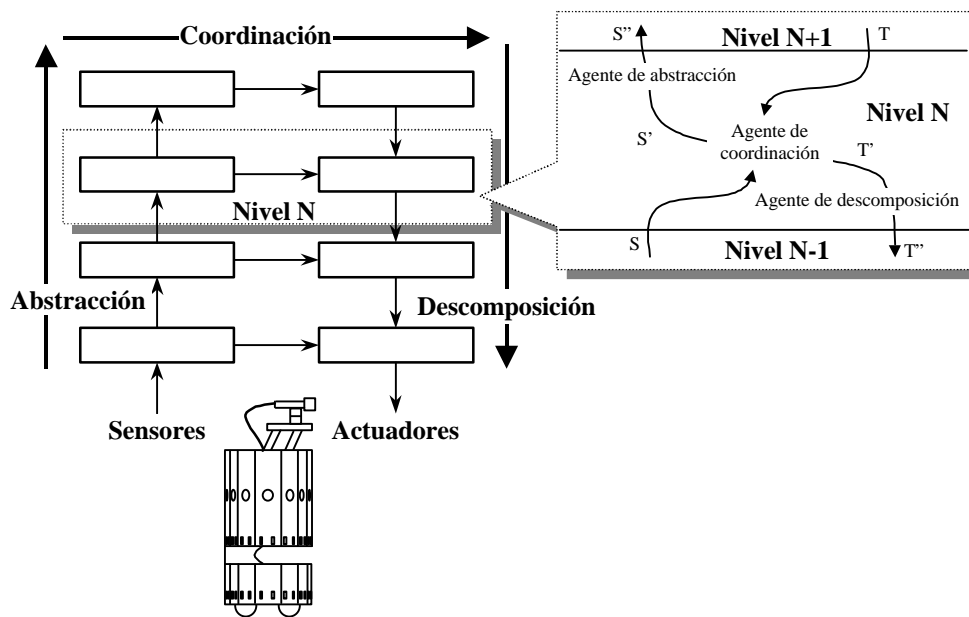


Figura 2.3. Arquitectura TOA

Otros autores [Kurihara 1998] se plantean el dilema reactivo/deliberativo mediante una arquitectura con tres tipos de agentes: agentes que implementan comportamientos, agentes de planificación y agentes de selección de comportamientos que coordinan la activación de los distintos comportamientos. Este tipo de arquitecturas se tratan en el siguiente apartado.

2.5.2 Arquitecturas basadas en comportamientos

La característica más relevante de este tipo de arquitecturas es la descomposición del problema de control en comportamientos. Cada uno de éstos puede controlar de forma directa el robot y están constituidos por un bucle del que forman parte los procesos de observación, cálculo y actuación. La complejidad de estos comportamientos puede ir desde sencillos movimientos a procesos de seguimiento de referencias, evitación de obstáculos, etc.

En este grupo de arquitecturas [Ogasawara 1991] [Brooks 1986] [Saffiotti 1993] [Xu 1997] [Balch 1998] [Parker 1998] [Bernardino 1999] las tareas se implementan como distintos comportamientos que compiten por el control del robot. Las tareas se ejecutan en paralelo produciendo actuaciones deseadas mientras un módulo, que hace de árbitro, decide la actuación final en función de las prioridades de cada uno de los módulos y sus tareas asociadas.

Uno de los puntos fuertes que se atribuyen a estos sistemas es su robustez pues un fallo en un comportamiento no tiene porque llevar a un fallo en el sistema. La respuesta en tiempo real está asegurada por la sencillez de los bucles de los comportamientos al relacionar de forma sencilla la lectura de los sensores con las actuaciones.

Un ejemplo de este tipo de arquitecturas es el bien conocido **SA** (*Subsumption architecture*) que divide el problema de control en comportamientos con complejidad creciente [Brooks 1986]. Se trata de un sistema con múltiples capas donde cada una representa un comportamiento cuyas entradas son directamente las lecturas de los sensores. La salida de las distintas capas es tenida en cuenta de tal forma que los comportamientos de más alto nivel predominan sobre los de bajo nivel. Las realizaciones de esta arquitectura han estado ligadas a comportamientos reactivos, como la evitación de obstáculos y seguimiento de paredes, obteniendo resultados aceptables. Sin embargo, no está nada claro que se pueda utilizar para integrar comportamientos de niveles superiores o deliberativos como por ejemplo la planificación de trayectorias, etc.

Existen arquitecturas similares como **RA** (“Reflexive Architecture”) [Bellingham 1989] donde el problema se divide en comportamientos independientes. Todas las capas monitorizan las lecturas directas de los sensores y pueden disparar diversos comportamientos. Estos tienen un rango asociado que se utiliza a la hora de eliminar contingencias cuando varios quieren controlar el vehículo. Aquellos con mayor rango tienen prioridad sobre los de menor y transmitirán las órdenes al controlador. Este sistema se ha implementado en una plataforma que opera bajo el agua (MIT Sea Grant) con comportamientos de bajo nivel.

Comportamientos

La realización de los comportamientos varia según las arquitecturas. Mientras en [Ogasawara 1991] se usan comportamientos de evitar obstáculos, explorar, construir mapas, identificar objetos y detectar cambios en el entorno, en [Brooks 1986] se usan comportamientos para evitar obstáculos, moverse hacia el objetivo y construcción de mapas distribuidos de acuerdo con la SA. Los comportamientos en [Ogasawara 1991] están distribuidos en una jerarquía de niveles donde los niveles superiores incluyen la funcionalidad de los inferiores, mientras que en [Brooks 1986] dichos comportamientos son independientes. En Flakey [Saffiotti 1993] sólo el nivel inferior esta formado por una serie de comportamientos desarrollados usando reglas borrosas.

En otras arquitecturas se integra este modelo en un nivel de la jerarquía, por ejemplo en [Gómez 1996] se usan comportamientos como “seguir pared”, “seguir pasillo”, “girar”, “detección de colisión”, “modo operador” en la capa de generación de referencias (fig 2.8).

Algunos autores [Xu 1997] incluyen un gran número de comportamientos que abarcan distintas categorías: movimientos reflexivos como “protección” y “evitación”, movilidad como “atracción del objetivo” o “movimientos aleatorios” y adquisición de conocimiento entre los que destaca la “estimación de estado” y “detección de características del entorno”.

También se están utilizando modelos similares en el control de equipos de robots. En [Balch 1998] se describe un ejemplo donde los comportamientos básicos (“moverse hacia el objetivo”, “evitar obstáculos” y “evitar otros robots”) son extendidos con otros como “mantener la formación”.

Decisor

Uno de los principales problemas de este tipo de arquitecturas radica en el control de los comportamientos. En cada instante, el robot sólo puede realizar un tipo de movimiento con lo cual se necesita un sistema de decisión para determinar qué comportamiento controla el robot. Las soluciones adoptadas son muy diversas. Así, en [Brooks 1986] el sistema jerárquico implica que las decisiones tomadas por las capas superiores inhiben las de las capas inferiores (*subsumption architecture*).

En [Ogasawara 1991] el decisor es un poco más complejo basándose en una estructura jerárquica junto con un esquema de probabilidades. Resumiendo, cada uno de los comportamientos C presenta:

a) Función de utilidad:

$$U(x) \Rightarrow x \in X \text{ \{Conjunto de posibles estados del sistema\}}$$

b) Probabilidad de transición:

$$P(x/d, \epsilon) \Rightarrow x, \epsilon \in X; d \in A \text{ \{Conjunto de todas las posibles acciones\}}$$

La acción a ejecutar es aquella que maximize la siguiente ecuación:

$$\sum_{i \in C} \sum_{x \in X} U_i(x) P_i(x(d, \epsilon)) \quad [Eq. 2.1]$$

donde X es el conjunto de todos los posibles estados o situaciones en las que se puede encontrar el sistema y C el conjunto de comportamientos que en este caso son tres.

En [Balch 1998] cada comportamiento (denominado *schema*) genera un vector indicando la actuación deseada. Para el cálculo de la actuación global se tiene en cuenta una ganancia que indica el peso de cada uno de los schemas.

Existen algunas arquitecturas más elaboradas como la presentada en [Parker 1998] donde los comportamientos se dividen en conjuntos de forma que sólo un conjunto puede estar activo a la vez. Los más básicos como evitar obstáculos están siempre activos. La división se realiza porque hay misiones que no pueden ser realizadas en paralelo. Este modelo fue desarrollado para facilitar la cooperación multirobot de forma distribuida y los conjuntos de comportamientos se activan mediante *motivaciones* que indican la disponibilidad del robot a la consecución de una determinada misión.

Otros, como en [Liscano 1995] [Xu 1997], utilizan un sistema de pizarra para compartir información entre actividades y asignar el control del robot a una actividad en cada instante. Estos enfoques se verán un poco más adelante.

2.5.3 Arquitecturas jerárquicas

Este grupo engloba un amplio abanico de arquitecturas sobre las que se trabaja en la actualidad y se caracterizan por poseer una disposición de módulos jerárquica o en forma de árbol. Aquellos situados en la capa superior se encargan de las tareas de alto nivel. Otro conjunto de módulos está destinado al procesado y abstracción de los datos obtenidos por el sistema sensorial. Por último, existe un conjunto que, bajo las

directrices de los módulos del nivel superior y usando los sensores virtuales proporcionados por los módulos de nivel inferior, obtienen las directivas de control del agente. En este caso, no todos los módulos producen señales de control. El fin de alguno de ellos es simplemente proporcionar información a otros de más alto nivel o bien construir un modelo más sencillo a partir del sistema físico.

Otra característica de este tipo de arquitecturas es la división jerárquica en distintos niveles de abstracción de datos. Esta abstracción se realiza tanto con los datos de los sensores (abstracción a medida que se comunican a niveles superiores), como de las órdenes (descomposición de órdenes a medida que pasan a niveles inferiores). Las arquitecturas jerárquicas permiten la división de tareas en subtareas [Kortenkamp 1993] [Diéguez 1995]. No obstante, en muchas ocasiones esto puede degradar en una estructura jerárquica de órdenes.

El principal problema de las estructuras jerárquicas tradicionales radica en su lentitud en la respuesta ante cambios en el sistema, al tener que pasar los datos a través de los distintos niveles y formatos. Otro problema que comúnmente se le atribuye es el de la falta de robustez debido al procesamiento secuencial, dado que un fallo en uno de los niveles probablemente conlleve un fallo en el sistema.

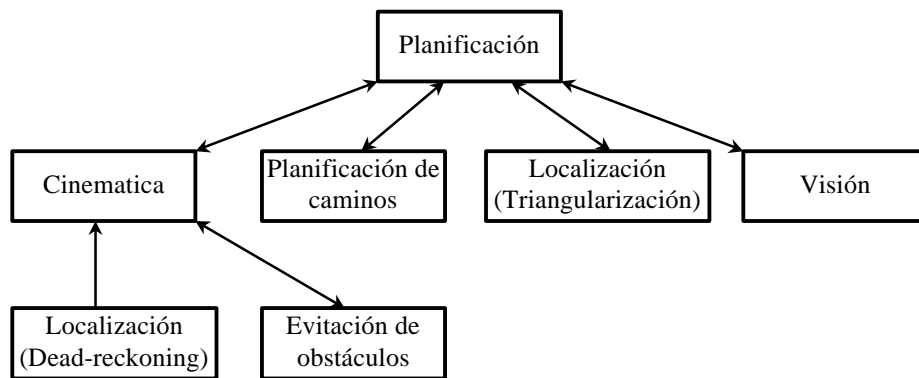


Figura 2.4

En términos de diseño, estos sistemas son fácilmente divisibles en módulos funcionales, aunque algunas estructuras son poco flexibles con vistas a su posterior modificación.

Entre las realizaciones de este tipo se puede destacar la de [Kortenkamp 1993] donde se presenta una arquitectura jerárquica correspondiente al esquema de la figura 2.4. Los módulos de la arquitectura están distribuidos en varios niveles. En el nivel superior un módulo de supervisión decide lo que el robot debe de hacer en cada paso, realizando las correspondientes llamadas a los de nivel inferior. Los módulos del siguiente nivel son los encargados de la planificación de caminos globales, determinar la posición del robot y localización de objetos. Por último, en el nivel inferior se encuentran las rutinas de bajo nivel, encargadas del control de los sensores para realizar las lecturas solicitadas por los demás módulos y tareas reactivas tales como la evitación de obstáculos.

En [Burgard 1998][Beetz 1998] se presenta una arquitectura modular que integra localización, evitación de obstáculos, construcción de mapas y diversos módulos encargados de la interfaz con el usuario.

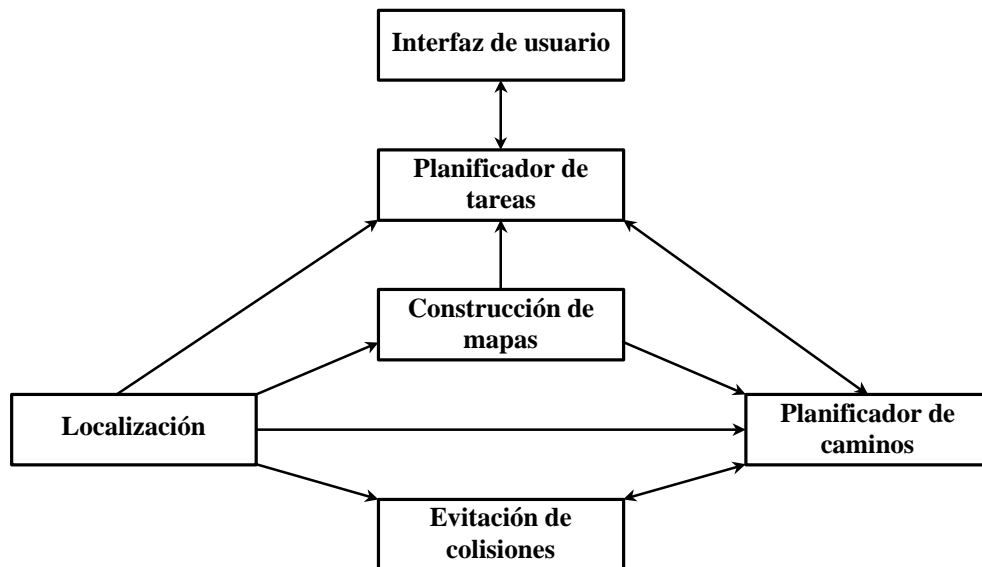


Figura 2.5

El sistema total consta de 25 módulos que son ejecutados como procesos independientes en 3 PCs a bordo del robot y 2 estaciones de trabajo SUN fijas. La transferencia de información entre las distintas tareas se realiza a través de un módulo de comunicaciones denominado *TCX* [Fedor 1993] desarrollado en la universidad de *Carnegie Mellon* y que la empresa *RWII* distribuye junto con su base *B-21*.

La estructura jerárquica simplificada se muestra en la figura 2.5. donde, al igual que en el caso anterior, se han suprimido los módulos de más bajo nivel que controlan directamente a los dispositivos.

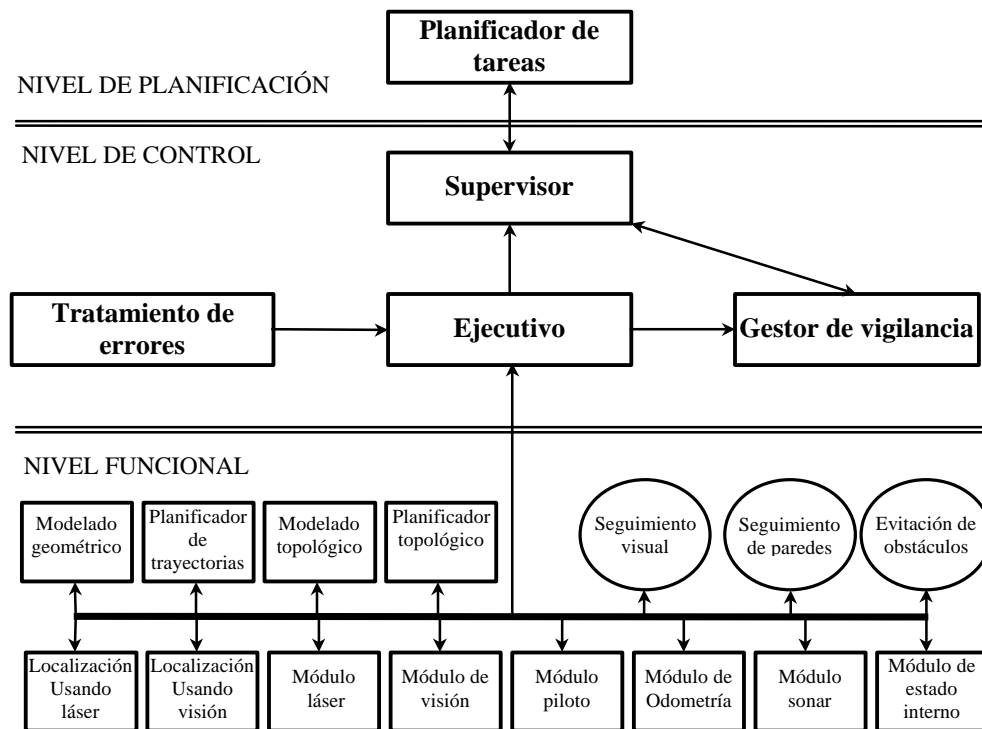


Figura 2.6

En [Noreils 1995] se presenta una arquitectura a tres niveles, tal y como se muestra en la figura 2.6:

- 1) El nivel inferior, denominado nivel funcional, está compuesto por una serie de módulos que proveen al agente las capacidades básicas, tales como obtención de lecturas de los distintos sensores (sonar, laser, odometría o visión), localización y modelado topológico del entorno. También incluye algunos procesos de control (lazo cerrado según el autor), entre los que destaca la evitación de obstáculos, seguimiento de paredes y seguimiento visual. Los módulos de este nivel intercambian datos entre sí y con los del nivel de control.
- 2) El siguiente nivel, o nivel de control, es el encargado de interpretar los planes que elabora el nivel superior, traduciéndolos en tareas a realizar por los distintos módulos del nivel funcional. Se encarga, además, de hacer las correspondientes llamadas a dichos módulos para controlar las acciones del robot y llevar a cabo las tareas encomendadas. Este nivel consta de cuatro módulos tal como se muestra en la figura 2.6: supervisor, gestor de la vigilancia, ejecutivo y módulo de diagnosis y recuperación de errores.
- 3) El nivel superior, denominado nivel de planificación, genera planes en forma de secuencia de tareas para alcanzar los objetivos solicitados por el usuario. Este nivel consta de un único módulo que es el planificador de tareas, si bien se apunta la posibilidad de añadir planificadores más específicos.

Otros autores consideran distintas divisiones como la arquitectura MCA (Model-centered Architecture) que se basa en modelar el entorno en el que opera el agente móvil [Moravec 1988]. También está basado en una estructura en capas donde se usan representaciones del modelo a diferentes niveles de abstracción. Es decir, se utilizará un modelo diferente según se trate de planificación de caminos, navegación, evitación de obstáculos, aprendizaje, etc. Esta estructura ha sido implementada con bastante éxito integrando comportamientos tanto de niveles reactivos (evitación de obstáculos), como deliberativos (planificación de caminos y tareas) en algunas plataformas elaboradas en la actualidad. En la práctica, la mayor parte de las arquitecturas jerárquicas presentan estas características.

2.5.4 Arquitecturas jerárquicas basadas en comportamientos

Para el caso de un sistema de planificación automática, el tiempo necesario para producir un plan es considerablemente largo, lo cual impide utilizar esta solución dentro del lazo reactivo del sistema de control. El problema reside en la incorporación

de cálculos deliberativos, relativamente lentos, en un sistema que tiene que ser capaz de reaccionar en tiempo real. La solución, en la mayor parte de los sistemas, se basa en que ambos tipos de proceso se ejecuten de forma paralela y que la salida de los comportamientos deliberativos sirva de referencia a los reactivos.

Siguiendo estos principios se ha desarrollado un conjunto de arquitecturas jerárquicas en las que existe un nivel reactivo formado por una serie de comportamientos controlados por los niveles superiores.

En [Gat 1991] se define el concepto de *actividad* como una parte del proceso (físico o computacional) que realiza el robot durante un período de tiempo determinado. Algunas de estas actividades se pueden descomponer en otras más sencillas o de bajo nivel que se caracterizan por:

- ✓ Normalmente requieren menos procesado.
- ✓ Las decisiones tomadas afectan a períodos de tiempo más cortos que en el caso de las de alto nivel.

La estructura jerárquica considerada en [Gat 1991] está formada por tres capas (figura 2.7):

- 1) La capa de control es similar a un mecanismo de control clásico de tipo reactivo que consiste en una serie de módulos que calculan funciones de transferencia no lineales. Es decir, se encarga del control de las actividades básicas.
- 2) La capa de secuenciamiento es responsable del control de las actividades a un nivel intermedio formadas por secuencias de otras básicas. Es la encargada de iniciar y parar las actividades básicas y supervisar su funcionamiento. Trabaja a un nivel de abstracción superior al de control y, mientras que las salidas de la anterior eran actuaciones directas sobre los controladores de dispositivos, en este caso son órdenes para iniciar o parar las actividades de la capa de control activando o desactivando distintos módulos.
- 3) Por último, la capa deliberativa es la encargada de realizar las tareas de planificación y mantenimiento del estado del modelo. A su vez se encarga de gestionar las actividades de la capa de secuenciamiento iniciándolas o parándolas cuando sea necesario.

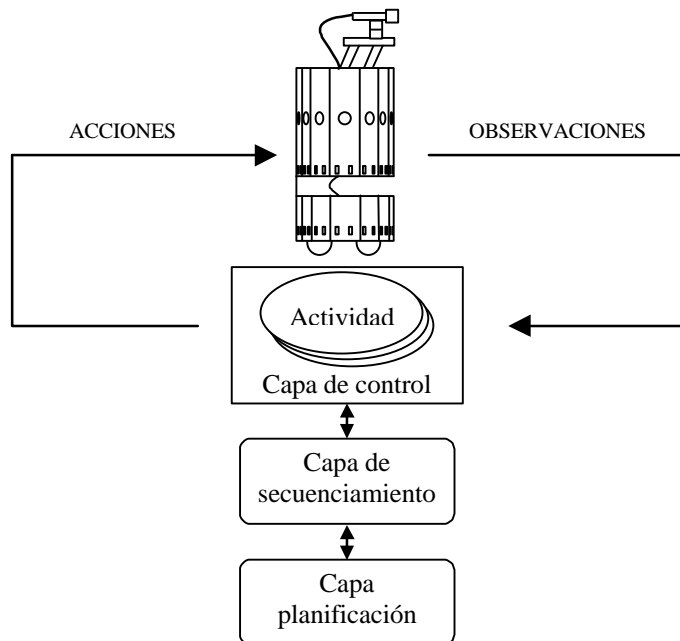


Figura 2.7. Estructura jerárquica en tres niveles (control, secuenciamiento y planificación).

La arquitectura presentada en [Gómez 1996] combina conceptos de dos tipos de arquitecturas ya vistas (basadas en comportamientos y jerárquicas). El robot (AURORA), además de poseer la capacidad de navegación de forma autónoma, puede ser controlado por un operador remoto a distintos niveles (desde tareas a control por *joystick*). El operador puede supervisar, cooperar o tomar parte integral en el control del sistema. El robot ha sido diseñado para aplicaciones agrícolas como sulfatado de plantas. El entorno de trabajo para el que fue concebido es el de invernaderos que poseen la característica de ser bastante estructurados y con alta densidad de obstáculos.

La arquitectura sigue una descomposición jerárquica en cinco niveles (usuario, supervisor, generación de referencias, ejecutivo y servocontrol), como se muestra en la gráfica 2.8. Sin embargo, no se trata de una estructura puramente jerárquica, dado que

las referencias de control son generadas de acuerdo a un esquema de comportamientos o actividades.

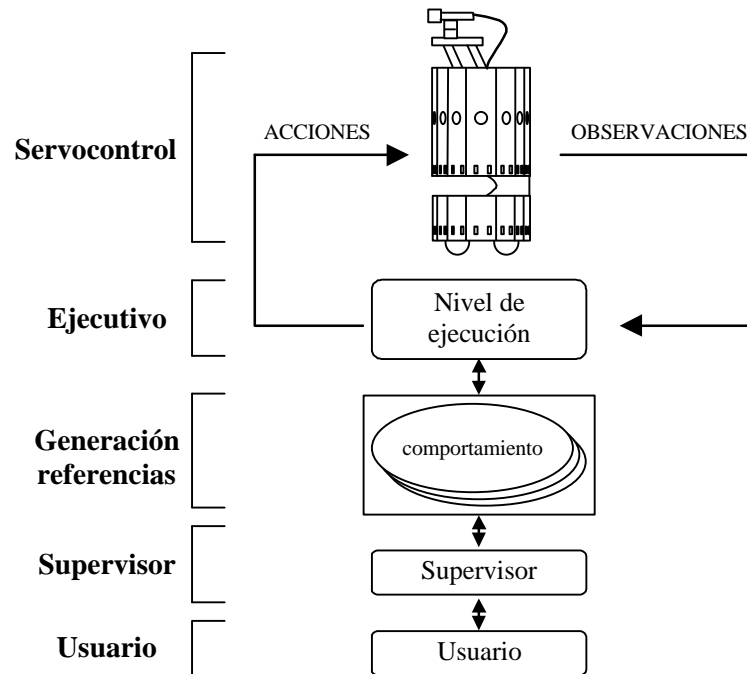


Figura 2.8. Estructura jerárquica en cinco niveles.

Estas actividades pueden incluir planificación, con lo cual no son solamente reactivas y además pueden coexistir varias al mismo tiempo.

Nivel de usuario. Gestiona la interfaz de usuario para la programación de tareas, encendido, apagado, mantenimiento, etc.

Nivel supervisor. Coordina los comportamientos a través de eventos. El conjunto de comportamientos activos, así como el comportamiento predominante, son determinados por los eventos de entrada. A su vez, este nivel genera eventos para iniciar las distintas actividades.

Nivel de generación de referencias. Está compuesto por una serie de comportamientos controlados por el nivel anterior y cuyas salidas se combinan para obtener una única referencia al nivel que se encuentra justamente por debajo. Entre estos comportamientos se pueden mencionar: “seguir pared”, “seguir pasillo”, “girar”, “detección de colisión”, “modo operador”.

Nivel ejecutivo. Este módulo supervisa la activación de los lazos de control de bajo nivel, estableciendo el estado del vehículo (“comienzo”, “normal”, “fallo”, “apagado”) y la transición entre estos. Otra tarea de este nivel es la gestión del sistema de sensores.

Nivel de servocontrol. Se encarga de proporcionar una serie de funciones para controlar los motores y demás dispositivos de abordó. Entre otras tareas, se encarga del control de la velocidad lineal y angular del vehículo, activar los sonar, etc.

Existen otros ejemplos de este tipo de arquitecturas como el presentado en [Behringer 1998] donde se diferencian tres capas. La de más alto nivel es un planificador de tareas que determina cuál es el siguiente objetivo. La capa intermedia es un sistema basado en reglas que actúa sobre la inferior. Esta última asegura el comportamiento reactivo del robot. En otras investigaciones [Masliah 1998] se incluye un operador humano en el bucle de decisión de más alto nivel.

2.5.5 Arquitecturas basadas en sistemas de pizarra (*blackboard*)

Este tipo de sistemas usa módulos distribuidos expertos para observación, acción y procesado. Cada uno de estos agentes tiene sus propios mecanismos y conocimiento local para realizar tareas específicas, y existe además una base de conocimientos común accesible por todos los módulos implicados en el sistema. En los sistemas tradicionales, estos módulos trabajan cooperativamente leyendo y escribiendo información en esta base de datos común para la obtención de referencias o la resolución de un problema. En otros sistemas, como el descrito en [Liscano 1995] [Xu 1997], se usa tanto la base de conocimientos, como el sistema de producción asociado para determinar la actividad o comportamiento que ha de controlar al robot durante el próximo período de tiempo.

La mayor parte de los sistemas de control para robots móviles son mixtos, combinando aportaciones de los modelos jerárquico, basados en comportamientos y en algunos casos, sistemas de pizarra. De esta forma, un tipo de estructuras que está teniendo bastante éxito son aquellas que presentan una descomposición jerárquica con comportamientos reactivos en el nivel inferior, resultando un sistema bastante flexible, capaz de contemplar una mayor diversidad de situaciones y, por tanto, mucho más robusto.

La arquitectura diseñada en [Liscano 1995] usa un sistema de pizarra para coordinar e integrar varias actividades. Definiendo éstas como módulos que desempeñan una determinada función, por ejemplo, seguir una pared o evitar obstáculos. En cierto modo, estas actividades son muy similares a las descritas por Brooks [Brooks 1986] pero no así la forma de control e interacción entre ellas. La realización de una tarea implica a varias actividades que deben ser coordinadas por algún mecanismo, dado que las directrices de control del robot debe venir dadas por una única actividad en todo momento.

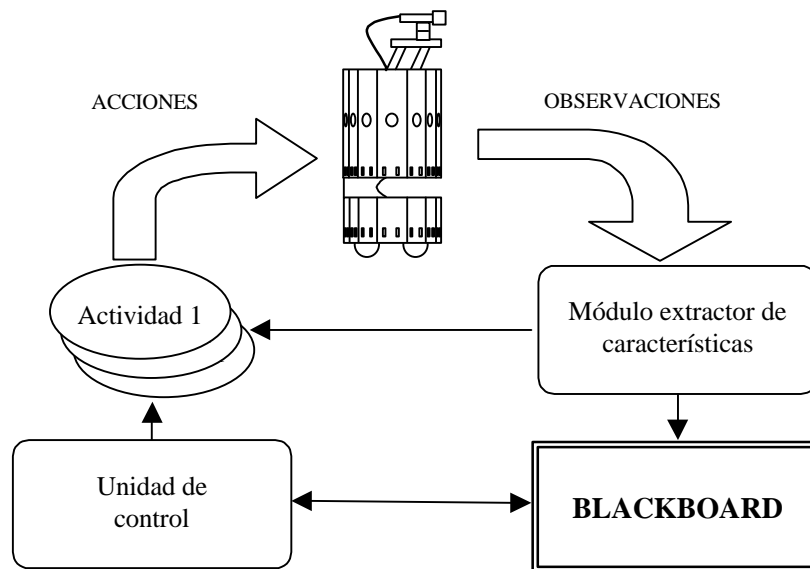


Figura 2.9.

En la figura 2.9 se muestra de forma esquemática la descripción funcional de esta arquitectura. Se trata de una arquitectura híbrida, entre las basadas en comportamientos y las basadas en sistemas de pizarra, donde los comportamientos o actividades son coordinadas por una unidad de control basada en un sistema de pizarra. De esta forma, se separa la parte reactiva, llevada a cabo por las actividades, de la parte de planificación y razonamiento estratégico, regulada por el sistema de producción asociado al de pizarra. Las actividades se ejecutan de forma paralela pero sólo una

controla el robot. Es el sistema de pizarra el encargado de determinar quien controla el agente, usando el conocimiento que posee acerca del estado de las actividades (publicado por éstas en la base de conocimientos) y la especificación del usuario.

El extractor de características se encarga de procesar la información sensorial y expresarla en distintos formatos para las actividades y la base de conocimientos.

Las actividades determinan la dirección y velocidad del robot en función de la información obtenida del extractor de características. A su vez, reciben referencias de la unidad de control, dado que ésta es la que tiene constancia del estado global del robot.

Las actividades implementadas en [Liscano 1995] son:

- ✓ *Evitación de colisiones*: se encarga de guiar al robot a través del espacio abierto siguiendo una dirección de referencia, utilizando un mapa de celdas del entorno construido a partir de las lecturas sonar. Aunque utiliza información bastante precisa acerca del entorno del robot, ésta no es enviada directamente a la pizarra sino una información cualitativa en los cuatro puntos cardinales (delante, detrás, derecha, izquierda). Este filtrado se realiza para evitar sobrecargar la pizarra con información y reducir el ancho de banda requerido entre módulos.
- ✓ *Anomalías en el suelo*: Esta actividad se encarga de guiar al robot para lograr que éste supere ciertas anomalías en la superficie del suelo tales como posibles surcos, un cable, etc.
- ✓ *Planificación de caminos*: Tiene asignada la función de planificar caminos alrededor de objetos que la actividad de evitación de obstáculos no podría superar.

El módulo del sistema de pizarra consiste en una base de conocimientos donde se encuentra la información global del sistema y de una unidad de control basado en reglas de producción que opera sobre la base de conocimientos.

En [Xu 1997] el control de los distintos comportamientos es planteado como un mecanismo de solución de problemas usando un sistema de pizarra. Para ello se utiliza un método denominado de *atención* basado en la atención que prestamos los humanos para realizar distintas tareas. Este parámetro define la necesidad de un comportamiento en obtener el control del robot y varía con el tiempo debido a diversos factores. Por ejemplo, la atención del comportamiento de evitación de obstáculos se incrementa cuando hay obstáculos muy cercanos.

Arquitecturas similares pueden verse por ejemplo en [Piaggio 1998] donde la información es almacenada en una memoria activa (AIM). Los agentes (denominados expertos) pueden leer y actualizar los datos almacenados en esta memoria.

2.6 Control de alto nivel e interconexión entre módulos

Como se ha visto anteriormente, en la mayor parte de las arquitecturas existen diversos módulos encargados de realizar distintas tareas. Esto hace necesario un mecanismo de control de alto nivel que se encargue de coordinar el funcionamiento de los demás. Es evidente que hay arquitecturas, tales como las puramente reactivas, que no tienen control de alto nivel.

Existen distintas plataformas para el control de alto nivel y coordinación entre los módulos que forman parte del agente.

En las arquitecturas pizarra, la comunicación entre módulos se lleva a cabo escribiendo y leyendo mensajes de la pizarra [Congdom 1993], mientras que en las arquitecturas jerárquicas se acostumbra a establecer un mecanismo específico para cada caso (por ejemplo el TCA [Simmons 1991b]).

En Flakey [Congdom 1993] (un robot construido en el “Stanford Research Institute”), la coordinación entre procesos se realiza usando una estructura de pizarra (**LPS**). El sistema de pizarra posee una estructura que almacena la información a intercambiar por los módulos y mecanismos de control de acceso a los datos de esa estructura, como se ha visto anteriormente para [Liscano 1995]. Algunos de estos módulos son productores de datos (por ejemplo, aquellos que obtienen datos sensoriales) que pueden ser necesarios a otros y, por tanto, los publican en la pizarra mientras otros módulos leen estos datos de esta pizarra cuando sea necesario. Entre otras estructuras de pizarra usadas en agentes móviles se encuentra el **BB1** [Hayes 1989] desarrollado en Lisp o su versión en C++ denominada **BBK**.

Rhino [Burgard 1998] usa un módulo de comunicaciones de procesos **TCX** [Fedor 1993], desarrollado en la universidad de Carnegie Mellon, basado en paso de mensajes. Un poco más elaborado, el **TCA** [Simmons 1991b], usado en robots como Ambler [Simmons 1991] Odysseus o Xavier [Simmons 1999], presenta una filosofía similar.

Tanto el **BB1** como **TCA** incorporan mayor funcionalidad que la simple comunicación entre módulos, proveyendo ventajas en términos de descomposición, coordinación y planificación de tareas. En la arquitectura BB1 el uso de un plan de control facilita la restricción de recursos por las tareas, entrelazar ejecución con planificación y el establecimiento de tareas de monitorización. Sin embargo, existe un riesgo mayor de embotellamiento dada la cantidad de información que en un robot debe mantener la pizarra mientras que otras arquitecturas como **TCA** o **TCX** dicha información se encuentra en los respectivos procesos, transferiéndose solamente cuando sea necesario tomar alguna decisión basada en estos datos en otro módulo.

Este último tipo de soluciones se tratará más en detalle en el capítulo quinto cuando se describa la arquitectura usada en esta tesis.

3 Detección, diagnosis y recuperación de fallos

3.1 Introducción

La forma en que distintos autores enfocan el problema del tratamiento de fallos (detección, diagnosis y recuperación) es muy variada, debido en gran medida a la diversidad de tipos de fallos existentes y los métodos de recuperación disponibles. Esta disparidad se hace también patente en la terminología, dando lugar a que la nomenclatura utilizada por los diversos autores no siempre coincida. Para unificar criterios en cuanto a definiciones de los vocablos empleados en este tema, el comité técnico del IFAC (*International Federation of Automatic Control*) denominado *SAFEPROCESS (Fault Detection, Supervision and Safety for Technical Processes)* [Iserman 1997] ha promovido una iniciativa para definir una terminología común que sirve como referencia a lo largo de esta tesis. A continuación se describen algunos de estos términos.

En cuanto a estados y señales, se utilizan los siguientes conceptos:

- ✓ **Fallo:** Desviación no permitida de al menos una característica, propiedad o parámetro del sistema con respecto a su valor aceptable, nominal o estándar.
- ✓ **Avería:** Interrupción permanente de la disponibilidad del sistema a realizar una función requerida bajo condiciones de operación especificadas.
- ✓ **Error:** Desviación entre el valor medido o calculado a partir de una variable de salida del sistema y el valor correcto, de acuerdo con las especificaciones o calculado teóricamente.
- ✓ **Residuo:** Indicador de fallo, basado en una desviación entre medidas del sistema y cálculos obtenidos del modelo del mismo.
- ✓ **Síntoma:** Desviación de un parámetro observable de su valor nominal (valor o rango de valores que puede tener cuando el sistema funciona correctamente).

Por lo que respecta a las propiedades del sistema, se distinguen los siguientes conceptos:

- ✓ **Fiabilidad:** Disponibilidad del sistema a realizar una función requerida bajo ciertas condiciones, en un determinado ámbito y durante un período de tiempo establecido.
- ✓ **Seguridad:** Disponibilidad de un sistema a no causar daños a personas, equipos o al entorno.
- ✓ **Disponibilidad:** Probabilidad de que un sistema opere satisfactoriamente en cualquier momento arbitrario.

También se usarán a menudo otros conceptos definidos por el comité anteriormente mencionado:

- ✓ **Monitorización:** Tarea continua en tiempo real que determina las condiciones del sistema recopilando información del mismo, reconociendo y notificando anomalías.
- ✓ **Supervisión:** Además de la monitorización del sistema, toma las acciones oportunas para mantener la operación en caso de fallos.

A partir de las anteriores definiciones se puede considerar que, en general, el tratamiento de errores en sistemas de control automático se puede realizar a distintos niveles:

Nivel 1. Monitorización: Se generan alarmas para el operador cuando alguna de las variables observables del sistema se encuentra fuera de las tolerancias permitidas. Se trata, pues, de un sistema de generación de alarmas dependiente de las tolerancias de las variables medidas en el proceso.

Nivel 2. Supervisión: Al igual que en el caso anterior, se verifica que el valor de las variables observadas en el proceso estén dentro del rango de tolerancias permitidas. La diferencia estriba en que en este caso en lugar de notificarlo al operador mediante un sistema de alarmas, se toman las acciones oportunas para recuperar el normal funcionamiento.

Nivel 3. Supervisión con diagnosis de fallos: Las variables obtenidas del sistema son procesadas para obtener síntomas que servirán para diagnosticar los posibles fallos del sistema y, posteriormente, tomar las acciones oportunas. La diferencia entre este caso y el anterior estriba en la existencia de un proceso de diagnosis completo.

La principal ventaja del primer nivel (monitorización) es su sencillez y, por tanto, rapidez en la detección, pero sólo es capaz de detectar fallos producidos por grandes cambios en una determinada característica, o bien, pequeños errores incrementales durante un largo período de tiempo. Otro inconveniente de este nivel es que deja al operador la tarea de recuperación.

El segundo nivel (supervisión) también suele ser sencillo de implantar e incluye mecanismos de recuperación, pero no contempla un sistema de diagnosis capaz de detectar y tratar fallos que no sean directamente observables. Solo serán considerados los detectados utilizando rangos de variables observables o bien relaciones simples entre éstas. Se tiende, en lo posible, a minimizar la intervención del operador humano dejando al nivel más sencillo el sistema de alarmas.

En cierta medida, el nivel de supervisión engloba los demás niveles. En efecto, si se considera un sistema de diagnosis muy sencillo basado en la tolerancia de las variables, el sistema de diagnosis se reduce al nivel dos. Si además se considera como acción de recuperación la de avisar al supervisor (si ésta se puede considerar como tal), se convierte en una monitorización (nivel 1).

En muchos sistemas industriales conviven los tres niveles donde las alarmas son generadas mediante monitores, los fallos más sencillos mediante simples supervisores y aquellos que presentan una mayor complejidad, supervisores con diagnosis.

La mayor parte de la investigación en cuanto a tratamiento de fallos se ha llevado a cabo en procesos industriales modelando el sistema a controlar matemáticamente y comparando los datos reales con los estimados según el modelo. Algunos de los métodos son aplicables a agentes móviles, si bien los métodos de navegación presentan aspectos particulares al problema, tal como se verá más adelante. En este capítulo se presenta en primer lugar las fases del sistema de tratamiento de fallos y a continuación se describe el funcionamiento de cada una: generación de síntomas (apartado 3), diagnosis (apartado 4) y recuperación (apartado 5).

3.2 Fases en el tratamiento de errores

De acuerdo con la terminología empleada en el comité técnico del IFAC SAFEPROCESS, la tarea de diagnosis de fallos puede dividirse en diferentes funciones:

- ✓ *Detección del fallo*: Determinación de la existencia de algún fallo en el sistema.
- ✓ *Ubicación del fallo*: Sería el paso siguiente al anterior, se trata de aislar el fallo, determinar la posición del mismo, tanto en lugar como tiempo.
- ✓ *Identificación del fallo*: Después de la detección y ubicación, se estima el tipo o naturaleza del fallo y dimensiones o alcance del mismo.
- ✓ *Diagnosis del fallo*: Incluye los tres anteriores. Es decir, determina la existencia del fallo, posición, tipo y tamaño.

El proceso completo de tratamiento de fallos se divide en tres fases, tal como se muestra en la figura 3.1. En primer lugar, el módulo generador de síntomas se encarga de obtener éstos a partir de medidas del sistema. A continuación, el módulo de diagnosis determina la existencia de posibles fallos y, en caso de detección positiva, ubica e identifica el fallo y sus características. Por último, el proceso de recuperación trata de volver el sistema a su funcionamiento normal. En la figura también se puede observar la división de la tarea de diagnosis en las funciones de detección, ubicación e identificación. Estas tres funciones no están siempre desarrolladas de forma aislada como es el caso de las de ubicación e identificación que, en la mayoría de los sistemas, aparecen como un único módulo.

La generación de síntomas puede realizarse a partir del conocimiento analítico acerca del proceso, o bien, conocimiento heurístico a través de observación humana.

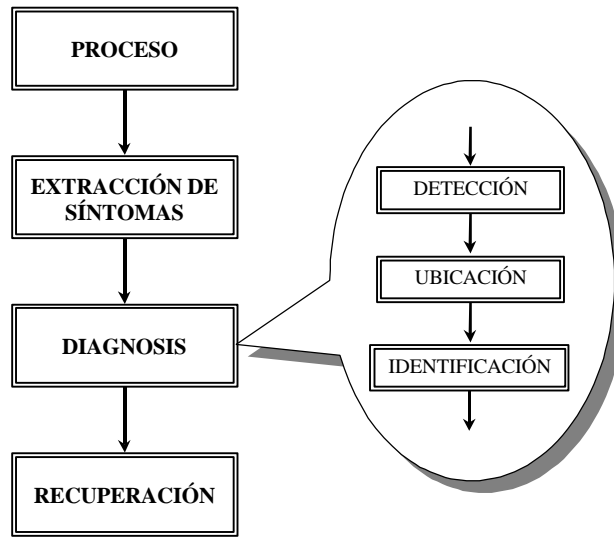


Figura 3.1 Fases del tratamiento de fallos.

Generación analítica de síntomas

Se usa conocimiento analítico acerca del proceso para generar síntomas sobre posibles fallos. Este conocimiento se obtiene a partir de variables observables en el sistema. El procesamiento de estos datos para la extracción de valores característicos se puede realizar de distintas formas:

- 1- Comprobación de valores límite de variables medidas sobre el proceso. Es necesario el conocimiento previo de rangos y tolerancias de las distintas señales del sistema.
- 2- Análisis de señales observadas usando medidas, tales como correlación, espectro en frecuencia, estimación de modelos paramétricos o valores característicos (variancias, amplitudes, parámetros modales, etc.).
- 3- Seguimiento del proceso usando modelos matemáticos del mismo mediante estimación de parámetros, modelos de estado, etc.

En muchos casos, se extraen características especiales del sistema físico tales como coeficientes o residuos. Estas características se comparan luego con las que el sistema debería tener en funcionamiento normal, verificando si están dentro de un margen de tolerancia. Los resultados de estas comparaciones son usados después por el sistema de diagnosis de fallos.

Generación heurística de síntomas

Además de la generación de síntomas usando información analítica, se pueden obtener síntomas a partir de información cualitativa suministrada por algún operador humano. Esta información puede ser proporcionada de forma cualitativa mediante lógica borrosa, variables lingüísticas (pequeño, grande, ...), etc. El operador obtiene esa información de su experiencia con el sistema, como por ejemplo, que un determinado ruido en un motor suele preceder a un fallo en el mismo, que una combinación de valores en el panel de control es un preludio de futuros problemas en el sistema, etc.

Diagnosis de fallos

La detección consiste simplemente en determinar si existe un fallo, mientras que la identificación es el proceso que clasifica el tipo del fallo. Aunque en algunos sistemas este módulo y el de diagnosis van unidos, la ventaja de separarlos [Stuck 1992] [Stuck 1995] radica en que el proceso de detección, por lo general, tiene menor carga computacional de forma que el módulo de diagnosis sólo se activará cuando el de detección encuentre un error.

La tarea de ubicación e identificación de fallos consiste en determinar las características del fallo (tipo de fallo, localización del fallo, tiempo en que se produjo, etc) que el sistema necesita con vistas a su posterior recuperación. Para ello, se parte de los síntomas analíticos y heurísticos proporcionados por el mecanismo de generación. Cuando no se dispone de información de causalidad acerca de los posibles fallos se pueden usar mecanismos de clasificación, tales como los geométricos, redes neuronales o lógica borrosa. Si, por el contrario, se dispone de información de causalidad acerca de los fallos-síntomas, se pueden usar estrategias de diagnosis basadas en lógica propositiva [Russell 1996] cuyo núcleo es un motor de inferencia que trabaja sobre un conjunto de reglas almacenadas en una base de conocimiento y hechos (datos) obtenidos del funcionamiento del sistema

3.3 Generación de síntomas

El objetivo principal de la generación de síntomas es la reducción de los datos a procesar en las fases de detección y diagnosis filtrando información no relevante.

En estos últimos años se han desarrollado muchos métodos y la elección de un método específico depende del proceso a supervisar y de los síntomas a extraer. No obstante, se pueden distinguir dos tipos generales dependiendo de las fuentes de datos sobre las cuales se obtendrán los síntomas: Generación de síntomas a partir de conocimiento analítico acerca del proceso o bien conocimiento heurístico a través de la observación humana.

3.3.1 Generación analítica de síntomas

Aquí se usa conocimiento analítico del proceso para generar síntomas sobre posibles fallos basándose en las variables obtenidas del proceso a supervisar. La generación analítica de síntomas puede considerarse, pues, una función G del espacio de señales al espacio de síntomas:

$$\begin{aligned} G: \mathfrak{R}^k &\rightarrow \mathfrak{R}^m \\ x &\rightarrow G(x) = s \end{aligned}$$

donde k es la dimensión del espacio de las señales que se obtienen del proceso y m es la dimensión del espacio de síntomas producidos. La mayor parte de las aproximaciones que se han elaborado en estos últimos años [Chen 1998] se basan en la detección de síntomas usando dependencias entre señales medidas en el proceso real, que son comparadas con las obtenidas mediante modelos matemáticos del proceso. De esta forma, se obtiene una señal de error que será usada por el módulo de detección para la determinación de la existencia de posibles fallos en el proceso, actuadores o sensores.

La generación de síntomas puede dividirse en dos fases: primero se extraen datos del proceso y, a continuación, se obtienen los síntomas a partir de la información generada en la primera fase y los parámetros de referencia obtenidos a partir del modelo del proceso. En el próximo apartado se comentan de forma breve los modelos matemáticos del sistema más utilizados (proceso a controlar y fallos). Luego, en la siguiente sección se verá la extracción de datos

3.3.1.1 Modelado del sistema

El término genérico sistema, en este caso, incluye tanto el proceso a controlar como la parte de control, dado que se pueden producir fallos en ambos. La manera de modelar dicho sistema depende en gran medida del proceso a supervisar. Algunos son demasiado complejos y se modela solamente una parte del mismo. La detección de errores en otros se basa, simplemente, en aprovechar la información que se dispone acerca de algunas señales, tales como valores mínimos, máximos, estadísticos, derivadas (aceleraciones), caracterización espectral, etc.

Modelado del proceso

En la mayor parte de los sistemas de detección de fallos que usan modelos matemáticos, el proceso viene dado por una serie de funciones que caracterizan su comportamiento o salida en función de las entradas pudiendo haber al mismo tiempo observaciones de otras variables que forman parte del proceso.

Afortunadamente, el comportamiento de muchos sistemas físicos se puede describir mediante una ecuación diferencial lineal. Para una gran parte de los sistemas restantes también se puede usar una aproximación lineal de sus funciones de transferencia sobre un rango específico de operación. Es por ello muy común describir el comportamiento del proceso mediante una ecuación diferencial lineal ordinaria usando en otros casos ecuaciones de estado u otras soluciones similares.

Especial importancia para el caso de agentes móviles lo constituyen aquellos sistemas cuyo comportamiento es demasiado complejo para poder modelarse matemáticamente, pero sí se puede prever que los valores de alguna de las salidas deben pertenecer a un conjunto posible de valores [Kleer 1987], o bien, estar dentro de un intervalo con cierta probabilidad [Ferrell 1994]. Por último, también hay sistemas que, si bien no se puede modelar su comportamiento, se puede comprobar si determinadas acciones se han llevado a cabo correctamente o, por el contrario, no se han realizado, en cuyo caso, se puede seleccionar una serie de tests (en forma de árbol por ejemplo [Sankaran 1977] [Noreils1990]), o bien, usar información sobre el pasado para diagnosticar posibles fallos [Stuck 1992][Stuck 1995].

Modelado de los fallos

Como ya se ha mencionado, un fallo es una desviación no permitida del valor de una variable característica del sistema con respecto a su valor nominal. Ello no quiere decir que un fallo conlleve siempre una malfunción en el sistema, pero sí que puede ser una posible causa. En [Isermann 1997] se clasifican los fallos con respecto al proceso en aditivos y multiplicativos. Se consideran fallos aditivos f con respecto a una variable y a aquellos que se superponen a la señal, de forma que si ésta fuera observada, el resultado sería la suma de los dos valores ($f+y$). En los fallos multiplicativos, por el contrario, la señal resultante es el producto de ambas ($f*y$).

3.3.1.2 Extracción de datos

El conocimiento analítico acerca del proceso se usa para producir información analítica cuantificable. A su vez, esta misma información se extrae del proceso real y la comparación de esta información, obtenida por dos fuentes distintas, da lugar a señales de referencia que servirán para generar finalmente los síntomas. Los datos a comparar pueden ser las salidas del proceso, los parámetros de la ecuación de transferencia, el estado (caso de utilizar ecuaciones de estado) o cualquier señal intermedia del proceso cuyo comportamiento sea conocido o se pueda estimar a partir del modelo. La extracción de datos del proceso real puede consistir, por tanto, en la simple observación de una señal o tareas más complejas como la estimación de parámetros o la estimación del estado a partir de otras señales medidas en el proceso.

Comparación de las salidas del modelo

Si se considera el modelo del proceso totalmente conocido, el método más directo de obtener síntomas acerca de posibles fallos es la comparación entre las salidas del modelo G_m y el sistema real G_r , tal y como se muestra en la figura 3.3. La diferencia o residuo $r(s)$ entre ambas salidas es:

$$r(s) = G_m(s)u(s) - G_r(s)u(s) \quad [Eq. 3.1]$$

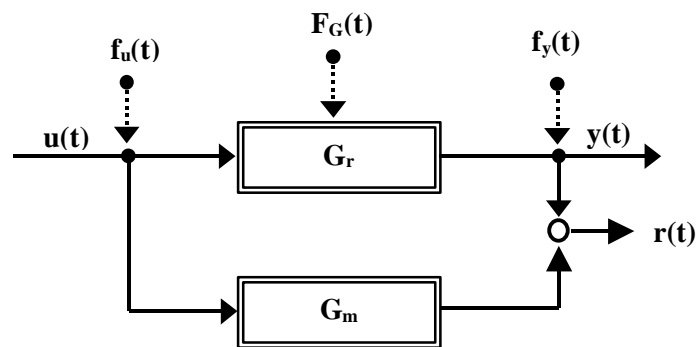


Figura 3.3. Comparación de salidas

Existen otras posibilidades, como por ejemplo generar una señal de acuerdo con la figura 3.4 obteniéndose un residuo en forma de polinomio:

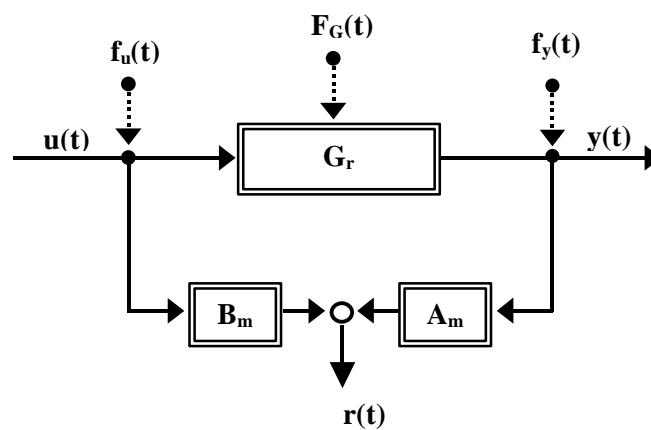


Figura 3.4. Ecuación de error

$$r(s) = A_m(s)y(s) - B_m(s)u(s) \quad [Eq. 3.2]$$

A las ecuaciones 3.1 y 3.2 se les suele denominar ecuaciones de paridad [Gertler, 1991] . A estas señales residuales se les aplica normalmente una serie de filtros con el objetivo de extraer algunas de sus propiedades y eliminar información irrelevante:

$$r'(s) = F(s)r(s) \quad [Eq. 3.3]$$

donde $F(s)$ representa al filtro y $r(s)$ al residuo anteriormente calculado. En [Isermann 1997][Gerler, 1991] o [Patton 1991] se describe la aplicación de filtros para distintos tipos de sistemas.

Estimación de parámetros

Existen varias formas de extraer los parámetros dependiendo del modelo del sistema. Uno de los más extendidos se realiza usando el error de la salida y mínimos cuadrados. Se basa en minimizar el error cuadrático medio de la salida como se muestra, por ejemplo, en [Isermann 1997].

Modelando el proceso de la siguiente forma:

$$y(t) = \bar{\Psi}^T(t) \bar{\Theta}$$

donde:

$$\bar{\Psi}^T(t) = [-y^1(t) \dots -y^n(t) \quad u(t) \dots u^m(t)]$$

$$\bar{\Theta} = [a_1 \dots a_n \quad b_1 \dots b_n]$$

El error entre las salidas viene dado por¹:

¹ Se ha denotado los valores estimados usando el acento circunflejo (^) encima de la variable que los representa.

$$e(t) = y(t) - \bar{\Psi}^T(t) \hat{\Theta} \quad [Eq. 3.4]$$

Después de muestrear la señal, solamente para la estimación de parámetros, se pueden aplicar las fórmulas para minimizar el error cuadrático medio:

$$V = \sum_{k=1}^N e^2(k)$$

$$\frac{dV}{d\hat{\Theta}} = 0$$

Obteniéndose:

$$\hat{\Theta}(N) = [\bar{\Psi}^T \bar{\Psi}]^{-1} \bar{\Psi}^T \bar{Y} \quad [Eq. 3.5]$$

Normalmente, la estimación paramétrica se realiza de forma recurrente aplicando las siguientes ecuaciones:

$$\hat{\Theta}_{k+1} = \hat{\Theta}_k + \bar{\Gamma}_k (y_{k+1} - \bar{\Psi}_{k+1}^T \hat{\Theta}_k) \quad [Eq. 3.6]$$

$$\bar{\Gamma}_k = \frac{1}{\bar{\Psi}_{k+1}^T \bar{P}_k \bar{\Psi}_{k+1} + 1} \bar{P}_k \bar{\Psi}_{k+1} \quad [Eq. 3.7]$$

$$\bar{P}_{k+1} = (I - \bar{\Gamma}_k \bar{\Psi}_{k+1}^T) \bar{P}_k \quad [Eq. 3.8]$$

En [Isermann 1992] se describen otras formas de estimación de parámetros usando el error de salida.

Estimación de estados

Otra forma de extracción de datos se basa en la estimación de los estados para luego obtener los síntomas. Basándose en las ecuaciones de estados, se trata de obtener los indicios de fallo estimando el estado del sistema \mathbf{x} . Para ello, considerando

conocidos los parámetros del sistema, se utiliza un observador para estimar las variables de estado que no pueden ser medidas directamente. Este observador utiliza las entradas y salidas medidas para obtener los estados como se muestra en la figura 3.5.

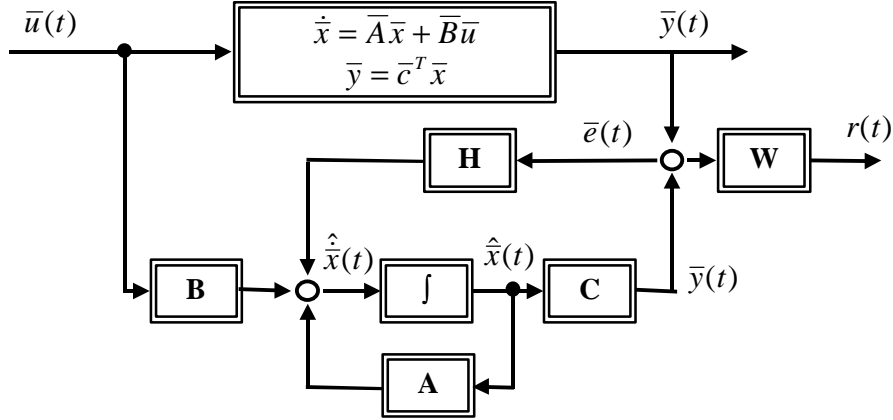


Figura 3.5.

Figura 3.5.

$$\begin{aligned}\hat{\dot{x}}(t) &= \bar{A}\hat{x}(t) + \bar{B}\bar{u}(t) + \bar{H}\bar{e}(t) \\ \hat{\dot{y}}(t) &= \bar{c}^T \hat{x}(t)\end{aligned}\quad [Eqs. 3.9]$$

Definiendo el error en la estimación del estado como:

$$\tilde{x}(t) = \bar{x}(t) - \hat{x}(t) \quad [Eq. 3.10]$$

De las ecuaciones 3.9 y 3.10 se obtienen las ecuaciones 3.11.

$$\begin{aligned}\dot{\tilde{x}} &= \dot{\hat{x}} - \dot{\hat{x}} \\ \dot{\tilde{x}} &= [A - HC]\tilde{x}(t)\end{aligned}\quad [Eqs. 3.11]$$

Este error tiende asintóticamente a 0 si el observador es estable, lo cual puede garantizarse con el adecuado diseño de H . Puede comprobarse [Isermann 1997] que señales de ruido sumadas a la entrada o salida tienen un efecto aditivo en la señal de error, mientras que, variaciones en los parámetros del proceso (ΔA , ΔB , ΔC) tienen un efecto multiplicativo.

Los métodos de detección de fallos en procesos con múltiples salidas que siguen este esquema son muy variados dependiendo del tipo de observador utilizado. Así, en [Clark 1987] se usa un observador excitado por una salida mientras las demás son reconstruidas y comparadas con las obtenidas del sistema, mientras en [Willsky 1976] se usan múltiples observadores excitados por distintas salidas. Otras técnicas de detección, por el contrario, se basan en métodos estocásticos considerando que el error $e(t)$ es ruido blanco y que varía sus propiedades estadísticas cuando se produce un error. La detección de estos cambios se puede llevar a cabo usando herramientas como los filtros de Kalman.

Cuando no existe especial interés en la reconstrucción de las variables de estado, otra posibilidad es el uso de observadores de salida [Tsui 1993].

3.3.1.3 Obtención de síntomas

Una vez obtenida la información necesaria: señales de error, residuos y todos los demás restos descritos anteriormente, se generan los síntomas utilizando métodos que dependen del tipo y características de dicha información.

En algunos casos, la extracción de datos se limita a la observación de alguna señal del proceso real, dado que se pueden extraer síntomas en función de ciertas características de estas señales. Por ejemplo, muchas señales medidas en los sistemas muestran oscilaciones bien de naturaleza estocástica o armónica. Cambios en estas señales (período, intensidad, etc) pueden estar relacionados con fallos en el proceso.

En estos últimos años ha proliferado el uso de diverso tipo de sensores de reducido coste. Por ejemplo, en algunas máquinas se usan sensores de posición, velocidad y aceleración para detectar fallos de desgaste o desequilibrado. En otros casos se usan sensores para obtener presiones, flujos, distancias, temperatura, luminancia, etc. Las restricciones aplicables a estas señales pueden ser diversas, desde

rangos de amplitudes ($\text{MIN} < |X(t)| < \text{MAX}$), periodicidad, autocorrelación a restricciones en su ancho de banda, densidad espectral, etc. En otras ocasiones, más que la señal en sí, interesa la derivada de ésta, es decir, su variación con el tiempo de forma que cambios bruscos en algunas señales pueden ser un indicio de fallo.

En cuanto a variables estocásticas, las señales medidas así como parámetros, variables de estado o residuos, se suelen caracterizar por su media y varianza para luego obtener síntomas a partir de los rangos posibles de estos valores o sus variaciones.

3.3.2 Generación heurística de síntomas

Mientras la generación analítica de síntomas se basa en medidas directas sobre el proceso, la generación heurística obtiene los síntomas a partir de información cualitativa proporcionada por algún operador humano. Expresiones como “algo debe ir mal ya que esta máquina hace ruido” pueden ser útiles pero son difíciles de cuantificar. Esta información puede ser proporcionada usando diferentes técnicas de modelado cualitativo, entre las cuales están la lógica borrosa y el uso de variables lingüísticas (pequeño, grande, ...). El operador obtiene esa información de su experiencia con el sistema, como por ejemplo, que una determinada secuencia de señales en un panel de control suele ser preludio de problemas.

3.4 Diagnosis

El proceso de diagnosis de fallos consiste en determinar las características del fallo (tipo de fallo, localización del fallo, tiempo en que se produjo, etc) a partir de los síntomas obtenidos por cualquiera de los procedimientos enunciados en el apartado anterior. Esta información acerca del fallo se utilizará para decidir el mecanismo de recuperación. En la mayoría de los sistemas, la elección del mecanismo de recuperación se basa solamente en el tipo de fallo. En estos casos se puede definir la diagnosis como una transformación del espacio de síntomas al espacio de fallos:

$$D: \mathcal{R}^m \rightarrow B^k$$

$$\bar{s} \rightarrow D(\bar{s}) = \bar{f} \in \{0,1\}^k$$

donde el vector **s** es un vector que representa los síntomas, mientras que el vector **f** es un vector binario de dimensión igual al número posible de fallos del sistema.

Se pueden distinguir tres tipos distintos de métodos de diagnosis de acuerdo con el conocimiento acerca de la relación fallo \leftrightarrow síntoma de que se dispone [Leonhardt 1997]:

- 1.- Métodos de clasificación.
- 2.- Métodos de inferencia.
- 3.- Métodos mixtos (combinación de los anteriores).

En el primer tipo de métodos, cuando no se dispone de información de causalidad acerca de los posibles fallos, se pueden usar mecanismos de clasificación tales como clasificadores geométricos [Dasarthy 1990], clasificadores estadísticos [Friedman 1997], redes neuronales [Bishop 1995]. En estos métodos se usan unos patrones de referencia a partir de los cuales se obtienen los bordes (métodos geométricos) o bien pesos mediante entrenamiento en otros casos (redes neuronales, clasificadores estadísticos, etc).

Si, por el contrario, se dispone de información de causalidad acerca de los fallos \leftrightarrow síntomas, se pueden usar estrategias de diagnosis basadas en sistemas de inferencia en los que se emplean motores de inferencia que trabajan sobre un conjunto de reglas de inferencia almacenados en una base de conocimiento y hechos obtenidos del funcionamiento del sistema. Las reglas de inferencia están basadas en reglas lingüísticas en lugar de patrones de referencia como en el caso anterior. Estas reglas son programadas en la mayor parte de los casos por un operador humano, experto en el proceso, aunque existen algunos sistemas que van aprendiendo dichas reglas con la experiencia [Mitchell 1997]. En este segundo grupo también se incluyen los métodos basados en reglas borrosas que permiten la decisión sobre conjuntos borrosos.

Por último, tratando de combinar las ventajas de estas dos metodologías, existe una serie de sistemas que incorporan métodos de diagnosis mixtos donde conviven partes de los dos anteriores [Isermann 1996]. Entre éstos destacan los sistemas *borroso-neuronal* (neuro-fuzzy).

3.4.1 Métodos de clasificación

En general, un sistema de decisión mediante métodos de clasificación D(s) usa una serie de vectores de referencia almacenados en una base de datos tal como se muestra en la figura 3.6

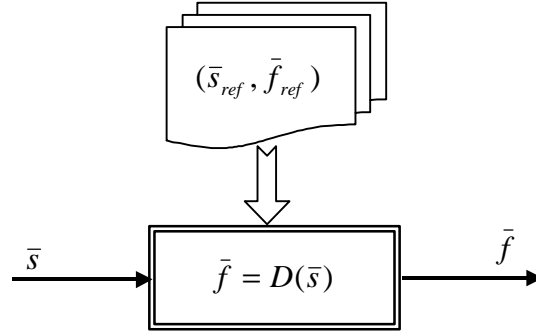


Figura 3.6.

Los métodos de clasificación más utilizados son los clasificadores geométricos, los estadísticos y las redes neuronales.

3.4.1.1 Clasificadores geométricos

Este grupo de clasificadores es uno de los más sencillos y quizás más usado en la práctica. Su funcionamiento se basa en dividir el espacio de los síntomas en zonas, donde cada una esta asociada a ninguno, uno o varios fallos. La división de las distintas zonas se realiza a partir de los vectores de referencia que son aquellos que caracterizan las regiones del espacio [Dasarthy 1990]. Según la clase de zonas empleadas y la forma de obtener éstas se definen distintos tipos de clasificadores geométricos. Uno de los más usados consiste en dividir el espacio de forma que un determinado vector síntoma \bar{S} pertenece a la zona definida por el vector de referencia más cercano \bar{S}_{ref} . En este tipo de clasificadores, el vector de referencia que caracteriza a una zona es el punto central de ésta. Es decir, suponiendo que hay p vectores de referencia (p zonas por lo tanto) y que la dimensión del espacio de síntomas es n , la zona Z asociada a un vector \bar{S} viene dada por :

$$d_i = dist(\bar{S}, \bar{S}_{ref,i}) = \sqrt{\sum_{j=1}^n (S_j - S_{ref,i,j})^2} \quad [Eqs. 3.12]$$

$$Z = \arg \min_i \{d_i\}$$

Dentro de una zona Z_{ref} , el fallo asociado es f_{ref} .

En este caso existirán tantas zonas como vectores de referencia. En el caso de que el número de vectores de referencia sea elevado, se suele acudir a métodos de aprendizaje muy extendidos en la actualidad [Mitchell 1997], donde el espacio se divide en tantas zonas como posibles salidas (fallos o combinaciones de éstos en el caso que aquí nos ocupa). Los vectores de referencia se utilizan para delimitar los bordes de estas zonas usando distintos mecanismos para minimizar la posibilidad de un falso diagnóstico.

Los clasificadores lineales utilizan funciones de discriminación lineal, mientras que los clasificadores cuadráticos usan una función de discriminación cuadrática tales como círculos, elipses, parábolas, etc.

3.4.1.2 Clasificadores estadísticos

Los clasificadores estadísticos consideran a los síntomas como variables aleatorias con funciones de distribución conocidas. Uno de los más difundidos son los clasificadores de Bayes [Friedman 1997] cuyo funcionamiento se basa en minimizar la probabilidad de error en la diagnosis.

Suponiendo que el proceso puede encontrarse en tres diferentes estados w (dos situaciones de fallo posibles (f_1, f_2) más una situación sin fallos (N)) y que para ello se dispone de un síntoma observable s obtenido mediante cualquiera de los métodos vistos anteriormente o por simple observación de los datos obtenidos de un sensor. Las probabilidades a priori de los fallos ($P(f_1), P(f_2), P(N)$) se suponen conocidas al igual que las distribuciones de probabilidad del síntoma condicionado cuando se produce un fallo ($P(s/f_1), P(s/f_2), P(s/N)$). Estas distribuciones se obtienen normalmente mediante aprendizaje o a partir de las características del sistema. Bajo estas condiciones, si se observa un valor x del síntoma s , ¿cuál de las posibles situaciones debemos diagnosticar para que la probabilidad de equivocarnos sea la mínima?. La respuesta a esta pregunta nos la da el clasificador de Bayes óptimo:

- Escoge aquella situación que tiene la más alta probabilidad dado el valor:

$$\begin{aligned} w_{opt} &= \arg \max_k P(w_k / x) \\ &= \arg \max_k \{P(x / w_k)P(w_k)\} \end{aligned} \quad [Eqs.3.13]$$

La importancia del teorema de Bayes radica en el hecho de expresar la probabilidad a posteriori en términos de cantidades que generalmente son mucho más fáciles de calcular.

En un caso general \mathbf{x} es un vector cuya dimensión coincidirá con la del espacio de síntomas. Para simplificar, en muchas situaciones se suponen distribuciones de probabilidad de tipo gaussiano. En particular, existen dos métodos muy utilizados:

Método 1: $P(\bar{\mathbf{x}}/w_k)$ gaussiana, media = $\bar{\mathbf{m}}_k$, covarianza = $S^{-2}I$

Se supone que las componentes del vector de síntomas son independientes entre sí.

$$P(\bar{\mathbf{x}} / w_i) = Ke^{\left(\frac{-1}{2S^2} \sum_j (x_j - m_{ij})^2 \right)} \quad [Eq. 3.14]$$

El clasificador de Bayes seleccionará:

$$\begin{aligned} \arg \max_i \{P(\mathbf{x} / w_i)P(w_i)\} &= \arg \max_i \left\{ Ke^{\left(\frac{-1}{2S^2} \sum_j (x_j - m_{ij})^2 \right)} P(w_i) \right\} \\ &= \arg \max_i \left\{ \frac{-1}{2S^2} \sum_j (x_j - m_{ij})^2 + \log P(w_i) \right\} \\ &= \arg \min_i \left\{ \sum_j (x_j - m_{ij})^2 - 2S^2 \log P(w_i) \right\} \\ &= \arg \min_i \left\{ \|\bar{\mathbf{x}} - \bar{\mathbf{m}}_i\|^2 - 2S^2 \log P(w_i) \right\} \end{aligned}$$

De donde se puede observar que, si las probabilidades a priori de las distintas situaciones son iguales ($P(w_i) = P(w_j) \forall i,j$), se reduce a seleccionar el vecino más cercano.

Método 2 : $P(\bar{x}/w_k)$ gaussiana, media = \bar{m}_k , covarianza = Γ

En este caso la matriz de covarianza es arbitraria (G). Haciendo un desarrollo similar caso anterior, se llega a la conclusión de que se pueden aplicar las mismas reglas pero con distinta métrica:

$$dist(\bar{x}, \bar{m}_i) = (\bar{x} - \bar{m}_i)^T \Gamma^{-1} (\bar{x} - \bar{m}_i) \quad [Eq. 3.15]$$

Redes de Bayes

Las redes de Bayes [Hamscher 1991] son grafos dirigidos acíclicos donde los nodos son variables aleatorias lógicas (figura 3.7). Los arcos entre dos nodos definen las dependencias (probabilidades condicionales) entre las variables aleatorias. La red viene definida por los pesos de los arcos (probabilidades condicionales) y las probabilidades de los nodos de entrada.

Distintos modelos de redes que integran información simbólica y probabilística han sido objeto de intensivo estudio en el área de Inteligencia Artificial. En particular, un tipo de redes conocidas como redes de probabilidad de Bayes (*Bayesian Belief Network*), redes de inferencia de Bayes (*Bayesian Inference Network*), redes de probabilidades (*Belief Network*) [Cooper 1990] o simplemente, redes de Bayes (*Bayesian Networks*).

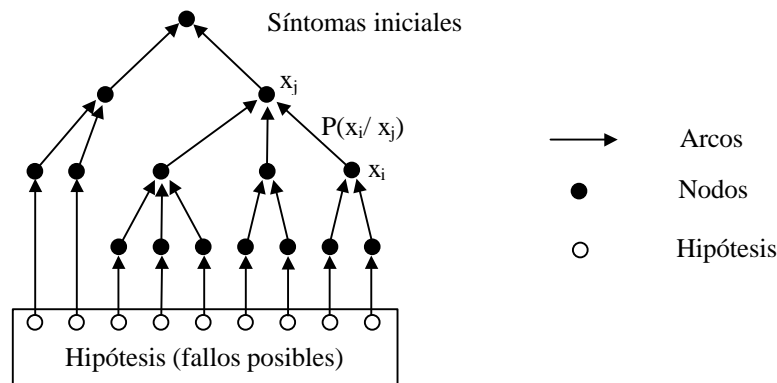


Figura 3.7. Redes de Bayes para sistemas de diagnosis.

En las redes de Bayes usadas en diagnosis, los nodos de entrada representan los posibles fallos (hipótesis), mientras que los nodos de salida se asocian con los síntomas iniciales. Los nodos intermedios se relacionan con posibles tests que se pueden hacer al sistema y cuyo resultado puede dar positivo o negativo.

Las relaciones causa-efecto vienen dadas como probabilidades condicionales que, en algunos casos, se aprenden a partir de los vectores de referencia y en otros se obtienen a partir del conocimiento sobre el sistema. También se suponen conocidas a priori las probabilidades de fallo.

Las redes de Bayes para diagnosis se pueden definir formalmente como:

$$\Phi \equiv (X, C, H, P, T)$$

donde:

X: conjunto de variables aleatorias (nodos de la red).

$$X = \{x_1, x_2, x_3, \dots, x_n\}$$

C: conjunto de conexiones (arcos de la red).

$$C \subset \{(H \cup X) \times (H \cup X)\}$$

H: conjunto de hipótesis o fallos (nodos de entrada).

$$H = \{h_1, h_2, h_3, \dots, h_n\}$$

P: conjunto de pesos de los arcos.

$$P = \{P(x_j / x_i); \forall x_j, x_i \in X / \{x_j, x_i\} \in C\}$$

T: conjunto de tests.

$$T = \{T_i, T_i(x_i) = \{\text{Cierto}, \text{Falso}\}\}$$

El planteamiento de la diagnosis, en la mayor parte de las aplicaciones tratadas con este tipo de redes, consiste en buscar la secuencia mínima de tests necesarios para la determinación del (o de los) fallo (fallos), en lugar de encontrar el conjunto de fallos más probable dados una serie de síntomas.

Se pueden aplicar algoritmos como el G^* [Agre 1997] en determinados tipos de Redes de Bayes con objeto de obtener los tests necesarios para determinar el fallo que se ha producido.

En la mayoría de las redes de Bayes [Agre 1997] aplicadas en diagnosis se considera que las relaciones causales entre los fallos y sus manifestaciones son deterministas. Esto no es cierto en todos los casos, como se verá más adelante. La principal aplicación de estos sistemas es determinar la secuencia óptima de test a realizar para diagnosticar los fallos. Este tipo de redes se ha aplicado también en el campo de la medicina [Friedman 1997]. Un estudio más detallado puede verse en [Agre 1992].

3.4.1.3 Redes neuronales

Entre las muchas aplicaciones de las redes neuronales [Rumelhart 1994] [Hunt 1992] se han usado también en sistemas de diagnosis [Bishop 1995]. Al igual que se puede entrenar una red neuronal para reconocer fonemas en una señal de voz o para reconocer ciertas formas en una imagen [Hanazawa 1989], se puede entrenar para reconocer determinados fallos en un sistema. Bastará con entrenarla con los ejemplos de referencia adecuados que representen todas las posibilidades de fallo.

Como se muestra en la figura 3.8, el funcionamiento de la red neuronal [Hunt 1992] está basado en unidades o neuronas que tratan de simular el funcionamiento del sistema nervioso en un ser humano. Estas neuronas toman como entrada un vector de valores reales, calculan una combinación lineal de sus entradas y a dicho resultado le aplica una transformación no lineal. Las transformaciones más utilizadas son:

♦ Escalón:

$$y = F(x) = \begin{cases} 1 \leftrightarrow x > a_0 \\ -1 \leftrightarrow x < a_0 \end{cases} \quad [Eq. 3.16]$$

♦ Sigmoide. La transformación se define mediante la ecuación 3.17.

$$y = F(x) = \frac{1}{1 + e^{-x}} \quad [Eq. 3.17]$$

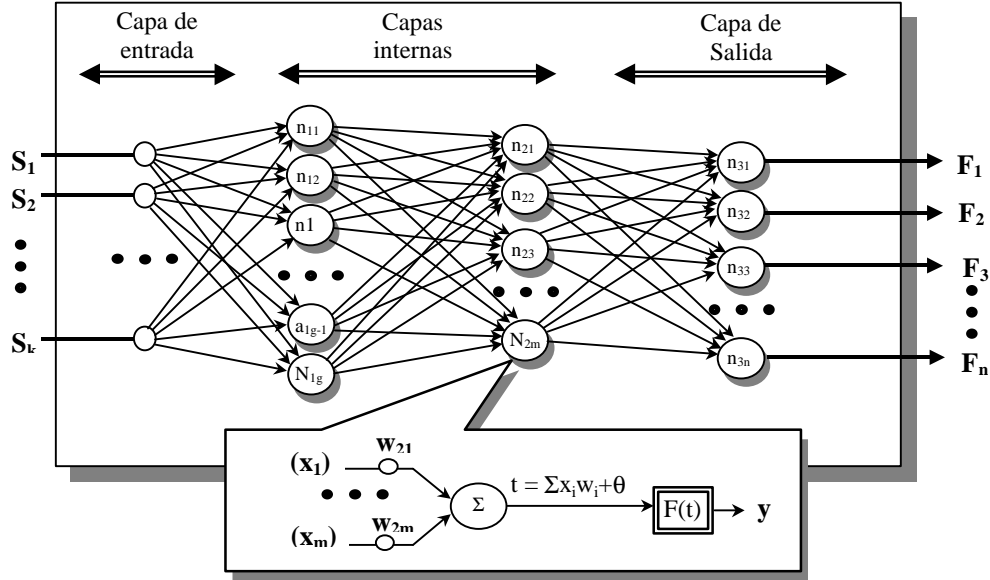


Figura 3.8. Esquema de una red neuronal con tres capas de neuronas.

Al igual que en otras aplicaciones de este tipo de redes, en la creación de un sistema de diagnosis con redes neuronales existen dos fases bien diferenciadas. La primera es la de aprendizaje en la cual la red neuronal usa los vectores de referencia (S_{ref} , F_{ref}) para ajustar los pesos de las neuronas. Uno de los algoritmos más usados para dicha tarea, a pesar de su lentitud es el de *backpropagation* [Chauvin 1995]. El objetivo es ajustar los pesos de las neuronas para que, con los síntomas S_{ref} , se obtengan las situaciones F_{ref} . Una vez acabada dicha fase, la información proporcionada por los vectores de referencia se encuentra de alguna forma almacenada en los pesos de las neuronas. Durante la fase de ejecución, los síntomas obtenidos serán clasificados por la red en alguno de los estados F.

El aprendizaje con redes neuronales se adapta bien a problemas en los que los datos de entrada son obtenidos de sensores con ciertos niveles de ruido tales como micrófonos, cámaras, etc. Una ventaja de este tipo de métodos es el aprendizaje de forma automática a partir de los datos de referencia. No obstante, una vez entrenada, la información se almacena en los pesos de las neuronas siendo muy difícil su interpretación por un operador humano. Si, posteriormente se desea modificar su comportamiento, no es posible hacerlo de forma directa manipulando los pesos dado que se desconoce el efecto que esto va a tener. El entrenamiento es el único medio de variar su forma de funcionamiento.

3.4.2 Sistemas basados en conocimiento

A diferencia del caso anterior, en lugar de usar los vectores de referencia, un sistema de decisión basado en conocimiento, $D(s)$, usa información de causalidad expresada normalmente a través un conjunto de reglas que permite relacionar los síntomas con los fallos usando métodos deductivos [Williams 1997]. En la figura 3.9 se muestra el esquema de funcionamiento general.

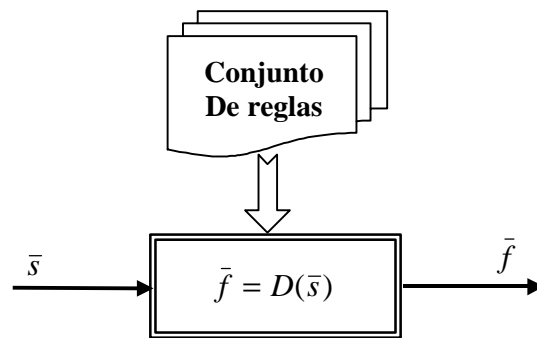


Figura 3.9. Esquema de funcionamiento de un sistema basado en conocimiento

Entre otros ejemplos de sistemas basados en conocimiento para diagnosis de fallos se puede mencionar *FALCON* desarrollado por *Foxboro Corporation*, *du Pont* y la *Universidad de Delaware* para diagnosis de fallos en un reactor, o *YES/MVS* para supervisar el funcionamiento de grandes estaciones de trabajo IBM [Milliken 1986].

3.4.2.1 Sistemas expertos

El esquema del funcionamiento de estos sistemas puede verse en la figura 3.10. La información está normalmente representada como un conjunto de reglas que reflejan la experiencia y conocimiento humano sobre la causalidad de los fallos del sistema en función de los síntomas. Cada vez que se modifica la base de hechos, bien por la llegada de nuevos síntomas, o por el disparo de otras reglas que añaden información a la base de hechos, el motor de inferencias comprueba si esta información hace que se verifiquen los antecedentes de alguna regla y, en caso afirmativo, se tomarán las acciones oportunas de acuerdo con los consecuentes de la regla.

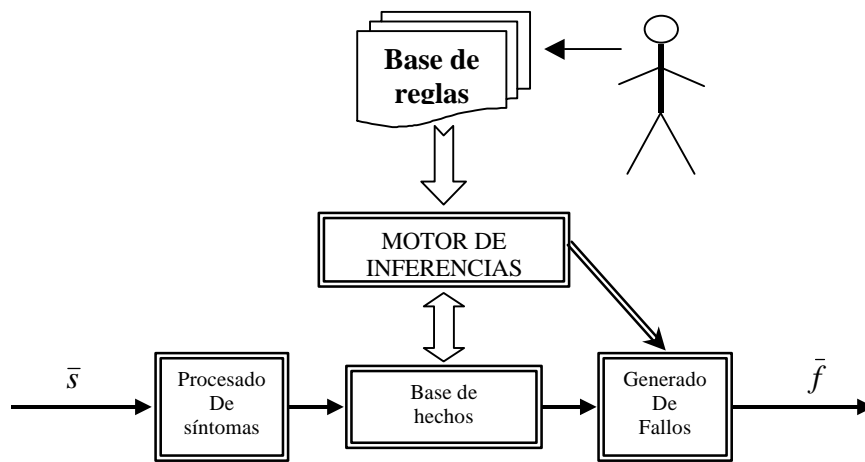


Figura 3.10 Funcionamiento de un sistema experto aplicado a diagnosis.

Esta solución es especialmente atractiva cuando la información de que se dispone está representada mediante asociaciones empíricas adquiridas de la experiencia en el manejo del proceso. Otra ventaja de estos sistemas es que al mismo tiempo que se detecta un fallo, se puede presentar el razonamiento que llevó a tal deducción siguiendo las reglas aplicadas. La base de reglas esta formada por un conjunto de reglas del tipo:

SI <antecedentes> ENTONCES <consecuentes>

Donde los antecedentes están formados por un conjunto de predicados o condiciones a_i unidos por relaciones lógicas (básicamente **Y**, **O** y **NOT**). Para el caso de diagnosis, los antecedentes a_i son las condiciones sobre los síntomas y los consecuentes c_i pueden activar (poner a cierto) un fallo o bien aportar información que será añadida a la base de hechos. También puede haber reglas cuyos antecedentes y/o consecuentes no estén referidos directamente a los síntomas y/o fallos, sino a variables intermedias. Este conjunto de reglas permite relacionar los síntomas con los fallos en una forma similar a la expresada por un humano, lo cual constituye uno de sus mayores atractivos con respecto a los métodos de clasificación del apartado anterior.

Cada vez que se produce un síntoma o éste cambia significativamente de valor hay que comprobar si esta variación implica que se active (cambie el estado de los consecuentes) alguna regla, que a su vez puede modificar el valor de alguna variable intermedia y activar otras reglas. Este proceso de deducción lo realiza el motor de inferencias [Escalada 1994]. Entre los problemas que presentan estos sistemas se encuentra la cantidad de procesamiento necesario cuando la base de reglas obtiene un tamaño considerable, la detección de bucles y las limitaciones de la lógica binaria. Se han desarrollado algoritmos que evitan tener que verificar en cada ciclo de deducción toda la base de reglas tales como el RETE [Forgy 1982] o las desarrolladas en *Livingstone* [Nayak 1996].

3.4.2.2 Sistemas basados en conocimiento con razonamiento aproximado

En lugar de las reglas basadas en lógica binaria, utilizadas en los sistemas expertos, se emplean reglas usando lógica borrosa [Ulieru 1996]. De esta forma, se pueden tratar datos imprecisos basados en términos lingüísticos análogos a los utilizados en las expresiones del conocimiento entre personas, como “mucho”, “poco”, “alto”, “bajo”, etc.

Un cierto síntoma s_i se evalúa de acuerdo a la función de pertenencia asociada a ese síntoma ($\mu(s_i)$) para distintos conjuntos borrosos. Normalmente los valores de las funciones de pertenencia están normalizadas al rango [0..1]. Como ejemplo puede verse en la figura 3.11 la función de pertenencia asociada a un síntoma (s_i) con tres conjuntos borrosos (pequeño, mediano, grande). Las funciones de pertenencia pueden ser de distinta forma (gaussianas, poligonales, etc). En la figura 3.11 se presenta un ejemplo de las más utilizadas. De acuerdo con dicha figura, si s_i tiene el valor V , el grado de pertenencia al conjunto pequeño es P , el grado de pertenencia al conjunto mediano es M y al grande es cero.

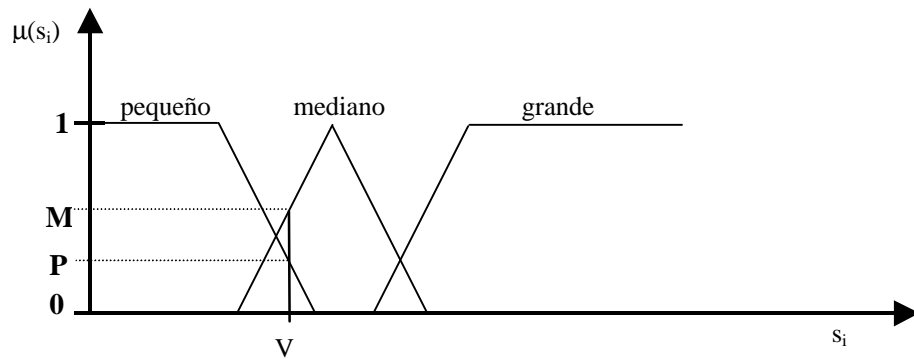


Figura 3.11. Función de pertenencia para tres conjuntos borrosos.

Una vez expresados los síntomas de esta forma, las reglas de inferencia borrosa son similares a las de los sistemas de producción pero utilizando operadores borrosos. Los operadores borrosos permiten la enunciación de relaciones lógicas entre los síntomas. Si bien lingüísticamente los operadores empleados son los mismos, existen diferentes formas para realizar el proceso de inferencia. De éstas, las más habituales son:

1- Máximo – mínimo:

$$\begin{aligned} Y &\Rightarrow \mu(s_1) \text{ Y } \mu(s_2) = \text{mínimo } \{\mu(s_1), \mu(s_2)\} \\ O &\Rightarrow \mu(s_1) \text{ O } \mu(s_2) = \text{máximo } \{\mu(s_1), \mu(s_2)\} \\ NO &\Rightarrow NO \mu(s_1) = 1 - \mu(s_1) \end{aligned}$$

2- Producto - suma:

$$\begin{aligned} y &\Rightarrow \mu(s_1) \text{ Y } \mu(s_2) = \mu(s_1) \times \mu(s_2) \\ O &\Rightarrow \mu(s_1) \text{ O } \mu(s_2) = 1 - (\mu(s_1) \times \mu(s_2)) \\ NO &\Rightarrow NO \mu(s_1) = 1 - \mu(s_1) \end{aligned}$$

En la figura 3.12 se representa un esquema general de cómo se aplican las reglas a los valores de dos síntomas (S_1 y S_2) para la determinación de la existencia del fallo F_1 . El primer paso es un proceso de normalización del valor de los síntomas al intervalo $[0..1]$. A continuación, se evalúa el grado de pertenencia de ese valor a cada uno de los conjuntos borrosos que forman el predicado de la regla. A estos valores se les aplican los operadores de las reglas para obtener el grado de pertenencia de los consecuentes de la regla en función del grado de los antecedentes.

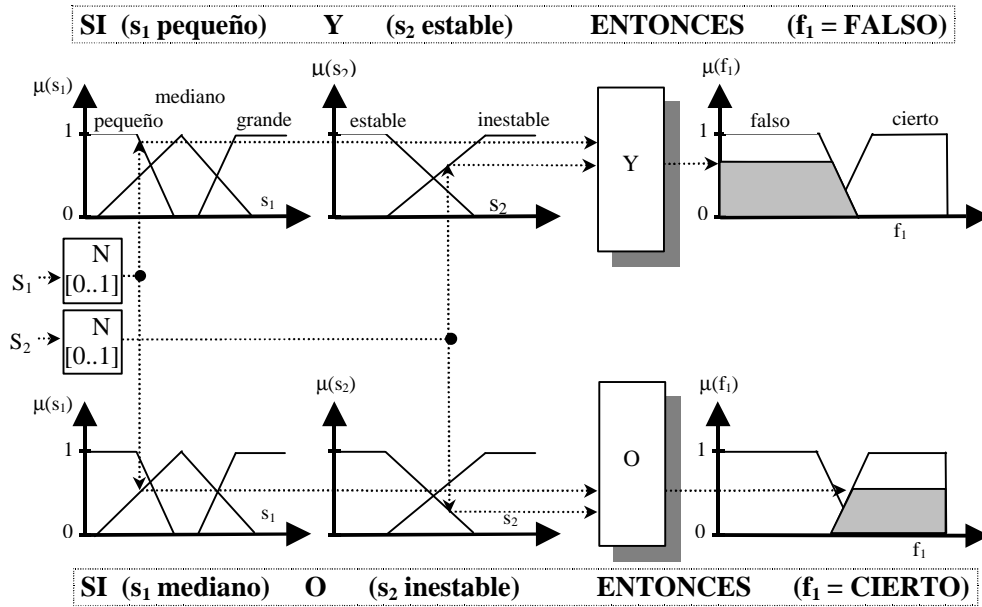


Figura 3.12. Esquema de un sistema de diagnosis usando técnicas borrosas.

Por último, después de la aplicación de las reglas, es necesario un proceso de reconversión de los valores borrosos para obtener un valor de salida de las funciones de pertenencia resultantes. Diversos autores [Driankov 1993] utilizan distintos métodos siendo los más utilizados (Fig 3.13):

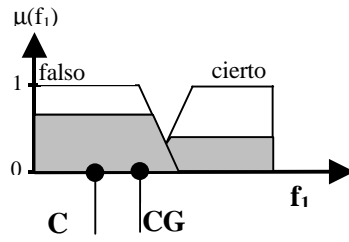


Figura 3.13. Técnicas de reconversión de valores borrosos

- ◆ Centro de gravedad (CG).

$$CG = \frac{\int f \times m(f) df}{\int m(f) \times df} \quad [Eq. 3.18]$$

- ◆ Centro del máximo (CM)

$$CM = \frac{\int f \times m_{falso}(f) df}{\int m_{falso}(f) \times df} \quad [Eq. 3.19]$$

Las salidas del sistema de diagnosis forman un vector de valores binarios en la mayor parte de las plataformas de tratamientos de errores, si bien existen otros muchos métodos que se utilizan sobre todo en sistemas de control borroso de procesos. Los consecuentes están, pues, formados por dos conjuntos (fallo/no fallo) y basta con elegir uno de estos dos basándose en cualquiera de las técnicas descritas anteriormente. Esto es, aquel con el máximo en la función de pertenencia o con área máxima.

Otra posibilidad es usar el método de Takagz-Sugeno [Driankov 1993] donde los consecuentes de las reglas toman valores discretos en lugar de conjuntos borrosos. Suponiendo las reglas de la siguiente forma:

R_1 : SI <antecedentes lógica borrosa> ENTONCES $F_1 = f_1$

R_2 : SI <antecedentes lógica borrosa > ENTONCES $F_1 = f_2$

...
 R_n : SI <antecedentes lógica borrosa > ENTONCES $F_n = f_n$

Al igual que antes, de los antecedentes borrosos se obtendrá un grado de pertenencia (μ_1 para la R1 μ_2 para la regla R2 ... μ_n para la regla RN) y el resultado final vendrá dado por:

$$F_1 = \frac{\sum_{i=1}^n \mu_i f_i}{\sum_{i=1}^n \mu_i} \quad [Eq. 3.20]$$

El principal inconveniente de los sistemas de lógica borrosa es el proceso de sintonización de las reglas. Dicha tarea consiste, en la mayor parte de los casos, en la observación de las relaciones causa/efecto (síntoma/fallo) durante un largo período de tiempo. Además estas reglas deben ser revisadas para poder introducir nuevas condiciones de fallos. Entre las ventajas cabe destacar la facilidad para modelar los fallos de manera explícita, sobre todo, si se comparan con las redes neuronales donde recordemos que la información esta almacenada en los pesos a las entradas de las neuronas, que son difícilmente interpretables por el operador humano.

3.4.3 Sistemas mixtos (redes neuronales - lógica borrosa)

Comparando las ventajas e inconvenientes de los sistemas de lógica borrosa con las redes neuronales podemos ver que éstos se complementan en algunos puntos. Con los sistemas mixtos se pretende obtener la facilidad para modelar los fallos de manera explícita de los conjuntos borrosos, junto con el aprendizaje automático y adaptación de las redes neuronales. Esta simbiosis se está abriendo paso en un gran número de aplicaciones [Altrock 1995] [Ayoubi 1997] entre las que se encuentran la diagnosis de errores. Existen varias aproximaciones que podemos subdividir según el nivel de participación de las dos teorías y la arquitectura predominante.

3.4.3.1 Arquitectura de redes neuronales en la que se integra la lógica borrosa

En este grupo, la arquitectura está basada en neuronas al igual que en las redes neuronales si bien se añaden modificaciones basadas en lógica borrosa. El objetivo es, por tanto, mejorar las redes neuronales añadiendo lógica borrosa a distintos niveles [Kiszka 1990] [Lin 1994].

Al nivel de neurona, algunos autores [Kiszka 1990] introducen cambios en las neuronas para que éstas puedan tratar información borrosa. La *neurona basada en lógica borrosa* (fuzzy-neurona) se diseña de forma similar a una neurona normal pero ahora se procesa información borrosa.

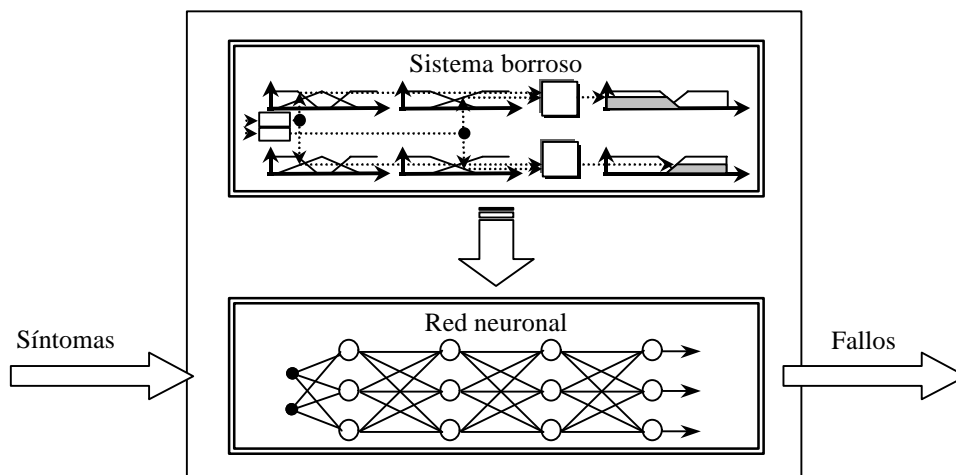


Figura 3.14. Arquitectura de redes neuronales en la que se integra lógica borrosa.

A un nivel diferente otros autores [Lin 1994] utilizan una arquitectura similar a la que se muestra en la figura 3.14 donde el sistema borroso se usa para mejorar la adaptación y aprendizaje de la red neuronal. La lógica borrosa se usa para establecer los pesos iniciales de forma que el tiempo de adaptación sea menor o bien para mejorar el sistema de aprendizaje como en [Lin 1994] donde se presenta una extensión del popular algoritmo de “back-propagation” empleado en el entrenamiento de redes neuronales.

3.4.3.2 Arquitectura de lógica borrosa en la que se integran redes neuronales

Al contrario del caso anterior, en este grupo, la arquitectura está basada en reglas similares a las empleadas en la lógica borrosa, si bien, se añaden modificaciones basadas en redes neuronales. Existen distintos modos de incluir las redes neuronales en un sistema de lógica borrosa. Cabe destacar aquellos en que ambos sistemas se conectan en serie o paralelo (figuras 3.15a y 3.15b) y los sistemas de redes neuronales con topología equivalente a un sistema de reglas borroso.

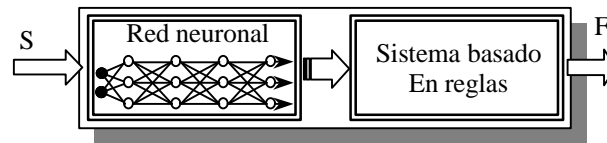


Figura 3.15a. configuración en serie

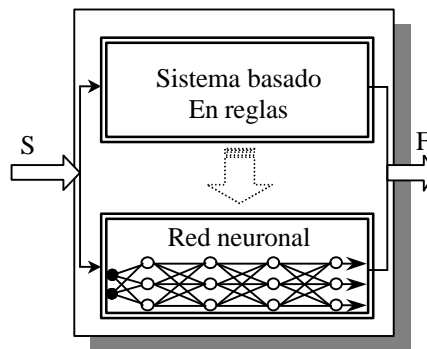


Figura 3.15b. Configuración en paralelo

La configuración en serie fig 3.15a se suele utilizar para la eliminación de ruido, reducir los datos de entrada, etc. La configuración en paralelo, (fig 3.15b) puede usarse para adaptar los parámetros del sistema borroso, o bien para calcular los

resultados de ambos sistemas en paralelo y utilizar los obtenidos por la red neuronal para mejorar los del sistema borroso. En los primeros, el objetivo es extender los sistemas de reglas añadiendo un componente más que provea la capacidad de aprender y adaptar parámetros fig 3.15a. Para ello es necesario la transferencia de información entre ambos componentes y que la estructura de las reglas se corresponda con la de la red.

Por último, los sistemas más integrados en este tipo de arquitecturas son los modelos de redes neuronales con topología similar a la estructura de reglas de un sistema borroso. El sistema denominado SARAH (*System for Adaptive Rule Acquisition with Hebbian Learning*) [Ayoubi 1996] es un ejemplo de estos sistemas cuyas líneas principales de funcionamiento se presentan en la figura 3.16.

Al igual que cualquier regla borrosa, el sistema tiene tres partes que se corresponden con tres capas de una red neuronal.

- ✓ La parte de los antecedentes realiza la conversión de los síntomas a valores borrosos. Es decir, obtiene el grado de pertenencia de las entradas a los distintos conjuntos borrosos. Para llevar a cabo dicha tarea se usa una neurona cuya función de activación es una función gaussiana tal y como se muestra en la figura 3.16. Las salidas de esta capa son, pues, los grados de pertenencia de las entradas a los distintos conjuntos borrosos ($\mu_m(S_k)$).
- ✓ La parte del operador consiste en una serie de perceptrones que realizan las operaciones lógicas entre los antecedentes. La operación lógica a realizar (Y/O) se establece seleccionando del umbral (α_0) de la función sigmoideal asociada a cada perceptron.
- ✓ Los consecuentes se representan mediante “singletons”.

Al igual que los anteriores, este método sigue presentando el problema de completitud pues es posible que exista una combinación de síntomas no contemplada por ninguna de las reglas del sistema. Esto sigue siendo un problema a resolver por el experto que programa las reglas.

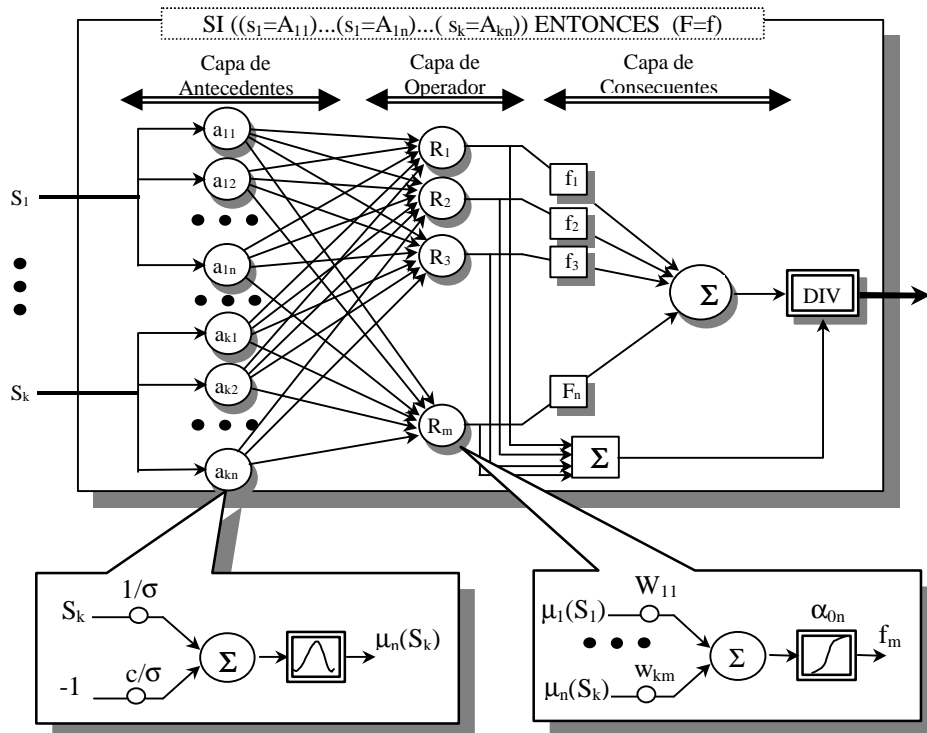


Figura 3.16. Red borroso-neuronal (neuro-fuzzy) con k síntomas, correspondiente a m reglas y n conjuntos borrosos para detectar un fallo F

4 Detección, diagnosis y recuperación de fallos en robótica móvil

4.1 Introducción

Una vez visto, en líneas generales, las formas más comunes de detección y diagnosis de fallos, el siguiente paso es estudiar como se han abordado estos aspectos en el campo de la robótica móvil. En el capítulo anterior se mencionó que distintos autores tratan el problema de forma muy dispar debido, en gran medida, a la diversidad de los tipos de fallos y su forma de tratamiento. Esta característica se hace más evidente en el ámbito de la robótica móvil, dada la complejidad de los sistemas de navegación, la multitud de sensores presentes y la imposibilidad de modelar completamente el entorno.

Los fallos comúnmente tratados son los debidos al sistema sensorial ya sea sólo a nivel hardware [Stergios 1998b] [Stergios 1998] o incluyendo también fallos software [Murphy 1996]. Mientras algunas arquitecturas [Ferrell 1994][Ferrell 1993] sólo consideran fallos en sensores y actuadores, en otras [Noreils 1990] [Sankaran 1977][Stuck 1992] [Stuck 1995] también se tienen en cuenta los fallos a nivel navegación. El tratamiento de fallos en este último grupo está siempre condicionado por la arquitectura de navegación usada, dando lugar a diferentes sistemas de supervisión, detección y recuperación ante fallos.

No sólo existe una gran variedad en la forma de tratamiento de fallos sino que ésta se extiende a su definición. El término fallo presenta distintos matices según el autor que lo emplea, si bien está mejor delimitado cuando se refiere al nivel sensorial. La diversidad, como se verá más adelante, viene de la forma en que se calculan las expectativas y los márgenes de error permitidos. A nivel de sensores se considera que existe un fallo cuando la lectura producida por un sensor no concuerda con las *expectativas* según las especificaciones del sensor. En navegación, la definición de fallo es más difusa y se utilizan distintos términos (equivocación, error, fallo, ...). En

[Stuck 1992] [Stuck 1995] se utiliza el término equivocación (*mistake*) para referirse a eventos en la percepción, conocimiento del navegador o eventos del sistema motriz que desvían al robot de la ruta original. Otros autores [Noreils 1990][Sankaran 1977] consideran que se produce un error cuando una tarea o acción no se finaliza correctamente. Esta última definición puede ser, en algunos casos, demasiado general. Supóngase, por ejemplo, que la tarea en cuestión es atravesar un pasillo. El primer problema surge al tratar de definir la finalización correcta de la tarea. Si por ello se entiende que el robot debe estar en el otro extremo del pasillo en un determinado tiempo, puede surgir la siguiente situación: Supongamos que hay un obstáculo insalvable en el pasillo, el robot intentará bordearlo pero no podrá terminar la tarea encomendada pero, ¿existe en realidad algún fallo?. Es evidente que el sistema funciona de acuerdo a lo planeado.

La dificultad de la tarea de detección de errores queda patente en las siguientes situaciones que se pueden producir en el ejemplo anterior:

- ✓ El robot puede creer que ha llegado al final del pasillo, cuando en realidad se ha desviado y se encuentra al final de otro pasillo.
- ✓ En determinadas circunstancias es difícil estimar el tiempo que el robot empleará en ir de un extremo al otro de un pasillo. Si en el pasillo hay personas, tiene que evitarlos y, por tanto, le llevará mucho más tiempo llegar a cumplir el objetivo.

Este último punto pone de manifiesto que la selección de valores permisibles de algunas variables empleadas en la detección de fallos conlleva un compromiso entre detección de falsas alarmas o falsos fallos y la omisión o retraso en la detección de otros. Ello se debe a la difusa separación entre los valores de las variables para una situación de fallo y situación nominal o de buen funcionamiento. Las tolerancias pueden medirse de muy distintas formas pero en la mayor parte de los casos se aplica una simple verificación de rangos de variables.

Algunos de los métodos analizados en el capítulo anterior se aplican a agentes móviles, si bien los métodos de navegación presentan aspectos particulares que lo caracterizan con respecto a otros sistemas:

- ✓ El comportamiento de estos sistemas es no lineal y tiene parámetros que varían con el tiempo.
- ✓ Debido a la interacción con el entorno, se encuentran continuamente sometidos a intervenciones inesperadas lo que significa que son sistemas discontinuos e impredecibles.

- ✓ Como consecuencia de lo expuesto anteriormente, el sistema de navegación es demasiado complejo para poder encontrar un modelo matemático con el cual estimar su comportamiento en detalle, utilizándose en su lugar *modelos lógicos*.
- ✓ Debido a la complejidad de los sistemas, existe la necesidad de analizarlo y controlarlo a distintos niveles jerárquicos.
- ✓ La complejidad del sistema y la división en capas de la mayor parte de las arquitecturas de navegación permite aislar el tratamiento de errores en distintas capas. Un ejemplo claro de ello puede encontrarse en el robot Hannibal [Ferrell 1994] donde se definen unos sensores virtuales, de forma que algunos problemas en los sensores físicos son transparentes a las capas de control.
- ✓ El tratamiento de errores es una función secundaria que debe estar integrada en el sistema.
- ✓ A veces es muy complicado discernir entre situaciones nominales y de fallo utilizando solamente información numérica debido al amplio margen de valores que pueden tomar los parámetros en una situación nominal y la difícil diferenciación con respecto a una situación de fallo.
- ✓ Uno de los puntos clave es el del tratamiento de la incertidumbre en distintos niveles: modelado, actuación y sensorial. La resolución de problemas con modelado perfecto es, a todas luces, mucho más sencillo que cuando éste no es perfecto debido a la incertidumbre.

Las soluciones que abordan la detección y recuperación de errores en navegación de robots móviles están extremadamente ligadas a la arquitectura del sistema de navegación. Por ejemplo, la recuperación de errores en navegación es raramente usada en sistemas reactivos [Brooks 1986] dado que estos sistemas básicamente reaccionan a eventos. En el contexto de navegación en robots móviles, existen varias arquitecturas que tratan con fallos de diferentes formas [Noreils 1990] [Stuck 1995] y a distintos niveles. En el nivel más bajo, se tratan los errores de los sensores [Stergios 1998b] [Ferrell 1994] [Ferrell 1993]. Los procesos de monitorización y recuperación ante errores son responsables de que el robot ejecute correctamente las tareas encomendadas, detectando cuando esto no es así y ejecutando las acciones de recuperación oportunas.

En algunas arquitecturas de tipo deliverativo [Stuck 1992], la monitorización se limita a verificar la correcta finalización de cada uno de los pasos trazados en la planificación.

La arquitectura de monitorización y recuperación de errores presentada en esta tesis es independiente de la de navegación. De hecho, la arquitectura de navegación del robot combina comportamientos reactivos y planificados. Como se verá más en detalle en el próximo capítulo, la parte reactiva se encarga de la evitación de obstáculos usando el método CVM [Simmons 1996], al mismo tiempo que tratará de seguir en lo posible las consignas determinada por el nivel de planificación. En este último se incluyen las tareas de planificación de trayectorias y estimación de la posición.

Incluso en sistemas reactivos, es posible extraer algunas observaciones acerca del comportamiento del robot y cuantificar la forma en que éste se ajusta al comportamiento esperado aunque dichas observaciones, debido a la complejidad del sistema, no son deterministas. Estos síntomas pueden considerarse, por tanto, independientes de la arquitectura si bien distintas arquitecturas tendrán diferentes síntomas con diversos grados de fiabilidad.

La organización de este capítulo es similar a la del anterior dado que se trata del mismo tema aplicado al campo de la robótica móvil. En el siguiente apartado se describirán los distintos niveles de tratamiento de fallos en robots móviles dividiéndolos en dos grandes grupos: sensorial, que es descrito en el apartado tres y de navegación que será tratado en el apartado cuatro. Dentro de cada apartado se seguirá la misma estructura utilizada en la descripción de tratamientos de fallos en general del capítulo anterior. Esto es, caracterizando en primer lugar los tipos de fallos a detectar y luego describiendo las distintas fases del proceso: extracción de síntomas, detección, diagnosis y recuperación, incluyendo al final comentarios sobre los puntos fuertes de cada método.

4.2 Niveles de tratamiento de fallos

La detección y diagnosis de fallos en agentes móviles se pueden realizar a distintos niveles (hardware, control y supervisión). El tratamiento de errores cobra especial relevancia si se tienen en cuenta las aplicaciones de los robots móviles autónomos. Basta para ello recordar que, la mayor parte de ellos se construyen con la finalidad de operar en entornos que pueden ser peligrosos para un operador humano tales como centrales nucleares, exploraciones planetarias [Angle 1990] [Bares 1990], las profundidades de los océanos [Payton 1992] o cráteres de volcanes. Estas tareas requieren que el robot debe estar operativo sin la intervención del operador humano para efectuar ninguna reparación. En estas aplicaciones, es importante que el robot sea

capaz de recuperarse ante cualquier tipo de error, incluyendo los posibles fallos a nivel hardware.

Los niveles a los que se pueden producir los fallos dependen del tipo de arquitectura dado que distintas arquitecturas poseen diferentes niveles, sin embargo, como ya se ha comentado, existen algunos comunes a todas ellas.

- ♦ **Nivel hardware.** Incluye todos aquellos fallos bien en los sensores o bien en los actuadores.
- ♦ **Nivel de servocontrol.** Se refiere a los procesos que se encargan de obtener las lecturas del sistema sensorial y enviar las señales de control. Es decir, las capas que proveen a las que están por encima con una serie de sensores virtuales.
- ♦ **Nivel reactivo.** El módulo de evitación de obstáculos en algunas arquitecturas [Simmons 1997b] [Burgard 1998] o bien los comportamientos [Parker 1998] [Brooks 1986] en otras.
- ♦ **Nivel de planificación.** Por último se incluyen aquellos fallos que se pueden producir en la capa deliberativa (planificación, navegación, posicionamiento).

La recuperación de los fallos también se puede llevar a cabo en distintos niveles. No obstante, los fallos producidos en un determinado nivel sólo podrán ser tratados en un nivel superior o igual. Así un fallo producido en el hardware puede tratarse en cualquiera de ellos. Los niveles de tratamiento de fallo son pues:

- ♦ **Nivel hardware.** Para aumentar la robustez del sistema se suele duplicar algunos dispositivos de forma que si uno dejara de funcionar se activaría automáticamente el otro. Varios sistemas comerciales [Gray 1990] y algunas aplicaciones en robótica [Kabuka 1990] utilizan este sistema para aumentar su fiabilidad. De esta forma se podrían tratar fallos hardware de forma transparente al software. El principal problema de la redundancia de dispositivos es el precio de los componentes hardware, sobre todo algunos de los sensores de alta precisión usados en estos robots. Otro problema es el incremento en el peso, lo cual puede ser crítico en algunas aplicaciones, tales como en misiones interplanetarias.

- ♦ **Nivel de servocontrol.** Los sensores virtuales [ferrell 1994] proporcionan a los sistemas de control de las capas superiores información de los sensores ya procesada. El objetivo de los sistemas que tratan los fallos a este nivel es que los sensores virtuales provean la información correcta aun cuando alguno de los sensores físicos no estén funcionando aunque lo hagan a costa de pérdida de las capacidades del sistema sensorial. Para ello es necesario una cierta redundancia en la información, de manera que los datos del sensor estropeado se puedan obtener o deducir de la de sus vecinos o de otros redundantes. De todas formas, si el fallo es global, [ferrell 1993] debe ser tratado a un nivel superior.
- ♦ **Nivel reactivo.** A este nivel se pueden usar conjuntos de estrategias de control [Payton 1992] con prestaciones diferentes según la situación o el estado de funcionamiento del robot. Si las prestaciones obtenidas por la estrategia de control actual no es la adecuada, entonces se intentará usar una estrategia distinta. Los fallos que se acostumbran a tratar a este nivel son más bien fallos de las capas de control o sensores virtuales. Si el problema no se puede resolver a un nivel inferior es sin embargo inevitable tratarlos con una estrategia de control diferente o bien a nivel de planificación cambiar la tarea o realizarla de forma alternativa.

4.3 Tratamiento de fallos en el sistema sensorial

En este apartado se hará una descripción de las distintas formas en que se ha tratado el problema de detección, diagnosis y recuperación de fallos a nivel sensorial. Al igual que se ha hecho en el capítulo anterior, en primer lugar se caracterizan los tipos de fallos a detectar y luego se describen las distintas fases del proceso: extracción de síntomas, detección, diagnosis y recuperación incluyendo al final comentarios sobre los puntos fuertes de cada método.

4.3.1 Modelos y tipos de fallos a detectar

En el capítulo anterior se ha definido un fallo como la desviación no permitida del valor de una variable característica del sistema con respecto a su valor nominal. Esta definición sigue siendo válida para el caso de robótica móvil si bien es más amplia para algunos autores como se ha resaltado al principio de este capítulo. Los fallos a detectar en el nivel sensorial son muy parecidos a cualquier otro proceso de los descritos anteriormente aunque depende del tipo de sensor. Basta con ver algún ejemplo que se ha tratado en la bibliografía sobre el tema.

En [Stergios 1998b] se trata la detección e identificación de fallos en los sensores considerando dos tipos de fallos: Fallos *graves* como aquellos en los que el valor del sensor se bloquea en un valor fijo y fallos *suaves* como aquellos en los que el sensor va degradando la precisión de los resultados paulatinamente. Los sensores tenidos en cuenta en esta referencia son un giroscopio en el cual se consideran los dos tipos de fallos y los codificadores angulares (encoders) asociados a cada una de las ruedas para los cuales se tratan sólo los fallos *graves*. En [Stergios 1998] se usa la misma arquitectura pero los fallos a detectar son “rueda derecha pinchada” o “rueda izquierda pinchada”.

En [Murphy 1996] se considera que el agente realiza una tarea haciendo uso de comportamientos independientes entre sí (comportamientos distintos tienen sistemas sensoriales lógicos distintos). Es una extensión a la arquitectura de fusión sensorial (SFX) presentada en [Murphy 1992] a la que se le añade el tratamiento de excepciones dando lugar a SFX-EH. En este caso no se limita a fallos hardware sino a cualquier fallo en el sistema sensorial. Es decir, cualquier evento que conlleve una percepción defectuosa. Incluyendo por tanto problemas a nivel software al controlar los fallos sobre sensores virtuales. El gestor de eventos supone que hay sólo un fallo a la vez lo cual no siempre es cierto.

En [Ferrell 1994] [Ferrell 1993] se presenta un sistema de identificación y recuperación de fallos sensoriales en Hannibal [Angle 1990], un robot de seis patas construido en el MIT. Los fallos que se tienen en cuenta son los producidos en sensores y actuadores. Los errores son tratados en el nivel más bajo posible para que las capas superiores no se enteren de tales fallos.

4.3.2 Extracción de síntomas

De las distintas formas de obtención de síntomas descritas en el capítulo anterior, la más utilizada a nivel sensorial es la generación analítica mediante filtros o análisis de las lecturas basadas en redundancia. Un ejemplo se puede ver en [Stergios 1998b] [Stergios 1998] donde la fase de extracción de síntomas es un generador analítico formado por un banco de filtros de Kalman como se muestra en la figura 4.1. Los síntomas generados son los residuos de los filtros que serán utilizados por los módulos de detección e identificación. El modelo del sistema es el modelo cinemático del robot donde las lecturas de los sensores y las estimadas vienen dados por:

$$z = [w_L, w_R, \dot{f}]^T$$

$$\hat{z} = \begin{bmatrix} \hat{w}_L, \hat{w}_R, \hat{f} \end{bmatrix}^T$$

$$r = z - \hat{z}$$

donde w_L y w_R son las velocidades angulares de las ruedas izquierda y derecha medidas por los codificadores angulares (encoders), f es la rotación medida por el giroscopio y r es el residuo. Los valores con acento circunflejo (^) son los valores estimados.

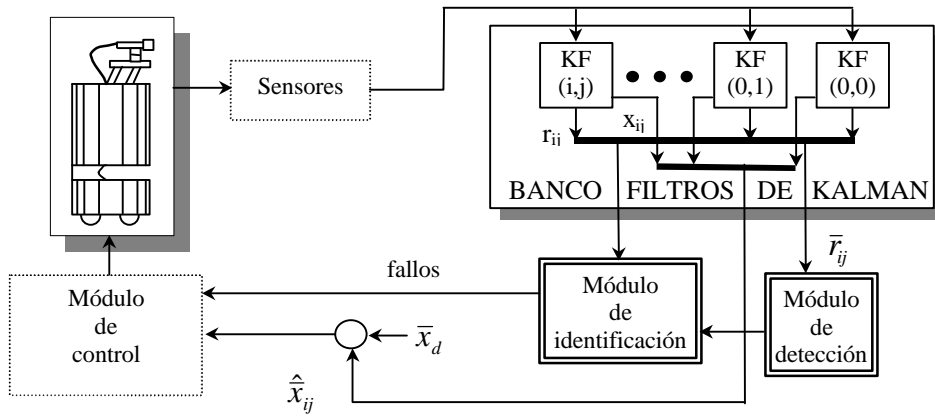


Figura 4.1. Un banco de filtros de kalman extrae los residuos para la detección de fallos. [Stergios 1998] .

Existe un filtro de Kalman para cada una de las condiciones de error y otro para el estado nominal resultando un total de cuatro filtros en [Stergios 1998b] (uno para el caso nominal, otro para cada tipo de error) y tres en [Stergios 1998] (uno para el estado nominal, otro para el caso de la rueda derecha pinchada y el último para el caso de la rueda izquierda pinchada).

En [Murphy 1996] se obtienen los síntomas a partir de la falta de consenso en las observaciones. Es decir, se usa información entre sensores contiguos así como características conocidas de las lecturas y se contrastan los resultados. Se consideran tres tipos diferentes de síntomas:

- ✓ No hay lecturas disponibles.
- ✓ Incertidumbre demasiado alta en los datos.
- ✓ Alto grado de incoherencia en los datos.
- ✓ Poca claridad en los datos.

Otros sistemas están basados en un modelo de estados pero, en lugar de usar la ecuaciones de estado como se hacía en el apartado 3.3.1.2 que, por otra parte, serían difícilmente obtenibles, se genera un modelo de estados en base al conocimiento específico del comportamiento del sistema.

Por ejemplo, cada una de las patas del robot Hannibal [Ferrell 1994] [Ferrell 1993] puede estar en cuatro posibles estados (S_{inicio} , S_{medio} , S_{fin} y $S_{\text{excepción}}$) y cada uno de los sensores de la pata debe tener unos valores para cada estado. La transformación de estados de la pata a valores de sensores se establece experimentalmente de los datos obtenidos mientras el robot se mueve. Los síntomas vienen dado por señales (“*pain level*”) que representan el grado de dañado que se encuentra el sensor, dicho valor se obtiene de la comparación de los datos del modelo con los datos reales:

- ✓ Un proceso (“*consensus monitor process*”) determina el estado en que se encuentra cada pata mediante un mecanismo de consenso. Cada sensor informa sobre el estado en que considera se encuentra la pata de acuerdo con la medidas que posee. Para eliminar la influencia de los sensores con fallos, sólo se tienen en cuenta los informes de aquellos sensores que en el anterior paso se han considerado que funcionan correctamente.
- ✓ A continuación se establecen los síntomas (“*pain level*”) en función del desacuerdo de los valores del sensor con los valores que debería de tener en ese estado. Más concretamente, cada vez que hay desacuerdo un agente (“*injury agent*”) incrementa el nivel de gravedad (“*pain level*”), y si hay acuerdo lo disminuye.

En general, se puede concluir que los síntomas generados dependen de los fallos y sensores tratados, haciéndose necesaria la construcción de un método a medida para cada caso.

4.3.3 Detección y diagnosis

Recordemos del capítulo anterior (figura 3.1) que la tarea de diagnosis se puede dividir en las funciones de detección, ubicación e identificación. Estas tres funciones no están siempre desarrolladas de forma aislada como es el caso de las de ubicación e identificación que, en la mayoría de los sistemas, aparecen como un único módulo.

En algunos casos [Stergios 1998b] [Stergios 1998] se utilizan módulos diferentes para detección e identificación, dado que este último necesita, por lo general, mayor tiempo de procesado. La fase de detección de fallos (FD) se basa en un filtro que produce varios residuos (uno para cada estado) que cuantifican el nivel de discrepancia con cada estado. No existe un fallo si el residuo menor es el correspondiente al estado nominal, en cualquier otro caso se considera que existe un fallo y se activará el módulo de identificación para determinar el tipo de fallo. El mecanismo de identificación consiste en un test de hipótesis sobre los residuos.

En [Murphy 1996] se usa un método de clasificación siguiendo el procedimiento básico de generación y comprobación:

- ✓ Generar todas las posibles causas basadas en el síntoma producido.
- ✓ Ordenar la lista de test asociados y ejecutarlos para confirmar las posibles causas.
- ✓ Clasificar, una vez finalizados, todos los test.

Realiza una búsqueda exhaustiva de todos los posibles fallos, por tanto, se detectarán aquellos fallos tenidos en consideración.

El proceso de diagnosis analizado en [Ferrell 1994] [Ferrell 1993] consiste en comparar la señal “*pain level*” de cada uno de los sensores o actuadores con un valor umbral. Si se supera dicho umbral entonces se considera el sensor dañado, en caso contrario, se supone en buen estado.

Se han aplicado otro tipo de técnicas, como las basadas en redes neuronales [Vemuri 1998] en campos de la robótica más estudiados como son los manipuladores industriales.

4.3.4 Recuperación

Los algoritmos de recuperación son implementados en muy pocos procesos de tratamiento de errores en robótica móvil. Así, mientras en [Stergios 1998b] [Stergios 1998] no se aborda ese problema, en [Ferrell 1994] [Ferrell 1993] el mecanismo de recuperación puede llevarse a cabo a distintos niveles, según se trate de un fallo

enmascarable a nivel sensor virtual o de un fallo grave que tendrá que ser resuelto por un nivel superior. A nivel sensor virtual, la recuperación se produce obteniendo las lecturas a partir de otras fuentes que no sea el sensor fallido mediante un proceso de enmascarado de forma que los fallos son transparentes a los niveles superiores salvo que sean irrecuperables. Está clara la necesidad de una cierta redundancia porque, de otra forma, no sería posible obtener la información necesaria para la recuperación del sistema en las capas bajas.

Para que el sensor vuelva a considerarse operativo se establecen los siguientes mecanismos:

- ✓ Reintento: Aunque se descartan sus lecturas el agente dañado (*injury agent*) continúa actualizando el nivel de gravedad (*pain level*) para ver si el sensor vuelve a su estado normal.
- ✓ Recalibración dinámica: Debido a que los sensores se van deteriorando con el tiempo se produce una recalibración del modelo entre los estados de la pata y los valores del sensor.

Por último, los fallos globales (*catastrophic failures*) son compensados por el control de alto nivel.

4.3.5 Ventajas e inconvenientes

El principal problema de los sistemas que modelan el comportamiento de los sensores de forma matemática como en [Stergios 1998b] [Stergios 1998] es que los fallos tienen que estar modelados analíticamente, de forma que el estado del sistema se pueda evaluar utilizando un estimador tal como un filtro de Kalman, lo cual es bastante restrictivo en muchos casos prácticos. Sería muy difícil aplicar esta técnica para determinar fallos en otro tipo de sensores, tales como los sonar cuyas lecturas dependen del entorno en el que existen objetos no modelados. Tampoco es aplicable a fallos generales en el proceso de navegación dado que su complejidad y naturaleza abstracta hace imposible encontrar un modelo analítico del mismo.

Sistemas como el presentado en [Murphy 1996] tienen el problema que los test aplicados se basan en un dominio dependiente del conocimiento. Si bien la arquitectura puede ser aplicable en otros dominios, la generación de test y su aplicación tiene que ser reconstruido completamente o bien es necesario un sistema de aprendizaje automático. Además, al hacer una búsqueda exhaustiva, el sistema tiende a hacerse demasiado lento.

Para modelar el comportamiento de algunos sensores se han propuesto distintas arquitecturas donde la detección, en la mayor parte de los casos se basa en el

uso de información redundante entre sensores contiguos, de forma similar a los casos anteriormente citados. También hay que destacar la complejidad de la detección en ciertos sensores cuyas salidas no pueden ser previstas como pueden ser los sensores sonar debido a la presencia en el entorno de objetos no modelados y objetos móviles. Hasta la fecha, pocos autores los tienen en cuenta para el tratamiento de errores. Esto presenta un reto en la detección de errores producidos a nivel sensorial que, posiblemente deba tratarse a un nivel superior.

4.4 Control de errores al nivel de navegación

En el caso de tratamiento general de errores en el sistema de navegación no se utilizan modelos matemáticos por distintos motivos:

- ✓ Complejidad del modelo. Basta con estudiar algunas de las arquitecturas desarrolladas a lo largo de la bibliografía técnica, como las presentadas en el segundo capítulo de esta tesis, para contemplar el alto número de procesos que intervienen así como su compleja interconexión.
- ✓ Imposibilidad de modelar eventos imprevistos. El entorno del robot no es un entorno estático y no se pueden prever todos los eventos que pueden ocurrir.
- ✓ Dificultad en la separación entre lo que constituye un fallo y lo que constituye una situación nominal.

Las divisiones de las fases de obtención de síntomas, detección, diagnosis y recuperación no están tan claras como en los casos anteriores debido a que el tratamiento de errores a este nivel depende en gran medida de la arquitectura de control previamente construida.

4.4.1 Modelos y tipos de fallos a detectar

Conviene en este punto aclarar lo que se entiende por error en la navegación. En la mayoría de los casos se detecta un error cuando no se cumplen los objetivos de una determinada tarea. De esta forma, retomando el ejemplo utilizado al principio de este capítulo en el que el robot debería ir de un extremo a otro de un pasillo, si finalmente no alcanza el otro extremo, se ha producido un error. No obstante, puede ser que todos los sistemas (hardware y software) del robot estén funcionando correctamente y el problema radique en que el pasillo está bloqueado por cualquier tipo

de objeto. En lo que sigue, se utilizará el término *situaciones de excepción* en lugar de error para incluir este tipo de eventualidades no previstas.

Uno de los primeros trabajos en recuperación de errores en robots móviles se debe a Sankaran [Sankaran 1977]. El sistema se aplica a un robot móvil con un brazo articulado para coger rocas.

El funcionamiento general del anterior sistema se describe en la figura 4.2. Un *plan* está formado por varias acciones que se ejecutan de forma consecutiva. Al finalizar cada acción se comprueba que ésta ha llegado a buen término y, en caso negativo, se activa el analizador de fallos que construye un árbol con todas las posibilidades. Con la ayuda de un analizador que puede realizar y comprobar los resultados de algunos test, se identifica el fallo y se modifica el plan si es necesario o bien se repara directamente el fallo.

En [Noreils 1990], al igual que en todos los casos que abordan la recuperación de fallos al nivel de navegación, la arquitectura del detector de fallos depende, en gran medida, de la arquitectura de navegación, no siendo directamente aplicables a otros sistemas de navegación. La arquitectura del sistema de control [Noreils 1995] posee un módulo de supervisión que recibe las órdenes a través del interfaz de usuario, los interpreta y envía las tareas que componen las órdenes al módulo ejecutivo.

El módulo ejecutivo funciona como un sistema operativo gestionando los recursos del robot. Mientras las tareas se realicen correctamente el ejecutivo se encarga de ir lanzando las tareas de forma secuencial. Cuando una tarea falla, el ejecutivo hace una llamada al módulo de diagnosis y recuperación de fallos quien se responsabiliza de determinar el fallo y realizar las acciones oportunas para la recuperación. Los tipos de fallos a detectar pueden ser bien situaciones de excepción externas al robot debidas a cambios en el entorno o fallos internos del robot como fallos en el funcionamiento de los sensores.

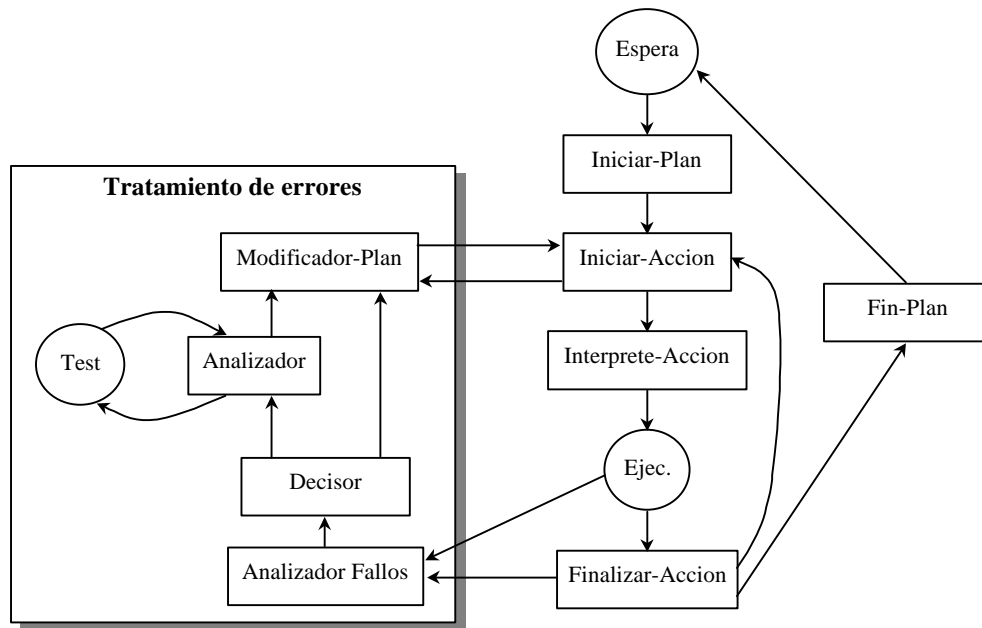


Figura 4.2. Modelo de tratamiento de errores planteado en [Sankaran 1977].

En [Stuck 1992] [Stuck 1995] se utiliza el término equivocación (mistake) para referirse tanto a eventos en la percepción, problemas del navegador o eventos del sistema motriz que desvían al robot de la ruta original. En este sistema de detección y recuperación se emplea, además de procesamiento cuantitativo, descripciones cualitativas acerca de las referencias del entorno. Tampoco aquí se contemplan fallos en el sistema de control.

El robot Hilare [Fleury 1994] [Ghallab 1985] posee un monitor de la ejecución utilizando un sistema de producción. Entre otras, puede detectar situaciones de excepción, tales como ciertos fallos en los sensores, niveles de carga de la batería demasiado bajos, obstáculos inesperados o marcas de referencia en lugares distintos de los esperados.

4.4.2 Extracción de síntomas

En [Sankaran 1977] la obtención de síntomas se realiza de dos formas. La primera se lleva a cabo verificando las precondiciones de cada acción para ver si ésta se puede realizar, y la segunda es comprobando si la tarea se ha terminado correctamente.

En [Noreils 1990] [Noreils 1995] el ejecutivo es el encargado de la detección de errores. Cada tarea tiene unos objetivos que cumplir y si dichos objetivos no se cumplen o, durante la realización de la tarea se detecta algún problema, se supone que se ha producido un error. Los síntomas de un fallo son las manifestaciones de éste en la consecución de las tareas. Por ejemplo, si la cámara no esta funcionando mientras se ejecutan tareas que no la necesitan, dicho fallo no será detectado y será irrelevante; pero, por el contrario, si la tarea consiste en acercarse a un objeto que debe ser identificado usando la cámara, esta tarea no se llevará a cabo al no poderse identificar dicho objeto.

En [Stuck 1992] [Stuck 1995] se caracteriza el entorno con una serie de balizas que el robot utiliza como referencias. Las referencias visuales del entorno denominadas *entidades* poseen tres tipos diferentes de atributos:

- ✓ Atributos independientes de la posición. Por ejemplo, forma de la entidad, tamaño, etc.
- ✓ Atributos dependientes de la posición, tal como la orientación de la entidad, posición, etc.
- ✓ Semejanzas/diferencias con otras entidades. Se utilizan para diagnosticar posibles equivocaciones.

La descripción de una ruta de navegación viene dada por una serie de instrucciones de dos tipos:

- ✓ Instrucciones que especifican movimientos a ejecutar en términos de distancias, direcciones, etc.
- ✓ Relación de entidades que el robot ha de ir viendo a lo largo del camino.

Los síntomas son discrepancias entre los valores esperados (entidades que se supone que debe ir viendo el robot) y los observados en realidad definidos ambos como información visual simbólica. Para obtener los síntomas que en este caso es una secuencia de comparaciones entre entidades, se ha de comparar lo que el robot “ve” con la lista de entidades descrita en la ruta de navegación y también con todas las entidades posibles para ver cual es la que tiene más parecido en los dos casos. Los

resultados de las comparaciones producen un número entre cero y uno denominado *compatibilidad* que describe la semejanza entre entidades o grupos de entidades.

Considera tres tipos diferentes de equivocaciones :

- ✓ Falsos negativos cuando el robot no “ve” una entidad. Normalmente conlleva a que el robot se pase de largo.
- ✓ Falsa identificación cuando el robot confunde una identidad con otra. Dependiendo de si la entidad identificada esté o no en las previsiones puede conllevar a que el robot no detecte una referencia y se pase de largo o que la detecte donde no exista.
- ✓ Fallos en el control del motor (deriva en la dirección o velocidad). Las consecuencias pueden ser diferencias en los desplazamientos o fallos en los giros.

Todo ello se aplica bajo el entorno de simulación MUCKLE.

4.4.3 Detección y diagnosis

Al igual que en etapas anteriores, se han realizado multitud de mecanismos diferentes. Por ejemplo, en [Sankaran 1977] la diagnosis se lleva a cabo mediante la construcción de un árbol que relaciona acciones con posibles fallos así como tests que se pueden realizar para la diferenciación entre errores. Dicho árbol se construye a partir de una base de conocimientos donde se almacena información sobre los errores y acciones.

Para la diagnosis en [Noreils 1990] [Noreils 1995] se utiliza información causal almacenada para cada una de las distintas tareas. Dicha información incluye una relación de los módulos implicados en la tarea. También se indica si la implicación es de forma sistemática o de forma eventual. Se incluye, además, una relación de mensajes de error por cada módulo, e información acerca de la recuperación para cada caso.

Una vez que se detecta un error, la situación se analiza mediante la construcción de un árbol de fallos a partir de la información asociada a la tarea en curso. La selección del fallo entre todos los posibles se realiza mediante un método de poda o eliminación a partir del árbol de fallos.

En [Fleury 1994] [Ghallab 1985] se utiliza un sistema de producción para determinar en qué situación de excepción se encuentra el robot y en función de la misma se toma una decisión.

En otros sistemas [Stuck 1992] [Stuck 1995] se utiliza estimación de tiempos para calcular los valores simulados (o esperados) para determinar la relación de entidades que el robot ha de ir viendo mientras sigue el camino. Para determinar si se ha producido una *equivocación* debe tener en cuenta no sólo la instrucción actual, sino la evolución de los resultados obtenidos a lo largo del tiempo. Para ello utiliza una medida denominada *convicción* que representa la certeza que tiene el robot a cerca de su buen funcionamiento. Se usan funciones sigma para ir incorporando información sobre comparaciones entre valores reales y simulados al factor *convicción*. En este caso, la diagnosis es más completa dado que necesita saber en qué punto del camino se produjo la equivocación y cual fue la causa. Una vez detectada una *equivocación*, el proceso de diagnosis se realiza en cuatro pasos. El primero consiste en analizar la historia para producir una serie de estadísticos acerca de las duraciones de las instrucciones y otros parámetros observables del sistema. El segundo consiste en utilizar tres tipos diferentes de estrategias para tratar de encontrar evidencias de cada una de las tres equivocaciones posibles. El tercer paso es un refinamiento de las evidencias, y el último usa conocimiento heurístico para seleccionar la mejor hipótesis.

4.4.4 Recuperación

Las acciones a tomar dependen de la situación de excepción en la que se encuentre el robot. En algunos casos vienen dados por los consecuentes de reglas [Ghallab 1985] [Noreils 1990] o bien se asocia una acción de recuperación para cada una de las posibles situaciones. Por ejemplo, en el caso de encontrarse un obstáculo que bloquea completamente el camino que impida completamente el paso, la acción a tomar consiste, en la mayor parte de las veces, en buscar un camino alternativo si éste existe. En general, este es uno de los aspectos menos tratados en los sistemas desarrollados hasta el momento.

El problema de tratamiento de fallos también se ha tratado en cooperación multirobot. En [Parker 1998] se desarrolla una arquitectura tolerante a fallos. Esta característica se consigue disponiendo de varios robots para la realización de las tareas y utilizando un control distribuido. Normalmente cada tarea es realizada por un único robot pero dependiendo de dos factores (*impaciencia* y *consentimiento*), otro robot puede pasar a realizar dicha tarea si cree que el primero no será capaz de finalizarla.

5 Arquitecturas y sistema de navegación

En este capítulo se describe la plataforma de robot móvil sobre la cual se ha llevado a cabo la investigación, al mismo tiempo que se indican las modificaciones que han tenido que ser hechas para la introducción del módulo de supervisión.

En cuanto a la plataforma hardware, se han usado tres robots móviles apodados *Xavier*, *Amelia* y *Rato*. *Xavier* es un robot construido en CMU (*Carnegie Mellon University*) usado más tarde por la compañía RWI (*Real World Interface*) como prototipo para la elaboración del robot B-21. *Amelia* es un robot modelo B-21 adquirido por CMU como plataforma de investigación. Por último, el robot disponible en el departamento de Ingeniería de Sistemas y Automática, *Rato*, es del mismo modelo que *Amelia* pero con menor capacidad sensorial; no dispone de las cámaras, cabeza móvil para orientarlas ni de sistema láser.

La mayor parte de los tests se han realizado sobre el primero (*Xavier*), aunque todos comparten la misma arquitectura de navegación. La facilidad de migración entre plataformas refleja la flexibilidad del sistema de navegación y supervisión, si bien entre los tres modelos mencionados anteriormente existen bastantes similitudes físicas.

Ya en julio de 1996 *Xavier* había completado unas 1800 peticiones y viajado más de 75km desde que comenzó a funcionar en Internet (diciembre de 1995) [Simmons 1997b]. La robustez de esta arquitectura queda patente en la tasa de consecución con éxito de tareas que se sitúa en torno al 95%.

En el siguiente apartado se describen los componentes hardware de *Xavier* haciendo especial hincapié en el sistema sensorial. A continuación, se caracterizan las diferencias de *Rato* y *Amelia* con respecto a *Xavier*. En el tercer apartado se analiza la arquitectura del sistema de navegación, los distintos niveles funcionales y el flujo de información entre ellos. Por último, se describe la división en procesos y la forma de interconexión de estos.



Xavier

5.1 *Xavier*

Con objeto de participar en la competición de robots móviles organizada por la AAAI en 1993, un grupo de estudiantes de doctorado bajo la dirección del profesor Reid Simmons diseñaron el robot Xavier en el laboratorio *Robot Learning Laboratory* de la universidad de Carnegie Mellon. Desde entonces se han seguido haciendo modificaciones sobre esta plataforma hasta llegar a la configuración actual mostrada en

la figura 5.1 donde se pueden distinguir algunos de los distintos componentes, así como su aspecto externo.

El robot consta de una base cilíndrica, suministrada por RWI, donde se ubican las baterías y el sistema motriz, y un cuerpo alrededor del cual se encuentra un anillo de 24 sensores sonar, con un láser en la parte delantera y una cámara en la parte superior.

5.1.1 Base

Xavier está construido sobre una base móvil (RWI B21) de 60,1 cm. de diámetro. La base incluye las baterías (cuatro baterías de 14 voltios cada una) que suministran energía al resto del sistema, motores que le proporcionan la capacidad de movimiento, circuitos necesarios para el control y codificadores angulares (encoders) para obtener la información odométrica. Además, está rodeada por una serie de sensores táctiles para detectar posibles choques con obstáculos.

El sistema de dirección está constituido por cuatro ruedas sincronizadas. Para hacer un giro, cada rueda gira sobre sí misma de forma solidaria con el tronco, mientras que la base permanece siempre orientada en la misma dirección. Esta característica, junto con la forma circular del robot, simplifica la tarea al programador que no tiene que preocuparse de evitar girar el robot cuando éste se encuentra pegado a un obstáculo o pared. Esto no sería posible si el robot fuera poligonal o no pudiera girar sobre su centro geométrico. La forma redonda del robot junto con la distribución simétrica de los sensores también simplifica la realización del simulador y los programas de control reactivo como se verá más adelante. La velocidad máxima está en torno a los 90 cm. por segundo. Los movimientos de rotación y translación pueden superponerse de forma que el robot es capaz de describir cualquier trayectoria.

La alimentación de los servomotores y del resto de los dispositivos a bordo del robot proviene de las cuatro baterías situadas en la base. Estas baterías están conectadas en serie, proporcionando un total de 56 voltios (carga completa) que permiten una autonomía de hasta 6 horas cuando están nuevas y completamente cargadas.

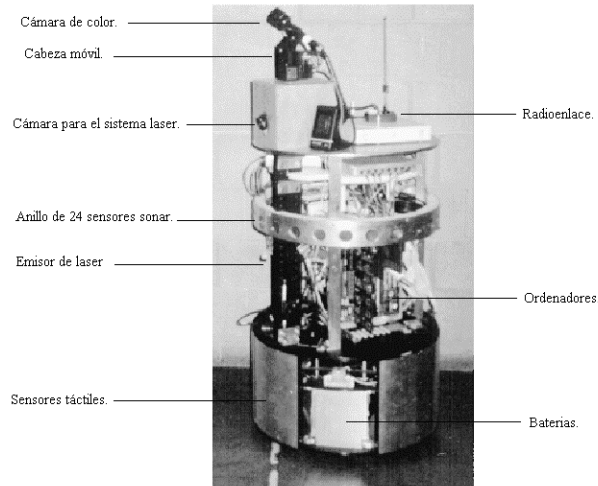


Figura 1. Arquitectura hardware del robot desarrollado en Carnegie Mellon (Xavier)

En la base también se incluye el control básico del sistema motriz. Dos motores proporcionan al robot los movimientos de traslación y rotación, mientras que un procesador de control básico libera a los ordenadores de a bordo del cálculo de las actuaciones directas sobre los motores.

La odometría es el método de posicionamiento más utilizado para la navegación de robots móviles, si bien se suele utilizar conjuntamente con otros métodos para corregir los errores inherentes a la misma. Usando este método, la base del robot proporciona información de estado acerca de su posición (x,y) y orientación (θ) con respecto a un punto de referencia (lugar de arranque) basándose en datos obtenidos de los codificadores angulares (encoders). Estas medidas de posición y de orientación presentan errores que, si bien son pequeños para desplazamientos cortos, pueden llegar a ser considerables en grandes desplazamientos al tratarse de errores acumulativos. La principal fuente de problemas en estas medidas se debe a las pequeñas variaciones en la orientación que pueden provocar grandes errores en la posición final si estos no se corrigen. Existen varias fuentes de error como pueden ser

una mala alineación de las ruedas, deslizamiento de las mismas en cierto tipo de superficies, etc.

La cubierta alrededor de la base posee ocho sensores táctiles que, en caso que el robot choque con un obstáculo, permite conocer la dirección en que se encuentra el obstáculo.

La conexión de la base con el resto del robot se hace a través de un conector de 24 patillas (pines) por donde suministra la energía necesaria para alimentar al resto de los dispositivos y se comunica con el procesador de uno de los ordenadores de a bordo.

5.1.2 Sistema sensorial

Además de la información proporcionada por el sistema de odometría y los sensores táctiles situados en la base, el robot posee una serie de sensores que le proporcionan información acerca del entorno, tales como distancias a objetos y posición de los mismos. Resumiendo, los principales tipos de sensores son:

1. Odometría. Se ha visto anteriormente que la base del robot obtiene una estimación de la posición del robot con respecto a la inicial.
2. Sonar. Formado por un anillo de 24 sensores. Son los más usados debido a su amplio campo de detección, pero son bastante susceptibles al ruido y tienen problemas con algunos tipos de superficies.
3. Láser. El sistema está compuesto por dos emisores de luz láser y una cámara con un filtro que separa el haz del láser (reflejado por los objetos) del resto de la imagen.
4. Visión. Está formado por una cámara de color sobre una cabeza móvil. Actualmente no se usa para navegación debido, entre otras razones, a los requerimientos de tiempo necesarios en el procesado de imágenes. Sin embargo, se utiliza para otras tareas, tales como detectar puertas de los despachos y aulas, capturar imágenes y enviarlas a través de Internet bajo demanda, etc.
5. Sensores táctiles. Se usan como sistema de seguridad, dado que sólo detectan objetos cuando el robot choca con ellos.

En lo que sigue, se describe más en detalle los principales sensores utilizados por los programas de control del sistema de navegación.

5.1.2.1 Sonar

Xavier posee un anillo de 24 sensores sonar equidistantes situados alrededor de su parte superior, soportados por una chapa metálica. Según los diseñadores [O'Sullivan 1996] se ha elegido este número para abarcar todo el espacio alrededor del

robot sin dejar zonas muertas, en las que el robot no pueda detectar los obstáculos. Estos sensores sonar Polaroid comprenden un rango entre 10,16cm. y 609cm. modificable a través de un transductor. Los sensores se encuentran conectados a tres módulos de control que a su vez van a unidos a una unidad central, conectada a uno de los ordenadores de a bordo.

Uno de los principales problemas de estos sensores es su baja cadencia en la obtención de medidas. Considerando la distancia máxima a la que es posible detectar objetos, y que la señal se desplaza a la velocidad del sonido, se necesitaría aproximadamente 0,05 segundos para cada sensor manteniendo unos márgenes de seguridad. Ello quiere decir que los 24 sonar necesitan 1,2 segundos para actualizar las lecturas de todos ellos. Este tiempo es demasiado grande considerando las velocidades a las que se suele mover el robot. El tiempo total de muestreo se reduce aproximadamente a 0,25 segundos cuando se disparan 4 sonar a la vez [O'Sullivan 1996]. Para eliminar las interferencias se disparan los cuatro más alejados entre ellos (es decir, separados 90 grados). De esta forma, se obtiene una frecuencia de muestreo sobre 4 veces por segundo.

La comunicación de los ordenadores de a bordo con estos sensores se realiza mediante una conexión RS-232.

5.1.2.2 Láser

En la figura 5.2 se muestra un esquema sobre el funcionamiento del láser. Como se aprecia en la figura, el sistema esta formado por dos emisores de luz (de unos 20 grados de apertura cada uno) y una cámara con un filtro para extraer una línea del láser de la imagen. Por ejemplo, para el entorno presentado en la figura 5.2 donde el robot se encuentra frente a un recipiente como un cubo, la imagen producida por el sistema es la que aparece en la figura 5.3.

Este sistema ha sido adaptado del original, proporcionado por la empresa **Nomadics**, cambiando la orientación de uno de los emisores.

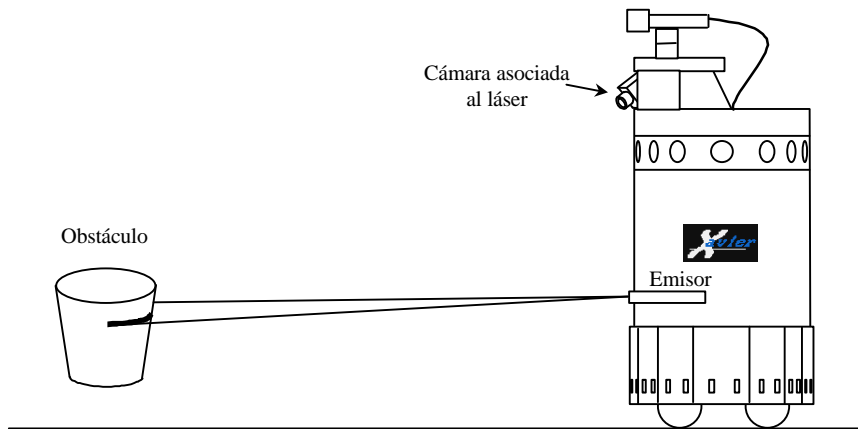


Fig 5.2 Funcionamiento del láser. El haz de luz incide sobre los objetos quedando patente en la imagen obtenida por la cámara.

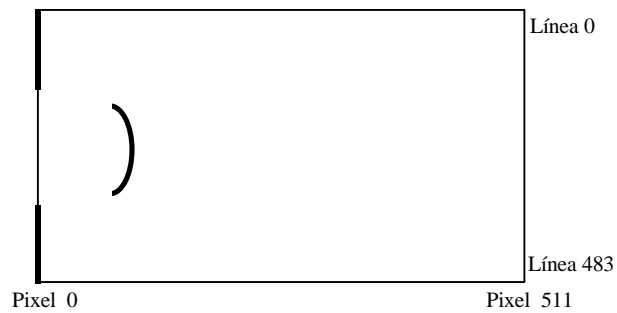


Fig 5.3 Imagen obtenida a la salida del sistema láser para el escenario de la figura anterior.

En la actualidad, *Xavier* posee dos emisores de luz: uno está enfocado hacia adelante y el otro se ha orientado hacia el suelo para la realización de esta tesis. La razón de orientar el láser al suelo es doble:

- ✓ Como se puede observar en la figura 5.5, permite detectar objetos negativos. En particular, se ha adaptado para detectar las escaleras con rapidez suficiente para frenar en el caso peor, que circule a velocidad máxima. Los cálculos detallados de parámetros como la inclinación o períodos de muestreo se describen en [Fernández 1997]
- ✓ Permite detectar objetos que quedan por debajo del haz de luz láser orientado hacia adelante y que, a su vez, pueden no ser detectados por el sonar.

La conversión de puntos de la imagen filtrada (*pixels*) a posiciones (coordenadas) relativas al centro del robot se realiza con la ayuda de un archivo de calibración, en el cual se encuentran las coordenadas asociadas a cada punto de la imagen. Dicho archivo se obtiene a través de un proceso de calibrado [Fernández 1997]. Debido a que la orientación de los dos emisores ha sido modificada, situándose en distintos ángulos verticales respecto al robot, es necesario disponer de dos archivos de calibración y variar también el funcionamiento de los programas de control originales.

La elaboración de estos programas de calibrado se describe en [Fernández 1997] y en las figuras 5.3 y 5.4 se muestra una vista en 3D de los valores (x e y) de los archivos resultantes de la calibración. En la figura 5.3 se muestra la tabla de calibración para el láser orientado hacia adelante donde el eje z se corresponde con la coordenada x (caso a) e y (caso b) con respecto al centro del robot. La figura 5.4 representa las tablas para el caso del láser orientado hacia el suelo. La situación del robot con respecto a la imagen se puede observar en la figura 5.2 donde se observa que el pixel 0 de cualquier línea es el más alejado del robot. Hay que aclarar que los valores de X e Y para la mayor parte de los puntos de la gráfica 5.4 es -1 porque ese es el valor asignado a aquellos pixels de la imagen más allá del punto donde el haz del láser incide en el suelo.

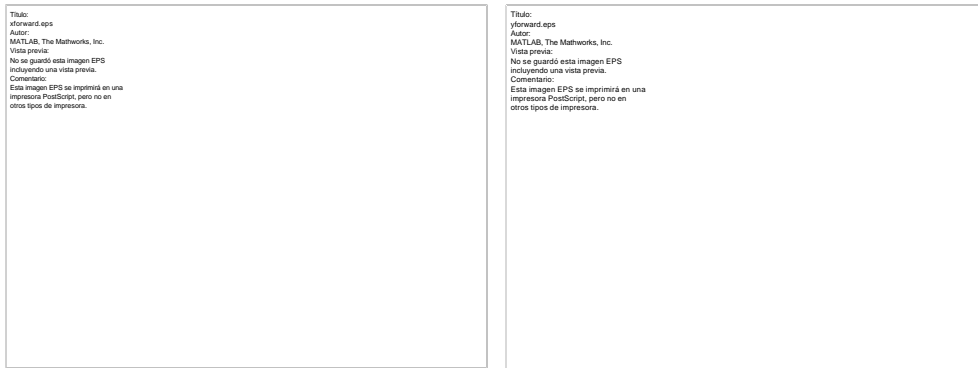


Fig 5.3 Tablas de calibración para el láser dirigido hacia el frente.

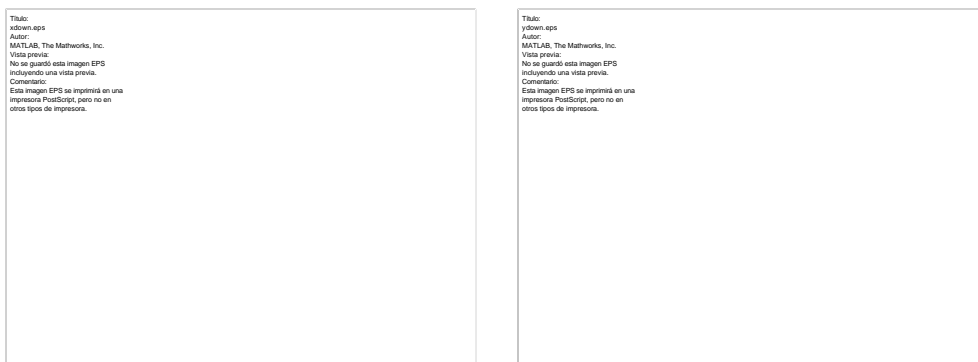


Fig 5.4 Tablas de calibración para el láser dirigido hacia el suelo.

Con esta orientación de los emisores (uno hacia delante y otro hacia abajo), en la imagen obtenida por la cámara, aparecerán dos líneas paralelas cortas en lugar de una larga como ocurría con la configuración inicial cuando los dos emisores estaban en

el mismo plano horizontal. Sin embargo, la salida del sistema es una única línea y, por tanto, no se pueden activar los dos emisores a la vez.

La solución adoptada consiste en emitir un único haz en la obtención de cada imagen para que, alternativamente, se obtenga información de los dos. La cámara asociada al láser puede proporcionar entre 20 y 24 imágenes por segundo. Esto significa una imagen cada 0,05 segundos aproximadamente. Con el nuevo sistema, una de cada dos imágenes pertenece al mismo láser. Debido a los retardos en la conmutación de los emisores y actualización de la imagen, el límite de imágenes capturadas se ha establecido de forma experimental en 10 imágenes por segundo (5 para cada láser).

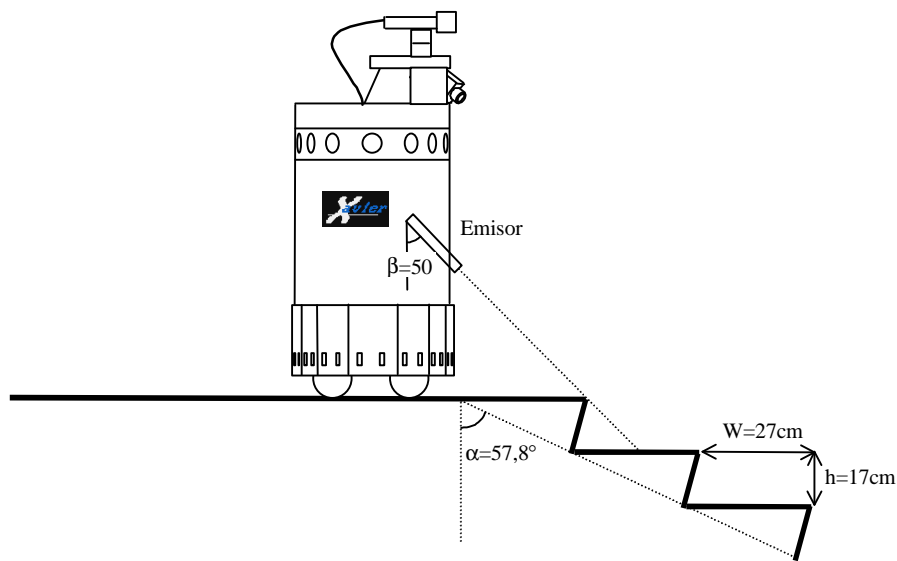


Fig 5.5 Un láser está inclinado hacia abajo para detectar objetos negativos tales como el hueco de la escalera.

Un problema no completamente resuelto con el láser son los reflejos de algunas luces y ventanas que ocasionan la obtención de falsas lecturas en algunos puntos del entorno. Esto puede ser en parte resuelto modificando la apertura de la

cámara pero, en la práctica, se ha comprobado la imposibilidad de encontrar una posición que funcione para todas las situaciones.

5.1.2.3 Sistema de visión

El sistema de visión de Xavier consta de una cámara instalada sobre una cabeza móvil con un rango de giro de 340 grados en horizontal y 78 en vertical. Las características de la cámara así como la tarjeta de adquisición pueden verse en el manual de Xavier [O'Sullivan 1996]. En esta investigación no se ha usado el sistema de visión.

5.1.3 Ordenadores de a bordo.

Las distintas necesidades de procesado a bordo del robot se encuentran cubiertas por tres ordenadores. Dentro del robot hay dos ordenadores PC tipo Pentium al que están conectados todas las tarjetas de control de los distintos dispositivos. Sobre el robot, un ordenador portátil sirve de interfaz con el usuario.

5.1.4 Sistema de voz

Xavier posee además la capacidad de articular expresiones y reconocer una serie de órdenes expresadas en lenguaje natural gracias a una tarjeta de sonido (V8601 de RC Systems) junto con un par de altavoces. Se dispone, además, de algoritmos internos que permiten la articulación de textos en inglés suministrados en formato ASCII.

El sistema de reconocimiento de voz (Sphinx II) ha sido realizado en otro laboratorio de la misma universidad (CMU) y se ejecuta en otra máquina fuera del robot debido a sus necesidades del procesado.



5.2 Amelia y Rato

Se trata, en ambos casos, de modelos **B-21** suministrados por la empresa *RWI* (*Real World Interface*). La diferencia entre ellos es que *Rato* no dispone de la misma capacidad sensorial que *Amelia* como se puede apreciar en la imagen anterior. Básicamente, *Amelia* posee los siguientes sistemas sobre los de *Rato*:

- ✓ Sistema de visión al que se pueden acoplar distintos tipos de cámara.

- ✓ Cabeza móvil para orientar las cámaras en la dirección deseada, tanto vertical como horizontalmente.
- ✓ Escáner de proximidad láser **LMS 200** con un rango aproximado de hasta 30 metros, precisión que puede variar entre ± 1 y ± 10 mm, ángulo de barrido de 180° y resolución angular entre $0,25^\circ$ y 1° .
- ✓ Sistema de voz.
- ✓ *Amelia* posee un brazo retráctil que le permite asir objetos, mientras que *Rato* no dispone de esta facilidad.

Xavier y el modelo B-21 son bastante parecidos, no en vano el primero ha servido de prototipo para la realización del segundo. No obstante, sí hay diferencias relevantes entre las que cabe destacar:

- ✓ Dimensiones del robot. Mientras el diámetro de *Xavier* mide en torno a los 60,1 cm, el de *Rato*, y por tanto *Amelia*, se sitúa en los 51,4 cm.
- ✓ El sistema láser es bastante diferente. El escáner de proximidad **LMS 200** emite un haz de luz láser en un plano paralelo al suelo con una apertura de 180° , mientras que, el usado en *Xavier* tiene dos haces, cuya orientación ha sido modificada, como parte de este trabajo, para detectar objetos de pequeño tamaño y agujeros en el suelo (apartado 5.1.2.2).
- ✓ *Amelia* posee dos anillos de sensores infrarrojos, uno en la base y otro por debajo del anillo de sensores sonar.
- ✓ Además de los sensores táctiles de la base, el modelo B-21 posee una serie de este tipo de sensores en la parte superior.

Los ordenadores de a bordo son diferentes en los tres casos, tanto en lo que se refiere a su número como capacidades de procesado. *Rato* posee un único ordenador de a bordo al que, últimamente, se ha añadido un portátil que sirve de interfaz con el usuario. Tanto *Amelia* como *Xavier* están dotados con dos ordenadores en su interior y un portátil en la parte superior.

5.3 Arquitectura de navegación

Si bien la investigación ha sido llevada a cabo en distintos robots, la arquitectura de navegación usada en todos ellos es la misma. Como se muestra en la figura 5.6, está formada por varias capas funcionales [Simmons 1997b]: servocontrol,

reactivo (evitación de obstáculos), navegación, planificación de trayectorias y planificación de tareas. Cada capa trabaja con la información a un nivel de abstracción provisto por la capa inmediatamente inferior. Se trata de un sistema de navegación que combina una parte reactiva usada en las capas inferiores con una deliberativa representada por las capas superiores. De esta forma se obtienen las ventajas de ambos sistemas; rapidez de respuesta para tratar los eventos que así lo requieran a través de la parte reactiva, sin perder por ello la eficacia de los sistemas deliberativos que determinan las pautas a seguir mediante planificación.

En el nivel inferior se encuentran los mecanismos de servocontrol que permiten acceder a los recursos proporcionados por el hardware disponible (base, cabeza móvil, cámara, sensores, etc.).

La siguiente capa es el módulo reactivo que se encarga del control ante eventos que necesitan respuesta inmediata como la evitación de obstáculos, tratamiento de colisiones detectadas por los sensores táctiles, etc.

Justo por encima se encuentra el sistema de navegación que asume la tarea de localización del robot, además de producir direcciones de referencia a seguir por el módulo reactivo.

Planificación de tareas (PRODIGY)
Planificación de trayectorias (Decision-Theoretic Plan)
Navegación (POMDP)
Módulo reactivo (CVM, "lane method", etc)
Hardware (sensores, motores, ...)

Fig 5.6. Capas en la arquitectura de control del robot.

Los caminos que debe seguir el robot son calculados por el planificador de trayectorias en función de la posición origen proporcionada por la capa de navegación y el destino seleccionado por el planificador de tareas.

5.3.1 Flujo de información

En la figura 5.5 se muestran las principales estructuras de datos presentes en el modelo (representadas por rectángulos), las funciones de cada capa de la arquitectura (descritas mediante elipses) y el flujo de información entre las mismas.

Las tareas solicitadas por el usuario, vía “Zephyr” (utilidad de comunicación a través de mensajes por red) o “Web”, son gestionadas por el planificador de tareas que determina los lugares a visitar y el orden en que deben ser ejecutadas. Mientras existan tareas pendientes, este módulo se encarga de la selección de nuevos destinos a medida que se van alcanzando los anteriores.

El destino, junto con la posición estimada, son usados por el planificador de trayectorias para obtener un *plan* que consiste, en última instancia, en la asignación de una dirección a cada una de las celdas del plano (celdas de un metro cuadrado [Simmons 1995]).

El *plan* calculado por el planificador de caminos y la distribución de probabilidades sobre las celdas del plano sirve al navegador para obtener la dirección en la que debe moverse el robot en cada momento. Esta dirección se envía al módulo reactivo, al cual servirá como referencia para el cálculo de las órdenes que controlan los movimientos de la base.

El sistema reactivo tratará de seguir esta dirección en lo posible evitando los obstáculos que puedan aparecer. Por otra parte, la información obtenida de los sensores es integrada en la elaboración de una rejilla (grid) que caracteriza el entorno del robot. Esta rejilla se usa tanto para la evitación de obstáculos, como para la estimación de posición por parte del navegador.

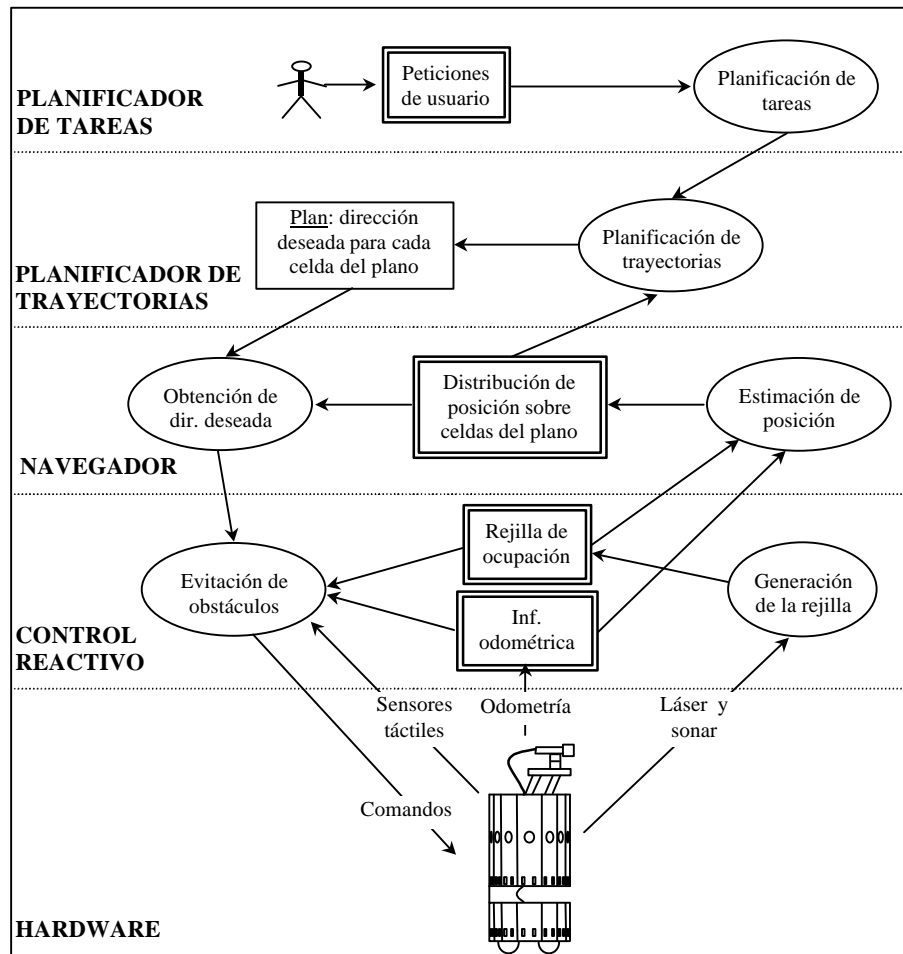


Fig 5.7. Flujo de información.

En el siguiente apartado se describe el funcionamiento de las capas implicadas en la arquitectura con excepción del hardware pues éste ya ha sido descrito en el apartado anterior.

5.3.2 Descripción de las capas

Control reactivo

La función básica del control reactivo consiste en la determinación de los órdenes de movimiento para seguir la dirección indicada por el navegador, evitando los obstáculos no modelados (es decir, no presentes en el plano), dado que los modelados ya son tenidos en cuenta por el planificador de trayectorias.

Aunque el sistema actual permite la elección de varios métodos alternativos, los más usados son: el método de curvatura de la velocidad CVM (Curvature Velocity Method) [Simmons 1996] y su variante LCM (Lane Curvature Method) [N.Y. Ko 1998]. El CVM formula la tarea de evitación de obstáculos como un problema de optimización con restricciones en el espacio de velocidades del robot. Los parámetros de salida son las velocidades lineal y angular, mientras que las restricciones vienen dadas por el entorno (evitar las colisiones) y las limitaciones físicas del robot (velocidades máximas). La función objetivo representa un compromiso entre velocidad, seguridad y diferencia entre la dirección de movimiento con respecto a la de referencia. En [Simmons 1996] pueden verse detalles acerca de su funcionamiento e implantación.

El segundo método (LCM) es una variante que combina el primero con el método de los carriles (lane method). Este último se basa en dividir el espacio en pistas (carriles) paralelas por las que el robot puede circular y elegir la mejor de acuerdo con la dirección de referencia. El LCM calcula primero la pista a seguir de acuerdo con el método de los carriles, a continuación se obtiene la dirección para alcanzar esa pista y dicha dirección es usada por el CVM como referencia para calcular las velocidades finales.

La segunda función de esta capa es la integración de las medidas sensoriales para construir una rejilla que es utilizada para la evitación de obstáculos y la determinación de la posición por los niveles superiores. La rejilla está formada por un conjunto de celdas con probabilidades de ocupación que se calculan a partir de las lecturas sonar y láser.

Navegador

En la figura 5.8 se muestra el interfaz gráfico del navegador que incluye el plano del entorno descrito por el usuario en un archivo de texto como una serie de pasillos, habitaciones, espacios abiertos, puertas y demás objetos. La posición del robot se muestra mediante un círculo rojo con una línea indicando la orientación del mismo.

La tarea del navegador es doble: actualizar en cada paso la distribución de probabilidades sobre la posición del robot y calcular la dirección de referencia para el sistema reactivo.

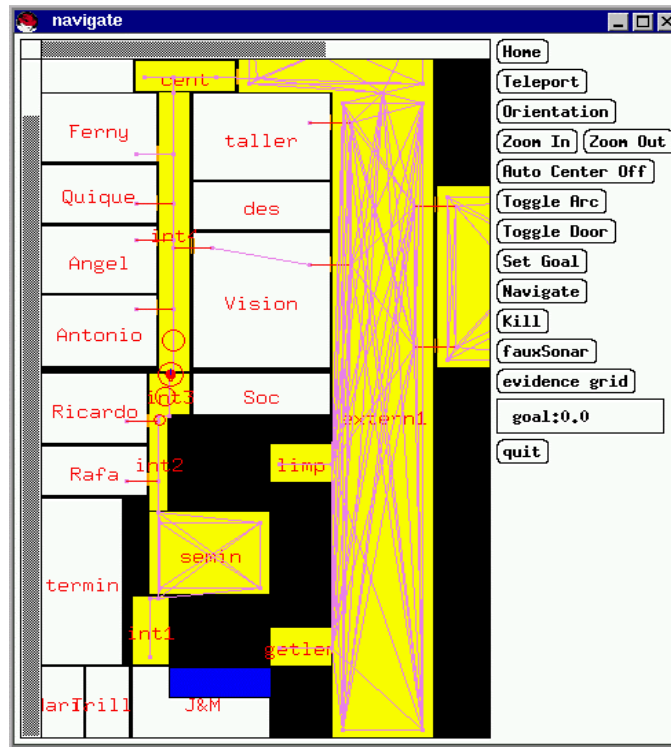


Fig 5.8. Interfaz gráfico del navegador con el plano del Departamento de Ing. Sistemas y Automática(Universidad de Vigo)

En cuanto a la posición, el plano se divide en celdas de un metro cuadrado con cuatro posibles orientaciones correspondientes a los cuatro puntos cardinales. El tiempo también se divide en intervalos de forma que la dirección de referencia es la misma durante todo el intervalo. La posición del robot se almacena como una distribución de probabilidades sobre estas celdas en lugar de mantener su posición

exacta, la cual es desconocida. En la figura 5.8 se pueden apreciar una serie de circunferencias cercanas al robot. Cada una de ellas se encuentra sobre una celda de Markov y representa la probabilidad de ocupación; a mayor probabilidad, mayor es el radio del círculo.

Para actualizar la posición se ha de tener en cuenta la distribución de probabilidades en el instante anterior, las lecturas de los sensores junto con el mapa del entorno, la información odométrica y la dirección de referencia en el instante anterior. La solución adoptada se basa en la construcción de un modelo estadístico del sistema y la actualización del estado usando el teorema de Bayes. La elaboración del modelo es bastante compleja y se estudian con detalle en [Simmons 1995] pero las características generales se resumen a continuación.

Se corresponde con la construcción de un modelo de Markov donde los estados son las posibles posiciones (cuatro por cada celda) en las que se puede encontrar el robot, las acciones son las direcciones enviadas a la capa reactiva (una para cada punto cardinal), y las observaciones son las lecturas de los sensores y odometría. En primer lugar se obtienen las celdas y sus interconexiones a partir del plano mediante un proceso fuera de línea.

Para simplificar el modelo se trabaja con sensores virtuales cuyas lecturas se calculan a partir de la rejilla obtenida por la capa reactiva. Estos sensores caracterizan el entorno cercano al robot en tres posibles estados para cada uno de las posiciones relativas ortogonales (*delante*, *detrás*, *derecha*, *izquierda*). Los estados para cada una de estas direcciones pueden ser: *pared* si hay un obstáculo que lo bloquea completamente, *puerta* si existe un pequeño espacio no ocupado y *libre* si el espacio está completamente libre.

Por otro lado, las lecturas de los sensores virtuales pueden ser *pared*, *espacio-pequeño*, *espacio-medio*, *espacio-grande* o *nada*. El comportamiento de cada sensor virtual queda caracterizado por las probabilidades condicionales de obtener una lectura para cada estado, es decir, $P(\text{lectura/estado})$ para cada lectura y estado posible.

La determinación de la dirección de referencia se calcula a partir de la distribución de probabilidades de posición sobre las celdas y la dirección deseada asociada a cada una de estas celdas. En particular, se escoge la acción asociada a la celda con mayor probabilidad.

Planificador de trayectorias

Su objetivo es la realización de un “plan” que consiste en la asignación de una dirección a cada una de las celdas del plano a partir de las posiciones origen y destino.

Dado que la planificación usando directamente las celdas del plano según el modelo POMDP es intratable por el tamaño del espacio de estados (por cada celda hay cuatro estados), se realiza a distintos niveles mediante refinamientos progresivos [Goodwin 1996]. De todas formas este proceso se lleva a cabo fuera de línea, antes de que el robot comience a moverse se dispone de un plan completo de cómo llegar al objetivo. En esta tarea se usa un plano topológico compuesto por arcos y nodos como se muestra en la figura 5.8. Un nodo representa la unión de dos caminos como por ejemplo entre pasillos, pasillo con oficinas, etc. mientras que los arcos sirven para unir estos nodos indicando que existe un camino entre ellos. Estos arcos tienen asociado un coste que representan la dificultad en ser atravesados. El valor del coste viene dado por la longitud y el tipo de entorno en el que se ubican (es más sencillo para el robot ir a través de un pasillo que cruzar una puerta).

El planificador original usaba técnicas de inteligencia artificial combinadas con medidas obtenidas de teoría de la decisión [Goodwin 1996]. En primer lugar, a partir del plano topológico, se obtenía una serie de trayectorias (secuencia de arcos) con rangos (coste mínimo y máximo) asociados a cada uno. Basándose en estos rangos, se descartan aquellos que, claramente, presentan costes superiores mientras que, para los candidatos restantes, se procede a un proceso de refinamiento del coste asociado, hasta obtener la certeza de que un camino es tan bueno o mejor que los demás.

El planificador utilizado en esta investigación está basado en un algoritmo de búsqueda A^* . En el cálculo de los costes se tienen en cuenta factores que dificultan el camino al robot tales como probabilidad de pasarse de largo un cruce, probabilidad de que una puerta esté cerrada, etc.

A partir del camino topológico, se asocia una dirección a cada celda cercana a dicho camino lo que constituye el *plan* de actuación para llegar a la posición de destino.

Planificador de tareas

Los usuarios solicitan tareas vía “Zephyr” o “Web” de forma asíncrona pudiendo haber varias tareas pendientes en un momento dado. Este planificador determina los lugares a visitar así como el orden de visita. Las tareas se planifican de forma eficiente considerando la manera de alcanzar el mayor número de objetivos (destinos cercanos) y suspendiendo temporalmente tareas si fuese necesario. La realización de este módulo está basada en PRODIGY [Veloso 1995] permitiendo la selección dinámica de objetivos.

5.4 Procesos y comunicación

Las distintas funciones descritas en el apartado anterior se encuentran implementadas en una serie de módulos independientes.

Los más importantes se muestran en la figura 5.9. Cada uno de estos módulos se ejecuta como un proceso en Linux, comunicándose entre ellos a través de TCA (*Task Control Architecture*). La arquitectura TCA [Simmons 1994] y sus herramientas asociadas [Simmons 1997] simplifican la construcción de sistemas de control para robots móviles, y se ha estado empleando y mejorando desde la construcción del robot *Dante* en una docena de proyectos. Permite construir un sistema distribuido sin tener que preocuparse de los mecanismos de comunicación entre los componentes de dicho sistema. Esto es especialmente importante en el caso de robots móviles, debido a que los procesos de control y supervisión están distribuidos entre distintos ordenadores.

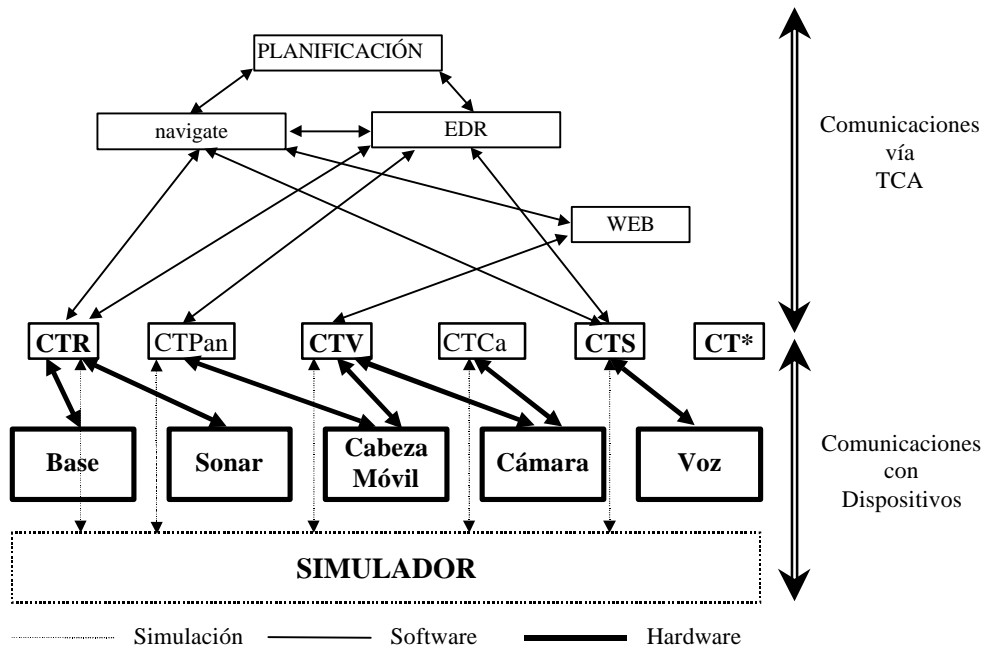


Figura 5.9. Arquitectura software del robot desarrollado en CMU (Xavier)

Función de los distintos procesos

A continuación se describe brevemente la funcionalidad de los distintos módulos presentados en la figura 5.9.

- ✓ **CTR.-** Controla la base, el movimiento del robot, realiza la navegación reactiva (evitación de obstáculos) y responde a peticiones relacionadas con la lectura de los sensores sonar, láser y táctiles. Además, también provee información odométrica acerca de la posición y orientación del robot junto con la tensión de las baterías.
- ✓ **CTPan.-** Controla los movimientos de la cabeza móvil sobre la que está instalada la cámara. Puede girarla horizontalmente y/o verticalmente.
- ✓ **CTCamera.-** Se encarga de controlar la cámara y proporcionar imágenes bajo demanda a otros módulos.
- ✓ **CTV.-** Este módulo engloba el funcionamiento de todo el sistema de visión. Es decir, combina las funciones de los dos módulos anteriores.
- ✓ **CTS.-** Controla el sistema de voz del robot que sirve para emitir mensajes a los usuarios.
- ✓ **CTA.-** Controla el brazo del robot. Este módulo sólo se usa en el robot *Amelia*, dado que *Xavier* actualmente no dispone de ningún brazo articulado.
- ✓ **Navigate.-** Se encarga de realizar las tareas de navegación de acuerdo con el sistema anteriormente descrito.
- ✓ **EDR.-** Este módulo, desarrollado como parte de esta tesis, es el encargado de la monitorización y supervisión. Su funcionamiento detallado se describirá en los siguientes capítulos.

Estos son los módulos a los que se hará referencia más adelante. No obstante, es necesario hacer notar que existen otros módulos como los encargados de la presentación en Internet, sistemas de aprendizaje, etc. que aquí no se detallan al no ser relevantes para el supervisor.

5.4.1 Arquitectura de control de tareas

Cada uno de los módulos se ejecuta como un proceso independiente bajo Linux y la comunicación entre ellos se realiza usando TCA (Task Control Architecture) [Simmons 1994]. TCA es más que un simple módulo de comunicación entre procesos, ya que provee una estructura y utilidades para la planificación, asignación de recursos, especificación de relaciones temporales, establecimiento de monitores y gestión de excepciones. Está especialmente diseñado para simplificar la

tarea de creación de sistemas de robots móviles complejos que combinan control reactivo y deliberativo.

En cuanto a la faceta de comunicaciones, TCA provee un proceso (*central*) con el cual se deben registrar todos los que deseen comunicarse usando esta filosofía. Nada más arrancar, cada uno de los procesos implicados en esta arquitectura tiene que comunicar al *central* los servicios que proveen. Todos los mensajes se pasan a través del *central* quien se encarga de enviarlos al proceso que gestiona esos recursos. Estos mensajes pueden ser de diversa índole (peticiones, órdenes, monitores, objetivos, informes y excepciones) y pueden ser servicios proporcionados por otros procesos, o bien directamente por el *central* como los destinados a establecer “espías” para detectar y tratar irregularidades en el funcionamiento del sistema.

Con respecto a la tarea de planificación, permite establecer restricciones entre mensajes, soportando una descomposición jerárquica, secuenciamiento y sincronización de tareas, gestión de recursos, monitorización y gestión de excepciones. Las tareas se organizan en una estructura jerárquica en forma de árbol, permitiendo la división de una tarea en subtareas y el establecimiento de restricciones entre éstas. Este árbol se construye de forma dinámica a medida que las tareas se van realizando pueden crear otras e incluso cancelar ciertas ramas del árbol.

Entre otras arquitecturas similares a la aquí descrita cabe mencionar también RAPs (Reactive Action Packages) [Firby 1989]. La principal diferencia es que, aunque dentro de la clasificación reactivo/deliberativo son las dos mixtas, TCA tiende a ser más deliberativa que RAPs dado que esta última se preocupa más del tiempo de respuesta (tiempo real) mientras que en la primera se ha puesto especial hincapié en el tema de gestión de tareas complejas, división e interacción entre ellas.

La comunicación entre los procesos y los dispositivos depende del tipo de dispositivo. Cuando se utiliza el simulador, las conexiones se realizan con éste a través de “sockets” y los mensajes son los mismos que se intercambiarían con los dispositivos.

6 Modelo determinista

6.1 Introducción

Una de las cualidades buscadas en los robots móviles es un alto grado de autonomía. Como ya se ha comentado en capítulos anteriores, en la mayor parte de las aplicaciones se procura que éstos permanezcan operativos durante un largo período de tiempo sin intervención externa. El objetivo de los sistemas de supervisión, además de la monitorización del sistema, es tomar las decisiones oportunas para mantener el robot funcionando correctamente en caso de fallos. Para que esto sea posible es necesario un sistema de navegación muy robusto que funcione ante cualquier situación. Es, por tanto, de vital importancia incluir un mecanismo de tratamiento de fallos.

En este capítulo se presenta una primera solución utilizando un sistema de monitores estructurado en capas de menor a mayor especificidad. El objetivo es proveer cobertura para el mayor número posible de situaciones de excepción, es decir, aquellas que no pueden ser gestionadas de forma estándar por el sistema de navegación. Los monitores situados en las capas superiores son muy generales y detectan un rango muy amplio de situaciones de excepción, mientras los situados en las capas inferiores son más específicos. Por ser más generales, los monitores de las capas superiores suelen ser más lentos en la detección, mientras que los monitores específicos de las capas inferiores son mucho más rápidos y están enfocados en un problema concreto. En la fase de diseño se puede comenzar con los monitores más generales y, a medida que se va obteniendo experiencia e información acerca de problemas específicos, ir realizando monitores para éstos.

El sistema supervisor aquí presentado posee tiempos de respuesta muy cortos, dado que la fase de diagnóstico global es muy sencilla. En la fase de detección se usa una serie de monitores donde cada uno de ellos constituye un sistema de extracción de síntomas.

En el siguiente apartado se introducen los conceptos básicos en los que se asienta esta estructura, en particular, la división del espacio de estados en situaciones

de excepción y situaciones nominales. También se explica la división de monitores y su efecto en el espacio de estados. El tercer apartado de este capítulo está dedicado a la realización de los monitores que componen el modelo y la forma en que estos obtienen los síntomas a partir de los restos. En el penúltimo apartado se describe un sencillo sistema de diagnosis suponiendo los monitores ideales, dejando el tratamiento de la incertidumbre para el modelo más elaborado del capítulo siguiente.

6.2 Situaciones de excepción

Una de las características diferenciales del tratamiento de errores en robótica móvil es el tipo de errores con los que se trata. Aunque algunos autores, como se ha visto en el capítulo anterior [Ferrell 1994], se ocupan solamente de errores propiamente dichos, en otros [Stuck 1995] se extiende a situaciones de excepción, considerando como tales aquellas situaciones no tenidas en cuenta a la hora del diseño y al sistema como un supervisor. A lo largo de esta sección se explica en detalle dicho concepto.

Ante la imposibilidad de tener en cuenta todas las situaciones en las que el agente se puede encontrar, se hace necesario un mecanismo para la gestión de situaciones inesperadas. El origen de tales situaciones puede ser:

- ✓ Fallos propiamente dichos. Pueden ser fallos en algún dispositivo como los tratados en [Stergios 1998b] [Ferrell 1994] o fallos en el sistema de navegación [Noreils 1990][Sankaran 1977].
- ✓ Situaciones no contempladas en el desarrollo del sistema de navegación que no pueden ser consideradas como fallos. El sistema funciona de acuerdo a las expectativas, pero el problema surge porque en la elaboración del sistema de navegación no se han contemplado estas situaciones.

Con el siguiente ejemplo sencillo se pretende aclarar la diferencia entre ambos tipos de situaciones:

Supongamos que un robot móvil para interiores utiliza un sistema de navegación jerárquico con dos capas. En la capa superior un planificador calcula la trayectoria a seguir dados los puntos origen y destino. Dicha planificación la realiza partiendo de un plano del edificio en el que se mueve. En un nivel inferior existe un módulo reactivo que se encarga de seguir la trayectoria determinada por el planificador, evitando obstáculos que encuentra en el camino. La información entre los

módulos se produce en una única dirección. Es decir, no hay flujo de información del nivel reactivo al de planificación. Mientras el robot trata de seguir una trayectoria a lo largo de un pasillo se pueden producir diferentes situaciones:

- ✓ Situación 1: Puede ser que el robot no sea capaz de seguir la trayectoria porque un fallo en los sensores le indican que hay un objeto en el camino y pretenda evitarlo por lo cual no podrá avanzar.
- ✓ Situación 2: Puede ser que el pasillo esté realmente bloqueado y el nivel reactivo intente bordearlo. Al encontrarse en un pasillo cerrado no saldrá de esta situación a menos que el planificador le indique un camino alternativo, pero como el flujo de información se realiza en una única dirección, la trayectoria será siempre la misma.

La primera situación puede considerarse que existe un fallo debido a que un componente no funciona de acuerdo a las especificaciones, mientras que en el segundo caso no se puede decir que haya un fallo aunque el sistema no responda según lo esperado. El sistema de navegación en la segunda situación funciona de forma correcta, al igual que el sistema sensorial y demás partes del robot. El problema, en este caso, es que el robot se ha encontrado en una situación que no se había tenido en cuenta en la fase de diseño.

En lo que sigue se utilizará el término *excepciones o situaciones de excepción* para referirse tanto a situaciones de fallo como situaciones no contempladas. Para que el robot opere de forma autónoma y continuada se han de tener en cuenta ambas. El sistema de detección y diagnóstico de fallos será considerado, a partir de ahora, como un sistema de detección y diagnóstico de excepciones.

Como se ha indicado en el capítulo cuarto, la mayoría de los esfuerzos en el campo de la detección de fallos se han enfocado al problema de la generación de residuos, argumentando que la parte de decisión basada en residuos bien diseñados es relativamente fácil [Chen 1998]. Sin embargo, resulta muy difícil modelar matemáticamente el comportamiento global del robot como se hace en otros sistemas, pero sí se puede modelar el funcionamiento de algunos dispositivos o módulos para detectar fallos locales. En ese caso se puede aplicar alguna de las técnicas descritas en el capítulo anterior para extraer síntomas.

Tanto el modelo determinista presentado en este capítulo, como el estocástico presentado en el siguiente capítulo tratan los fallos desde un punto de vista general y modular, de forma que es posible añadir nuevos síntomas y mecanismos de recuperación de forma incremental, a medida que se van detectando nuevas situaciones de excepción. La extracción de síntomas aquí presentada se aplicará al funcionamiento

general del robot, pues ésta es la característica diferenciada con respecto a otro tipo de procesos. No obstante, el objetivo final es la construcción de una arquitectura que permita tratar las excepciones de forma general e independiente de su naturaleza y de la forma de obtener los síntomas.

6.2.1 Espacio de estados

Se puede representar simbólicamente el conjunto de todos los estados del robot como en la Fig. 6.1. En dicha figura, el espacio de estados del robot está dividido en dos subespacios: El subespacio de *excepciones*, que engloba aquellas situaciones en las cuales el robot no llegará a acabar la tarea encomendada o bien no lo hará en el tiempo requerido; y el subespacio *nominal*, que incluye el resto de las situaciones en las cuales el robot, de seguir así, realizará la tarea con éxito. Algunos de los estados de excepción (representados por cuadrados vacíos) son conocidos de antemano, mientras que existen otras de estas situaciones (representados por cuadrados sombreados) que son totalmente desconocidas. Los estados nominales se representan con círculos.

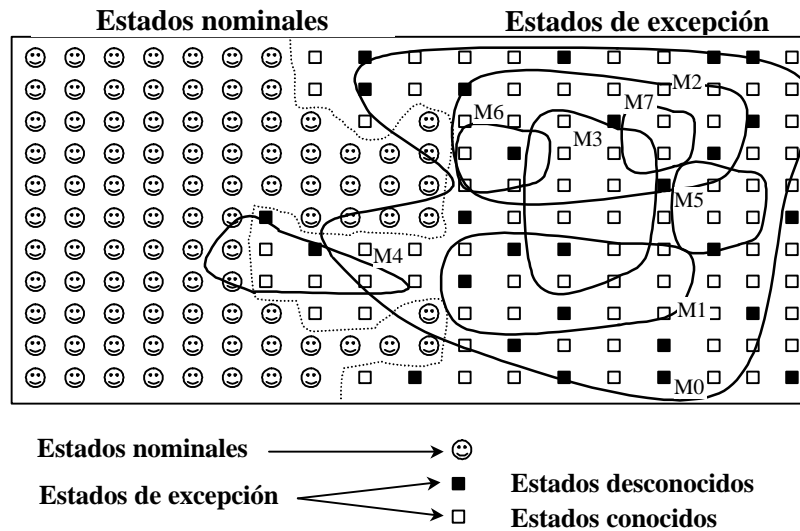


Figura 6.1 División del espacio de estados. Monitores

A pesar de que conceptualmente la separación entre ambos subespacios está bien definida, en la práctica no es tarea nada sencilla la realización de un sistema que detecte tales situaciones por diversas razones:

- ✓ La información de que dispone el robot a través de los sensores es una información parcial de la realidad.
- ✓ Los datos de los sensores presentan ruido asociado a la información.
- ✓ Dependiendo de las condiciones del entorno, las lecturas que en un momento indican una situación de bloqueo, en otro instante puede no serlo.
- ✓ Puede haber un número muy grande de excepciones, la mayoría de las cuales son desconocidas en el momento de diseño del supervisor.

La solución planteada en esta tesis se basa en la realización de una arquitectura de monitores organizados jerárquicamente. Cada uno de estos monitores es un módulo que supervisa el comportamiento del robot buscando síntomas o indicios de error en el robot. Un síntoma es una manifestación de malas prestaciones o valores de los residuos fuera de los márgenes permitidos.

Hay que tener en cuenta que, en función del criterio a usar como medida de prestaciones o los residuos a utilizar, se obtendrán diferentes síntomas. Por ejemplo, si la tarea consiste en la navegación de una posición a otra, la distancia al objetivo podría ser un parámetro a tener en cuenta como evolución hacia la consecución del objetivo aunque, como se verá más adelante, no es siempre una buena referencia.

Algunos síntomas consisten en detectar determinados tipos de comportamientos. Por ejemplo, no suele ser normal que el robot se ponga a dar vueltas sobre sí mismo sin desplazarse de sitio. Otros síntomas se basan en determinadas situaciones o posiciones en el entorno. Así, siguiendo con la tarea de navegación, se sabe de antemano que, en posiciones del entorno con determinadas características, el robot se puede quedar atrapado en un ciclo repetitivo de acciones provocando que éste nunca alcance el objetivo.

Los monitores empleados en supervisión detectan generalmente situaciones de excepción, pero algunas veces también se pueden disparar ante situaciones nominales.

Es bastante difícil encontrar un síntoma que permita discernir las situaciones de excepción de las nominales para todas las tareas posibles en un tiempo razonable; por el contrario, es más sencillo detectar situaciones concretas en tiempos relativamente cortos. Por ejemplo, si se activa un monitor para controlar el tiempo de ejecución de las tareas, este monitor es aplicable a todas ellas, ya que se puede

determinar un tiempo máximo necesario para finalizar cada tarea y, por tanto, el monitor simplemente se dispararía cuando el tiempo empleado en la realización de la tarea supere este máximo. Sin embargo, volviendo al ejemplo de navegación, está claro que el tiempo empleado por el agente móvil en llegar a cumplir el objetivo especificado depende de la gente que haya en los pasillos, el estado de las baterías, etc. Por lo tanto, es necesario establecer un margen de tiempo suficiente para evitar que el monitor se dispare en una situación nominal. Esto implica, por ejemplo, que en el caso de que el robot se bloquee al principio del trayecto, dicho problema será detectado con bastante retraso si únicamente se utiliza un monitor de tiempo. No obstante, se puede desarrollar un monitor diferente que controle otros parámetros como la carga de la batería, que sería activado cuando su nivel se sitúe por debajo de un umbral determinado, resolviendo así un problema muy específico. Este monitor posee la ventaja de que la detección de ese problema es casi inmediata. También es importante señalar que esta situación sería detectada finalmente por el monitor temporal, al bajar la batería a un nivel insuficiente para mover el robot, pero este último monitor sólo daría la señal de alarma al sobrepasar el tiempo máximo estimado para finalizar la tarea en curso.

Es importante resaltar que cada monitor está asociado con un síntoma o un conjunto de síntomas. Mediante esos síntomas el monitor podrá detectar un conjunto de situaciones excepcionales, como se muestra en la figura 6.1, donde los subconjuntos ***M0***, ***M1***,... representan los estados bajo los cuales los monitores ***m0***, ***m1***,... son disparados. Se trata de evitar, en lo posible, que los monitores se disparen ante estados nominales, como en el caso de ***m0*** y ***m4*** (figura 6.1). Al mismo tiempo, interesa que un monitor abarque el mayor número de situaciones de excepción posibles. Algunas de tales situaciones son detectadas por varios monitores, llegando a casos, como en el de ***m6*** y ***m7***, donde todas las excepciones detectadas por ambos son también detectadas por ***M2***.

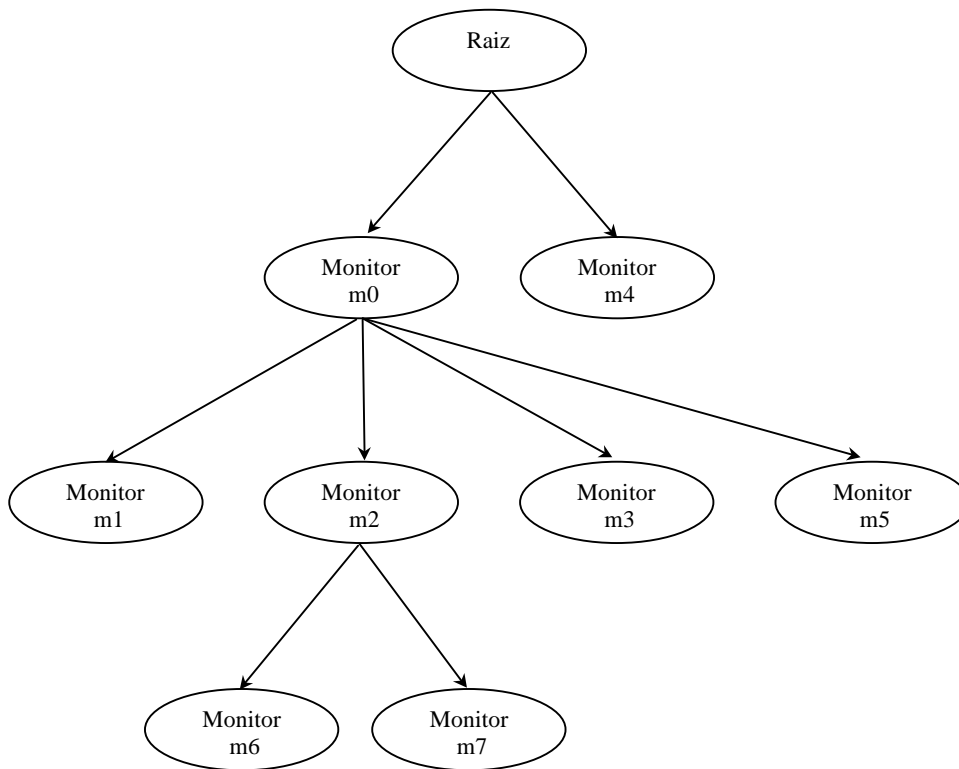


Figura 6.2 Estructura de monitores para el espacio representado en la figura 6.1

Sin embargo, la existencia de estos monitores tiene su justificación por dos razones:

- ✓ Como se veía anteriormente, los monitores más específicos (**m6** y **m7**) suelen detectar las excepciones en menor tiempo.
- ✓ Los monitores más específicos proporcionan más información del problema. Un disparo de dicho monitor implica una mayor delimitación del problema. De esta forma, se acota el número de fuentes de error de cara al proceso de diagnosis.

Por todo ello, el conjunto de monitores se ha organizado en una estructura en forma de árbol (fig 6.2 y 6.3). Los monitores situados en las capas superiores (cerca de la raíz) son los más generales: Detectan un amplio rango de excepciones y son aplicables a un amplio abanico de tareas. Los monitores situados en la capa inferior (hojas del árbol) son los monitores más específicos. Puede haber subconjuntos de excepciones detectadas por dos monitores situados en distintas ramas, pero todas las excepciones detectadas por un monitor han de ser detectadas por aquel del que desciende. En esta estructura, los monitores situados en las hojas del árbol, tienen prioridad sobre los situados en la raíz en caso de ser disparados al mismo tiempo. Otra ventaja de esta estructura se refiere a la adición de nuevos monitores: en un principio se realizan los monitores más globales, dado que no se puede prever todos los problemas y, a medida que se detectan nuevas excepciones, se van añadiendo monitores más específicos al sistema para detectar esas nuevas situaciones.

6.3 Extracción de síntomas (monitores)

Los monitores se han implementado como una serie de módulos independientes que usan TCA, para coordinar su ejecución junto con las tareas de navegación. Dado que todas las tareas se comunican mediante mensajes pasados a través del proceso central de TCA, es posible solicitar a dicho módulo que envíe una copia de determinado tipo de mensajes, o hacer otro tipo de solicitudes, tal como se muestra en [Simmons 1997c].

En el ámbito de esta Tesis se usan los siguientes términos para representar el estado de un monitor:

- ✓ Un monitor está activo cuando se encuentra comprobando de forma continua las situaciones o residuos para las que fue diseñado.
- ✓ Un monitor se dispara cuando algún residuo bajo su control excede los márgenes permitidos o detecta algún otro tipo de síntoma. Cuando esto se produce, se informa al módulo de identificación y recuperación.

En la figura 6.3 se muestra el árbol correspondiente a los monitores desarrollados. A continuación se describe la realización de cada uno de ellos, de acuerdo con los parámetros y residuos asociados, así como las excepciones que detectan.

6.3.1 Monitorización del tiempo

El monitor más global en esta estructura se basa en el tiempo máximo asignado a cada tarea. El monitor se dispara durante la ejecución de una tarea si el tiempo transcurrido desde el comienzo de ésta es mayor que el máximo permitido. La información que provee con vistas a la recuperación del posible error es casi nula, pues lo único que indica es que la tarea no se realizó en el tiempo esperado.

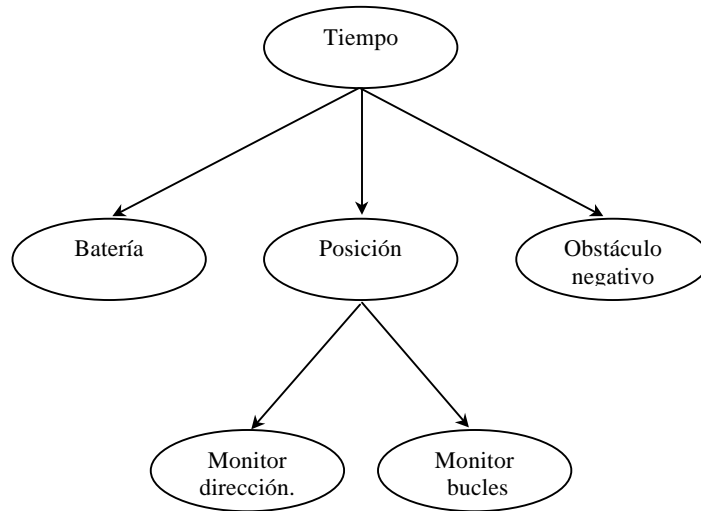


Figura 6.3 Arquitectura de monitores utilizada

La realización de este monitor es bastante sencilla. Se trata de estimar el tiempo máximo que el robot necesita para realizar la tarea y, a continuación, disparar el monitor una vez transcurrido ese tiempo, si la tarea no ha finalizado.

La ventaja de este monitor es que, por ser global, se puede aplicar a todas las tareas y permite detectar un amplio número de excepciones. Esto presenta ciertas desventajas que son apreciables en el ejemplo de la tarea de navegación, que es común a todo agente móvil. En primer lugar, es muy difícil hacer una estimación exacta del tiempo máximo que el robot emplea en ir de una posición a otra ya que depende de múltiples factores. El problema surge por la elevada variación en los tiempos empleados en el seguimiento de la misma trayectoria [Haigh 1998] ocasionada, entre

otras razones, por condiciones del entorno, como pueden ser, el número de obstáculos no modelados con los que el robot se puede encontrar o las condiciones internas al robot como la carga de las baterías. Para que el monitor no se dispare en una situación nominal es necesario tomar como referencia la situación más desfavorable. Este tiempo suele ser elevado, acarreando un retraso en la detección de la posible situación de excepción. Es evidente que, este retraso será aun mayor si el recorrido que ha de hacer es largo y el problema ocurre al comienzo del recorrido. Como veremos más adelante, este problema se puede evitar haciendo un seguimiento más exhaustivo de la navegación del robot, pero estaremos perdiendo generalidad al convertirse en un monitor particular para la tarea de navegación.

La práctica en la puesta a punto del sistema de supervisión nos ha conducido a combinar este monitor con otros más específicos, dado que sigue resultando un buen indicador para aquellas situaciones de excepción no contempladas si la excepción no está incluida en el ámbito de ninguno de los específicos.

Para la estimación del tiempo empleado por las tareas de navegación se usa información del planificador. El tiempo necesario para recorrer una determinada trayectoria compuesta por una serie de arcos es función de la longitud de éstos y de un parámetro que refleja la medida de dificultad en recorrer cada arco. La velocidad del robot es función de la complejidad del entorno. Por ejemplo, navegar a través de un pasillo suficientemente ancho y despejado de obstáculos es mucho más fácil y el robot puede incrementar su velocidad, mientras que, cuando tiene que atravesar una puerta, debe reducir la velocidad. En el plano topológico esta situación se ha reflejado aumentando los pesos de los arcos asociados a aquellas zonas que suponen mayor dificultad para la navegación. En el futuro estos pesos se podrían obtener mediante métodos de aprendizaje similares a los descritos en [Haigh 1998] o [Diéguez 1998].

6.3.2 Seguimiento de la posición

El monitor que usa este síntoma está ligado a la tarea de navegación, tratándose de un monitor más específico que en el caso anterior.

Una medida que indica el avance del robot a lo largo de la ruta planeada es la distancia entre la posición real del robot y la posición en la cual el robot debería estar si todo ocurriese según lo previsto. Para ello, se activa un monitor que periódicamente (cada segundo) calcula la diferencia entre la posición *esperada* (posición donde el robot debería de estar) y la posición real.

Para calcular la posición *esperada* (S) del robot, se supone que el robot recorre todo el arco (trozo de camino planeado con iguales características) con una velocidad constante, sin tener en cuenta diversos factores que fuerzan al controlador reactivo a

reducirla como pueden ser objetos no modelados con los que el robot se puede encontrar.

Por otra parte, es necesario conocer la posición real del robot. El sistema de navegación basado en un modelo POMDP (capítulo 5) no dispone de la posición exacta del robot, sino de una distribución de probabilidades $P(s_i)$ sobre los estados de Markov asociados a posiciones del plano S_i . Usando ambas, la posición esperada y la distribución de probabilidades, se calcula una medida de error denominada retraso como:

$$retraso = \sum_{i=1}^n \frac{P_i}{\sum_{j=1}^n P_j} \times d(S_i, S) \quad [Eq. 6.1]$$

donde $d(c, S)$ es la distancia Manhattan entre S_i y S , mientras que n es el número de nodos que poseen una probabilidad de ocupación relevante. Esto es, en lugar de seleccionar todos los nodos con probabilidad mayor que cero, se seleccionan los que superan un umbral muy bajo para reducir el tiempo de procesado. Ésta es la razón por la cual aparece la división por:

$$\sum_{j=1}^n P_j \quad [Eq. 6.2]$$

Identificando términos con los métodos de detección descritos en el cuarto capítulo, el síntoma obtenido en este monitor es mediante simulación en lugar del modelo exacto utilizado entonces. El retraso es un residuo, pues no es más que la desviación entre una medida del sistema (posición del robot) y una medida obtenida del modelo (posición estimada mediante simulación). Hay una serie de hechos que hacen que el valor de este parámetro fluctúe, algunos de los cuales se tratan en el siguiente ejemplo práctico.

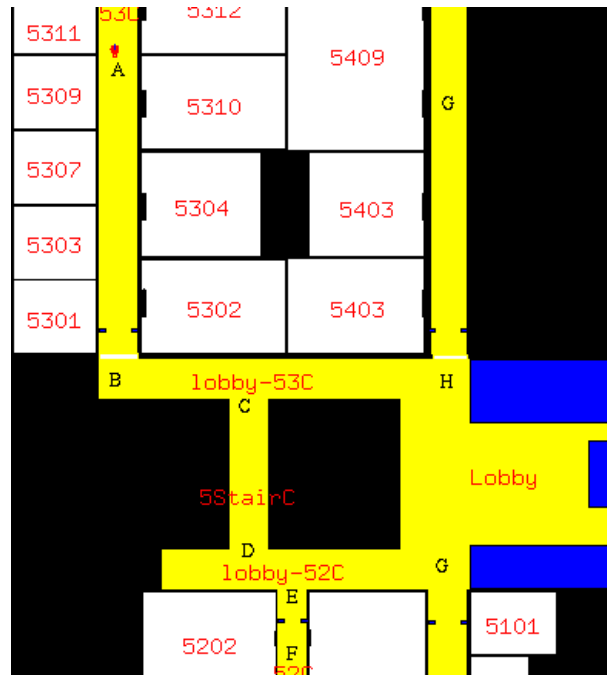


Figura 6.4 Mapa de una parte de la 5ª planta del edificio Wean Hall en la universidad de Carnegie Mellon.

La figura 6.5 representa los valores obtenidos del retraso mientras el robot (Xavier, en este caso) navega desde la posición *A* a la posición *F* en el mapa representado en la figura 6.4. El mapa se corresponde con una parte de la quinta planta del edificio de *Computer Science* de la *Universidad de Carnegie Mellon*. El camino a seguir es *A-B-C-D-E-F*. La figura 6.6 representa otro ejemplo, pero en este caso el robot se encuentra el camino bloqueado. La línea continua representa valores instantáneos, mientras que la línea discontinua representa la media calculada con una ventana de diez muestras (diez segundos).

Una característica común a todos los experimentos realizados y que se observa en las figuras 6.5 y 6.6 son los picos que aparecen cada dos muestras debidos a que la periodicidad del monitor de posición es de un segundo, mientras que la periodicidad

del módulo que actualiza la posición es aproximadamente el doble. Ello provoca que, aunque la posición esperada se actualice en cada muestra, la distribución de probabilidades de ocupación sólo lo hace cada dos y, por tanto, el error se incrementa en aquellas muestras en las que la posición del robot (distribución de probabilidades) no se actualiza, mientras que no ocurre en las de simulación.

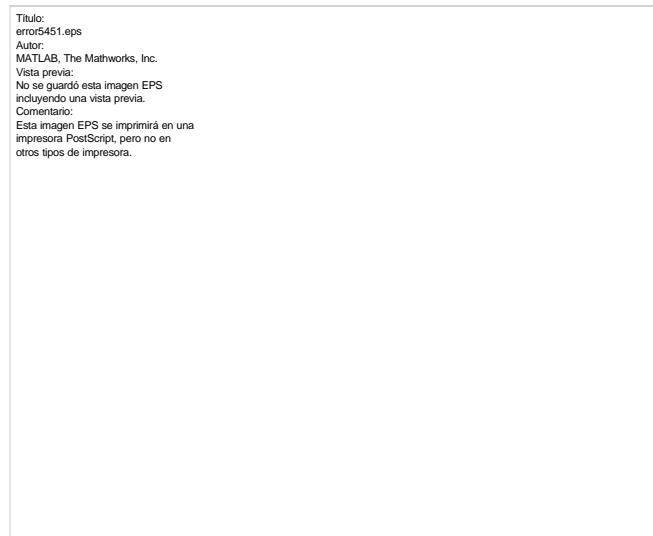


Figura 6.5 Retardo registrado mientras el robot navega de A a F en el mapa de la fig 6.4

Otra característica de este parámetro es su variabilidad. Existen diversas causas que la provocan²:

El sistema de navegación basado en POMDP obtiene información acerca de la posición de los sensores de odometría que proporcionan la posición y orientación. Sin embargo, como ya se sabe, la precisión de esta información suele ser insuficiente para saber la posición exacta del robot y el sistema de navegación usa también información

² Hay también que considerar que el espacio se divide en celdas de un metro cuadrado y por cada celda existen cuatro nodos de Markov correspondiendo a distintas orientaciones del robot.

proveniente del sonar y del láser. Esta información se procesa de forma que el modelo trabaja con sensores virtuales. En [Simmons 1995] se describe con detalle el sistema de navegación utilizando el modelo POMDP. Estos sensores caracterizan el entorno cercano al robot en tres posibles estados para cada uno de los posiciones relativas ortogonales (delante, detrás, derecha, izquierda).

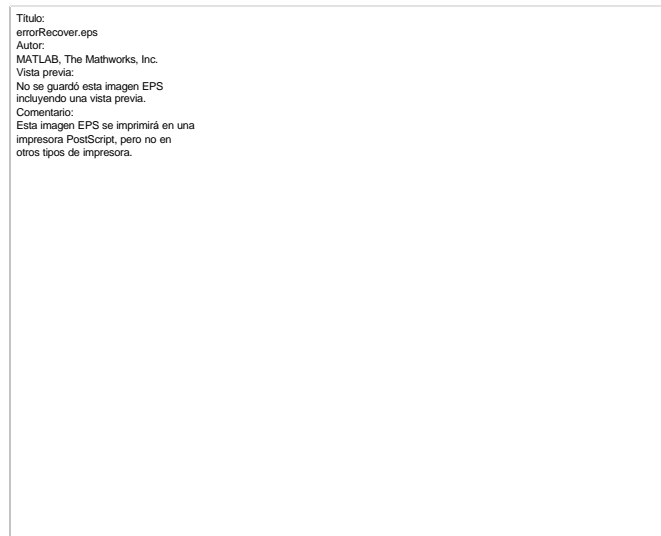


Figura 6.6 Retardo siguiendo trayectoria A-B-C-D-E-F con C-D bloqueado y replanificación C-H-G-E-F

Los estados para cada una de estas direcciones pueden ser:

- ✓ “pared” si hay un obstáculo que lo bloquea completamente.
- ✓ “puerta” si hay un espacio libre.
- ✓ “libre” si no hay nada.

Cuando el robot navega en un entorno uniforme (por ejemplo, en un pasillo largo sin puertas o en un espacio abierto) usa básicamente información de los sensores de odometría, dado que el resto de los sensores no le proporcionan información relevante para determinar su posición (lecturas similares entre celdas contiguas). Esta

falta de referencias ocasiona que la distribución de probabilidades se disperse entre más y más nodos de Markov hasta que encuentre un punto con una característica distintiva (puerta, cruce de pasillos) y eleve la probabilidad de ese nodo a un valor mucho más alto que los demás. Este comportamiento, aunque deseado en navegación, produce variaciones en el retraso (ecuación 6.1).

Otra causa de las fluctuaciones de este parámetro se debe a la variabilidad del entorno. En un edificio de oficinas real, como el que se ha usado en este trabajo, existen muchos objetos (sillas, mesas) y personas que el robot tiene que bordearlos disminuyendo su velocidad.

Los niveles de la batería y algunos problemas mecánicos también pueden provocar variaciones en la velocidad del robot.

Como consecuencia de todos estos factores, el retraso se acumula a medida que el robot recorre la trayectoria calculada por el planificador. Estas variaciones impiden usar el valor del retraso directamente como un síntoma, dado que es imposible establecer el umbral de dicho parámetro. Es posible, no obstante, usar la derivada o incremento del retraso, evitando así los errores acumulados durante la navegación. En definitiva, se trata de usar como síntoma un determinado valor de la variación del retraso, se busca una situación en la que, durante un tiempo determinado ($TMAX$), el retraso se incrementa con una velocidad muy cercana a la velocidad con la que el robot debería circular. De esta forma, este monitor sólo se dispara cuando el robot no hace ningún progreso durante el tiempo $TMAX$ que ha sido calculado de forma experimental. La selección de ese tiempo supone establecer un compromiso entre velocidad de detección de la excepción y la eliminación de falsas detecciones. Si se selecciona un tiempo muy pequeño se puede disparar el monitor para una situación nominal, pero las situaciones excepcionales se detectarán más rápidamente. En nuestro caso, se ha asignado un valor bastante alto (20seg) para evitar falsas detecciones.

En la figura 6.6 se representan los valores del retardo (residuo calculado de acuerdo a la ecuación 6.1) obtenidos mientras el robot se encuentra siguiendo una trayectoria bloqueada por obstáculos que no se pueden evitar. El entorno es el mismo que el utilizado anteriormente (figura 6.4) y la trayectoria inicial es la que une los puntos **A-B-C-D-E-F**. El obstáculo que impide pasar al robot se encuentra en el tramo que une los puntos **C** y **D**. Cuando el robot se encuentra en este tramo (segundo 58 de la gráfica 6.6), el retardo se empieza a incrementar a una velocidad igual a la esperada del robot en ese tramo, velocidad a la que el robot debería moverse mientras está intentando evitar el obstáculo. Sobre el segundo 74 se dispara el monitor de posición y se calcula una ruta alternativa (**C-H-G-E-F**).

Algunas veces, debido a que el robot avanza muy despacio, la posición estimada alcanza el destino antes que el robot. A partir de este momento el monitor cambiará de referencia de disparo y en lugar de utilizar una pendiente en el retraso igual a la velocidad esperada, usará pendiente cero³.

6.3.3 Seguimiento de vueltas

Aunque no es muy común y aún no sabemos muy bien la razón de este comportamiento (se sospecha que se debe a la respuesta del algoritmo reactivo ante determinadas formas del entorno), algunas veces el robot permanece dando vueltas en el mismo sitio sin desplazarse en absoluto. Para detectar este problema se ha desarrollado un monitor que sólo detecta este tipo de excepciones siendo por tanto uno de los más específicos que, de momento, se han realizado.

El funcionamiento de este monitor se basa en controlar el número de vueltas que el robot da en un mismo sitio. Este problema también podía ser detectado por el monitor analizado en el apartado anterior, que usa como síntoma el retardo de posición del robot. El motivo por el cual se ha desarrollado este monitor es que, por una parte es más rápida la detección del problema y, por otra parte, al dispararse este monitor, se dispone de más información acerca de la situación y es posible establecer un mecanismo de recuperación específico para este problema.

6.3.4 Detección negativa de intersecciones

En algunas intersecciones como la que se ilustra en la figura 6.7, el robot no detecta la intersección y permanece dando vueltas de forma indefinida. En este ejemplo, al pasillo por donde se desplaza el robot desemboca otro pasillo (por la derecha en este caso) por el cual debe pasar. La trayectoria inicial calculada por el planificador es aquella que une los puntos A-B-C. Sin embargo, el sistema sensorial del robot no detecta el cruce de pasillos y continua hasta el punto D donde existen evidencias (odometría) que ha ido demasiado lejos y vuelve hacia atrás. En su camino de retorno ocurre lo mismo y se acerca al punto A. Esta situación puede repetirse indefinidamente, aunque eventualmente el robot puede llegar a localizar la intersección y salir del problema.

³ De no hacerlo así, no se detectarían los bloqueos ocurridos en el último tramo del camino.

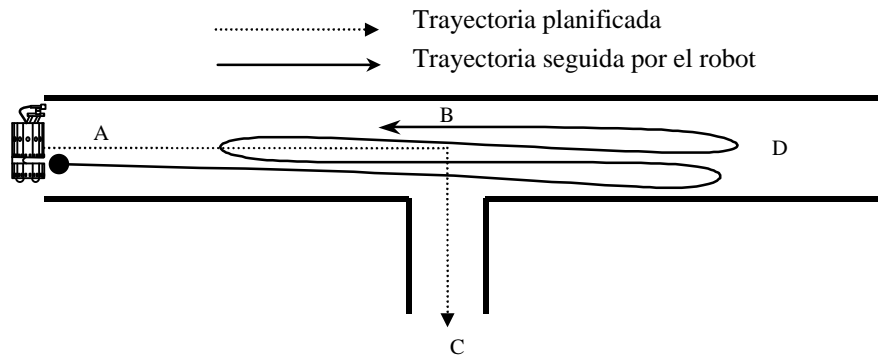


Figura 6.7 El robot tiene que seguir la trayectoria A-B-C pero repetidamente no ve la intersección con lo cual sigue hasta D donde se da cuenta que ha llegado demasiado lejos.

Para detectar dicha situación, el monitor solicita al TCA una copia de todos los mensajes que implican una rotación del robot de 180 grados, activando un espía (*tap*). Este espía es una utilidad de TCA que permite al módulo que lo establece recibir una copia de determinados tipos de mensajes.

6.3.5 Carga de la batería

Este monitor, al contrario del resto de los analizados hasta el momento, trata de predecir un posible problema, en lugar de detectarlo una vez que se produzca. Se trata de evitar la situación en la cual el robot se quede sin batería suficiente para volver a la estación de recarga, que en un principio, es el laboratorio. Este monitor, por lo tanto, ha de disponer de datos suficientes para predecir la descarga de las baterías a lo largo del tiempo, así como de los factores que más influyen en dicho proceso. La mayor parte de la energía eléctrica proporcionada por las baterías es consumida por los motores de la base para realizar los desplazamientos y giros, como se puede apreciar en las figuras 6.8 y 6.9. que representan datos en dos situaciones diferentes. En la primera (sin carga), el robot permanece quieto hasta que se queda sin baterías y, en la segunda el robot se mantiene en movimiento hasta quedarse también sin carga. Con las baterías completamente cargadas, el voltaje máximo son 54 voltios y empíricamente se sabe que Xavier comienza a tener problemas cuando la tensión desciende por debajo de los 32 voltios.

De la comparación entre las figuras 6.8 y 6.9 se deduce que un parámetro fundamental a la hora de predecir la descarga de las baterías es el movimiento del robot. Para determinar el nivel de carga al finalizar la tarea actual es necesario calcular el tiempo que el robot necesita para realizarla y estimar qué parte de este tiempo el robot se mantendrá en movimiento. A partir de esos tiempos, con las curvas de las figuras 6.8 y 6.9 se puede estimar la carga final.

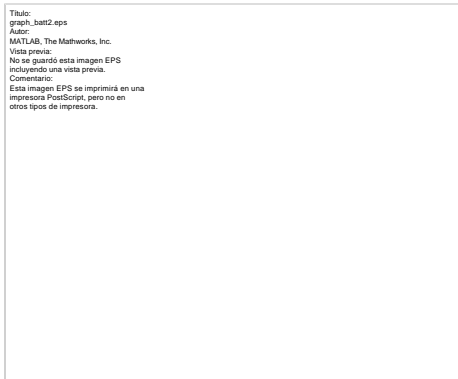


Figura 6.8 Evolución de la descarga de las baterías nuevas con el robot moviéndose y con el robot parado.

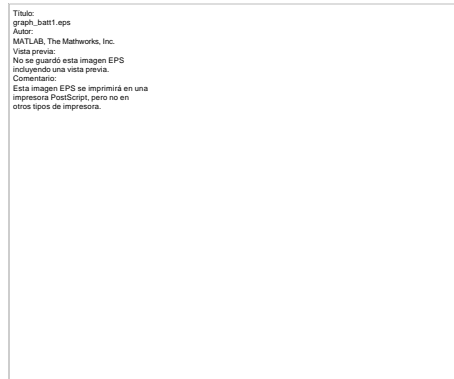


Figura 6.9 Evolución de la descarga de las baterías viejas con el robot moviéndose y con el robot parado.

Por distintas razones entre las que destacan la variabilidad del entorno, como se ha visto en el apartado 6.3.2, es muy difícil predecir los tiempos para realizar la tarea de seguimiento de una trayectoria. Por lo tanto, en lugar de estimar el nivel de carga de las baterías al final de dicha tarea, se calcula un rango de valores (representados por un máximo y mínimo) calculado a partir de un intervalo de tiempos.

Se usa información del planificador [Goodwin 1996] para estimar los tiempos de ejecución de las tareas de navegación.

Este monitor sólo empieza a funcionar cuando el nivel de la batería se reduce por debajo de un determinado valor, con el fin de no sobrecargar el sistema. Una vez alcanzado dicho nivel, se mide periódicamente la tensión de la batería para comprobar si tiene carga suficiente para finalizar la tarea actual y regresar al laboratorio. Para determinar si es suficiente, primero se mide la carga de la batería. A continuación,

usando información del planificador, se calculan los rangos de tiempo estimados para finalizar la tarea y volver al laboratorio (t_{mov} , $t_{vacío}$) y, por último, con estos datos y las curvas de descarga de las baterías, se determina el nivel de tensión que tendría la batería al finalizar la tarea y volver al laboratorio ($nivel_{final}$).

Hay que resaltar que t_{mov} , $t_{vacío}$ y, por tanto, $nivel_{final}$ son intervalos con un valor máximo, medio y mínimo, en lugar de valores fijos. Dependiendo de la ubicación de los valores máximo, mínimo y medio del $nivel_{final}$ se clasifica el estado de las baterías en uno de los cuatro posibles estados, de acuerdo con la figura 6.10.

6.4 Diagnóstico y recuperación

Cuando el robot se encuentra a la espera de una nueva tarea, el único monitor activado es el que verifica el nivel de la batería. Los demás se activarán cuando se empieza a ejecutar la tarea que lleva asociada. De la misma manera, al finalizar la tarea o bien al ser ésta cancelada (las tareas pueden ser canceladas, por ejemplo, por el sistema de recuperación), los monitores asociados a dicha tarea se han de desactivar. Cada vez que se dispara un monitor (es decir, cuando algún residuo bajo su control excede los márgenes permitidos o detecta algún otro tipo de síntoma) se envía un mensaje al módulo de diagnóstico.

La fase de diagnóstico o identificación es bastante sencilla. Su finalidad es identificar la excepción en la que se encuentra el robot, en función de los mensajes enviados por los distintos monitores.

La prioridad de los mensajes viene determinada por la jerarquía de los monitores en la estructura arborescente anteriormente descrita. Siempre tienen preferencia los mensajes de los monitores más específicos. La razón es que, desde el punto de vista de recuperación interesa ejecutar el método de recuperación más específico posible. Así, la recepción de un mensaje del monitor del tiempo, debido precisamente a su amplia cobertura, indicará que algo va mal pero no se podrá aislar la situación de excepción, dado que cualquier problema o situación de excepción puede hacer disparar este monitor. A menos que se obtenga información de otros monitores, sólo se podrá informar acerca de la existencia de un problema al operador o realizar una serie de tests para determinar el estado del robot. En el otro extremo, si el monitor disparado es el monitor de giros indicando que el robot se encuentra dando vueltas en un mismo sitio, la situación está completamente identificada y se puede llevar a cabo un proceso de recuperación concreto (moverse a otra posición, por ejemplo).

Las excepciones contempladas por el supervisor se agrupan en conjuntos con un monitor asociado a cada conjunto. El primer paso en el proceso de identificación es registrar los mensajes de los monitores, junto con información acerca del instante de llegada, en una cola de mensajes. A continuación se ejecutan una serie de funciones de verificación (una por cada conjunto de excepciones), de acuerdo con el esquema de prioridades anteriormente visto. Este conjunto de funciones también se encarga de eliminar de la cola de mensajes aquellos que han caducado.

Cada vez que llega un mensaje nuevo, se hace un análisis de la cola donde se almacenan buscando patrones, de acuerdo con unas reglas preestablecidas. Por cada conjunto de excepciones, debe estar registrada una función encargada de comprobar si, con los mensajes existentes en la cola, se puede afirmar que el robot se encuentra en una de las excepciones asociadas a ese conjunto. Si la detección es positiva, se borra la cola de mensajes y se activa el mecanismo de recuperación correspondiente.

A continuación, se describen las líneas de funcionamiento (detección y recuperación) para cada uno de los distintos conjuntos de excepciones.

6.4.1 Nivel de batería bajo

En este caso, la detección se basa solamente en los mensajes del monitor de la batería. Existen tres tipos diferentes que se corresponden con tres posibles estados de la batería a tratar (*BATT-BAJA*, *BATT-ALERTA*, *BATT-ALARMA*⁴) de acuerdo con la figura 6.10. El cuarto estado se produce cuando tiene carga suficiente (*BATT-OK*) y, en ese caso, el monitor permanece silencioso.

Las acciones a realizar están directamente asociadas con el mensaje, con lo cual la fase de diagnóstico se reduce a identificar el tipo de mensaje:

- ✓ *BATT-BAJA*: El nivel de la batería es suficiente si todo va bien para acabar la tarea actual y al finalizarla ir a recargar. Si la tarea actual dura más de lo previsto, puede recibirse un mensaje de *BATT-ALERTA*.
- ✓ *BATT-ALERTA*: El nivel de la batería será suficiente en la situación más favorable pero hay pocas garantías de ello.
- ✓ *BATT-ALARMA*: Este caso supone que el nivel de las baterías es demasiado bajo para mantener el robot moviéndose. Podría ser incluso insuficiente para llegar a la estación de carga.

⁴ En realidad, antes de generar este mensaje, se estima la carga de la batería después de volver a la estación de carga sin contemplar la tarea actual. Si no es suficiente, se emite un *BATT-ALARMA* y, si es suficiente, se emite un *BATT-ALERTA*.

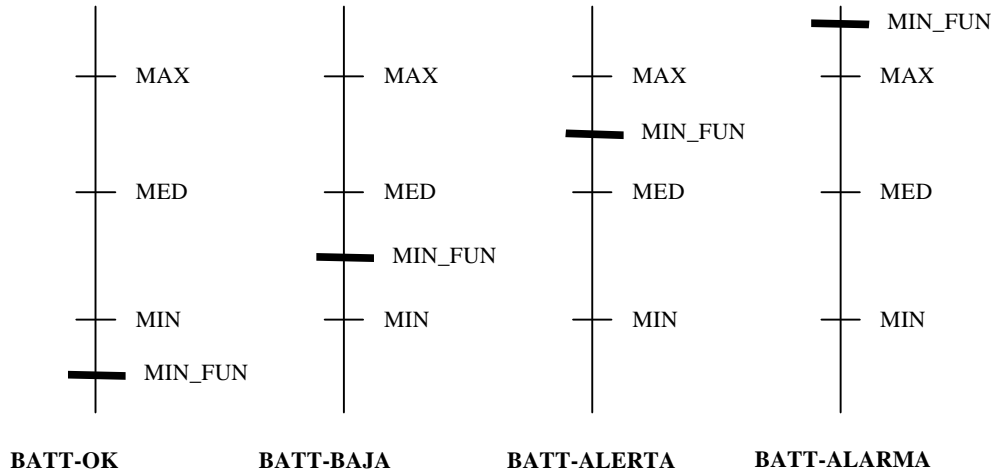


Figura 6.10 Estados posibles de la carga de la batería. MIN_FUN es el nivel mínimo necesario (32v). MIN, MAX, MED son los estimados

Las acciones a llevar a cabo están directamente asociadas con el tipo de mensaje:

- ✓ **BATT-BAJA:** Acabar la tarea actual y al finalizarla ir a recargar.
- ✓ **BATT-ALERTA:** Se aborta la tarea actual y se planifica una nueva tarea para dirigirse a la estación de recarga.
- ✓ **BATT-ALARMA:** Se envía un mensaje al operador.

6.4.2 Demasiadas vueltas

La identificación es positiva si se encuentra en la cola un mensaje proveniente de este monitor, junto con otro que provenga de él o de cualquiera de los que se encuentran por encima en el árbol.

La recuperación se basa simplemente en suspender temporalmente la acción de seguimiento de la trayectoria, alejar el robot de la posición actual y continuar con la navegación. Se ha de resaltar que la actualización de la posición, por parte del módulo

de navegación, sigue activa en todo momento. Algunas veces, la situación se vuelve a repetir al volver el robot a la misma posición, pero la probabilidad que esto ocurra es bastante remota, dado que la probabilidad de esta excepción en sí es también bastante reducida.

6.4.3 Error de posición

La fase de identificación es muy similar a la del caso anterior.

En el capítulo cinco se ha descrito como el planificador de caminos, en el nivel más alto, modela el entorno en forma de un mapa topológico con nodos y arcos. Los nodos representan, en su mayor parte, uniones de dos espacios (unión entre pasillos, puertas, etc.), mientras que los arcos representan conexiones entre nodos (por ejemplo, entre los dos extremos de un pasillo debe haber un arco). Cada arco tiene asociado una serie de parámetros entre los que destaca la probabilidad de bloqueo. En primera instancia, el planificador obtiene una trayectoria entre el punto origen y el destino como una secuencia de nodos y arcos, usando una estrategia de búsqueda basada en el algoritmo de búsqueda A^* [Goodwin, 1996].

La recuperación, en este caso, se hace determinando cuál es el arco con problemas y calculando un camino alternativo que evite dicho arco. Para ello se dispone de una copia del camino topológico que el robot está siguiendo. El monitor de posición va actualizando el estado de los arcos pertenecientes al camino, clasificándolos como *recorridos* cuando se dispone de suficiente certeza de que el robot se encuentra en el siguiente arco. Una vez identificado la excepción, lo primero que hace el módulo de recuperación es enviar un mensaje al planificador para que clasifique el arco como bloqueado (aumentando su probabilidad de bloqueo). A continuación se solicita un nuevo camino.

Cuando el robot se encuentra bastante lejos del nodo más cercano, el planificador añade un nuevo nodo, correspondiente a la posición actual, unido mediante nuevos arcos a los nodos más próximos. Por lo tanto, en la mayor parte de los casos, el camino que el planificador calcula es el mismo que tenía, a diferencia del primer nodo. Se ha desarrollado un mecanismo para detectar esta situación y, en caso de que el primer arco tenga como nodo final el mismo que el arco con problemas, indicar al planificador que ponga ese nodo como no navegable y calcular un nuevo camino.

El robot puede desviarse de la ruta original al tratar de evitar obstáculos, con lo cual, el sistema de seguimiento del robot a través de la trayectoria se complica. Para resolver este problema, se ha ampliado la trayectoria topológica que el supervisor posee con todos aquellos arcos en los cuales el robot se va situando. Esto se hace de

forma dinámica, cuando el robot comienza a seguir una trayectoria obtiene una copia de esta y va añadiendo un arco, junto con un nodo, cada vez que la distribución de probabilidades de los nodos de Markov indiquen que el robot se encuentra en un nuevo arco que no esté en la trayectoria original.

El modelo de navegación basado en POMDP obtiene los estados más probables en cada instante, pero no la secuencia de estados más probable. Por otra parte, el algoritmo de Viterbi [Rabiner 1986] provee una forma de encontrar la secuencia de estados más probable. No obstante, este algoritmo no explota la información que proporciona el modelo de Markov. En [Haigh 1998] se puede ver una extensión a este algoritmo para identificar la secuencia de estados más probable, usada después para aprender parámetros como los tiempos empleados en atravesar los distintos arcos. La utilización de dicho algoritmo podría conseguir más precisión a la hora de determinar el arco en el cual el robot está teniendo problemas. No obstante, en el sistema de diagnóstico se ha usado la información proporcionada directamente el modelo de Markov.

Queda por describir la forma en que se obtienen de los arcos a partir de los nodos empleados por el sistema de navegación. Este último divide el plano en el que se mueve el robot en celdas de un metro cuadrado, mientras el planificador usa nodos y arcos. La diferencia puede verse en la figura 6.11. El sistema ha sido diseñado para crear el modelo de Markov a partir del plano topológico, pero no para realizar el proceso inverso. En general, un nodo de Markov puede pertenecer a varios arcos. En [Haigh 1996] se presenta una posible forma de resolver este problema seleccionando un conjunto de arcos en lugar de uno único. No obstante, la situación que en esta Tesis se plantea es diferente e interesa seleccionar uno sólo. En concreto, se selecciona el arco más cercano al camino planeado, preferiblemente que pertenezca a éste. Si no hay ninguno, se selecciona uno conectándolo al plano topológico que mantiene el monitor. La selección se basa en los nodos extremos del arco que tienen que estar ya en otro nodo del plano del monitor. Si el nodo está asociado a varios arcos del camino, entonces se selecciona el más avanzado.

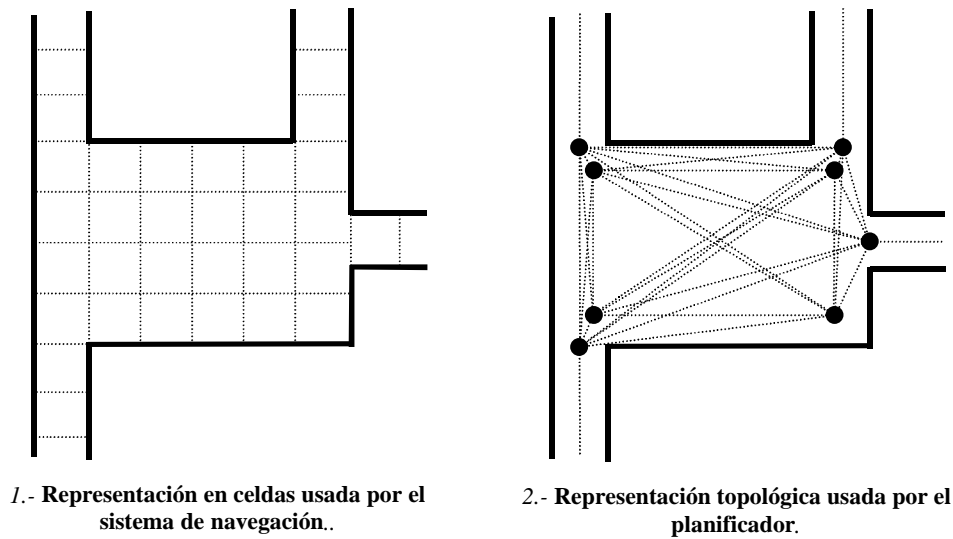


Figura 6.11 Diferentes representaciones de una parte del plano (sala a la que concurren cuatro pasillos).

Por último, puede ocurrir que no haya camino alternativo, en cuyo caso se aborta la tarea y se pasa a la siguiente, al mismo tiempo que se envía un mensaje al operador.

6.4.4 Otras excepciones

De momento no se han establecido más mecanismos de recuperación. No obstante, algunos monitores tales como el que se dispara cuando una tarea excede del tiempo máximo asignado, pueden informar acerca de una serie de problemas en el robot que no han sido contemplados (fallos en el hardware, problemas con algún módulo software, etc). Como la probabilidad de que se produzcan éstas es muy baja, la única medida a tomar es enviar un mensaje indicando posibles problemas con el robot. Se pueden ir añadiendo más mecanismos a medida que manifiestan nuevas situaciones.

7 Modelo estocástico

7.1 Introducción

En el capítulo anterior se ha planteado la necesidad de tratar la incertidumbre de la información de entrada al sistema de supervisión. Esto es debido, en gran medida, a la dificultad de encontrar monitores (síntomas) que se disparen (identifiquen) únicamente ante situaciones de excepción y, en ningún caso, ante situaciones normales. Las situaciones de excepción se habían definido como aquellas en las que el robot no finaliza con éxito la tarea encomendada, ya sea debido a fallos en el sistema o por encontrarse ante situaciones no contempladas en la fase de diseño.

Mientras que algunos monitores pueden ser disparados en situaciones nominales, en otros, con el fin de evitar este problema, se han seleccionado rangos sobre los restos (señales que supervisan) muy amplios, al considerar el caso más desfavorable. Esta sobredimensión de rangos conlleva normalmente un retardo en la detección, de forma que, en algunas situaciones, es preferible detectar “falsas alarmas” (siempre que ocurran con muy poca frecuencia) que asumir estos retardos. Este efecto puede verse, por ejemplo, en la monitorización del tiempo de las tareas (apartado 6.3.1)

En la arquitectura presentada en el capítulo anterior, las decisiones se toman directamente a partir de los mensajes obtenidos por los monitores. Esto es, se supone que si se dispara un monitor, el robot se encuentra en una de las situaciones para las que fue diseñado. Nada más lejos de la realidad cuando se trata con robots móviles. Como ejemplo, se ha visto que el sistema de monitorización de vueltas que da el robot busca unos patrones que, aunque con poca probabilidad, podrían darse en algunas situaciones sin que ello implique ninguna anomalía. Es, por tanto, necesario tener en cuenta la incertidumbre sobre el conocimiento del estado del sistema. Esta incertidumbre es debida a que las observaciones sobre el proceso no son completamente fiables.

Algo similar ocurre con el resultado de las acciones. En este caso se consideran como acciones las subtareas que el robot puede realizar tales como

“solicitar camino alternativo”, “seguir camino”, “abortar tarea”, etc. Debido a la complejidad del sistema y a posibles situaciones no previstas, algunas acciones no siempre llegan a buen término. No obstante, sí se puede evaluar numéricamente mediante probabilidades de éxito.

Por último, si se tiene en cuenta que el estado del sistema no es siempre conocido de forma determinista, también se debe tener en cuenta la bonificación/penalización de la realización de las distintas acciones para cada uno de los posibles estados. Por ejemplo, la penalización de la acción “abortar tarea” cuando el sistema considera la posición destino inalcanzable dependerá de si el robot se encuentra realmente ante dicha situación y de la importancia de la tarea cancelada.

La mayoría de los esfuerzos en el campo del tratamiento de fallos se ha enfocado al problema de la generación de síntomas, argumentando que la parte de decisión basada en residuos bien diseñados es relativamente fácil. En algunos procesos como el que nos ocupa, la tarea de obtención de síntomas fiables no es tan simple. La resolución de problemas con modelado perfecto es, a todas luces, mucho más sencilla que cuando no es posible modelar completamente el sistema.

El enfoque presentado en este capítulo trata de optimizar el proceso de decisión asumiendo que los síntomas no son completamente fiables (trabajando así con probabilidades). Este modelo, al igual que el descrito en el capítulo anterior, está basado en los mensajes de los monitores, pero teniendo en cuenta la incertidumbre acerca de su relación con una situación de excepción, la posibilidad de fallo de las acciones y el coste de las mismas. El objetivo es que el supervisor tome las decisiones óptimas conociendo de antemano las probabilidades de fallo de las observaciones y de las acciones. Se considera una decisión óptima cuando, con la información disponible, dicha decisión conlleva una bonificación máxima a largo plazo. El sistema de bonificación es definido por el usuario, quien decide qué tareas son más importantes o qué recursos son más costosos.

En el primer apartado se presentan las características del sistema desde el punto de vista de la supervisión. Luego se describen dos soluciones aplicables al problema, y la solución aquí adoptada. En el tercer apartado se describe con detalle el modelo desarrollado en esta tesis. Se pospone para el capítulo siguiente el análisis de los resultados obtenidos tanto de simulación como de experiencias prácticas con los diferentes robots móviles.

7.2 Características del sistema

En este apartado se presentan las características del sistema a supervisar para luego decidir qué modelo de supervisión se adapta mejor a esta situación. Interesa, sobre todo, saber cuál es la información que se puede obtener del sistema para conocer su estado, los métodos de recuperación o acciones que el robot puede llevar a cabo y los objetivos del sistema.

7.2.1 Información de estado

La información acerca del estado del sistema viene dada por:

1. Los monitores que a intervalos regulares de tiempo suministran información al sistema. Dicha información puede ser originada, bien porque se ha disparado el monitor, o por la ausencia de disparo (indicando que no se verifican las condiciones de disparo).
2. Peticiones a distintos módulos que integran el sistema. Tales como solicitar lecturas sonar al gestor de sensores, posición al navegador, etc.
3. El resultado de la ejecución de algunas acciones puede proporcionar información sobre el estado del sistema. Por ejemplo, es posible diseñar una acción encargada de buscar un espacio libre en el camino para que el robot pueda pasar. Se puede realizar haciendo un barrido con el láser (girando el robot, dado que el ángulo del láser son 15 grados aproximadamente). Dicha acción proporciona información sobre el estado del camino (bloqueado o libre).

En lo que sigue, se usará el término *observación* para referirse a la información proporcionada por estas fuentes.

Las características de esta información son:

1. Todas las fuentes de información son no deterministas, lo cual representa un problema. Ya se ha comentado antes que los monitores pueden ser disparados en una situación nominal. Lo mismo ocurre con la salida de algunas acciones. Continuando con el ejemplo de la acción de buscar un espacio libre en el camino, es probable que no se detecten obstáculos, debido a que el láser no es capaz de ver cierto tipo de objetos, informando que existe un espacio libre, cuando en realidad no lo hay. Lo mismo puede ocurrir con las peticiones a otros módulos.
2. Otra característica es que el coste para obtener dicha información no es el mismo. Buscar un espacio libre significa parar el robot ejecutar un proceso que puede

durar varios segundos, mientras que los monitores obtienen información sin ninguna interrupción en el funcionamiento normal del robot.

7.2.2 Acciones

Se pueden diferenciar dos tipos de acciones destinadas a conseguir un determinado objetivo final: *acciones reductoras de entropía* y *acciones directas*. Las primeras son aquellas cuyo objetivo inmediato es reducir la entropía de la información de estado del sistema, es decir, reducir la incertidumbre sobre el estado del sistema. Por otro lado, las *acciones directas* son aquellas cuya finalidad inmediata es la consecución del objetivo, suponiendo que no hay incertidumbre en el sistema. Son pues acciones cuya realización con éxito supone un paso más en la consecución del objetivo final, mientras que la realización de las acciones reductoras de entropía supondrá obtener información extra para disminuir la incertidumbre acerca de su estado. Por ejemplo, considerando la tarea de navegación, si se dispone de una acción de localización que determina la posición del robot, ésta pertenecería al grupo de acciones reductoras de entropía, mientras que la acción de moverse hacia el destino sería una acción directa.

Existen muchas acciones que combinan ambos objetivos. Un ejemplo es el de navegación usando modelos de Markov [Simmons 1995]. La acción de seguir la trayectoria planificada en zonas del entorno con características diferenciables (esquinas, cruce de pasillos, etc) supone, tanto un acercamiento al objetivo, como una reducción de la entropía. La reducción de la entropía se produce porque el robot utiliza estas características del entorno para ajustar su posición.

En realidad, las *acciones reductoras de entropía* también tienen como último fin la consecución del objetivo. Dependiendo de la situación, su contribución puede ser mayor (en media a lo largo del tiempo) que ninguna otra acción. En efecto, siguiendo con el ejemplo de navegación, supongamos la situación en la que la posición del agente móvil es completamente desconocida (alta incertidumbre acerca del estado del sistema). En tal situación, la acción de moverse hacia el objetivo en una determinada dirección puede implicar tanto un acercamiento al objetivo como alejarse del mismo. Una acción destinada a reducir la incertidumbre de la posición del sistema (reducción de la entropía) es la más adecuada.

Otra característica de las acciones a ejecutar por el agente móvil, es que la mayor parte de las mismas no son deterministas. Al ejecutar la acción de ir a través de un espacio estrecho en el camino (puertas o espacio entre dos obstáculos por ejemplo), bien sea debido a problemas de percepción (es sabido que el sistema sensorial tiene problemas con cierto tipo de objetos), o a problemas del módulo reactivo, dicha acción

puede fracasar, aún cuando haya espacio suficiente para pasar. Esto es aplicable a casi la totalidad de las acciones.

Por último, cada acción tiene asignado un coste en cuanto a tiempo y recursos. El coste de una acción puede ser variable según la situación en la que se encuentre el robot. Por ejemplo, ejecutar una acción que sea seguir un camino alternativo al que está siguiendo actualmente tendrá un coste que depende de la diferencia entre ambos caminos, el coste de cancelar una tarea porque el destino se cree inalcanzable, depende de la importancia de esa tarea, etc.

7.2.3 Objetivos

Aunque se pueden tener en cuenta objetivos diferentes, de acuerdo con lo descrito anteriormente, se necesita un modelo que considere una serie de factores en sus decisiones. Resumiendo, se puede decir que el modelo ha de tener en cuenta:

- ✓ Los tipos de acciones anteriormente descritas y use la más adecuada según la entropía de la información de estado en cada momento.
- ✓ La probabilidad de éxito de las acciones.
- ✓ La incertidumbre de las observaciones.
- ✓ El coste de las acciones. Teniendo en cuenta que éste puede variar con el tiempo y el estado del agente.
- ✓ Además, sería deseable que el modelo fuese incremental; es decir, que fuese fácil añadir información a medida que ésta es adquirida con la experiencia, o bien por modificaciones que se puedan hacer en el robot. Por ejemplo, si se añade la posibilidad de que el robot pueda realizar una acción más, el objetivo sería caracterizar de forma sencilla su comportamiento para que sea tenido en cuenta por el supervisor.

En definitiva, se pretende que las decisiones tomadas por el robot sean parecidas a las que haría un ser humano si dispusiera de la misma información que el robot posee. Como por ejemplo, sería deseable que el robot tuviera reacciones similares a las siguientes:

- ✓ Si hay un obstáculo en el camino que los sensores del robot no pueden detectar, este chocará con el y reintentará seguir el camino. No obstante, el supervisor ha de considerar también la posibilidad de la existencia de un obstáculo no visible y tratará de usar otro tipo de métodos de detección. Si el problema persiste, debe optar por seguir otra trayectoria alternativa.

- ✓ En la misma situación anterior, si el camino alternativo es muy parecido al actual y el coste de usar métodos de detección de obstáculos alternativos es mayor que la diferencia entre los dos caminos (costes esperados), debería elegir la trayectoria alternativa directamente.

7.3 Modelos de diagnóstico aplicables.

Existen varias formas de tratar la incertidumbre en las observaciones, usando en casi todas ellas herramientas estadísticas. Entre las técnicas más extendidas se encuentran las redes de Bayes, sistemas de decisión basados en procesos de Markov y métodos basados en filtros de Kalman.

7.3.1 Redes de Bayes en supervisión, diagnosis y recuperación

Este tipo de redes, cuyo funcionamiento se ha descrito en el tercer capítulo, está teniendo gran éxito en sistemas de diagnosis. Sin embargo, presenta varios problemas en la aplicación a sistemas de supervisión con características similares al que aquí se trata.

En primer lugar, asume que los resultados de los tests son exactos (la incertidumbre radica en que puede haber distintos problemas con el mismo síntoma), lo cual puede ser cierto en aplicaciones tales como la diagnosis medica, pero no en el caso de la navegación en robots móviles.

Otro problema viene dado por no tener en cuenta el efecto negativo que puede ocasionar una falsa detección o, dicho de otra forma, no considerar el coste de las acciones de recuperación ante el problema detectado.

Estos sistemas de diagnosis están más orientados a determinar el mínimo número de test para detectar cual es el problema, mientras que en navegación, los monitores que indican el estado del sistema se suponen siempre activos.

Por último, no es nada sencillo ni intuitivo encontrar las variables que representan los nodos del árbol, así como las probabilidades de transición para modelar un sistema complejo.

7.3.2 Procesos de Markov

Los procesos de decisión de Markov parcialmente observables POMDP están siendo objeto de un amplio estudio por parte de la comunidad científica que se dedica a planificación y control dentro del área de Inteligencia Artificial. A continuación se

describen brevemente las características de un proceso de Markov parcialmente observable (POMDP), comenzando para ello definiendo el concepto de proceso de Markov completamente observable (COMDP). Una descripción más detallada puede verse en el apéndice A.

7.3.2.1 COMDP

Si bien algunos autores [Hauskrecht 1996] emplean el término MDP para referirse a los procesos de decisión de Markov completamente observables (COMDP), en lo que sigue se usa este término para englobar tanto a los POMDP como a los COMDP, de acuerdo con la terminología usada por [Cassandra 1998].

Un COMDP está definido por:

$$\Xi \equiv (S, A, R, T)$$

donde:

S: conjunto de estados del sistema a modelar.

A: conjunto de acciones posibles a realizar.

R: recompensa o ganancia.

$$R : S \times A \times S \rightarrow \Re$$

T: probabilidad de transición entre estados al ejecutar una acción.

$$T : S \times A \rightarrow \Pi(s)$$

La solución del COMDP se basa en encontrar la secuencia de acciones que maximice la bonificación a lo largo del tiempo. Cuando este tiempo es limitado, el proceso se dice de *horizonte finito* y, cuando es ilimitado de *horizonte infinito*. En la mayor parte de los casos se acostumbra a utilizar un factor de penalización con el tiempo (γ).

Se trata en definitiva de maximizar⁵ una de las siguientes funciones:

a) Horizonte finito

⁵ En el contexto de esta tesis se usa la bonificación positiva.

$$E \left[\sum_{t=0}^{T-1} r^t r(s^t, A^t) \right] \quad 0 \leq r \leq \infty \quad [Eq. 7.1]$$

b) Horizonte infinito

$$E \left[\sum_{t=0}^{T-1} r^t r(s^t, A^t) \right] \quad 0 \leq r \leq \infty \quad [Eq. 7.2]$$

La solución es una **política** (\mathbf{p}) o plan de actuación, esto es, una secuencia de reglas de decisión que define para cada intervalo de tiempo y cada estado una acción a realizar:

$$(d^0, d^1, d^2, d^3, \dots)$$

donde cada regla es una aplicación del espacio de estados al de acciones:

$$d^t : S \rightarrow A$$

Si las reglas son dependientes del tiempo, la política se denomina **no estacionaria**, frente a aquellas que no dependen del tiempo, llamadas **estacionarias**. Un interesante teorema [Hauskrecht 1996] establece que para un horizonte infinito hay siempre una política estacionaria que es óptima. Esta se puede calcular iterando la función valor o la política de acuerdo con los algoritmos descritos en el apéndice A. Se denominan funciones valor a aquellas que evalúan la ganancia obtenida al actuar de acuerdo con una determinada política.

7.3.2.2 POMDP

Si bien un proceso de Markov completamente observable tiene en cuenta la probabilidad de fallo de las acciones, es un modelo determinista en cuanto a las observaciones, dado que se conoce el estado del sistema en todo momento. Al considerar ambos tipos de incertidumbre, se obtienen los modelos de Markov parcialmente observables (POMDP) definidos:

$$\Xi \equiv (S, A, Z, R, T, O)$$

S: conjunto de estados.

A: conjunto de acciones.

Z: conjunto de observaciones.

R: recompensa o ganancia.

$$R : S \times A \times S \times Z \rightarrow \Re$$

T: transiciones.

$$T : S \times A \rightarrow \Pi(s)$$

O: probabilidades de observación.

$$O : S \times A \rightarrow \Pi(z)$$

$$o(a, s, z) = P(o^t = z / s^t = s, A^{t-1} = a)$$

La **información de estado** es una distribución de probabilidades ($P(s)$) entre los distintos estados (s). Esta distribución es un estadístico suficiente, dado que condensa toda la información pasada sobre el proceso.

7.3.2.3 Aplicación en supervisión, diagnóstico y recuperación

El modelo COMDP tiene en cuenta la probabilidad de fallo de las acciones así como su coste:

- ✓ La probabilidad de fallo viene dada por las probabilidades de transición entre estados. $\{T: S \times A \rightarrow \Pi(s)\}$
- ✓ El coste de las acciones viene dado por el esquema de bonificación o ganancia. $\{R: S \times A \times S \rightarrow \Re\}$

Además, la solución a dicho modelo es relativamente sencilla. No obstante, para obtenerla (apéndice 1) se considera que el estado del proceso es conocido en todo momento. Esto quiere decir que no considera la incertidumbre en la información de estado, hipótesis poco adecuada para procesos como el del robot móvil. Esta simplificación implica a su vez no tener en cuenta las *acciones reductoras de entropía*, dado que éstas nunca serán seleccionadas. La razón es bastante obvia: si el estado del sistema es conocido, la ejecución de una acción para determinar su estado no tiene ningún sentido.

Por otra parte, el modelo POMDP añade al anterior el tratamiento de incertidumbre en el estado del agente. El coste que esto conlleva es un incremento en la complejidad del modelo. Por tanto, tiene en cuenta lo siguiente:

- ✓ Cualquier tipo de acciones (*Acciones reductoras de entropía y acciones directas*).
- ✓ Probabilidad de éxito de las acciones.
- ✓ Incertidumbre de las observaciones.
- ✓ Coste de las acciones.

Parece que los requisitos iniciales se ven satisfechos con esta solución pero esto tiene un coste en cuanto a la complejidad del modelo, tanto de cara a su definición, como el cálculo de la política óptima. Su solución puede hacerse inalcanzable aplicando los algoritmos de que se dispone en la actualidad. El desarrollo de nuevos y más eficaces algoritmos es un tema que ocupa a muchos grupos de investigación.

La definición del modelo adecuado es otro de las dificultades del empleo de este sistema. En efecto, como se verá a lo largo del capítulo, no es sencillo determinar todos los parámetros (la mayor parte en forma de probabilidades) para determinar el modelo. Cabe recordar que otro de los objetivos es que el modelo del proceso sea incremental; es decir, que sea fácil añadir información a medida que ésta es adquirida con la experiencia, o bien por modificaciones que se puedan hacer en el robot. En esta línea se presentarán métodos para la reducción de la complejidad en la definición de parámetros al determinar el modelo, lo cual facilita a su vez su posterior manipulación.

En resumen, este modelo es útil para describir tareas de planificación en las cuales el decisor no tiene conocimiento completo acerca del estado del proceso. Provee una forma conveniente de razonamiento sobre el compromiso entre ejecutar acciones para obtener información y acciones para obtener bonificación.

7.4 Descripción del modelo

El modelo utilizado es un POMDP con ciertas modificaciones que se irán describiendo a lo largo de este capítulo. Primero se presenta la identificación de los parámetros, y luego el funcionamiento del decisor. Debido a que actualmente sólo se pueden calcular soluciones exactas sobre modelos POMDP para problemas de pequeñas dimensiones [Washington 1996], se aplicarán algunas aproximaciones.

Resumiendo, las principales características del sistema a construir son:

- ✓ El proceso es descrito usando un conjunto finito de estados.
- ✓ El sistema de supervisión puede actuar sobre el proceso a través de un conjunto de acciones.
- ✓ La dinámica del sistema se describe mediante un esquema de probabilidades de transición entre estados.
- ✓ El supervisor posee información sobre el estado del proceso a través de un conjunto finito de observadores.
- ✓ La calidad de la supervisión viene dada por medio de un sistema de bonificación o costes asociados a estados o transiciones.
- ✓ El objetivo del supervisor es maximizar una función que combina los costes y bonificaciones obtenidas a lo largo del tiempo.

El proceso de decisión de Markov parcialmente observable (POMDP) está, por lo tanto, definido mediante cada uno de sus componentes:

$$\Xi \equiv (S, A, Z, R, T, O)$$

Donde S , A y Z son tres conjuntos que representan los estados, las acciones y las observaciones, mientras que R , T y O representan la política de bonificaciones y las distribuciones de probabilidad de transiciones y observaciones.

En las siguientes secciones se describe en detalle cómo se han definido estos parámetros para el caso del robot móvil.

7.4.1 Estados

Los estados a modelar son las situaciones relevantes, desde el punto de vista de supervisión, en las que el robot se puede encontrar.

El objetivo principal del módulo de supervisión es que el robot se mantenga funcionando correctamente durante largos períodos de tiempo para realizar un mayor número de tareas de forma autónoma. El estado nominal del robot durante la navegación se da cuando éste sigue la trayectoria planeada sin problemas. Dicho estado se denotará como **PF** (Path Following). Con respecto a las excepciones, se puede añadir un estado por cada una de estas situaciones, al igual que en el modelo presentado en el capítulo anterior.

Las situaciones de excepción se pueden considerar de forma aislada como se hizo anteriormente, o bien como combinaciones entre ellas. Por ejemplo, el robot puede estar en la situación donde el camino está bloqueado y además se encuentra con problemas en el módulo reactivo, dando vueltas sobre sí mismo, con posibles problemas en el sistema sensorial, etc. Esta situación estaría resuelta anteriormente con el esquema de prioridades de los monitores. Sin embargo, el modelo POMDP exige que el sistema sólo pueda estar en un estado a la vez. La solución a este problema consiste en considerar el conjunto de estados como una combinación de estas situaciones básicas, denominadas *componentes* en esta tesis.

En la figura 7.1 se presenta una posible división de los problemas con los que el robot se puede encontrar.

Hay que resaltar que, según lo descrito en la figura 7.1, existen multitud de componentes que pueden fallar, e incluso puede ser que una parte del sistema de control, tal como el módulo reactivo, en ciertas situaciones no se comporte como debería. Pero aquí sólo se consideran aquellos fallos que conllevan un problema grave del robot. Es decir, cuyo resultado sea que el robot no logre su objetivo, o lo haga demasiado tarde.

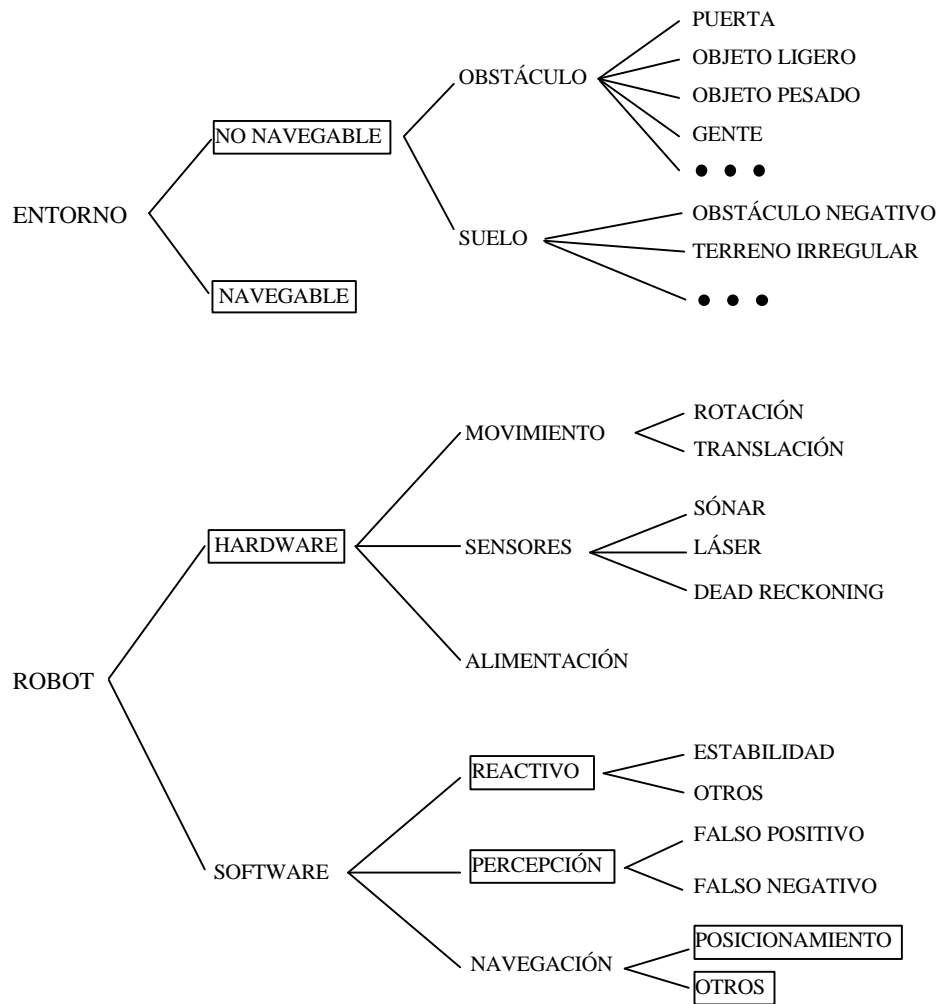


Figura 7.1 Clasificación de posibles problemas en la navegación.

La solución al POMDP se puede simplificar si las acciones no tienen efecto en el estado del sistema. Aunque esta hipótesis ha sido utilizada por algunos autores [Washington 1998] en supuestos teóricos, resulta bastante restrictiva para situaciones reales como la que nos ocupa. No obstante, se puede hacer uso de otras características como pueden ser la independencia entre algunos parámetros o la posibilidad de la eliminación de algunos estados. Si dos estados tienen las mismas observaciones y probabilidades de transición, el sistema no será capaz de discernir entre ambos. Lo mismo ocurre en cuanto a las acciones y bonificaciones. Si dos estados tienen iguales probabilidades de transición, observaciones e idéntica bonificación para todas las acciones, la política óptima asignará las mismas acciones para ambos estados.

Por ejemplo, considerando la división de la figura 7.1, para la situación en la que el robot tiene un obstáculo en su camino, considerar la naturaleza de dicho obstáculo sólo será relevante si se dispone de algoritmos de recuperación distintos para cada una de las situaciones, así como de observaciones y/o probabilidades de transición que permitan discernir entre ambos estados. Como ejemplo, se podían diferenciar las siguientes situaciones:

- ✓ Personas en el camino. Una acción que el robot puede tomar para saber si hay gente es solicitar que se aparten utilizando el mecanismo de voz. Dicha acción no sería determinista, ya que depende de la actitud de las personas que obstaculizan el camino.
- ✓ Posibilidad de mover el obstáculo. Diferenciar entre un objeto pesado o ligero es útil, porque el objeto ligero (una caja vacía por ejemplo) puede ser movido por el robot empujándolo.

No obstante, si no se dispone de observaciones para diferenciar tales situaciones y las acciones son las mismas con los mismos resultados (mismas probabilidades de transición), no tiene sentido considerar estados nuevos. Tampoco habrá forma de caracterizarlos, con lo cual se pueden agrupar y definir de forma conjunta.

En el modelo propuesto se aprovechan los monitores desarrollados en el capítulo anterior y se añaden otros métodos de obtener información acerca del estado del robot descritos más adelante.

Las excepciones tenidas en cuenta son los estados resultantes de todas las posibles combinaciones de los problemas que aparecen encuadrados en la figura 7.1. También se puede, en el futuro, modelar más en detalle cualquiera de estos problemas si se dispone de mecanismos suficientes para su tratamiento. Siempre que sea sencillo añadir y modificar al modelo POMDP, no hay nada que limite una posterior división.

7.4.1.1 Descripción de los estados (componentes)

Los estados que han sido seleccionados son el estado nominal (**PF**) y cualquier combinación de estos componentes enmarcados en la tabla 7.1

La clasificación de los problemas del software se hace atendiendo a los módulos básicos que participan en la navegación (ver capítulo 5).

Camino no navegable (NNP)

Incluye todos los problemas en el entorno que hacen imposible al robot seguir una determinada trayectoria. Teóricamente puede ser posible distinguir varias situaciones dependiendo de la naturaleza del problema. Ello sería interesante si se dispusiese de distintas acciones para tratar con estos problemas de forma individual y se dotase al robot con mecanismos de observación para distinguir y caracterizar tales situaciones.

Problemas hardware (HP)

Existen diferentes componentes hardware, cuyo funcionamiento incorrecto puede ocasionar el fracaso de alguna tarea. Aquí se incluyen todos bajo el nombre **HP**.

Estos problemas pueden ser debidos a fallos en cualquier dispositivo, siendo posible su división, al igual que en el caso anterior, en componentes pero de momento no se dispone de observadores, acciones de recuperación ni forma de caracterizar estas situaciones.

Tabla 71. Componentes de los estados.

componentes	descripción
NNP	Camino no navegable (<i>Non Navigable Path</i>).
HP	Problemas con el hardware (<i>Hardware Problems</i>).
RT	Problemas del módulo reactivo (<i>Reactive Problems</i>).
PP	Problemas de percepción (<i>Perception Problems</i>).
WD	La dirección de referencia no es correcta (<i>Wrong Direction</i>).
LST	Problemas de localización (<i>Robot Lost</i>).
NNA	No hay camino alternativo (<i>Non Navigable Alternative</i>).

Problemas en el nivel reactivo (RT)

RT incluye los problemas que pudiera tener el módulo reactivo, exceptuando los problemas de percepción. En la actualidad se conocen algunos problemas de estabilidad en ciertas situaciones que ocurren con muy poca frecuencia y que se tratan de modelar con este estado.

Problemas de percepción (PP)

PP engloba los problemas de percepción. Tal y como se describe en la figura 7.1 estos problemas no abarcan los causados por fallos del hardware. No obstante, debido a que ambos están muy relacionados y es muy difícil diferenciarlos, se ha decidido englobarlos bajo el nombre de **PP**. Por lo tanto, **PP** incluye los problemas hardware de los sensores y **HP** excluye dichos problemas.

Aunque los sensores funcionen de acuerdo a las especificaciones (por lo tanto, no hay fallos), existen situaciones de excepción **PP** siempre que el robot no es capaz de detectar un objeto que se interpone en su camino. Este tipo de situaciones, como ya se ha mencionado, son la razón del cambio de notación, usando el término *situaciones de excepción* donde otros autores emplean *fallos*.

Esta componente resulta fundamental para diferenciar ciertas situaciones. Por ejemplo, cuando se combina esta componente con la de entorno no navegable permite diferenciar dos situaciones: la primera, que el camino esté bloqueado y sea detectable por el sistema sensorial (**NNP**), y la segunda, que esté bloqueado pero no sea detectable (**NNP & PP**).

Dirección de referencia errónea (WD)

Como se sabe, la función del navegador es doble. Por un lado ha de seguir la posición del robot y, por otro, planificar y enviar direcciones de referencia al módulo reactivo. **WD** incluye los problemas que pueden inducir al envío repetitivo de direcciones de referencia erróneas, que no se deban a que el robot está en una posición distinta a la que el navegador cree estar. Estos últimos problemas se incluyen en **LST**. Normalmente los problemas de tipo **WD** se manifiestan en el robot mediante comportamientos cíclicos, si bien esta situación se da en contadas ocasiones.

Robot perdido (LST).

El otro posible problema del navegador que se ha denominado **LST**, se presenta cuando el robot cree estar en una posición distinta a la que en realidad está y esto conlleva a que el robot no pueda llegar a su objetivo.

Inexistencia de camino alternativo (NNA)

NNA Sirve en navegación para distinguir entre la situación en la cual existe un camino alternativo para llegar al objetivo de aquella en la que no existe alternativa. En sí no constituye un problema, pero se ha añadido dicho componente dado que, el conocimiento de esta situación proporciona información interesante a la hora de tomar decisiones.

7.4.1.2 Independencia de los componentes (reducción de estados)

Para simplificar el modelo de estados, se considera la mayor parte de los componentes independientes entre sí. Es decir, que la ocurrencia de camino no navegable es independiente de los problemas hardware, etc. Esto es cierto en todos los casos con excepción de **LST**, dado que problemas en la percepción pueden llevar a que el robot se pierda. Ello no quiere decir que no se tengan en cuenta combinaciones de diferentes problemas, como se verá más adelante. Dichas consideraciones van a simplificar enormemente la tarea de especificación del modelo.

Para constatar la necesidad de considerar combinaciones de situaciones (componentes), supóngase que existe una acción de verificar el camino y éste se encuentre cortado por algún tipo de obstáculo no detectable por los sensores. Considerando solamente la situación de camino no navegable (**NNP**) podría ocurrir lo siguiente: En un principio, la probabilidad de que el camino sea no navegable va aumentando, debido al mensaje del monitor de posición, hasta llegar a un punto en que la probabilidad de **NNP** se hace lo suficientemente grande para llevar al decisor a tomar la determinación de verificar el camino. Como el obstáculo es no detectable, la observación de espacio libre en el camino reduce la probabilidad de que el camino esté bloqueado. Con lo cual, se decide intentar de nuevo partiendo de la misma situación del principio y, por tanto, el robot nunca llegaría a salir de tal situación. Sin embargo, este problema no llega a darse si existe otro componente (**PP**) que, como puede verse en el apéndice C, su probabilidad no disminuye al verificar el camino. Esto llevaría a un punto donde el problema no sería solamente **NNP** sino **NNP & PP** y, en esta situación, el decisor opta por intentar un nuevo camino (ver apéndice D).

El número total de estados es pues $2^{|\text{componentes}|} = 2^7 = 128$ estados.

7.4.2 Acciones

La selección de las acciones está ligada a la selección de estados, como se muestra en la tabla 7.2.

Tabla 7.2 Acciones

acciones	descripción
FP	Acción nominal: seguir el camino planeado (<i>Follow Path</i>)
NP	Actualizar la probabilidad de bloqueo y solicitar un camino nuevo (<i>New Path</i>)
RL	Intentar calcular la posición del robot usando otras alternativas (<i>Relocalize</i>)
GO	Buscar un espacio libre en el camino y cruzarlo si es posible (<i>Go through Opening</i>)
MA	Alejarse de la posición actual (<i>Move Away</i>)
SH	Resolver problemas hardware (<i>Solve Hardware</i>)
GU	Dar esta tarea como imposible de hacer y pasar a la siguiente (<i>Give Up</i>)

Seguir camino planeado (FP)

Esta es la acción que se debería ejecutar cuando el robot se encuentra en el estado nominal. Consiste simplemente en que el robot debe seguir el camino planeado por el planificador de trayectorias. Para ello, como se explica en el capítulo 5, el navegador indica en cada momento al módulo reactivo la dirección a seguir y éste tratará de moverse en esa dirección, evitando los obstáculos que encuentre a su paso. El navegador mantiene una distribución de probabilidades de posición del robot, tal y como se describe en [Simmons 1995]. Por defecto, al arrancar una tarea de navegación, se supone que todo funciona correctamente y comienza a ejecutar **FP** de forma continua hasta que el supervisor decida realizar una acción diferente.

Nuevo camino (NP)

Una posible forma de resolver algunos problemas de navegación, tales como el de camino no navegable o incluso problemas reactivos es usando un camino alternativo si este existe. **NP** solicita un camino nuevo al planificador de trayectorias, aumentando la probabilidad de que un arco del anterior camino (el arco con problemas) se encuentre bloqueado. Esto provoca que el coste de atravesar ese arco sea mucho mayor, con lo cual, si hay un camino alternativo de coste similar, el planificador cambiará la trayectoria a seguir. Como la probabilidad de arco cortado se va acumulando, si no existe una alternativa o es muy costosa, cada vez que se reintente se aumenta la probabilidad de arco cortado, llegando al valor máximo (1) al cabo de varios intentos. En este caso, el planificador ya no podrá seguir usando dicho arco al calcular la próxima trayectoria.

Localización (RL)

Usando las lecturas de los sensores, busca aquella o aquellas posiciones del plano donde se pueden obtener unas medidas similares. Se trata pues de buscar una posición (x,y,q) , donde x e y son las coordenadas cartesianas de la posición del robot en centímetros, con respecto a un sistema de referencia previamente definido, y q es la orientación del robot en radianes.

La solución propuesta en esta tesis se basa en la aproximación presentada en [Raimundez 1998] con algunas modificaciones en la función de adecuación (*fitness*) y en el procedimiento de búsqueda.

Si el robot está en una posición que no es la que el navegador posee, se puede recurrir a dos fuentes de información para calcular su posición: Por una parte, se podría usar el conocimiento de las posiciones donde ha estado anteriormente y los movimientos que ha hecho para saber dónde se encuentra ahora. Así se obtendría una posición aproximada, debido a los errores en la odometría. De todas formas, esta información ya es tenida en cuenta por el navegador. Por otro lado, se pueden usar datos sensoriales para reconocer su posición, de idéntica forma a como lo haría una persona si se le tapan los ojos y la se deja en un sitio aleatorio de un edificio. En este caso, los ojos del robot son los sensores y, al igual que una persona trata de encontrar aquel sitio de los que tiene en su memoria que mejor coincide con lo que está viendo, el robot debe encontrar la posición en el plano cuyas lecturas sensoriales se adecúan mejor con lo que el robot está viendo en estos momentos.

Queda por decidir la forma de llevar a cabo dicha búsqueda. El espacio de búsqueda es tridimensional (x,y,q) , donde el rango de las dos primeras variables

depende de la dimensión del plano, o un área del mismo, en donde se desea ubicar el robot, y la tercera es la orientación, medida en radianes, que varía entre cero y 2π . Se necesita una función $F(x_b, y_b, q_b)$ (función de adecuación) que cuantifique cómo se ciñen las lecturas sensoriales obtenidas por el robot a las que se esperaría tener para cada posición (x_b, y_b, q_b) del espacio.

En [Raimundez 1998] se propone una medida de la función de adecuación basada en la distancia de los puntos obtenidos por la lectura de sensores⁶ a los segmentos del plano. Para evaluar la adecuación de una posición (x_b, y_b, q_b) , se sitúan las lecturas sobre el plano suponiendo el robot en esa posición y se calcula la media de la distancia de cada uno de esos puntos al segmento más cercano. Aunque es una aproximación bastante acertada que puede funcionar en la mayor parte de los casos, esta aproximación ocasiona falsos resultados al no tener en cuenta que algunas líneas representan objetos ocultos. En la figura 7.2 se puede ver un ejemplo de dicho problema, donde se presentan dos casos que producirían igual valor de la función de adecuación, aunque las lecturas han sido proporcionadas por el robot en la situación del caso *a* y son claramente distinguibles. Las estrellas representan las lecturas de los sensores proporcionadas por el robot y las líneas discontinuas las distancias para cada punto.

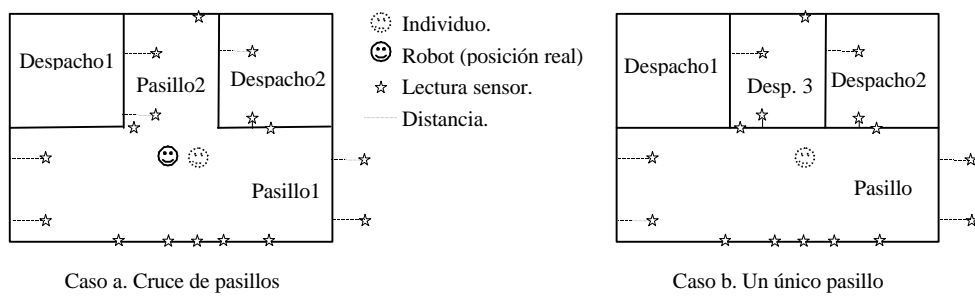


Figura 7.2 Dos situaciones diferentes que producen el mismo valor de la función de adecuación calculada según [Raimundez 1998].

⁶ En [Raimundez 1998] se usan lecturas laser mientras que aquí se parte de lecturas sonar.

Para el algoritmo de búsqueda empleado en este trabajo, la función de adecuación de un individuo (x_i, y_i, q_i) se obtiene calculando las lecturas de los sensores del individuo mediante simulación, suponiendo el robot en la posición del individuo y, a continuación, se evalúa la función de adecuación como la media de las distancias entre las lecturas obtenidas mediante simulación y las proporcionadas por el robot. En la figura 7.3 se muestra un ejemplo donde, al igual que en el caso anterior, los círculos representan las lecturas simuladas y las líneas discontinuas las distancias para cada punto.

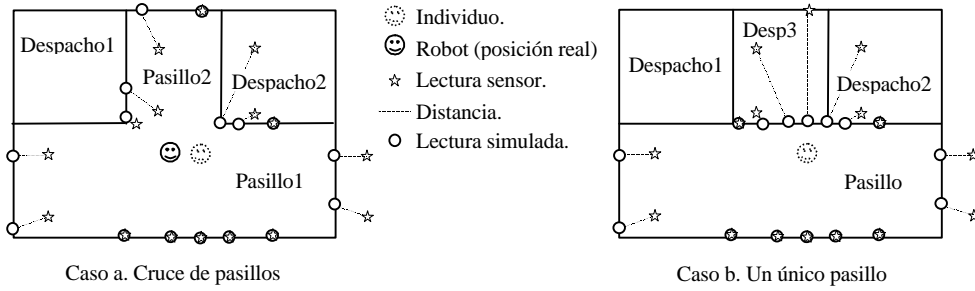


Figura 7.3 Cálculo de la función de adecuación (nuevo método).

Se podrían usar las lecturas del láser junto con las de los sonar, no obstante, sólo se han usado las 24 lecturas de los sensores sonar para minimizar el número de cálculos. Además el láser situado en Xavier tiene un ángulo de apertura muy reducido (15 grados) y proporciona, por tanto, información de un sector muy pequeño del entorno, a menos que se haga un barrido girando el robot. Denotando por s_i las lecturas sonar simuladas y l_i a las lecturas proporcionadas por el robot, la función de adecuación se calcula:

$$F(x_i, y_i, q) = \frac{1}{K} \sum_{i=1}^{24} |s_i - l_i| \quad [Eq.7.4]$$

Hay que recordar que los valores s_i y l_i son las distancias desde el robot al objeto más cercano a éste en la dirección en la que está orientado el sensor. Dicho

objeto tiene que estar en el campo de acción del sensor sonar y debe ser detectable por éste.

Una vez definida dicha función, el objetivo es encontrar la posición (x_b, y_b, q_b) para la cual su valor se hace mínimo. El problema está en que dicha función no es continua y presenta múltiples mínimos locales. Por lo cual muchos de los métodos tradicionales se quedarían en estos mínimos locales y una búsqueda exhaustiva sería imposible por la dimensión del espacio de posiciones asociado. La solución adoptada en esta tesis es una variante de algoritmos evolutivos propuesta en [Sprave 1994].

Pasar a través de un paso estrecho (GO)

El controlador reactivo presenta algunos problemas para pasar por lugares cuya sección transversal no es mucho mayor que el diámetro del robot. Con esta acción se pretende que el robot pase a través de un espacio estrecho. Para ello, primero hay que encontrar un espacio libre suficiente que se ciña lo máximo posible a la dirección de referencia indicada por el navegador. Esto se realiza en dos etapas. En la primera, usando el sensor láser se realiza una búsqueda de espacios libres superiores a un umbral (**WIDE_MIN**) para que el robot pueda pasar y se selecciona una. En la segunda, se trata de guiar al robot a través del espacio libre que mejor se ciña a la dirección que tiene que seguir el robot.

La abertura del haz láser abarca sólo 15 grados pero hay que buscar posibles espacios libres en un rango mayor. El rango de búsqueda seleccionado es de 180 grados en la dirección de referencia (90 a cada lado). Para abarcar con el láser los 180 grados, se van tomando muestras y girando el robot hasta que se obtiene una lista con los puntos de lectura del láser ordenados en sentido de las agujas del reloj, eliminando aquellos puntos que se superpongan en lecturas consecutivas. A continuación, se procesa la lista de puntos leídos por el sonar para generar una lista de espacios libres de acuerdo con el siguiente algoritmo⁷ 7.1.

⁷ Para los extremos son necesarias consideraciones especiales.

Algoritmo 7.1. Detección de espacios libres

Para cada punto P_i en la lista:
Si la distancia al punto siguiente es mayor que $WIDE_MIN$ entonces
CONJUNTO1 = conjunto de puntos anteriores a P_i incluido éste
CONJUNTO2 = conjunto de todos los puntos posteriores a P_i .
Si $d(\text{CONJUNTO1}, \text{CONJUNTO2}) > WIDE_MIN$ entonces
Añadir a la lista de posibles espacios libres
Fin para cada P_i .

Por último, se selecciona de la lista de espacios libres, aquel cuya dirección del punto centro con respecto al robot se parezca más a la dirección de referencia calculada por el navegador.

En la segunda etapa se trata de situar al robot enfrente al centro del espacio libre, si esto es posible, dado que podría existir algún objeto no detectable por el láser y, por último, se establece la dirección de referencia hacia delante para que el robot trate de atravesar ese espacio libre.

Alejarse de un punto (MA)

Si bien en muy rara ocasión, es sabido de algunas situaciones en las cuales el módulo reactivo mantiene al robot dando vueltas indefinidamente, o bien girando entre dos orientaciones también de forma casi indefinida. En algunas ocasiones, al variar un poco las condiciones, el robot abandona dicha situación. Estas situaciones se producen fundamentalmente ante objetos de forma determinada y en posiciones muy puntuales. Es por ello que, la mayor parte de las veces, dicha situación puede ser resuelta sin más que realizar un pequeño desplazamiento en cualquier dirección. En este caso, se selecciona la dirección en la que el robot tenga más espacio libre.

Llamar al operador (SH)

Como no es objetivo de esta tesis el tratar con problemas de hardware, la acción **SH** se limita a enviar un mensaje al operador para que se disponga a arreglar el posible fallo. Tampoco se han diseñado monitores para detectar tales situaciones, si

bien se podrían aplicar técnicas como las descritas en [Ferrell 1994] u otras ya analizadas en el capítulo 3.

Abandonar tarea (GU)

Existen situaciones en las que la tarea encomendada no se podrá llevar a cabo por diferentes motivos. Por ejemplo, todos los caminos que llevan al objetivo se encuentran cortados. Ante estas situaciones es deseable que el robot pase a la siguiente tarea dando la actual por no realizable.

7.4.3 Observaciones

La información acerca del estado del sistema proviene fundamentalmente de dos tipos de observaciones. Por una parte, una serie de monitores al igual que en el caso de la arquitectura en árbol y, por otra parte, también se obtiene información como resultado de alguna de las acciones descritas en el apartado anterior.

En la tabla 7.3 se relaciona a las observaciones con las fuentes que las producen (acciones o monitores).

Tabla 7.3 Observaciones

Observación	Descripción	Fuente
EPS	<i>Error Position (robot Stopped)</i>	Monitor
EPM	<i>Error Position (robot moving)</i>	Monitor
LD	<i>Loop Detected</i>	Monitor
SD	<i>Spinning Detected</i>	Monitor
BO	<i>Blockage Overtaken</i>	Acción (GO)
ALT	<i>Alternative path found</i>	Acción (NP)

7.4.3.1 Monitores

Los monitores se han implementado, al igual que en el caso del modelo determinista, como un proceso independiente que usa **TCA** para coordinar su ejecución junto con las tareas de navegación. Dado que todas las tareas se comunican mediante mensajes a través del **central** de **TCA**, se puede solicitar a dicho módulo una copia de determinado tipo de mensaje o hacer otro tipo de solicitudes, tal como se muestra en [Simmons 1997c]. A continuación se comentan algunas modificaciones que se han introducido en algunos de ellos.

Síntoma: Tiempo excedido

Permanece igual que el realizado en el capítulo anterior para el modelo determinista. Se trata de estimar el tiempo máximo que el robot necesita para realizar la tarea y, a continuación, disparar el monitor una vez transcurrido ese tiempo, si la tarea no ha finalizado.

Síntoma: Error de posición

El funcionamiento de este monitor tampoco ha sido modificado. Se obtiene un residuo (el retraso visto en el capítulo anterior) que indica el progreso en el seguimiento de la ruta planeada. Dicho residuo es la distancia entre la posición real del robot y la posición en la cual el robot debería estar si el robot hubiera seguido la trayectoria preestablecida. El factor a tener en cuenta es la derivada de dicho residuo.

Síntoma: Girando sobre si mismo

El funcionamiento de este monitor se basa en controlar el número de vueltas que el robot da en un mismo sitio. Tampoco ha sido modificado con respecto al modelo determinista.

Síntoma: Excesivo número de vueltas

La detección de este síntoma también permanece invariable con respecto al modelo descrito en el capítulo anterior. Se basa en supervisar los giros de 180 grados (buscando ciertos patrones). Para detectar dicha situación, el monitor solicita de TCA una copia de todas las órdenes que implican una rotación del robot de 180 grados activando un “espía”.

Síntoma: Nivel bajo de la batería

Este monitor, si bien no se ha descartado, su funcionamiento es determinista e independiente de los demás, con lo cual se ha tratado de forma paralela al modelo estocástico. Además, la tarea de este monitor esta más enfocada a la prevención. Su realización no ha sido modificada con respecto a lo que se ha presentado en el capítulo anterior.

7.4.3.2 Observaciones provenientes de las acciones

Como ya se ha comentado, el resultado de algunas acciones también se consideran como observaciones dado que proveen información acerca del estado del sistema.

Sobrepasado espacio estrecho con éxito BO

Al ejecutar la acción **GO** puede ocurrir que el robot haya logrado pasar a través del espacio libre o no haya tenido éxito. Una vez situado el robot frente al centro de la abertura se considera la operación ha tenido éxito y se ejecuta **FP**, algunas veces se ha comprobado que el robot continúa sin poder pasar al otro lado, ya sea porque el espacio libre es demasiado estrecho o por problemas con determinados tipos de objetos y esquinas.

Debido al sistema de navegación basado en probabilidades, es bastante difícil conocer la posición exacta del robot, con lo cual se considera un resultado de **GO** exitoso (**BO**) si se detecta un espacio libre suficiente para pasar (aunque puede ser que existan objetos no detectados por el láser) en lugar de situar el robot al otro lado.

Existencia de camino alternativo ALT

Cada vez que se solicita al planificador de trayectorias un nuevo camino, éste indica si hay alternativas. En caso positivo, dicha opción será siempre válida, pues el robot, en teoría, podría volver al punto de partida y desde allí seleccionar un camino alternativo. Sin embargo, si no hay camino alternativo, dicha situación no tiene porqué mantenerse pues, a medida que el robot avanza pueden aparecer nuevas alternativas.

7.4.4 Probabilidades de transición

Las probabilidades de transición han sido seleccionadas de acuerdo a la experiencia acumulada por diversas personas que han trabajado con el robot Xavier desde su puesta en funcionamiento en diciembre de 1995 [Simmons 1997b]. Una de las líneas futuras de investigación es desarrollar un método de aprendizaje automático para estas probabilidades.

La función de transición de estados es una función de densidad de probabilidades:

$$T : S \times A \rightarrow \Pi(s)$$

Se denomina $T(s'/a, s)$ a la probabilidad de que el robot finalice en el estado s' cuando se ejecuta la acción a en el estado s . Lo ideal sería obtener dichas probabilidades con algoritmos de aprendizaje automático usando el robot. El principal problema es que se necesitarían bastantes muestras y, dada la baja probabilidad de fallo del robot, implicaría mantener el robot funcionando durante mucho tiempo, lo cual hace inviable dicha aproximación. En su lugar, se pretenden establecer estas probabilidades aprovechando la experiencia obtenida sobre la frecuencia de fallos de Xavier y dejar la parte de aprendizaje automático para futuros desarrollos.

Para determinar las probabilidades de transición, considerando todos los casos, se necesitaría definir $|S| \times |A| \times |S| = 128 \times 7 \times 128 = 114.688$ valores diferentes. Éste número, tan elevado, se reduce considerando que los problemas (componentes) descritos en la figura 7.1 son independientes, lo cual es cierto para la mayor parte de ellos. Para cada componente hay que especificar la probabilidad de que este problema ocurra durante cada una de las posibles acciones, así como la probabilidad de que dicho problema desaparezca. Es decir, para cada componente c y cada acción a hay que especificar:

$$p(c/\bar{c}, a)$$

$$p(\bar{c}/c, a)$$

Las probabilidades que el sistema pase de un estado $s1$ a otro $s2$ viene dada por el producto de las transiciones de cada uno de los estados. Es decir, dado:

$$s1 \rightarrow (\bar{c}_1, \bar{c}_2, \bar{c}_3, \bar{c}_4, c_5, \bar{c}_6, \bar{c}_7)$$

$$s2 \rightarrow (c_1, \bar{c}_2, \bar{c}_3, \bar{c}_4, \bar{c}_5, \bar{c}_6, \bar{c}_7)$$

se tiene que:

$$\begin{aligned} p(s2/s1, a) &= p(\bar{c}_1/c_1, a) \times p(\bar{c}_2/\bar{c}_2, a) \times p(\bar{c}_3/\bar{c}_3, a) \times p(\bar{c}_4/\bar{c}_4, a) \times p(c_5/\bar{c}_5, a) \times \\ &\quad \times p(\bar{c}_6/\bar{c}_6, a) \times p(\bar{c}_7/\bar{c}_7, a) \\ &= p(\bar{c}_1/c_1, a) \times (1 - p(\bar{c}_2/c_2, a)) \times (1 - p(\bar{c}_3/c_3, a)) \times (1 - p(\bar{c}_4/c_4, a)) \\ &\quad \times p(c_5/\bar{c}_5, a) \times (1 - p(\bar{c}_6/c_6, a)) \times (1 - p(\bar{c}_7/c_7, a)) \end{aligned}$$

De esta forma, el número de probabilidades de transición se reduce a $|C| \times |A| \times 2 = 7 \times 7 \times 2 = 98$. No obstante, la simplificación de independencia no siempre se puede aplicar, por lo que el modelo posee la posibilidad de añadir probabilidades de transición entre estados como combinaciones de componentes, tal y como se puede observar en la sintaxis de descripción del modelo descrita en el apéndice C.

7.4.5 Probabilidades de observación

La probabilidad de obtener una determinada observación dependerá del estado del robot y de la acción que esté llevando a cabo. En este caso sólo interesa el estado final, si bien para algunos autores las observaciones pueden ser dependientes tanto del estado inicial como del final.

$$O : S \times A \rightarrow \Pi(z)$$

$$o(z/a, s) = P(o^t = z / s^t = s, A^{t-1} = a)$$

Se denomina $o(z/a, s)$ a la probabilidad de observar z cuando se ha ejecutado a , sabiendo que el estado resultante es s . En este caso, el número de observaciones posibles es $|S| \times |A| \times |Z| = 57.344$ pero, al igual que en el caso anterior, dicho número se puede reducir considerablemente aprovechando algunas características del proceso.

Tabla 7.4 Observaciones posibles en cada acción

Acciones	Observaciones posibles			
FP	EPS	EPM	LD	SD
GO	BO			
RL				
NP	ALT			
MA				
GU				
SH				

En primer lugar, hay que constatar que no todas las observaciones son posibles para todas las acciones. Así, **BO** sólo puede producirse cuando se ejecuta la acción **GO**, pues dicha observación es el resultado de esta acción. Por otra parte, los monitores

entran en funcionamiento cuando el navegador envía la orden de empezar a navegar a **CTR** a través de **TCA** y este remite una copia de dicho mensaje al módulo que gestiona los monitores. En la tabla 7.4 se resumen las relaciones de las observaciones que se pueden llegar a detectar para cada una de las acciones.

Otra característica de este proceso que permite reducir el número de probabilidades de observación a determinar es que las componentes de observación (z_{ci}) son independientes unas de otras dado el estado y la acción ejecutada. Es decir:

$$p(z_{ci} / a, s, z_{cj}) = p(z_{ci} / a, s) \leftrightarrow \forall i \neq j$$

Con lo cual, no será necesario especificar las probabilidades de observación para cada posible combinación de las componentes; si bien es necesario tener en cuenta que se pueden dar estas combinaciones pero la probabilidad de una determinada es el producto de las probabilidades de obtener cada componente que la forma:

$$p(EPM, \overline{SD}, BO, \dots / a, s) = p(EPM / a, s) \times p(\overline{SD} / a, s) \times p(BO / a, s) \times \dots$$

Por último, al igual que se hizo en el caso anterior, se pueden especificar dichas probabilidades en función de los componentes del estado, reduciendo de nuevo el número de parámetros para aquellas que sean independientes como se muestra en la sintaxis del archivo de especificaciones del modelo descrito en el apéndice C.

Retraso en las observaciones

En un modelo POMDP estándar, las observaciones obtenidas al final del período ***n-1*** (que son las usadas en el período siguiente para obtener la información de estado) dependen de la acción llevada a cabo durante ese período, del estado actual y del estado final. De forma gráfica, estas relaciones se describen en la figura 7.4 si bien puede expresarse de forma más clara en un diagrama de dependencias de la figura 7.5.

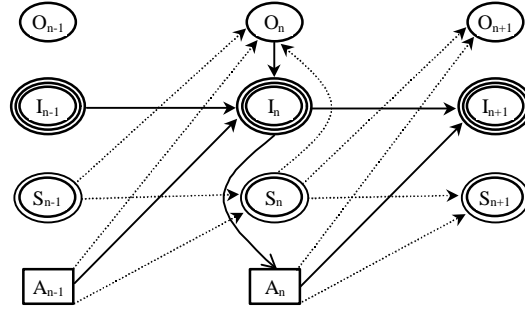


Figura 7.4 Las observaciones dependen de la acción llevada a cabo y de los estados inicial y final.

Por tanto:

- ✓ I_n se calcula al principio del intervalo n y su valor será constante en ese intervalo.
- ✓ A_n es la acción seleccionada al principio del intervalo n y su ejecución se prolonga durante dicho intervalo.
- ✓ S_n es el estado del sistema al empezar el intervalo n cuando se empezó a ejecutar la acción A_n , o lo que debería ser lo mismo, el estado final del intervalo $n-1$.
- ✓ O_n es el conjunto de todas las observaciones.

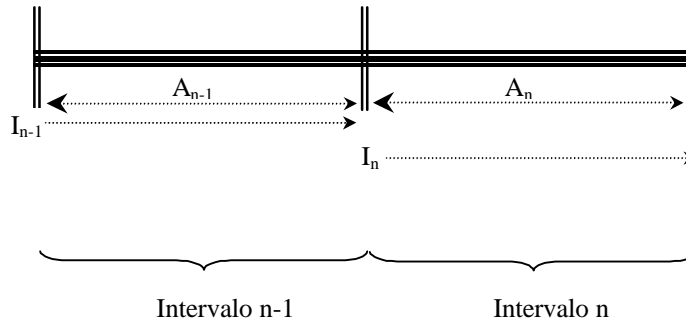


Figura 7.5 Diagrama temporal.

7.4.6 Bonificación

Las medidas a tomar para evaluar el coste de las acciones pueden ser muy diversas. Se puede considerar el tiempo en realizar dicha acción, los recursos consumidos, o cualquier otro criterio definido por el usuario.

Es deseable que la primera de las acciones se produzca en la situación nominal, cuando el robot esta navegando sin problemas, mientras que el resto de las acciones se activaran cuando el sistema tenga que salir de alguna situación de excepción. Hay que resaltar que la correspondencia entre cada situación de excepción y la acción a tomar no es siempre única, ya que es posible salir de alguna situación de excepción con distintas acciones.

Igual que en el capítulo anterior, es bastante improbable que el supervisor establezca la situación en que se encuentra el sistema sin error alguno. No obstante, lo que se pretende no es simplemente reducir la probabilidad de error en la determinación del estado del robot, sino que la penalización por “decisiones erróneas” sea lo más pequeña posible. Al mismo tiempo, la forma de expresar los costes o bonificaciones debe ser sencilla y, a partir de ahí, el programa ha de evaluar la penalización o bonificación necesaria para el modelo.

Las bonificaciones determinan las acciones y situaciones a evitar y cuales son deseables que se produzcan. Se ha asignado un coste a cada acción que en la mayoría de los casos es proporcional al tiempo en realizar dicha acción. Se ha añadido una bonificación a aquellas situaciones en las cuales el robot pasa de una situación de excepción a la situación nominal. Por último, también se premian las acciones que mantienen al robot en correcto funcionamiento siguiendo la trayectoria planeada (apéndice C).

El sistema de bonificación viene dado por una función R :

$$R : SxAxSxZ \rightarrow \mathfrak{R}$$

Que en este caso se ha simplificado a:

$$R : SxAxS \rightarrow \mathfrak{R}$$

La sintaxis del archivo de especificaciones permite indicar un coste para cada acción y considerar situaciones particulares que se sobreescriben a estas penalizaciones.

También es posible establecer bonificaciones, tales como la descrita por la siguiente regla: *para cualquier acción, y para cualquier estado de partida diferente del nominal, siempre que este sea el estado final, se añadirá una recompensa de x* , u otras reglas dependientes de alguno de los componentes del estado, tales como: *si el estado inicial incluye la componente **NNP** y no **NNA** se asignará una penalización x* .

Como se puede observar en el apéndice C, el sistema de recompensas de este modelo es relativamente sencillo y fácil de especificar según las preferencias del usuario.

7.5 El supervisor

Para tomar decisiones sobre la siguiente acción a ejecutar es necesario tener en cuenta información sobre el pasado. Si dicha información no fuese considerada, el robot podría llegar a quedar atrapado en un bucle, realizando una serie de operaciones de forma repetitiva.

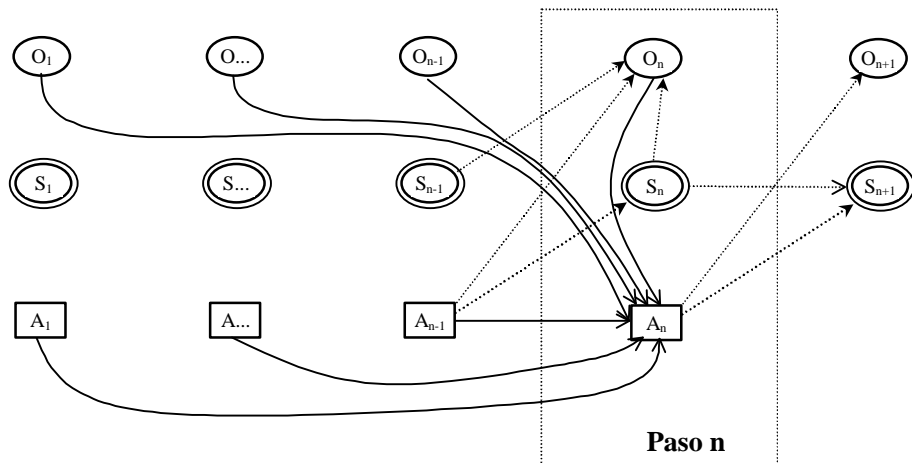


Figura 7.6. Diagrama de dependencias entre observaciones, estados y acciones.

Este problema queda patente con el ejemplo de camino bloqueado por un obstáculo no detectable a través del sistema sensorial. Al no poder avanzar el robot, se disparará el monitor de posición y se ejecutará la acción **GO**. Esta acción encontrará el camino vacío al no detectar el obstáculo, se sitúa al robot enfrente al espacio libre y se considera que ha tenido éxito, pasa a ejecutar **FP** y el proceso vuelve a la situación inicial. Al introducir información sobre el pasado, si el modelo es el adecuado y caracteriza completamente el estado del robot, se tiene en cuenta que dicha situación ya se produjo. Esta historia sobre el pasado viene dada por la *información de estado*.

El tiempo se divide en intervalos (en este sistema los intervalos son de un segundo). Durante el intervalo n se ejecuta una única acción A_n y se obtendrá una observación o_{n+1} que será tomada en cuenta para la selección de la acción que se va a activar en el intervalo $n+1$.

El diagrama de dependencias [Hauskrecht 1996] de este sistema es similar al descrito en la figura 7.6, donde se pone de manifiesto que tanto las observaciones pasadas, como las acciones influyen sobre la decisión de la próxima acción A_n .

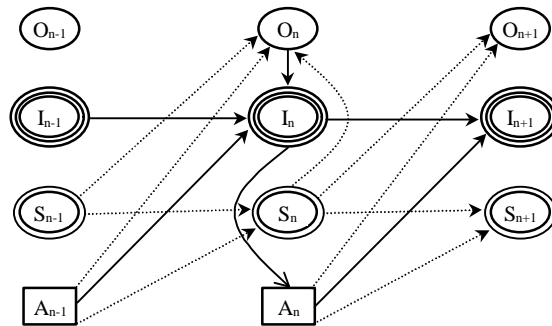


Figura 7.7. Diagrama de dependencias añadiendo la información de estado

En el diagrama de dependencias, las líneas discontinuas representan la interacción “afecta a” en el sistema, mientras que las líneas continuas representan los parámetros a tener en cuenta a la hora de calcular las variables correspondientes.

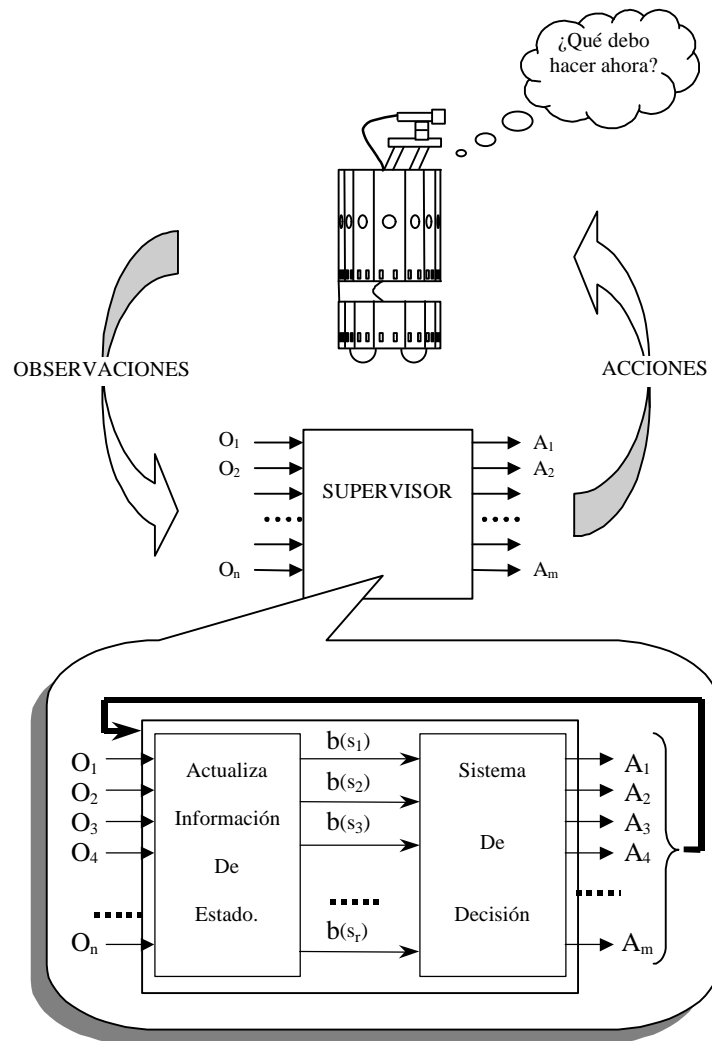


Figura 7.8. Pasos en la toma de decisión:
Actualizar información de estado y decisión.

En un proceso POMDP la información relevante sobre el pasado se sintetiza en un vector (distribución de probabilidades sobre estados) que se denomina *información de estado* (I_n) (ver apéndice A). El diagrama de dependencias resultante se muestra en la figura 7.7, donde se puede observar que la acción a ejecutar en el paso n depende solamente de la información de estado en ese instante.

La toma de decisiones se divide pues en dos pasos (figura 7.8). En la primera fase se actualiza la información de estado. Es decir, se actualiza la distribución de probabilidades entre los estados del sistema que define la información que el robot posee acerca de su estado. En la segunda fase se decide cual es la siguiente acción a ejecutar en función de la *información de estado*.

7.5.1 Actualización de la información de estado

En esta sección se describe como calcular la información de estado para un determinado paso:

$$I_n = t(I_{n-1}, O_n, a_{n-1})$$

Si del diagrama general de dependencias para el paso n se extrae la parte que no influye en la *información de estado*, se obtiene la figura 7.9.

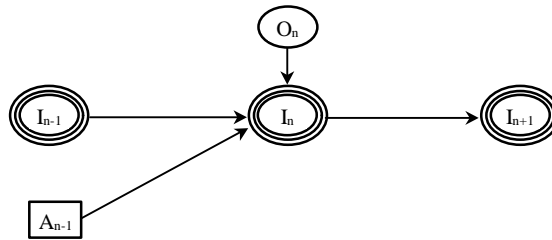


Figura 7.9. Actualización de la información de estado

La *información de estado* (I) se calcula a partir de la que se tenía en el instante anterior, la acción que se ha ejecutado y la observación obtenida.

Una característica muy importante de los procesos POMDP estándar es que la distribución de probabilidades entre los estados \mathbf{b} ($\mathbf{b}(s)$) es la probabilidad de que el

sistema se encuentre en el estado s) es un estadístico suficiente [Hauskrecht 1996] que puede ser usado como información de estado. Es necesario mantener la distribución de probabilidades de estado de la forma más fiable posible. Considérese un paso general t en el que, con una distribución a priori de probabilidades de estado $b^t(s)$, se ejecuta la acción a y se observa z . Ahora hay que obtener la distribución de probabilidades de estado a posteriori b^{t+1} , o lo que es lo mismo, $b^{t+1}(s)$ para cualquier estado s . Para ello, se dispone de las probabilidades de transición entre estados y la probabilidad de observación. Existen, por tanto, dos estadísticos que influyen en el proceso: las probabilidades de transición y las de observación.

Las probabilidades de transición $p(s'/s, a)$ indican la probabilidad de que estando en un estado s y ejecutando una acción a , se finalice en otro estado diferente s' . Para calcular las probabilidades a posteriori para una distribución de probabilidades inicial $b(s)$ y ejecutada la acción a se aplicaría la fórmula de Bayes junto con la probabilidad marginal:

$$\begin{aligned} b^{t+1}(s') &= P(s' / b^t, a) = \sum_s p(s', s / b^t, a) \\ &= \sum_s p(s' / s, b^t, a) p(s / b^t, a) = \sum_s p(s' / s, b^t, a) p(s / b^t) \\ &= \sum_s p(s' / s, b^t, a) b^t(s) \end{aligned}$$

donde s' es un estado en $t+1$ y s es un estado en t .

Por otra parte, hay que tener en cuenta las observaciones obtenidas y las probabilidades de obtener dicha observación para cada estado, resultando una expresión algo más compleja:

$$\begin{aligned} b^{t+1}(s') &= P(s' / b^t, a, z) \\ &= \frac{p(s', b^t, a, z)}{p(b^t, a, z)} = \frac{p(z / s', b^t, a) p(s', b^t, a)}{p(z / b^t, a) p(b^t, a)} \end{aligned}$$

$$\begin{aligned}
 b^{t+1}(s') &= \frac{[p(z/s', b^t, a)p(s'/b^t, a)]p(b^t, a)}{\left[\sum_s \sum_{s''} p(z, s, s''/b^t, a)\right]p(b^t, a)} = \frac{p(z/s', b^t, a)p(s'/b^t, a)}{\sum_s \sum_{s''} p(z, s, s''/b^t, a)} \\
 &= \frac{p(z/s', a)p(s'/b^t, a)}{\sum_s \sum_{s''} p(z/b^t, a, s, s'')p(s, s''/b^t, a)} = \frac{p(z/s', a)\sum_s p(s'/b^t, a, s)p(s/b^t, a)}{\sum_s \sum_{s''} p(z/a, s'')p(s, s''/b^t, a)} \\
 &= \frac{p(z/s', a)\sum_s p(s'/a, s)p(s/b^t)}{\sum_s \sum_{s''} p(z/a, s'')p(s''/s, b^t, a)p(s/b^t, a)} = \frac{p(z/s', a)\sum_s p(s'/a, s)p(s/b^t)}{\sum_s \sum_{s''} p(z/a, s'')p(s''/s, a)p(s/b^t)}
 \end{aligned}$$

Identificando en la ecuación anterior las probabilidades de transición, las de observación y la distribución de estados, se obtiene:

$$b^{t+1}(s') = \frac{o(s, s', z) \sum_s t(s, a, s') b^t(s)}{\sum_{s, s'} o(a, s'', z) t(s, a, s'') b^t(s)} \quad [Eq. 7.5]$$

Usando la anterior ecuación para cada uno de los estados se obtiene la distribución de probabilidades para el paso $t+1$ (b^{t+1}), dadas las de t (b^t), conocida la acción ejecutada a y la observación obtenida z .

7.5.2 Planificación y toma de decisiones

Dado que la información de estado es un estadístico suficiente, mantiene toda la información relevante para que el sistema de decisión determine la secuencia óptima de acciones a ejecutar. Una vez determinada, en el apartado anterior, la forma de actualizar esta información a cada paso, se procederá a la selección de un mecanismo para tomar la decisión acerca de qué acción a_n se ha de ejecutar en el siguiente intervalo:

$$a_n = \text{decisor}(I_n)$$

Con el esquema de bonificaciones definido por el usuario, el objetivo es encontrar un plan $p_n = \{m_1, m_2, m_3, \dots, m_i, \dots, m_n\}$ para obtener una bonificación media máxima. Dicho plan ha de definir a cada paso una política m_i que relacione cada posible valor de la información de estado I_i con una acción A_{m_i} .

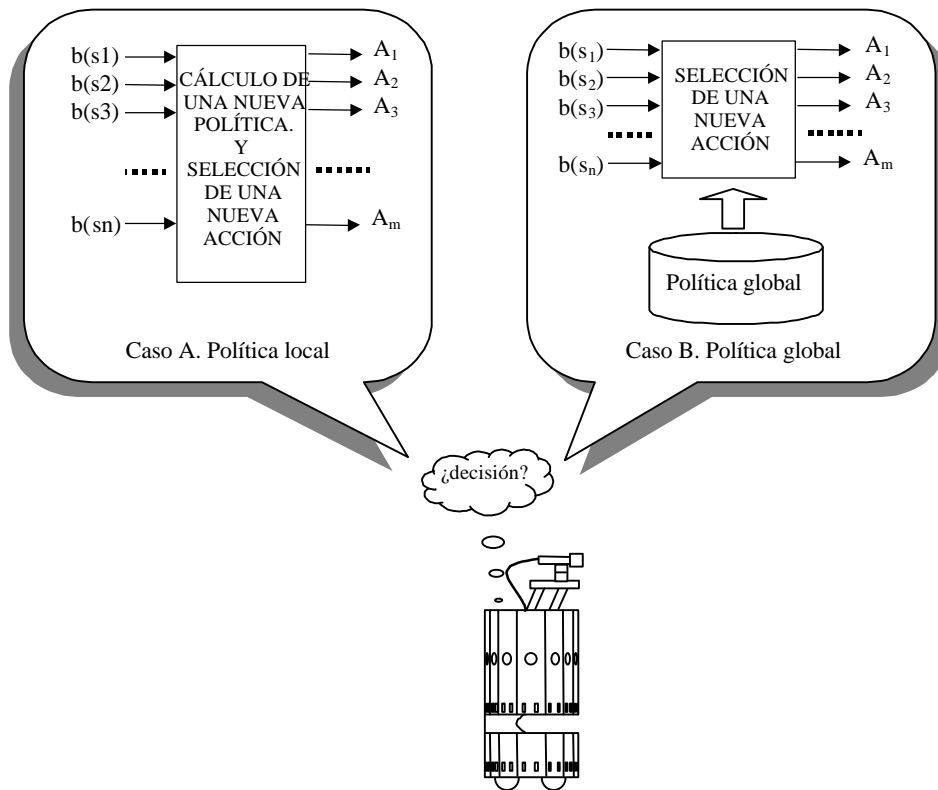


Figura 7.10 La política de decisión puede ser local o global

El problema se puede plantear de dos formas diferentes como se muestra en la figura 7.10. La primera se basa en calcular un plan local a cada paso. Es decir, determinar cuál es la acción a ejecutar para la información de estado actual. Usando la segunda forma, se calcula un plan p estático general fuera de línea (*off-line*) para todos los posibles valores de la información de estado, de forma que cada vez que el sistema tenga que tomar una decisión se busca en la política general p la acción asignada para el valor actual de la información de estado.

La ventaja del plan fuera de línea es la rapidez de respuesta en el bucle de control pues todos los cálculos se realizan a priori (fuera de línea). En la planificación en línea sin embargo, el tiempo de planificación suele ser un punto crítico ya que hay que calcular un nuevo plan a cada paso. Su ventaja, sin embargo, radica en que se puede aplicar en sistemas dinámicos donde los costes (bonificaciones) y demás parámetros del modelo pueden variar con el tiempo.

7.5.2.1 Política local

La política local es un plan a medida para la situación en la que se encuentra actualmente el sistema. Dicha situación viene dada por la información de estado. Se trata en definitiva de, partiendo de la situación actual, encontrar el plan (secuencia de acciones) que proporcione en media la mayor bonificación.

Esto se puede ilustrar con un ejemplo sencillo cuyo planteamiento puede verse en el apéndice B. Un robot está navegando por un pasillo que puede tener dos estados **NN** (No Navegable) o **NV** (Navegable). El robot puede decidir entre seguir el camino **SC**, verificar el camino **CC** o elegir un camino alternativo **CA**. Además, como consecuencia de **CC**, pueden obtenerse dos observaciones diferentes: **BD**(Camino No Navegable) ó **BN** (Camino Navegable). Se obtiene, por tanto:

- ✓ Estados: **NN**, **NV**.
- ✓ Acciones: **SC**, **CC**, **CA**.
- ✓ Observaciones: **BD**, **BN**.

Estableciendo también un sistema de bonificación/penalización tal como el que se muestra en la tabla B.1. Las probabilidades de transición se muestran en la tabla B.2.

Se presupone en principio que no hay incertidumbre en las observaciones y que la distribución de probabilidades sobre estados es inicialmente 0,7 para el estado **NV** y 0,3 para el **NN**. Desarrollando el conjunto de situaciones posibles a un paso vista en el futuro, se obtiene la situación de la figura 7.11.

Las probabilidades a posteriori se han calculado de acuerdo con la ecuación 7.5, que en este ejemplo se puede simplificar bastante. Al no haber incertidumbre en las observaciones, el estado después de ejecutarse CC viene dado por la observación, mientras que para las demás acciones se puede calcular usando la ecuación 7.6.

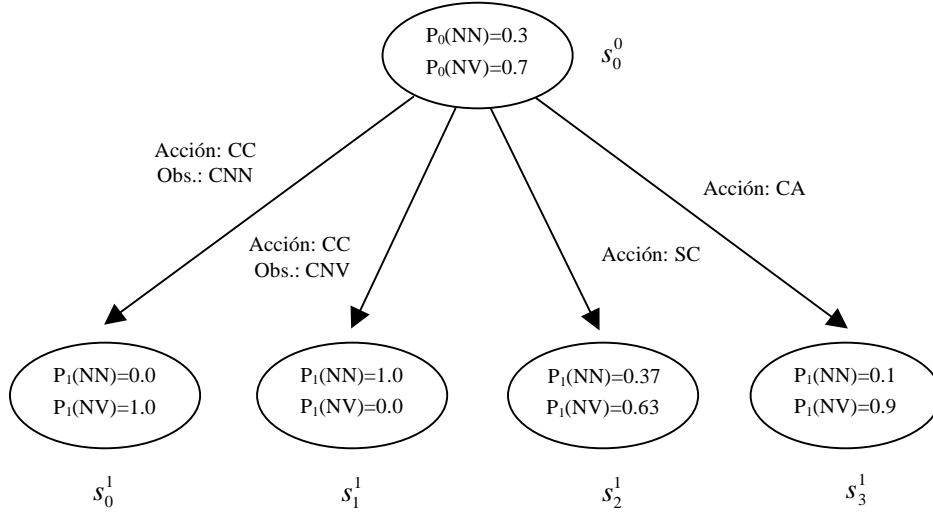


Figura 7.11 Posibilidades de Información de estado a un paso vista.

$$p^n(s' / b^{n-1}, a) = \sum_s p(s' / a, s) b^{n-1}(s) \quad [Eq.7.6]$$

Las recompensas para cualquier acción a se pueden calcular a partir de la tabla B.1 teniendo en cuenta la acción y las probabilidades iniciales, ya que el sistema de bonificación es independiente de las observaciones y probabilidades de estado final:

$$R_a = P(NN)R(a / NN) + P(NV)R(a / NV) \quad [Eq.7.7]$$

Se obtendría una bonificación de -23 para **SC**, -1 para **CC** y -67 para **CA**, con lo cual, la acción con mayor bonificación a un paso vista es **CC**.

Repitiendo el mismo proceso, se podría estimar la bonificación a n pasos en el futuro creando un árbol de n pasos con la bonificación en las hojas. La acción a seleccionar en ese caso sería aquella en cuyas ramas se encuentre la hoja con mayor bonificación. También se puede añadir un factor de descuento r a la bonificación de las acciones por llevarlas a cabo demasiado tarde. El modelo así descrito es un POMDP de horizonte finito (n) con un factor de descuento r .

El problema es que el factor de ramificación es exponencial con el numero de acciones y observaciones, con lo cual, para el modelo real se hace intratable un planificador de este tipo al tener que hacerse a cada paso. En el apéndice B se presenta un ejemplo más completo del problema.

7.5.2.2 Política global

En el segundo caso, se trata de encontrar una política “off line” de forma que, en tiempo de ejecución del sistema, el decisor sólo tenga que consultar esta política para tomar decisiones.

Así, siguiendo con el mismo ejemplo usado en la política local, la política global depende de la información de estado, que en este caso viene dada por la probabilidad de que el sistema esté en un estado. En el apéndice B se muestra un ejemplo que aclara los conceptos aquí expuestos.

Como se describe en el apéndice A, la bonificación total de un plan $p_n = \{m_1, m_2, m_3, \dots, m_n\}$ de n pasos viene dada por la función valor V_n , que se puede calcular de forma recurrente:

$$V_1^{p_n}(I_1) = r(I_1, m(I_1)) + r \sum_{I_2} P(I_2 / I_1, m(I_1)) V_2^{p_{n-1}}(I_2) \quad [Eq. 7.8]$$

donde I_1 es la información de estado para el primer paso, m_k es la política de control para el paso n que asigna una acción para cada valor posible de la información de estado. El sumatorio se aplica para todos los posibles valores de información de estado en el siguiente paso I_2 . Ello quiere decir que el número de posibles informaciones de estado es igual al número de posibles observaciones. $r(I_1, m(I_1))$ es la bonificación esperada de ejecutar la acción A_I asignada a I_1 por la política m . Es decir, $m(I_1) = A_I$. Esta bonificación puede ser calculada a través de la ecuación 7.9.

$$r(I_1, m(I_1)) = \sum_{s \in S} r(s, m(I_1)) p(s / I_1) \quad [Eq. 7.9]$$

El modelo de horizonte finito no parece adaptarse demasiado bien a nuestro problema, pues el número de pasos en los que el posible fallo quedará resuelto es desconocido. Cabe recordar que no se está planificando una serie de pasos para lograr que el robot llegue a una posición destino, el objetivo ahora es mantener al robot funcionando de forma robusta. Si bien sería más completo considerar al mismo tiempo la planificación de trayectorias junto con la supervisión y tratamiento de situaciones de excepción, dicho planteamiento se hace intratable por su complejidad. Además, otro de los objetivos es que el módulo de detección y recuperación de fallos sea independiente de los demás en lo posible, y que sea fácil de añadir nuevas acciones o monitores para tratar con nuevos problemas según vayan surgiendo.

Dado que el objetivo es mantener al robot funcionando de forma correcta el máximo tiempo posible, se puede considerar un sistema de horizonte infinito con política estacionaria en el cual el robot siempre tiene alguna tarea por hacer. El objetivo es ahora encontrar una política \mathbf{p} que relacione cada posible valor de la información de estado \mathbf{I} con una acción A_I . Tarea que sigue siendo bastante compleja como veremos a continuación.

Tratándose de una política estacionaria \mathbf{p} , la ecuación 7.8 pasa a ser:

$$V_1^p(I_1) = r(I_1, m(I_1)) + r \sum_{I_2} P(I_2 / I_1, m(I_1)) V_2^p(I_2) \quad [Eq. 7.10]$$

Como se ha visto anteriormente, dada la información de estado en el paso actual I_1 , los distintos valores para el paso siguiente aplicando una determinada política dependen de las posibles observaciones, con lo cual se puede realizar el sumatorio sobre el conjunto de observaciones:

$$\begin{aligned} V_1^p(I_1) &= r(I_1, m(I_1)) + r \sum_{o \in Z} P(o / I_1, m(I_1)) V_2^p(I_2) \\ &= \sum_{s \in S} r(s, A_1) p(s / I_1) + r \sum_{o \in Z} P(o / I_1, A_1) V_2^p(I_2) \end{aligned}$$

$$\begin{aligned}
 V_1^p(I_1) &= \sum_{s \in S} r(s, A_1) p(s / I_1) + r \sum_{o \in Z} \sum_{s_1 \in S} \sum_{s_2 \in S} P(o, s_1, s_2 / I_1, A_1) V_2^p(I_2) \\
 &= \sum_{s \in S} r(s, A_1) p(s / I_1) + \\
 & r \sum_{o \in Z} \sum_{s_1 \in S} \sum_{s_2 \in S} P(o / s_1, s_2, I_1, A_1) p(s_1, s_2 / I_1, A_1) V_2^p(I_2) \quad [Eqs. 7.11] \\
 &= \sum_{s \in S} r(s, A_1) p(s / I_1) + \\
 & r \sum_{o \in Z} \sum_{s_1 \in S} \sum_{s_2 \in S} P(o / s_1, s_2, I_1, A_1) p(s_2 / s_1, I_1, A_1) p(s_1 / I_1, A_1) V_2^p(I_2) \\
 &= \sum_{s \in S} r(s, A_1) p(s / I_1) + \\
 & r \sum_{o \in Z} \sum_{s_1 \in S} \sum_{s_2 \in S} P(o / s_1, s_2, A_1) p(s_2 / s_1, A_1) p(s_1 / I_1) V_2^p(I_2)
 \end{aligned}$$

donde A_I es la acción determinada por la política $\mathfrak{m}(I_I)$.

Identificando términos con los proporcionados en el modelo se obtiene:

$$V_1^p(I_1) = \sum_{s \in S} r(s, A_1) b(s) + r \sum_{o \in Z} \sum_{s_1 \in S} \sum_{s_2 \in S} P(o / s_1, s_2, A_1) b(s_1) p(s_2 / I_1, A_1) V_2^p(I_2)$$

Para calcular la política óptima hay que calcular la acción A_I que maximize la anterior función valor. Es decir:

$$\mathfrak{m}(I_1) = \max_{A_I \in A} \left[\sum_{s \in S} r(s, A_1) b(s) + r \sum_{o \in Z} \sum_{s_1 \in S} \sum_{s_2 \in S} P(o / s_1, s_2, A_1) b(s_1) p(s_2 / I_1, A_1) V_2^p(I_2) \right]$$

Se han propuesto distintos algoritmos para la obtención de la política óptima exacta. Entre ellos se puede destacar el algoritmo de testigo (*witness algorithm*) [Littman 1994], el algoritmo de poda incremental (*incremental pruning*) [Cassandra 1997] [Zhang 1996], basados en métodos de aprendizaje [Littman 1995] y también otros que se analizan en [Cassandra 1998]

No obstante, ninguno de estos métodos es aplicable a sistemas con un número de estados, acciones y observaciones relativamente alto. Ello se debe, en la mayor parte de los casos, al tiempo que se necesitaría para obtener la solución y la representación de ésta.

Aquí se han usado soluciones aproximadas similares a las sugeridas por diversos autores en otras aplicaciones diferentes a la robótica. Estas aproximaciones, si bien no garantizan una política óptima, sí producen buenos resultados en tiempos aceptables.

La dificultad de calcular la política óptima para este problema hace muy difícil evaluar las soluciones obtenidas por las aproximaciones heurísticas. Al igual que en cualquier evaluación empírica donde no se tiene acceso a la solución óptima, se debe poner especial cuidado a la hora de evaluar los resultados y sacar conclusiones. A continuación se describen los algoritmos que se han implementado. En el próximo capítulo se describirán algunos resultados de comparaciones entre las distintas soluciones aquí presentadas.

7.5.2.3 Estrategias de decisión heurísticas.

En este apartado se describen las técnicas de decisión utilizadas en este trabajo. Se han tomado como referencia para contrastar los resultados de los métodos aquí elaborados otras técnicas basadas en métodos presentados en diversas publicaciones [Simmons 1995] [Hauskrecht 1996] [Cassandra 1998].

Dado que se está considerando un modelo estacionario (de otra forma habría que calcular un plan a cada paso, lo cual es inviable), la solución del POMDP proporcionaría una función que asigna a cada posible distribución de probabilidades de estado, es decir a cada valor de la información de estado, una acción. Se calcularía, por lo tanto, la solución al POMDP fuera de línea. En cada paso durante la navegación, después de actualizar las probabilidades de los estados, se selecciona la acción correspondiente a esas probabilidades. El esquema de dicho funcionamiento puede verse en la figura 7.10. Como ya se ha comentado, la solución exacta del POMDP no es factible hoy en día, si bien se continúa invirtiendo considerable esfuerzo en este campo con objeto de encontrar algoritmos más rápidos. Las soluciones aproximadas aquí propuestas siguen el mismo esquema de la figura 7.8.

Existen diversas estrategias de decisión heurísticas que aproximan la solución óptima del POMDP. Algunas de ellas aprovechan características especiales del modelo a tratar [Washington 1998] que simplifican los cálculos pero no son aplicables más que en cierto tipo de sistemas. Otras por el contrario, son aproximaciones basadas en COMDP o métodos de aprendizaje [Washington 1996] [Kaelbling 1996].

La idea común bajo algunos de estos métodos es resolver el problema como si se tratara de un COMDP para obtener una política general $G_{co}(b)$. A diferencia del POMDP, la política así obtenida asigna a cada estado una acción y garantiza que esa política sea óptima si se conoce con certeza en todo momento el estado del robot.

No se dispone pues de una acción deseada para cada distribución de probabilidades de estado (información de estado), sino para cada estado. Sin embargo, durante la navegación no se sabe con certeza el estado del robot y sólo se dispone de la distribución de probabilidades sobre posibles estados. La acción a ejecutar se selecciona en función de la política $G_{co}(b)$ calculada por el COMDP y la información de estado actual (b). La forma de realizar esta selección es lo que diferencia a los métodos aproximados que a continuación se presentan.

La solución del COMDP se puede llevar a cabo usando los algoritmos de iteración de política o iteración de la función valor descritos en el apéndice A. Existen otras soluciones basadas en aprendizaje por refuerzo [Cassandra 1998] que no han sido consideradas aquí tanto por el tiempo de aprendizaje, como por la dificultad de obtener datos suficientes para llevar a cabo el aprendizaje.

Estado más probable MLS (*Most Likely State*)

Considerando que el sistema mantiene la información de estado, la idea más sencilla es simplemente actuar como si el sistema se encontrara en el estado más probable. Es decir, seleccionar la acción óptima (dada por la solución al COMDP $G_{co}(b)$ ya calculada) para el estado con mayor probabilidad. Se han hecho por lo tanto dos simplificaciones para aplicar este método. La primera que se ha considerado el sistema totalmente observable para calcular la política óptima, y la segunda hipótesis es asumir que el sistema se encuentra en el estado más probable. Los efectos de ambas simplificaciones son muy difíciles de evaluar y dependen, en gran medida, del modelo a controlar. Se presentarán más adelante algunos resultados obtenidos para el modelo que nos ocupa.

Se define la política de control heurístico del estado más probable **MLS** (*Most Likely state*) $G_{MLS}(b)$ de la siguiente forma:

$$\Gamma_{MLS}(b) = \Gamma_{COMDP}(\arg \max_s b(s)) \quad [Eq.7.12]$$

En caso de haber dos estados con igual probabilidad, este método escogerá aleatoriamente uno de ellos.

Votación AV (*Action Voting*)

La política de decisión por votación **AV** (*Action Voting*) $G_{AV}(b)$ se basa en calcular una distribución de probabilidades sobre las acciones $P_a(b)$ a partir de la distribución de probabilidades sobre estados $b(s)$ y la política óptima para el COMDP $G_{Co}(b)$. La probabilidad de cada acción será la suma de las probabilidades de aquellos estados que tengan esa acción como óptima en la política $G_{Co}(b)$. Es decir:

$$P_a(b) = \sum_s b(s) d(\Gamma_{Co}(s), a) \quad [Eq.7.13]$$

Donde la función $d(x,y)$ tiene valor cero para todos los valores de x e y excepto en el caso que x e y sean iguales, en cuyo caso valdrá uno:

$$\delta(x,y) = \begin{cases} 1 & \text{si } x == y \\ 0 & \text{en caso contrario} \end{cases}$$

A partir de la distribución de probabilidades de las acciones $P_a(b)$, el decisor seleccionará aquella acción con mayor probabilidad:

$$\Gamma_{AV}(b) = \arg \max_a P_a(b) \quad [Eq.7.14]$$

Esta estrategia fue usada por primera vez por Simmons y Koenig [Simmons 1995] en el sistema de navegación descrito en el capítulo quinto de la tesis. Si bien, en esta primera aproximación se usa un planificador heurístico en lugar de resolver el COMDP subyacente para calcular la acción asociada a cada estado.

Método Q_MDP

Las anteriores soluciones sólo tienen en cuenta la acción asociada a cada estado de acuerdo con la política $G_{Co}(b)$ pero, al mismo tiempo que se calcula esta, se puede calcular la bonificación esperada para cada una de las acciones en cada uno de los estados. Q_MDP es similar a AV pero varía la forma de calcular la distribución de probabilidades sobre el espacio de acciones. En este caso se pondera la probabilidad de que el sistema se encuentre en ese estado por la bonificación esperada:

$$P_a(b) = \sum_s b(s) V^a(s) \quad [Eq.7.15]$$

donde $V^a(s)$ es la función valor resultante de ejecutar a cuando el robot está en el estado s y, a partir de entonces, se aplica siempre la política óptima suponiendo el estado conocido ($G_{CO}(b)$). Esto quiere decir que, si el sistema fuera completamente observable a partir de la siguiente acción, esta estrategia de decisión sería óptima. La desventaja es que, precisamente esta consideración lleva al sistema a veces a ejecutar una acción con coste bajo, puesto que la incertidumbre desaparecerá a partir del próximo estado.

Al igual que en el caso anterior (votación), a partir de la distribución de probabilidades de las acciones $P_a(b)$, el decisor seleccionará aquella acción con mayor probabilidad:

$$\Gamma_{AV}(b) = \arg \max_a P_a(b) \quad [Eq.7.16]$$

Modo Dual

Los métodos anteriores no tienen en cuenta las acciones reductoras de entropía, es decir acciones cuyo objetivo es obtener información acerca del estado del sistema. Si existe una acción de este tipo, aunque tenga un coste reducido, nunca sería seleccionada por ninguna de las estrategias anteriores. En primer lugar, dicha acción no formará parte de la política del COMDP, pues éste supone el estado conocido, con lo cual no tiene ventaja ninguna ejecutar dicha acción. Por no formar parte de la política óptima del COMDP no será nunca seleccionada por ninguno de los dos métodos anteriores. El sistema de control modo dual trata de resolver este problema considerando la entropía de la información de estado.

Cuando la incertidumbre en el sistema es muy elevada, éste puede empezar a realizar acciones arbitrarias. En ese caso, la ejecución de una acción que reduzca dicha incertidumbre podría mejorar el rendimiento a largo plazo. Este es el principio en que se basan los sistemas de control en modo dual.

Trabajan en dos modos, de ahí su nombre. El primero se activa cuando la incertidumbre está por debajo de un nivel (K) y usan cualquiera de los sistemas de decisión anteriormente descritos. El segundo se activa cuando la incertidumbre del

sistema supera un determinado umbral (**K**) pasando a ejecutar la acción que más reduce la incertidumbre.

Es necesaria establecer una forma de evaluar la incertidumbre sobre el estado del sistema y para ello se usa la entropía, que se calcula a partir de la función de distribución de probabilidad y mide la concentración o esparcimiento de la masa de probabilidad. Dada una función de distribución de probabilidad $f()$ sobre un espacio discreto X , la entropía viene definida por la siguiente ecuación:

$$H(f) = -\sum_x \log(f(x))f(x) \quad [Eq.7.17]$$

La entropía alcanza su valor extremo cuando la incertidumbre es máxima, es decir, cuando la función de distribución de probabilidad es la uniforme:

$$\forall x / x \in X, f(x) = \frac{1}{|X|}$$

Cuya entropía es:

$$H(f) = -\log\left(\frac{1}{|x|}\right)$$

Queda por determinar la forma de calcular la acción que minimice la entropía en el siguiente paso. La entropía esperada $SH(b,a)$ después de ejecutar la acción a cuando el sistema tiene una distribución de probabilidades sobre los estados b viene dada por:

$$SH(b,a) = \sum_o p(o/b,a)H(b_o^a)$$

Conocida como **entropía esperada de estado**. Donde :

$$H(b_o^a) = -\sum_s \log(b_o^a(s))b_o^a(s) \quad [Eq.7.18]$$

Por tanto, la acción a escoger para reducir la entropía será aquella que minimice $SH(b,a)$:

$$\Gamma_{DM}(b) = \arg \min_a SH(b, a) \quad [Eq.7.19]$$

Quedando, por tanto, la estrategia como sigue:

$$\Gamma_{DM-x}(b) = \begin{cases} \arg \min_a SH(b, a) & \text{Si } \bar{H}(b) > ENT_MAX \\ \Gamma_x(b) & \text{En cualquier otro caso.} \end{cases} \quad [Eq.7.20]$$

Se ha normalizado la función entropía al intervalo [0..1], es decir:

$$\bar{H}(b) = \frac{H(b)}{-\log(1/|b|)}$$

Usando $G_{AV}(b)$ como $G_X(b)$ dará lugar a la estrategia $G_{DM-AV}(b)$ y con la misma se pueden obtener $G_{DM-MLS}(b)$ y $G_{DM-Q-MDP}(b)$.

La misma razón que induce al uso de **AV** en lugar de **MLS** se puede argumentar para la utilización de una variante del control dual **DM**, denominada **ADM**, que se basa en minimizar la entropía de la función de distribución de las acciones. Es decir, en lugar de calcular la *entropía esperada de estado* calculamos la *entropía esperada de acción*:

$$AH(b, a) = \sum_o p(o/b, a) H(P_a(b_o^a)) \quad [Eq.7.21]$$

Por tanto, la acción a escoger para reducir la entropía será aquella que minimice $AH(b,a)$:

$$\Gamma_{ADM}(b) = \arg \min_a AH(b, a) \quad [Eq.7.22]$$

Quedando por tanto la estrategia como sigue:

$$\Gamma_{ADM-x}(b) = \begin{cases} \arg \min_a AH(b, a) & \text{si } \overline{H}(b) > ENT_MAX \\ \Gamma_x(b) & \text{En cualquier otro caso.} \end{cases} \quad [Eq.7.23]$$

Al igual que en el caso anterior, se pueden obtener las variantes: $G_{ADM-MLS}(b)$, $G_{ADM-AV}(b)$ y $G_{ADM-Q-MDP}(b)$

7.5.2.4 Estrategias de decisión heurísticas con predicción

La mayor parte de los algoritmos de decisión presentados anteriormente necesitan poco tiempo de procesado para calcular la acción a ejecutar comparado con el tiempo necesario para ejecutar las distintas acciones. En el caso opuesto, las políticas de decisión local requieren demasiado tiempo para obtener una solución aceptable. En vista de esto, sería interesante disponer de métodos que puedan obtener una primera solución aplicando una política global y mejorar dicha solución en función del tiempo disponible. Esto se puede ver como una extensión de los métodos anteriores a algoritmos *anytime*.

De la teoría de los POMDP se sabe que la función valor de la política óptima para el horizonte infinito (π) puede ser aproximada tanto como se desee por la de horizonte finito (π^T) considerando éste suficientemente largo [Sawaki 1978].

$$\lim_{T \rightarrow \infty} \|V(\pi^T, s) - V(\pi, s)\| = 0$$

Parece lógico esperar que, en general, cuanto mayor sea T , la política se aproximará más a la óptima. No obstante, no existen garantías de que a cada paso se mejore la política, siendo relativamente fácil encontrar ejemplos que demuestren lo contrario.

El cálculo de todas las posibles políticas para un horizonte T determinado puede verse como la construcción de un árbol, donde el factor de ramificación viene determinado por el número de combinaciones posibles de acciones y observaciones. En

cada paso, se plantea la posibilidad de ejecutar cualquier acción y, como consecuencia, se pueden observar un conjunto de observaciones. En el apéndice A se indica un ejemplo sencillo de la construcción de este árbol para 1, 2, 3 y 4 pasos.

Calcular el árbol con T suficientemente grande se presenta como una tarea difícil de abordar, a menos que se usen factores de penalización temporal muy altos (valores de ρ cercanos a 0) y sistemas con factores de ramificación pequeños.

No obstante, aquí se plantea la introducción de estos métodos de predicción combinados con los algoritmos de decisión descritos en el apartado anterior. La decisión se toma en función de la rama del árbol asociada a la acción con mayor bonificación. El árbol se construye contemplando todas las posibles combinaciones de acciones a ejecutar y observaciones obtenidas. La profundidad del árbol viene determinada por el tiempo que se disponga para calcular la decisión.

La función valor asociada a la política óptima para el POMDP de horizonte finito viene dada por la ecuación A.15 que se repite a continuación:

$$V_n(\mathbf{p}^T, b) = \max_{a \in A} \left\{ w(b, a) + \gamma \sum_{b' \in B'(b, a)} P(b, a, b') V_{n+1}(\mathbf{p}^T, b') \right\}$$

El primer término $w(b, a)$ se debe a la bonificación inmediata como consecuencia de ejecutar la acción a cuando el estado del sistema está definido por b , mientras que el segundo término representa el valor esperado en el futuro atenuado por el factor de penalización temporal. La bonificación futura es el sumatorio sobre todos los posibles estados en los que el sistema se puede encontrar en el siguiente paso de las bonificaciones esperadas en esos estados ponderadas por la probabilidad de que éstos ocurran. A su vez, se puede desarrollar este último término obteniéndose la ecuación 7.24.

$$V_n(\mathbf{p}^T, b) = \max_{a \in A} \left\{ w(b, a) + \sum_{b' \in B'(b, a)} P(b, a, b') \times \right. \\ \left. \max_{a' \in A} \left[\gamma w(b', a') + \gamma^2 \sum_{b'' \in B''(b', a')} P(b', a', b'') \times V_{n+2}(\mathbf{p}^T, b'') \right] \right\} \quad [Eq. 7.24]$$

Los términos que implican la función valor aparecen multiplicados por r^2 y, en general, si se expande n pasos, el factor será r^n . Por lo tanto, a medida que se expande esta función, los valores de bonificación restantes (futuros) tienen menos peso sobre el valor final. Esta apreciación se hace más relevante para valores de r pequeños.

Los métodos de decisión desarrollados en este apartado calculan las bonificaciones futuras usando los algoritmos de decisión analizados en el apartado anterior. La acción a ejecutar será aquella que maximice la ecuación 7.24. De esta forma, cuando se disponga de tiempo, se extenderá la ecuación en el futuro tanto como se pueda. En el último paso se aplicará alguno de los métodos de aproximación como los descritos anteriormente (AV, MLS, Q_MDP, ADM, ...). A los métodos resultantes se les denominará en lo que sigue FH, dando lugar a FH_AV, FH_MLS, FH_Q_MDP, etc.

Con este método se soluciona además el problema de la inclusión de acciones reductoras de entropía que no son tenidas en cuenta por otros métodos como puede ser MLS o AV.

Desarrollo del algoritmo

El proceso de expansión de la función valor puede considerarse como el de creación de un árbol. Cada nodo del árbol representa un estado (información de estado) diferente al que se puede llegar. El desarrollo de una capa del árbol equivale a la expansión de todos los nodos terminales (hojas). Para cada nodo se consideran las posibles acciones a ejecutar y las observaciones que se pueden producir, obteniéndose de esta forma un factor de ramificación $|A| \times |Z|$ que se corresponde con el sumatorio de la ecuación A.15 o su equivalente sobre las observaciones (ecuación A.16). El algoritmo 8.1 resume dicho proceso.

Algoritmo 8.1 Desarrollo de una nueva capa del árbol

```

Expansion(hojas_antiguas, hojas_nuevas, profundidad n)
Para cada hoja en hojas_antiguas
  Para cada acción a
    Para cada observación o posible al ejecutar esa acción.
      Crear una nueva hoja:
        -Acción paso n+1=a
        -Observación paso n+1=o
        -Información de estado (usar ecuación 7.5)
        -Prob paso n+1 (denominador ecuación 7.5)
        -Bonificación inmediata paso n+1 (usar ecuación 7.26)
        -Bonificación futura (usar ecuación 7.25)
      Añadir esa hoja a la lista de hojas_nuevas
    Fin para cada observación.
  Fin para cada acción
Fin para cada hoja
Fin expansión.

```

Una vez expandido el árbol se calcula la bonificación esperada para cada conjunto de ramas del nodo raíz (un conjunto engloba todas las ramas con la misma acción inicial). Para ello, se parte de la bonificación esperada de los nodos terminales cuyo cálculo depende del método (AV, MLS, etc) basándose en la mayor parte en la bonificación obtenida por el COMDP asociado.

Para calcular la bonificación futura de los nodos finales, primero se determina la acción a de acuerdo con el algoritmo asociado (para FH_AV será el AV, etc). Una vez obtenida la acción a para ese nodo, caracterizado por la información de estado b , se calcula la bonificación $V(b,a)$ de acuerdo con la siguiente fórmula:

$$V(b,a) = \sum_{s \in S} b(s)V'^a(s) \quad [Eq. 7.25]$$

donde $V'^a(s)$ es la función valor resultante de ejecutar a cuando el robot está en el estado s y, a partir de entonces, se aplica siempre la política óptima suponiendo el estado conocido($G_{Co}(b)$). Este valor se obtiene al principio de la solución del COMDP.

Esta bonificación de los nodos finales (hojas) se propaga hacia arriba de acuerdo con la ecuación 7.24 donde la bonificación inmediata se calcula utilizando la ecuación 7.26

$$w(b,a) = \sum_{s \in S} b(s)r(s,a) \quad [Eq. 7.26]$$

Como la lista de hojas se encuentra ordenada por acciones y observaciones, la bonificación se calcula de forma recurrente comenzando por las hojas. Este proceso se describe de forma resumida en el algoritmo 8.2

Algoritmo 8.2 Decisión de la acción a ejecutar

```

Mejor_action(hojas, n)
Iniciar tablas v_act[MAX_NIVELES] a 0 y v_max[MAX_NIVELES] a MIN_V.
A_ant = acción último paso de la primera hoja
Mientras queden hojas en la lista
    Siguiendo hoja h
    A = acción último paso de h
    Si la acción A es la misma que la A_ant
        v_act[n] = v_act[n] + prob_hoja x bonificación futura
    Sino
        Si v_act[n] > v_max[n]
            v_max[n] = v_act[n]
        Si n = 0
            Best_act = a
        Si n ≠ 0 y la acción u obs. de h y la anterior para n-1 son distintas
            Mej_act = Propaga_bonificación(hojas, h, v_act, v_max, n-1)
            v_act[n] = prob_hoja x bonificación futura
        A_ant = a
Fin mientras hojas.
Mej_act es la mejor acción
Fin mejor_action
    
```

Cada vez que se ejecuta una acción, se eliminarán aquellas ramas del árbol que no tengan dicha acción en el paso inicial.

Algoritmo 8.3 Propagación de la bonificación a los nodos superiores del árbol

```

Propaga_bonificación(hojas, h, v_act, v_max, n)
A = acción paso n de h
A_ant = acción paso n de la hoja anterior a h
Si la acción A es la misma que la A_ant
     $V\_act[n] = v\_act[n] + prob(h-1, n-1) \times v\_max[n-1]$ 
Sino
     $V\_act[n] = v\_act[n] + prob(h-1, n-1) \times v\_max[n-1]$ 
     $V\_act[n] = v\_act[n] \times r + immediate\_rew[h-1, n]$ 
    Si  $v\_act[n] > V\_max[n]$ 
         $V\_max[n] = v\_act[n]$ 
    Si  $n = 0$ 
        Best_act = a
         $v\_act[level] = 0$ 
    Si  $n \neq 0$  y la acción u obs. de h y la anterior para n-1 son distintas
        Best_act = Propaga_bonificación(hojas, h, v_act, v_max, n-1)
         $V\_max[level] = MIN\_V$ 
         $V\_act[n] = prob \text{ de hoja } \times v\_max[n-1]$ 
Fin propaga_bonificación.8

```

Algoritmo 8.4 Poda del árbol después de la ejecución de una acción

```

Poda(hojas, act, ob)
Para cada hoja de hojas
    Si la acción y observación del primer paso no coinciden con act y ob
        Eliminar hoja de hojas.
    Sino
        Para cada paso n de profundidad
            Datos hoja paso n = Datos hoja paso n+1
Fin para cada hoja
Fin poda

```

⁸ Son necesarias consideraciones especiales para la primera y última hoja.

Reducción de cálculos

Existen diversas formas de simplificar el número de cálculos a realizar dependiendo de la naturaleza del proceso y aprovechando algunas características, entre las que destacan:

- ✓ Gran parte de las observaciones no se pueden producir con algunas acciones. Este hecho reduce el factor de ramificación. En particular, para el modelo del sistema aquí utilizado pasa de $|Z|x|A|$ (64×7) a 24 (ver tabla 7.4).
- ✓ No es necesario mantener una estructura con todos los nodos del árbol sino solamente los nodos terminales en los cuales se almacena la secuencia de acciones y observaciones desde el estado actual hasta el estado de ese nodo.
- ✓ Se aprovecha que las matrices de probabilidades de observación y transición son matrices huecas. De esta forma, utilizando técnicas de tratamiento de matrices huecas, se reduce de forma considerable el número de operaciones. Esta propiedad es bastante común en este tipo de problemas.
- ✓ Algunos términos de las ecuaciones usadas en el algoritmo de expansión son almacenadas en los nodos para evitar tener que ser calculadas múltiples veces como por ejemplo el denominador de la ecuación 7.5 es necesario cada vez que se calcula la bonificación.
- ✓ Los algoritmos de expansión y decisión se ejecutan de forma entrelazada para reducir el número de cálculos.

Esta técnica puede verse como un método de predicción donde se van calculando las ramas del árbol tan profundo como se pueda y la decisión se toma en función de los nodos de la última capa generada, al mismo tiempo que se realiza una poda de todas las hojas que no procedan de la acción seleccionada.

En el siguiente capítulo se presentan algunos resultados de las prestaciones de estos métodos de decisión comparados con otros descritos anteriormente. Como las prestaciones dependen, en gran medida, de las características del problema, se utilizarán también problemas tipo obtenidos de la bibliografía técnica acerca de sistemas de decisión basados en procesos de Markov.

8 Validación del sistema de supervisión

8.1 Técnicas de validación

El dominio de los robots móviles constituye uno de los mayores retos en los temas de planificación y supervisión dentro de la Inteligencia Artificial. Su entorno no es casi nunca ni modelable ni predecible y las acciones tampoco son deterministas. Estas características dificultan también la tarea de evaluación de las prestaciones para distintas arquitecturas.

La variedad de los entornos de trabajo, junto con la relativa incertidumbre en el resultado de las acciones del robot, demandan la realización de un considerable número de pruebas bajo multitud de condiciones.

En muchas aplicaciones de Inteligencia Artificial y, más concretamente, en robótica móvil, se hace muy complejo un estudio teórico en detalle para determinar las prestaciones de un nuevo modelo, arquitectura o método, haciéndose necesaria la aplicación de otros métodos de validación [Howe 1999]. En estos casos, los tipos más comunes de evaluación se basan en comparativas con otros sistemas o realización de tests de hipótesis utilizando técnicas de simulación. A continuación se introducen estos métodos de evaluación que serán usados a lo largo de este capítulo.

Comparaciones

A falta de valoraciones absolutas (estudios teóricos), se acostumbran a usar comparaciones entre distintos modelos y métodos. Estas comparativas se pueden llevar a cabo de diversas formas como puede ser *comparación de prestaciones sobre problemas tipo (benchmarks)*, bien conocidos y difundidos en la bibliografía sobre este tema, o *competiciones de robots*, por ejemplo las distintas competiciones de la AAAI [Murphy 1997] [Arkin 1998], de fútbol en RoboCup-97 [Stone 1998] o las de planificación de AIPS [McDermott 1999].

Un problema común en estas comparaciones y, en especial, lo que se refiere a arquitecturas es que se diseñan con objetivos diferentes. Por ejemplo, un sistema que incluya aprendizaje, puede no presentar buenos resultados al principio pero éstos probablemente mejoren con el tiempo, superando a otros modelos que presenten mejores prestaciones al empezar a ejecutarse.

Las comparaciones usando como patrones problemas bien conocidos en la bibliografía no son muy usados en robótica móvil con la excepción de algunos casos de planificación. La razón puede radicar en que estos tipos de problemas han de ser fácilmente representables y perfectamente definidos, características que no se suelen encontrar en este ámbito donde la mayoría de los investigadores desarrollan sistemas de navegación y supervisión para entornos y plataformas específicas.

Tests de hipótesis

Los tests de hipótesis se basan en la realización de experimentos controlados. Se manipula el entorno y el sistema de forma sistemática para observar la influencia de estas alteraciones en las prestaciones.

En algunas ocasiones se permite que el entorno varíe libremente y se observan los efectos producidos. Este caso es el más parecido a una situación real, pero puede ocurrir que algunas situaciones no se produzcan en el tiempo que transcurre el experimento. Es necesario que el sistema permanezca funcionando durante largos períodos de tiempo para asegurar un mayor grado de robustez y fiabilidad. Por ejemplo, la fiabilidad del robot Xavier queda perfectamente caracterizada a través del gran número de horas [Simmons 1997b] que ha permanecido operativo permitiendo a usuarios de internet enviar órdenes de navegación en un entorno real como es un edificio de la universidad de Carnegie Mellon (Wean Hall).

Simulación

La mayoría de los métodos de evaluación se basan en procedimientos estadísticos para los cuales es necesaria la realización de un considerable número de pruebas bajo multitud de condiciones con el fin de obtener resultados fiables. No obstante, la realización de experimentos en robótica móvil es costosa en tiempo y recursos, además la reproducción exacta de un experimento es una tarea complicada debido principalmente a la dificultad en el establecimiento de idénticas condiciones de partida.

Para obviar esos problemas de coste y tiempo se suelen usar simuladores que permiten realizar multitud de pruebas e incluso evaluar propiedades específicas del

diseño. Con la simulación se pretende reproducir las condiciones del experimento mediante un programa. Para ello, se suele simplificar el sistema real, modelando solamente aquellos factores que puedan afectar a las prestaciones del sistema o método a evaluar. El efecto que puede tener en el sistema real esta abstracción no es tenido en cuenta al efectuar las evaluaciones utilizando este tipo de simuladores. Sin embargo, a medida que la potencia de cálculo de los ordenadores se incrementa, se van construyendo simuladores que representan de una forma más precisa tanto el comportamiento del robot como del entorno.

8.2 Descripción de los tests de evaluación

Para evaluar las prestaciones de la solución aquí presentada se podría comparar ésta con otras desarrolladas para otros robots como las descritas en el capítulo cuarto. No obstante, no es posible una comparación exacta porque se trata de planteamientos diferentes y robots con distintas características. Ninguna de ellas considera la incertidumbre de la misma forma, ni incluye un sistema de bonificación/penalización. Tampoco contemplan un sistema de monitorización similar.

Por otro lado, además de evaluar la arquitectura, sería también necesario validar el modelo del sistema que engloba tres aspectos diferentes:

1. La definición del modelo: estados, observaciones y acciones seleccionadas.
2. Los parámetros del modelo: las probabilidades de transición, sistema de bonificación/penalización, etc.
3. Por último, la solución del modelo. Es decir, la calidad de las decisiones tomadas por el supervisor.

Con respecto al primer punto, para otro robot con otras características se podría seleccionar otro conjunto de estados, observaciones y acciones. Incluso para el caso aquí descrito se pueden seguir añadiendo acciones de recuperación y observaciones a medida que se vayan desarrollando. La selección de los estados, observaciones y acciones se han realizado de forma lógica, pero ello no quiere decir que no pueda existir un modelo mejor. En cuanto a las acciones, distintos agentes pueden realizar otros tipos de acciones o también se pueden añadir más acciones al modelo desarrollado. La determinación de las probabilidades viene a definir el comportamiento dinámico del sistema. Las decisiones por las cuales se ha seleccionado este modelo han sido explicadas en capítulos anteriores. Este capítulo está dedicado fundamentalmente

a validar el sistema de decisión sobre el modelo ya definido. Es decir, el tercer apartado de los enunciados anteriormente, si bien se describirán algunos experimentos realizados con el sistema completo funcionando sobre el robot real.

La dificultad de calcular la política óptima hace muy difícil evaluar las soluciones obtenidas por las aproximaciones heurísticas. Al igual que en cualquier evaluación empírica donde no se tiene acceso a la solución óptima (en este caso sería la solución aportada por la solución exacta al POMDP), se debe poner especial cuidado a la hora de evaluar los resultados y extraer conclusiones. El objetivo de este capítulo es validar el modelo y presentar resultados comparativos entre distintos métodos de decisión aproximada para el robot móvil y sistemas similares. Es difícil extrapolar estos resultados para un caso general, pudiéndose obtener diferentes prestaciones al seleccionar otros estados y transiciones. No obstante, se pueden extraer algunas ideas generales acerca del comportamiento en los métodos de decisión aproximados aquí usados como se verá a lo largo de este capítulo. Aplicando estos métodos a problemas tipo desarrollados en la bibliografía se trata de abarcar un mayor rango de posibilidades.

Interesa también comparar los resultados con aquéllos que se obtendrían de haber supuesto un modelo determinista semejante al presentado en el capítulo sexto. Esto, sin embargo, no es sencillo debido principalmente a que ambos modelos son diferentes.

Dada la baja probabilidad de fallo del robot (la tasa de consecución con éxito de tareas se sitúa en torno al 95% [Simmons 1997b]), la obtención de muestras suficientes para poder comparar resultados entre los distintos decisores se hace demasiado costoso en tiempo, por lo cual, se ha desarrollado un simulador de eventos en su lugar, acelerando de forma considerable el proceso de comparación. El inconveniente es que, de esta forma, si bien se pueden comparar los resultados entre los distintos decisores, no es posible validar el modelo con el robot real. Con objeto de analizar la respuesta del sistema implementado en el robot, se ha creado una serie de escenarios con situaciones de error y situaciones de funcionamiento normal.

En la siguiente sección se describen los resultados obtenidos en las distintas evaluaciones comparativas. En primer lugar, se comparan los resultados obtenidos mediante simulación para las distintas variaciones del sistema de decisión descritas anteriormente sobre el modelo de supervisión cuyos parámetros se definen en el apéndice C. Como para dicho modelo no es posible obtener la solución exacta al POMDP, también se presentan los resultados obtenidos para un ejemplo mucho más sencillo (apéndice B) con el fin de contrastar las prestaciones con el modelo real. Por último, respecto a comparaciones, se muestran los resultados obtenidos para distintos

problemas que vienen siendo utilizados como referencia en la bibliografía sobre procesos de Markov para ver en qué ocasiones es conveniente usar los nuevos algoritmos *anytime* propuestos en la tesis. En la sección 9.4 se presentan algunas de las respuestas del robot ante distintos escenarios donde existen situaciones de excepción, esto es, los tests de hipótesis mencionados en el apartado anterior.

8.3 Evaluación comparativa

Como se ha mencionado en la introducción de este capítulo, a falta de valoraciones absolutas, se acostumbra usar comparaciones entre distintos modelos y métodos. Estas comparativas se pueden llevar a cabo de diversas formas, como puede ser la comparación de prestaciones sobre problemas bien conocidos y difundidos en la bibliografía sobre este tema (*benchmarks*) o competiciones de robots.

Aunque el robot Xavier sí ha participado con éxito en competiciones (AAAI-95), usando la misma arquitectura de navegación, no había sido incluido aún el sistema de supervisión, detección y recuperación de errores desarrollado en esta tesis.

En el siguiente apartado se describen los resultados obtenidos en los tests sobre el modelo descrito en el apéndice C (modelo real) por los métodos de decisión descritos en el capítulo anterior. En el apartado 8.3.2 se analizan los resultados de los experimentos sobre un modelo sencillo (descrito en el apéndice B) y en el apartado 8.3.3 los resultados sobre problemas tipo de diversa índole.

8.3.1 Comparativa sobre el modelo

En la mayoría de los métodos de evaluación surge la necesidad de realizar un gran número de experimentos que sean fácilmente reproducibles. Como ya se ha mencionado, la realización de experimentos en robótica móvil es costosa en tiempo y recursos, mientras que la reproducción exacta de un experimento es una tarea compleja debido principalmente a la dificultad en el establecimiento de idénticas condiciones de partida. Una pequeña variación en la posición inicial del robot, calibración de los sensores o ligeras modificaciones del entorno puede llevar a una situación bastante diferente del experimento inicialmente previsto.

Casi todos los sistemas de navegación y supervisión son verificados con simuladores con objeto de reducir tiempo y trabajo. Esto permite realizar multitud de pruebas e, incluso, evaluar propiedades específicas del diseño.

Un análisis teórico general de las distintas soluciones heurísticas no tiene mucho interés, dado que su comportamiento depende en gran medida de las características del problema [Hauskrecht 1996]. Sin garantías teóricas acerca de los resultados que presentan los distintos decisores, se ha usado un simulador de eventos para analizar las prestaciones en el modelo aquí descrito. En particular, interesa saber qué decisor obtiene mejores bonificaciones con un tiempo de respuesta aceptable.

Incluso la evaluación de los resultados simulados no es tarea fácil dado que no es posible disponer de la solución óptima (solución exacta del POMDP) pues ésta es inviable para sistemas relativamente grandes. En ausencia de dicha solución que sirva de referencia, se han incluido los resultados obtenidos por un decisor omnisciente (OMNI) para establecer una medida de comparación. Se trata básicamente de un supervisor que conoce el estado real del sistema reduciendo así el planteamiento de un modelo POMDP a un COMDP. Dicho decisor (supervisor) ha sido ya utilizado en la literatura con fines similares por algunos investigadores en el campo de los sistemas de metacontrol y decisión [Cassandra 1998]. Las soluciones encontradas por este controlador omnisciente constituyen, pues, una cota superior en cuanto a la calidad de la solución que muchas veces el decisor óptimo no podrá alcanzar dado que éste no dispone exactamente del estado del sistema, sino sólo de observaciones. En principio, cuanta más incertidumbre haya en las observaciones, mayor será la diferencia entre los resultados obtenidos por ambos decisores.

Es, por tanto, posible afirmar que, si los resultados del controlador heurístico son similares a los del decisor omnisciente, dichos resultados son buenos, pero si se obtienen resultados bastante peores que los del controlador omnisciente, no hay forma de saber cuan lejos están de los que pueda obtener el controlador óptimo.

8.3.1.1 Realización de los experimentos

Se ha desarrollado un simulador de eventos basado en los parámetros del modelo. Se considera que el robot tiene siempre tareas que hacer y se ha usado un factor de descuento temporal de 0,8.

Siguiendo una secuencia de test similares a la utilizada por algunos autores [Cassandra 1998] para evaluar otros algoritmos de decisión, se han realizado distintas ejecuciones (10.000 para todo algoritmo). Cada ejecución consiste en una serie de pasos (100) partiendo del mismo estado inicial. De esta forma, se obtiene la bonificación como la media de las obtenidas en las distintas simulaciones. Dado que no existe una gran dispersión de los datos, nos limitaremos a analizar los valores de la media resaltando otras propiedades estadísticas en aquellos casos que sea necesario.

De la simulación se obtiene un vector de estados y acciones que se han producido a lo largo del tiempo:

$$\bar{S} = (s^1, s^2, s^3, \dots, s^N)$$

$$\bar{A} = (a^1, a^2, a^3, \dots, a^N)$$

La evaluación de la bonificación obtenida se puede calcular de acuerdo con la siguiente ecuación:

$$V(\bar{A}, \bar{S}) = \sum_{i=1}^{N-1} \gamma^{i-1} r(s^i, a^i, s^{i+1}) \quad [Eq. 8.1]$$

donde $r(s^i, a^i, s^{i+1})$ es la bonificación obtenida al ejecutar la acción a^i y pasar del estado s^i al s^{i+1} . Dicha bonificación viene determinada por el modelo de Markov (apéndice A).

Para ese factor de penalización γ (0,8) se puede despreciar la influencia de la bonificación más allá de los 100 primeros pasos ($0,8^{100} = 2 \cdot 10^{-10}$). Teniendo en cuenta esto, se realizan simulaciones de cien pasos consecutivos y se calcula la bonificación para cada uno de ellos. La bonificación para un determinado algoritmo es la media de todas las ejecuciones.

También hay que recordar que para los métodos FH se han usado técnicas destinadas a reducir el número de cálculos al ser las matrices de transición y observaciones matrices huecas. Se han ahorrado un 32% de operaciones en los cálculos de la actualización de la información de estado y un 41% en las operaciones de decisión basadas en predicciones futuras. El factor de ramificación es de 24.

Las distintas pruebas se han llevado a cabo para dos situaciones iniciales:

- ✓ Situación 1: se parte de la situación donde existe un problema que será con igual probabilidad cualquiera de los componentes contemplados en el modelo.
- ✓ Situación 2: se considera que inicialmente todo funciona bien.

A continuación se comentan los resultados obtenidos para las dos situaciones por los distintos métodos de decisión.

8.3.1.2 Resultados

En la tabla 8.1 se resumen las bonificaciones para los distintos métodos. Con respecto a los algoritmos parametrizados (ADM, DM y FH), en esta tabla se presentan los mejores resultados sobre los distintos valores de los parámetros (K y número de pasos n) y más adelante se verá la influencia de estos parámetros. Los valores que aparecen en negrita son las bonificaciones más altas para cada situación.

De la tabla de resultados se puede observar que los algoritmos aquí propuestos presenta bonificaciones más altas que cualquier otra estrategia. Las estrategias de reducción de entropía no son una buena elección, no sólo por los malos resultados de bonificación, sino también por el tiempo empleado en tomar las decisiones para reducir la entropía.

Tabla 8.1. Resultados simulación para el modelo de supervisión del robot

Método de decisión	Bonificación situación 1	Bonificación situación 2
OMNI	107,987318	137,709945
MLS	66,798200	126,426957
AV	50,885809	126,317324
Q_MDP	65,564121	126,529791
F_AV (1 paso)	67,520942	126,616205
F_MLS (1 paso)	67,727635	126,673176
F_Q_MDP(1 paso)	67,712675	126,686770
ADM_AV	63,793469	126,346198
ADM_MLS	66,808004	126,428610
ADM_Q_MDP	65,59772	126,541399
DM_AV	59,436386	126,176805
DM_MLS	66,79820	126,427310
DM_Q_MDP	65,56412	126,529791

Si bien AV presentaba buenos resultados para el navegador [Simmons 1995], en nuestro caso MLS presenta mejores prestaciones porque el escoger una acción cuando en realidad se encuentra en otro estado puede acarrear grandes costes. Por ejemplo, suponiendo la siguiente situación:

Estado	Probabilidad	Acción
E1	0,4	A1
E2	0,2	A2
E3	0,2	A2
E4	0,2	A2

La acción a elegir por AV será A2 aun cuando el sistema esté probablemente en E1. La razón por la cual el navegador presenta mejores prestaciones en esta situación es que estados contiguos acostumbran a tener acciones similares pero no opuestas y la probabilidad acostumbra a concentrarse en estados contiguos.

Otro punto a tener en cuenta es que, en general, los controladores duales no son capaces de mejorar de forma considerable los resultados. Esto se puede achacar principalmente a que no existen acciones exclusivamente informativas y a que, cuando se toma una decisión para reducir la entropía, no se tienen en cuenta otras consideraciones que pueden ocasionar grandes costes en la solución final. El caso extremo ocurre con factores límite pequeños para los cuales el sistema se limita todo el tiempo a ejecutar acciones para reducir la entropía.

La variación de los resultados con diferentes valores de la entropía umbral K para los controladores duales puede verse en la tabla 8.2. Se ha normalizado la entropía para que su valor este en el intervalo $[0...1]$ por lo tanto K estará también entre dichos valores. Sólo se han presentado los valores para la situación 1 debido a que los resultados obtenidos para la situación 2 son muy similares.

Los algoritmos DM y ADM con valores de K igual o superior a 0,8 presentan, en la mayor parte de los casos, los mismos resultados que aquellos de los que derivan (DM_AV de AV por ejemplo). La razón radica en que el decisor, en estas situaciones, nunca opta por ejecutar una acción para reducir la entropía acerca de la información de estado. En general, los algoritmos ADM presentan un máximo en torno a $K=0,5$, mientras que los algoritmos DM no presentan grandes mejoras con respecto a los originales y sus máximos están localizados para valores de K próximos al máximo.

La solución al modelo COMDP asociado, necesaria para la aplicación de alguno de los métodos de la tabla 8.1, se muestra en el apéndice D. Para cada estado, se presenta la acción a ejecutar de acuerdo con dicha solución y la bonificación media

esperada para cada acción. Esta bonificación es la obtenida de ejecutar dicha acción en el siguiente paso y, a partir de ahí, se supone que se va a aplicar la política óptima y que no habrá incertidumbre.

Tabla 8.2. Bonificación para los métodos ADM y DM con distinto K (situación1)

Método	K (Entropía umbral)						
	0,3	0,4	0,5	0,6	0,7	0,8	0,9
ADM AV	59,03239	63,79346	50,86060	50,89854	50,88735	50,88612	50,88580
ADM MLS	59,04191	63,82088	66,77586	66,80800	66,79875	66,79820	66,79820
ADM Q_MDP	61,68116	64,18993	65,59772	65,57866	65,56361	65,56412	65,56412
DM AV	39,81847	59,43638	50,00617	50,86091	50,88725	50,88612	50,88580
DM MLS	39,32500	59,27425	66,48268	66,78753	66,79526	66,79820	66,79820
DM Q_MDP	40,67081	60,18557	65,36158	65,56254	65,56382	65,56412	65,56412

Probar que un decisor funciona mejor que otro o calificar cuan bueno es un decisor no es tarea trivial y depende mucho del proceso sobre el cual se toman las decisiones. En este caso AV o MLS proveen en algunos casos tan buenos o mejores resultados que los que tienen en cuenta la entropía y ello se debe en parte a que las acciones son mixtas. Es decir, al mismo tiempo que suponen una evolución hacia el objetivo, también proporcionan información del estado del sistema. No obstante, si se añade la acción CS (check status), dicha acción jamás será seleccionada por ninguno de los dos métodos (AV ni MLS) dado que tampoco formará parte de la política resuelta por el COMDP. La razón estriba en que el COMDP supone que el estado del sistema es conocido, con lo cual la acción CS no sería seleccionada. Sin embargo, en otros decisores como DM sí es seleccionada cuando existe mucha incertidumbre en el estado del sistema. El mismo razonamiento permite explicar que la acción GO (Go through opening) sea más veces seleccionada por los algoritmos DM y ADM. Estos efectos se verán mejor con el ejemplo del siguiente apartado.

La bonificación del controlador omnisciente (OMNI) es superior a cualquiera de los otros casos. Resulta muy difícil saber cuan lejos está el decisor óptimo del omnisciente pero parece lógico pensar que esta distancia sea proporcional a la

incertidumbre del sistema y, dado que ésta es relativamente alta en nuestro sistema, nos lleva a intuir que los resultados serán también diferentes.

En la tabla 8.3 se presentan los resultados de los algoritmos FH para distintos pasos en el futuro en la situación 1. Se puede observar que en todos los casos se mejoran los resultados a medida que se incrementa el número de pasos. No obstante, para problemas con un número de estados considerable como éste, predicciones con más de 3 pasos suele ser inviable por el tiempo necesario para obtener la solución, como queda patente en la tabla de tiempos (tabla 8.4).

Tabla 8.3. Bonificación para los métodos FH con distinto horizonte (situación 1)

Método	N (Pasos en el futuro)		
	0	1	2
F_AV	50,885809	67,520942	68,094312
F_MLS	66,798200	67,727635	68,070499
F_Q_MDP	65,564121	67,712675	66,632149

Tabla 8.4. Tiempo medio en segundos para el cálculo de una decisión

Algoritmo	Valores del parámetro K						
	0,3	0,4	0,5	0,6	0,7	0,8	0,9
ADM_AV	0,279391	0,062854	0,042346	0,002817	0,000130	0,000116	0,000076
ADM_MLS	0,281956	0,063149	0,039856	0,003115	0,000153	0,000063	0,000062
ADM_Q_MDP	0,204921	0,057954	0,043159	0,004032	0,000373	0,000241	0,000224
DM_AV	0,067834	0,024643	0,023237	0,002047	0,000116	0,000079	0,000077
DM_MLS	0,069069	0,024080	0,021421	0,002225	0,000132	0,000063	0,000064
DM_Q_MDP	0,061881	0,021699	0,023120	0,002896	0,000320	0,000240	0,000226
Algoritmo	Horizonte						
	0 pasos		1 paso		2 pasos		
F_AV	0,000036		0,083561		2,254796		
F_MLS	0,000024		0,084376		2,278207		
F_Q_MDP	0,000201		0,087601		2,477204		

En la tabla de tiempos (tabla 8.4) puede verse como el tiempo de decisión a cada paso se va multiplicando aproximadamente por el factor de ramificación (24). Por esta razón se ha reducido el número de pruebas para los métodos con dos pasos vista a 100000 decisiones, es decir, 1000 pruebas de 100 pasos cada una. Las ejecuciones se han llevado a cabo en un ordenador Pentium MMX a 200Mhz con 64 Mbytes de memoria RAM utilizando el sistema operativo LINUX

8.3.2 Análisis sobre un modelo más sencillo

Debido a la complejidad del modelo anterior, no es fácil obtener una idea clara de la influencia de los distintos parámetros en las prestaciones del sistema, así como la forma en que se producen dichas relaciones. En esta sección se realizan una serie de ensayos similares a los del apartado anterior pero sobre un modelo más simple descrito en el apéndice B.

Cada uno de los valores de bonificación que a continuación se presentan se han obtenido como la media de 10.000 ensayos diferentes, donde cada uno consiste en la ejecución de 100 pasos (100 decisiones) comenzando siempre con la misma información de estado. El estado inicial se genera de forma aleatoria de acuerdo con las probabilidades de información de estado. En concreto, se han considerado tres condiciones iniciales diferentes que se corresponden con tres situaciones:

- ✓ Bajo grado de incertidumbre:
 - Alta probabilidad (0,9) que el sistema se encuentre en una situación de trayectoria no navegable (NN).
 - Baja probabilidad (0,1) que el sistema se encuentre en una situación de trayectoria no navegable (NN).
- ✓ Alto grado de incertidumbre. Es igualmente probable que el sistema se encuentre en cualquiera de los estados posibles.

A continuación se describen los resultados para cada una de las distintas condiciones iniciales.

Bajo grado de incertidumbre

En la tabla 8.5 se muestran los resultados de la bonificación para las distintas simulaciones en los dos situaciones de partida con poca incertidumbre. En negrita se resaltan los mejores valores obtenidos que, debido a la sencillez del problema, coinciden con los óptimos (POMDP). El número de ejecuciones de la acción CC, la única cuyo objetivo es la reducción de la entropía, es cero para los métodos AV, MLS y OMNI.

Tabla 8.5. Resultados simulación para el ejemplo simplificado de supervisión del robot

Método de decisión	Bonificación $P(NN) = 0,1$	Bonificación $P(NN) = 0,9$
OMNI	99,986179	99,997070
POMDP	-26,263083	-25,781372
MLS	-223,101293	-203,389914
AV	-223,101293	-203,389914
Q_MDP	-31,985575	-29,788089
F_AV (1 paso)	-26,263083	-25,903365
F_MLS (1 paso)	-26,263083	-25,903365
F_Q_MDP(1 paso)	-27,637682	-25,903365
ADM_AV	-26,263083	-25,781372
ADM_MLS	-26,263083	-25,781372
ADM_Q_MDP	-26,263083	-25,781372
DM_AV	-26,263083	-25,781372
DM_MLS	-26,263083	-25,781372
DM_Q_MDP	-26,263083	-25,781372

La solución al COMDP asociado, necesaria para la aplicación de alguno de los métodos de la tabla 8.5, se muestra en la tabla 8.6. Por cada estado se presenta la acción a ejecutar, de acuerdo con dicha solución y la bonificación media esperada para cada acción. Esta bonificación es la obtenida de ejecutar dicha acción en el siguiente paso, suponiendo que a partir de ahí se va a aplicar la política óptima y desaparecerá la incertidumbre acerca del estado tal y como establecen los procesos COMDP.

Tabla 8.6. Solución COMDP para el ejemplo simplificado de supervisión del robot

Estado inicial COMDP	Mejor acción COMDP	Bonificación SC	Bonificación CC	Bonificación CA
NN	CA	-10	80	100
NV	SC	100	80	-10

Algunos resultados de la tabla 8.5, debido a la sencillez del modelo, se pueden formular matemáticamente. Así, para el caso del decisor omnisciente, dado que sabe en todo momento el estado del sistema, aplicará la política dada por la tabla 8.6 obteniendo a cada paso la misma bonificación (100). La bonificación total será pues, de acuerdo con la ecuación 8.1:

$$V = \sum_{i=1}^{N-1} \gamma^{i-1} r(s^i, a^i, s^{i+1}) = \sum_{i=1}^{N-1} (0.9)^{i-1} 100 \approx 100 \frac{1}{(1-0.9)} = 1000$$

En la simulación no se ha alcanzado este valor dado que el sumatorio se trunca en los cien primeros números de la serie, pero si se han obtenido valores muy aproximados, como se muestra en la tabla 8.5.

Otro detalle a observar es que la bonificación, así como el número de veces que se ha ejecutado la acción CC, es el mismo para AV y MLS. Este hecho se debe a dos motivos. En primer lugar, las decisiones de ambos métodos son las mismas. En efecto, la decisión de MLS consiste en seleccionar la acción asociada al estado más probable de acuerdo con la tabla 8.6. A su vez, la acción a ejecutar por AV es aquella tal que la suma de las probabilidades de los estados que la tengan asignada de acuerdo con la tabla 8.6 sea mayor. Como existen dos estados y cada uno tiene asignada una acción diferente, ambos métodos producen los mismos resultados. En segundo lugar, los resultados para ambos métodos son idénticos porque se utilizan las mismas secuencias de números aleatorios en todas las simulaciones.

También cabe destacar las malas prestaciones de estos métodos debidas a la suposición del COMDP de conocimiento del estado que lleva a la ausencia de ejecución de la acción CC.

El planificador omnisciente presenta resultados mucho mejores que el óptimo (POMDP) para ambas situaciones, lo cual resulta razonable debido a la alta penalización de una decisión errónea y al coste de la acción CC que el controlador omnisciente nunca tiene que ejecutar.

Los resultados de los decisores de modo dual (DM y ADM) son función de la entropía umbral K . En la tabla 8.5 se muestran los resultados obtenidos para el mejor de los casos, mientras que en la tabla 8.7 se muestran resultados para diversos valores de K con condiciones iniciales $P(NN)=0,9$. En este caso se obtienen las mejores prestaciones para K cercanos a 0,5.

Tabla 8.7. Bonificación para los métodos ADM y DM con distinto K

Método	K (Entropía umbral)							
	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
ADM AV	-28,4092	-28,4092	-28,4092	-25,7813	-29,7880	-29,7880	-60,4696	-86,9406
ADM MLS	-28,4092	-28,4092	-28,4092	-25,7813	-29,7880	-29,7880	-60,4696	-86,9406
ADM Q_MDP	-28,4092	-28,4092	-28,4092	-25,7813	-29,7880	-29,7880	-29,7880	-29,7880
DM AV	-28,4092	-28,4092	-28,4092	-25,7813	-29,7880	-29,7880	-60,4696	-86,9406
DM MLS	-28,4092	-28,4092	-28,4092	-25,7813	-29,7880	-29,7880	-60,4696	-86,9406
DM Q_MDP	-28,4092	-28,4092	-28,4092	-25,7813	-29,7880	-29,7880	-29,7880	-29,7880

Cabe destacar, en primer lugar, que los resultados para DM y ADM son idénticos. De nuevo hay que tener en cuenta que la secuencia de números aleatorios usada por el simulador es la misma para ambas simulaciones, con lo cual, si las decisiones coinciden, se producirán idénticos eventos y se ejecutarán las mismas acciones. En efecto, las decisiones son iguales porque la entropía esperada para las acciones tiene valores equivalentes a la de los estados (similar razonamiento usado en la comparación AV y MLS).

También se puede observar que el comportamiento de DM y ADM para Q_MDP coincide con el de DM y ADM para AV y MLS cuando los valores de K son relativamente pequeños. Veamos más en detalle el funcionamiento de estos decisores. En primer lugar, cuando la entropía se encuentra por encima del umbral K , la decisión es independiente de si se trata de MLS, AV o Q_MDP. Para estudiar el comportamiento de Q_MDP, supóngase p la probabilidad de que el sistema se encuentre en estado NN. Aplicando la ecuación 7.15 se obtiene:

$$P_{FP}(b) = \sum_s b(s) V^{FP}(s) = p(V^{FP}(NN)) + (1-p)(V^{FP}(NV))$$

$$P_{NP}(b) = \sum_s b(s) V^{NP}(s) = p(V^{NP}(NN)) + (1-p)(V^{NP}(NV))$$

$$P_{CP}(b) = \sum_s b(s) V^{CP}(s) = p(V^{CP}(NN)) + (1-p)(V^{CP}(NV))$$

Identificando los valores de la función V con los obtenidos en la solución del COMDP (tabla 8.6) se tiene:

$$P_{FP}(b) = p(-10) + (1-p)100 = -110p + 100$$

$$P_{NP}(b) = p(100) + (1-p)(-10) = 110p + 10p$$

$$P_{CP}(b) = p(80) + (1-p)80 = 80$$

De acuerdo con la ecuación 7.16, la acción FP será elegida en detrimento de NP cuando se verifique:

$$-110p + 100 > 110p - 10 \Leftrightarrow p > \frac{1}{2} \quad [Eq. 8.2]$$

Mientras tanto, la acción CP se ejecutará cuando se cumplan las siguientes condiciones:

$$\left. \begin{array}{l} 80 > -110p + 100 \\ 80 > 110p - 10 \end{array} \right\} \Leftrightarrow \frac{2}{11} < p < \frac{9}{11} \quad [Eq. 8.3]$$

Se concluye, pues, que cuando la incertidumbre es relativamente alta se verifica la ecuación 8.3 (valores de p cercanos a $\frac{1}{2}$) y se ejecuta la acción CP mientras que, al disminuir la entropía (valores de p cercanos a 1 ó 0), se aplica la acción NP o FP, según se verifique o no la ecuación 8.2. En este último caso, la política a seguir coincide con los métodos MLS y AV, es decir, se ejecuta la acción asociada al estado con mayor probabilidad. De esta forma se explica que para valores bajos de la entropía

umbral, los métodos DM y ADM correspondientes a las tres políticas produzcan los mismos resultados.

Por último, se incluyen en la tabla 8.8 los resultados de las simulaciones para los algoritmos FH partiendo de un estado inicial con $P(NN)=0,1$. En general, los resultados mejoran al aplicar dichas modificaciones, si bien no está garantizado el incremento en cada paso como se puede observar en esta tabla.

Tabla 8.8. Bonificación para los métodos FH con distinto horizonte

Método	N (Pasos en el futuro)				
	0	1	2	3	4
F_AV	-203,38991	-25,903365	-28,409239	-29,788089	-25,781372
F_MLS	-203,38991	-25,903365	-28,409239	-29,788089	-25,781372
F_Q_MDP	-29,788089	-25,903365	-25,903365	-25,903365	-25,781372
F_POMDP	-25,781372	-25,781372	-25,781372	-25,781372	-25,781372

El método F_POMDP se incluye para comprobar que, si el método aplicado en las hojas del árbol es el óptimo, los resultados de aplicar FH también serán los óptimos. Esta conclusión es conocida de antemano a partir de la definición de política óptima y la función V (apéndice A).

Alto grado de incertidumbre

En la tabla 8.9 se muestran los resultados obtenidos cuando el estado inicial es totalmente desconocido $\{P(NN)=P(NV)=0,5\}$. En la primera columna se representan los valores de bonificación medios y, en la segunda, el número de ejecuciones de la acción CC durante la simulación. Los resultados son muy parecidos al caso anterior pero las bonificaciones son un poco menores debido a la incertidumbre inicial.

Tabla 8.9. Resultados simulación para el ejemplo simplificado de supervisión del robot

Método de decisión	Bonificación media	Ejecuciones CC
OMNI	99,986179	0
POMDP	-32,723167	539928
MLS	-262,660751	0
AV	-262,660751	0
Q_MDP	-36,632362	413234
F_AV (1 paso)	-32,723167	539928
F_MLS (1 paso)	-32,723167	539928
F_Q_MDP(1 paso)	-34,457006	570713
F_POMDP(1 paso)	-32,723167	570713
ADM_AV	-32,723167	539928
ADM_MLS	-32,723167	539928
ADM_Q_MDP	-32,723167	539928
DM_AV	-32,723167	539928
DM_MLS	-32,723167	539928
DM_Q_MDP	-32,723167	539928

En la tabla 8.11 se presentan los resultados de los algoritmos FH con distintos horizontes. Al igual que en el caso anterior, la inclusión de un solo paso aumenta las prestaciones, pero el incremento de un paso más no es siempre garantía de mejora con respecto al anterior.

8.3.3 Comparaciones sobre problemas de referencia

Aunque las comparaciones usando como patrones problemas ampliamente estudiados en la bibliografía no son muy usados en robótica móvil para evaluar las prestaciones de arquitecturas, sí se pueden aplicar al análisis parcial de algunas de las tareas implicadas en el proceso de navegación. En nuestro caso, se pretenden evaluar el comportamiento de las variaciones de los algoritmos heurísticos de solución aproximada del POMDP utilizadas en la realización del decisor. Los problemas que se tomarán como referencia son problemas bien conocidos en la bibliografía referente a procesos de Markov que se irán describiendo a lo largo de este apartado.

Tabla 8.10. Bonificación para los métodos ADM y DM con distinto K

Método	K (Entropía umbral)							
	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
ADM AV	-35,3120	-35,3120	-35,3120	-32,7231	-36,6323	-36,6323	-66,8952	-90,3637
ADM MLS	-35,3120	-35,3120	-35,3120	-32,7231	-36,6323	-36,6323	-66,8952	-90,3637
ADM Q_MDP	-35,3120	-35,3120	-35,3120	-32,7231	-36,6323	-36,6323	-36,6323	-36,6323
DM AV	-35,3120	-35,3120	-35,3120	-32,7231	-36,6323	-36,6323	-66,8952	-90,3637
DM MLS	-35,3120	-35,3120	-35,3120	-32,7231	-36,6323	-36,6323	-66,8952	-90,3637
DM Q_MDP	-35,3120	-35,3120	-35,3120	-32,7231	-36,6323	-36,6323	-36,6323	-36,6323

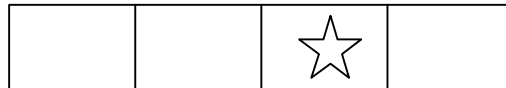
Tabla 8.11. Bonificación para los métodos FH con distinto horizonte

Método de decisión	N (Pasos en el futuro)				
	0	1	2	3	4
F_AV	-262,66075	-32,723167	-35,312086	-36,632362	-34,457006
F_MLS	-262,66075	-32,723167	-35,312086	-36,632362	-34,457006
F_Q_MDP	-36,632362	-34,457006	-34,457006	-34,457006	-32,723167

8.3.3.1 Problemas simples con solución óptima disponible

Se han utilizado cuatro problemas diferentes tomados como referencia por diversos autores. Estos son conocidos como:

- ✓ TIGER. Un agente situado frente a dos puertas cerradas tiene que decidir si abrir una de ellas o escuchar los sonidos provenientes del otro lado. Detrás de una hay algo muy deseado por el agente (bonificación muy alta), mientras que detrás de la otra se encuentra un peligroso tigre (penalización alta). Las acciones del agente pueden ser abrir una de las puertas o escuchar. Esta escucha tiene asignado un coste y además no es una escucha completamente fiable. La descripción detallada de este problema aparece por primera vez en [Cassandra 1994].
- ✓ 1DMAZE. Este ejemplo ilustrativo se basa en un sistema compuesto de cuatro estados que se corresponden con cuatro posiciones, tal como se muestra en la figura 8.1, donde la posición marcada con la estrella es el objetivo. Se dispone de dos observaciones que indican si el agente se encuentra en el objetivo y dos posibles acciones de movimiento a la derecha e izquierda. La formulación detallada de este ejercicio puede verse, por ejemplo, en [Kaelbling 1998].



*Figura 8.1 Estados para el problema 1DMAZE [Kaelbling 1998].
Cada celda es un estado y la celda que contiene la estrella es el estado objetivo.*

- ✓ 4x4. Es un ejemplo similar al anterior pero un poco más complejo. El entorno, en este caso es un área de 16 celdas (4x4) y las acciones posibles son cuatro que se corresponden con movimientos en cuatro direcciones ortogonales [Kaelbling 1998].
- ✓ NONLIN. Descrito en [Parr 1995], se trata de un dominio simple cuyo diagrama de estado se muestra en la figura 8.2. Entre los siete estados, existen dos (etiquetados como A) que son indistinguibles. Las acciones posibles son a, b y c.

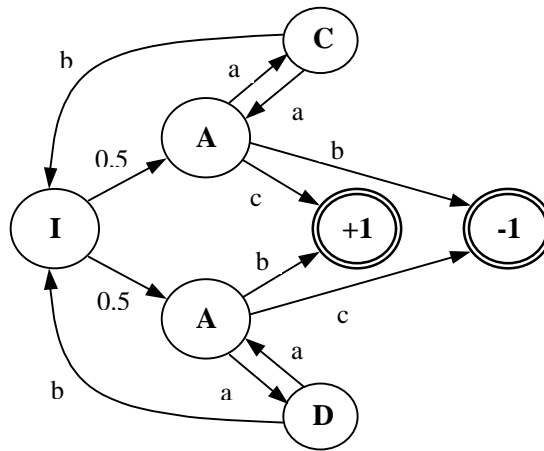


Figura 8.2 Diagramas de transiciones entre estados para el problema NONLIN [Parr 1995]

En los ensayos realizados en la tesis, se han mantenido los mismos estados iniciales que aparecen en las respectivas referencias así como el resto de los parámetros. El número de estados, acciones y observaciones para cada problema se presenta en la tabla 8.12.

Tabla 8.12. Parámetros de los problemas sencillos tipo

Problema	Parámetros					
	Nº de estados	Nº de acciones	Nº de observaciones	ρ	Factor de ramificación	Iteraciones por ensayo
TIGER	2	3	2	0,95	6	200
1DMAZE	4	2	2	0,75	4	100
4X4	16	4	2	0,95	8	200
NONLIN	7	3	6	0,95	18	200

Cada uno de los valores de bonificación que a continuación se presentan se han obtenido como la media de 10.000 ensayos diferentes con las mismas condiciones

iniciales. Por razones expuestas anteriormente, el número de pasos en cada ensayo se ha establecido en función del factor de atenuación temporal del problema: 100 para aquellos problemas con r pequeño (1DMAZE) y 200 para aquellos con r mayor (TIGER, 4X4 y NONLIN). Ambos valores se presentan en la tabla 8.12 junto con el factor de ramificación para los algoritmos FH.

Los resultados obtenidos en simulación aplicando los distintos algoritmos se muestran en la tabla 8.13. Conviene señalar que, para los métodos parametrizados (DM, ADM y FH) sólo se presentan los mejores resultados en el rango de valores de los parámetros (entropía umbral y número de pasos) sobre los que se hicieron los tests.

Tabla 8.13. Bonificación para problemas tipo simples.

Método de decisión	Problema			
	1DMAZE	4x4	TIGER	NONLIN
OMNI	1,898397	4,679971	199,981593	12,655069
POMDP	1,358784	3,736654	19,641558	7,200711
MLS	1,350784	3,736654	-901,2841	6,328266
AV	1,228585	3,677407	-898,6487	6,326803
Q_MDP	1,350784	3,734477	19,641558	6,326803
F_AV (1 paso)	1,35878	3,677407	-19,999262	6,326803
F_MLS (1 paso)	1,35878	3,736654	-19,999262	6,326803
F_Q_MDP(1 paso)	1,35878	3,734477	16,414812	6,326803
ADM_AV	1,228585	3,740596	-21,086787	7,200711
ADM_MLS	1,358784	3,740596	-21,086787	7,200711
ADM_Q_MDP	1,358784	3,733809	19,641558	7,200711
DM_AV	1,228585	3,689611	19,641558	7,200711
DM_MLS	1,358784	3,738977	19,641558	7,200711
DM_Q_MDP	1,358784	3,734477	19,641558	7,200711

Cabe recordar que se ha usado idéntica secuencia de números aleatorios para las distintas simulaciones, de ahí que se hayan obtenido resultados similares en

algoritmos ligeramente diferentes. Esto quiere decir que las decisiones tomadas por los algoritmos en las situaciones presentadas durante la simulación han sido las mismas.

En el problema TIGER los resultados de Q_MDP son mucho mejores que los de AV y MLS, al igual que ocurría en el caso sencillo de supervisión del apartado anterior. Ni MLS ni AV seleccionan nunca ninguna acción cuyo único objetivo sea la reducción de la entropía del sistema. La acción de escuchar en este caso representa a ese tipo de acciones y tiene un coste pequeño. Por lo tanto, su aplicación en situaciones donde la incertidumbre acerca del estado del sistema es elevado puede ser muy rentable, dado que la ejecución de cualquiera de las otras acciones en un estado equivocado conlleva un elevado coste. De nuevo, la aplicación de los algoritmos FH mejoran las prestaciones tanto de AV como MLS al tener en consideración dichas acciones.

Se puede observar que MLS y AV rara vez proporcionan los mejores resultados mientras que la adición de predicción a estos métodos supone, por lo general, una mejora considerable. Los algoritmos de control dual producen buenos resultados en todos los casos, seleccionando los umbrales de entropía apropiados, pero no hay una forma de determinar éstos para un caso general. En la tabla 8.14 se presenta, a modo de ejemplo, la bonificación media de los métodos de control dual, los resultados de aplicar DM_MLS para distintos valores de K .

Tabla 814. Bonificación DM-MLS con distinto K

Problema	K (Entropía umbral)							
	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
1DMAZE	1,18473	1,18473	1,18473	1,18473	1,18473	1,18473	1,35878	1,35878
4X4	3,51538	3,51538	3,51538	3,51538	3,51538	3,67820	3,67820	3,73897
TIGER	19,6415	19,6415	19,6415	19,6415	19,6415	-72,5791	-72,5791	-72,5791
NONLIN	7,20071	7,20071	6,32826	6,32826	6,32826	6,32826	6,32826	6,32826

La evolución de los algoritmos FH con el número de pasos se muestra en la tabla 8.15. Como se puede observar, se produce una respuesta igual o mejor en todos los casos al añadir la predicción de un solo paso, con excepción del Q_MDP para el problema del TIGER. El caso más notable es el de F_AV y F_MLS del mismo problema produciéndose una considerable mejora. Como en cualquiera de los métodos heurísticos, no se pueden garantizar sus resultados para todos los casos, se hace necesario un estudio detallado para cada caso en concreto. No obstante, los resultados

aquí obtenidos demuestran que constituyen una considerable mejora en la mayor parte de los casos.

Con respecto al problema NONLIN, los algoritmos AV y MLS (tabla 8.13) realizan la selección entre las acciones b y c (para el estado A) de forma aleatoria. Lo mismo ocurre para los algoritmos FH con horizonte inferior a 3, de ahí los resultados de la tabla 8.15. Aunque no se ha mostrado en las tablas, se ha observado un comportamiento similar para los métodos de control dual con K superior a 0,3

Tabla 8.15. Bonificación para los métodos FH con distinto horizonte

Problema	Método	N (Pasos en el futuro)				
		0	1	2	3	4
1DMAZE	F_AV	1,228585	1,35878	1,35878	1,35878	1,35878
	F_MLS	1,350784	1,35878	1,35878	1,35878	1,35878
	F_Q_MDP	1,350784	1,35878	1,35878	1,35878	1,35878
4x4	F_AV	3,677407	3,735495	3,736654	3,736654	
	F_MLS	3,736654	3,735406	3,736654	3,736654	
	F_Q_MDP	3,734477	3,735406	3,736654	3,736654	
TIGER	F_AV	-898,6487	-19,999262	-19,999262	-19,999262	19,641558
	F_MLS	-901,2841	-19,999262	-19,999262	-19,999262	19,641558
	F_Q_MDP	19,641558	16,414812	16,414812	16,414812	16,414812
NONLIN	F_AV	6,326803	6,326803	6,326803	7,200711	
	F_MLS	6,328266	6,326803	6,326803	7,200711	
	F_Q_MDP	6,326803	6,326803	6,326803	7,200711	

Otro factor a tener en cuenta es el tiempo empleado en las decisiones, pero dicho análisis es más relevante en problemas de mayor dimensión como los que se describen en el siguiente apartado. No obstante, en la tabla 8.16 se muestra, a modo de ejemplo, los tiempos necesarios para obtener una decisión en el caso 4x4. Cabe observar que el incremento de tiempos a medida que se aumenta el horizonte en los algoritmos FH es proporcional al factor de ramificación.

Tabla 8.16. Tiempo medio en segundos para el cálculo de una decisión para el caso 4x4

Algoritmo	Valores del parámetro K						
	0,2	0,3	0,4	0,5	0,6	0,7	0,8
ADM_AV	0,000474	0,000474	0,000420	0,000420	0,000342	0,000342	0,000247
ADM_MLS	0,000475	0,000475	0,000420	0,000420	0,000342	0,000342	0,000247
ADM_Q_MDP	0,000477	0,000477	0,000423	0,000423	0,000345	0,000345	0,000251
DM_AV	0,000332	0,000294	0,000236	0,000236	0,000236	0,000179	0,000179
DM_MLS	0,000332	0,000294	0,000236	0,000236	0,000236	0,000178	0,000178
DM_Q_MDP	0,000336	0,000296	0,000239	0,000239	0,000239	0,000182	0,000182
Algoritmo	Horizonte						
	0 pasos		1 paso		2 pasos		
F_AV	0,000187		0,000919		0,008949		
F_MLS	0,000370		0,001800		0,009609		
F_Q_MDP	0,000420		0,002059		0,010875		

8.3.3.2 Problemas de mayor dimensión con solución óptima no disponible

En los casos anteriores, se hacían notar los efectos específicos de los problemas a tratar, destacando los problemas de algunos decisores. Las soluciones óptimas están formadas por un reducido número de vectores que definen también un limitado número de regiones de decisión. Con este panorama, no es de extrañar que algunos métodos se aproximen mucho al óptimo. No obstante, en la realidad, las soluciones son mucho más complejas e interesa conocer la respuesta de los métodos heurísticos en este tipo de problemas. En estos problemas, debido a su dimensión, no se dispone de la solución óptima (POMDP) y se usará como cota superior el controlador omnisciente.

Se han utilizado como base cuatro problemas diferentes tomados como referencia por diversos autores. Estos son conocidos como:

- ✓ MIT. Problema de navegación en un entorno de oficinas que se corresponde con el entorno A de la referencia [Kaelbling 1996]. El modelo es similar al presentado en el capítulo cinco para el caso de Xavier. Consta de un total de 204 estados que representan las distintas poses que puede tener el robot. Existen 28 posibles observaciones dadas por los sensores virtuales del robot.

- ✓ CIT Problema de navegación similar al caso anterior pero con un entorno diferente [Kaelbling 1996]. En este caso el número de estados es de 284.
- ✓ ALOHA10. El aloha ranurado es una estrategia de control de acceso a un canal o medio de comunicaciones por varias estaciones. El objetivo es minimizar el número de colisiones. Cada paquete esperando para ser enviado se transmitirá en la próxima ranura temporal con probabilidad a . Si en un instante determinado el número de paquetes en espera para ser transmitidos por las distintas estaciones que comparten el medio es N , se sabe que la estrategia óptima es retransmitir con probabilidad $1/N$. El problema es que cada estación desconoce el número de mensajes en espera en las estaciones restantes. Las observaciones se corresponden con los posibles estados del medio (vacío, transmitiendo o colisión). Los estados del sistema POMDP son el número de posibles mensajes en espera combinado con el estado del medio en la ranura anterior. Las acciones son las probabilidades de transmisión que se reduce a la serie de valores $\{1/n, n \in \{1, 2, \dots, N\}\}$. Las probabilidades de transición se calculan considerando llegadas de paquetes de acuerdo con una distribución de Poisson. Entre otras referencias, se encuentra descrito en [Cassandra 1998].
- ✓ ALOHA30. Problema similar al anterior con N igual a 30 descrito en [Cassandra 1998]

De forma similar al apartado anterior, el número de estados, acciones y observaciones para cada problema se presenta en la tabla 8.17. El número de pasos en cada ensayo es 100 en todos los casos y cada valor de bonificación se obtiene como la media de 10.000 ensayos. Ese número de pasos es el suficiente para que en los problemas de navegación el robot alcance el destino en casi todos los casos. Se han mantenido los mismos estados iniciales que aparecen en las respectivas referencias así como el resto de los parámetros.

Tabla 8.17. Parámetros de los problemas de mayor dimensión

Problema	Parámetros					
	Nº de estados	Nº de acciones	Nº de observaciones	ρ	Factor de ramificación	Iteraciones por ensayo
MIT	209	4	28	0,99	108	100
CIT	284	4	28	0,99	108	100
ALOHA10	30	9	3	0,999	27	100
ALOHA30	90	29	3	0,999	87	100

La tabla 8.18 muestra los resultados obtenidos por los distintos algoritmos donde, al igual que en los casos anteriores, se presentan los mejores resultados para los algoritmos DM y ADM sobre el rango de valores de K , mientras que la influencia de este parámetro se refleja en la tabla 8.19,

Tabla 8.18. Bonificación para problemas de mayor dimensión.

Método	Problema			
	MIT	CIT	ALOHA10	ALOHA30
OMNI	0,988888	0,983384	145,373074	931,431731
MLS	0,957555	0,945407	125,528187	850,953443
AV	0,959031	0,944263	125,452797	850,071557
Q_MDP	0,907594	0,979274	126,730828	852,695960
F_AV (2 pasos)	0,987550	--	126,666918	886,681840
F_MLS (2 pasos)	0,968424	--	126,780768	883,457473
F_Q_MDP(2 pasos)	0,987746	0,982555	126,803981	884,561949
ADM_AV	0,959031	0,944263	125,452797	850,071557
ADM_MLS	0,957555	0,945407	125,528187	850,953443
ADM_Q_MDP	0,907594	0,979274	126,730828	852,695960
DM_AV	0,959031	0,944263	125,452797	850,071557
DM_MLS	0,957555	0,945407	125,528187	850,953443
DM_Q_MDP	0,907594	0,979274	126,730828	852,695960

Los mejores resultados, resaltados en negrita, los presentan los algoritmos FH, al igual que en el modelo de supervisión. Parece que, en problemas de dimensiones relativamente grandes, el comportamiento de estos métodos mejora en detrimento del resto de algoritmos.

Tabla 8.19. Bonificación DM-MLS con distinto K

Problema	K (Entropía umbral)							
	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
MIT	0,89887	0,95686	0,95735	0,95755	0,95755	0,95755	0,95755	0,95755
CIT	-0,66625	0,85888	0,93360	0,94540	0,94540	0,94540	0,94540	0,94540
ALOHA10	97,3933	104,688	112,625	118,072	125,190	125,528	125,528	125,528
ALOHA30	685,835	691,918	709,900	756,103	839,270	850,953	850,953	850,953

A pesar de los elevados valores del parámetro temporal ρ , se manifiestan de nuevo los buenos resultados de los algoritmos FH presentados en la tabla 8.20. En esta tabla también se puede constatar que el incremento del número de pasos constituye, salvo raras excepciones, una mejora de la solución.

Tabla 8.20. Bonificación para los métodos FH con distinto horizonte

Problema	Algoritmo	N (Pasos en el futuro)		
		0	1	2
MIT	F_AV	0,959031	0,974241	0,987550
	F_MLS	0,957555	0,973465	0,968424
	F_Q_MDP	0,907594	0,942696	0,987746
CIT	F_AV	0,944263	--	--
	F_MLS	0,945407	--	--
	F_Q_MDP	0,979274	0,979897	0,982555
ALOHA10	F_AV	125,452797	126,643640	126,666918
	F_MLS	125,528187	126,640689	126,780768
	F_Q_MDP	126,730828	126,789463	126,803981
ALOHA30	F_AV	850,071557	852,568316	886,681840
	F_MLS	850,953443	852,483949	883,457473
	F_Q_MDP	852,695960	852,659456	884,561949

En general, se puede concluir que hay algoritmos heurísticos que producen buenas políticas. Sin embargo, de entre los estudiados, no existe un único heurístico que predomine para cualquier problema. La naturaleza de las acciones implicadas en el proceso es un factor decisivo a la hora de elegir un sistema de decisión. En los

problemas donde existan acciones cuyo único objetivo sea la reducción de la entropía, las técnicas de AV y MLS no son, por lo general, muy adecuadas porque dichas acciones no serán nunca seleccionadas. Este es el caso de TIGER por ejemplo (tabla 8.13). No obstante estas técnicas pueden obtener buenos resultados cuando el objetivo de las acciones sea diferente como ocurre en los problemas de navegación (CIT y MIT). En ambas situaciones, la adición de los métodos con predicción, salvo en raras ocasiones, mejora las prestaciones. En particular, para problemas de relativa dimensión se ha observado siempre una mejora al aplicar los métodos F_*. Salvo estas consideraciones, es necesaria la realización de un análisis para establecer qué método es aplicable en cada caso.

Aparte de lo mencionado anteriormente, el tiempo empleado en el cálculo de la decisión es un parámetro que puede ser determinante a la hora de decidir el método a utilizar. En la tabla 8.21 se muestran los tiempos obtenidos en simulación.

Tabla 8.21. Tiempo medio en segundos para el cálculo de una decisión para el caso MIT

Algoritmo	Valores del parámetro K						
	0,2	0,3	0,4	0,5	0,6	0,7	0,8
ADM_AV	0,007817	0,000194	0,000058	0,000055	0,000056	0,000068	0,000055
ADM_MLS	0,004050	0,000151	0,000047	0,000044	0,000045	0,000043	0,000044
ADM_Q_MDP	0,008309	0,000621	0,000197	0,000203	0,000199	0,000198	0,000196
DM_AV	0,002153	0,000191	0,000060	0,000057	0,000054	0,000061	0,000056
DM_MLS	0,001808	0,000185	0,000045	0,000045	0,000054	0,000048	0,000045
DM_Q_MDP	0,003118	0,000431	0,000207	0,000208	0,000194	0,000199	0,000216
Algoritmo	Horizonte						
	0 pasos		1 paso		2 pasos		
F_AV	0,000097		0,041887				
F_MLS	0,000050		0,040454				
F_Q_MDP	0,000363		0,050992				

Tabla 8.22. Tiempo medio en segundos para el cálculo de una decisión para el caso ALOHA10

Algoritmo	Valores del parámetro K						
	0,2	0,3	0,4	0,5	0,6	0,7	0,8
ADM_AV	0,001758	0,001107	0,000499	0,000261	0,000032	0,000019	0,000017
ADM_MLS	0,001747	0,001106	0,000497	0,000263	0,000031	0,000017	0,000020
ADM_Q_MDP	0,001768	0,001120	0,000541	0,000295	0,000056	0,000037	0,000036
DM_AV	0,001212	0,000813	0,000362	0,000187	0,000027	0,000019	0,000020
DM_MLS	0,001217	0,000817	0,000360	0,000182	0,000026	0,000018	0,000016
DM_Q_MDP	0,001229	0,000829	0,000396	0,000216	0,000048	0,000035	0,000037

Tabla 8.22. Tiempo medio en segundos para el cálculo de una decisión para el caso ALOHA10

Algoritmo	Horizonte		
	0 pasos	1 paso	2 pasos
F_AV	0,000007	0,001721	0,044737
F_MLS	0,000007	0,001700	0,043917
F_Q_MDP	0,000027	0,002146	0,055551

8.4 Pruebas en el robot (tests de hipótesis)

El salto entre la compleja realidad del robot y la representación simbólica usada por los programas de control y supervisión es considerable. La mayor parte de los simuladores utilizan una representación simbólica similar a la de los programas de control y supervisión. A medida que la potencia de cálculo de los ordenadores se incrementa, se van construyendo simuladores que representan de una forma más precisa tanto el comportamiento del robot, como del entorno. No obstante, sigue siendo necesario, en última instancia, verificar el funcionamiento sobre el sistema físico.

Los tests de hipótesis se basan en la realización de experimentos controlados. Se manipula el entorno y el sistema de forma sistemática para observar la influencia de estas alteraciones en las prestaciones.

Usando MLS como decisor, se han establecido una serie de escenarios para analizar el comportamiento del robot ante ciertas situaciones. Entre otras, se han aplicado a aquellas situaciones descritas en el capítulo seis sobre los comportamientos deseados del robot:

- ✓ Si hay un obstáculo en el camino que los sensores del robot no pueden detectar, este chocará con el y reintentará seguir el camino. No obstante, el supervisor ha de considerar también la posibilidad de la existencia de un obstáculo no visible. Si el problema persiste, debe optar por seguir otra trayectoria alternativa.
- ✓ En la misma situación anterior, si el camino alternativo es muy parecido al actual debería elegir la trayectoria alternativa directamente.

Las pruebas se han llevado a cabo en el robot Xavier de la Universidad de Carnegie Mellon. En el transcurso de dichas pruebas se ha comprobado la poca utilidad de algunas acciones, como la relocalización que se ha eliminado del sistema. El principal motivo de las malas prestaciones de la acción de relocalización es que existen

demasiados obstáculos no modelados y el método debe ser completado con un mapa sensorial por ejemplo.

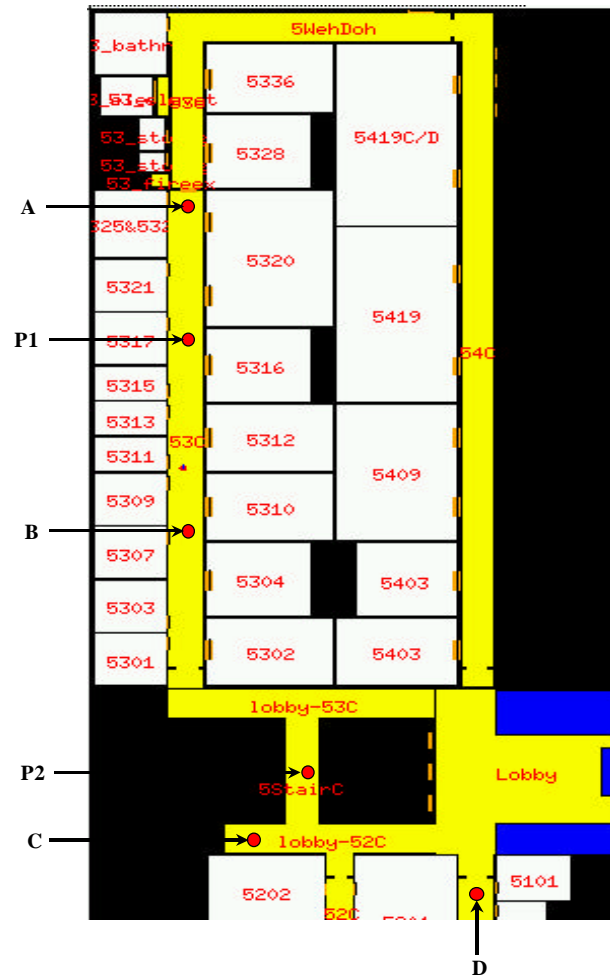


Figura 8.3 Parte del plano de la 5ª planta de Wean Hall (edificio del dep. de CS de CMU).

El plano de la figura 8.3 se corresponde con la 5ª planta del edificio del departamento de “Computer Science” de la Universidad de Carnegie Mellon donde se hicieron las pruebas. El sistema de navegación utiliza este plano para llevar a cabo su tarea. Se han seleccionado varios escenarios diferentes, cuatro de los cuales se describen a continuación.

Escenario 1

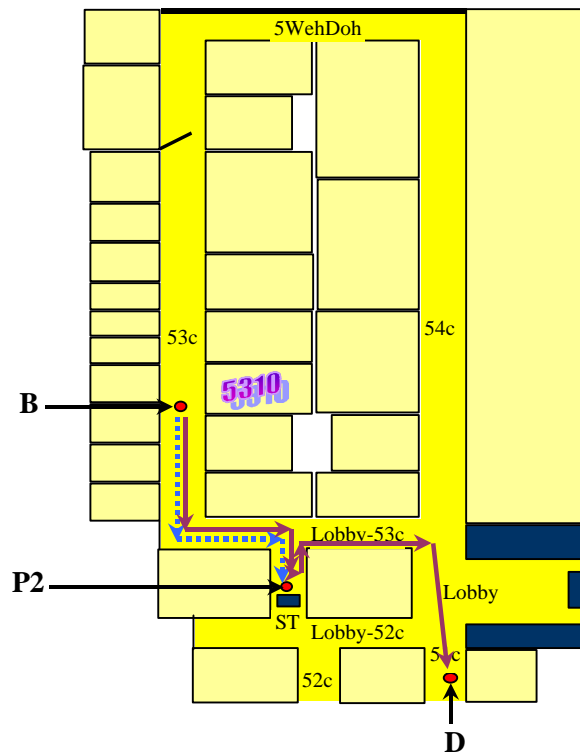


Figura 8.4. Escenario 1

En el primer escenario, el robot parte del punto B y trata de alcanzar el punto D. El camino planeado es a través de los pasillos “53c”, “lobby-53C”, “5StairC”,

“lobby-52C” como muestra la línea en azul de la figura 8.4. El planificador selecciona este camino tratando de evitar el paso del robot a través del “Lobby” que es una zona conflictiva. El problema surge en el pasillo “5StairC” que se encuentra obstruido en el punto P2.

El robot no tiene mayores dificultades mientras se mueve a través de “53c” y “lobby-53C”, pero al llegar al punto P2 y no poder avanzar, el monitor de posición empieza a dispararse. Por cada observación de este monitor se incrementa la probabilidad de que el robot se encuentre ante una situación de bloqueo, hasta que llega a un punto donde la distribución de probabilidades indica que se debe solicitar al planificador un nuevo camino. Se incrementa el coste del arco en el que se encuentra, se planifica de nuevo,⁹ y como el coste del camino actual es bastante similar al de la alternativa de ir a través del “lobby”, el nuevo camino será a través del “lobby”, realizando al final la trayectoria que se muestra en trazo sólido en la figura 8.4.

Escenario 2

El origen es de nuevo en el punto B y el destino el punto A. El pasillo “53C” se encuentra bloqueado en el punto P1. Como el camino alternativo es de mucho mayor coste, el planificador, aun aumentando el coste del arco, obtiene el mismo camino. Al obtener el mismo camino, se ejecuta antes la acción de **GO**. Detecta que no hay espacio suficiente para pasar por ese camino, la probabilidad de camino cortado se vuelve a incrementar y ahora se obtiene un camino nuevo. Esta situación se muestra en la siguiente figura donde se han usado los mismos trazos de referencia que en el caso anterior. La diferencia con el caso anterior es que, aquí merece la pena intentarlo de nuevo y, como el planificador no proporciona un camino alternativo sino el mismo, se ejecuta la acción **GO**, que falla. En el caso anterior, como el coste del camino alternativo es similar al actual, directamente se opta por el alternativo.

⁹ En realidad se crea un nuevo nodo (posición del robot) con un par de arcos asociados y el planificador insiste en el mismo camino al no estar incluido el arco con problemas. El supervisor detecta esta situación, cambia el peso del nuevo arco y solicita otro camino.

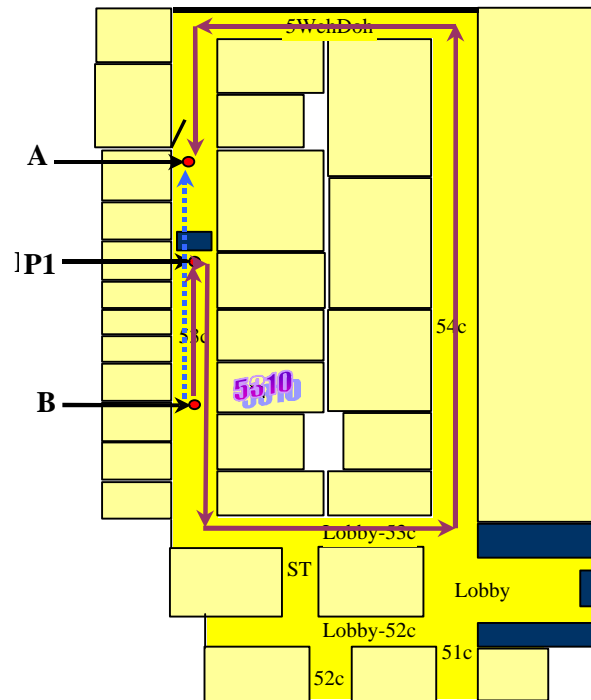


Figura 8.5. Escenario 2.

Escenario 3

Continuando con los mismos objetivos, con la diferencia que esta vez el robot sí puede pasar. La situación aparece descrita en la figura 8.6 usando los mismos tipos de línea para el camino original y final que en los casos anteriores. En este caso, en principio también se obstruye el camino pero luego se aparta el obstáculo, el robot detecta la posibilidad de paso y vuelve a intentarlo, esta vez con éxito.

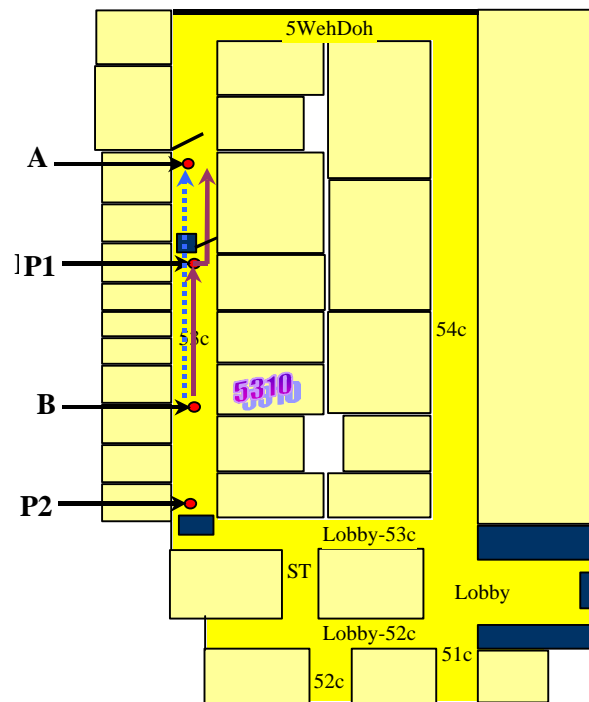


Figura 8.6 . Escenario 3

Escenario 4

Este es un caso típico en el cual no hay alternativa posible para llegar al destino, con lo cual se tiene que abortar la operación. Como en los casos anteriores, los mensajes del monitor de posición hacen cambiar la acción nominal y buscar una alternativa. Como la alternativa es muy costosa dado que hay mucha diferencia entre ambos caminos, el planificador obtiene el mismo camino (merece la pena intentarlo de nuevo) y se verifica el camino, encuentra apertura y lo intenta de nuevo. Detecta otra vez bloqueado, se incrementa de nuevo el coste del arco y ahora se obtiene un camino

nuevo(53c → lobby-53c → 54c → 5WehDoh). Al final, este nuevo camino aparece bloqueado en P2, lo detecta y da por imposible finalizar la tarea.

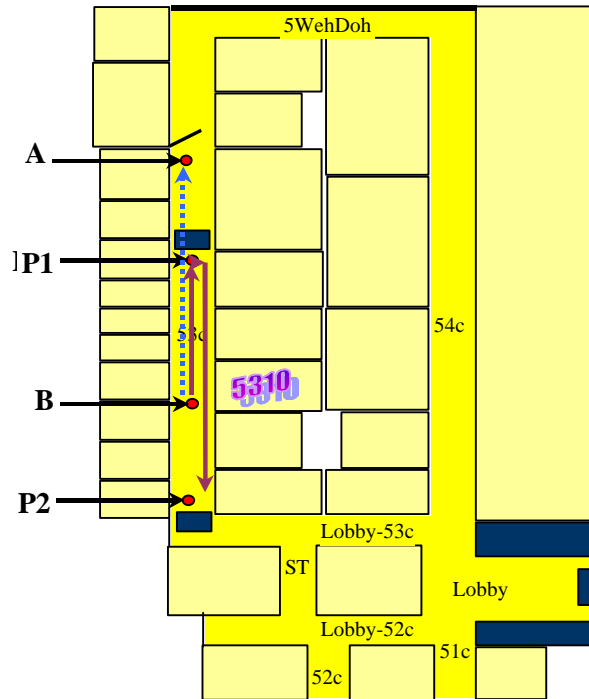


Figura 8.7. Escenario 4

Durante y después de la realización de las pruebas con el robot se ha modificado el modelo, añadiendo más estados para poder distinguir ciertas situaciones como el caso de la existencia de camino alternativo y se ha constatado la necesidad de que éste sea fácilmente modificable. También fue necesario un reajuste de los parámetros para lo cual fue muy útil la tabla de solución del COMDP (apéndice D).

9 Conclusiones

La inteligencia artificial ha encontrado en el dominio de los robots móviles una plataforma idónea para el desarrollo y verificación de algoritmos relacionados con temas de supervisión, planificación y aprendizaje. Debido a limitaciones de los sensores, la mayor parte de la información adquirida a través de estos es incompleta e imprecisa. Su entorno no es casi nunca ni modelable ni predecible y las acciones tampoco son deterministas.

La solución de los problemas de supervisión y detección de fallos en robótica móvil requiere la integración de varias metodologías. A lo largo de la bibliografía sobre el tema se puede observar que el tratamiento de fallos en robótica móvil está muy ligado a la arquitectura utilizada en la navegación y a las tareas de obtención de síntomas, detección, diagnóstico y recuperación no están tan aisladas como en los sistemas industriales convencionales.

9.1 Contribuciones

En esta tesis se han presentado dos aproximaciones a la solución del problema de supervisión y tratamiento de situaciones de excepción en robots móviles. En la primera (determinista) se utiliza un sistema jerárquico de monitores. Además se presenta un conjunto original de éstos para detectar situaciones de excepción durante la navegación. En resumen, se presentan las siguientes innovaciones:

- ✓ *Formalización de una estructura jerárquica de monitores para la monitorización robusta de las tareas de navegación.*
- ✓ *Conjunto original de monitores para detectar situaciones de excepción durante la navegación.*

En la segunda aproximación se usa un modelo estocástico que tiene en cuenta la incertidumbre acerca del estado del sistema proveniente del ruido asociado a las observaciones y la incertidumbre en el resultado de las acciones. Para ello, utilizando como observadores los monitores ya desarrollados, se ha construido un modelo basado

en procesos de Markov parcialmente observables. De forma breve, se presentan las siguientes innovaciones:

- ✓ *Modelo de tratamiento de situaciones de excepción basado en procesos de Markov parcialmente observables. Dicho modelo, además de tener en cuenta el coste de las acciones, incluye el tratamiento de la incertidumbre a distintos niveles:*
 - *Incertidumbre en las observaciones*
 - *Incertidumbre en el resultado de las acciones.*
- ✓ *Formalización de sintaxis para la definición simplificada del modelo mediante probabilidades y costes.*
- ✓ *Sistema de decisión heurístico basado en el anterior modelo, teniendo en cuenta el coste de las acciones de recuperación.*

9.2 Modelo determinista

Con este modelo se ha presentado una primera solución a la monitorización del funcionamiento de un robot móvil, usando una estructura jerárquica de monitores para detectar excepciones. En la parte más alta de esta arquitectura se sitúan los monitores más generales que detectan un gran número de excepciones, mientras que en capas inferiores se encuentran monitores más específicos que detectan situaciones concretas. El disparo de un monitor de bajo nivel provee más información que el disparo de los de alto nivel, pues el número de posibilidades en cuanto al estado del robot es menor. Además, los monitores específicos son mucho más rápidos al verificar parámetros bien determinados.

Si bien no se asegura una cobertura completa de todo el espacio de excepciones, un gran número de ellas quedan cubiertas por los monitores de alto nivel. Para el desarrollo de algunos monitores se necesita un conocimiento detallado del funcionamiento del robot y el entorno en el que se mueve, mientras que para otros basta con conocer datos generales. Entre estos últimos se puede destacar el monitor de tiempo que sólo necesita una cota superior de la duración estimada de la tarea en curso. Lo ideal sería obtener estos parámetros mediante un mecanismo de aprendizaje, pero esto no se engloba dentro de los objetivos de esta tesis.

El esquema desarrollado permite la fácil adición de nuevos monitores a medida que se van identificando nuevas excepciones y además se puede aplicar en otros

dominios. Debido a la dificultad de encontrar síntomas que separen las excepciones de las situaciones nominales de forma fiable, queda pendiente el problema de las falsas detecciones que pueden surgir. Entre otros parámetros, será necesario tener en cuenta el coste de aplicar un algoritmo de recuperación en una situación equivocada. Todos estos detalles se han tenido en cuenta en la elaboración de un modelo estocástico.

9.3 Modelo estocástico

Debido a la dificultad para obtener síntomas que permitan discernir con certeza entre las situaciones en las que puede encontrarse el agente, se ha elaborado una aproximación estocástica presentada como alternativa al modelo determinista. El nuevo modelo de tratamiento de situaciones de excepción se basa en procesos de Markov parcialmente observables. Dicho modelo además de tener en cuenta el coste de las acciones incluye el tratamiento de la incertidumbre a distintos niveles:

- ✓ Incertidumbre en las observaciones
- ✓ Incertidumbre en el resultado de las acciones.

Por otra parte, se ha formalizado la definición simplificada del modelo mediante probabilidades y costes, utilizando una sintaxis basada en la descrita para los POMDP. Por último, se ha desarrollado un sistema de decisión basado en el anterior modelo junto con acciones de recuperación.

El problema de detección de errores en la navegación de robots móviles y recuperación ante éstos ha sido tratado por otros investigadores e incluso se han desarrollado tesis como es el caso de la doctora Elisabeth Stuck [Stuck 1992]. No obstante, el enfoque aquí presentado es completamente diferente tanto al tratar la incertidumbre acerca de la validez de la información proveniente de los observadores, como en la toma de decisiones.

Si bien en esta tesis se ha trabajado con un modelo determinado, la arquitectura desarrollada es general y puede aplicarse a otros problemas similares sin más que cambiar el modelo (observaciones, acciones, estados) junto con sus parámetros (estadísticos que caracterizan el modelo). Aunque los parámetros del modelo han sido seleccionados basándose en la experiencia, en otros casos pueden ser aprendidos con métodos de aprendizaje automático tales como aprendizaje por refuerzo. El sistema de supervisión de la figura 9.3 es válido para todos los problemas

que se puedan modelar mediante un POMDP simplificado. No hay más que sustituir la definición del modelo por la del nuevo proceso.

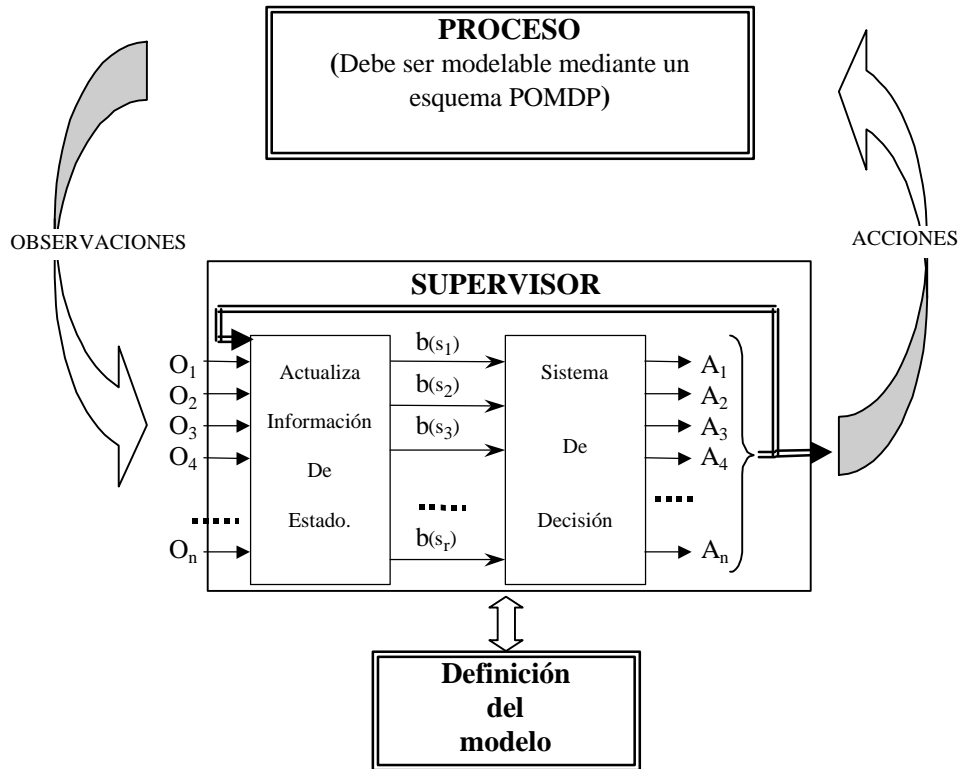


Figura 8.3. El sistema es válido para cualquier proceso que sea posible modelar como un POMDP

Para modelar el sistema es necesario:

1. Definir los estados, observaciones y acciones.
2. Estimar los distintos parámetros. es decir, seleccionar de forma correcta las probabilidades de transición, observación, etc.

3. Por ultimo, hay que calcular la solución del COMDP asociado al POMDP y seleccionar el método de decisión que mejores prestaciones obtenga con ese modelo. Dicha solución, junto con el modelo, va a ser usada por el supervisor.

9.4 Sistemas de decisión

Ante la imposibilidad que existe en la actualidad de obtener la solución exacta del POMDP para problemas de relativa complejidad, se suelen usar aproximaciones heurísticas. Existen diversas estrategias de decisión heurísticas que aproximan la solución óptima del POMDP. Algunas de ellas aprovechan características especiales del modelo a tratar [Washington 1998] que simplifican los cálculos, pero no son aplicables más que en cierto tipo de sistemas. Otras, por el contrario, son aproximaciones basadas en COMDP o métodos de aprendizaje [Washington 1996] [Kaelbling 1996].

La idea común que subyace bajo algunos de estos métodos es resolver el problema como si se tratara de un COMDP para obtener una política general $G_{co}(b)$. A diferencia del POMDP, la política así obtenida asigna a cada estado una acción y garantiza que esa política sea óptima si se conoce con certeza en todo momento el estado del robot.

Se han desarrollado métodos de decisión que mejoran las prestaciones de los métodos heurísticos existentes en gran parte de los problemas sobre los que se han probado.

El funcionamiento de estos algoritmos se basa en la introducción de métodos de predicción combinados con los algoritmos de decisión heurísticos ya existentes (AV, MLS, etc). El proceso de predicción de la función valor se considera como el de creación de un árbol. Cada nodo del árbol representa un estado (información de estado) diferente al que se puede llegar. La decisión se toma en función de la rama del árbol asociada a la acción con mayor bonificación. El árbol se construye contemplando todas las posibles combinaciones de acciones a ejecutar y observaciones que se pueden obtener. La profundidad del árbol viene determinada por el tiempo que se disponga para calcular la decisión. La función valor para el último paso se estima aplicando alguno de los métodos de decisión heurísticos clásicos

9.5 Líneas futuras

Como en todo desarrollo, a pesar de la obtención de resultados relativamente buenos, siempre hay aspectos que se pueden mejorar y, en la elaboración de esta tesis, se ha observado que hay dos temas bastante interesantes con vistas a investigaciones futuras: consideración de costes variables con el tiempo y aprendizaje automático de los parámetros del modelo.

En el modelo se ha supuesto que los costes de las acciones son constantes durante la ejecución de las tareas. Esta restricción en el modelo aquí tratado no causa, en principio, ningún problema, pero puede hacerlo al aplicar esta metodología en un caso general. Sería por tanto interesante eliminar la restricción de que la política sea estática. Esta relajación de condiciones podría producir beneficios incluso en el caso de supervisión simplificando el modelo de estados.

La elaboración del modelo, junto con las probabilidades, de transición y la elección de las bonificaciones, es una tarea laboriosa que sería interesante poder generar o aprender de forma automática. Mientras que el sistema de bonificación debe ser establecido por el usuario para manifestar sus preferencias, cabe la posibilidad de automatizar el modelo y las probabilidades de transición. Otra futura línea de investigación por lo tanto es la de aprendizaje automático tanto del modelo, como de las probabilidades de transición. Una posible solución podría estar basada en un sistema de procesos ocultos de Markov HMDP hacia donde estamos enfocando nuestra futura investigación.

APÉNDICES