UNIVERSIDAD REY JUAN CARLOS

Ingeniería Técnica en Informática de Sistemas

Escuela Superior de Ciencias Experimentales y Tecnología

Curso académico 2003-2004

Proyecto Fin de Carrera

Comportamiento sigue persona con visión direccional

Tutor: José María Cañas Plaza

Autor: Roberto Calvo Palomino

Madrid 2004

A mis padres y hermano,

por la oportunidad que me dieron de

estudiar lo que verdaderamente quería

y por el apoyo incondicional.

Gracias.

Agradecimientos

Quiero dar las gracias a todos los miembros del Grupo de Robótica de la Universidad Rey Juan Carlos por su apoyo y colaboración. En particular agradezco la ayuda recibida por parte de José María Cañas y Vicente Matellán.

De manera especial quiero agradecer a José María el apoyo, dedicación y conocimientos facilitados para la realización de este proyecto.

Resumen

El presente proyecto aborda el desarrollo de un comportamiento de seguimiento de personas en tiempo real. Este comportamiento se basa en seguir a una persona por un entorno de oficinas sorteando los obstáculos. Para ello se utiliza una cámara como principal sensor para determinar donde se encuentra la persona a seguir y una base motora para aproximarse a la posición de dicha persona. Este proyecto va orientado hacia la línea de investigación que intenta introducir la robótica en los hogares basándose en la interacción con personas.

Es necesaria una gran vivacidad en el comportamiento para que el robot tenga rapidez en sus acciones. Para conseguir dicha vivacidad se usará un control reactivo en el robot, para cada acción que se produzca en el entorno deberá responder con una reacción lo más rápida posible. La navegación está basada en la visión ya que mediante ésta se localiza a la persona que queremos seguir. Para localizar a la persona se utiliza un filtro de color.

Para realizar este comportamiento se ha diseñado un reparto de tareas en dos bloques. El bloque de seguimiento visual se encarga del manejo de un cuello mecánico que mediante un control proporcional centra al objetivo en la imagen. El otro bloque de seguimiento de la base se encarga del control de la base motora del robot, cuyo objetivo es alinearse con la dirección del cuello. Los dos bloques no son independientes uno de otro, y su funcionamiento en concurrente materializa el comportamiento deseado. Cada bloque tiene a su vez sub-bloques que pueden ser perceptivos (encargados de obtener información) o actuadores (toman decisiones sobre el robot). Estos sub-bloques se denominan esquemas.

Para llegar a conseguir un comportamiento lo más óptimo posible, hemos realizado numerosos experimentos sobre la plataforma real. En los simuladores normalmente el comportamiento se ejecuta correctamente. Pero la ejecución en los robots reales nunca es perfecta y siempre es necesario realizar ciertos ajustes que son fruto de las numerosas pruebas realizadas.

Este proyecto ha sido desarrollado sobre una plataforma GNU/Linux, utilizando las herramientas que proporciona este sistema operativo, desde el editor para escribir los programas, hasta el compilador de C para generar el programa final que se ejecuta en el Pioneer.

Índice general

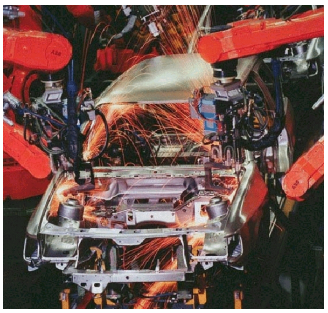
Índice de figuras

Capítulo 1

Introducción

La palabra robot proviene de la palabra eslava *robota* que se refiere al trabajo realizado de manera forzosa. Tenemos conocimiento de los primeros robots móviles en los años 70, donde el hombre tenía la intención de crear objetos artificiales con un elevado grado de autonomía. Tuvieron una gran utilización en los años 80, donde se implantaron las primeras máquinas en las fábricas con un objetivo sencillo y para facilitar las tareas repetitivas. A partir de la década de los 80, es familiar ver a robots y autómatas en el sector de la industria.

La industria fue pionera en los llamados robots manipuladores. Los primeros manipuladores eran controlados por humanos, de tal forma que el humano le comandaba las órdenes al manipulador. Los manipuladores más comunes que se pueden encontrar en una planta industrial se dedican a mover herramientas, piezas, cortar, soldar, etc. Estos manipuladores evolucionaron de tal forma que no necesitaban el control humano, ellos mismos eran capaces de tomar decisiones sobre sus movimientos. A partir de este momento se podía hablar de autonomía aunque ésta fuese muy reducida. El primer tipo de brazo articulado que existió fue el PUMA (*Programmable Universal Manipulator for Assembly*). Dos de las principales ventajas que ofrece la brazo robotizado frente el brazo humano son la precisión y rapidez.



(a)

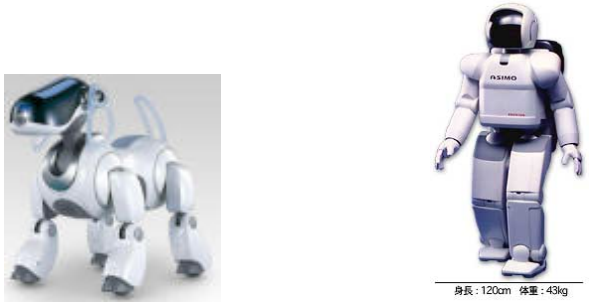
(b)

Figura 1.1: Robots soldadores en una factoría (a) y brazo transportando materiales (b).

Uno de los campos de investigación que está captando interés creciente es la

robótica móvil. Dentro de la robótica móvil tenemos dos campos bien diferenciados. Los robots móviles teleoperados son aquellos que no poseen autonomía y todos sus movimientos son comandados desde la distancia por el ser humano. Desde el nacimiento del robot teleoperado, éstos se han venido usando para desempeñar tareas peligrosas como desactivación de bombas, rastreo de minas, exploraciones submarinas o exploraciones en otros planetas como Marte (*Spirit y Opportunity*). Estos últimos cada vez tienen mayor autonomía pero en esencia son teleoperados. Por otro lado tenemos a los robots totalmente autónomos, no necesitan la ayuda de ningún ser humano para funcionar y moverse por un entorno, como los robots guía de un museo o biblioteca[1].

Cabe destacar que el campo de la inteligencia artificial siempre ha estado altamente relacionado con el de la robótica. Esta última se apoya en el razonamiento artificial para crear comportamientos autónomos y a la vez que sean capaces de ir aprendiendo. Como ejemplo de esto último tenemos a las mascotas robotizadas que están dotadas de cierta inteligencia y capacidad de aprendizaje.



(a) (b)

Figura 1.2: Mascota AIBO (a) y el famoso androide ASIMO (b).

La robótica no es más que otra disciplina que cada vez tiene más auge en la sociedad debido a la gran autonomía que pueden llegar a tener los robots. Los robots serán más útiles en el futuro y más populares cuanta mayor autonomía posean, como el androide ASIMO que se puede observar en la figura 1.2(b). El interés actual es dotar a los robots de la mayor autonomía posible para desempeñar labores en el hogar como: planchar, barrer, lavar o cualquier otra tarea doméstica.

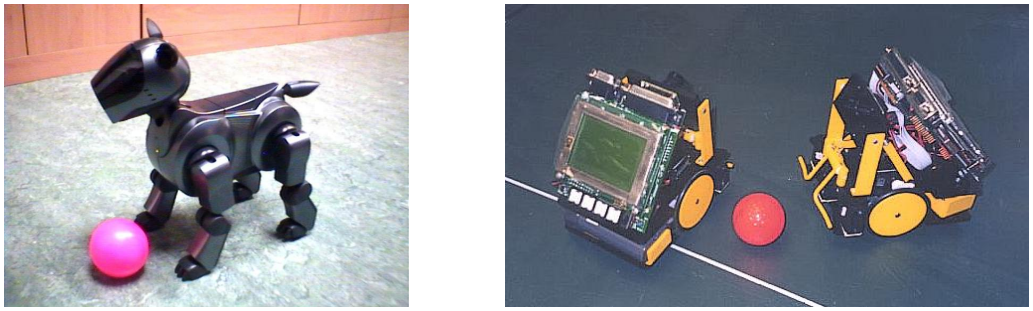
1.1. Grupo de Robótica de la URJC

El grupo de Robótica[2] de la Universidad Rey Juan Carlos está compuesto por profesores y alumnos. Los intereses de este grupo se basan en la *generación de comportamiento artificial en robots*. Conceptualmente este grupo se dedica a la programación de robots, lo que conlleva tener un enlace directo con la Ingeniería Informática de Sistemas. En este

[1] http://www.spacedaily.com/news/robot-03z.html
[2] http://gsyc.escet.urjc.es/robotica

grupo se dispone de 2 robots de interior Pioneer, 3 perros de Sony, 6 robots EyeBot, 30 robots Lego y 2 cuellos mecánicos.

Una de las líneas de investigación del grupo de Robótica es crear un equipo de robots capaz de participar en la RoboCup[3] [Peño, 2003]. Este campeonato mundial es uno de los más conocidos en robótica móvil. Inicialmente comenzaron usando los robots EyeBot y actualmente los equipos se forman con los famosos perros mascota de Sony. En esta línea se han desarrollado algunos proyectos fin de carrera como el comportamiento sigue pelota [San Martín, 2002], comportamiento sigue pelota con visión cenital [Martínez, 2003] o el comportamiento sigue pared [Gómez, 2002].



(a) (b)

Figura 1.3: Mascota AIBO de Sony (a) y Eyebot.(b)

Otra de las líneas de investigación en las que está centrado el grupo es la creación de *comportamientos autónomos en entornos de oficina* [Isado, 2004]. Justamente en este último campo se ubica este proyecto de seguimiento de persona. Actualmente hay varios proyectos en paralelo en el grupo de robótica de localización, navegación reactiva y navegación deliberativa. Por citar algunos tenemos: comportamiento sigue persona [Ortiz, 2004], evitación de obstáculos basada en ventana dinámica [Lobato, 2003] y navegación global utilizando método del gradiente [Isado, 2004].

Este tipo de comportamientos se programan sobre una plataforma llamada JDE (*Jerarquía Dinámica de Esquemas*) que se ha desarrollado íntegramente en el grupo [Cañas, 2002]. Esta arquitectura de control se basa en pequeños esquemas o tareas. Cada esquema realiza una acción muy determinada y la unión de varios esquemas dan lugar a un comportamiento complejo. Estos esquemas o tareas son de dos tipos, perceptivos si se dedican a obtener información sensorial y actuadores si su función es establecer parámetros de control al robot u otros actuadores. La gran ventaja de esta arquitectura de control es que escala muy bien a comportamientos crecientemente complejos.

Actualmente la robótica móvil sigue siendo un campo de investigación donde se realizan números avances y donde centran numerosas universidades su esfuerzo y dedicación.

[3]http://www.robocup.org

1.2. Comportamiento sigue persona con visión direccional

La motivación de este proyecto es comenzar a crear comportamientos en robots móviles capaces de desenvolverse en entornos de oficina. Un comportamiento útil en este escenario es el de seguir a una persona. Este proyecto consiste en que el robot siga a una persona, identificando a ésta mediante una camiseta de color distinguible, y que sea capaz de sortear los posibles obstáculos que existan en el entorno.

La localización de la persona se puede realizar mediante una cámara que está situada encima de un cuello mecánico. Gracias al cuello se dispone de una visión direccional en dos sentidos (vertical y horizontal) similar a un ojo humano. Gracias a que podemos mover la cámara mediante el cuello mecánico podemos buscar a la persona cuando ésta no esté localizada. La detección de obstáculos se realiza gracias a los sensores láser y de ultrasonido que incorpora el robot. En un planteamiento de un comportamiento sigue persona normalmente no existe una planificación clásica de trayectorias ni razonamientos lógicos encadenados, se decantan más por utilizar controles reactivos.

El seguimiento se realiza utilizando un robot de interior, en concreto el Pioneer 3. Se trata de un robot potente y manejable, ideal para entornos de oficina e interiores. El comportamiento se desarrollará utilizando la arquitectura JDE, en la cual basa sus trabajos de esta línea de investigación el Grupo de Robótica de la Universidad Rey Juan Carlos.

La utilidad de este proyecto va encaminada a conseguir una interacción natural con las personas. En una casa, un robot puede desempeñar muchas funciones. Una de ellas puede ser, sin lugar a duda, el seguimiento a su dueño.

Figura 1.4: Robot siguiendo a una persona

1.2.1. Navegación local

La navegación local consiste en avanzar en cierta dirección evitando el choque con obstáculos próximos del entorno que se perciben con los sensores. Esta navegación es muy usada en robots que tienen que desenvolverse en entornos de oficinas. También podemos

denominar la navegación local como navegación reactiva ya que se intenta evitar todos los obstáculos que podemos observar con los sensores.

Existen muchas técnicas de navegación reactivas, desde las más sencillas que se basan en girar al lado opuesto de donde se ha detectado un obstáculo hasta técnicas basadas en campos de potenciales o algoritmos inteligentes de evitación.

En este proyecto se utilizará una variante del algoritmo de navegación `VFF` [Borenstein y Koren, 1989]. Este tipo de navegación es local, es decir, se deciden las acciones dependiendo de la información que capturan por sus sensores. Esta técnica consiste en crear un campo de fuerzas virtuales alrededor del robot. Este campo de fuerzas está compuesto básicamente por dos fuerzas, una repulsiva y otra atractiva. La fuerza repulsiva se construye teniendo en cuenta la cercanía de los obstáculos, cuanto más cerca se encuentre el obstáculo mayor será la fuerza repulsiva sobre el robot. La fuerza atractiva suele ser una fuerza fija o variable que hace que el robot se dirija en esa dirección.

Además existen otras técnicas como LVM [GSyC, 2004] (se basa en dividir el entorno en carriles) o CVM [GSyC, 2004] (se basa en realizar trayectorias circulares).

1.2.2. Seguimiento de personas

Actualmente existen varias técnicas aplicables al seguimiento de una persona u objeto. Las primeras técnicas que aparecieron con este fin son bastantes sencillas y poco robustas ya que definen un entorno totalmente conocido. Lo que se hacía era crear un entorno controlado, de tal forma que todo el entorno estaba pintado de un mismo color y por tanto la persona a seguir tenía un color distinto. Aplicando un filtro sencillo de bordes se puede saber donde se encuentra el objetivo a seguir. El problema de esta técnica es obvio, no se puede utilizar cualquier entorno, tiene que ser un entorno artificial.

Otro modo de realizar un seguimiento de personas es disponer de un entorno totalmente natural, sin modificación alguna y usar un distintivo inequívoco en la persona a seguir. Un buen distintivo suele ser una prenda de ropa de un determinado color. En principio este modo de seguimiento presenta problemas con la calidad del filtro y con la existencia en el entorno de objetos del mismo color que el objetivo. Este modo de detectar a la persona es el que se ha utilizado en el presente proyecto.

Una de la técnicas más complejas y avanzadas que existen en el seguimiento de personas es el uso conjunto de la detección de caras [Worz, 1997] y el anclaje (*anchoring*) [Fink y Sagerer, 2002]. La detección de caras se realiza mediante segmentación de la imagen y obteniendo la silueta de dichas caras. Esta técnica es muy avanzada pero tiene el inconveniente de que el seguimiento se ha de realizar siempre de frente a la cara de la persona a seguir.

El *anclaje multimodal* es una técnica moderna que trata de fijar el estímulo *persona* en observaciones de dos sensores diferentes de distinta naturaleza. Por ejemplo si el análisis de la imagen nos detecta una cara de una persona, y la lectura de los lasers nos informa que tenemos dos obstáculos pequeños en frente del robot, el *anclaje* nos resuelve si esos dos estímulos pertenecen a un mismo objeto del mundo real, es decir, si los obstáculos que detecta el láser se tratan de las piernas de la persona que tenemos delante.

En otros artículos [García y Bustos, 2001] tratan el seguimiento de personas como un concepto muy parecido al que se presenta en este proyecto. Básicamente se trata de dos bloques, uno que controla la visión del objetivo y otro que controla la base motora. Además

optan por realizar segmentación en la imagen para obtener el objetivo aún existiendo otros objetos del mismo color.

Sin embargo otras investigadores [Worz, 1997] optan para la detección del objetivo mediante visión, combinando filtros de color e histogramas para separar al objetivo del entorno. Esto implica una mayor calidad en el comportamiento ya que el filtro aporta información más fiable, pero este método conlleva un mayor coste computacional.

La presente memoria detalla todos los aspectos relevantes que conlleva el desarrollo de este comportamiento de seguimiento. Esta memoria se compone de 5 capítulos. El capítulo 2 trata los objetivos y requisitos planteados a la hora de realizar el proyecto. El capítulo 3 ofrece una descripción del entorno de trabajo sobre el que se ha realizado el comportamiento, tanto a nivel hardware como software. La descripción informática, implementación y detalles del comportamiento se exponen en el capítulo 4. Por último en el capítulo 5 se reflejan las conclusiones extraídas sobre la realización de este proyecto y las posibles líneas futuras de mejora que se pueden aportar.

Capítulo 2

Objetivos y Metodología

Una vez presentado el contexto genérico en el capítulo anterior, en este capítulo se describen los objetivos del presente proyecto, así como los requisitos y la metodología empleada en la creación del comportamiento sigue-persona con visión direccional.

El objetivo principal de este proyecto es la creación de un comportamiento sigue persona con visión direccional. Este comportamiento debe ser capaz de evitar obstáculos en el seguimiento. El método de navegación se basa en `VFF` sobre el robot Pioneer utilizando para ello la arquitectura de control JDE desarrollada por el grupo de robótica de la URJC que se describirá en detalle en el capítulo 3.

2.1. Objetivos

El principal objetivo que se persigue en este proyecto es realizar un comportamiento de seguimiento de personas sin chocar con ningún obstáculo. Además, este comportamiento debe funcionar en un robot real de interiores.

El objetivo global es generar el comportamiento de seguimiento en un robot real. Dicho comportamiento consiste en seguir a la persona de tal modo que si ésta gira a la derecha el robot girará a la derecha. Además el comportamiento trata de tenerlo a una cierta distancia prudencial de tal modo que el robot debe detenerse si tiene a la persona muy cerca. De igual modo el robot debe buscar a la persona si ésta se pierde. Además este comportamiento debe hacer todo lo anterior sin chocarse con ningún obstáculo y con suavidad en sus movimientos. Es importante destacar que el compromiso de seguir a la persona y evitar obstáculos pueden ser a veces contradictorios.

El objetivo global anteriormente descrito se puede articular en tres subobjetivos: diseño, implementación y experimentación. El objetivo inicial del diseño es la documentación y puesta al día de las técnicas e investigaciones que se han realizado para este tipo de comportamientos. Como objetivo secundario del diseño tenemos la creación de dos bloques: uno que se encargue del seguimiento visual del objetivo y un segundo bloque que se encargue del seguimiento de la base motora.

Como objetivo de implementación tenemos el desarrollo de programas para poder generar el comportamiento y que éste se comunique con los diferentes dispositivos y el robot. Unido al punto anterior tenemos el hecho de probar y ensayar con la arquitectura software JDE. Esta arquitectura es reciente en el grupo de robótica de la URJC y este proyecto va a servir para probarla, testearla y encontrar nuevas funcionalidades que doten

de una mayor robustez a la plataforma.

Como objetivo enfocado a la experimentación se debe tener en cuenta que en la investigación en robótica se necesita inevitablemente de experimentos más allá del uso de simuladores o diseños lógicos. Los experimentos en un proyecto de investigación aportan numerosas mejoras y ayudan a afinar bastante los comportamientos.

Por último como objetivo personal tenemos el hecho de aprender y conocer como se realiza un proyecto de investigación y lo que eso supone.

2.2. Requisitos

El desarrollo del proyecto estará guiado por los objetivos comentados anteriormente y deberá ajustarse a los requisitos que comentaremos a continuación para asegurar un buen comportamiento sobre el robot.

Estos requisitos son:

1. Como se ha comentado en el apartado de objetivos, se trata de implementar un comportamiento en un robot real. Pues bien, este robot real se trata del Pioneer 3. Es un robot idóneo para la navegación por interiores y oficinas. Este requisito es posible debido a la existencia de robots reales en el grupo de robótica.

2. En cuanto al software, se ha de desarrollar el comportamiento del robot en la plataforma denominada JDE que se explicará detalladamente en el siguiente capítulo. La utilización de esta plataforma facilita el desarrollo del comportamiento.

3. Para la detección de la persona a seguir se utilizará la visión. Mediante la visión es más fácil detectar la existencia de nuestro objetivo que cualquier otra lectura sensorial. Esta elección facilita el desarrollo del comportamiento pero implica un mayor coste de computación al tener que analizar las imágenes.

4. Para que el seguimiento se produzca correctamente, debemos tener una gran robustez frente a cambios de iluminación en el entorno. No debe implicar en el funcionamiento del comportamiento que estemos situados en una zona de mucha o poca intensidad de luz.

5. Para que tenga sentido la realización de este comportamiento, éste debe tener una gran vivacidad. Debe responder rápido a las acciones que se producen en el entorno. No tiene sentido este comportamiento si no conseguimos una ejecución en tiempo real.

2.3. Metodología empleada

En esta sección se verá la metodología utilizada para la realización de este proyecto. Básicamente se basa en realizar iteraciones que se componen de: diseño, implementación y experimentos.

Para la realización de un proyecto se deben establecer unas tareas a realizar entre la idea del proyecto hasta la realización del mismo. Este modelo de desarrollo establece unos requisitos de entrada para producir salidas satisfactorias.

El desarrollo de este proyecto se basará en el modelo de desarrollo en espiral basado en prototipos. La elección de este modelo de desarrollo se basa en la necesidad de separar el comportamiento final en varias subtareas más sencillas para luego fusionarlas. Cada tarea finalizada aporta los requisitos e información necesaria para abordar la siguiente iteración del modelo de desarrollo.

La gran ventaja de este modelo de desarrollo es la existencia de puntos de control al finalizar cada iteración. Además es altamente flexible en cuanto al cambio de requisitos, hecho muy común en este tipo de proyectos de investigación.

2.3.1. Desarrollo en espiral

En este tipo de desarrollos, los productos son creados gracias al número de iteraciones que se da en el proceso de vida de software.

- Determinar los objetivos. Los objetivos de un ciclo de desarrollo deben de ser identificados y especificados.
- Valorar y reducir los riesgos. Los riesgos son valorados y ciertas actividades son puestas en vigor para reducir los riesgos claves.
- Desarrollar y validar. El sistema se desarrolla y es validado usando pruebas que testean el cumplimiento de los requisitos fijados.
- Planificar. El proyecto es repasado y la próxima fase de la espiral es planificada.

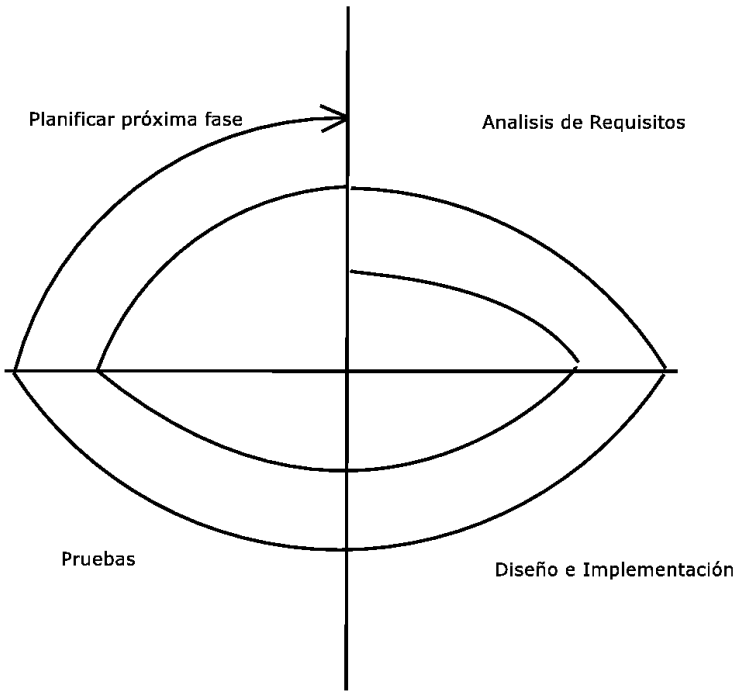


Figura 2.1: Modelo en espiral

En la figura 2.1 se puede observar los cuatro ciclos que forman este modelo de desarrollo: Análisis de requisitos, diseño e implementación, pruebas y planificación del próximo ciclo de desarrollo.

2.3.2. Prototipos del proyecto

Un prototipo es una versión preliminar de un sistema con fines de demostración o evaluación de ciertos requisitos. Se suele estimar la finalización de una iteración en el modelo de desarrollo como la obtención de un nuevo prototipo. De hecho, el modelo que hemos seleccionado se basa justamente en esto, en cada finalización de iteración se obtiene un nuevo prototipo. Cierto es que la obtención de un prototipo no implica que se utilice para el producto final, si se cree necesario se puede desechar dicho prototipo.

- **Prototipo 1:** Seguimiento visual. En este prototipo se trata de obtener un comportamiento en el cuello mecánico que sea capaz de seguir y mantener centrado al objetivo en la imagen.
- **Prototipo 2:** Navegación reactiva. En este prototipo se realizaron ciertos algoritmos para que el robot navegue por entornos sin chocarse con los obstáculos.
- **Prototipo 3:** Seguimiento visual + Navegación reactiva. El objetivo de este prototipo es integrar los dos prototipos anteriores dando como resultando un primer comportamiento sigue-persona.
- **Prototipo 4:** Seguimiento visual + Navegación reactiva + Optimizaciones. En este prototipo se realizaron ciertos ajustes y experimentos obteniendo una mejor calidad y aumento del rendimiento del comportamiento.

Capítulo 3

Plataforma de desarrollo

En el capítulo 2 se describió el objetivo de este proyecto. Antes de explicar en el capítulo 4 la solución desarrollada, detallaremos en el presente capítulo las herramientas sobre las que nos hemos apoyado para la realización de este comportamiento.

La plataforma hardware consiste básicamente en un robot real con sus correspondientes dispositivos sensoriales y de actuación. La plataforma software se compone de un conjunto de librerías y programas en los cuales se apoya el código desarrollado para este proyecto.

3.1. Plataforma Hardware: El robot Pioneer

En esencia el hardware usado en este proyecto se basa en el robot Pioneer 3, al que se le ha añadido una cámara y un cuello mecánico. Como todo robot dispone de su procesador que le da capacidad de cálculo, sus sensores para medir el estado del entorno y unos motores que le permiten moverse.

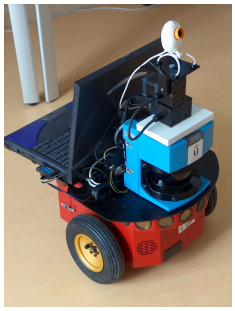
El robot que utilizamos como referencia en este proyecto es el Pioneer3x, ideal para entornos de interior. Este robot lo comercializa la empresa norteamericana ActivMedia[1] y dispone de soporte técnico activo. El robot dispone de un equipo sensorial para medir el estado de su entorno y unos motores que le permiten moverse.

Este modelo de la gama Pioneer viene equipado con un procesador de 18 MHz Hitachi H8S/2357 con 32Kb RAM y 128Kb Memoria flash. Tiene una autonomía de 5 horas y es capaz de adquirir una velocidad lineal máxima de 1,8 m/s y una velocidad angular máxima de 360º/s. Soporta un máximo de 23 Kg de carga y gracias a su pequeño tamaño (40x30cm) es ideal para desenvolverse en entornos de oficinas.

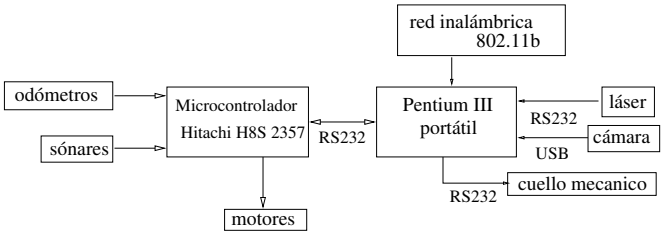
El Pioneer dispone de una corona de 16 sensores de ultrasonido que rodean al robot y lleva incorporados unos odómetros (*encoders*) asociados a las ruedas para saber cuánto han girado éstas. En nuestro proyecto no usamos los odómetros porque nuestro modo de navegación no se basa en la posición del robot en el mundo. Además le hemos incorporado un sensor láser y una cámara de visión tal y como se muestra en la figura 3.1(a). Los actuadores principales del robot son dos motores de continua, cada uno asociado a una rueda, que dotan al robot de un movimiento diferencial tipo tanque. Además dispone de una tercera rueda loca sin tracción que unido al movimiento que permiten los dos motores de continua favorece notablemente los giros del robot.

[1] http://www.activmedia.com

Tal como se aprecia en la figura 3.1(a) la plataforma final se ha configurado con un ordenador portátil al que están conectados la base motora, la cámara y el cuello mecánico. El ordenador portátil se trata de un Pentium III a 750 MHz equipado con 128MB de RAM sobre el que corre un GNU/Linux con el kernel 2.4.25. En este ordenador portátil se ejecutarán la totalidad de los programas que componen el comportamiento de seguimiento de personas. Esta configuración del robot permite reemplazar fácilmente el ordenador portátil cuando éste quede obsoleto, ya que se trata de un componente estándar en el mercado actual.



(a)



(b)

Figura 3.1: El robot Pioneer (a) y su diagrama de bloques del hardware (b)

Cabe destacar que las conexiones entre el robot y los demás dispositivos como la cámara, el cuello mecánico (*pantilt*) y láser se han realizado convenientemente siguiendo las recomendaciones de voltaje descritas en las instrucciones del fabricante. Los dispositivos como el cuello mecánico, corona de ultrasonidos y láser no venían instalados y configurados en el robot Pioneer. Por ello obtenemos la alimentación para estos dispositivos a través de las baterías del robot. Para el cuello mecánico se usa una toma directa desde la batería con un voltaje de 12V. Para el sensor láser hace falta una alimentación de 24V, que se consigue gracias a una placa elevadora de tensión desde los 12V que ofrecen la baterías del robot.

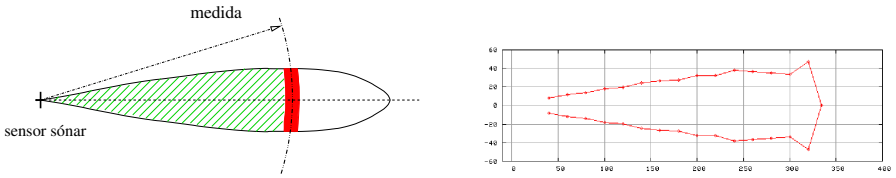
Por último, el ordenador portátil está conectado a la red exterior mediante un enlace inalámbrico, con una tarjeta de red 802.11 que le proporciona una velocidad de 11Mbps en sus comunicaciones. De esta forma el programa de control puede correr a bordo del portátil o en cualquier otro ordenador.

3.1.1. El sensor sonar

Rodeando al robot tenemos una corona de 16 sensores de ultrasonido, 8 en la parte delantera y 8 en la parte trasera. Este sensor lo podemos denominar activo (emite una onda ultrasónica en el entorno), sencillo y con un rango desde 20cm a 3metros con una resolución de centímetros. Cada segundo se obtienen 3 lecturas completas de los 16 sensores. Los sensores de ultrasonido miden distancia a los objetos cercanos al robot, y esto se traduce a *información de obstáculos en las proximidades del robot.*

El funcionamiento del sensor de ultrasonido es simple, se dispone de una membrana que mediante un impulso mecánico emite una onda ultrasónica que rebota en un posible

obstáculo que pueda existir y dicha onda regresa de nuevo a la membrana que la emitió. El tiempo que tarda desde que se lanza la onda hasta que se recibe se denomina *tiempo de vuelo*. Este tiempo de vuelo es medido por la electrónica del propio sensor sonar. Dicho tiempo permite estimar las distancias a las que se encuentran los obstáculos de nuestro robot. La energía ultrasónica se propaga con un frente de onda circular que se expande y conforma un patrón de energía de forma lobular como el que se muestra en la figura 3.2(a). En la figura 3.2(b) se muestra el modelo medido experimentalmente para los sensores sónares de nuestro Pioneer.



(a) (b)

Figura 3.2: Interpretación de la lectura sónar (a) y alcance experimental (b)

Los sensores de ultrasonido tienen problemas en ciertos entornos. Por ejemplo, un sensor de ultrasonido nunca nos va a medir obstáculos que estén a menos de 15 cm puesto que desde que se lanza el ultrasonido hasta que se recoge, no ha dado tiempo a que la membrana se estabilice y por tanto la lectura leída del tiempo de vuelo será errónea. Este caso se denomina *intervalo de blanqueo*. Este problema se puede evitar si se usan dos membranas, una para emitir el pulso mecánico y otra para recibir el eco. Para la implementación de nuestro proyecto, que se expondrá en el capítulo siguiente, hemos optados por usar las membranas en modo *pulso-eco*, es decir una misma membrana se encarga de emitir el pulso mecánico y recibir el eco.

Otro problema conocido en los sensores de ultrasonido son los reflejos especulares que se producen, los cuales dependen de la rugosidad del objeto. Las superficies rugosas son idóneas para estos sensores ya que la dirección de rebote de la onda es la misma en la que llegó, lo que supone que se dirija directamente al sonar de la que procedía. El problema con las superficies lisas es que la onda rebota pero no sale hacia la dirección del sensor. Además debido a las limitaciones de la propia onda de ultrasonido, no sabemos en que ángulo se encuentra el obstáculo, sólo sabemos la distancia tal y como se muestra en la figura 3.2. Igualmente influye la disposición angular del robot con los objetivos, ya que si existe un ángulo muy abierto entre el robot y el obstáculo, el rebote en la superficie no se producirá correctamente hacia el sensor. Cuanto más perpendicular sea la onda de ultrasonido con respecto al obstáculo, mayor fiabilidad se tendrá en la medida obtenida. En todos estos casos tendremos una cierta incertidumbre en la medida proporcionada por el sonar.

3.1.2. El sensor láser

En la parte delantera del robot se ha instalado un sensor láser de marca Sick. El sensor láser funciona de manera similar al sonar, a diferencia que el láser emite una onda

electromagnética en vez de una onda ultrasónica. Este sensor es mucho más preciso y fiable que los sensores de ultrasonido. Este sensor se suele utilizar para obtener información de los obstáculos en el entorno.

En concreto este modelo de láser que utilizamos, ofrece una visión total de 180 grados. Es capaz de barrer estos 180 grados con una precisión de 1 grado y 2cm (configurable para otras precisiones). El láser se comunica con el PC mediante una conexión serie RS-232 y se basa en un protocolo propio de bajo nivel [las, 2002]. Dependiendo de la configuración del puerto serie, el láser es capaz de realizar más o menos barridos por segundo. Las velocidades soportadas son 9600bps (2,5 barridos por segundo), 19200bps (5 barridos por segundo) y 38400bps (10 barridos por segundo).

Figura 3.3: Sensor láser

El sensor láser tiene ciertos problemas con objetos de colores muy oscuros como el negro, o con superficies como espejos o cristales. Estos problemas se deben a que no rebota correctamente la onda electromagnética, en el caso de los objetos de color oscuro, y a que dicha onda atraviesa los objetos y no rebota, en el caso de cristales totalmente transparentes.

Debido a que el láser utiliza una tecnología potente como es el electromagnetismo, se poseen resultados más fiables y precisos que con el sensor sonar, que utiliza simples impulsos mecánicos. Debido a su fiabilidad, el sensor láser es usado actualmente en aplicaciones de seguridad en coches. La gran desventaja frente al sensor sonar es su alto precio, que hace imposible que robots de gamas medias incorporen dicho sensor en su plataforma.

3.1.3. El cuello mecánico

Un cuello mecánico o *pantilt* es un dispositivo que permite dos movimientos diferentes perpendiculares entre sí. Un movimiento se denomina PAN (horizontal) y el otro se denomina TILT (vertical). A la unidad pantilt le denominamos cuello mecánico puesto que simula los movimientos de un cuello humano. Este tipo de dispositivos, debido a su gran movilidad, suelen utilizarse en los campos de robótica, visión computacional, cámara de seguridad, teleconferencias y webcams, monitorizar sistemas avanzados y seguimiento de personas.

El cuello mecánico que usamos en nuestra arquitectura se trata del modelo PTU-46-17.5 [pan, 2001] que está alimentada con un voltaje de 12V y se comunica con el ordenador mediante el interfaz serie RS-232. Además dispone de un controlador interno capaz de interpretar las consignas de control que se mandan por el puerto serie y las materializa en movimientos del cuello. Este dispositivo permite un ángulo de actuación de 318º en el eje horizontal (159º hacia cada lado) y 76º en el eje vertical (30º hacia arriba y 46º hacia abajo). Es capaz de llegar a velocidades máximas de 360º/s y con una resolución de 0.514º.

En su parte superior tiene soporte para llevar una cámara que permite la orientación en la dirección deseada. Soporta un peso máximo de 1.82Kg.

Sobre este cuello mecánico no existía ninguna librería de comunicación para interactuar con el dispositivo. Por tanto se tuvo que programar una librería[2] de comunicación con el puerto serie, siguiendo el protocolo especificado por el fabricante [pan, 2001].

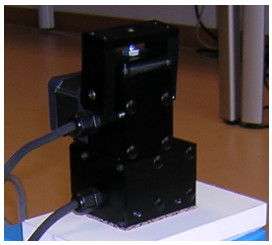


Figura 3.4: Unidad Pan-Tilt

Este cuello mecánico permite gobernarlo en diferentes modos: en posición, en velocidad y en posición parametrizando la velocidad. Como se explicará en el capítulo 4 de implementación, eligiendo uno u otro modo influirá notablemente en la obtención de un movimiento suave y continuo del cuello o por el contrario se observará un movimiento brusco y con tirones. El `control en posición` se basa en decirle exactamente la posición PAN y posición TILT donde quieres que se posicione el cuello mecánico. El `control en posición parametrizando la velocidad` consisten en darle la posición PAN y TILT donde se quiere posicionar al cuello, pero además se le comanda la velocidad a la que debe ir. Y por último el `control en velocidad` trata de comandar velocidades al cuello, de tal forma que velocidades positivas moverá el cuello hacia al derecha y velocidades negativas llevarán el cuello hacia la izquierda.

3.1.4. La cámara

Este componente es muy importante para el seguimiento de personas ya que proporciona la información básica de dónde se encuentra dicha persona. Una imagen bien analizada puede ofrecer mucha más información que un conjunto de medidas obtenidas mediante sensores como el sonar o láser. En nuestro caso hemos usado una cámara de videoconferencia Philips PCVC-740K. Es una cámara CCD, digital, y por lo tanto no requiere tarjetas digitalizadores. Se conecta a través del puerto USB y tiene soporte Linux[3]. Esta cámara puede procesar imágenes de 640x480 pixeles a un ritmo de 15fps o imágenes de 320x240 a 30fps (utilizando módulo de compresión en ambos casos). Las imágenes que nos proporciona la cámara están en formato RGB. La cámara tiene una apertura horizontal de 60 grados y una apertura vertical de 40 grados. La cámara que usamos se puede observar en la figura 3.5

Hay que tener en cuenta que la información que obtenemos mediante la cámara es muy sensible a cambios de iluminación que se producen en el entorno. No es la misma información que obtenemos en un entorno con luz natural que con luz artificial, o en el

[2]Esta librería de comunicación se puede encontrar en http://pantuflo.escet.urjc.es/ rocapal/pfc
[3]http://www.smcc.demon.nl/webcam

mismo sentido en un entorno al aire libre en un día soleado que en un día nublado. Como veremos para minimizar estos problemas en este proyecto se aplican filtros que no tienen en cuenta la intensidad del color.

Un problema añadido al análisis de la imagen es que esta cámara tiene `autoiris`. El `autoiris` es un control electrónico que posee la cámara y que proporciona ajustes automáticos en la imagen para adaptarse a los diferentes niveles de iluminación.

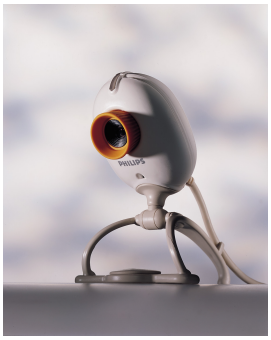


Figura 3.5: Cámara de videoconferencia

Para recoger las imágenes que nos ofrece la cámara usamos la interfaz Video4linux que explicaremos y comentaremos en la siguiente sección de software.

3.2. Plataforma Software: JDE

El software que se ha utilizado para realizar todo el proyecto se basa en la plataforma JDE (Jerarquía Dinámica de Esquemas) [Cañas, 2003]. Esta plataforma se ha desarrollado en el grupo de robótica de la URJC y está en fase de pruebas y experimentos. JDE utiliza una librería denominada ARIA que nos facilita la comunicación con los actuadores y sensores del robot. Además JDE se apoya en video4linux para el acceso a las imágenes de la cámara.

JDE es un entorno de desarrollo que facilita la creación de programas para que el robot se comporte de una determinada manera y exhiba conductas autónomas. Esta plataforma facilita el acceso a los sensores y actuadores. Este entorno de programación se puede utilizar de dos formas: Programación con los servidores JDE ó Programación con los esquemas JDE.

Gracias a JDE y su arquitectura que explicaremos en este capítulo, podemos interactuar con el robot Pioneer y los demás dispositivos como la cámara, cuello mecánico o el láser. Esta infraestructura ha sido creada por el grupo de Robótica de la URJC[4]. Este entorno de programación se encuentra a disposición de todo el mundo debido a su carácter de software libre [5].

3.2.1. Programación con los servidores JDE

Típicamente para ejecutar cualquier programa en el robot que utilizara los sensores o motores debía ejecutarse obligatoriamente en el portátil de a bordo del robot. Para superar

[4]http://gsyc.escet.urjc.es/jmplaza/tesis-65.pdf
[5]http://gsyc.escet.urjc.es/jmplaza/software.html

esta limitación nació la arquitectura de servidores y clientes de JDE, tal y como ilustra la figura 3.6.

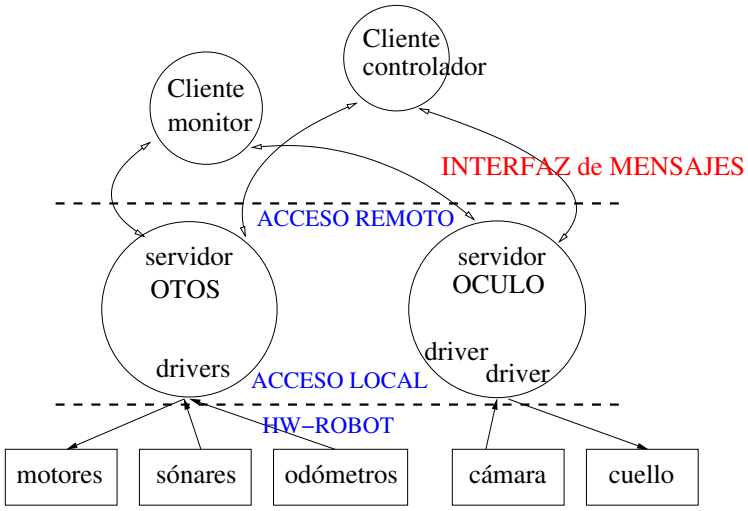


Figura 3.6: Arquitectura software con servidores y clientes de JDE

La tarea de los servidores es ofrecer acceso a los sensores y a los comandos motrices a cualquier programa cliente que los requiera. El medio de los clientes para solicitar estos servicios es mediante la red. Los clientes se suscriben a los servidores mediante mensajes de protocolo para solicitar sus servicios. Cada servidor se encarga de ciertos sensores y actuadores que existen en la máquina y ofrece su funcionalidad al resto de clientes mediante una *API de mensajes*.

Entre los servidores y los clientes se establece un diálogo, una interacción que está regulada por el protocolo desarrollado en JDE. Este protocolo JDE entre servidores y clientes fija los posibles mensajes entre ambos y las reglas o semántica que los acompañan. Está pensado para el envío continuo (*streaming*) y se han establecido tres patrones de interacción: la suscripción a sensores cortos, el envío bajo demanda de imágenes y las órdenes de movimiento.

El servidor `otos` reúne los servicios de los sensores de proximidad como los ultrasonidos y el láser. Para todos estos sensores ofrece un *servicio de suscripción directa*, que consisten en que el cliente, una vez conectado, se suscribe a un determinado sensor y el servidor le entrega datos de ese sensor cada vez que existan nuevas medidas. Los clientes pueden suscribirse exclusivamente a los sensores de su interés y desuscribirse a voluntad en cualquier momento.

La principal información que suministra `oculo` es la visual, las imágenes capturadas con la cámara. Éstas se ofrecen a los clientes a través de un servicio de *imágenes bajo demanda*: cada vez que el cliente necesita una imagen la solicita con un mensaje explícito y el servidor le responde con dos mensajes, uno de cabecera donde se especifica el tamaño de la imagen subsiguiente, y otro especial donde va la imagen en sí misma.

En el último tipo de comunicación entre servidores y clientes, el flujo se realiza desde el cliente al servidor. Los clientes mandan las *órdenes de movimiento* a los servidores, y éstos las transmiten directamente a los motores tanto de la base del robot como del cuello mecánico.

Entre un cliente y un servidor se establece una conexión `tcp` por la cual se comunican los dos. Este modo de programación bajo JDE tiene ciertas desventajas como el aumento de los retardos entre que se lee el dato sensorial y que éste llegue, a través de la red, al programa donde se realiza el procesamiento. Otro inconveniente es la existencia de cierta desincronización de las medidas.

3.2.2. Programación con los esquemas JDE

La programación sobre los servidores JDE obliga a que la propia aplicación se encargue de enviar y recibir mensajes a/de los servidores. Esto implica que el programador gaste cierto tiempo en crear los buffer de recepción y envío oportunos. Además debe tener en cuenta el tiempo de cómputo de su programa dedicado a las comunicaciones.

La programación en esquemas libera a la aplicación de esta tarea, ya que se incorporan varios *esquemas de servicio* que se encargan: de las comunicaciones, recoger de los servidores la información de los sensores del robot, y enviar a los motores las órdenes de movimiento a través de los mismos servidores. Este método de programación con los esquemas de JDE se representa en la figura 3.7.

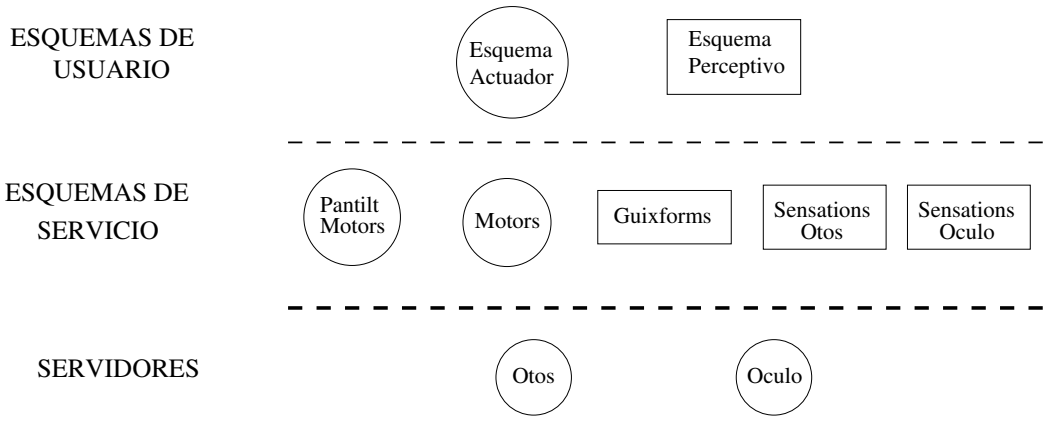


Figura 3.7: Arquitectura software del sistema programado con esquemas

La idea básica de un comportamiento se refleja en la figura 3.7. Se tienen varios esquemas de usuario, que pueden ser perceptivos o actuadores. Estos esquemas realizan su función basándose en los esquemas de servicio que aporta la arquitectura de JDE. Y en un nivel más bajo tenemos los servidores, en los que se apoyan algunos de los esquemas de servicio.

La programación con esquemas JDE define la aplicación robótica como un conjunto de esquemas que se ejecutan simultáneamente y en paralelo. Los esquemas no son más que procesos que funcionan concurrentemente y cada uno de ellos se encarga de una tarea sencilla y concreta. La ejecución concurrente de varios esquemas dan lugar a un *comportamiento*. La arquitectura de JDE en esquemas contempla un *comportamiento* como la combinación de percepción y control de la plataforma.

Los esquemas se implementan mediante hebras. Existen dos tipos de esquemas en JDE: el esquema perceptivo y el esquema motor o de actuación. El esquema perceptivo se encarga de producir y almacenar información que puede ser leída por otros esquemas. Esta

información puede provenir de medidas sensoriales o de estímulos relevantes del entorno. El esquema motor o de actuación, a partir de la información que generan los esquemas perceptivos, toma decisiones sobre los motores o activación de otros esquemas de nivel inferior.

Los esquemas se pueden organizar en niveles de tal forma que los esquemas de un nivel inferior son activados o desactivados por los esquemas del nivel superior. De esta organización podemos inferir la existencia de esquemas padre y esquemas hijo. Es probable que un padre tenga varios hijos y es el mismo padre el que se encarga de activar y desactivar a sus hijos y de modularles convenientemente. El padre en ciertas ocasiones puede hacer de árbitro entre sus hijos actuadores para decidir quien gana y toma el control.

El paso de información de unos esquemas a otros se produce mediante variables globales, lo que denominamos como un *API de variables*. Estas variables son utilizadas por varios esquemas a la vez por lo que podemos tener concurrencia. El uso de semáforos evitaría las condiciones de carrera, pero JDE opta por no usar semáforos para no producir retardos en las lecturas y apuesta por no proteger estas variables de información.

Como se ha comentado, JDE trae incorporado varios *esquemas de servicio* que funcionan como clientes de los servidores `otos` y `oculo`. A continuación pasamos a comentar estos *esquemas de servicio*.

Esquema sensationsoculo

Como ya se ha comentado, el control e información de la cámara y del cuello se realiza mediante un modelo cliente-servidor compuesto por el servidor `oculo` y los clientes `sensationsoculo` y `pantiltmotors`. Este esquema de servicio se ejecuta a una velocidad de 17 iteraciones por segundo. Todos los datos referentes a iteraciones de esquemas dependen directamente de la potencia de la máquina donde se ejecuta.

Las imágenes las recibe a una resolución de 320x240 pixeles en la estructura denominada `colorbuf`. El formato de imagen usado en este sistema es `RGB`. Aunque hay que tener en cuenta que internamente oculo manda la imagen en `BGR`, primer byte el de azul, el segundo el de verde y el último byte el de rojo. `Oculo` también permite el envío de imágenes en niveles de gris. El envío de imágenes se produce sin compresión alguna.

`Oculo` dispone de un servicio bajo demanda que permite desacoplar el ritmo de captura del ritmo al que es capaz de procesar imágenes el cliente (en nuestro caso `sensationsoculo`). Muy importante es esta opción que nos ofrece oculo ya que no tiene sentido que estemos recibiendo 25 imágenes por segundo cuando tan sólo somos capaces de procesar 10 imágenes por segundo. En este sentido el servicio bajo demanda permite que al cliente le lleguen imágenes exactamente al ritmo que es capaz de procesar.

Para el acceso a las imágenes y envió de las mismas la arquitectura no opta por el uso de semáforos. Esto implica que puedan existir inconsistencias en las imágenes que se reciben pero minimiza el tiempo de espera entre peticiones ya que no es necesario esperar para acceder a la imagen.

En cuanto al dispositivo del cuello mecánico, `sensationsoculo` nos proporciona el ángulo de desviación actual que poseen los dos ejes del cuello. Esta información se materializa en dos variables llamadas `pan-angle` y `tilt-angle`. El frente del cuello se encuentra situada en el mismo sentido que el frente del robot, por tanto sabiendo la desviación angular del cuello podemos saber donde se encuentra el objetivo con respecto al

robot. La interface del esquema de servicio `pantiltmotors` nos permite controlar el cuello mediante el envío del ángulo relativo que queremos que se mueva con respecto a la posición actual.

Esquema sensationsotos

Los datos sensoriales del robot son ofrecidos por el esquema perceptivo `sensationsotos` que funciona como cliente del servidor `otos` que es el encargado de interactuar directamente con el robot. Este esquema de servicio se ejecuta a 17 iteraciones por segundo.

La interface del cliente de otos (en este caso `sensationsotos`) nos ofrece diversas variables para leer los datos sensoriales del robot. El acceso a estos valores sensoriales del robot se realizan mediante subscripción explicita de los sensores deseados. Cada vez que existan nuevas medidas, el servidor enviará la información a los clientes subscritos. Esta información sensorial se implementa en variables. Una de estas variables es `us[16]`, se trata de un array de 16 posiciones que ofrecen las medidas de los 16 sensores de ultrasonido. Otra estructura importante es `laser[180]` en la que se guardan las 180 medidas que es capaz de procesar el láser. También dispone de información de odometría y situación del robot en la estructura `robot[5]`.

Esquema guixforms

JDE ofrece la posibilidad, gracias a este esquema de servicio, de mostrar una interface para que el usuario interactúe con ella.

Este esquema de servicio, se encarga de refrescar la interfaz gráfica periódicamente y muestrea las acciones del robot en el frontal de la aplicación. Este esquema tiene una gran utilidad en cuanto a la depuración del comportamiento generado, ya que podemos visualizar estructuras internas y activar o desactivar esquemas a voluntad para realizar trazas del comportamiento. Además podemos decidir en todo momento que variables internas visualizar, como pueden ser los sónares o el láser. También ofrece la posibilidad de joysticks para teleoperar el robot o el cuello mecánico.

Este esquema se ejecuta a una frecuencia de 5 veces por segundo, suficiente para generar un refresco aceptable al usuario. El interfaz de este esquema se ilustra en la figura 3.8.

Esquema motors y pantiltmotors

El esquema actuador `motors` se encarga de enviar al servidor `otos` la velocidad lineal y angular que se quiere comandar a la base motora en *mm/s*. Estos datos se guardan en estructuras internas, así se accede a la velocidad lineal mediante la variable `v` y a la velocidad angular mediante la variable `w`. Este esquema de servicio se ejecuta a una frecuencia máxima de 10 veces por segundo.

El esquema de servicio `pantiltmotors` se comunica con el servidor `oculo` y su función es la de enviar a dicho servidor los ángulos absolutos en los se quiere situar al cuello mecánico, tanto el eje horizontal como el vertical. Estos ángulos se materializan respectivamente en las variables `latitude` y `longitude`. Este esquema de servicio se ejecuta a una frecuencia máxima de 20 veces por segundo.

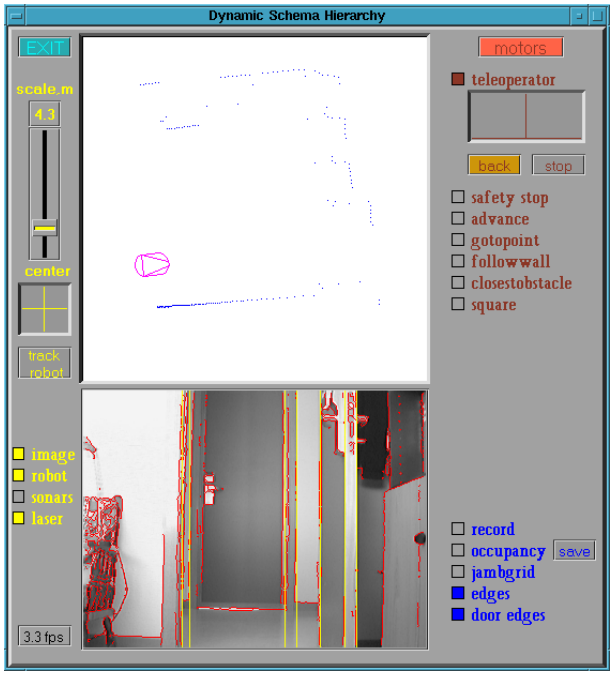


Figura 3.8: Interfaz del esquema de servicio guixforms

Estos dos esquemas de servicio se encuentran activados constantemente, desde que se inicia el programa hasta su finalización. En este transcurso de tiempo se dedican a mandar los datos especificados anteriormente a sus respectivos servidores.

Capítulo 4

Descripción Informática

En este capítulo se describe la implementación, es decir cómo se ha hecho el comportamiento basándose en los componentes software y hardware que se han descrito en el capítulo anterior. Se explicará detalladamente el funcionamiento de cada parte del comportamiento y se describirán los experimentos realizados.

4.1. Diseño Global

Como se ha comentado anteriormente, el objetivo final de este comportamiento es el seguimiento de una persona en un entorno de oficinas sorteando los posibles obstáculos.

El diseño general del comportamiento se resume en dos controles o ramas que se ejecutan concurrentemente. Una de las ramas se encarga de realizar un seguimiento visual con el cuello mecánico. La otra rama realiza un seguimiento controlando la base motora del robot de tal manera que se intenta alinear con el cuello mecánico mediante movimientos suaves y continuos.

La infraestructura general de la implementación está basada en JDE explicada en el capítulo 3. Como ya hemos comentado uno de los modos de programación con JDE se basa en la distribución de todo el comportamiento en tareas o esquemas sencillos. Este modo de programación en JDE es por el que se ha optado para desarrollar el comportamiento.

El comportamiento se materializa dentro de este marco conceptual como una colección de esquemas perceptivos y esquemas actuadores que se añaden a los esquemas de servicio que ya se han descrito en el capítulo 3. La colección de esquemas para el comportamiento de seguimiento está representada en la figura 4.1.

Se tienen dos esquemas perceptivos que se encargan respectivamente de obtener información visual mediante la cámara, obtener información del cuello mecánico (*pantilt*), obtener información mediante los sensores de ultrasonido y elaborar información sobre las *fuerzas virtuales* que influyen en el robot. Los seis esquemas actuadores toman decisiones de control sobre el cuello mecánico y sobre la base del robot. Esta decisiones de control son llevadas a cabo por los esquemas de servicio `motors` y `pantiltmotors`. Además se dispone de un esquema padre llamado `SiguePersona` que se encarga de activar/desactivar a sus hijos y coordinarlos.

Como se observa en la figura 4.1 existen dos bloques de control bien diferenciados: el bloque de seguimiento visual que se encarga de controlar los movimientos del cuello y el bloque de seguimiento de la base motora que se encarga de controlar el movimiento de

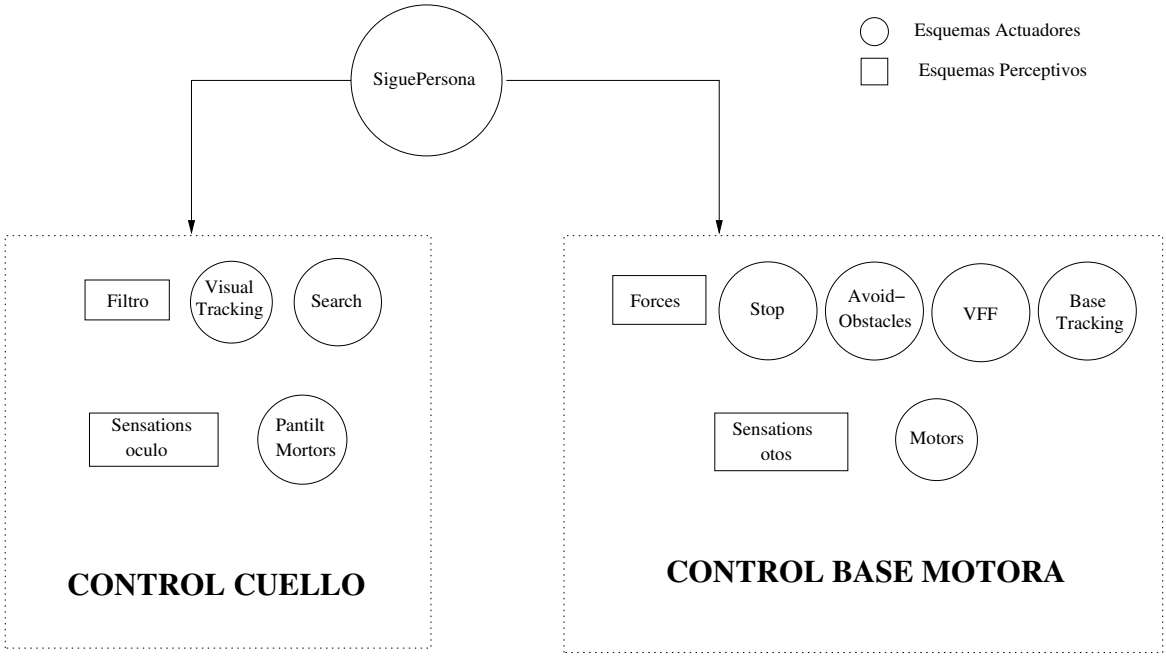


Figura 4.1: Jerarquía de esquemas del comportamiento

la plataforma móvil. Son bloques con fines distintos, pero el bloque de seguimiento de la base motora utiliza información del bloque del seguimiento visual. Por ejemplo el bloque encargado de controlar el robot utiliza la desviación angular del cuello mecánico para saber hacia dónde y con qué velocidad debe mover la base del robot. La conjunción de ambos bloques genera el comportamiento de seguimiento deseable.

Esquemas del comportamiento

Cada uno de los dos bloques de los que forman el comportamiento se compone de un conjunto de esquemas perceptivos o actuadores. Estos esquemas se apoyan en los diferentes esquemas de servicio de JDE para realizar sus funciones.

El esquema `filtro` se encarga de analizar la imagen ofrecida por `sensationsoculo` y filtrarla para obtener los datos de donde se encuentra nuestro objetivo. El esquema `visualtracking` se encarga de mantener centrado en la imagen el objetivo[1]. Si el objetivo no existe en la imagen entra en funcionamiento el esquema `search` que se encarga de buscar mediante el cuello mecánico al objetivo.

Con la información de obstáculos que almacena `sensationsotos`, el esquema `forces` genera un sistema de fuerzas sobre el robot que influirá en el movimiento del robot para que no choque con ningún obstáculo. El esquema `stoprobot` se encarga de detener al robot si el objetivo se encuentra muy cerca o si el objetivo no se tiene localizado. El esquema `avoidobstacles` se encarga de evitar obstáculos cercanos al robot. El esquema `vff` implementa un sistema de navegación basado en fuerzas virtuales. Y por último el esquema `basetracking` se encarga de realizar el seguimiento al objetivo a la velocidad máxima sin presencia de obstáculos.

[1]El objetivo se refiere a la camiseta que debe llevar la persona

Finalmente tenemos el esquema padre `siguepersona` que además de ser el responsable del comportamiento final, es el encargado de activar a todos los hijos y además arbitra el control si ninguno, dos, o más hijos quieren tomar el control del robot. Para el arbitraje es necesario que cada hijo tenga unas `precondiciones`, de tal forma que el cumplimiento de dichas precondiciones establecen que los hijos deseen o no tomar el control del robot.

4.2. Seguimiento visual

En esta sección se describen los esquemas que intervienen en las decisiones de control del cuello mecánico. La idea básica es tener centrado al objetivo en la imagen, para ello el cuello realiza movimientos sacádicos (movimientos cortos y rápidos) para conseguir vivacidad en las reacciones. Estos movimientos no llevan coordinación con la base motora del robot, es independiente en qué situación se encuentra el robot, el cuello está constantemente localizando al objetivo y centrándolo en la imagen.

El funcionamiento de esta parte del comportamiento es sencillo. Básicamente se trata de lo siguiente: primero `sensationsoculo` recibe imágenes del servidor. Después estas imágenes son procesadas y filtradas por el esquema `filtro` para verificar la existencia del objetivo en la imagen, y si es así calcular la posición de éste en la imagen. Dependiendo de la información que haya obtenido se activará el esquema `visualtracking`, si existe el objetivo, o se activará el esquema `search` si no se ha encontrado al objetivo en la imagen. En la figura 4.2 se puede ver el diagrama que refleja el funcionamiento de esta parte del comportamiento.

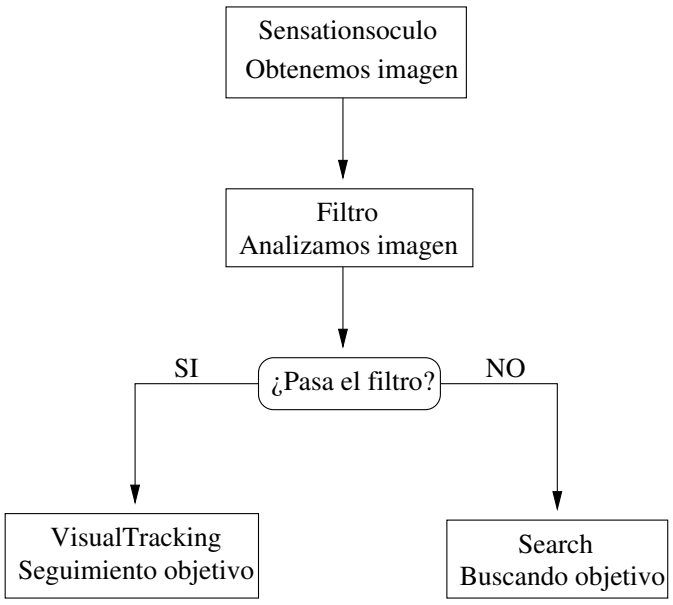


Figura 4.2: Diagrama del control de la pantilt

En el apartado de experimentos realizados de esta sección se comentará más detalladamente, pero el diagrama mostrado en la figura 4.2 no se implementó mediante hebras asíncronas por motivos de eficiencia. El objetivo es llegar al mayor número de iteraciones por segundo para conseguir movimientos sacádicos, similares a los de los ojos

humanos. La velocidad idónea de iteraciones sería 25 por segundo, justo el número de imágenes por segundo que nos puede proporcionar la cámara. Al final se consiguió una velocidad de 16 iteraciones por segundo gracias al uso de `callbacks`, que se explicará más a fondo en la sección de experimentos.

4.2.1. Esquema filtro de color

La función de este esquema es verificar la existencia del objetivo en la imagen y su posición dentro de la misma. Se trata de un esquema perceptivo que tiene como entrada las imágenes ofrecidas por `sensationsoculo`. La información que genera este esquema se puede observar a continuación y es utilizada por varios esquemas, entre ellos el esquema `VisualTracking` para decidir hacia dónde orienta el cuello mecánico y el esquema `stoprobot` para decidir si debe detener la base motora.

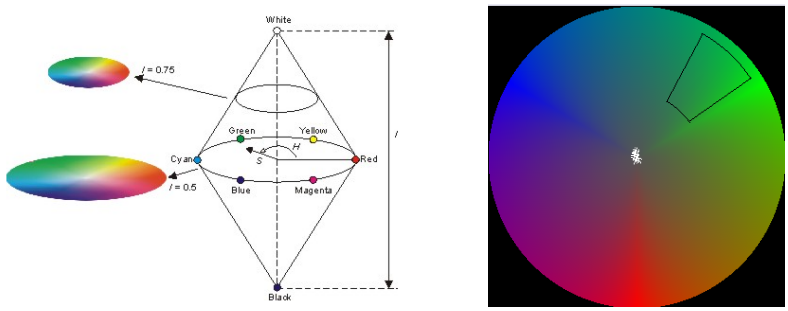
```
struct dfiltro {
  int x;              /* Coordenada X del centro de masas */
  int y;              /* Coordenada Y del centro de masas */
  int pixeles;        /* Numero de pixeles que pasan el filtro */
  int cuadrante;      /* Cuadrante en el que se encuentra el centro de masas */
  double distancia;   /* Distancia del centro de la imagen al centro de masas */
  int lineas;         /* Numero de líneas horizontales en las que al menos
                         un pixel pasa el filtro */
};
```

Este esquema es una de las piezas claves en el resultado del comportamiento final, ya que si el filtro no funciona correctamente las acciones que se toman serán erróneas. La calidad del comportamiento final depende en gran medida de la calidad de las percepciones que se consiga en este esquema del filtro de color.

Espacio HSI

Como ya se ha comentado, la imagen que nos ofrece `oculo` están en formato RGB. Este formato es muy sensible a los cambios de iluminación del entorno. Esto es un problema para el seguimiento ya que simplemente por el hecho de pasar muy cerca de una fuente de luz como una lámpara puede hacer fallar al filtro y que no encuentre al objetivo. Las pruebas que se realizaron confirmaron que utilizar el espacio RGB para realizar un filtro no es una buena solución puesto que es bastante sensible en los cambios de iluminación y contraste.

Por ello se optó por utilizar un filtro en HSI. El modelo de color HSI fue diseñado teniendo en mente el modo en que artistas y diseñadores gráficos usan los términos de saturación (pureza del color), tono (el color en si mismo) e intensidad (brillo del color). Este modelo se representa geométricamente en un cono como se muestra en la figura 4.3(a). En el modelo HSI, las componentes HS se mantienen invariables frente a los cambios de iluminación que se producen en el entorno. Estos cambios de iluminación quedan absorbidos por la componente I. La gran desventaja de este espacio de color es que la mayoría de las tarjetas digitalizadoras actuales no entregan las imágenes en este formato y esto conlleva cierta carga computacional para convertir las imágenes de la cámara a este espacio de color.



(a) (b)

Figura 4.3: Modelo cónico (a) y disco de color (b) del espacio de color HSI.

En este caso, la imagen nos viene en formato RGB directamente desde la cámara usb por lo tanto estamos obligados a realizar el cambio de espacio de colores al HSI. Además una razón añadida para realizar el cambio de espacio es que la cámara Philips posee `autoris`, que ya comentamos en el capítulo 3 los problemas que esto conllevaba. Este cambio lo realizamos teniendo en cuenta las formulas siguientes:

$$H = cos^{-1}\frac{\frac{1}{2}((R-G)+(R-B))}{\sqrt{((R-G)^2+(R-B)*(G-B))}}$$

$$S = 1 - \frac{3}{(R+G+B)}min(R,G,B)$$

$$I = \frac{1}{3}(R+G+B)$$

La intensidad (I) y saturación (S) están normalizadas (entre cero y uno) y el tono (H) está entre 0 y 360 grados. La función que aparece en el calculo de la S, *min(R,G,B)* selecciona el menor valor de entre los tres.

Hay que tener en cuenta que en ciertas ocasiones, dependiendo de los valores R,G y B, la fórmula no se aplica tal cual. Hay casos que debemos contemplar como por ejemplo cuando los tres valores son iguales a cero. Para ellos es conveniente ver el código empleado para realiza el cambio en espacios de color. El siguiente código representa dicho cambio en un pixel:

```
if (((R-G)*(R-G)+(R-B)*(G-B))<=0)
  H = -1;
else
  H = acos ((0.5*((R-G)+(R-B))) / sqrt((R-G)*(R-G)+(R-B)*(G-B)));

if ((R+G+B) == 0.0)
  S=1.0;
else{
  S = 1.0 - (3.0/(R+G+B)) * min(R,G,B);
```

```
  I = (1/3)*(R+G+B);
}
H = H*RADTODEG;
```

Una vez que ya se ha descrito el proceso para hacer la conversión a un espacio de color que nos favorece, pasemos a tratar el análisis de la imagen en HSI.

Análisis de la imagen

El análisis de la imagen se realiza de izquierda a derecha y de arriba a abajo. Para cada pixel analizado miramos si está entre los rangos de H y S que tenemos ya estipulados a priori. Si esto se satisface ese pixel pasa satisfactoriamente el filtro y tenemos que tenerlo en cuenta para nuestros cálculos. Lo que hacemos es ir sumando las coordenadas de los pixeles que pasan el filtro y una vez terminado el análisis de la imagen hallamos la media de todos los pixeles que han pasado el filtro, de esta forma obtenemos la coordenada del centro de masas. Además llevamos la cuenta del número de pixeles que pasan el filtro, como el número de líneas horizontales en las que al menos un pixel pasa el filtro. Después del análisis de la imagen calculamos dos datos importantes que se pueden observar en la figura 4.4, el cuadrante donde se encuentra el centro de masas y la distancia del centro de la imagen al centro de masas.

Es importante comentar que el ritmo de captura de `sensationsoculo` es de 18 iteraciones por segundo. Si además añadimos el filtro de color, el número de iteraciones baja hasta las 14. Con este número de iteraciones es suficiente para realizar un análisis bastante rápido, por ello prescindimos del uso de una ventana de atención en el filtro [Ortiz, 2004].

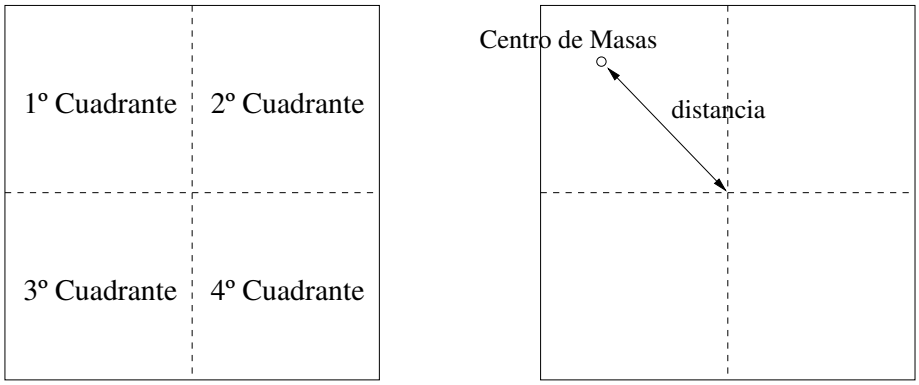


Figura 4.4: Disposición de los cuadrantes (a) y distancia al centro de masas.(b)

Recordamos que uno de los objetivos de este proyecto es ponérselo fácil al aspecto perceptivo, por ello el color que se ha utilizado para el seguimiento del objetivo ha sido el verde, ya que es suficientemente discriminante en el escenario de pruebas. Concretamente el rango de color que se ha utilizado se puede observar en la figura 4.3(b). Los rangos que se han utilizado para discriminar los pixeles pertenecientes al objetivo se han hallado empíricamente. Estos rangos son los siguientes:

```
#define H_MAX 181.98  /* Máxima componente de contraste  */
#define H_MIN 163.0   /* Mínima componente de contraste  */
#define S_MAX 0.99    /* Máxima componente de saturación */
#define S_MIN 0.40    /* Mínima componente de saturación */
```

Gracias al modo de programación que se ha optado para el filtro de color, podemos realizar fácilmente un cambio en el color del filtro [Matute, 2002]. Esto es posible porque se han usado constantes para facilitar la parametrización de los rangos de color.

El resultado de todo lo que se ha expuesto hasta el momento sobre el filtro lo podemos observar en la figura 4.5. Se puede observar como en la imagen resultante los pixeles que pasan el filtro aparecen en verde.

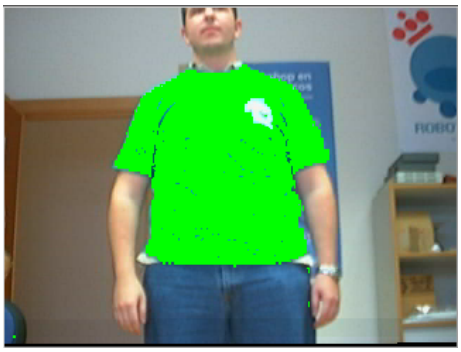


Figura 4.5: Visualización del filtro bajo JDE

Un problema que existe en el modo que se ha realizado el filtro y el análisis de la imagen, es que cuando aparecen dos objetos verdes en la imagen, el filtro no es capaz de diferenciarlos. Es más, el filtro no tiene constancia de que existan dos objetos diferentes, porque como ya se ha comentado, el filtro calcula un centro de masas sobre toda la nube de pixeles que pasan el filtro. Esto provoca que más de un objeto de color verde en una escena pueda llevar a decisiones erróneas sobre el comportamiento.

4.2.2. Esquema visualtracking

La función de este esquema es tratar de tener centrado el objetivo en la imagen. Para ello se realiza un control realimentado proporcional a la situación del objetivo, es decir, si el objetivo se encuentra a la izquierda de la imagen, se moverá el cuello hacia la derecha. Este esquema motor o de actuación toma su entrada de la estructura que genera el esquema perceptivo `filtro` y su salida son las decisiones de control que generan movimiento en el cuello mecánico.

Este esquema sólo entra en funcionamiento cuando el número de pixeles que han pasado el filtro es mayor a un número determinado denominado `MIN-PIXELES`. Esto evita que el cuello se mueva si se han encontrado pocos pixeles que pasan el filtro en toda la imagen. Si ésto pasa es porque se produce ruido en la imagen y no debemos tratar ese ruido como parte del objetivo. Empíricamente hemos estimado que `MIN-PIXELES` debe tener un valor cercano a 400. Hay que tener en cuenta que el número de pixeles máximo de la imagen es 320*240=76800.

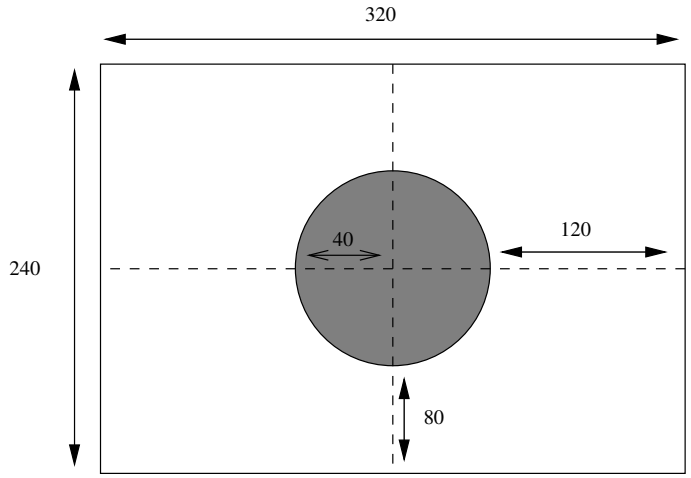


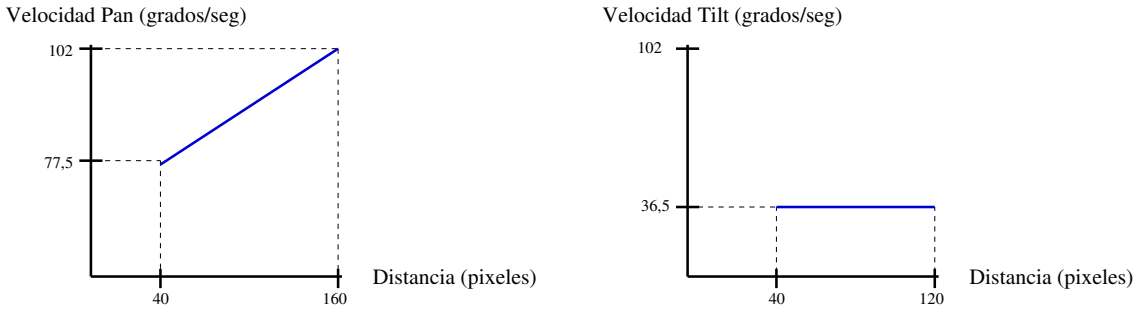
Figura 4.6: Banda muerta aplicada al control de la pantilt

Para evitar que el cuello se mueva constantemente haciendo movimientos innecesarios y excesivamente cortos, se crea una *banda muerta*. Esta banda se sitúa en la zona central de la imagen creando un perímetro tal y como se muestra en la figura 4.6. La banda muerta está acotada en una circunferencia de radio 40 pixeles (conviene recordar que la imagen es de tamaño 320x240). El funcionamiento de la banda muerta es sencilla: si el centro de masas del objetivo está dentro de dicha banda no se mueve el cuello; si por el contrario el centro de masas se encuentra fuera de la banda muerta, movemos el cuello hasta centrar dicho centro de masas. Con esta decisión de control se logra que el cuello no oscile a pequeños cambios y se encuentre estable, siempre y cuando tenga centrado el objetivo.

El cuello mecánico ofrece tres modos incorporados en el protocolo para gobernar su movimiento. El primero se trata del *control en posición*, este modo se basa en darle al cuello la posición a la que se quiere ir. En este modo cuando el cuello llega a la posición comandada se para a la espera del siguiente comando de posición. Debido a esta espera, este modo de control produce movimientos bruscos y no continuos. El segundo modo de control se trata del *control en velocidad*. Este modo se basa en comandar velocidades al cuello, de tal forma que velocidad positiva gira hacia la derecha y velocidad negativa gira hacia la izquierda. Con este modo, el cuello nunca se llega a pararse porque siempre se le está comandando velocidades. Así se consigue un movimiento más suave y continuo porque dependiendo de lo que se tenga que mover el cuello, se le comanda una velocidad mayor o menor.

El último modo de controlar el cuello se trata del *control en posición modulando la velocidad*. Este modo se basa en comandar la posición al cuello y además se le comanda la velocidad del eje a mover (pan o tilt). Así si el movimiento del cuello es grande le comandaremos una velocidad alta y si por el contrario el movimiento es pequeño, le comandaremos una velocidad pequeña. De este modo conseguimos que el movimiento sea muy suave y continuo ya que para movimientos grandes el cuello irá rápido y para movimientos pequeños el cuello irá más lento. Para controlar el cuello mecánico en este proyecto se ha optado por un comportamiento de control basado en la posición modulando la velocidad.

Los perfiles de realimentación del comportamiento del eje pan y del eje tilt se pueden



(a) (b)

Figura 4.7: Perfil de realimentación del movimiento pan (a) y del tilt (b)

observar en la figura 4.7. La realimentación proporcional que se utiliza en el eje pan produce una corrección más rápida en errores mayores. Para el eje tilt se optó por una velocidad constante ya que en este eje no se sufren grandes desplazamientos del objetivo.

Para cada centro de masas calculado por el esquema `filtro` se tendrán que generar cuatro valores que se comandarán al cuello. Estos cuatro valores son `POS-PAN` (posición en el eje horizontal), `VEL-PAN` (velocidad del eje horizontal), `POS-TILT` (posición en el eje vertical) y `VEL-TILT` (velocidad del eje vertical).

En la realización de este esquema se invirtió tiempo en retocar y mejorar el servicio de control de cuello que ofrece JDE. Esta plataforma software ofrecía únicamente el envío de ángulos al servidor `oculo` y éste comandaba los parámetros nativos al cuello en posición. Para realizar la implementación descrita anteriormente era necesario tener acceso desde JDE al envío de comandos nativos de posición al cuello, por ello se le añadió este soporte a la plataforma.

4.2.3. Esquema search

Este esquema actuador se encarga de buscar y localizar al objetivo cuando éste no se encuentra en la imagen. Para realizar la búsqueda se mueve el cuello de una lado hacia otro hasta que el objetivo es localizado. Esta búsqueda provoca que el objetivo vuelva a ser localizado.

La entrada de este esquema es la información que produce el esquema `filtro`, más concretamente tiene en cuenta el número de pixeles y el número de líneas. La salida produce un movimiento de búsqueda en el cuello mecánico con el objetivo de encontrar a la persona. La precondición de activación de este esquema viene dada por el número de pixeles que pasan el filtro.

La búsqueda se realiza solamente en el eje horizontal porque el objetivo tiende a perderse por los lados debido a que va más rápido que el cuello, éste caso suele darse a la hora de realizar giros en las esquinas. Después de pruebas y experimentos se decidió no buscar en el eje vertical (tilt) puesto que no es común que el objetivo desapareciera por arriba o por debajo de la imagen. La búsqueda se realiza a una velocidad lenta para ayudar al procesamiento de la imagen.

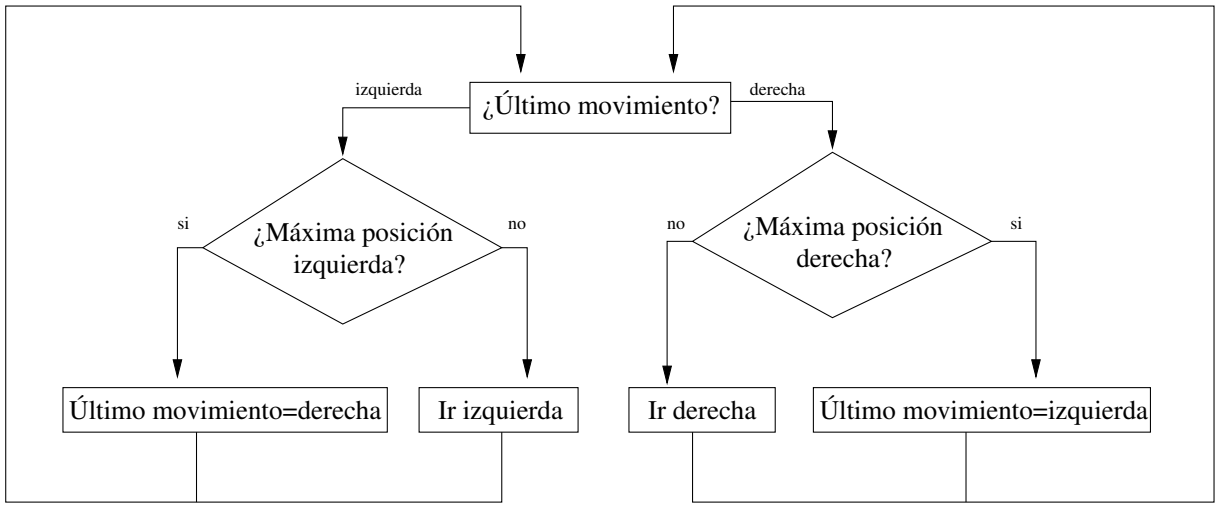


Figura 4.8: Diagrama de búsqueda de la pantilt

Gracias a que el esquema `visualtracking` guarda el estado de los movimientos del cuello se empieza a buscar por el lado hacia el que el cuello realizó su último movimiento. Es decir si el cuello ha realizado su último movimiento a la derecha y en el instante después ha perdido al objetivo porque éste iba rápido, lo más normal es que el objetivo se encuentre a la derecha de la posición actual del cuello. Mientras el esquema `search` está activado sigue el algoritmo de búsqueda que se muestra en la figura 4.8 para materializar el barrido hacia la derecha e izquierda.

4.2.4. Experimentos

En esta sección se van a comentar los experimentos y pruebas que se han ido realizando y que han llevado a un ajuste fino del sistema de seguimiento visual desarrollado. Vamos a comentar los experimentos mas relevantes.

Hebras paralelas vs callbacks

Quizás el punto más importante de esta sección es la discusión entre la utilización de hebras asíncronas o hebras síncronas. La utilización de hebras asíncronas consiste en ejecutar concurrentemente y asíncronamente cada uno de los esquemas que se han explicado anteriormente. El uso de hebras síncronas, gracias a los callbacks, es una posibilidad que proporciona JDE y que se descubrió en el desarrollo de este proyecto. Se trata de lanzar una única hebra y que ésta llame a funciones de otros esquemas que no están ejecutándose concurrentemente con él.

El problema de las hebras asíncronas es que si por ejemplo tenemos 3 hebras que se tienen que intercambiar datos, en media se produce un retardo de $\frac{t}{2}$ siendo t el periodo de ejecución cada hebras. Por lo tanto para la ejecución de tres hebras asíncronamente tendríamos un retardo total de $3 * \frac{t}{2}$. Estos retardos provocan oscilaciones de control en el comportamiento cuando se tienen actuadores rápidos, como el cuello, e información muy vivaz como la de la cámara.

Inicialmente los esquemas `filtro`, `visualtracking` y `search` se ejecutaban concurrentemente unos con otros. Esta forma de ejecución llevaba al sistema a una inestabilidad en el control del cuello mecánico, debido a la latencia y retardos.

Por ello optamos por usar las `callbacks`. Este método consiste en lanzar una única hebra, la del esquema `filtro` y desde este esquema y dependiendo de los resultados del filtro, se llama a la función de ejecución del esquema `visualtracking` o del esquema `search`. Pero estos dos últimos esquemas no se están ejecutando como hebras, ahora funcionan como meras funciones de una librería.

Este cambio en la estructura de la ejecución hizo que se ganara rapidez y vivacidad en el comportamiento, ya que eliminamos las esperas y latencia entre hebras.

Control del cuello

En el control del cuello mecánico es donde más tiempo se ha invertido para su correcta calibración ya que se quería conseguir un movimiento suave y que no produjera inestabilidad ni saltos bruscos.

En un primer momento el control del cuello se hacía a velocidad constante, simplemente comandándole la posición donde debía ir. Este modo de control del cuello era muy lento, porque al ir a la misma velocidad siempre era indiferente si se tenía que desplazar mucho o poco sobre el eje. También se implementó el control en velocidad para el cuello, pero no aportaba mejoras significativas.

Por ello se optó por hacer un control en posición modulando la velocidad. Es decir, dependiendo de la distancia que tuviera que moverse el cuello, comandaríamos una velocidad mayor o menor. Con esto conseguimos que desviaciones grandes se corrijan en menos tiempo al ir a una mayor velocidad.

Otra decisión que se tomó fundada en los experimentos es que en el eje vertical (tilt) tener velocidad proporcional al error de desviación no era necesario, ya que en este eje no se producen errores grandes. El objetivo suele moverse más por el eje horizontal, por eso en el eje tilt la velocidad es constante en el control de movimiento.

Búsqueda del objetivo

Por último, la búsqueda del objetivo se planteaba como un aporte de inteligencia al comportamiento. No era algo que se tuviera en mente desde el principio, pero debido a que el sistema de seguimiento puede perder al objetivo se decidió implementar un sistema de búsqueda.

Al principio el sistema de búsqueda visual se basaba en mover el cuello mecánico en los dos ejes para buscar al objetivo. Además la velocidad de los ejes era elevada para encontrar al objetivo lo más deprisa una vez que se hubiera perdido.

En base a las pruebas realizadas y sus resultados se estableció que la búsqueda en el eje vertical (tilt) era innecesaria, ya que el objetivo en la mayoría de los casos se pierde por la derecha o por la izquierda. Por ello simplemente con mover el eje horizontal (pan) es suficiente para volver a localizar al objetivo. Además la velocidad de búsqueda del eje horizontal se bajó considerablemente, ya que a altas velocidades no da tiempo a analizar y procesar la imagen.

4.3. Seguimiento con la base motora

En esta sección se describen los esquemas que se encargan de actuar directamente sobre los motores de la plataforma física. La idea básica es que el robot avance hacia el objetivo sin chocarse con los obstáculos de su entorno, para ello utiliza la información de los sónares. Dependiendo de la cercanía de los obstáculos se tomarán decisiones conservadoras o arriesgadas. La información de dónde está el objetivo la proporciona el bloque de seguimiento visual. Con dicha información y con los datos sensoriales del entorno, el robot debe ser capaz de llegar hasta el objetivo sin chocar con ningún obstáculo.

El funcionamiento de esta parte del comportamiento es más complejo que el control del cuello. Se han identificado diferentes casos relevantes en los que se tomarán decisiones distintas, dependiendo siempre de la cercanía de los obstáculos. Estos casos se abordan separadamente en los diferentes esquemas que componen esta sección.

En esta sección se sitúa el esquema perceptivo llamado `forces` (se apoya en el esquema de servicio `sensationsotos`) que es el cerebro de todo el seguimiento de la base motora. Él es el encargado de construir las fuerzas atractoras y repulsivas del robot sobre las que se basan las decisiones de movimiento. Si no existen obstáculos cercanos al robot se activa el comportamiento `basetracking` que se encarga de seguir al objetivo sin tener en cuenta los obstáculos. Si se detectan obstáculos a media distancia entra en funcionamiento el esquema `vff` que implementa un algoritmo de navegación basado en fuerzas virtuales. Este algoritmo combina la atracción hacia el objetivo con la repulsión ante los obstáculos. Cuando el robot se encuentra muy cerca de un obstáculo se activa el esquema `avoidobstacles` encargado de sortear un obstáculo cercano. Y por último si no tenemos el objetivo visual localizado o está muy cercano al robot se activa el esquema `stoprobot`. El diseño de esta parte del comportamiento se puede observar en la figura 4.9.

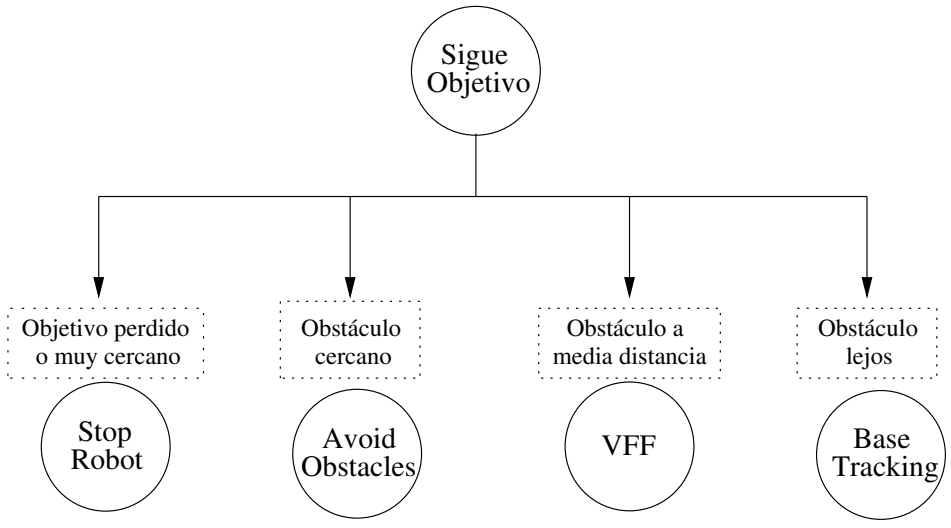


Figura 4.9: Diagrama de la navegación del robot

4.3.1. Esquema `forces`

La navegación del robot se basa en fuerzas y dirección objetivo. Para ello se tiene este esquema perceptivo denominado `forces`. Este esquema recoge y resume la información de ocupación existente en el entorno cercano al robot y la posición del objeto a seguir. Para generar esta información el esquema se basa en los datos sensoriales del robot y en la información sobre la posición actual del cuello. Al tratarse de un esquema perceptivo se está ejecutando continuamente.

Las fuerzas repulsivas que afectan al robot se generan mediante las lecturas de los sensores de ultrasonido. Únicamente utilizamos la corona de sensores de ultrasonido delantera. La información que podamos obtener por detrás del robot es irrelevante porque nuestro robot siempre trata de ir hacia delante, nunca colisionará con obstáculos que estén detrás del robot. En los experimentos de esta sección se explicará en detalle las causas que llevaron a no usar los sensores de ultrasonido traseros. En la figura 4.10 se puede observar como de los sensores de ultrasonido (líneas verdes) se obtiene una fuerza repulsiva (línea roja) que resume la tendencia a alejarse de los obstáculos.

En la sección de `vff` se explica más detalladamente el funcionamiento de la navegación basada en fuerzas virtuales.

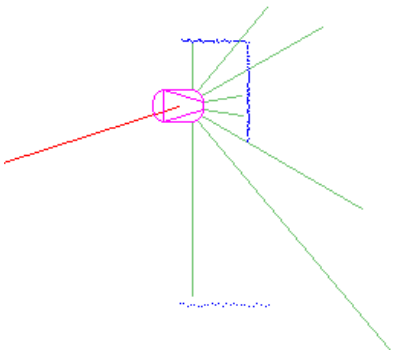
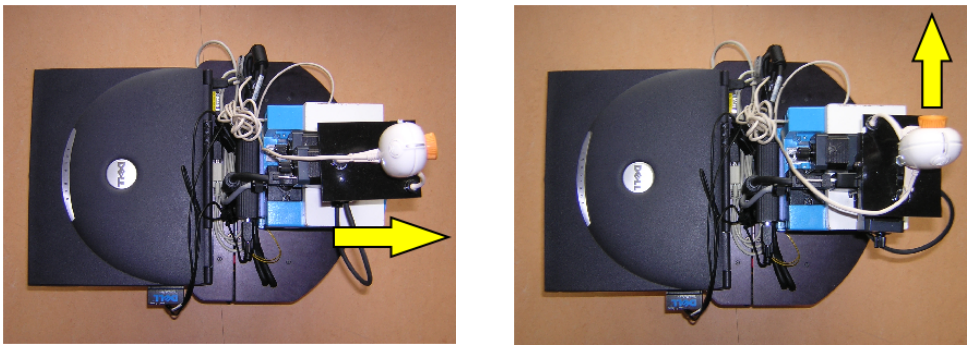


Figura 4.10: Fuerza repulsiva sobre el robot, generada con los sónares.

Además este esquema se encarga de generar la fuerza objetivo, es decir, resume la tendencia a acercarse al objetivo a seguir. Denominamos fuerza objetivo a la fuerza atractora que lleva al robot hacia la persona. Esta fuerza se calcula teniendo en cuenta la posición actual del cuello, esta información sensorial nos la proporciona `sensationsoculo`. Mirando la desviación angular del cuello respecto del frente del robot podemos inferir la situación del objetivo con respecto al robot. Así, por ejemplo, si el cuello en un momento dado está a 60 grados hacia la derecha, se puede concluir que el objetivo está a 60 grados hacia la derecha de la base motora del robot. Esto es posible porque el cuello se encuentra situado en el mismo eje y en la misma orientación que el robot, tal como muestra la figura 4.11



(a) (b)

Figura 4.11: Cuello alineado con el robot (a). Cuello con una desviación de 90º con respecto al robot

4.3.2. Esquema `stoprobot`

La función de este esquema es detener la base móvil del robot cuando el objetivo se encuentra cercano o cuando el objetivo se ha perdido. El esquema `stoprobot` se basa en la información que genera el esquema `filtro`. Básicamente utiliza dos características del filtro que son: el número de pixeles que ha pasado el filtro, y el número de líneas horizontales en las que al menos un pixel ha pasado el filtro. Estas dos características del filtro se implementan en las constantes `MIN-PIXELES` y `MAX-LINEAS`.

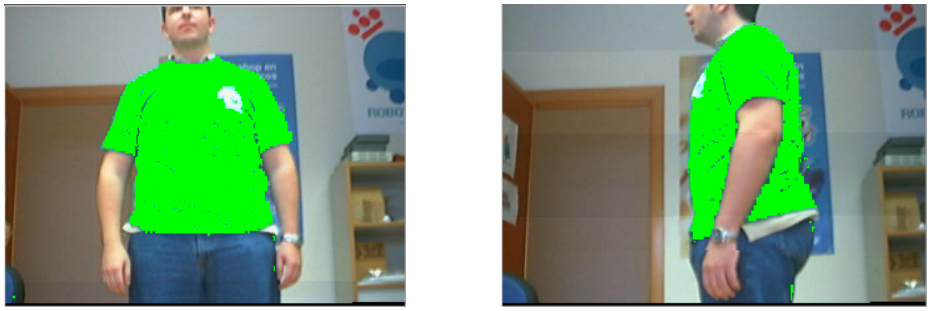
Este esquema tiene una `precondición` basada en los datos del filtro. La activación de este esquema se producirá si se cumple una de las dos precondiciones siguientes:

- Si el número de pixeles que han pasado el filtro es menor a 400. Esto refleja que se ha perdido al objetivo.
- Si el número de líneas horizontales, en las que al menos un pixel ha pasado al filtro, es mayor de 155. Esto refleja que el objetivo está demasiado cerca.

Si se cumple alguna de las anteriores precondiciones, entonces el esquema `stoprobot` se activará y detendrá el robot.

Para determinar si el objetivo se encuentra cerca o lejos del robot se ha diseñado una técnica simple pero muy eficaz. Se trata de llevar la cuenta del número de líneas horizontales en las que algún pixel ha pasado el filtro correctamente. De esta forma aunque la persona se encuentre de frente o de perfil, el número de líneas en las que al menos un pixel ha pasado el filtro será similar tal y como se muestra en la figura 4.12. Esta técnica es más óptima que utilizar un umbral de pixeles para detectar la cercanía del objetivo.

En las dos imágenes que se observan en la figura 4.12 se aprecia claramente que en la figura figura 4.12(a) hay un número mayor de pixeles que en la figura 4.12(b). Aun así, el control de detección de cercanía del objetivo las trata de igual modo, porque en las dos figuras existe aproximadamente el mismo número de lineas horizontales en las que al menos un pixel ha pasado el filtro, que en este caso el número de líneas es 185.



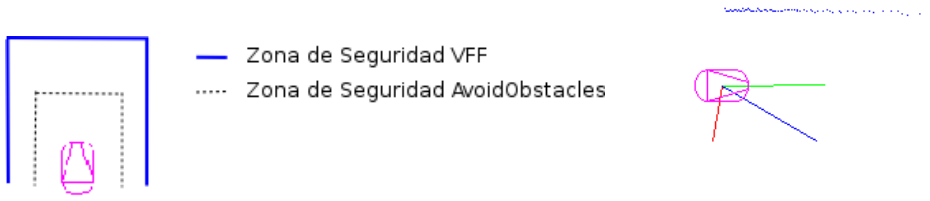
(a) (b)

Figura 4.12: Filtro frontal del objetivo (a) y de perfil (b)

4.3.3. Esquema `vff`

Este esquema se encarga de dirigir al robot hacia el objetivo por un entorno con obstáculos a media distancia. Para ello utiliza el método de navegación descrito anteriormente. La idea básica de este esquema es dirigir al robot hacia una trayectoria que intente alejarse de los obstáculos y acercarse al objetivo.

`VFF` son las iniciales de *Virtual Forces Field* [Borenstein y Koren, 1989] (Campo de Fuerzas Virtuales). Esta técnica de navegación se basa en construir una serie de fuerzas que intervienen en el movimiento del robot, y que afectan a éste de un modo directo. Hay dos tipos de fuerzas, las fuerzas repulsivas que las generan los obstáculos cercanos al robot y las fuerzas atractoras que son generadas por el objetivo al que hay que seguir. Las dos resultantes de cada tipo de fuerza se combinan mediante una suma vectorial para obtener la fuerza final.



(a) (b)

Figura 4.13: Región de seguridad (a). Fuerzas del sistema VFF (b)

En este método de navegación se trata a los obstáculos como campos de fuerzas repulsivas sobre el robot. De tal forma que si un obstáculo se encuentra enfrente del robot se generará una fuerza repulsiva en sentido contrario, es decir hacia detrás del robot. Del mismo modo si un obstáculo se encuentra cerca del robot, la fuerza repulsiva tendrá un módulo mayor que si el obstáculo se encontrara más lejos. Por tanto la fuerza repulsiva que se ejerce sobre el robot es inversamente proporcional a la distancia existente al obstáculo.

Este esquema tiene como entrada la información generada por el esquema `forces`, para generar las fuerzas resultantes y las lecturas de los sensores de ultrasonido para verificar sus precondiciones. A partir de la diferencia angular (α) entre la fuerza resultante final y la fuerza objetivo se generan las velocidades lineales (`v`) y angulares (`w`) correspondientes que se comandan a los motores. La precondición de este esquema es la existencia de algún obstáculo en su región de seguridad que se muestra en la figura 4.13(a)

La funcionalidad que se busca con este comportamiento es que se aleje de los obstáculos de una manera suave, continua y sin necesidad de parar la base motora. Las velocidades que se comandan a dicha base motora dependen de la desviación del ángulo entre la dirección objetivo y la dirección resultante que se ha obtenido mediante el `vff`. Es decir, observando la figura 4.13(b) se trata del ángulo que forma la fuerza resultante (línea azul) con la fuerza objetivo (línea verde).

Las velocidades angulares y lineales se modulan dependiendo del ángulo de desviación comentado anteriormente. Los perfiles de velocidades se pueden observar en la figura 4.14

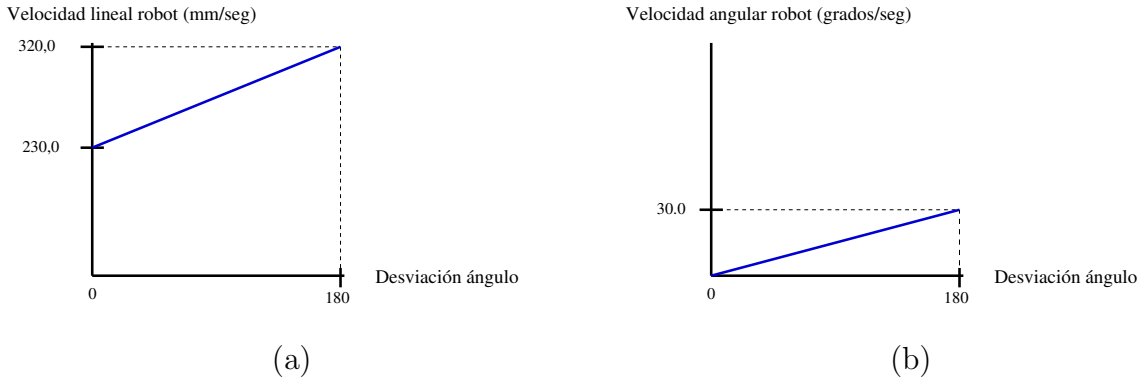


Figura 4.14: Perfil de velocidad lineal (a) y velocidad angular (b) del esquema VFF

4.3.4. Esquema `avoidobstacles`

Cuando el robot se encuentra cercano a un obstáculo entra en funcionamiento este esquema. Este esquema se encarga de evitar el choque contra obstáculos muy cercanos al robot, para ello trata de hacer girar al robot sobre si mismo hasta encontrar una salida.

Las entradas de este esquema son los datos generados por el esquema `forces` y las lecturas de los sensores de ultrasonido para evaluar sus precondiciones. Con esta información produce una salida que se transmite a los motores de la base móvil. Como este esquema trata de girar al robot sobre si mismo buscando evitar el obstáculo cercano sólo comanda velocidades angulares, a diferencia del esquema `vff` que comanda velocidades lineales y angulares.

Para detectar que obstáculos están muy cerca en su entorno, tiene una región de seguridad que se puede observar en la figura 4.15. Esta región de seguridad está definida empíricamente y está implementada mediante la información proporcionada por los sensores de ultrasonido y del láser. La precondición para que este esquema se active y entre

en funcionamiento es la detección de algún obstáculo dentro de la región de seguridad de `avoidobstacles`.

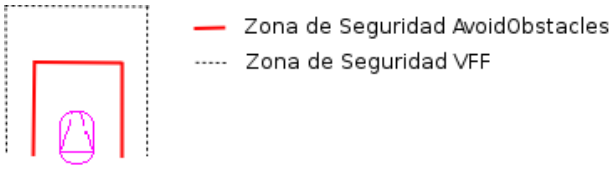
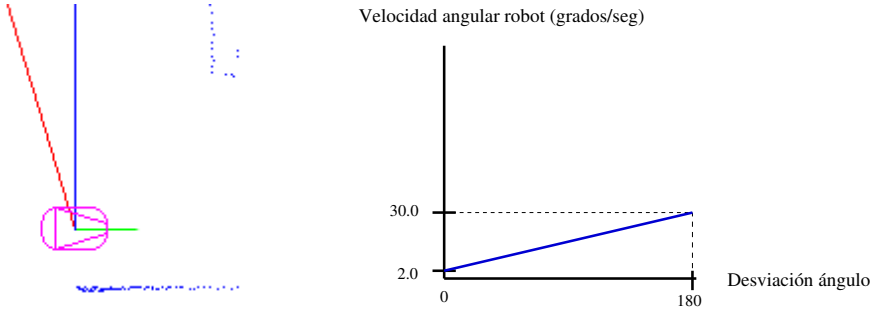


Figura 4.15: Región de seguridad para avoidobstacles

Habiendo visto ya las precondiciones de los esquemas `avoidobstacles` y `vff` y observando la figura 4.15 podemos ver que cuando un obstáculo se encuentra dentro de la región de seguridad de `avoidobstacles`, las precondiciones de los dos esquemas se cumplen. Para solucionar quién de los dos debe tomar el control de la base motora se utiliza un arbitraje distribuido basado en prioridades que se verá con más detalle al final de este capítulo.

El esquema `avoidobstacles` se basa en la navegación `VFF`. Pero en este esquema sólo se modula la velocidad angular, la velocidad lineal se establece a cero. El funcionamiento se basa en girar el robot hacia el objetivo hasta que en la región de seguridad no exista ningún obstáculo. En ese momento las precondiciones de este esquema no se cumplirán y por tanto no se activará su funcionamiento. En la figura 4.16(a) se puede observar el conjunto de fuerzas que se calculan. La línea verde se trata de la *fuerza atractora* hacia el objetivo. La línea roja representa la *fuerza repulsiva* que generan los sensores de ultrasonido delanteros. La *fuerza objetivo* está representada por la línea azul, que es la resultante de las otras dos fuerzas y hacia donde debe dirigirse el robot. Las líneas punteadas de color azul representan los obstáculos del entorno. Teniendo toda esta información de fuerzas, el robot deberá girar sobre si mismo hasta que su región de seguridad este totalmente vacía.



(a) (b)

Figura 4.16: Fuerzas aplicadas en avoidobstacles (a). Perfil velocidad angular (b)

Al igual que en el esquema `vff`, la velocidad se modula dependiendo del ángulo formado por la fuerza resultante y la fuerza atractora del robot. El perfil de la velocidad angular de este esquema, que se puede observar en la figura 4.16(b), es similar al perfil de la velocidad angular del esquema `vff`.

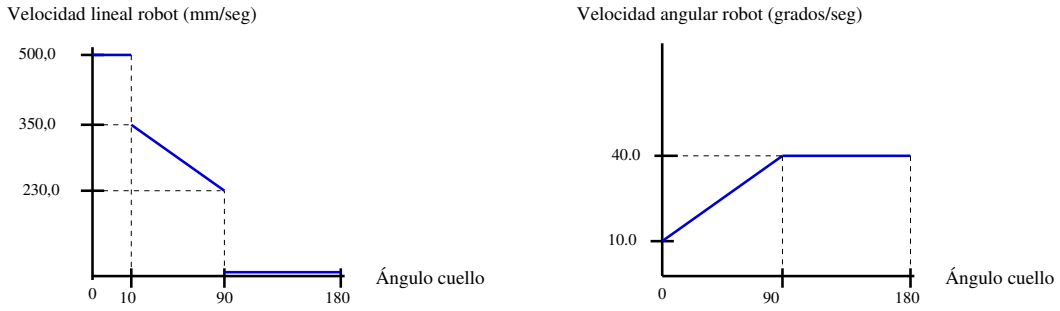
4.3.5. Esquema basetracking

Cuando no existe ningún obstáculo cercano al robot entra en funcionamiento este esquema. El esquema `basetracking` trata de seguir a la máxima velocidad posible al objetivo cuando no existe ningún obstáculo significativo en la dirección del objetivo.

Como el resto de esquemas que controlan la base motora del robot, tiene como entrada la información generada por el esquema `forces` y las lecturas de los sensores de ultrasonido y como salida comanda velocidades a los motores. Este esquema actuador se activa cuando se cumple su precondición y ésta se trata que no exista ningún obstáculo dentro de las regiones de seguridad de `vff` ni `avoidobstacles`. Esto se resuelve fácilmente mediante el arbitraje que se ha realizado y que se comentará en detalle en la siguiente sección.

Lo ideal del comportamiento sigue persona sería que este esquema estuviera el mayor tiempo posible activado, ya que es el esquema que sigue al objetivo a la mayor velocidad posible [Lobato, 2003] y sin peligro de chocar contra un obstáculo. Este esquema no se basa en fuerzas virtuales como los dos anteriores. Simplemente se basa en la fuerza objetivo, que la obtiene mediante la desviación que posea el cuello mecánico, para modular la velocidad del robot mediante una corrección proporcional similar a la utilizada para gobernar el cuello mecánico.

La posición del cuello mecánico es la información básica de este esquema, ya que la desviación angular del cuello implica directamente en las decisiones de control sobre la base del robot. Se ha creado una banda muerta de 20 grados al frente, en los que el robot no realiza giros, sólo se dirige recto hacia el objetivo a la máxima velocidad posible. Si se sale de esta banda muerta, el esquema comanda las velocidades lineales y angulares correspondientes. La velocidad máxima de seguimiento del objetivo es de 500mm/seg. Los perfiles de velocidad de este esquema se pueden observar en la figura 4.17



(a) (b)

Figura 4.17: Perfil de velocidad lineal (a) y velocidad angular (b) del esquema basetracking

4.3.6. Experimentos

Del mismo modo que en el seguimiento visual, vamos a comentar los experimentos que se han ido realizando en esta parte del seguimiento con la base motora. Los experimentos han resultado muy importantes para esta parte, ya que gracias a ellos se consigue ajustar el comportamiento deseado en la plataforma real.

Diferentes esquemas para el control del robot

Inicialmente el control de la base motora se reducía al uso de dos esquemas: un esquema que implementaba la navegación VFF y otro esquema que evitaba obstáculos cercanos. Pero se observó que no era necesaria la navegación basada en vff cuando no teníamos obstáculos. Con esto conseguimos eliminar la información de los obstáculos cuando no son relevantes.

La navegación mediante VFF era lenta para ciertos casos en los que el robot debía seguir al objetivo a una velocidad rápida y sin tener en cuenta los obstáculos, ya que éstos no representaban un peligro para el robot. Por eso se decidió establecer más casos para el control del robot. El diseño final quedó con 4 esquemas de control de la base móvil del robot: `stoprobot` (objetivo muy cerca o perdido), `avoidobstacles` (obstáculo cerca del robot), `vff` (obstáculo a media distancia) y `basetracking` (no se tiene en cuenta los obstáculos).

Con esta repartición de casos, se consiguió un comportamiento más vivaz y rápido, ya que cuando no existe ningún obstáculo en la dirección del objetivo el robot se dirige directamente hacia dicho objetivo.

VFF basado en sónares delanteros

Al comienzo se implementó el sistema de navegación VFF mediante todos los sensores que disponíamos en el robot, es decir, con toda la corona de sensores de ultrasonido, tanto la delantera como la trasera. En principio se puede pensar que cuanta más información se tenga del entorno del robot, mejor serán las decisiones que se toman en el movimiento. En este caso no es así, ya que los sensores de ultrasonido traseros no nos aportan información relevante y esto hacía que el robot oscilara en su movimiento.

Si el robot utiliza toda su corona de sensores de ultrasonido es capaz de detectar obstáculos tanto por delante como por detrás. Situándonos en un ejemplo concreto, supongamos que el robot navega por los pasillos de una oficina donde se encuentran varias puertas abiertas. Cuando el robot avanza y ha rebasado una puerta, los sensores de ultrasonido traseros detectan la puerta como un hueco sin obstáculos y ésto provoca que la fuerza resultante repulsiva cambie. Este cambio produce oscilaciones innecesarias en el movimiento del robot, ya que hemos pasado la altura de la puerta y aún la tiene en cuenta para los cálculos.

Como el robot siempre va a moverse hacia delante no tiene sentido tener información de los obstáculos por detrás del robot. Por ello se optó por prescindir de los sensores de ultrasonido traseros. El robot sólo va hacia el frente realizando giros de mayor o menor grado, pero nunca se mueve hacia atrás. No se tiene en cuenta que un obstáculo móvil aparezca detrás del robot. Con estas conclusiones se cambió la implementación usando únicamente los sensores delanteros de ultrasonido lo que provocó una mejora notable en la movimiento del robot ya que desaparecieron las oscilaciones.

Velocidades del robot

El motivo de diferenciar varios casos en la navegación de la base móvil del robot es consecuencia de la dificultad de crear un controlador proporcional de velocidad que contemple todas las posibles situaciones que se puedan presentar. El comportamiento de navegación VFF necesita unas velocidades lineales y angulares medias para producir una

sensación de continuidad. Mientras que el esquema `basetracking` necesita velocidades altas para seguir al objetivo.

En este punto se han realizado numerosos experimentos para conseguir una continuidad en el movimiento del robot, tanto dentro como fuera de los esquemas. El comportamiento perfecto en este caso, sería tener un controlador que contemple todos los casos e intente llevar al robot a la situación óptima en el menor tiempo posible. Este tipo de comportamientos se puede conseguir mediante un controlador borroso y obtener de este modo transiciones más suaves.

4.4. Seguimiento Global

Como se ha comentado durante la descripción de todo el comportamiento, existe un nivel superior por encima de todos los esquemas cuya funciones son la de arbitraje, coordinar y activar a las dos ramas principales del comportamiento. Este esquema decide en todo momento cual de los esquemas es el ganador de entre los que han cumplido sus precondiciones y quieren tomar el control del robot.

En este esquema se centraliza la actividad de todo lo necesario para el seguimiento del objetivo. Esta encapsulación de todo el comportamiento en un sólo esquema es un requisito propio de JDE. Además esto favorece la escalabilidad del comportamiento en la arquitectura de JDE si en un futuro un comportamiento nuevo utiliza como sub-bloque nuestro comportamiento.

Al inicio este esquema padre activa sus dos bloques de hijos. El primer bloque se encarga del seguimiento visual del objetivo que se lanza como una única hebra. El segundo bloque de hijos se encarga del seguimiento con la base motora. Este bloque se lanza con 4 hebras asíncronas ejecutándose en paralelo. Es necesario que por cada bloque sólo se active a un hijo a la vez para que al actuador, ya sea el cuello o robot, le lleguen las órdenes coherentes.

Es necesario un árbitro puesto que las precondiciones de los esquemas no son disjuntas y pueden solaparse. Como ya se comentó, hay situaciones en la que los esquemas `vff` y `avoidobstacles` quieren el control del robot porque sus precondiciones se satisfacen. En estos casos entra en funcionamiento el sistema de arbitraje distribuido que ofrece la plataforma JDE. Dicha arquitectura establece que un esquema de un nivel superior, puede activar o desactivar a esquemas de nivel inferior, de este modo los esquema forman un sistema jerárquico donde las decisión de activar o no las toman los esquemas superiores.

Existen dos tipos de arbitrajes: implícito y explícito. El arbitraje implícito se da cuando sólo un esquema satisface sus precondiciones y por tanto no es necesario usar el árbitro que hemos desarrollado. El arbitraje explícito se da cuando dos o más esquemas cumplen sus precondiciones (*colisión de control*) o cuando ningún esquema quiere tomar el control ya que no se cumplen sus precondiciones (*vacío de control*).

El arbitraje explícito lo realizamos mediante prioridades. Como muestra la figura 4.18 los esquemas que quieren tomar el control de la base motora tienen asociados unas prioridades. En el caso de que 2 o más esquemas quieran tomar el control del robot, el árbitro da como ganador al esquema que más prioridad tenga. En el caso que ningún esquema quiera tomar el control, el árbitro fuerza como ganador al esquema que mayor prioridad tenga, en este caso al esquema `stoprobot`.

Las prioridades de los esquemas se han establecido de modo conservador, ya que en caso de dos o más esquemas siempre se da como ganador al esquema que menos riesgos

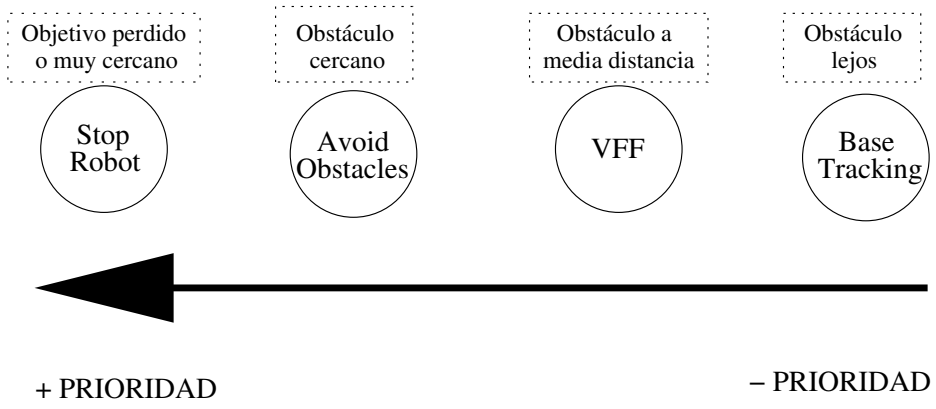


Figura 4.18: Prioridad en el arbitraje

suponga. Por ejemplo, si en un momento dado y por las circunstancias del entorno los esquemas `vff` y `avoidobstacles` quieren tomar el control del robot, el árbitro dará como ganador al esquema `avoidobstacles`.

4.4.1. Experimentos

Vamos a describir dos experimentos que se han realizado para comprobar el funcionamiento completo del seguimiento de personas. Concretamente se van a describir la evitación de obstáculos y la navegación por un pasillo siguiendo al objetivo.

Navegación por un pasillo

Este primer experimento trata de la navegación del robot por un pasillo con tránsito de personas que no son el objetivo.

En la figura 4.19 se puede observar la secuencia de como navega el robot por medio de un pasillo siguiendo al objetivo. Es conveniente observar que el objetivo se mueve a izquierda y derecha y el robot lo sigue realizando movimientos suaves para no producir brusquedad en el comportamiento. También se observa que aparecen más personas en la secuencia que no tiene en cuenta el robot como posibles objetivos. En esta situación de navegación sólo entra en funcionamiento los esquemas de `followtarget` y `vff` ya que no existen obstáculos muy cercanos al robot.

Evitación de obstáculos

Como se puede observar en la secuencia de la figura 4.20 el robot realiza una evitación de obstáculos, que en este caso se trata de una persona, sin perder en ningún momento al objetivo.

Inicialmente el esquema `followtarget` estaría activado ya que no existe ningún obstáculo cercano. Según se va moviendo el robot en la dirección del objetivo se va a encontrar un obstáculo y por tanto entrará en funcionamiento el esquema `vff`. Este esquema intentará sortear el obstáculo suavemente pero como no le da tiempo se acerca demasiado al éste, por lo que entra en acción el esquema `avoidobstacles` que hace girar

Figura 4.19: Navegación por un pasillo.

sobre si mismo el robot para evitar al obstáculo. Una vez que es sorteado el obstáculo entra en funcionamiento de nuevo el esquema `vff` e inmediatamente después será el esquema `follotarget` quien tomará el control del robot ya que no existen obstáculos. Es importante resaltar que toda el recorrido lo ha realizado teniendo en todo momento localizado al objetivo.

Figura 4.20: Secuencia de evitación de un obstáculo.

Capítulo 5

Conclusiones y trabajos futuros

Una vez descrita la solución propuesta para este comportamiento y la descripción de algunos experimentos relevantes, terminamos esta memoria resumiendo los principales aportes de este proyecto y sus posibles líneas futuras de trabajo.

5.1. Conclusiones

El objetivo principal descrito en el capítulo 2 era realizar un comportamiento de seguimiento de persona sin chocar con ningún obstáculo sobre un robot real. Bien, este objetivo se ha cumplido satisfactoriamente desarrollando dicho comportamiento en el robot Pioneer 3.

Como objetivo primario del diseño del comportamiento se comentó la necesidad de ponerse al día en las técnicas actuales para realizar el proyecto. Por ello durante el tramo inicial del proyecto se realizó un periodo de investigación y lectura de documentos técnicos sobre el estado del arte en navegación local y seguimiento de personas. Como objetivo secundario de diseño se propuso separar el comportamiento en dos bloques diferentes: uno encargado del seguimiento visual del objetivo y otro encargado del seguimiento con la base motora. Como se ha comentado en el capítulo 4 el diseño realizado se compone de dos bloques. El primer bloque realiza el seguimiento visual del objetivo mediante el cuello mecánico y se compone de un esquema perceptivo y dos esquemas actuadores. El segundo bloque realiza el seguimiento con la base motora del robot y se compone de un esquema perceptivo y cuatro esquemas de actuación. Cada uno de los bloques se componen de varios sub-bloques.

Como objetivo de implementación se comentó el desarrollo de programas para interactuar con el robot y sus actuadores. Debido a la utilización de la plataforma software de JDE se ha simplificado mucho la realización del comportamiento ya que no hemos tenido que programar a bajo nivel con los dispositivos. El bloque del seguimiento visual se ha implementado mediante hebras que sincronizan sus iteraciones mediante *callbacks*. Mientras que el bloque del seguimiento de la base motora se ha implementado mediante hebras asíncronas. Concretamente tenemos las siguientes hebras perceptivas: `forces.c` y `filtro.c` y las siguientes hebras actuadoras: `search.c`, `visualtracking.c`, `avoidobstacles.c`, `vff.c`, `follotarget.c` y `stoprobot.c`.

Como objetivo enfocado a la experimentación se comentó lo importante que es la realización de pruebas y experimentos en un proyecto de investigación como el presente.

Estos experimentos han influido notablemente en la calidad del comportamiento final. Gracias a estos experimentos se pudo ajustar el filtro de color para que se centre exclusivamente en el objetivo a seguir. Igualmente se consiguió un comportamiento de bastante calidad en el seguimiento del cuello y en la búsqueda del objetivo cuando éste se pierde. Gracias a las numerosas pruebas realizadas se consiguió un ajuste notable en la velocidad de la base motora, para que ésta no produjera cambios bruscos ni oscilaciones en su movimiento. Gracias a los experimentos que se han realizado sobre el robot se han resuelto varios problemas prácticos. Este punto es importantísimo en la robótica real ya que son necesarios dichos experimentos y pruebas para conseguir un ajuste aceptable en el comportamiento. Gracias a estas pruebas podemos concluir que la rapidez de proceso y tener unos controladores rápidos como el cuello mecánico, son fundamentales para poder generar un seguimiento en tiempo real y vivaz.

Los requisitos descritos en la sección 2.2 de la presente memoria deben ajustarse a los objetivos para asegurar el buen funcionamiento del comportamiento en el robot.

Uno de los requisitos más importantes era el número uno, que debido a la existencia de robots reales en el grupo de robótica pudimos desarrollar el comportamiento sobre el robot Pioneer 3. Este comportamiento, como se propuso en el requisito número dos, se implementará sobre la plataforma software JDE. Para que el comportamiento funcione correctamente debe ejecutarse sobre JDE (versión 2.4), oculo (versión 3.5) y otos (versión 4.7).

El seguimiento del objetivo, como se comentó en el requisito número 3, se realiza mediante visión gracias a la cámara instalada en el robot. La visión es un punto importante para la localización del objetivo. Mediante las imágenes que nos ofrece la cámara podemos obtener información muy completa de dónde se encuentra nuestro objetivo. Obtener dicha información es computacionalmente costosa, pero es bastante más eficaz que intentar localizar a una persona mediante un láser o sensores de ultrasonido. Para que el seguimiento visual se produzca correctamente debemos tener robustez ante cambios de iluminación, por ello y para cumplir el requisito número 4 se implementó el filtro en el espacio de color HSI, para no depender de la intensidad de la luz en el entorno.

Para que nuestro comportamiento tenga la calidad y robustez descrita en el requisito número 5, el robot debe responder lo más rápido posible a las acciones del entorno. Para ello debemos optimizar la programación de nuestro comportamiento y usar actuadores rápidos como el cuello mecánico. No usar el cuello [Ortiz, 2004] implica tener un actuador menos que controlar pero limita la visualización y búsqueda del objetivo.

Este proyecto ha servido para entender y experimentar por uno mismo, las fases en las que se divide un trabajo de esta importancia. Nos ha acercado hacia el mundo de la investigación y las implicaciones que conlleva esto, ya que hemos tenido que desarrollar nuevas librerías o retocar las aplicaciones existentes para resolver los problemas que se nos han planteado.

Todo el trabajo realizado ha servido para comprender lo largo y tedioso que puede llegar a ser un proyecto de mediano tamaño si no se siguen unas mínimas reglas de desarrollo. Han sido necesarios muchos ciclos de: estudio de la fase a realizar, discusiones o dudas con los tutores, profesores, compañeros de laboratorio; codificar, probar, depurar, documentar. Sería impensable el tiempo que hubiera requerido sin haber usado algún modelo de

desarrollo, en este caso, un modelo en espiral. También se han adquirido conocimientos sobre cómo almacenar las distintas versiones del código que se están generando e incluso aprender nuevos lenguajes de marcado como LaTeX sobre el que se ha redactado la presente memoria.

Además del trabajo reflejado por esta memoria, me gustaría destacar el trabajo que se realizó en el periodo de formación previo al proyecto. Fue necesaria esta labor de aprendizaje para tener un primer contacto con el hardware del robot, y del cuello mecánico. Fue necesario dedicar bastante tiempo al cuello, ya que se tuvo que implementar una librería de comunicación para este dispositivo bajo el sistema operativo *GNU/Linux*. También se realizó el montaje del robot y la colocación de sus dispositivos como el cuello y el láser. Este proceso ayudó a tener un primer contacto con los manuales de referencia de los dispositivos hardware y facilitó un mejor manejo en el entorno de desarrollo en el que se iba a programar el comportamiento.

5.2. Trabajos futuros

El comportamiento realizado en este proyecto funciona correctamente, cumpliendo todos los requisitos descritos en el capítulo 2. No obstante, se puede afinar y mejorar este comportamiento añadiendo nuevas funcionalidades. A continuación se detallan las posibles mejoras y funcionalidades que en un futuro se van a incluir en el presente proyecto.

1. Una mejora que se ha comentado durante toda la exposición del trabajo realizado es la detección de varios objetos del mismo color en la imagen. Esto implica que podamos diferenciar cuál de los dos objetos es el objetivo a seguir. La técnica empleada podría ser una segmentación [San Martín, 2002] o viendo donde se encontraba el objetivo en el pasado reciente mediante una ventana de seguimiento [Ortiz, 2004]. Además sería oportuno aplicar un filtro de reducción de ruido a la imagen para eliminar aquellas zonas que pasan el filtro pero que no pertenecen al objetivo.

2. Otra mejora referente a la percepción visual es la incorporación de una cámara firewire en sustitución de la cámara actual USB, ya que aumentaría notablemente el número de imágenes por segundo transmitidas al PC. Este punto es esencial para conseguir una mejora en la calidad del seguimiento.

3. Una aportación puede ser la implementación un controlador borroso para la movilidad de la plataforma motora del robot. Con el controlador borroso ganamos continuidad en el movimiento lo que conlleva un movimiento más suave del robot.

4. Por último, se puede optar por un seguimiento multimodal de la persona. Esto implica que si se pierde a la persona mediante la cámara, es capaz el sistema de seguimiento saber dónde se encuentra la persona mediante las lecturas del sensor láser.

Apéndice A

Librería de comunicación con el cuello mecánico

Como se ha comentado a lo largo de esta memoria, el cuello mecánico que se ha utilizado para realizar el seguimiento visual no tenia soporte ofrecido por el fabricante bajo la plataforma GNU/Linux. Por ello se dedicó un periodo de este proyecto a investigar sobre este dispositivo. Esta investigación dió como fruto el desarrollo de una librería de comunicación para interactuar con este dispositivo.

La comunicación que se realiza con el cuello mecánico se basa en el protocolo de bajo nivel RS-232[pan, 2001]. Esta librería ofrece las funcionalidades básicas de comunicación. Permite conectarse/desconectarse al cuello, enviar o recibir comandos y establecer la velocidad del puerto serie. Además esta librería incluye un conjunto de constantes que contienen comandos básicos del protocolo que utiliza el cuello[pan, 2001]

Cabeceras de la librería *controlPTU*

La librería *control-PTU* ofrece la siguiente interface al programador:

```
/* Abre conexión con la PANTILT */
int openPANTILT (char *filename, mode_t io_flags);

/* Cierra la conexion con la PANTILT */
int closePANTILT (int fd);

/* Establece la velocidad del puerto serie */
int SetBaudRate (int fd, speed_t baudRate);

/* Manda un comando a la PANTILT */
int SendCommand (int fd, char cmd[]);

/* Obtenemos la respuesta de la pantilt */
void ReadCmd (int serial_port);
```

Constantes definidas en el protocolo del cuello

Estas constantes favorecen la simplicidad de la programación de cara al programador, ya que no debe memorizar toda la sintaxis del protocolo que utiliza el cuello. A continuación se muestran los comandos definidos como constantes:

```
/* Configuracion del puerto serie */

#define SERIE "/dev/ttyS21"     /* Dispositivo del puerto serie */
#define RS232_BAUD_RATE B9600   /* Velocidad del puerto serie*/
#define MAX_MESSAGE 200         /* Máximo número de caracteres */

extern char respuesta [MAX_MESSAGE];  /* Respuesta del cuello */


/* Movimientos de la PAN-TILT   */

#define INIT_PAN "PP0 *"          /* Mueve el eje PAN al centro */
#define INIT_TILT "TP0 *"         /* Mueve el eje TILT al centro */
#define MAX_PAN "PP3090 *"        /* Maximo movimiento del eje PAN */
#define MIN_PAN "PP-3090 *"       /* Minimo movimiento del eje PAN */
#define MAX_TILT "TP604 *"        /* Maximo movimiento del eje TILT */
#define MIN_TILT "TP-907 *"       /* Minimo movimiento del eje TILT */
#define STOP_ALL "H *"            /* Para los ejes inmediatamente */
#define STOP_PAN "HP *"           /* Para el eje PAN inmediatamente */
#define STOP_TILT "HT *"          /* Para el eje TILT inmediatamente */
#define WAIT_A_COMMAND "A *"      /* Espera la ejecucion del comando
                                   actual para ejecutar el siguiente */
#define MONITOR_ON "ME *"         /* Activa el monitor (Autoscan) */
#define MONITOR_OFF "MD *"        /* Desactiva el monitor (Autoscan) */
#define MONITOR_INFO "MQ *"       /* Pregunta el estado (Autoscan) */


/* Velocidad de la PAN-TILT */

#define SPEED_PAN "PD *"          /* Pregunta por la velocidad PAN */
#define SPEED_TILT "TD *"         /* Pregunta por la velocidad TILT */
#define SPEED_MAX_PAN "PU *"      /* Pregunta por la velocidad máxima PAN */
#define SPEED_MIN_PAN "PL *"      /* Pregunta por la velocidad mínima PAN */
#define SPEED_MAX_TILT "TU *"     /* Pregunta por la velocidad máxima TILT */
#define SPEED_MIN_TILT "TL *"     /* Pregunta por la velocidad mínima TILT*/
#define SPEED_BASE_PAN "PB *"     /* Pregunta la vel base PAN */
#define SPEED_BASE_TILT "TB *"    /* Pregunta la vel base TILT */
#define ACELARACION_PAN "PA *"    /* Pregunta por la acleracion PAN */
#define ACELERACION_TILT "TA *"   /* Pregunta por la aceleracion TILT */

/* Comandos PAN-TILT */
```

```
#define RESET_ON "R *"            /* Ejecuta el reseteo de la pantilt */
#define RESET_OFF "RD *"          /* Desactiva el reset de los ejes */
#define RESET_ALL "RE *"          /* Activa el reset de los dos ejes */
#define RESET_PAN "RP *"          /* Activa el reset del eje PAN */
#define RESET_TILT "RT *"         /* Activa el reset del eje TILT */

#define SAVE_SETTINGS "DS *"     /* Guarda la configuracion actual */
#define RESTORE_SETTINGS "DR *"  /* Restaura la configuracion predeterinada */
#define RESTORE_FACTORY "DF *"   /* Restaura la configuracion fabrica */

#define ECHO_INFO "E *"          /* Pregunta por el estado del echo */
#define ECHO_ON "EE *"           /* Activa el echo propio de la pantilt */
#define ECHO_OFF "ED *"          /* Desactiva el echo propio de la pantil */

#define ASCII_INFO "F *"         /* Pregunta el modo ASCII */
#define ASCII_VERBOSE "FV *"     /* Activa el modo ASCII VERBOSE. */
#define ASCII_TERSE "FT *"       /* Activa el modo ASCII TERSE. */

#define VERSION "V *"             /* Da informacion sobre la PAN-TILT */

/* Comandos de control de power en estatico */

#define POWER_INFO_PAN "PH *"        /* Pregunta el modo power del eje PAN */
#define POWER_REGULAR_PAN "PHR *"    /* Estable el modo 'regular' al eje PANT*/
#define POWER_LOW_PAN "PHL *"        /* Estable el modo 'low' al eje PANT */
#define POWER_OFF_PAN "PHO *"        /* Desactiva el power en el eje PANT */

#define POWER_INFO_TILT "TH *"       /* Pregunta el modo power del eje TILT */
#define POWER_REGULAR_TILT "THR *"   /* Estable el modo 'regular' al eje TILT */
#define POWER_LOW_TILT "THL *"       /* Estable el modo 'low' al eje TILT */
#define POWER_OFF_TILT "THO *"       /* Desactiva el power en el eje TILT */

/* Comandos de control de power en movimiento */
/* WARNING: No es recomendando que el eje este en
tránsito mas de la mitad cuando el modo es HIGH */

#define POWER_MOVE_INFO_PAN "PM *"     /* Pregunta el modo power PAN */
#define POWER_MOVE_HIGH_PAN "PMH *"    /* Establece el modo 'high' PANT */
#define POWER_MOVE_REGULAR_PAN "PMR *" /* Establece el modo 'regular' PANT */
#define POWER_MOVE_LOW_PAN "PML *"     /* Establece el modo 'low' PANT */

#define POWER_MOVE_INFO_TILT "TM *"     /* Pregunta el modo power TILT */
#define POWER_MOVE_HIGH_TILT "TMH *"    /* Establece el modo 'high' TILT */
#define POWER_MOVE_REGULAR_TILT "TMR *" /* Establece el modo 'regular' TILT */
#define POWER_MOVE_LOW_TILT "TML *"     /* Establece el modo 'low' TILT */
```

Bibliografía

[Borenstein y Koren, 1989] J. Borenstein y Y. Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Journal of Robotics and Automation*, 1989.

[c, 1987] *El entorno de programación UNIX*. Prentice Hall, 1987.

[Cañas, 2002] Jose María Cañas. Dynamic schema hierarchies for an autonomous robot. *Universidad de Sevilla*, Noviembre 2002.

[Cañas, 2003] Jose María Cañas. Jerarquía dinámica de esquemas para la generación de comportamiento autónomo. *Tesis Doctoral*, Diciembre 2003.

[Cañas, 2004] Jose María Cañas. Manual de programación de robots con jde. *Universidad Rey Juan Calos*, pages 1–36, abril 2004.

[Christensen, 1999] H. Sidenbladh D. Kragic H. I. Christensen. A person following behaviour for a mobile robot. *IEEE Journal of Robotics and Automation*, 1999.

[Crowley, 1985] James L. Crowley. Navigation for an intelligent mobile robot. *IEEE Journal of Robotics and Automation*, 1(1):31–41, March 1985.

[Feyrer y Zell, 2001] Tefan Feyrer y Andreas Zell. Robust real-time pursuit of persons with a mobile robot using multisensor fusion. *Wilhelm-Schickard-Institute, Department of Computer Architecture. University of Tübingen*, 2001.

[Fink y Sagerer, 2002] M. Kleinehagenbrock S. Lang J. Fritsch F. Lomker G. A. Fink y G. Sagerer. Person tracking with a mobile robot based on multi-modal anchoring. *Faculty of Technology, Bielefeld University*, septiembre 2002.

[García y Bustos, 2001] M.C. García y P. Bustos. Complex behaviour generacion on autonomouse robots: A case study. *Instituto de Automática Industrial*, 2001.

[GSyC, 2004] GSyC. Tema de navegación del temario de robotica. *Universidad Rey Juan Carlos-PFC*, 2004.

[Gómez, 2002] Victor Manuel Gómez. Comportamiento sigue pared en un robot con visión local. *Universidad Rey Juan Carlos-PFC*, 2002.

[Isado, 2004] José Raúl Isado. Navegación global utilizando método del gradiente. *Universidad Rey Juan Carlos-PFC*, 2004.

[las, 2002] *LMS. Definition of telegrams between the user interface and LMS systems via RS 232*, 2002.

[lat, 2000] *Latex, una imprenta en sus manos.* AD, 2000.

[Lobato, 2003] David Lobato. Evitación de obstáculos basada en ventana dinámica. *Universidad Rey Juan Carlos-PFC*, 2003.

[Martínez, 2003] Marta Martínez. Comportamiento sigue pelota con visión cenital. *Universidad Rey Juan Carlos-PFC*, 2003.

[Matute, 2002] Alfonso Matute. Filtro color configurable. *Universidad Rey Juan Carlos-PFC*, 2002.

[Ortiz, 2004] Ricardo Ortiz. Comportamiento sigue persona con visión. *Universidad Rey Juan Carlos-PFC*, 2004.

[pan, 2001] *Computer Controlled PanTilt Unit. User's Manual.*, 2001.

[Peño, 2003] Manuel Peño. Comportamiento sacar de banda en un robot futbolista autónomo. *Universidad Rey Juan Carlos-PFC*, 2003.

[San Martín, 2002] Félix San Martín. Comportamiento sigue pelota. *Universidad Rey Juan Carlos-PFC*, 2002.

[Worz, 1997] Christian Schlegel Jorg Illmann Heiko Jaberg Matthias Schuster Robert Worz. Vision based person tracking with a mobile robot. *Institute for Applied Knowledge Processing (FAW)*, 1997.