UNIVERSIDAD REY JUAN CARLOS

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

Escuela Superior de Ciencias Experimentales y Tecnología

Curso académico 2005-2006

Proyecto Fin de Carrera

Representación rica de la escena 3D alrededor de un robot móvil.

Autor: Ricardo Palacios Maya

Tutor: José María Cañas Plaza

Para mi novia, familia y amigos.

Agradecimientos.

Este proyecto no se habría podido realizar sin la generosa colaboración y tan buen asesoramiento de José María Cañas a quien expreso mi sincero agradecimiento.

Deseo extender un especial reconocimiento a mis compañeros de estudios, en particular a Fructuoso Martínez, Iván García, José María Esteban y José María Nacher, por el respaldo e interés demostrado para la presentación de este proyecto.

Quiero agradecer el apoyo y comprensión, en estos tres duros años de carrera, de mi familia y amigos, en especial a Alberto Gallego y Antonio Calero, que tanto han creído en mí.

Por último, deseo destacar a mi amiga, novia y confesora quien en estos años de convivencia ha sido tan paciente en los peores momentos. Gracias, Elena Suárez, por tu imprescindible compañía.

Resumen.

La robótica autónoma permite a un robot percibir la situación y actuar apropiadamente. Dotado de sensores fiables y con una representación más estructurada, el robot tendrá la capacidad de navegar autónomamente por el entorno dinámico con una mayor seguridad y vivacidad.

El propósito de este proyecto es dotar a un robot real con la capacidad de identificar y localizar diversos objetos cotidianos en la escena para facilitarle la interacción con ellos y, sobre todo, para navegar de manera segura y eficaz. Gracias al entendimiento de su entorno dinámico, puede tomar mejores decisiones a la hora de desplazarse de un lugar a otro o de interactuar con los objetos identificados.

Los objetos detectables son personas, congéneres, puertas y paredes; hemos elegido éstos por ser habituales en robótica de interiores. Para reconocerlos nos basamos en las medidas del láser y la odometría, así como en las imágenes capturadas por la cámara. Después de realizar la fusión sensorial, se decidirá si es realmente el objeto buscado aplicando *suma heterogénea de subestímulos*. Una vez identificado un objeto, es almacenado y clasificado en una memoria de corto plazo. Dicha memoria requiere un mantenimiento para permanecer coherente y, de este modo, poner la información correcta y estructurada a disposición del algoritmo de navegación, el cual se ha implementado usando la técnica de navegación local *Campo de Fuerzas Virtuales* (VFF).

La aplicación ha sido desarrollada apoyándose en la plataforma software JDE e implementada en forma de cinco hebras o esquemas, siendo cuatro perceptivas (una para cada objeto detectable) y una actuadora (encargada de la navegación). Además, un sexto esquema de servicio permite la visualización tridimensional del entorno reconocido por el robot.

Se han conseguido identificaciones suficientemente robustas y, gracias a ellas, una navegación fiable y vivaz. Se abre paso, además, a la posibilidad de interacciones complejas con los objetos que hemos citado.

Índice general

Índice de figuras

Capítulo 1
Introducción

El principal objetivo de este Proyecto Fin de Carrera es que un robot móvil identifique diversos objetos cotidianos, permitiendo memorizar a corto o medio plazo su localización, así como visualizar en el interfaz la escena cercana en tres dimensiones. Todo ello navegando de modo seguro, ya sea deambulando, explorando o ejecutando alguna orden del usuario.

En este capítulo se va a dar una visión global del contexto del presente proyecto: Comienza con una descripción general de la robótica; a continuación, se introduce un pequeño resumen de los términos navegación y visión en robots; por último, termina con una breve descripción del proyecto realizado.

1.1. Robótica

La robótica es una ciencia o rama de la tecnología que estudia el diseño y construcción de máquinas capaces de desempeñar tareas realizadas por el ser humano o que requieren del uso de inteligencia. Las ciencias y tecnologías de las que deriva podrían ser: el álgebra, los autómatas programables, las máquinas de estados y, sobre todo, la mecánica y la informática.

La historia de la robótica ha estado unida a la construcción de "artefactos", que trataban de materializar el deseo humano de crear seres semejantes a nosotros que nos descargasen del trabajo. Karel Capek, un escritor checo, acuño en 1921 el término Robot en su obra dramática "Rossum's Universal Robots / R.U.R.", a partir de la palabra checa *Robbota*, que significa servidumbre o trabajo forzado. El término robótica es acuñado por Isaac Asimov, definiendo a la ciencia que estudia a los robots. En la ciencia ficción el hombre ha imaginado a los robots visitando nuevos mundos, haciéndose con el poder o, simplemente, aliviándonos de las labores caseras.

Los robots son usados, hoy en día, para llevar a cabo tareas sucias, peligrosas, difíciles o repetitivas para los humanos. Esto, usualmente, toma la forma de un robot industrial usado en las líneas de producción. Otras aplicaciones incluyen la limpieza de

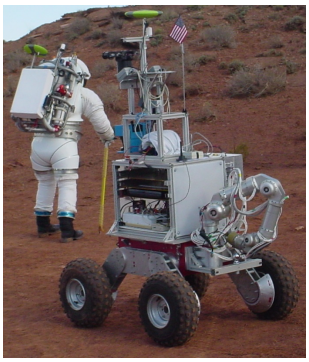
residuos tóxicos, exploración espacial, minería, búsqueda y rescate de personas y localización de minas terrestres. La manufactura continúa siendo el principal mercado donde los robots son utilizados. En particular, robots articulados (similares en capacidad de movimiento a un brazo humano) son los más usados comúnmente. Las aplicaciones incluyen soldado, pintado y carga de maquinaria. La industria automotriz ha obtenido gran beneficio de esta nueva tecnología donde los robots han sido programados para reemplazar el trabajo de los humanos en muchas tareas repetitivas (figura 1.1(a)).

Existe una gran esperanza, especialmente en Japón, de que el cuidado del hogar para la población de edad avanzada pueda ser llevado a cabo por robots. En esta línea se encuentra el robot-aspirador Roomba (figura 1.1(b)).

Recientemente, se ha logrado un gran avance en los robots teleoperados dedicados a la medicina, que han recibido la aprobación regulatoria en América del Norte, Europa y Asia para que dichos robots sean utilizados en procedimientos de cirugía invasiva mínima. La automatización de laboratorios también es un área en crecimiento. Aquí, los robots son utilizados para transportar muestras biológicas o químicas entre instrumentos tales como incubadoras, manejadores de líquidos y lectores. Otros lugares donde los robots están reemplazando a los humanos son la exploración del fondo oceánico y exploración espacial (figura 1.1(c)).

(a) Robots en cadena de producción

(b) Aspirador-Robot Roomba



(c) Robot astronauta

Figura 1.1: Algunas utilidades de los robots hoy en día

Robots alados experimentales y otros ejemplos que explotan el biomimetismo también están en fases previas. Se espera que los así llamados “nanomotores” y “cables inteligentes” simplifiquen drásticamente el poder de locomoción, mientras que la estabilización en vuelo parece haber sido mejorada sustancialmente por giroscopios extremadamente pequeños. Un impulsor muy significativo de este tipo de trabajo es el desarrollar equipos de espionaje militar (figura 1.2(a)).

(a) PatrolBot (b) Robot Aibo (c) Robot T

Figura 1.2: (a) Robot militar, (b) robot de entretenimiento y (c) robot de investigación

Por supuesto, no podemos olvidar los robots creados para el entretenimiento, con formas simpáticas y agradables a la vista (figura 1.2(b)), o los que se utilizan para investigación en universidades (figura 1.2(c)). Los cuales permiten ser programados por los usuarios.

Cualquier robot está compuesto por una serie de elementos imprescindibles como son los sensores, actuadores, controlador/es y ordenador/es.

Los *sensores* constituyen el sistema de percepción del robot. Son dispositivos físicos que miden cantidades físicas de propiedades (distancia, sonido, magnetismo, presión, altitud, velocidad, ...). Se pueden clasificar como internos o externos. Los internos proporcionan información sobre el propio robot como puede ser su posición, velocidad o aceleración. Los externos proporcionan información sobre lo que rodea al robot como la proximidad, tacto, fuerza o visión.

Los *actuadores* permiten al robot interaccionar con el mundo. Son dispositivos que ejerciendo fuerzas generan movimiento en los elementos del robot. Existen varios tipos de actuadores: neumáticos, hidráulicos o eléctricos.

Los *controladores* y *ordenadores* son los dispositivos encargados de coordinar la percepción y la actuación del robot, como puede observarse en el ejemplo propuesto (figura 1.3). Por tanto, necesitaremos programarlos para obtener el comportamiento deseado, como podría ser conseguir un alto grado de autonomía, es decir, que el robot sea capaz de percibir la situación y actuar apropiadamente.

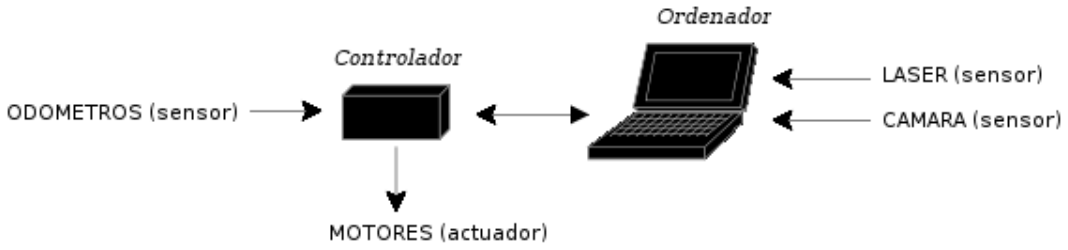


Figura 1.3: Coordinación de sensores y actuadores

Problemas abiertos en robótica, en concreto referentes a navegación y visualización, pueden ser varios:

Movimiento autónomo de robots que operan en escenarios complejos: Los principales problemas se encuentran en conseguir buenos movimientos en robots con pocos grados de libertad (móviles con ruedas), así como en obtener técnicas fiables para evitar obstáculos en robots con un elevado número de grados de libertad.

Localización y construcción de mapas concurrente: Construir, mantener y utilizar un mapa es altamente "costoso". La mayor dificultad deriva en garantizar un sistema de posicionamiento fiable del robot y de los elementos modelados del entorno (por el que el robot va navegando) a partir de sensores y actuadores limitados y/o ruidosos.

Visualización en entornos reales: Constituye una de las líneas de investigación que más está interesando. El mayor problema es la detección de objetos en la imagen en caso de deslumbramiento, o cuando dichos objetos aparecen en penumbra debido a una luz ambiente muy débil.

1.2. Navegación de robots

Se define navegación como la metodología (o arte) que permite guiar el curso de un robot móvil a través de un entorno con obstáculos. Existen diversos esquemas, pero todos ellos poseen en común el afán por llevar el vehículo a su destino de forma segura. La capacidad de reacción ante situaciones inesperadas debe ser la principal cualidad para desenvolverse, de modo eficaz, en entornos no estructurados. Dicha metodología distingue entre navegación global y local.

La navegación global se basa en calcular la ruta antes de empezar a moverse. Para ello, se tiene que conocer el entorno de antemano, bien con un mapa conocido a priori , o bien, generando un mapa del entorno con los sensores del robot. Una vez que el robot tiene conocimiento del entorno y sabe dónde se encuentra en ese mapa, se calcula

la ruta a seguir, cuyas técnicas son bastante costosas. Este tipo de navegación tiene el inconveniente de los obstáculos dinámicos, es decir, cuando el robot se dirige hacia su destino y se encuentra con un objeto que no tiene en el mapa (por ejemplo una persona), el robot no sabría evitarlo. Algunas técnicas desarrolladas en este tipo de navegación son: Grafo de visibilidad (Nilson, 1969) [López, 2005], diagramas de Voronoi y planificación basada en gradiente (Konolige, 2000) [Isado, 2005].

La navegación local es reactiva, ya que se basa sólo en las medidas obtenidas del entorno por los sensores en cada instante, sin la utilización de mapas. El robot toma una decisión en ese mismo momento. Así, con este tipo de navegación podemos seguir algo, navegar en una dirección determinada, pero no podemos ir de una parte de un edificio a otra, al no tener conocimiento a priori de un mapa. Una gran ventaja frente a la navegación global es la posibilidad de evitar obstáculos dinámicos. Algunas técnicas desarrolladas sobre láser(t) en este tipo de navegación son: Método de Velocidades y Curvaturas (CVM) [Lobato, 2005], Método de Carriles y Velocidad (VLM) y Campos de Fuerza Virtuales (VFF).

Debido a que estos dos tipos de navegación tienen inconvenientes, surge la navegación híbrida, que combina la planificación de caminos de la navegación global con la ejecución reactiva de la navegación local [Isado, 2005]. En la figura 1.4 puede observarse la planificación de la ruta a seguir basada en gradiente (navegación global), la cual está complementada con una técnica de navegación local.

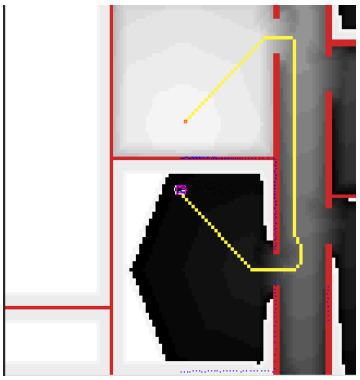


Figura 1.4: Campo de gradiente y el camino óptimo entre origen y destino

De este modo, se tienen estimaciones fiables de distancia respecto a cualquier obstáculo cercano, lo cual permite tomar decisiones para sortearlo. Sin embargo, no se hace distinción entre los obstáculos. La distancia entre el robot y un objeto inmóvil depende de la planificación y/o decisión del propio robot autónomo, no siendo así respecto de objetos móviles que podrían acercarse en sentido contrario o realizar un movimiento inesperado, lo cual crearía una situación peligrosa al no dar tiempo a evitarlo.

Por tanto, se debería extremar las precauciones si se encuentra cerca un objeto potencialmente dinámico. Para ello, es necesario una representación más rica del entorno que permita al robot tomar mejores decisiones, como podría ser disminuir la velocidad si tiene demasiado cerca otro robot, o no girar bruscamente si sabe que justo detrás se encuentra una persona.

1.3. Visión en robótica

Campo de la informática que estudia el uso de cámaras como sensores. Su función principal es reconocer y localizar objetos en el entorno mediante el procesamiento de las imágenes. La visión artificial estudia estos procesos para construir máquinas con capacidad de "entender" una escena o las características de una imagen. Con la incorporación de cámaras en los robots se obtiene un sensor capaz de procesar una muy alta cantidad de datos a bajo coste económico. Sin embargo, la extracción de la información a partir de dichos datos es compleja, lo que implica un alto coste computacional.

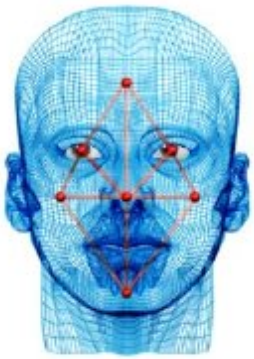


Figura 1.5: Reconocimiento facial

Los objetivos típicos de la visión artificial incluyen:

- La detección, segmentación, localización y reconocimiento de ciertos objetos en imágenes, por ejemplo caras humanas (figura 1.5).
- Identificar en diferentes imágenes una misma escena u objeto.
- Seguimiento de un objeto en una secuencia de imágenes.
- Mapeo de una escena para generar un modelo tridimensional de la escena; tal modelo podría ser usado por un robot para navegar por dicha escena.
- Estimación de las posturas tridimensionales de humanos.
- Búsqueda de imágenes digitales por su contenido.

Estos objetivos se consiguen por medio de reconocimiento de patrones, aprendizaje estadístico, geometría de proyección, procesado de imágenes, teoría de grafos y otros campos.

1.4. Representación rica de la escena alrededor de un robot móvil

Una vez presentado el contexto general de la robótica y dos de los campos en los que se sitúa este proyecto (navegación y visión en robots) pasamos a describir el contexto más cercano que motivó este proyecto: La representación tridimensional de una escena rica.

El comportamiento deseado de nuestro robot consiste en identificar diversos objetos (móviles e inmóviles) del mundo real que pueda encontrar mientras navega, memorizar a corto o medio plazo su localización con respecto a él, representarlos en 3D y facilitar la interacción con ellos.

Para intentar conseguir una identificación más robusta que el reconocimiento de patrones clásico, nos hemos basado en el principio de suma heterogénea de estímulos [Lorenz, 1978], el cual afirma que si la suma de un conjunto de estímulos supera cierto umbral motivan los comportamientos y/o producen la desencadenación de comportamiento. Un ejemplo de este principio es la apertura del pico de las crías de gaviota cuando viene el progenitor para alimentarles, intervienen varios factores: la forma del pico, una mancha roja en éste, forma y color de la cabeza, etc. Lo cual puede ser fácilmente extendido a otros estímulos [Hidalgo, 2006] apoyado en una atención visual [Martínez, 2003] para lograr nuestro objetivo: reconocer un objeto en la escena y representarlo en tres dimensiones.

Para tratar de explicar cómo se llevó a cabo, esta memoria está organizada, en primer lugar, con un capítulo dedicado a los objetivos y requisitos que ha de cumplir el comportamiento. Posteriormente, en el capítulo de herramientas, se analiza con detalle el robot sobre el que se realizaron los experimentos y los diferentes dispositivos de los que dispone. En la parte de software, hablaremos de la plataforma de programación sobre la que se ha desarrollado el comportamiento, llamada JDE. A continuación, encontraremos un capítulo dedicado a la descripción informática del citado comportamiento, donde se explica la solución desarrollada para este proyecto, describiendo los programas diseñados para conseguirlo. Una vez comentada la solución, analizaremos los experimentos realizados. Finalmente, se describirán las conclusiones y una línea de posibles trabajos futuros sobre el mismo proyecto.

Capítulo 2
Objetivos

Una vez presentado en el capítulo 1 el contexto general y particular de este proyecto, vamos a fijar los objetivos concretos de este proyecto fin de carrera y los requisitos que han condicionado su desarrollo.

2.1. Descripción del problema

El objetivo principal es obtener, a través de los sensores odométrico, láser, cámara y su correspondiente fusión sensorial, una representación rica de la escena en tres dimensiones.

Los objetivos son reconocer diversos objetos (personas, congéneres, puertas y paredes), localizarlos en el mapa que va creando dinámicamente (con una memoria a corto o medio plazo), representarlos en tres dimensiones y, por último, facilitar la interacción con ellos. Todo ello sin comprometer la integridad del propio robot y los objetos que lo rodean, es decir, navegar de modo seguro ya sea explorando el entorno o ejecutando alguna acción.

Para conseguir nuestro objetivo final, se han realizado los siguientes subobjetivos:

- Obtener de un objeto, a partir de los sensores odométrico y láser, su anchura y posición respecto del robot, así como identificar si es móvil o inmóvil.
- Obtener de un objeto, a partir de la cámara, su color y forma, además de poder reconocer sus posibles movimientos.
- Fusionar la información obtenida por los sensores para identificar y localizar, de una manera robusta y coherente, los objetos detectados.
- A partir de dicha identificación y localización, visualizar en el interfaz gráfico la representación tridimensional de la escena, permitiendo variadas perspectivas.
- Navegación local que permita explorar el entorno e interactuar con los objetos identificados de manera segura, dicha navegación está basada en campos de fuerza virtuales, apoyado tanto en láser(t) como en los objetos clasificados y localizados.

2.2. Requisitos

El desarrollo de este proyecto está guiado por los objetivos comentados anteriormente y deberá ajustarse a los requisitos de partida, asegurando de este modo un buen comportamiento.

1. El comportamiento diseñado debe realizarse en el robot real *Pioneer* 3-DX, pudiéndose apoyar en herramientas de simulación como es *Player/Stage*, sin embargo, el proyecto final deberá funcionar correctamente sobre el robot real. Como fuente de entrada nos basamos en tres sensores: odométrico, cámara y láser.

2. Para la generación del comportamiento deseado, debe usarse la plataforma software *JDE*, la cual permitirá la fácil integración y/o reutilización de los esquemas generados en futuros proyectos, así como la facilidad de incorporar esquemas ya existentes en el nuestro (alta cohesión, bajo acoplamiento y potencial reutilización). El entorno citado establece la programación de la aplicación con el lenguaje C, bajo el sistema operativo Linux.

3. La navegación debe ser reactiva en todo momento, así como vivaz. Los algoritmos de navegación no pueden tardar demasiado tiempo decidiendo el próximo movimiento a realizar, ya que ha de reaccionar lo más rápido posible para evitar colisionar con cualquier obstáculo.

4. Los algoritmos de identificación deben ser muy robustos, para que el robot no sea engañado con cualquier otro objeto de color, forma y tamaño parecido. Además, los objetos se podrán identificar en condiciones de luminosidad variables.

5. La representación de la escena en tres dimensiones debe ser fiel a la realidad, por tanto, debe asumir el dinamismo y la coherencia de la misma, además de un buen diseño gráfico para los objetos de la escena visualizados.

2.3. Metodología y plan de trabajo

El plan de trabajo llevado a cabo para la realización de este proyecto ha consistido 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. Con esto se aporta flexibilidad en cuanto a cambio de requisitos.

En este tipo de modelo de desarrollo existen cuatro etapas (figura 2.1): *Análisis de requisitos, diseño e implementación, pruebas y planificación del próximo ciclo de desarrollo.*

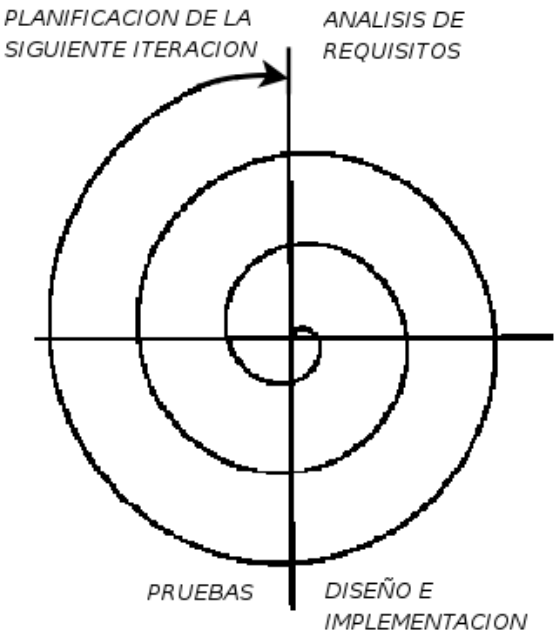


Figura 2.1: Modelo en espiral

Durante la realización del proyecto, se han acordado reuniones semanales con el tutor para establecer los puntos a llevar a cabo en cada etapa del mismo, así como estudiar los resultados de etapas anteriores. Este plan de trabajo se ha dividido a través del modelo en espiral en varias etapas más concretas:

1. Familiarización con la plataforma software utilizada en el desarrollo de este proyecto. Esta plataforma es JDE, la cual será descrita en el capítulo 3.

2. Segmentación de los puntos determinados por el sensor láser, así como la memorización y localización de los segmentos creados.

3. Tratamiento de imágenes: Filtros de color y movimiento, así como segmentación de las imágenes filtradas.

4. Diseño e integración de esquemas para la identificación de paredes, puertas, congéneres y personas; realizados a partir de la fusión sensorial (información obtenida de la odometría, láser y cámara).

5. Visualización tridimensional de la escena estudiada.

6. Generación de un esquema para la navegación local.

7. Experimentos globales en el robot real, con la integración de los prototipos creados que cumplen cada subobjetivo.

Capítulo 3
Entorno y plataforma de desarrollo

En este capítulo, presentamos las herramientas donde nos hemos apoyado para realizar el proyecto: el robot donde se ejecuta la aplicación y la plataforma software sobre la que se ha programado. También, comentaremos algunas bibliotecas auxiliares en las que nos hemos apoyado.

3.1. Infraestructura hardware: Robot Pioneer

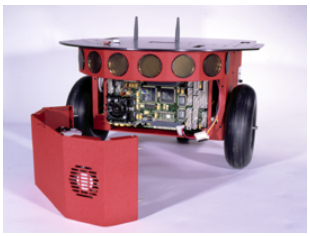
El robot en el que se desarrolla la aplicación es el *Pioneer 3DXE* (figura 3.1), y está comercializado por la empresa *ActivMedia*[1] . El *Pioneer* está dotado de unos sensores que le permiten medir el entorno, unos procesadores que le dan capacidad de cálculo y unos actuadores que le permiten moverse.

Figura 3.1: Robot Pioneer

En el apartado de procesamiento, el robot dispone de un único procesador de 18 MHz Hitachi H8S/2357 con 32Kb RAM y 128Kb de Memoria flash.Tiene una autonomía de cinco horas.Es capaz de adquirir una velocidad lineal de 1,8 m/s y una velocidad angular de 360 grados/s.Puede llegar a soportar una carga de 23 Kg. Encima de esta base se ha situado un ordenador portátil (Intel Centrino a 1.6 GHz, con 1Gb de RAM) donde ejecutaremos nuestra aplicación.

[1] http://www.activrobots.com/ROBOTS/index.html#p2dx

En el apartado sensorial, el robot dispone de unos odómetros (encoders) que cuentan las vueltas de cada rueda. Gracias a estos odómetros obtendremos información de posición (v,w,Θ) bidimensional. También posee una corona de 16 sensores de ultrasonido (figura 3.2(a)). A este equipamiento se le ha añadido un láser *SICK*[2] (figura 3.2(c)) y dos cámaras firewire *iSight*[3] (figura 3.2(b)). Los actuadores principales son dos motores de continua, cada uno asociado a una rueda motriz, con ellos se dota al robot de un movimiento de tracción diferencial. Además, posee un cuello mecánico que permite movilidad en longitud y latitud, sobre él se sitúan las cámaras citadas anteriormente.



(a) Base Pioneer

(b) Cámara iSight

(c) Láser SICK

Figura 3.2: Dispositivos hardware

3.2. Infraestructura software: La plataforma JDE.

En el grupo de robótica de la Universidad Rey Juan Carlos se ha desarrollado una plataforma software para la programación de los Pioneers: *JDE*[4] [Plaza, 2003]. La plataforma plantea las aplicaciones robóticas como un conjunto de esquemas que se ejecutan simultáneamente y en paralelo, que permiten realizar tareas sencillas y concretas. La ejecución simultánea de varios esquemas dan lugar a un comportamiento. Existen esquemas de distintos tipos: *de servicio* que se encargan de las comunicaciones recogiendo información de sensores y enviando órdenes a los actuadores, *perceptivos* que se encargan de producir y almacenar información sensorial o elaborada por otros esquemas, *de actuación* son los que toman decisiones sobre motores o la activación de esquemas de niveles inferiores a partir de la información que generan los perceptivos.

Los esquemas pueden organizarse en niveles estableciendo una jerarquía entre padre e hijo, siendo el esquema padre el que pueda activar y desactivar a los esquemas hijo.

[2] http://www.sick.es/es/productos/autoident/medicion_laser/interior/es.html
[3] http://www.apple.com/es/isight/
[4] http://gsyc.escet.urjc.es/jmplaza/software.html

La plataforma se ha implementado en una arquitectura software "jde.c" que ofrece una interfaz de variables de percepción y de actuación para acceder a sensores y actuadores del robot. Esta arquitectura software de la plataforma permite anclar estas variables a distintas fuentes (figura 3.3):

- robot real: las variables representan de forma directa a los sensores y actuadores del robot.
- simuladores: se utilizan principalmente dos simuladores: SRIsim de Shapira desarrollado en *SRI*[5] en conjunción con la biblioteca *ARIA*[6] y un simulador proporcionado por la plataforma Player/Stage[7] . Estos simuladores son capaces de simular perfectamente un robot Pioneer y cualquier entorno donde éste se encuentre. La utilización de ambos simuladores dependerá de la existencia de obstáculos dinámicos, es decir, imprevistos; por lo que se utilizará el simulador Player/Stage.
- servidores: existen dos servidores distintos que proporcionan acceso remoto a los sensores y actuadores. Detrás de los servidores puede haber un robot real o simuladores. Cada servidor se encarga de ciertos sensores y actuadores proporcionando la funcionalidad a los clientes a través de una *API de mensajes*. El servidor *Otos* proporciona el acceso a los motores, láser, infrarrojos y sónar. En cambio, el servidor *Oculo* se encarga de los sensores de imagen y los cuellos mecánicos que puedan existir en el robot.

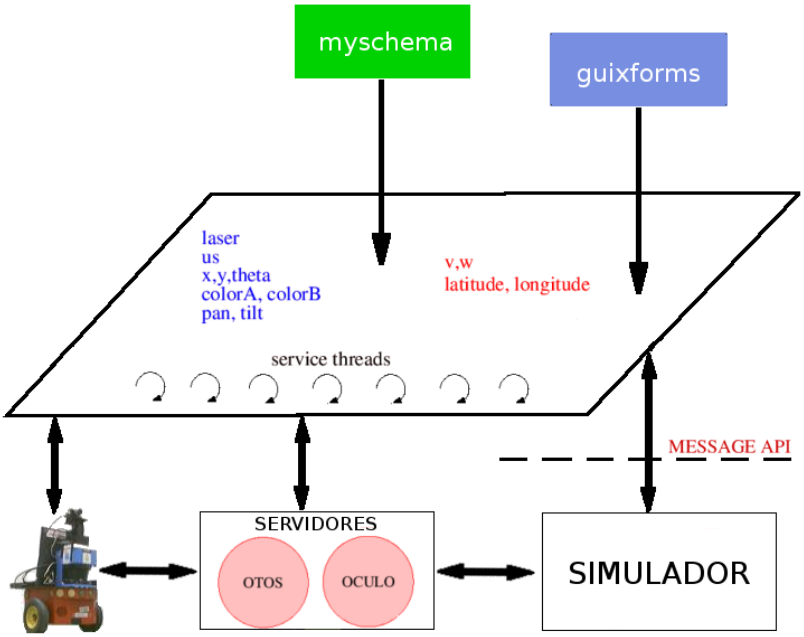


Figura 3.3: JDE Básico

[5]http://www.sri.com/
[6]http://www.activmedia.com
[7]http://playerstage.sourceforge.net

3.3. Bibliotecas auxiliares: Fuzzylib y Progeo

La biblioteca *Fuzzylib* aporta funcionalidad para el uso de controladores borrosos. La lógica borrosa es, en resumen, una lógica multievaluada que permite valores intermedios más allá de las evaluaciones binarias del tipo sí/no, verdadero/falso, negro/blanco. Las nociones como "más bien caliente", "poco frío" pueden formularse matemáticamente y ser procesadas. Así podemos obtener valores intermedios entre verdadero y falso, como "probable" o "muy poco probable". El controlador borroso define etiquetas para las variables de entrada y salida que maneja. Dicho controlador hace uso de un fichero de reglas que ofrecen un valor de salida según sea el valor de entrada. En este proyecto usaremos un controlador borroso para manejar las velocidades lineales y angulares del robot, que lo gobernarán en su navegación.

La biblioteca *PROGEO* (PROjective GEOmetry) nos permite conseguir información tridimensional, obteniendo un punto 3D a partir de un píxel, y viceversa. PROGEO se basa en el modelo de cámara *pin-hole*. De este modo, cualquier imagen captada por la cámara pasa a través del foco proyectándose en un plano y crea una imagen invertida respecto de la imagen real. Como se puede observar en la figura 3.4, la distancia entre el plano de imagen y el foco se corresponde con la llamada *distancia focal*. Para no trabajar con la imagen invertida, PROGEO trata con un plano que se encuentra entre el foco y la imagen real (plano de proyección), dicho plano se encuentra a una distancia respecto del foco igual a la distancia focal. Además, para poder extraer información correcta basándonos en el modelo *pin-hole*, tenemos que conocer los parámetros extrínsecos e intrínsecos de la cámara. Los parámetros intrínsecos son la distancia focal y el píxel por donde pasa el foco en el plano de proyección. Los parámetros extrínsecos, en cambio, son aquellos relacionados con la posición de la cámara, las coordenadas(x,y,z) con respecto de un sistema de referencia, el ángulo de la cámara respecto la horizontal y las coordenadas 3D hacia donde apunta el foco.

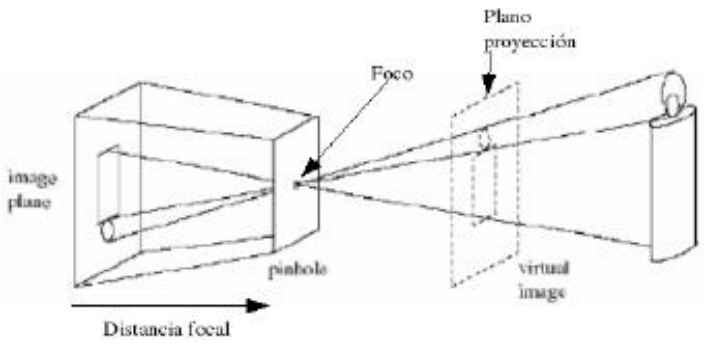


Figura 3.4: Modelo pin-hole

Capítulo 4
Descripción informática

En este capítulo describiremos la solución desarrollada para alcanzar el objetivo propuesto en el capítulo 2, implementada ésta sobre la plataforma descrita en el capítulo 3. Primero se dará una visión global de cómo se ha estructurado el comportamiento en forma de esquemas JDE, después se explicará más detalladamente cada uno de los esquemas.

4.1. Diseño general

El principal objetivo de este proyecto es dotar al robot Pioneer con la capacidad de identificar y localizar diversos objetos en la escena mientras navega autónomamente o ejecutando alguna orden. Esta capacidad le permitirá moverse en su entorno de modo más eficiente y seguro, es decir, tomar mejores decisiones gracias a que ha memorizado y entendido la escena que le rodea. Por tanto, también le será posible interactuar con los objetos sin la necesidad de detectarlos en todo momento con los sensores. Además, visualizaremos la escena tridimensionalmente en el interfaz gráfico, de modo que permita al usuario observarla desde el punto de vista del robot o desde cualquier otro punto.

El comportamiento buscado es generado mediante seis esquemas (figura 4.1). La identificación de cada objeto es implementada en un esquema diferente: *id-persona, id-congenere, id-puerta, id-pared*. La visualización tridimensional del entorno reconocido por el robot, es implementada en el esquema de servicio *gui3D*. Por último, la navegación local es implementada en el esquema *vff*. Más adelante, se explicará detalladamente cada uno de los esquemas citados; de momento, pasamos a explicar brevemente sus funcionalidades y relaciones, dando una visión panorámica de todo el sistema.

El esquema principal es *id-pared*, encargado de crear, localizar y memorizar segmentos rectos a partir de la odometría y de los puntos detectados por el láser. Este esquema establece una memoria de corto o medio plazo en la que inserta los segmentos que va construyendo a partir de los datos instantáneos del láser. Antes de insertar un nuevo segmento en memoria, se le otorga una salud o vida inicial que irá decreciendo

en función del tiempo y de la cantidad de movimiento realizado por el robot. También realiza una primera clasificación atendiendo a la longitud del segmento, pudiendo ser de "tipo desconocido" o de "tipo pared". Este esquema, además, lleva a cabo funciones de mantenimiento sobre dicha memoria de segmentos, como es el análisis de la coherencia y la decisión de fusionar un segmento nuevo con uno ya memorizado. Es fundamental que la memoria de segmentos sea compacta y coherente ya que es exportada, en forma de estructura de datos, a los demás esquemas (figura 4.1).

El resto de esquemas de identificación (*id-persona*, *id-congénere* y *id-puerta*) llevan a cabo una fusión sensorial para detectar correctamente, es decir, combinan y contrastan la información que les llega del sensor de visión (la cámara) y de la memoria de segmentos láser para realizar una detección suficientemente robusta. La información del láser ya les llega tratada y estructurada por el esquema *id-pared*, sin embargo, deben realizar el tratamiento de imágenes capturadas por la cámara. Dicho tratamiento consta de filtros de color o movimiento y su posterior segmentación (proceso de dividir la imagen en partes que corresponden a objetos). La relación en el espacio entre los segmentos láser y los segmentos de imagen se realiza a través de la biblioteca auxiliar PROGEO. Una vez tratada la imagen y contrastada con los segmentos láser, si se ha reconocido el objeto buscado (persona, congénere o puerta) entonces se clasificará al segmento como tal.

Después de la identificación, cada esquema realiza el mantenimiento de sus propios objetos reconocidos. Para mantener la coherencia de sus respectivas memorias de corto o medio plazo, asocian a cada objeto una salud y, además, actualizan las posiciones de dichos objetos si es necesario. En cuanto a la salud, es suficiente para que ésta no decrezca que siga siendo detectado el objeto por el láser, no siendo necesaria la detección por la cámara. Así como también, sólo se utiliza el láser para actualizar la localización del objeto en el caso que éste se desplace.

Para las identificaciones de personas, congéneres y puertas hemos utilizado, en lugar de técnicas clásicas de reconocimiento de patrones, una técnica basada en la etología, concretamente en el principio de suma heterogénea de subestímulos [Lorenz, 1978]. Este principio dice que si la suma de un conjunto de subestímulos supera cierto umbral entonces se producirá el desencadenamiento de un comportamiento. Con esta técnica se consigue una identificación robusta a partir de ciertos subestímulos sencillos sin la necesidad de una reconstrucción completa del mundo. Dichos subestímulos podrán provenir tanto del sensor láser como de las imágenes capturadas por la cámara, así como de la combinación de ambos.

El esquema que muestra la escena en tres dimensiones, *gui3D*, representa al propio robot en movimiento, los puntos del láser, así como los objetos que se han ido reconociendo y que están en la memoria de corto plazo que contiene la escena interiorizada. Para ello, necesita leer de la memoria de segmentos (ya clasificados por los esquemas de identificación) para conocer qué tipo de objetos son y su localización. De este modo, se generará una representación fiel del entorno. Además, para poder visualizar desde diferentes perspectivas la escena tridimensional, se ha creado una cámara virtual con la ayuda de PROGEO.

Por último, se podrá navegar de dos modos diferentes, uno autónomo y explorativo, el otro dirigiéndose hacia a algún objetivo seleccionado por el usuario. En cualquier caso, la navegación debe ser segura; por tanto, el esquema actuador *vff* implementa un algoritmo de navegación reactiva (*Virtual Force Field*) que permite evitar obstáculos tanto estáticos como dinámicos. Para ello, nos apoyamos en los puntos láser y, además, en los segmentos memorizados . De este modo, se puede obtener información de los obstáculos, tanto de los detectados en ese preciso momento por el láser como de los almacenados en la memoria de la escena.

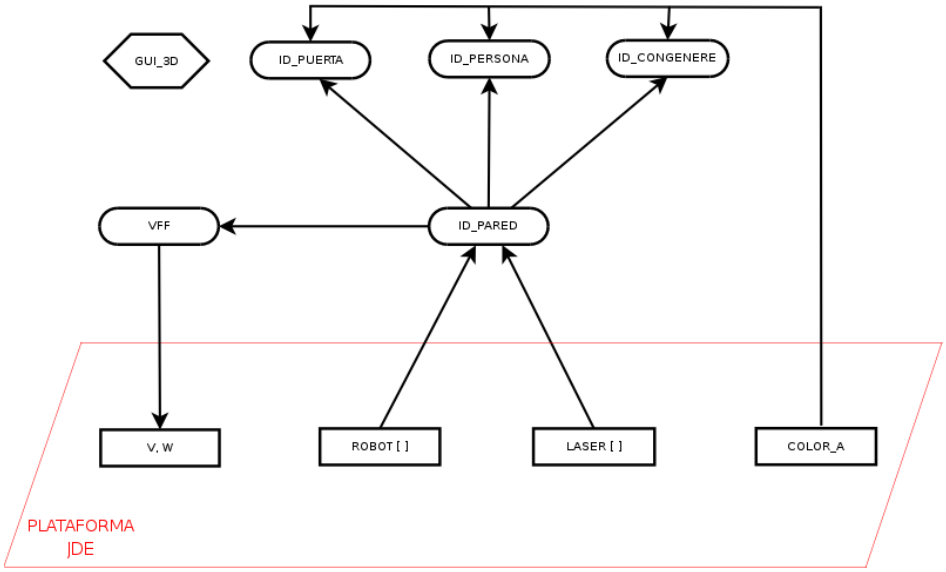


Figura 4.1: Diseño de la aplicación

En la figura 4.1 puede observarse la estructura diseñada, donde los símbolos representan:

Símbolos rectangulares: variables de JDE

- v y w : velocidad lineal y angular, respectivamente.
- *colorA* : imagen capturada por la cámara.

- *laser[]* : vector de las medidas de distancia tomadas por el láser.
- *robot[]* : vector de las medidas de localización tomadas por el sistema odométrico.

Símbolos circulares: Esquemas implementados

- *Id-Persona*, *Id-Congénere*, *Id-Puerta*, *Id-Pared*: esquemas de identificación.
- *VFF*: esquema actuador para la navegación local.

Símbolo hexagonal: esquema de servicio

- *gui-3D*: Interfaz gráfico desde donde se gobierna al robot y se visualizan los resultados.

Por último, las flechas representan las interrelaciones entre los esquemas y las variables de la plataforma.

4.2. Esquema de identificación de paredes

Este esquema es el encargado del tratamiento y organización de la información del sensor láser y la odometría, distribuyendo los datos obtenidos (segmentos rectos) y ya estructurados al resto de esquemas. También es el responsable de clasificar como paredes aquellos objetos detectados por el láser con un tamaño suficientemente grande, dejando el costoso procesamiento de imágenes para detecciones más complejas.

Lo primero que realiza este esquema es tomar como entrada los puntos determinados por el láser para crear un conjunto de segmentos bidimensionales. Cada segmento obtenido es clasificado como pared o desconocido en función de su longitud e insertado en una estructura, a la cual denominaremos memoria de segmentos. Por último, se lleva a cabo un mantenimiento sobre dicha memoria para mantenerla coherente, compacta y actualizada. Una vez procesada y organizada la información del láser, este esquema devuelve los segmentos láser memorizados en una estructura global en forma de lista.

Para cada segmento memorizado se guardará, además de una variable booleana que nos indica si su salud está agotada, la localización (x, y) de sus puntos inicial y final, su ecuación de la recta, su salud y su tipo. Inicialmente, el tipo será “desconocido” o “pared” hasta que alguno de los esquemas de identificación lo clasifiquen como “puerta”, “congénere” o “persona” cuando sea detectado como tal (figura 4.2).

```
ESTRUCTURA TIPO_SEGMENTO_LASER {
    T_PUNTO   p_inicial   Punto (x,y) inicial del segmento
    T_PUNTO   p_final     Punto (x,y) final del segmento
    T_RECTA   recta       Ecuacion de la recta ( y = ax + b )
    OBJETO    tipo        Objeto : [ Persona, Congenere, Puerta, Pared ]
    ENTERO    salud       Salud del segmento: 0 ... SALUD_inicial
    BOOLEANO  activo      Permanece en memoria o ha sido olvidado
}
```

Figura 4.2: Estructura de datos para un segmento del láser

4.2.1. Segmentación de los puntos láser

Para segmentar el láser se toman los 180 puntos diferentes (uno por cada grado respecto del robot) que éste nos ofrece. El algoritmo está basado en la idea intuitiva de ir comprobando punto a punto para ir construyendo los segmentos, apoyándose en las ecuaciones de mínimos cuadrados:
Para todo $P_i(x, y)$, sea $R_s : y = ax + b$

$$a = \frac{\sum_{i=1}^{N}(xy) - \frac{1}{N}\sum_{i=1}^{N} x \sum_{i=1}^{N} y}{\sum_{i=1}^{N} x^2 - \frac{1}{N}(\sum_{i=1}^{N} x)^2} \tag{4.1}$$

$$b = \frac{\sum_{i=1}^{N} y \sum_{i=1}^{N} x^2 - \sum_{i=1}^{N} x \sum_{i=1}^{N}(xy)}{N\sum_{i=1}^{N} x^2 - (\sum_{i=1}^{N} x)^2} \tag{4.2}$$

El algoritmo comienza comprobando que los puntos 0 y 1 estén a una distancia menor que cierto umbral X (si no fuera así pasaría a comprobar los puntos 1 y 2), con estos dos puntos se crea una recta (punto-pendiente). Ahora se comprueba el punto 2, el cual debe cumplir dos condiciones, es decir, P_i pertenece a un segmento si y sólo si:

1. $d(P_i, P_{i-1}) < X$. Está a una distancia, respecto del último punto, inferior a un umbral X.

2. $d(P_i, R_s) < Y$. Está a una distancia, respecto de la recta, inferior a un umbral Y.

Si no cumpliera alguna condición, se obtiene el segmento formado por los puntos 0 y 1, y se vuelve al principio, comprobando ahora los puntos 1 y 2. Si cumple ambas condiciones, se crea una nueva recta teniendo en cuenta, además de los puntos 0 y 1, el punto 2. Dicha recta es ajustada por mínimos cuadrados (ecuaciones 4.1 y 4.2). Ahora se pasa a comprobar el punto siguiente (punto 3) que deberá cumplir las condiciones anteriores. Continúa de este modo hasta que un punto i, no cumpla alguna de las condiciones y, entonces, obtenemos el segmento formado por los puntos inicial (punto

0, en este caso) e i-1, y se vuelve al principio, comprobando ahora los puntos i-1 e i. El algoritmo finaliza cuando haya comprobado los 180 puntos y, por tanto, con una complejidad en tiempo lineal.

En la figura 4.3 se muestra un ejemplo paso a paso de aplicación del algoritmo para cinco puntos, obteniéndose, en este caso, dos segmentos.

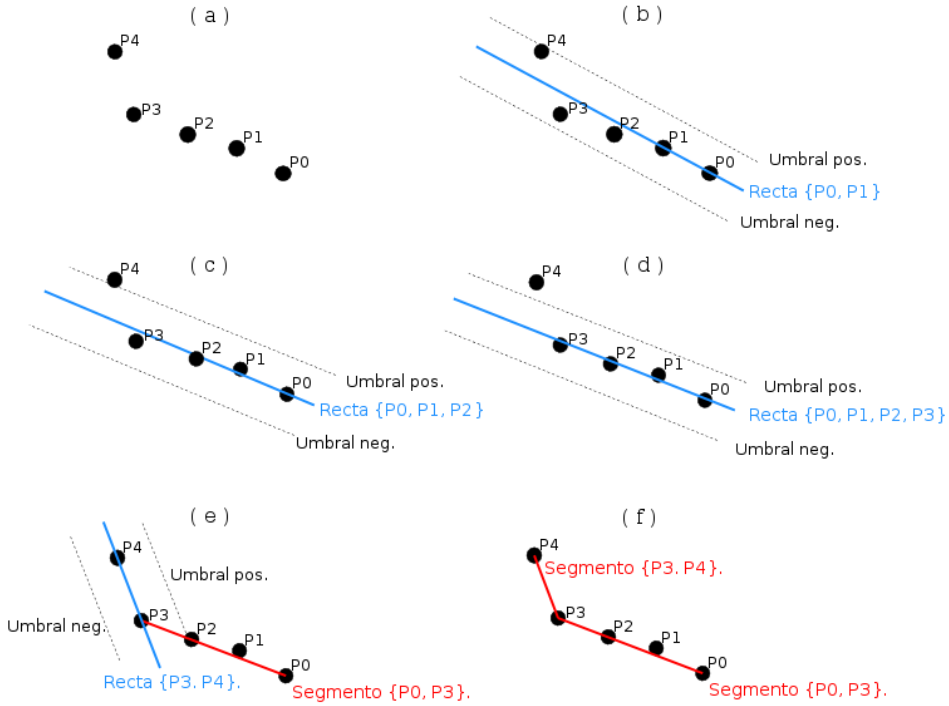
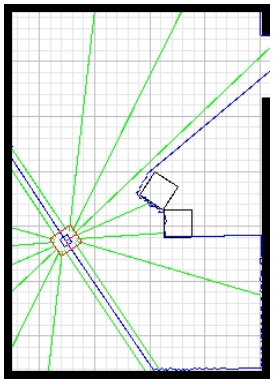
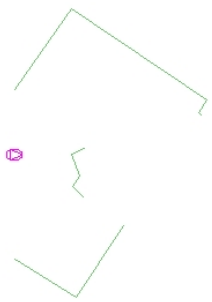


Figura 4.3: Segmentación basada en mínimos cuadrados

En la figura 4.4 se muestra el resultado final de la ejecución del algoritmo implementado para todos los puntos del láser (180) en un entorno simulado.



(a) Escena simulada



(b) Escena segmentada

Figura 4.4: Segmentación del láser

4.2.2. Memorización y mantenimiento de los segmentos láser

En cuanto al mantenimiento realizado, al insertar los segmentos creados en la memoria, se les otorgará una salud inicial que irá decreciendo proporcionalmente al tiempo que lleva sin detectarse por los sensores (no podemos estar seguros de que siga allí, sobre todo, si es un objeto móvil) y a la cantidad de movimiento que ha efectuado el robot (se ha podido acumular un excesivo ruido en la odometría y nos hace dudar de la localización exacta). Esto garantiza que termina saliendo de la memoria si pasado cierto tiempo no se vuelve a percibir, es decir, acaba olvidándose.

Además, se podrá ver drásticamente reducida la salud del segmento cuando sea incoherente con las medidas instantáneas del láser (figura 4.5). Es decir, será penalizado si el segmento hipotético desde el centro del robot a un punto aleatorio del láser(t) lo corta con una cierta distancia umbral (es muy fácil que esto ocurra con objetos dinámicos).

Justo antes de insertar en la memoria los nuevos segmentos, estudiaremos la coherencia citada para cada segmento ya memorizado y, también, comprobaremos si se ha agotado la salud para desactivarlo en caso afirmativo.

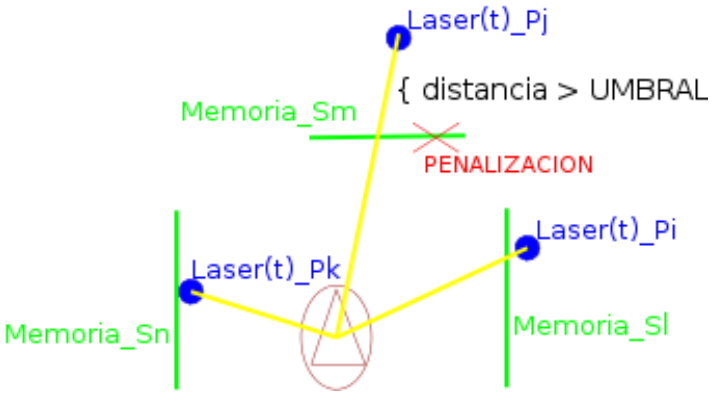


Figura 4.5: Análisis de coherencia en memoria segmentos

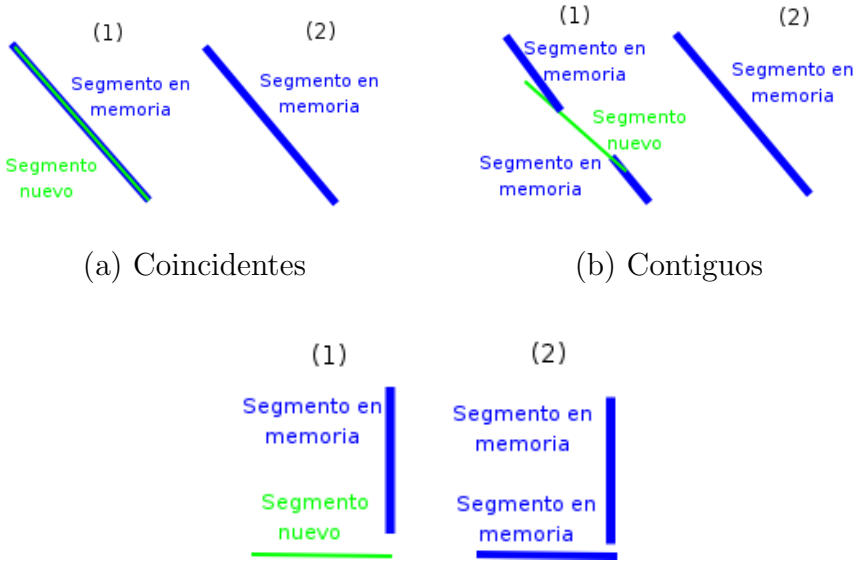
Una vez creados los nuevos segmentos a partir del láser son insertados en memoria (figura 4.6). Para realizarlo correctamente se contrastan con los segmentos memorizados, con tres posibles alternativas:

1. Es coincidente con algún segmento de la memoria, entonces aumentará la salud del memorizado (figura 4.6(a)).

2. Es contiguo a algún segmento de la memoria, entonces aumentará la salud del memorizado y se fusionarán (figura 4.6(b)).

3. Ninguna de las anteriores, entonces se insertará en memoria con la salud inicializada (figura 4.6(c)).

Dos segmentos son considerados equivalentes si tienen extremos iguales; y son considerados contiguos si su pendiente es muy similar y, además, se cortan en un punto o tienen parte común.



(a) Coincidentes (b) Contiguos

(c) Nuevo

Figura 4.6: Inserción de nuevos segmentos en memoria

Con el fin de evitar excesivas comparaciones entre los segmentos nuevos y los memorizados, se realizarán éstas sólo si ambos poseen partes comunes en el eje X y en el eje Y de coordenadas (figura: 4.7), con un cierto umbral de tolerancia. De este modo aceleramos el funcionamiento.

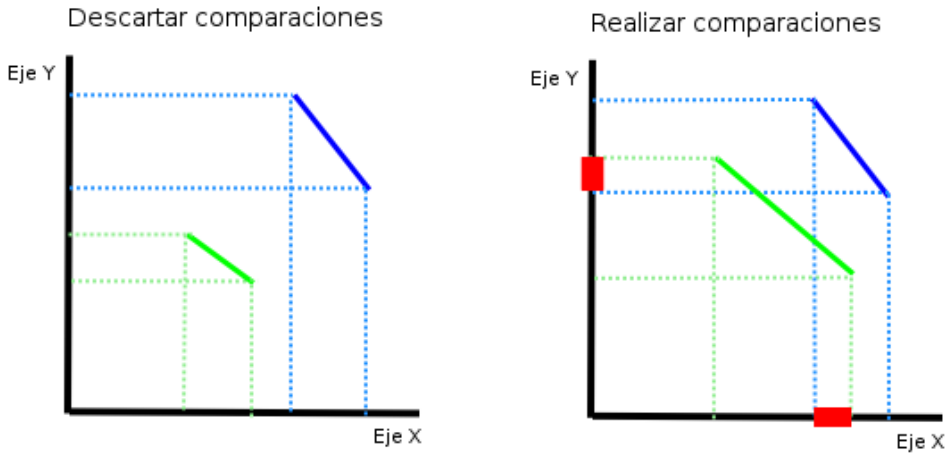
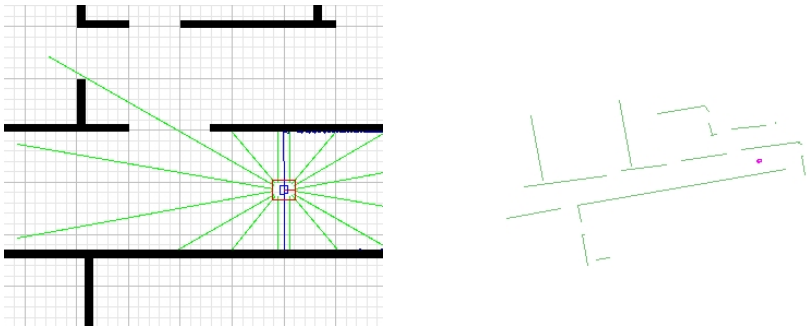


Figura 4.7: Descartar o realizar comparaciones entre segmentos

En la figura 4.8 se muestra el resultado final de la ejecución, en un entorno simulado, de la creación y mantenimiento de segmentos.



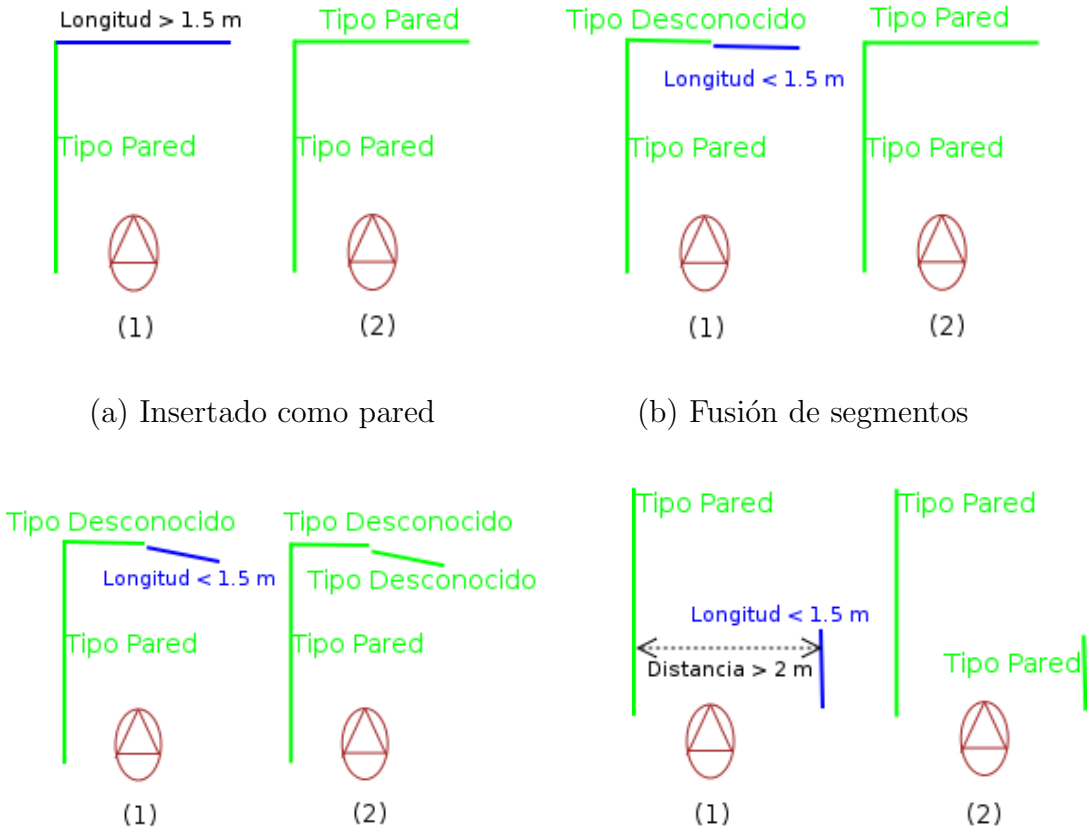
(a) Escena simulada (b) Escena memorizada

Figura 4.8: Memorización de los segmentos creados a partir del láser

4.2.3. Detección y mantenimiento de paredes

El factor necesario y suficiente para decidir si un objeto es una pared es el tamaño del segmento creado a partir del láser. Si éste tiene una longitud mayor de 1.5 metros, es directamente insertado en memoria como pared (figura 4.9(a)). Cuando se crea un segmento de longitud inferior a 1,5 metros es insertado como objeto desconocido. En el caso que se decida fusionar con otro segmento en memoria por sus condiciones similares, se volverá a tener en cuenta la longitud, ahora de la unión de ambos, para estimar si es pared o no (figura 4.9(b)).

Es importante que el mayor número posibles de segmentos en memoria estén clasificados, es decir, que se conozca qué clase de objetos representan. Esto permitirá tomar mejores decisiones a la hora de fusionarlos si fuera necesario o,simplemente, a la hora de navegar y/o actuar con los objetos identificados. Sin embargo, puede darse la situación de tener distintos segmentos no identificados, de longitud menor de metro y medio, que puedan ser una pared, ya sea porque aún no se haya detectado con el láser todo su tamaño real o porque el margen de error cometido en la toma de datos no le permita fusionarse con otro por su condición de objeto desconocido (figura 4.9(c)). Por esto, es importante prever si puede ser pared. Para contemplar este caso, comprobaremos si existe otro segmento paralelo a él, con una distancia entre ambos de al menos dos metros, y con una longitud igual o mayor que la suya para determinar un posible pasillo compuesto por ambos segmentos y, de este modo, clasificarlo o clasificarlos como pared en la memoria (figura 4.9(d)).



(a) Insertado como pared

(b) Fusión de segmentos

(c) Segmentos desconocidos

(d) Segmentos paralelos

Figura 4.9: Casos probables en la identificación de paredes

En la figura 4.9 se muestran, en dos momentos diferentes, los casos expuestos de posibles fusiones entre segmentos y sus correspondientes clasificaciones. Las líneas azules representan los segmentos creados actuales y las líneas verdes los segmentos en memoria. También se explicitiza su tipo, anterior y posterior a la ejecución de un ciclo de este esquema (1 y 2).

En cuanto a la salud inicial otorgada a los segmentos clasificados como pared, ésta es alta y decrece de manera muy lenta en función del tiempo sin detectarse por el sensor láser. Lo hemos resuelto así porque el láser ofrece una gran fiabilidad y, además, la pared es un objeto estático. Sin embargo, el sistema odométrico no es tan fiable y puede conducir a error cuando éste ha acumulado excesivo ruido, por tanto, la salud decrece de manera mucho más rápida en función de la cantidad de movimiento que ha realizado el robot (diferencial entre posición actual y posición de la última medida). De este modo, se olvidan los tramos más antiguos de la pared (figura 4.10) que puedan aparecer desviados debido a los errores en la odometría. Además, como se comentó en el apartado 4.2.2 y vimos en la figura 4.5, se reduce drásticamente su salud en caso de incoherencias.

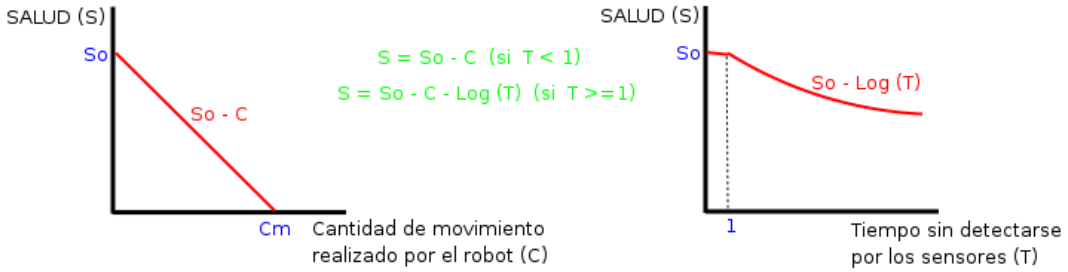


Figura 4.10: Salud de un segmento clasificado como pared

4.3. Esquema de identificación de puertas

La idea intuitiva para identificar puertas, ya sea abierta o cerrada, es que se encuentre junto a una pared y sea de un color determinado. Nuestro esquema para identificar paredes ya se ha encargado de clasificar los segmentos en memoria como tal, por tanto, sólo es necesario contrastar el segmento no identificado, muy próximo a una pared, con la imagen captada por la cámara para decidir si es realmente una puerta.

Lo primero que realiza este esquema es tomar como entrada de datos la imagen capturada por la cámara para realizar un filtro por el color marrón rojizo (característico de las puertas del Departamental II) y, posteriormente, segmentar la imagen filtrada (algoritmo explicado en el apartado 4.4.1). Si obtenemos un segmento más alto que ancho de dicho color (condición necesaria), se contrasta con los segmentos láser memorizados para decidir si es realmente una puerta y clasificarla como tal en memoria.

4.3.1. Filtro de color

Un filtro es una herramienta diseñada para tomar una imagen de entrada, aplicar un algoritmo matemático, y devolver la imagen modificada. Nuestro algoritmo decidirá para cada píxel si es similar en tono, saturación e intensidad al color por el que queremos filtrar; en caso positivo, podemos dejarlo como está (o resaltarlo) y, en caso negativo, transformarlo a negro. En la figura 4.11 se muestra un ejemplo de filtrado por el color marrón rojizo característico de las puertas del Departamental II.

Figura 4.11: (a) imagen real y (b) imagen filtrada por color

La plataforma JDE ofrece la imagen en formato RGB, donde cada píxel ocupa tres bytes, uno por cada color(rojo, verde y azul). Debido a que los filtros de color en imágenes en este formato no son demasiado robustos, sobre todo cuando son expuestos a diferentes condiciones de luz, se ha optado por realizar los filtros en imágenes con formato HSI. En éste, cada píxel está formado por tres componentes: matiz (H), saturación (S) e intensidad (I), lo que le da mayor robustez ante condiciones límites de luz. Geométricamente es representado como un cono (imagen 4.12), donde se puede observar la situación de los colores en él.

Siempre será necesario realizar la transformación de RGB a HSI antes de filtrar. Para llevarla a cabo se utilizan las siguientes ecuaciones:

$$H = \frac{\frac{-1}{2}(R-G)+(R-G)}{2\sqrt{(R-G)^2+(R-B)(G-B)}} \tag{4.3}$$

$$S = 1 + \frac{3}{R+G+B} * min(R,G,B) \tag{4.4}$$

$$I = \frac{R+G+B}{3} \tag{4.5}$$

La intensidad (I) y saturación (S) están normalizadas al rango $(0,1)$, y el tono (H) a $(-\Pi,\Pi)$.La función $min(R,G,B)$, utilizada para calcular la saturación (S), devuelve el menor valor de entre los tres. Por último, es necesario destacar que existen ciertos casos especiales donde no se aplican las fórmulas 4.3, 4.4 y 4.5. Por ejemplo, si $R+G+B=0$ entonces se asigna directamente el valor $S=1$; y si $(R-G)(R-G)+(R-B)(G-B) <= 0$ entonces se asigna el valor $H=-1$.

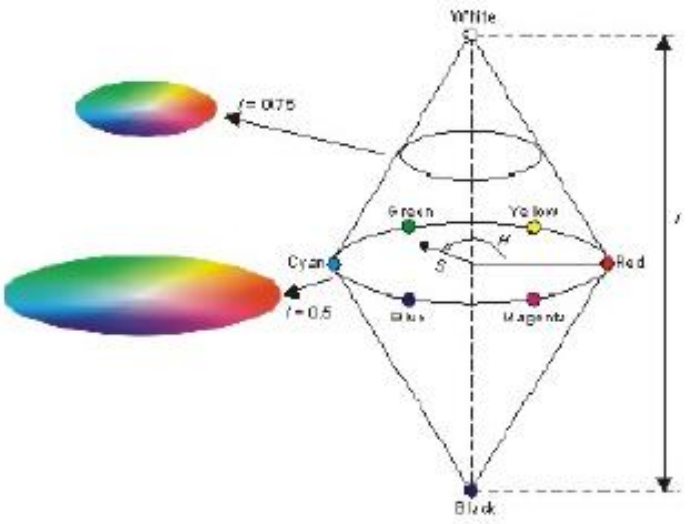


Figura 4.12: Representación geométrica del formato HSI

4.3.2. Detección y mantenimiento de puertas

El subestímulo que desencadenará la búsqueda es la obtención de un segmento rectangular, más alto que ancho, y de color marrón rojizo en la imagen. Entre el píxel central del segmento marrón y la cámara se proyecta un rayo en el espacio que se prolonga hasta que corte a un segmento producido por el láser (figura 4.13), obteniéndose la posición en el mundo del píxel que representa al segmento marrón. Si el segmento láser con el que corta el rayo tiene una longitud inferior a un metro y está muy próximo a un segmento clasificado como pared entonces se clasifica al segmento como puerta.

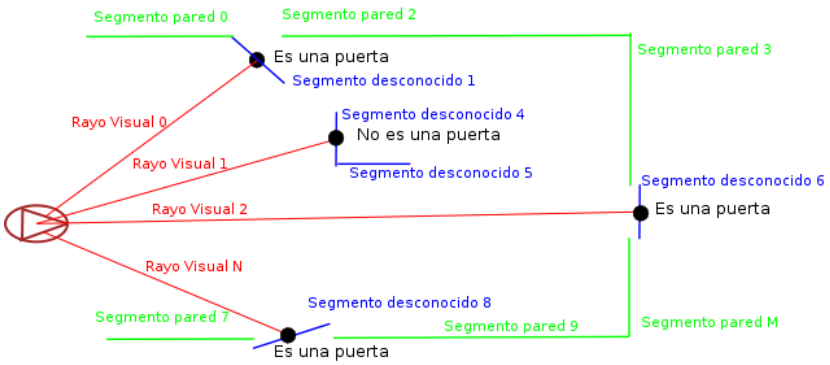


Figura 4.13: Fusión sensorial de visión y láser para identificar una puerta

El mayor problema a la hora de identificar una puerta es que ésta se encuentre práctica o completamente cerrada, ya que en el momento de insertar el segmento (candidato a puerta) en la memoria pueda ser fusionado con un segmento mayor y ser identificado como pared. Para evitarlo es necesario ser cuidadosos a la hora de fusionar en memoria un segmento de tamaño específico (mayor que 0.5 metros y menor que 1 metro) con uno igual o mayor. En este caso, al segmentar y memorizar el láser se exige una distancia y diferencia de pendientes prácticamente nulas para fusionar dos segmentos de estas características.

Para mantener y actualizar la salud del segmento identificado como puerta sólo es necesario detectarlo con el sensor láser, no siendo necesario, una vez identificado, que siga siendo detectado por la cámara. El tratamiento dado a la salud del segmento de tipo puerta es similar al dado a los segmentos clasificados como pared (apartado 4.2.3), debido a la fuerte dependencia de las puertas con las paredes ya que no se permite la existencia de las primeras sin las segundas. Sin embargo, sí se realiza una actualización de la posición en caso de haberse modificado (figura 4.14), ya sea porque ha sido abierta o cerrada después de la última detección.

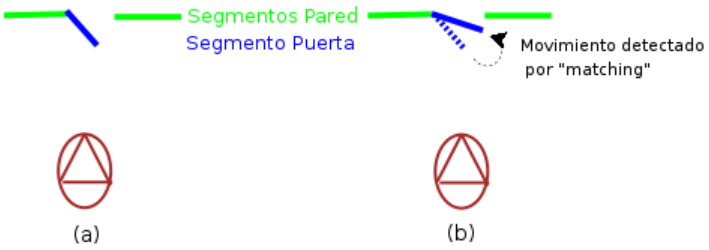


Figura 4.14: Actualización de posición en las puertas

4.4. Esquema de identificación de congéneres

Este esquema ha sido implementado tomando como base el esquema para identificar congéneres de [Hidalgo, 2006]. Se han añadido algunas mejoras y funcionalidades: una vez reconocido ya no es necesario el procesamiento de imágenes para localizarle; ya que se podrá actualizar su posición, en caso de movimiento, simplemente con el láser. Además, en caso de que tampoco sea detectado por el láser, se seguirá conociendo su última posición donde fue identificado hasta que se agote la vida del segmento láser que lo representa.

Lo primero que realiza este esquema es tomar como entrada de datos la imagen capturada por la cámara para realizar un filtro por el color rojo (característico de la base del congénere) y, posteriormente, segmentar la imagen filtrada. Si obtenemos un segmento más ancho que alto (condición necesaria), se repetirá el proceso para los colores azul y negro. Si se ha obtenido el segmento rojo y uno azul o negro, se contrasta la información con los segmentos láser memorizados para decidir si es realmente un congénere y clasificarlo como tal en memoria.

4.4.1. Segmentación de la imagen

Esta segmentación es un proceso que consiste en dividir una imagen digital en regiones homogéneas con respecto a una o más características con el fin de facilitar su posterior análisis y detección de objetos. La técnica que hemos empleado para segmentar está basada en histogramas.

Un histograma es un diagrama de barras que contiene frecuencias relativas a algunas características, en nuestro caso el color. Se tienen dos histogramas (figura 4.15), uno para las columnas y otro para las filas. En el histograma de las columnas, éstas son representadas en el eje de abscisas, mientras que en el eje de ordenadas se representa la frecuencia con la que aparece un píxel de un determinado color. El histograma de las filas es igual, con la diferencia que en el eje de abscisas se representan las filas.

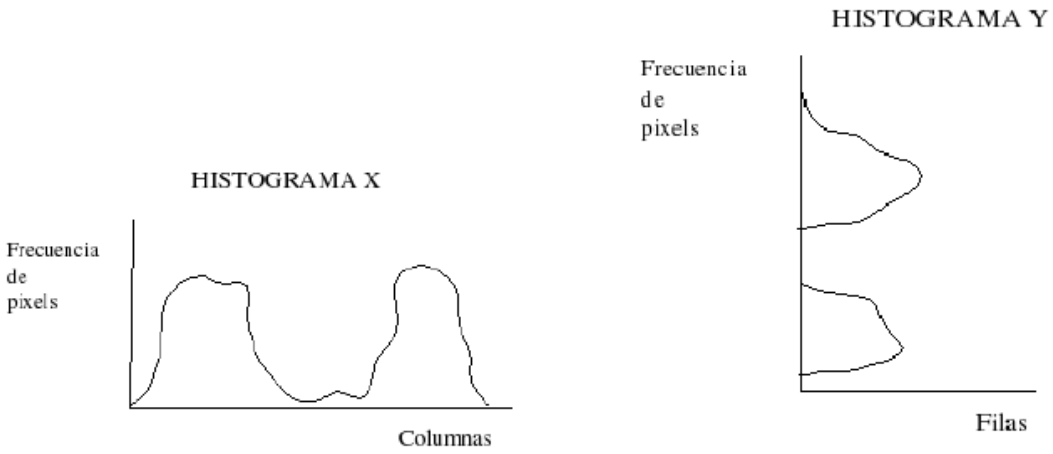


Figura 4.15: Histograma para las filas e histograma para las columnas

Para realizar la segmentación nos hemos basado en la técnica del segundo umbral [Martín, 2002] (figura: 4.16). Esta técnica pasa dos filtros (uno por cada umbral) a cada histograma, con el primero se eliminan puntos ruidosos y, si se pasa el segundo umbral, partiendo de los puntos que superaron el primero, entonces se crea el segmento. En la figura 4.16 se puede observar como tres zonas pasan el primer umbral, sin embargo, una de ellas no pasa el segundo, por lo que no dará lugar a ningún segmento.

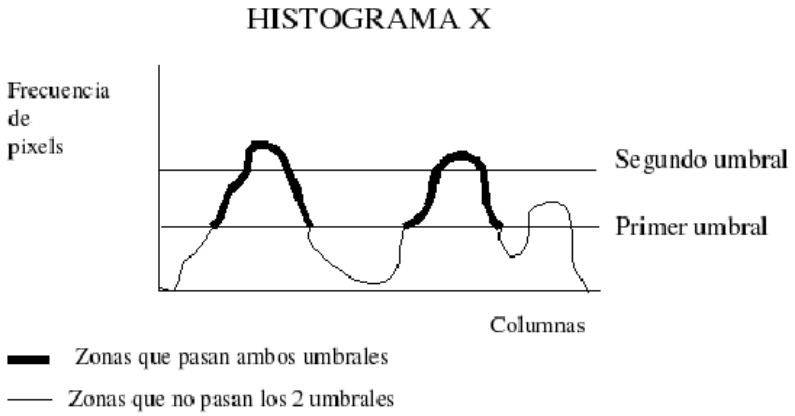
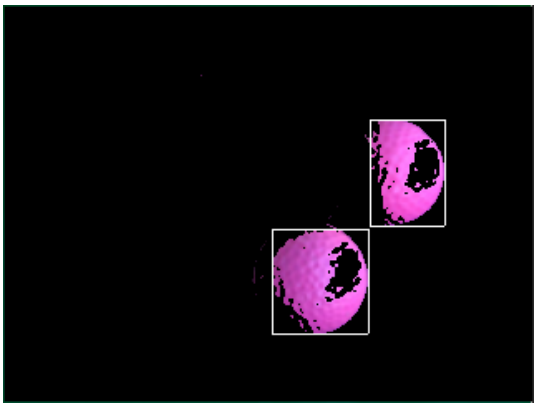


Figura 4.16: Histograma con doble umbral

Una mejora realizada a este algoritmo ha sido aplicarle una segmentación recursiva [Hidalgo, 2006] a un segmento que no supere un porcentaje de píxeles de color respecto de su tamaño total. Esta mejora permite ajustar el contorno de los objetos de manera más robusta y tan eficiente como la técnica del doble umbral básica. En la figura 4.17 se muestra un ejemplo de la aplicación de dicha técnica, ajustando al máximo los contornos (rectángulos blancos) de las pelotas de golf rosas con segmentación recursiva.

(a) Imagen real



(b) Imagen segmentada

Figura 4.17: Imagen filtrada por color rosa y aplicación de la segmentación recursiva

Para obtener los límites de cada zona, almacenamos el comienzo y el final de las distintas zonas obtenidas al pasar el segundo umbral en el histograma de las columnas, del mismo modo lo haremos para el histograma de las filas. Usando conjuntamente las zonas que superan ambos umbrales de las filas y de las columnas, obtendremos unos segmentos rectangulares de color en la imagen. Estos segmentos no son definitivos ya que, como vemos en la figura 4.18 se pueden llegar a crear falsos segmentos por lo que hay que comprobar que superan un número mínimo de píxeles.

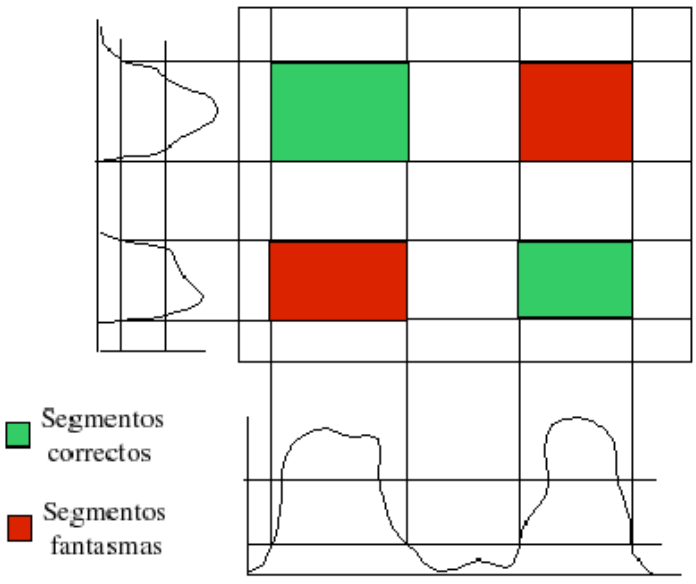
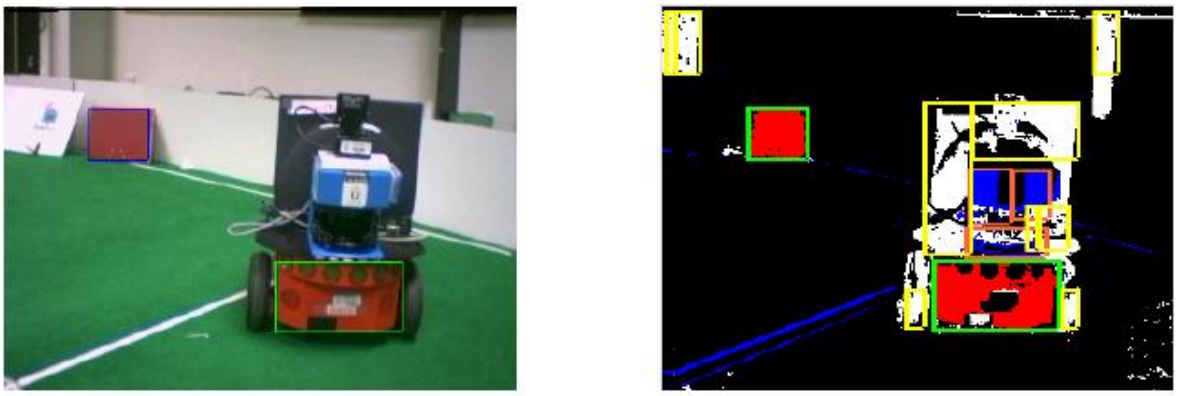


Figura 4.18: Ventanas en la segmentación de imagen

Una vez realizado todo este proceso, obtendremos como resultado los segmentos a partir de los colores que hemos querido filtrar. En la figura 4.19 se ha segmentado una imagen filtrada por los colores rojo, azul y negro.



(a) (b)

Figura 4.19: Ventanas en la segmentación de imagen

4.4.2. Detección y mantenimiento de congéneres

Para reconocer a un congénere, se requiere de un subestímulo imprescindible: un segmento en la imagen de color rojo. Éste es un invariante del robot congénere independiente de la posición desde donde sea observado. En caso de producirse este subestímulo, se desencadenará la búsqueda de otros, estos son:

1. El segmento rojo es más ancho que alto. Este subestímulo se corresponde con la base del robot, la cual es más ancha que alta desde todos los ángulos que se pueda mirar al robot. Esta característica incrementa la suma de los subestímulos en un punto.

2. Encima del segmento rojo se encuentra uno azul. Este subestímulo se corresponde con el sensor láser que está encima de la base del robot. Para tener en cuenta esta característica, se exige que el segmento azul sea más estrecho que el rojo,además, estará entre la columna comienzo y final del segmento rojo y, por último, el segmento azul estará en un rango de diez píxeles por encima del rojo. Esta característica produce un incremento en la suma de subestímulos de un punto.

3. Encima de segmento rojo se encuentra uno negro. Este estímulo se corresponde con la parte negra del sensor láser. Se le exige condiciones análogas que al segmento azul y produce un incremento igual, un punto.

4. El segmento rojo da sensación de profundidad. Para cada segmento rojo que tenga encima uno azul y/o uno negro, se obtiene, desde el píxel central del segmento, un rayo en el espacio 3D que se prolonga hasta intersecar con algún segmento del láser. Si este segmento láser cumple las características de longitud necesarias, se producirá un incremento de cuatro puntos.

Para determinar los segmentos rojo, azul y negro, se realiza un filtro por cada color y su posterior segmentación. Si uno o varios de los posibles segmentos rojos cumple su condición de forma y, además, encima de él se sitúa uno azul y/o uno negro con sus características correspondientes entonces entre el píxel central del segmento rojo y la cámara se proyecta un rayo visual virtual, de forma análoga a la identificación de puertas. Cuando dicho rayo corte a un segmento producido por el láser (figura 4.20), se obtiene la posición en el mundo del píxel que representa al segmento rojo. Para que este subestímulo se tome como válido, el segmento láser con el que corta el rayo visual debe ser inferior a 600 milímetros (ancho del robot) y, además, este segmento no debe tener cerca ningún otro segmento láser de mayor longitud que él (sensación de profundidad).

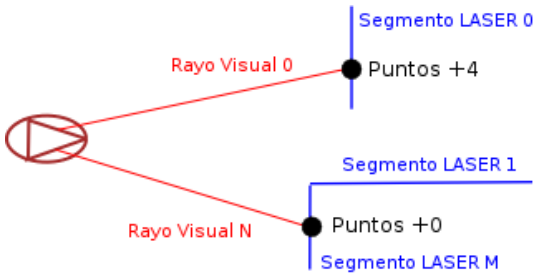


Figura 4.20: Correlación visión y láser para identificar un congénere

Si la puntuación total obtenida es igual o mayor a seis puntos entonces se dará como válida la identificación. Por tanto, se clasificará al segmento láser en memoria, con el que corta el rayo visual, como *Tipo Congénere*.

Al igual que en el caso del mantenimiento de puertas (apartado 4.3.2), actualizaremos la posición del congénere en el caso que ésta hubiera cambiado, así como también una vez identificado el segmento como congénere ya no es necesaria la detección en la imagen capturada por la cámara, tan sólo se necesita la detección del láser para mantener su posición y salud. Sin embargo, en este caso la salud decrece rápido en función del tiempo que lleva sin detectarse por el sensor láser, debido al potencial dinamismo del congénere, y muy lento en función del movimiento realizado por nuestro robot ya que los errores en la odometría apenas afectan, debido al pequeño tamaño del segmento y que no será fusionado con otros (figura 4.21).

Además, como se ha comentado en el apartado 4.2.2, todo segmento verá reducida drásticamente su salud en caso de existir incoherencias.

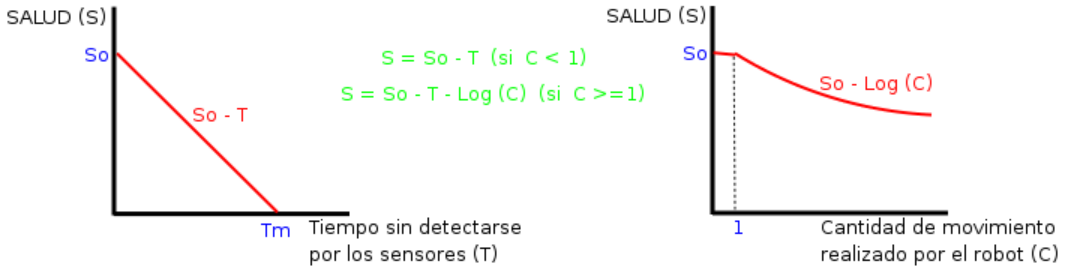


Figura 4.21: Salud de un segmento clasificado como congénere

4.5. Esquema de identificación de personas

Para identificar personas nos basamos en dos aspectos básicos: Es un objeto móvil (que no ha sido identificado como congénere) y, además, al segmentar las piernas a partir del láser tienen una forma característica.

Lo primero que haremos es buscar, en el conjunto de segmentos memorizados, un subconjunto de segmentos que cumplan ciertas características y que puedan ser agrupados de dos en dos (ambas piernas). Por tanto, obtendremos un conjunto de tuplas de segmentos que denominaremos conjunto de candidatos.

Para obtener el conjunto de candidatos, primero buscaremos en la memoria un segmento de pequeño tamaño que aún no haya sido identificado por ningún esquema y que no pertenezca a los candidatos; si se encuentra alguno que cumpla los requisitos entonces se buscará otro similar que complete la tupla. Para que sea considerado similar, es necesario que cumpla, además de los requisitos anteriores, una serie de condiciones como es la similitud en tamaño, en pendiente y en localización, con una salvedad: debe existir una diferencia en sus coordenadas Y desde el sistema de referencia solidario al robot (figura 4.22).

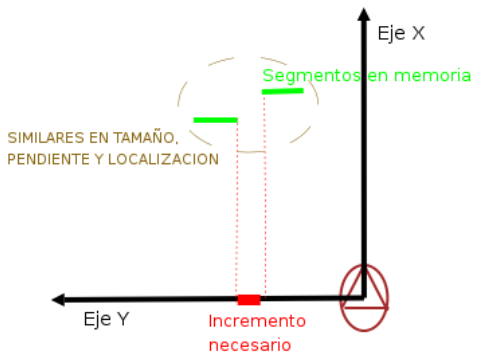


Figura 4.22: Búsqueda de posibles piernas en la memoria de segmentos

Una vez creado el conjunto de tuplas, es necesario estudiar si realizan algún tipo de movimiento para identificarlos como personas y actualizar su posición en memoria. Este estudio se realiza a través de dos estímulos: Movimiento en los segmentos candidatos o movimiento en la imagen.

4.5.1. Filtro de movimiento

El objetivo es detectar movimiento es una secuencia de imágenes para estimar el contorno y la posición del objeto móvil. El algoritmo utilizado para ello estudia la diferencia, píxel a píxel, de la secuencia de imágenes tomadas de dos en dos, es decir, imagen actual e imagen anterior.

Figura 4.23: Imágenes de una secuencia con dos personas andando

Figura 4.24: Movimiento detectado en la secuencia de imágenes de la figura 4.23

Partiendo de la secuencia de imágenes de la figura 4.23 se detecta la existencia de movimiento en la escena. En la figura 4.24 se puede observar el resultado del procesamiento que realiza el algoritmo citado. En esta caso, el píxel en donde haya existido una diferencia de color (con respecto a la imagen anterior) mayor que un cierto umbral es convertido a blanco, mientras que si no supera dicho umbral es convertido a negro. Ahora, sólo nos queda segmentar, como vimos en el apartado 4.4.1, para obtener el contorno y posición del objeto móvil.

Para detectar movimiento en la escena, es necesario que la cámara esté estática, es decir, cuando se quiera realizar un filtro de movimiento el robot deberá permanecer inmóvil; por tanto, será necesario sincronizar el esquema actuador *vff* con los esquemas de identificación de objetos que llamen a esta función.

4.5.2. Detección y mantenimiento de personas

Si este esquema detecta movimiento en la imagen (siempre y cuando el robot permanezca estático), a través de un filtro sobre la secuencia de imágenes capturadas, entonces se realiza una segmentación en la imagen filtrada para obtener la posición en el mundo donde haya existido tal movimiento (la posición se obtiene de modo análogo a la identificación del congénere o puerta). Ahora, sólo es necesario comprobar si la posición obtenida es la de alguno de los segmentos candidatos (figura 4.25).

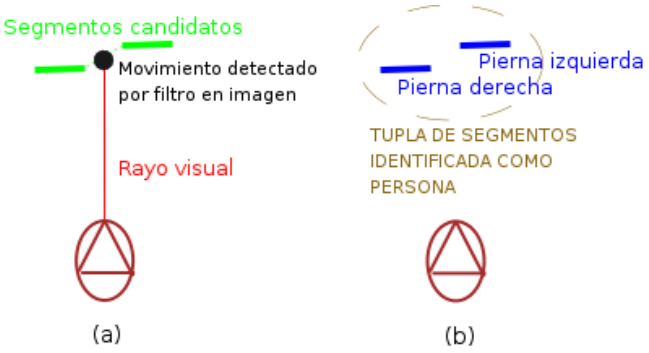


Figura 4.25: Movimiento en imagen para detectar personas

El otro modo de identificar como persona a una tupla de segmentos candidatos es realizando una correspondencia sobre los candidatos y los nuevos segmentos memorizados para comprobar si se han movido; además, necesitamos saber cuánto ha cambiado su posición para poder actualizarla (figura 4.26).

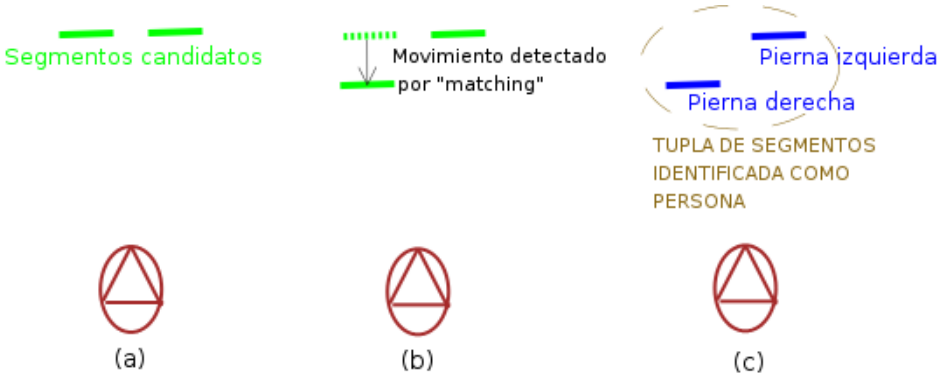


Figura 4.26: Movimiento en los segmentos para detectar personas

Una vez reconocida alguna tupla de segmentos candidatos como persona, este esquema clasificará en la memoria ambos segmentos identificados como pierna izquierda o derecha. En cuanto a la salud de los segmentos reconocidos como personas, se trata de igual forma que en el caso del congénere ya que ambos son de pequeño tamaño y dinámicos.

4.6. Esquema de navegación VFF

Este esquema es el encargado de comandar los movimientos del robot, por tanto, es un esquema de actuación. Se desarrolla la navegación local que implementa la técnica de Campos de Fuerzas Virtuales o *Virtual Force Fields* (VFF) [Borenstein, 1989]. Los movimientos realizados por el robot no deben ser bruscos, para ello, se ha añadido a la técnica VFF un controlador borroso que suaviza las trayectorias descritas por el movimiento del robot.

Para ejecutar la técnica VFF, se necesita conocer la localización de los objetos en el mundo, esto lo conseguimos con la memoria de segmentos (mapa local) y con láser(t). Además, se necesita un punto destino, que en nuestro caso puede ser elegido autónomamente por el robot para explorar, o bien, impuesto desde el GUI por el usuario.

4.6.1. Fuerzas virtuales

VFF se basa en fuerzas ficticias entorno al robot: Los obstáculos ejercerán una fuerza repulsiva y el destino una fuerza atractiva. La suma vectorial de ambas fuerzas, repulsivas y atractiva, generará una fuerza resultante que será la que orientará al robot. Con esta técnica se mantiene un compromiso entre evitar obstáculos y, simultáneamente, llegar al objetivo determinado.

El algoritmo que implementa esta técnica es reactivo, ya que en cada iteración calcula todas las fuerzas, permitiéndole responder ante obstáculos imprevistos y/o dinámicos. La fuerza atractiva vendrá determinada por la orientación respecto al robot del objetivo y las fuerzas repulsivas por la distancia y orientación de los puntos de la lectura actual del láser, así como de los segmentos memorizados (no visibles por el láser, ya que sólo cubre 180 grados); más concretamente, la fuerza virtual repulsiva se calcula como el cociente de una constante de repulsión entre la distancia del robot al punto detectado. Esta fuerza repulsiva es parecida a la fuerza de repulsión eléctrica, ya que al disminuir la distancia entre el robot y el obstáculo, aumenta la repulsión. La fuerza resultante será la suma vectorial de ambas fuerzas y proporcionará al esquema la dirección a seguir (figura:4.27).

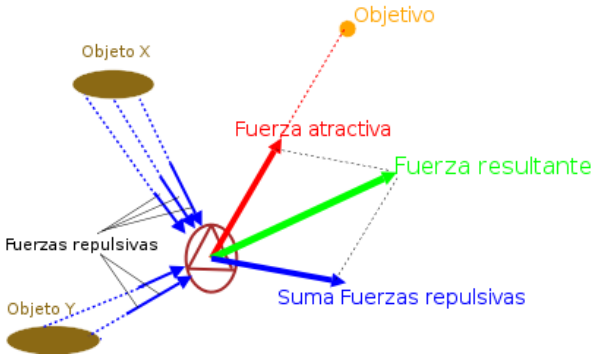


Figura 4.27: Obtención y visualización de las fuerzas virtuales

Para la obtención de las diferentes fuerzas se usarán las siguientes fórmulas.

$$X_{repulsiva} = \sum_{i=0}^{N} \frac{K_{repulsiva}}{distancia^2} * cos(\theta) \tag{4.6}$$

$$Y_{repulsiva} = \sum_{i=0}^{N} \frac{K_{repulsiva}}{distancia^2} * sin(\theta) \tag{4.7}$$

$$X_{atractiva} = K_{atractiva} * cos(\theta) \tag{4.8}$$

$$Y_{atractiva} = K_{atractiva} * sin(\theta) \tag{4.9}$$

Donde n es el número de puntos encontrados cercanos al robot, $K_{atractiva}$ la constante de atracción, $K_{repulsiva}$ la constante de repulsión y theta (θ) la orientación del punto, atractivo o repulsivo, respecto del robot.

Por tanto, se tiene como puntos que ejercen fuerza repulsiva los detectados por el láser(t) y, además, los puntos de objetos cercanos (almacenados en la memoria de la escena en forma de segmentos) que no son detectados por la lectura actual del sensor láser, debido a que estén situados en la parte posterior del robot. La obtención de los primeros, así como sus distancias respecto del robot, es directa ya que lo facilita la plataforma JDE. Sin embargo, para obtener los puntos, y sus respectivas distancias, de los objetos no detectados por el láser(t) es necesario realizar los siguientes cálculos: Para cada grado (desde 0 hasta 179) se obtiene la semirecta con punto inicial el centro del robot y pendiente la tangente de dicho grado. La semirecta se prolonga hasta intersecar con algún segmento memorizado, el cual debe estar próximo y a la "espalda" del robot. Una vez obtenido el punto de corte y calculada su distancia respecto del robot, la fuerza repulsiva que ejerce se suma a la fuerza repulsiva total (figura: 4.28).

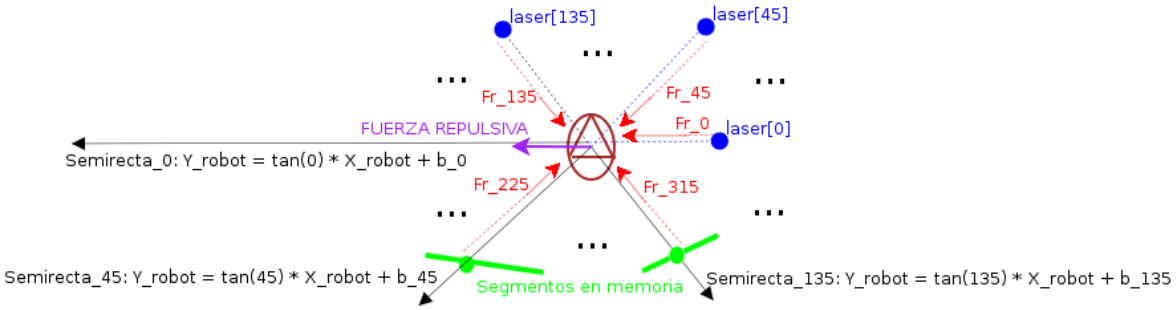


Figura 4.28: Fuerzas repulsivas detectadas y no detectadas por el láser

Este método de navegación local garantiza evitar obstáculos tanto estáticos como dinámicos, pudiendo tomar especial precaución con los últimos ya que se conoce tanto el tipo como la localización de éstos, gracias a la memoria de segmentos láser.

4.6.2. Controlador borroso

Una vez obtenida la orientación de la fuerza resultante, utilizaremos para decidir las velocidades angular y lineal (únicas órdenes que acepta el robot) un controlador borroso [Isado, 2005]. A grandes rasgos un controlador borroso consigue suavidad y rapidez en los movimientos. El controlador ha sido implementado usando la biblioteca auxiliar *Fuzzylib*.

El controlador implementado acepta como entrada una variables(*theta*) y devuelve la velocidad lineal y angular aconsejada (*vff-v* y *vff-w*) :

1. *theta*: Diferencia entre el ángulo actual del robot y el de la fuerza resultante, respecto del eje de abscisas.

2. *vff-v*: Velocidad lineal obtenida a partir de la entrada y las reglas definidas.

3. *vff-w*: Velocidad angular obtenida a partir de la entrada y las reglas definidas.

El controlador borroso define unas etiquetas sobre las variables de entrada y de salida. Estas etiquetas, por ejemplo, para la velocidad angular son:
etiqueta velocidad-angular alta-pos = 24 36 48 48
etiqueta velocidad-angular media-pos = 12 24 24 36
etiqueta velocidad-angular baja-pos = 0 12 12 24
etiqueta velocidad-angular nula = 0 0 0 0
etiqueta velocidad-angular baja-neg = -24 -12 -12 0
etiqueta velocidad-angular media-neg = -36 -24 -24 -12
etiqueta velocidad-angular alta-neg = -48 -48 -36 -24

También define reglas para determinar las variables de salida a partir de las variables de entrada:

IF (angulo = nulo) THEN (velocidad-angular = nula)
IF (angulo = bajo-pos) THEN (velocidad-angular = baja-pos)
IF (angulo = medio-pos) THEN (velocidad-angular = media-pos)
IF (angulo = alto-pos) THEN (velocidad-angular = alta-pos)
IF (angulo = bajo-neg) THEN (velocidad-angular = baja-neg)
IF (angulo = medio-neg) THEN (velocidad-angular = media-neg)
IF (angulo = alto-neg) THEN (velocidad-angular = alta-neg)

En la figura 4.29 puede observarse gráficamente las etiquetas para la velocidad angular.

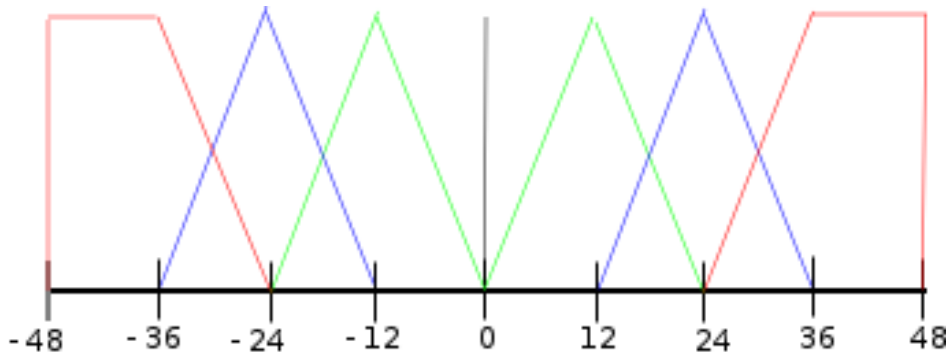
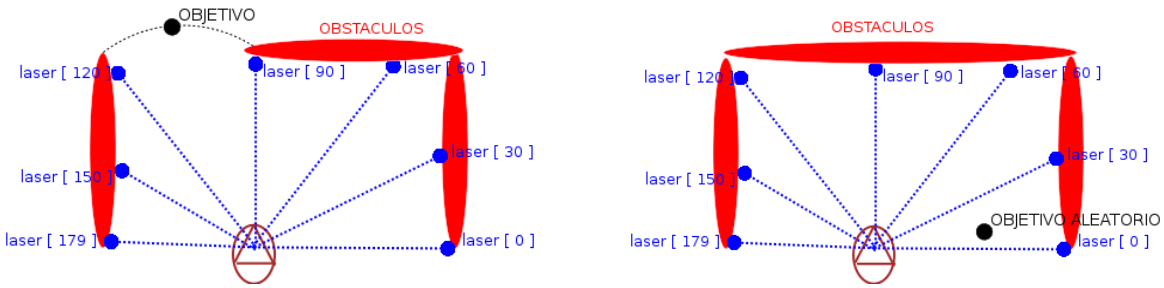


Figura 4.29: Gráfica de la velocidad angular en el controlador borroso

4.6.3. Navegación en modo "wandering"

Este modo autónomo de navegar consiste en explorar el entorno mientras se deambula por él. Es necesario seleccionar el punto objetivo actual del VFF al cual intentar llegar; para ello, se utiliza las medidas actuales del láser (de 0 a 179 grados), intentando encontrar una rodaja de, como mínimo, cinco grados donde no se haya detectado obstáculos a menos de tres metros (figura 4.30(a)). Si se encuentra tal rodaja intentaremos llegar hasta allí durante un cierto tiempo como máximo (si expira ese plazo se busca un nuevo objetivo). Si no se encuentra ninguna zona donde no haya obstáculos entonces se activará el modo aleatorio que hará dirigirse al robot a cualquier punto cercano a él hasta que detecte espacio libre (figura 4.30(b)). Los objetivos aleatorios tienen un corto tiempo de espera con la finalidad de encontrar pronto un espacio libre y no empecinarse en alcanzar un objetivo sin garantías.



(a) Exploración inteligente (b) Exploración aleatoria

Figura 4.30: Exploración del entorno: "wandering"

4.7. Interfaz gráfico

El interfaz gráfico se ha implementado a partir del esquema *guixforms*. Este esquema se considera de servicio, ya que sirve para teleoperar al robot y visualizar los resultados. La interfaz se ha programado con la biblioteca auxiliar *Xforms*. Al ser un esquema, se ejecuta iterativamente y, en cada iteración, comprueba el estado de los elementos del interfaz y muestra en pantalla los resultados que han sido solicitados.

Para poder gobernar al robot se utilizan objetos gráficos, que son leídos por el esquema *gui3D* (figura 4.31) para que ordene realizar o parar una acción. En nuestro interfaz gráfico se han mantenido algunos objetos gráficos del esquema *guixforms* original, estos son:

- *laser*: Permite la lectura del sensor láser, así como la visualización en pantalla de los 180 puntos detectados.
- *robot*: Permite la lectura del sensor odométrico, así como la visualización en pantalla del propio robot en el mundo.
- *colorA y colorB*: Permite la visualización en pantalla de las imágenes capturadas por las cámaras (par estereo).

Los principales objetos gráficos que se han añadido han sido:

- *virtual cam*: Permite la visualización tridimensional de la escena reconocida y, además, se visualizarán los puntos láser y el propio robot.
- *IDENTIFICAR persona, pared, congénere, puerta*: Permite la activación o desactivación de los esquemas para la identificación de dichos objetos en la escena, así como su visualización tridimensional en la imagen virtual.
- *VFF* Permite la activación del esquema para la navegación del robot.

- *WANDERING* Permite elegir si el robot navega de modo autónomo u obedeciendo órdenes del usuario.

Como se ha comentado, el botón *virtual cam* activa la posibilidad de visualizar tridimensionalmente la escena generada por nuestros esquemas. Para ello, fue necesario crear una cámara virtual con sus correspondientes parámetros extrínsecos e intrínsecos. Dicha cámara puede ser posicionada automáticamente con el botón *track robot*, el cual permite dejar la cámara estática o, por el contrario, hacerla volver a su posición original, que es sobre el robot, cuando éste se aleja demasiado en la imagen virtual creada. Además, gracias a diferentes objetos gráficos se puede modificar tanto la posición de la cámara como el foco de atención (hacia donde mira) de una manera cómoda.

Lo primero que mostraremos en la imagen es el suelo, el propio robot y los puntos detectados por el láser. Una vez creada esta base, iremos leyendo los segmentos en memoria uno a uno y, en el caso de estar identificado, se visualizará en tres dimensiones el objeto que representa. Como las localizaciones de los objetos están guardadas en coordenadas absolutas, se necesita convertirlas a coordenadas solidarias a la cámara virtual. Una vez hecho esto, con la función *project* de la biblioteca PROGEO obtenemos el píxel o los píxeles a partir de los cuales creamos la representación tridimensional de cada objeto identificado en la escena.

La representación tridimensional de la escena se muestra en la imagen superior derecha. Además, hemos dejado la imagen superior izquierda para la representación bidimensional y la imagen inferior izquierda para poder visualizar las capturas realizadas por la cámara (figura 4.31).

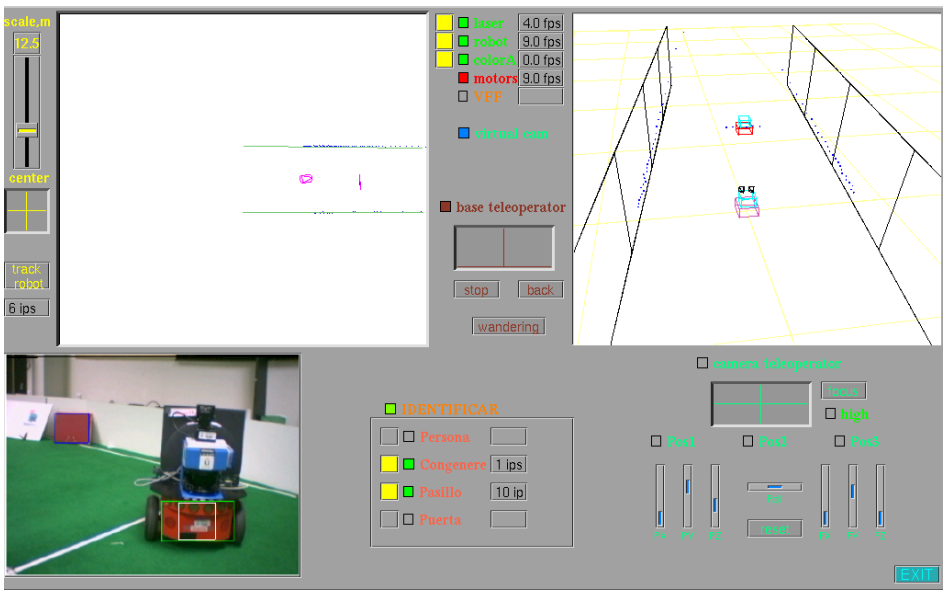


Figura 4.31: Interfaz gráfico de la aplicación

Capítulo 5
Resultados experimentales

En el capítulo anterior hemos descrito la solución final que se ha programado para alcanzar los objetivos de este proyecto. En este capítulo explicaremos qué pruebas y experimentos se han realizado para validar y mejorar la implementación en el camino hacia la solución conseguida. Comenzaremos detallando las pruebas realizadas en cada una de las identificaciones. A continuación, comentaremos las pruebas realizadas en la navegación. Por último, describiremos la ejecución típica del comportamiento que se ha generado, el cual integra el funcionamiento de todas las partes.

Las pruebas tanto parciales como finales, sobre el robot real descrito en el capítulo 3, se han realizado en un PC Intel Centrino a 1.6 GHz con 1 GByte de RAM.

5.1. Identificación de paredes

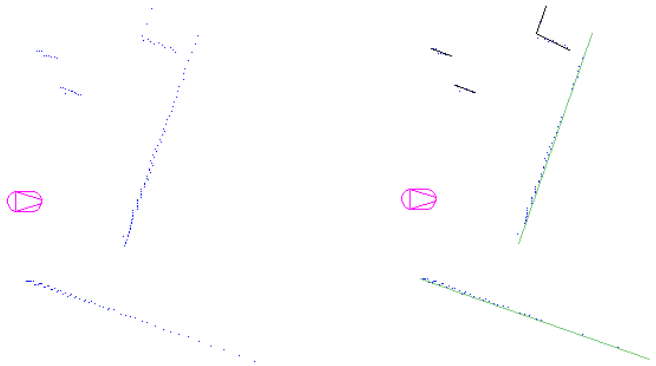
Antes de identificar paredes es necesario segmentar los puntos detectados por el láser, así como memorizar coherentemente los segmentos creados. Después de esto, ya podemos reconocer paredes y lo haremos principalmente por su longitud.

Inicialmente utilizamos la implementación de [Gómez, 2002] para segmentar los puntos láser con gran precisión y bajo coste computacional. Sin embargo, fue necesario disminuir aún más el coste, sin que ello afectara a la precisión, para satisfacer el requisito de vivacidad del capítulo 2 . Para ello, implementamos un algoritmo propio basado en mínimos cuadrados, explicado en el apartado 4.2.1, con una complejidad en tiempo lineal.

El mantenimiento de estos segmentos en memoria requiere una comparación continua entre los nuevos segmentos creados y los ya memorizados. Lo cual implica, inicialmente, una complejidad en tiempo cuadrática. Para disminuir el coste computacional, evitamos comparaciones inútiles entre segmentos actuales y memorizados de un modo rápido y eficaz: Antes de realizar el proceso de comparaciones es obligatorio que cumplan una sencilla condición: El segmento actual y el memorizado deben tener parte común en el eje de abscisas y en el eje de ordenadas, con una cierta flexibilidad. Con

esta optimización se acelera la correspondencia y, por tanto, la actualización de la memoria de segmentos.

En la figura 5.1 puede observarse el momento anterior (figura 5.1(a)) y posterior (figura 5.1(b)) a la segmentación de los puntos láser. Se aprecia que los segmentos captan bien las regularidades subyacentes en la colección de puntos láser.



(a) Puntos láser a segmentar

(b) Puntos láser segmentados

Figura 5.1: Creación y memorización de segmentos a partir de los puntos láser

En el camino hacia el algoritmo final nos hemos encontrado con unos problemas en los primeros prototipos que se han ido corrigiendo iterativamente. El primer problema que nos encontramos fue elegir correctamente el margen de error (distancia máxima del punto a la recta) que vamos a dar para considerar que un punto pertenece a un segmento. Si el umbral es muy alto, los segmentos creados son poco precisos y, por ejemplo, sería probable no distinguir una puerta de una pared; creando, por tanto, un único segmento cuando se debieron crear al menos dos. Si el umbral es muy bajo, corremos el riesgo de crear varios segmentos cuando debería ser sólo uno y, entonces, no podemos identificar la pared por su más importante característica: la longitud. Al final, se decidió trabajar con un margen de error de diez centímetros, con el cual podemos segmentar correctamente una pared y, muy posiblemente, distinguir una puerta aunque ésta se encuentre prácticamente cerrada (figura 5.2).

(a) Fusión de segmentos (b) Segmentos distintos

Figura 5.2: Fusión o no de posibles puertas con paredes

Otros problemas son los ocasionados por movimientos bruscos en el robot. Éstos dificultan realizar comparaciones entre los nuevos segmentos creados y los memorizados, debido a la desincronización entre el sensor láser y el sistema odométrico que es más lento, obteniéndose localizaciones de los segmentos que no son las reales. Incluso si es el mismo segmento, se pueden obtener distintas pendientes y localizaciones, lo que a menudo provoca la creación de segmentos fantasmas que no pueden ser fusionados con los memorizados. Para evitar en lo posible movimientos bruscos usamos la técnica de lógica borrosa en la navegación. Si no se pudiera evitar, por ejemplo, por la aparición imprevista de un obstáculo que se necesita sortear rápidamente, no se realizará la segmentación del láser en ese momento. Quizás otra posible solución sería aumentar la frecuencia de refresco en la odometría para disminuir su latencia y, de este modo, podría sincronizarse mejor con el sensor láser que es muy rápido.

El último problema que encontramos fue la imposibilidad, incluso con movimientos suaves, de fusionar los segmentos que representan las paredes de un pasillo completo después de que el robot acumulara suficiente ruido en la odometría, ya que diferían bastante las pendientes y localizaciones de los segmentos nuevos y los memorizados como para permitir su fusión. Para que no ocurriese se intentó corregir la diferencia de pendiente de una misma pared al fusionar los segmentos que la forman. Sin embargo, no fue posible fusionar los segmentos correctamente cuando la odometría acumulaba excesivo ruido. Al final se optó por olvidar el tramo más antiguo, de modo que la salud del segmento decrece proporcionalmente al tiempo sin detectarse por el láser y a la cantidad de movimiento que ha realizado el robot.

Aparte de los problemas que hemos ido encontrando, la segmentación del láser y la posterior identificación de paredes funciona aceptablemente bien, en parte porque el láser es un sensor muy fiable.

Este esquema se ha optimizado lo suficiente como para trabajar a más de diez iteraciones por segundo. Esto es fundamental ya que él se encarga de mantener la

memoria de segmentos láser compacta y coherente para distribuirla correctamente al resto de esquemas. Debido a que todos los esquemas necesitan tomar como entrada de datos parte o toda la memoria de segmentos, corría el riesgo de convertirse en el cuello de botella del sistema. Sin embargo, tras las optimizaciones, es uno de los esquemas que trabajan más rápido.

5.2. Identificación de puertas

En la identificación de puertas se han utilizado subestímulos sencillos como es el color marrón rojizo, forma y pertenencia a una pared. Estos subestímulos han sido suficientes para no obtener falsos positivos. El mayor problema que nos hemos encontrado en la detección de puertas es cuando el segmento láser que la representa ha sido fusionado con una pared. Cuando esto ocurre, debido a que la puerta esté completamente cerrada, no se puede garantizar la identificación. No damos como suficiente el subestímulo de la imagen para reconocerla como puerta, sino que es necesario, además, que el rayo visual entre la cámara y el objeto del color de la puerta corte con un segmento en memoria de longitud inferior a un metro.

Si la puerta se encuentra algo abierta, el segmento que se crea tendrá un tamaño y una diferencia de pendiente respecto a la pared que no permitirán la fusión y, por tanto, podrá detectarse fácilmente con la ayuda de la cámara (segmento en la imagen del color de la puerta que coincide con un segmento láser pequeño y junto a una pared).

Hemos realizado pruebas con diferentes posiciones, luminosidad y aperturas para tratar de identificar puertas. Por ejemplo, en la figura 5.3, se identifica estando la puerta abierta y, en la figura 5.4, estando entornada.

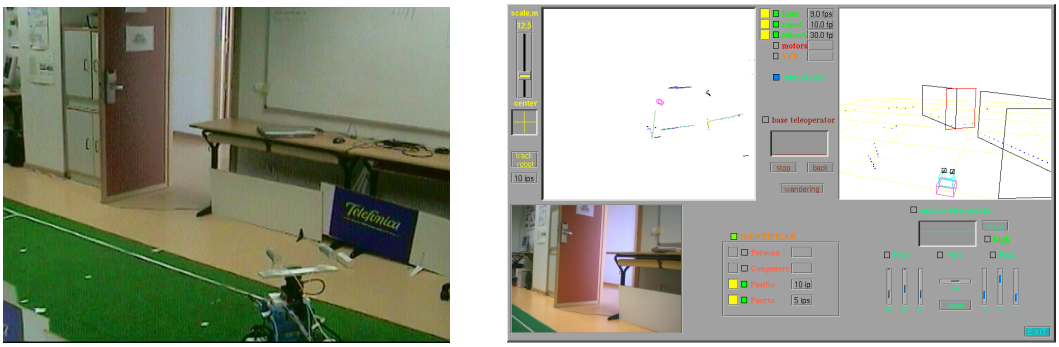


Figura 5.3: Identificación de una puerta abierta

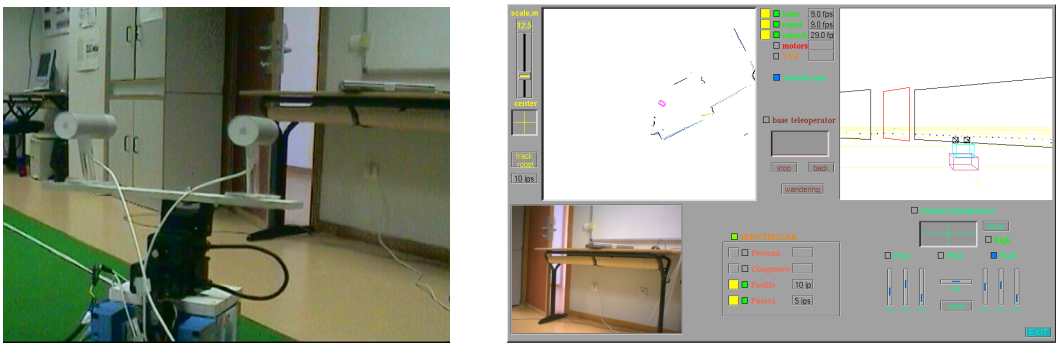


Figura 5.4: Identificación de una puerta entornada

Los resultados para las figuras 5.3 y 5.4 pueden observarse en la visualización tridimensional generada por el robot (imagen superior derecha del interfaz gráfico), comprobándose que ha identificado la puerta correctamente (pared de color negro y puerta de color rojo) tanto abierta como entornada.

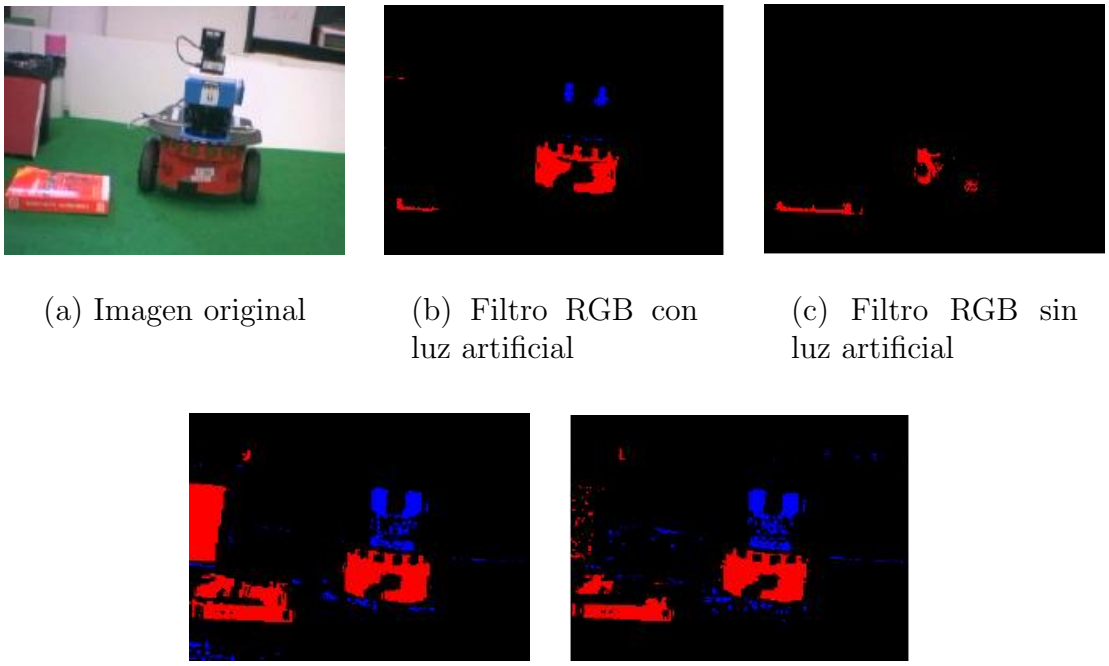
Este esquema puede trabajar entre 8 y 10 iteraciones por segundo, sin embargo, lo hemos limitado a 5 iteraciones por segundo para que no consuma demasiada CPU, ya que no es un esquema fundamental.

El balance de funcionamiento de este esquema es positivo. Se han obtenido identificaciones satisfactorias siempre y cuando la puerta esté abierta, siendo más difícil éstas cuando está casi o completamente cerrada. En este caso, si quisiéramos detectarla deberíamos obviar los datos de los segmentos láser y sólo tratar con la imagen y, posteriormente, dividir el segmento láser que representa la pared, en al menos dos porciones(pared y puerta). Esto no le otorgaría la robustez deseada, ya que para realizar la detección no sólo se exige color y forma (segmentos de imagen), sino también la anchura, así como cercanía y pendiente respecto a una pared (segmentos de láser).

5.3. Identificación de congéneres

En la identificación de congéneres se ha conseguido una combinación de subestímulos sencillos tal que no es posible engañar al robot con objetos de color y tamaño parecido. Además de que el robot tenga una base roja y encima de ésta tenga parte azul y/o negra, es imprescindible el subestímulo de profundidad. Como se explicó en el capítulo 4, para que haya sensación de profundidad, el rayo visual entre el segmento rojo en la imagen y la cámara tiene que cortar con un segmento láser memorizado; además, éste tiene que tener una longitud característica (entre 30 y 60 centímetros).

Para la realización de los filtros de color, decidimos que fuera en el formato HSI frente al formato RGB. Este último nos ofrece una mayor rapidez en la ejecución, ya que es el formato en el que obtenemos la imagen y no es necesaria la transformación de formato (píxel a píxel); sin embargo, es muy sensible a los cambios de luminosidad, como puede observarse en las figuras 5.5(b) y 5.5(c),lo cual le hace poco robusto para nuestros fines. Aún siendo necesaria la transformación continua de formato, es imprescindible la mayor robustez que ofrece el formato HSI, como puede observarse en las figuras 5.5(d) y 5.5(e), para obtener los resultados deseados (figura 5.5).



(a) Imagen original

(b) Filtro RGB con luz artificial

(c) Filtro RGB sin luz artificial

(d) Filtro HSI con luz artificial

(e) Filtro HSI sin luz artificial

Figura 5.5: Formato RGB frente a Formato HSI

Una vez obtenido un filtro de color bastante robusto, se lleva a cabo la segmentación basada en la imagen filtrada. Para segmentar utilizamos inicialmente el algoritmo de segmentación recursiva [Hidalgo, 2006]. Este algoritmo se había implementado para la identificación exclusiva de un congénere, por tanto, tanto los filtros como la segmentación venían dados por colores predeterminados (rojo, azul y negro), realizados en una sola iteración del algoritmo. Fue necesario modificar la implementación para que admitiera la generalidad que necesitamos (segmentación por cualquier color, pero de uno en uno).

Un problema es el alto coste computacional en algunas identificaciones, ya que debemos realizar el proceso (filtro y segmentación) hasta tres veces como en el caso del congénere. Se han limitado a tres las recursiones del algoritmo para que el coste no

fuese muy alto en el peor de los casos (varias recursiones del algoritmo). Además, sólo se filtra por azul o negro si hay algo rojo. Y, de este modo, permitir la detección de diversos objetos con ayuda de la cámara (como puertas, personas u otros posibles).

Hemos realizado pruebas con diferentes distancias, luminosidad y orientaciones para tratar de identificar al congénere, así como también, intentar engañarle con objetos de similar tamaño y color. Por ejemplo, en la figura 5.6, se identifica al congénere estando de lado y, en la figura 5.7, estando de espaldas.

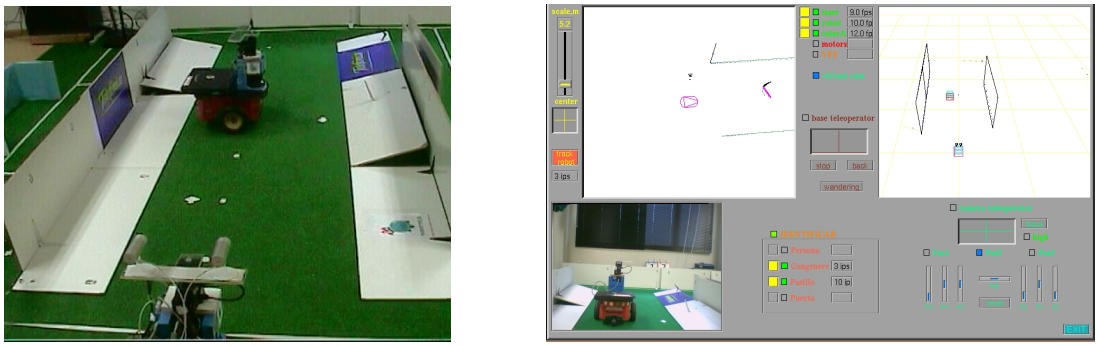


Figura 5.6: Identificación de un congénere de lado

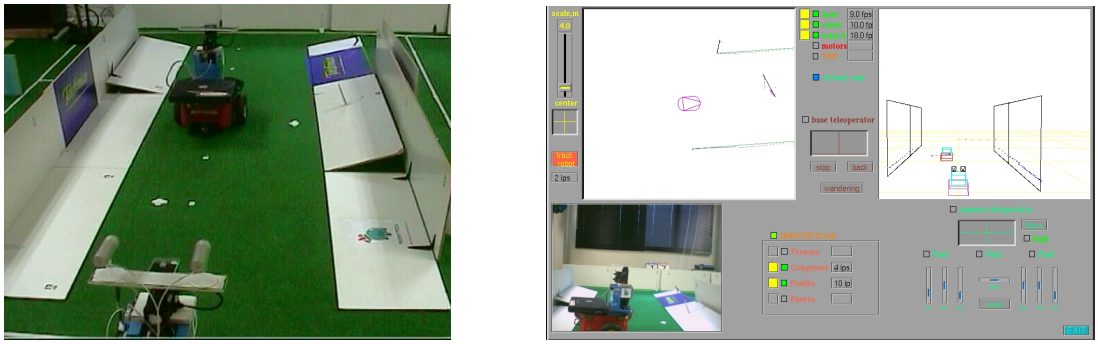


Figura 5.7: Identificación de un congénere de espaldas

Los resultados para las figuras 5.6 y 5.7 pueden observarse en la visualización tridimensional del entorno generada por el robot (imagen superior derecha del interfaz gráfico), comprobándose que ha identificado al congénere correctamente en ambas situaciones.

Se han obtenido identificaciones satisfactorias a una distancia media o corta, siendo más difícil éstas cuando el congénere se encuentra a más de seis metros; sin embargo, sí ha sido posible la identificación en todas las orientaciones del congénere y con condiciones de luz variables; además, no ha sido posible engañarle con objetos similares,

demostrando así la robustez deseada.

Este esquema es el más lento de todos, trabajando entre 1 y 3 iteraciones por segundo, debido a los costosos tratamiento de imágenes, que realiza hasta tres veces en el peor de los casos (por los colores rojo, azul y negro). Además debe encargarse de contrastar la información obtenida con los segmentos láser.

5.4. Identificación de personas

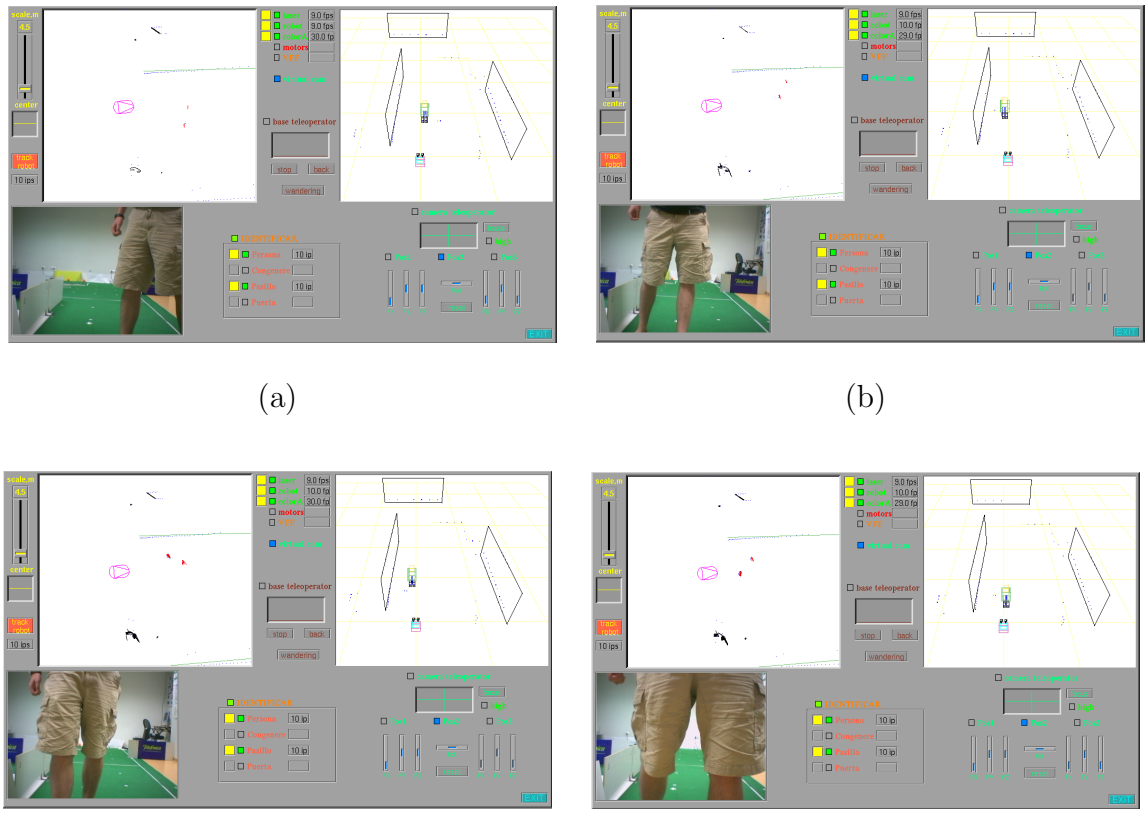
Se ha conseguido detectar personas desde todos los ángulos, siempre y cuando se consiga segmentar ambas piernas, ya que la identificación se basa en el tamaño y similitud de ambos segmentos. Para la detección es necesario el movimiento, ya sea detectado por el láser o por la cámara. Una vez identificado, ya no es necesario dicho movimiento, tan sólo realizar un seguimiento para que, en caso de desplazarse algún segmento, se actualice la nueva posición en la memoria y se elimine la posición anterior.

Las mayores dificultades han sido dos, la primera fue ajustar la sensibilidad de movimiento, es decir, cuánto desplazamiento espacial de un segmento candidato necesito para decidir que es persona. Si es poco, damos falsos positivos al creer que segmentos inmóviles se han movido, esto es debido fundamentalmente a errores en las medidas del láser cuando el robot realiza giros rápidos. En cambio, si es mucho, es posible no dar como válido un paso pequeño de una persona y, por tanto, no detectarla. Al final se optó por requerir más de medio metro de desplazamiento para decidir que es persona, ya que es preferible no detectar un paso a obtener falsos positivos.

El otro problema fue actualizar los pasos ,es decir, una vez que se ha detectado el movimiento en algún segmento candidato que representa alguna de las piernas, es necesario actualizar la posición y eliminar el segmento anterior. En una primera aproximación, se realizaron correspondencias por similitud de tamaño, pendiente y posición; una vez encontrado, se sustituía en memoria el segmento antiguo por el actual. Con este método, a menudo, se sustituía el segmento que representa la otra pierna, lo cual creaba un conflicto que nos llevaba a perder a la persona en memoria; por tanto, fue necesario ajustar más en la condición de posición. Desde la perspectiva del robot se exige una diferencia de distancia (entre segmento antiguo y nuevo) muy cortas para el movimiento lateral y bastante más flexible para el movimiento frontal. Es decir, los pasos de una persona andando hacia adelante (o atrás) puede desplazarse incluso un metro, mientras que andando hacia los lados el desplazamiento es bastante menor; esta idea nos ha permitido distinguir si se mueve la pierna izquierda o derecha para poder

actualizar el segmento adecuado en memoria.

En la figura 5.8 se muestran dos secuencias de identificación para distintos movimientos de una persona. En las figuras 5.8(a) y 5.8(b) la persona realiza un desplazamiento lateral respecto del robot que es detectado por éste y actualiza la posición. En las figuras 5.8(c) y 5.8(c), ahora el desplazamiento de la persona es frontal y, de nuevo, lo detecta y actualiza la posición.



(a) (b)

(c) (d)

Figura 5.8: Resultados de la identificación de una persona

Para poder identificar a una persona se necesitan crear dos segmentos láser que representen ambas piernas, lo que significa que deben ser detectadas cada una por al menos dos puntos láser. Esto nos limita a una distancia máxima de detección de unos cuatro metros. Además si las piernas están alineadas tampoco podrán crearse los dos segmentos necesarios aunque la persona esté cerca del robot. Basta un ligero desplazamiento en las piernas de la persona para que puedan actualizarse los segmentos en memoria y, si es posible, crear los dos segmentos candidatos. Una vez que se tienen éstos, si hay movimiento de las piernas (que sería detectado por el láser) o de los brazos (que sería detectado por la cámara, siempre y cuando el robot esté quieto), se clasificarán los segmentos como persona.

Como este esquema sólo trabaja con los segmentos candidatos, lo cuales tienen unas características muy especiales, la cantidad de estos es muy limitada en la escena y, por tanto, el coste de seguimiento, detección y mantenimiento es muy bajo. Además, el filtro de movimiento en la imagen sólo se puede realizar cuando el robot está quieto. Por todo esto, el esquema para identificar personas puede trabajar a mucho más de diez iteraciones por segundo, pero lo hemos limitado a ese número para dar más tiempo de ejecución a otros esquemas.

5.5. Navegación

Como vimos en el capítulo 4, la técnica que utilizamos para la navegación local es la de VFF, basándonos en los puntos detectados por el sensor láser y en los segmentos memorizados. Como destino tendremos un punto espacial donde exista espacio libre y, en caso que no haya espacio alrededor, un lugar aleatorio.

Antes de llevarlo al robot real, se realizaron las pruebas de navegación sobre el simulador *Player/Stage*, donde fuimos ajustando los valores de las etiquetas de la lógica borrosa para evitar movimientos bruscos y, también, fuimos evitando oscilaciones innecesarias, sobre todo al sortear obstáculos cercanos, gracias al conocimiento del entorno que rodea al robot. Fuimos dando diferentes pesos de repulsión a los puntos actuales del láser y a los puntos de intersección con los segmentos memorizados, situados tanto delante como detrás del robot, hasta que el robot fuera capaz de navegar de modo suave, rápido y seguro .

Una vez obtenida una navegación más que fiable sobre el simulador, pasamos a ejecutar sobre el robot real con una velocidades máximas de 200 mm/seg y 48 grados/seg, comprobando que la navegación se ajustaba bastante a lo esperado. Sin embargo, cuando el obstáculo a sortear era una persona, sólo era capaz de evitarla cuando se encontraba demasiado próximo a ella y, por tanto, corriendo el riego de colisionar. Esto es debido a que las piernas de una persona sólo son reflejadas por unos pocos puntos láser, los cuales no generan la suficiente fuerza repulsiva como para cambiar la dirección del robot hasta encontrarse muy cerca. Gracias a que podemos identificar objetos potencialmente móviles, se dan mayor peso de repulsión a los puntos que pertenecen a una persona o a un congénere para poder sortearlos con suficiente antelación y margen de seguridad. Una ventaja de navegación con una memoria a corto plazo es que si el robot sabe que detrás de él (zona no reconocida por el láser(t)) se encuentra un objeto móvil cercano, no realizará giros bruscos o rápidos para mayor seguridad.

En la figura 5.9, se muestra una secuencia donde el robot debe sortear a su congénere

para intentar llegar a su destino. En este caso, no activamos el esquema para identificar al congénere, por tanto, tan solo lo entiende como un obstáculo gracias al láser. Aunque sí consigue evitarlo, pasa demasiado cerca de él (a unos dos centímetros), con el riesgo que esto supone.

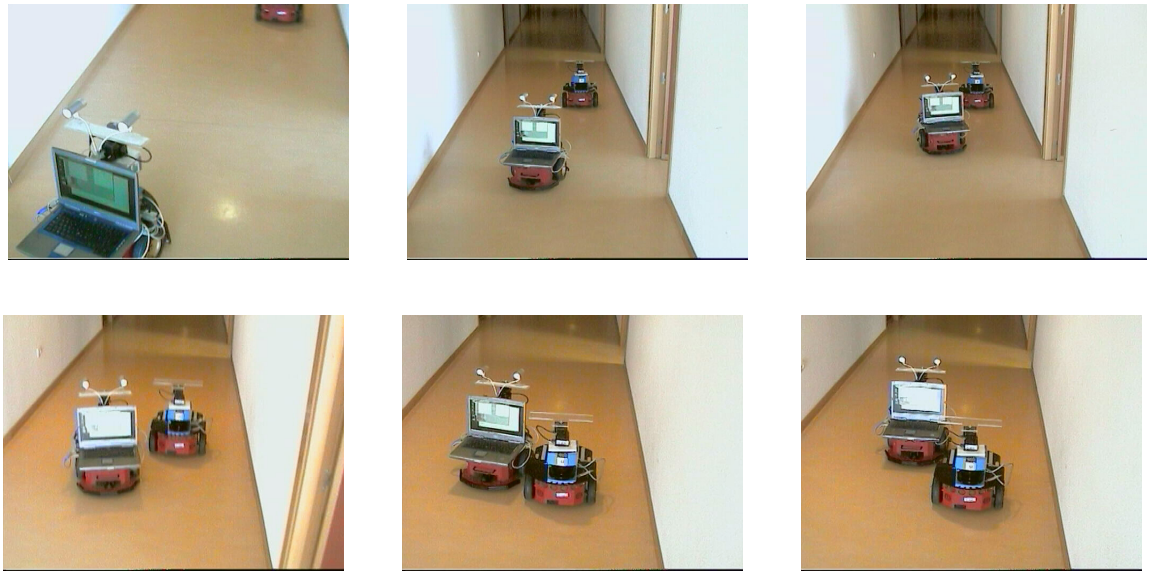


Figura 5.9: Sortear al congénere sin haberle identificado

Realizamos la misma prueba, pero ahora activando el esquema de identificación del congénere. Como puede verse en la figura 5.10, es capaz de sortearlo con mayor seguridad (pasa a unos diez centímetros) gracias a que lo ha reconocido, aunque los puntos láser repulsivos sean los mismos que en la figura 5.9.

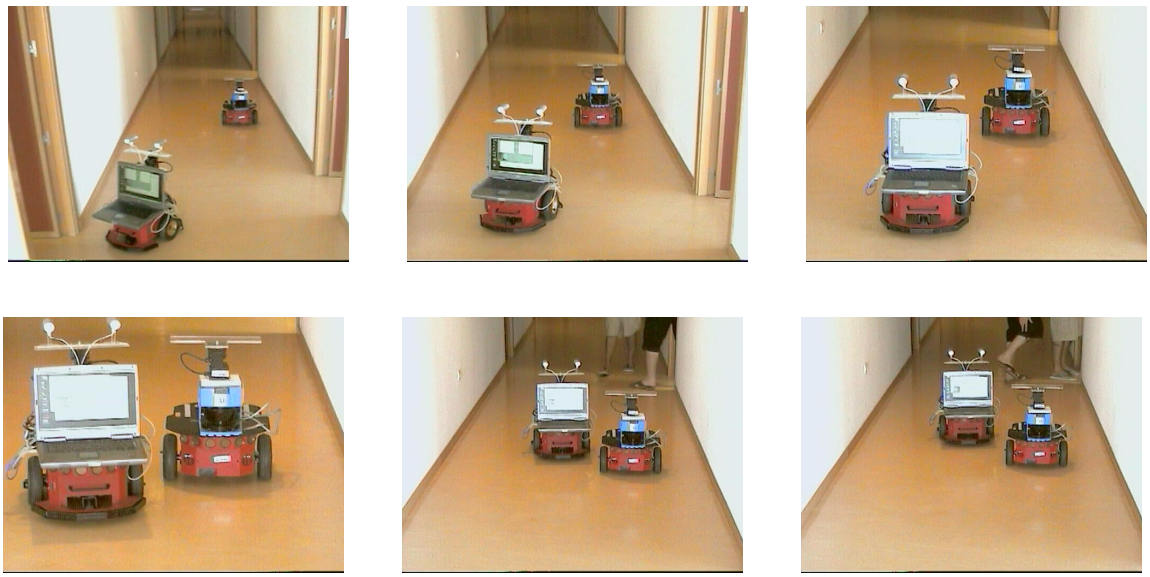


Figura 5.10: Sortear al congénere habiéndole identificado

Ahora lo comprobamos sorteando a una persona, donde los puntos repulsivos de-

tectados por el láser son aún menores que en el caso del congénere. En la figura 5.11 puede comprobarse, con el esquema de identificación de personas activo, que consigue evitarla con bastante seguridad.

Figura 5.11: Sortear a una persona habiéndola identificado

Como última prueba, con todos los esquemas activos, obstaculizamos dinámicamente la navegación del robot para comprobar la rapidez de reacción del esquema *vff* frente a situaciones de alto riesgo de colisión. Como este esquema puede y debe trabajar como mínimo a diez iteraciones por segundo, incluso con la carga computacional que implica ejecutar todos los esquemas a la vez, el robot consiguió salir airoso en todo momento, demostrando así la robustez y vivacidad esperadas (figura 5.12).

Figura 5.12: Sortear una persona muy cercana y en continuo movimiento

5.6. Ejecución típica del comportamiento

Cada identificación, así como la navegación, se han implementado en esquemas distintos que se ejecutan concurrentemente. En la ejecución de todo el sistema integrado podrán estar bien todos los esquemas activos, o bien, sólo los esquemas seleccionados. Mientras el robot deambula autónomamente por el entorno buscando espacio libre, va identificando los diversos objetos que aparezcan en su camino y memorizando su localización, lo cual le permitirá tomar mejores decisiones en su navegación. Podremos visualizar las identificaciones que ha realizado, así como el mantenimiento en la memoria de los objetos, en el interfaz gráfico donde se representa tridimensionalmente la escena reconstruida por el robot desde su información sensorial.

Como ejemplo, explicaremos una situación típica, en la cual el robot va identificando y explorando autónomamente a la vez que va construyendo su representación del entorno. Esto le ayudará a dirigirse de manera segura, y sin que nadie le comande, hacia un lugar donde haya encontrado espacio libre. En este caso, tomamos el recorrido de la figura 5.13.

(a) (b) (c)

Figura 5.13: Secuencia de una ejecución típica

En el resultado para la situación de la figura 5.13(a), tras arrancar el programa, se identifican las paredes y la puerta de enfrente como puede verse en la imagen superior derecha de la figura 5.14, donde se ha representado la pared en negro y la puerta en rojo. También se identifica y representa a una persona, aunque ésta es detectada sólo parcialmente por la cámara, el láser la ha detectado completamente.

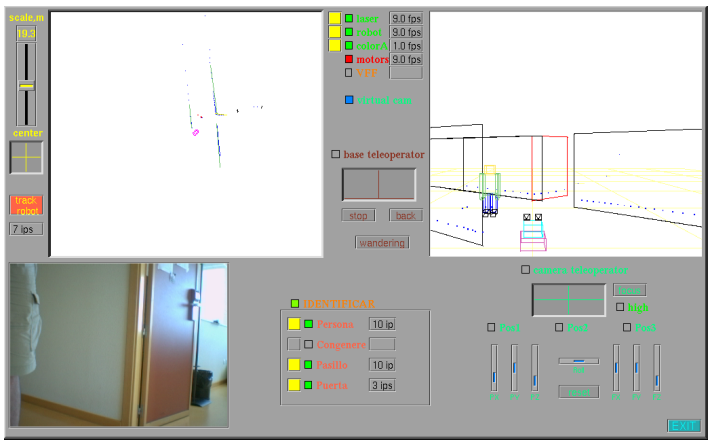


Figura 5.14: Solución a la figura5.13(a)

Una vez en el pasillo (figura 5.13(b)), ahora se identifica al congénere como puede verse en la figura 5.15. Además se ha olvidado a la persona y a la puerta, esto puede ocurrir porque se haya agotado su salud en memoria, o bien, porque hayamos desactivado el esquema de identificación concreto en ejecución, como en este caso.

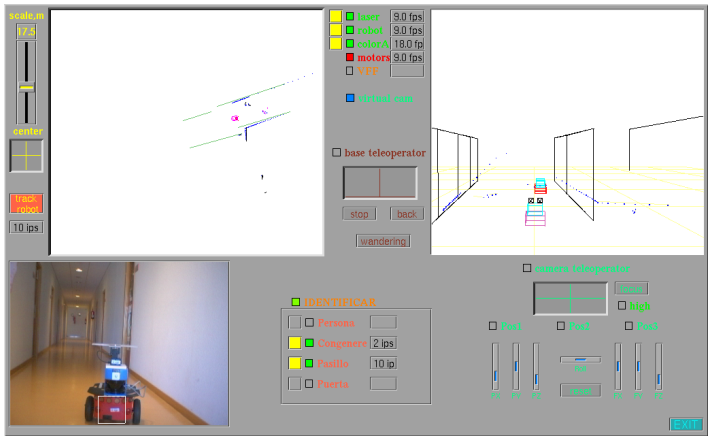


Figura 5.15: Solución a la figura 5.13(b)

Una vez identificado y rebasado al congénere (figura 5.13(c)), el robot conoce la localización de éste, aunque ya no pueda detectarle ni con el láser ni con la cámara, debido a que lo ha reconocido anteriormente en un lugar concreto. También se ha fusionado el tramo que se aprecia actualmente de la pared con el que se veía antes, desde más atrás; formando la pared completa (figura 5.16).

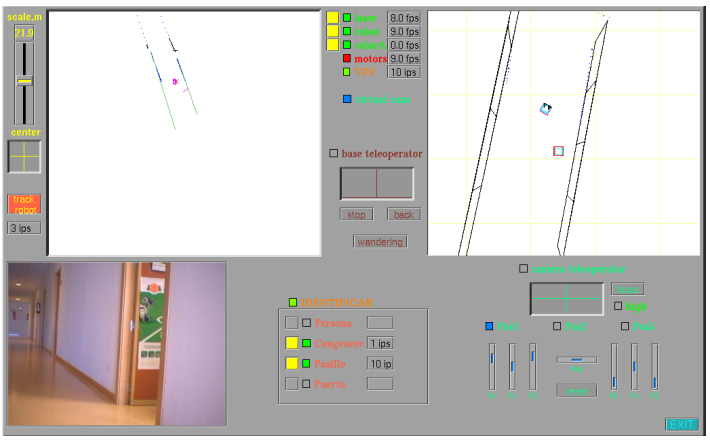


Figura 5.16: Solución a la figura 5.13(c)

Si iniciaramos la ejecución en la situación de la figura 5.13(c), sólo se podría tratar y, posteriormente, representar la información instantánea que nos llega de los sensores (figura 5.17). En contraste, como anteriormente se había explorado la zona trasera el robot, se tiene un mayor conocimiento del entorno ya que, además de la información sensorial instantánea, se cuenta con la información que proporciona la memoria (figura 5.16).

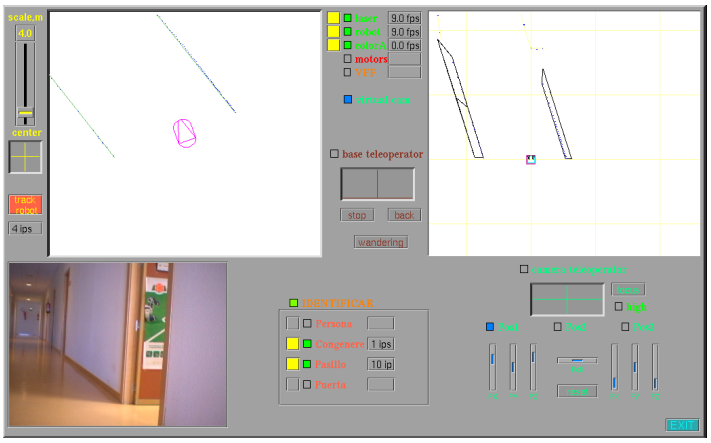


Figura 5.17: Información sensorial instantánea en la figura 5.13(c)

Puede ser muy importante a la hora de evitar colisiones contar con una memoria a corto o medio plazo, que permita conocer la situación de un objeto sin necesidad de tener que detectarle por los sensores en todo momento, es decir, tener información sobre la zona trasera del robot. De este modo, se pueden evitar maniobras arriesgadas y tomar mejores decisiones como podría ser, por ejemplo, alejarse un poco del obstáculo que tiene detrás antes de realizar un giro, sobre todo si se sabe que es un objeto móvil.

Además, tanto las identificaciones como la memoria favorecen las interacciones complejas con objetos. Ya sea en comportamientos de seguimiento, acompañamiento o huida, como también en la disputa de competiciones, dotándole de mayor inteligencia.

Capítulo 6
Conclusiones y trabajos futuros

Descrita la solución propuesta para este comportamiento y comentados algunos de los experimentos más relevantes, terminamos esta memoria exponiendo las conclusiones obtenidas de este proyecto y las posibles líneas futuras en las que se puede seguir investigando.

Haciendo un balance de los objetivos marcados en el capítulo 2, vamos a describir en qué medida se han cumplido éstos, satisfaciendo los requisitos, también expuestos en el capítulo 2.

6.1. Conclusiones

El primer subobjetivo, cuya solución se describe en el apartado 4.2.1, fue segmentar los puntos detectados por el sensor láser. Así como localizar los segmentos creados con ayuda de la odometría y memorizarlos coherentemente. Para cumplir con el requisitos de vivacidad, fue necesario optimizar al máximo el algoritmo de segmentación del láser, además de procurar que el mantenimiento de la memoria de segmentos fuese lo menos costoso posible. Se ha conseguido mantener una memoria a corto o medio plazo, con una vida inicial, para los segmentos memorizados, limitada por la acumulación excesiva de ruido en el sistema odométrico del robot, tal y como se explicó en el apartado 5.1. La principal conclusión, una vez conseguida la funcionalidad deseada, es la importancia de la optimización del código, máxime cuando se trabaja con sistemas en tiempo real; además, es necesario conocer las limitaciones del sistema hardware con el que trabajamos para obtener, en la medida de lo posible, un mayor rendimiento vía software.

El segundo subobjetivo, cuya solución se describe en el apartado 4.4.1, fue realizar filtros por color y movimiento sobre las imágenes capturadas por la cámara, así como una posterior segmentación sobre las imágenes filtradas para obtener información del objeto (en caso de encontrarse en la imagen). Si se detectara el objeto buscado por color y forma, dicha información es contrastada con la memoria de segmentos láser para una correcta identificación. Para obtener información correcta y, por tanto, una

robusta identificación del objeto, como se exige en los requisitos del apartado 2.2, es fundamental una alta eficacia en los algoritmos de tratamiento de imagen, incluso en condiciones de luminosidad críticas. De ahí el cambio de formato de RGB a HSI para la realización de filtros, como vimos en el apartado 4.3.1, así como la utilización del algoritmo de segmentación recursiva [Hidalgo, 2006], el cual modificamos para permitir una mayor generalidad.

El tercer subobjetivo era fusionar la información obtenida por los algoritmos de segmentación del láser y de los algoritmos de segmentación de la imagen con la ayuda de la biblioteca PROGEO. Para realizar la fusión sensorial, primero, se obtiene un rayo visual entre la cámara y el píxel central del segmento de imagen y, después, se estudia el punto de corte con algún segmento láser en memoria. De este modo obtenemos información de tamaño, forma y color de un objeto, éstos serán los subestímulos que permitirán su identificación.

En la identificación de paredes, únicamente nos basamos en el sensor láser y la odometría, siendo la principal característica su longitud. Por tanto, es el caso más sencillo, ya que en el resto de identificaciones se utilizará, además del láser y la odometría, la cámara. El resultado final ha sido bastante satisfactorio.

En la identificación de puertas, hemos tomado como subestímulos su color, tamaño y proximidad a una pared. Como se ha expuesto en el apartado 5.2, no se ha podido garantizar la detección en caso de estar completamente cerrada, debido a la fusión en memoria del segmento que la representa con su pared. Se detecta, en cambio, de un modo fiable cuando la puerta se encuentra algo abierta.

En la identificación del congénere, hemos tomado como subestímulos el color (rojo, azul y negro), la forma, el tamaño y la sensación de profundidad. Como vimos en el apartado 5.3 es la identificación más costosa, ya que debemos filtrar y segmentar la imagen hasta tres veces, una por cada color, además de estudiar la profundidad. Esto fue necesario para identificar robustamente al congénere sin riesgo de falsos positivos.

La identificación de la persona es la parte más novedosa de este proyecto fin de carrera, ya que no se basa en el color, si no que está basada en subestímulos de forma y movimiento. De este modo, podemos identificar a una persona sin necesidad de que vista con un color representativo. Los resultados de identificación y mantenimiento, el cual supone un esfuerzo añadido en caso de movimiento (al igual que en el congénere), los vimos en el apartado 5.4, con unos resultados razonablemente satisfactorios.

El cuarto subobjetivo, cuya solución se describe en el apartado 4.7, permite visualizar en tres dimensiones al propio robot en movimiento y al entorno estudiado por él. De este modo, podemos saber qué objetos a identificado, dónde los sitúa y cuándo los olvida. Para ello, se ha implementado una cámara virtual sobre el robot con la posibilidad de ajuste manual o automático de posición y foco de atención. Las imágenes virtuales en 3D, además de permitirnos la depuración del programa, permite al usuario un mejor entendimiento de los resultados obtenidos por el robot.

El quinto y último objetivo, cuya solución se describe en el apartado 4.6, permite al robot navegar por el entorno buscando lugares inexplorados. Se ha programado y ajustado el algoritmo de navegación local VFF [Borenstein, 1989], utilizando la técnica de lógica borrosa para evitar movimientos bruscos. Nos basamos en los puntos detectados por el láser en el momento y, sobre todo, en los segmentos memorizados, que al estar clasificados como persona, congénere, puerta o pared, nos permite tomar mejores decisiones en la navegación. De este modo, se pueden evitar obstáculos dinámicos, incluso en condiciones de máximo riesgo de colisión. Tal y como se ha explicado en el apartado 5.5, la identificación y localización de diversos objetos en el entorno permite al robot, además de una navegación completamente segura, la interacción con dichos objetos si se desea.

Como se comentó en el apartado 5.3, obtener información a partir del tratamiento de imágenes, además de costoso computacionalmente, es bastante complicado. Gracias a la modularidad que permite la plataforma "JDE", fue posible la integración en nuestro sistema de códigos ya generados. Sin estas ayudas, el tiempo de realización de este proyecto hubiese sido ostensiblemente mayor. Debo resaltar, por tanto, la importancia de un buen estilo de programación, así como la implementación de esquemas con un bajo o moderado acoplamiento y una alta cohesión, con vistas a una potencial reutilización.

Hemos trabajado con estímulos muy frecuentes en robótica de interiores, como pueden ser personas, puertas u otros robots. Uno de los mayores novedades que ha aportado este proyecto, en su línea continuista con otros proyectos ya realizados, ha sido la identificación y localización de personas sin la necesidad de que vistan de un modo determinado. También ha sido novedosa la identificación de puertas. Pero el aspecto fundamental es el mantenimiento de una memoria a corto plazo que integrando los objetos identificados ha permitido una mejor navegación local.

Para obtener los resultados esperados sobre el robot real, fue necesario calibrar sus cámaras, configurar el servidor *oculo* para que las imágenes nos llegaran perfectamente, así como configurar y sincronizar el servidor *otos* para obtener las medidas láser y los cálculos odométricos del robot. Esto me ha permitido, en combinación con el PC, trabajar con un nuevo dispositivo altamente complejo (robot con sensor láser, sónar, sistema odométrico y cámaras par estéreo); y, a pesar que los ajustes en el robot han requerido una dosis adicional de trabajo, se han podido contemplar los resultados más allá de la simulación.

Una vez verificado experimentalmente el comportamiento en el robot real, podemos concluir que ha sido generado correctamente. Resaltando la robustez de los algoritmos de identificación. Con ellos no se han obtenido falsos positivos, incluso con objetos similares, y sólo en casos excepcionales han dado falsos negativos como pudieran ser, en general, objetos a demasiada distancia del robot y, en particular, puertas cerradas completamente. Por último, debemos resaltar también la vivacidad del algoritmo de navegación. Con el cual el robot ha sido capaz de sortear cualquier obstáculo de forma segura, incluso cuando éstos se le acercaban rápida y peligrosamente.

6.2. Trabajos futuros

En este apartado se detallan algunas posibles mejoras que podrían realizarse sobre este proyecto y que pueden servir como nuevas líneas de investigación para otros proyectos fin de carrera futuros.

Usar como entrada de nuestro mapa local información de las dos cámaras par estéreo, eliminando incluso como entrada los valores obtenidos por el láser. Con esto, además del ahorro económico (el láser es mucho más caro que una cámara), detectaríamos objetos no sólo en un plano, como ocurre con el láser. También se evitaría el problema de los pasillos con ventanales a la altura del láser, ya que éste no es capaz de detectar como obstáculo un objeto transparente.

Con el par estéreo, utilizar el movimiento del cuello mecánico, consiguiendo así una atención visual 3D y, de este modo, buscar objetos en el espacio sin necesidad de mover la base del robot.

Permitir a los usuarios dar órdenes al robot con simples movimientos de brazos. Esto parece relativamente asequible: gracias a que ya hemos identificado y localizado en pantalla a la persona, solamente sería necesario realizar filtros de movimiento alrededor de dicha persona para detectar posibles movimientos de brazos en un lugar u otro de la imagen.

Tratar de identificar otros objetos cotidianos, que puedan encontrarse en el interior de edificios, como pudieran ser ventanas, fotocopiadoras, ordenadores, etc. También, implementar interacciones más complejas, como la citada anteriormente u otras.

Combinarlo con información estática y mapas del entorno para conseguir una navegación global. Así como también, localizarse mejor gracias al reconocimiento de puertas y paredes que pueden suponer fuertes discriminantes para una localización más rápida.

Bibliografía

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

[Calvo, 2004] Roberto Calvo. Comportamiento sigue persona con visión direccional. *Proyecto Fin de Carrera, URJC*, 2004.

[Gómez, 2002] Víctor Gómez. Comportamiento sigue pared en un robot con visión local. *Proyecto Fin de Carrera, URJC*, 2002.

[Hidalgo, 2006] Víctor Hidalgo. Comportamiento de persecución de un congénere con el robot pioneer. *Proyecto Fin de Carrera, URJC*, 2006.

[Isado, 2005] José Raúl Isado. Navegación global utilizando método del gradiente. *Proyecto Fin de Carrera, URJC*, 2005.

[Lobato, 2005] David Lobato. Arquitectura jde+ para el control de robots. *Proyecto Fin de Carrera, URJC*, 2005.

[Lorenz, 1978] K. Lorenz. Fundamentos de la etología. *Editorial Paidos*, 1978.

[López, 2005] Alejandro López. Navegación global utilizando grafo de visibilidad. *Proyecto Fin de Carrera, URJC*, 2005.

[Martín, 2002] Félix San Martín. Comportamiento sigue pelota en un robot con visión local. *Proyecto Fin de Carrera, URJC*, 2002.

[Martínez, 2003] Marta Martínez. Comportamiento sigue pelota con visión cenital. *Proyecto Fin de Carrera, URJC*, 2003.

[Plaza, 2003] José María Cañas Plaza. *Jerarquía Dinámica de Esquemas para la generación de comportamiento autónomo*. PhD thesis, Universidad Politécnica de Madrid, 2003.

[Plaza, 2004] José María Cañas Plaza. Manual de programación de robots con jde. *URJC*, pages 1–36, 2004.