UNIVERSIDAD REY JUAN CARLOS

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

Escuela Superior de Ciencias Experimentales y Tecnología

Curso académico 2004-2005

Proyecto Fin de Carrera

Localización de un robot con visión local.

Tutor: José María Cañas Plaza

Autor: José Alberto López Fernández

Para mi familia, amigos y mi novia.

Agradecimientos.

En primer lugar quería agradecer a José María Cañas, ante todo, su confianza en mi para la elaboración de este proyecto, el magnífico aporte que ha reportado trabajar con él, por sus conocimientos, su apoyo, su paciencia y su gran predisposición.

Dentro del grupo de robótica, agradecer también el apoyo de Pablo Barrera por facilitarme sus conocimientos y a mis compañeros Redouane, Pedro y Antonio por todo lo que nos hemos peleado juntos para llevar a cabo nuestros respectivos proyectos.

A mi familia por apoyarme siempre en todo, porque gracias a las facilidades que me han dado estoy ahora mismo escribiendo estas letras.

A mis amigos Rubén, Alberto, Javi, Juanan y sobretodo a mi novia Silvia por estar ahí siempre apoyándome en todos los momentos y por los buenísimos ratos juntos.

Carpediem

Índice general

Índice de figuras

Capítulo 1
Introducción

Uno de los problemas que se han abordado en el campo de la robótica móvil es el de la localización. La localización de robots móviles autónomos consiste en determinar la posición y orientación del robot dentro de su entorno. Se han desarrollado múltiples metodologías para la resolución de la localización. En este proyecto fin de carrera se han explorado dos de las técnicas probabilísticas para localizar un robot en función de lo que se observa con su sensor de visión, la cámara local.

En este capítulo se introducen los conceptos que dan contexto a este proyecto fin de carrera: una reseña histórica de los comienzos de la robótica en la época de la revolución industrial y su evolución a lo largo del tiempo. A continuación se introduce un pequeño resumen de qué es la localización en robótica y sus aplicaciones, así como las dos principales técnicas de localización que se han utilizado en este proyecto.

1.1. Robótica

La evolución del hombre en la industria siempre ha tenido un determinante fundamental y es el de cómo satisfacer sus necesidades a través de la ciencia y más en concreto, a través de la utilización de máquinas, es decir, de robots. La utilización de robots supondría mayor rapidez y precisión en las labores industriales.

En el siglo XIII aparecen los primeros autómatas (mecanismos cuyo objetivo era realizar tareas de forma mecánica conforme a un diseño). Según avanzaba la técnica se crearon las primeras máquinas autónomas o robots. Una obra checoslovaca publicada en 1921 por Karel Capek, denominada *Rossum's Universal Robots*, dio lugar al término "robot" proveniente de la palabra checa *robota*, que significa *trabajo forzado* y que se utilizó para nombrar a unas máquinas construidas por el hombre y dotadas de inteligencia que se ocupaban de los trabajos pesados.

Hoy en día disponemos de robots fundamentalmente en las industrias, conocidos como "robots manipuladores" y en centros de investigación conocidos como "robots

móviles". Los robots manipuladores nacen de las exigencias prácticas que tiene la producción en las industrias. Sin embargo los robot móviles nacen de la necesidad del hombre para resolver tareas más peligrosas y de investigación.

En la robótica industrial, los robots manipuladores, en un principio, eran controlados por humanos. Con el avance de la técnica estos robots fueron capaces de tomar decisiones por sí solos. En la figura 1.1 se muestra el primer tipo de brazo manipulador, el PUMA (*Programmable Universal Manipulator for Assembly).* Estos brazos robotizados eran capaces de realizar movimientos con mayor rapidez, precisión y potencia que el brazo humano.

(a) (b)

Figura 1.1: Brazo P.U.M.A. (a) y robot cargador (b)

Por otro lado, en la robótica móvil, para tareas de investigación se utilizan dos tipos de robots, los teleoperados, comandados a distancia por un operador humano y los autónomos, capaces de tomar decisiones de movimiento por sí mismos.

Los teleoperados se han usado desde su nacimiento para ejecutar tareas peligrosas como la desactivación de bombas, exploraciones en ambientes extremos y la exploración interplanetaria. En este último caso, recientemente, se encuentran las sondas americanas *Spirit y Opportunity.* Éstas se pueden considerar casi autónomas ya que, aunque se les indica qué tienen que hacer, son capaces de realizar autónomamente esas indicaciones.

Por otro lado, los robots móviles autónomos son caracterizados por su capacidad de toma de decisiones por sí solos a partir de la información que recogen del mundo en el que interactúan. Estos robots son utilizados principalmente en entornos cerrados, por ejemplo como enfermeros robóticos que recorren la planta de un hospital llevando a los enfermos la medicación o como guías de museos o bibliotecas. Este es el caso del robot

Minerva[1] (figura 1.2(b)), utilizado como guía en el *Smithsonian's National Museum of American History o Rhino* y que en la actualidad se encuentra ejerciendo esa labor en el *Deutsches Museum Bonn*. Con la evolución de los robots autónomos se han creado robots utilizados en espacios abiertos. Este es el caso de las cosechadoras automáticas.

(a) (b)

Figura 1.2: Sonda Opportunity (a) y Minerva (b)

En la actualidad la robótica ha avanzado en los modelos humanoides, tales como *QRIO* de Sony y *ASIMO* de Honda, consiguiendo comportamientos bípedos avanzados tales como andar y subir y bajar escaleras don dos piernas. También, en la actualidad, se introduce la robótica en el hogar como robots de servicio donde las grandes marcas como Honda, Sony, Fujitsu, Toyota son los principales creadores.

Los robots también han hecho su aparición en el sector de entretenimiento. Este es el caso de los robots *Lego Mindstorm* como kit de construcción y programación de robots con piezas *Lego*, los *perritos Aibo* de Sony, diseñados para su uso como mascotas en los hogares. Pero desde el año 1996 los *perritos* de Sony han pasado de ser simples mascotas a jugadores de fútbol en la competición mundial de robots que juegan al fútbol denominada Robocup[2]. Esta competición se inició con la intención de incentivar el interés por la ciencia y la tecnología ya que aporta un escenario de investigación tanto para la robótica como para la inteligencia artificial (IA).

[1] `http://www-2.cs.cmu.edu/minerva/`
[2] `http://www.robocup.org`

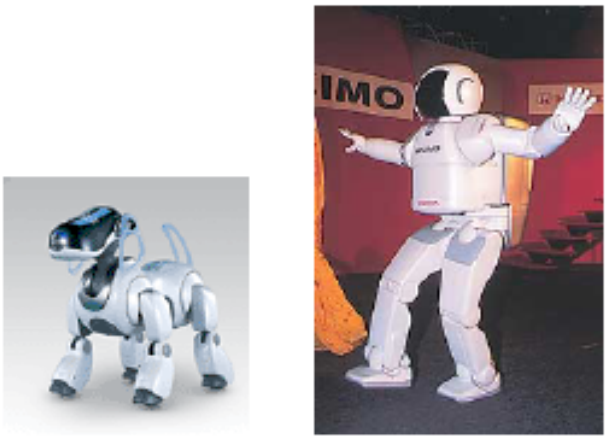


Figura 1.3: Perrito Aibo (a) y ASIMO (b)

1.2. Localización en robots móviles

Como se mencionaba en la sección anterior la localización en robots móviles es uno de los problemas que recientemente se han abordado. La localización consiste en determinar la posición y orientación del robot dentro de un entorno.

1.2.1. Visión global

En la localización se distinguen tres problemas fundamentales. El primero de ellos es resolver la localización *a partir de una posición inicial conocida*. La solución consiste en ir estimando la posición final del robot compensando los errores odométricos incrementales que acumulan los encoders del robot. El segundo problema, que se aborda en este proyecto fin de carrera, es la localización *a partir de una posición inicial desconocida*. Aquí es donde el robot debe manejar múltiples hipótesis a través de diferentes técnicas para determinar su posición y por ello los errores de estimación son mayores que en el caso anterior. En último lugar se encuentra la localización *de varios robots*. Este problema consiste en resolver la localización de un grupo de robots y que puede ser más interesante sí éstos pueden detectarse entre sí ya que existirían dependencias estadísticas en las estimaciones individuales de cada robot.

La localización se puede resolver a partir de *mapas* concretos y mediante la utilización de *sensores* que ayudan a determinar su posición. Una técnica de construcción de mapas es la conocida como SLAM (Simultaneous Localization and Mapping).

Para dotar al robot de autonomía en el movimiento y que el robot sea capaz de tomar sus propias decisiones es necesario almacenar la información del entorno a través

de mapas, en concreto, de *mapas de ocupación*. Estos mapas son necesarios para que el robot pueda tomar decisiones de cuánto avanzar, cuánto girar, a qué velocidad deben realizar los movimientos, etc. La localización se resuelve mediante la combinación de los mapas con la información sensorial.

La recogida de información de los sensores en los robots es un elemento indispensable en la resolución de la localización. Los principales sensores utilizados en localización son los *encoders* (figura 1.4(a)) que devuelven la posición del robot. Estos sensores presentan errores por desplazamiento, holguras, falta de precisión, etc. Este tipo de errores odométricos es necesario corregirlos ya que se hace más precisa la información de localización obtenida. Los errores son acumulativos y por ello si no se hace esta corrección periódica podemos llegar a obtener una localización demasiado imprecisa. Existen dos tipos de errores en la odometría: *sistemáticos* que dependen de las características del robot y sus sensores y *no sistemáticos* que son impredecibles y son debidos a agentes externos al robot.

Existen también sensores como las cámaras, los láser (figura 1.4 (c)), los sónar que por sí solos no dan información de posición y que para resolver el problema de la localización es necesario añadirles la información del mapa.

Otro tipo de sensor utilizado para resolver el problema de la localización es el sistema GPS (Global Positioning System, figura 1.4(b)). Este sistema está basado en el envío de coordenadas de posición y tiempo a receptores en la tierra a través de satélites. Este sistema es muy útil para la localización en exteriores ya que en interiores, como pueden ser los edificios, la señal proveniente de los satélites no entra con la suficiente fuerza. Este sistema es utilizado primordialmente para la localización y planificación de rutas en vehículos.

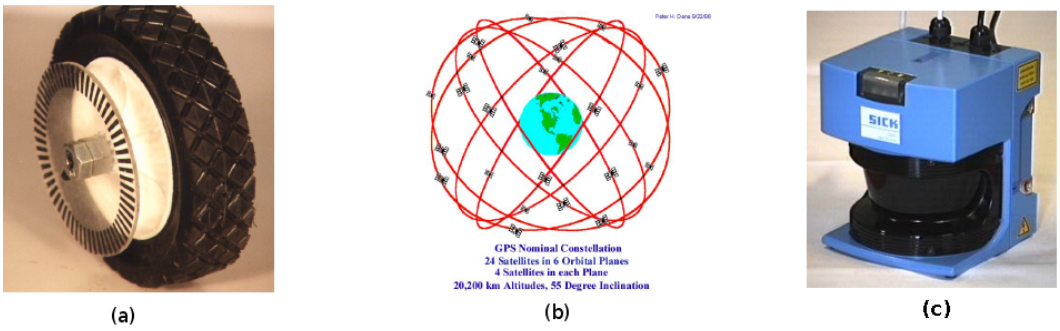


Figura 1.4: Sensores: Encoders (a), GPS (b) y Láser (c)

1.2.2. Técnicas de localización

Existe una gran cantidad de técnicas de localización que intentan resolver el problema de la localización sin el uso de sensores específicos:

- Localización con *balizas*: esta técnica permite localizar al robot en un entorno restringido mediante el emplazamiento específico de un determinado número de balizas con posiciones conocidas. Para estimar la posición se puede utilizar la *triangulación* basándose en el ángulo con que se ven las balizas y la *trilateración* basándose en la distancia a las balizas,

- Los *filtros de Kalman*: son otra técnica que trata de estimar recursiva y periódicamente la posición de mínima varianza fusionando información parcial e indirecta sobre localización. Su principal limitación es que es una técnica unimodal y exclusivamente gaussiana [Isard y Blake, 1998]. Esta técnica no soporta bien el ruido de lecturas sensoriales ni entornos dinámicos y no es capaz de manejar múltiples hipótesis,

- el *scan matching*: es otra técnica que utiliza mapas locales para compararlos con lecturas sensoriales alineando estas lecturas con los diferentes mapas en posiciones cercanas a la que creemos que está el robot, necesitando para ello una estimación de la posición inicial del robot representada como una distribución gaussiana que se va actualizando con las lecturas sensoriales,

- *localización probabilística* [Thrun, 2000]: es adecuada para interiores ya que incorpora incertidumbre de acciones y observaciones que se acoplan a la incertidumbre que muestran los sensores. Este tipo de localización consiste en determinar la probabilidad de que el robot se encuentre en una determinada posición a través de sus lecturas sensoriales y movimientos a lo largo del tiempo. A cada posible posición se le asocia una probabilidad reflejando la verosimilitud de ser la posición actual del robot. Esta probabilidad se va actualizando con la incorporación de nuevas lecturas y movimientos del robot. Estas técnicas nos permiten localizar al robot aún desconociendo su posición inicial permitiendo representar situaciones ambiguas que se irán desambiguando posteriormente. La eficiencia de estas técnicas generalmente depende del tamaño del mapa.

1.3. Localización probabilística visual

Dos de las técnicas probabilísticas más importantes y que se utilizan para resolver la localización en este proyecto son: la probabilística *sin muestro*, que

implica mucho tiempo de cómputo debido a que almacena y actualiza la distribución de probabilidades para todas las posibles posiciones y *con muestreo*, que se utilizan para agilizar el tiempo de cómputo y hacer de la localización un proceso escalable a grandes entornos.

La intuición de la localización probabilística se puede apreciar en la figura 1.5.

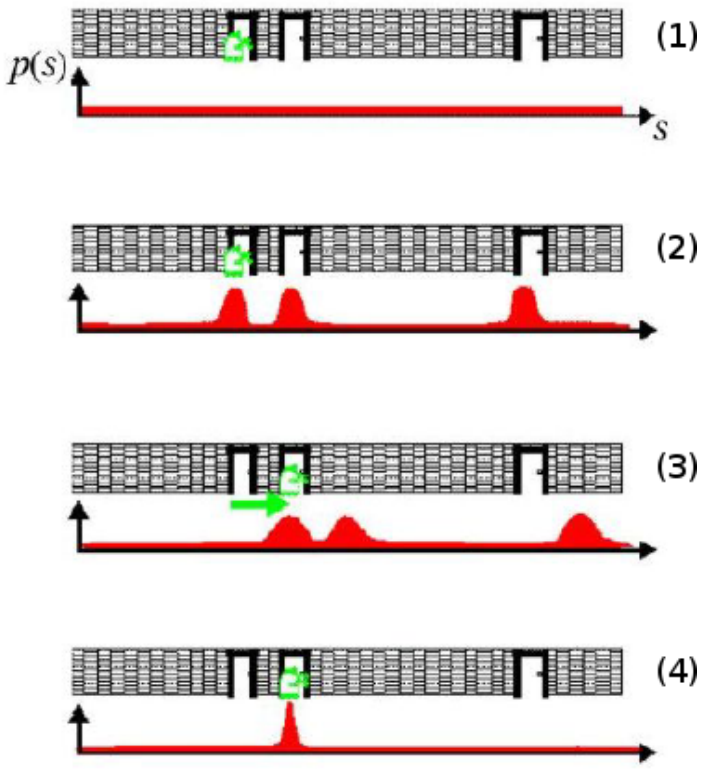


Figura 1.5: Localización probabilística

En un primer paso, se asume un espacio unidimensional en el que sólo se puede desplazar horizontalmente y se desconoce la posición inicial. El estado de incertidumbre se representa como una distribución uniforme sobre todas las posibles posiciones. En el siguiente paso, suponemos que el robot detecta mediante sus sensores que se encuentra enfrente de una puerta. Esto se refleja mediante el aumento de la verosimilitud en las zonas donde están las puertas y la disminución en zonas donde no hay puertas. La información disponible hasta el momento actual es insuficiente para poder determinar la posición del robot. Si el robot se desplaza, la distribución se desplaza de forma análoga como se refleja en el tercer paso. En el cuarto y último paso el robot ha detectado de nuevo, a través de sus sensores, que se encuentra frente a una puerta por ello aumenta la verosimilitud en esa posición que sumada a la acumulación del tercer paso, nos lleva a la conclusión de que es muy probable que el robot se encuentre en la segunda puerta.

En este proyecto se soluciona la localización mediante técnicas probabilísticas con y sin muestreo. Para la localización sin muestreo se utilizan *métodos de mallas de probabilidad* y para la localización con muestreo se utilizan *métodos de Montecarlo*, en concreto, los filtros de partículas.

La resolución de la localización mediante estas técnicas nace de las motivaciones del grupo de robótica de la Universidad Rey Juan Carlos por generar comportamientos artificiales en robots, tales como la visión artificial, estimación, lógica borrosa, inteligencia artificial, etc. Todos estos comportamientos son implementado en los diferentes robots de los que dispone el grupo (2 robot Pioneer, 6 robots EyeBot, 30 robots Lego y 10 perritos Aibo).

Una de las motivaciones del grupo es el equipo de robots que participan en la Robocup. La participación en el campeonato europeo *German Open 2004* y en el mundial *Osaka 2005* ha servido para ir mejorando el comportamiento y las técnicas futbolísticas de estos perros.

Otras de las motivaciones que se siguen en el grupo es la generación de comportamientos autónomos en entornos cerrados. Se han realizado muchos trabajos que implementan esta clase de comportamientos tales como el seguimiento de una persona en los pasillos de un edificio [Calvo, 2004], navegación local [Lobato, 2003], navegación global [Isado, 2005] [López, 2005] y localización [Crespo, 2003] [Benítez, 2004], donde también se encuadra este proyecto.

Este proyecto, junto con el de localización probabilística por láser [Kachach, 2005], nacen para poder resolver la navegación, que se había implementado en entornos de simulación [Lobato, 2003], [Isado, 2005], [López, 2005], en robots reales. Era necesario resolver la localización en robots reales para que los algoritmos de navegación pudieran portarse a la realidad. El entorno en el que se encuadra este proyecto es el propio entorno del grupo. El mapa corresponde con la planta del edificio donde se encuentra ubicado el grupo y el robot utilizado es un robot Pionner de los dos que disponemos.

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.

El objetivo principal de este proyecto es el desarrollo e implementación de algoritmos que resuelvan la localización en robots móviles. El robot navegará por el entorno y deberá ser capaz de localizarse dentro de él a través de observaciones sensoriales.

2.1. Descripción del problema

El objetivo principal se puede articular en varios subojetivos concretos que se desarrollan a lo largo de este proyecto. Estos subojetivos consisten en el desarrollo de algoritmos de localización a través de métodos de percepción probabilística sin muestreo (*mallas de probabilidad*) y con muestreo (*filtro de partículas*) y los experimentos pertinentes para verificar y optimizar los resultados.

- *Técnica probabilística sin muestreo*, se estudiará el comportamiento del robot y con qué precisión se localiza mediante el estudio del modelo sensorial y de movimiento y las reglas de fusión de evidencias.

- *Técnica probabilística con muestreo*, se estudiarán e implementarán técnicas de MonteCarlo, en concreto el filtro de partículas, para agilizar el proceso de localización. Se realizará el mismo estudio seguido para la implementación sin muestreo.

- *Experimentos*. Estos experimentos servirán para estudiar el comportamiento y los resultados de los algoritmos de localización probabilística anteriormente mencionados. Este estudio se facilitará mediante una interfaz gráfica que permite la visualización de ese comportamiento. Finalmente se compararán los resultados de ambas técnicas.

El punto de partida para el desarrollo de este proyecto es el robot Pioneer dotado con sensores de movimiento y de visión (cámara) y el entorno en el cual queremos que el robot se localice.

2.2. Requisitos

Varios de los requisitos básicos para este proyecto son la utilización del lenguaje C para la programación de los algoritmos bajo la plataforma JDE, que será descrita en el capítulo 3 y bajo un sistema operativo Linux. Es necesario también tener conocimientos básicos sobre visión computacional para facilitar el desarrollo de filtros de imagen y de estadística para la utilización de métodos probabilísticos.

Para localizarse, el robot cuenta con diferentes recursos: un mapa balizado del entorno del departamental II de la Universidad; una cámara que proporciona al robot imágenes de ese entorno; y los motores que permiten al robot desplazarse, girar y obtener información de posición.

A diferencia del proyecto fin de carrera [Crespo, 2003], donde no se exigía una implementación en el robot si no que se construía un entorno simulado, el requisito fundamental de este proyecto es aplicar estos algoritmos al robot real.

Para resolver este requisito es necesario que estos algoritmos funcionen capturando y procesando la información sensorial en tiempo real y se realice un desarrollo organizado en esquemas iterativos que estén continuamente procesando esa información. También es necesario que estos algoritmos sean robustos a cambios de iluminación que pueda presentar el entorno en la captura de imágenes reales.

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 aportamos flexibilidad en cuanto a cambio de requisitos.

En este tipo de modelo de desarrollo existen cuatro etapas que se pueden observar en la figura 2.1: **Análisis de requisitos, diseño e implementación, pruebas y planificación del próximo ciclo de desarrollo**.

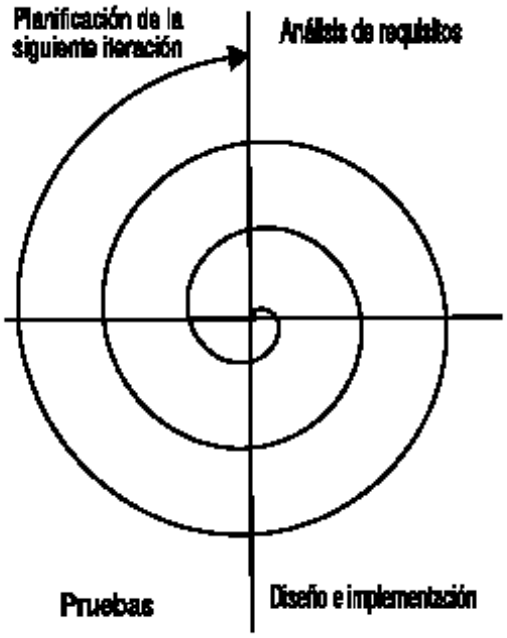


Figura 2.1: Modelo en espiral

A lo largo del tiempo de duración de este proyecto se han implantado reuniones periódicas con el tutor de este proyecto para establecer los puntos a llevar a cabo en cada etapa del mismo y comentar 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 que será descrita en el capítulo 3.

2. Estudio de las técnicas probabilísticas para el desarrollo de los algoritmos. Este estudio principalmente consiste en la lectura de documentos y el análisis y enfoque de los requisitos del proyecto sobre esas técnicas.

3. Diseño con esquemas de la implementación. Esto implica el estudio de la implementación de la plataforma JDE [Plaza, 2004].

4. Implementación de los esquemas diseñados anteriormente para los diferentes algoritmos de localización en lenguaje C.

5. Experimentos. Primeramente realizados sobre los simuladores y finalmente realizados sobre el robot real, comprobando su funcionamiento.

Capítulo 3
Entorno y plataforma de desarrollo

En este capítulo vamos a introducir la infraestructura hardware y software en la que se ha apoyado este proyecto fin de carrera, así como las bibliotecas auxiliares utilizadas.

3.1. Infraestructura hardware. Robot Pioneer

El *Pioneer* (figura 3.1(a)) es un robot fabricado por *ActivMedia*[1] de tamaño mediano para entornos de interiores. Sensorialmente está dotado de una corona de 16 ultrasonidos, unos *encoders* y adicionalmente se le ha añadido un láser *SICK*[2] para interiores. El *Pioneer* está compuesto de tres ruedas: dos motrices y una rueda loca. Es un robot que permite aplicaciones como construcción de mapas, navegación, etc. El cómputo necesario viene en un microcontrolador interno y un ordenador portátil que se coloca encima de la plataforma móvil. Ambos se comunican a través del puerto serie.

En nuestro proyecto necesitamos información de posición (x,y,Θ). Los *encoders* cuentan las vueltas que dan las ruedas del robot y trabajan con una resolución en mm/sg. Para sacar información de posición a partir de las vueltas contadas es necesario realizar una conversión [Garcia, 2002]. Adicionalmente, también le hemos añadido una cámara VIDERE[3] DCAM-L (figura 3.1(b)) en color para la captura de imágenes conectada al portátil por el puerto USB o el puerto FIREWIRE. Esta cámara presenta una resolución VGA pudiendo trabajar a una frecuencia de 30Hz (fps).

[1] http://www.activrobots.com/ROBOTS/index.html#p2dx
[2] http://www.sick.es/es/productos/autoident/medicion_laser/interior/es.html
[3] http://www.videredesign.com/products.htm

(a) Robot Pioneer

(b) VIDERE DCAM-L

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 nuestros Pioneers: *JDE*[4] [Plaza, 2003].

La plataforma plantea las aplicaciones robóticas como un conjunto de esquemas que se ejecutan simultáneamente y en paralelo y que realizan 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 esquemas perceptivos. Los esquemas pueden organizarse en niveles estableciendo una jerarquía entre esquemas padre y esquemas hijo siendo el padre el que pueda activar y desactivar a los esquemas hijo. En este proyecto se han diseñado y programado dos esquemas de percepción (*visualloc-mallas y visualloc-particulas*) y uno de servicio (*guivisualloc*).

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 3.1):

- robot real: las variables representan de forma directa a los sensores y actuadores del robot.
- simuladores: se utilizan principalmente dos simuladores: SRIsim de Shapira de-

[4]http://gsyc.escet.urjc.es/jmplaza/software.html

sarrollado en *SRI*[5] en conjunción con la biblioteca *ARIA* [ActivMedia, 2002][6] y un simulador proporcionado por la plataforma Player/Stage[7] [Brian P. Gerkey, 2003]. Estos simuladores son capaces de simular a la perfección un robot Pioneer y cualquier entorno donde éste se encuentre. La utilización de ambos simuladores depende de si existen obstáculos dinámicos, es decir, imprevistos, para lo que se utilizaría 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. El servidor *Oculo* se encarga de los sensores de imagen y los cuellos mecánicos que puedan existir en el robot.

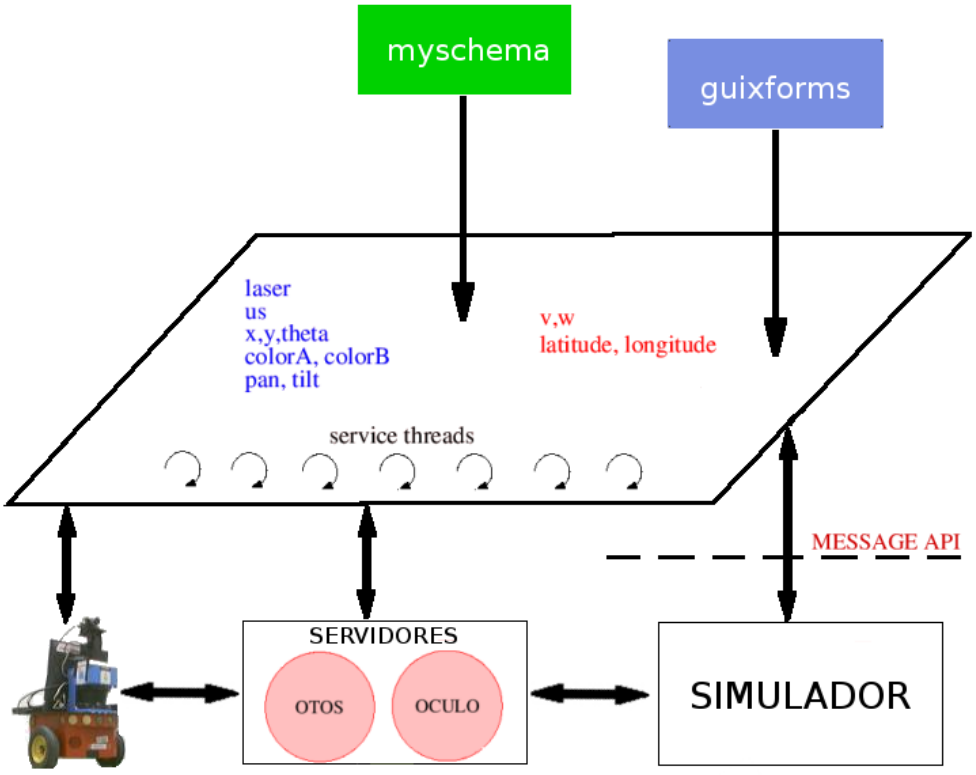


Figura 3.1: JDE Básico

La programación bajo JDE tiene ciertas desventajas como el retardo desde que se lee el dato hasta que llega al programa que realiza el procesamiento y una cierta desincronización de las medidas.

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

3.3. Bibliotecas auxiliares. Gridslib y XForms.

Las bibliotecas Gridslib y XForms se han utilizado en este proyecto para proporcionar diferentes funcionalidades a los esquemas programados.

Gridslib nos permite crear y manejar rejillas de ocupación. Esta biblioteca implementa diversas técnicas de construcción de mapas como la regla de Bayes, la regla Dempster-Shafer, rejillas histógramicas o rejillas borrosas. Esta biblioteca genera una rejilla cuadrada con celdas regulares de cierto tamaño. Estos tamaños son especificados en un archivo de configuración que la biblioteca se encarga de leer.

En nuestro proyecto esta biblioteca nos va servir para representar el mapa del escenario. Cada celda de la rejilla almacenará un valor que representará un punto del espacio. Estos valores pueden representar espacios vacíos, paredes, obstáculos.

La biblioteca XForms, basada en la librería Xlib, es utilizada por el esquema de servicio *guivisualloc* utilizado en este proyecto y nos permite crear una interfaz gráfica con la que visualizar y depurar los resultados de nuestro proyecto. Esta biblioteca proporciona objetos gráficos como botones, diales, barras de desplazamiento, menús, etc. para la creación de ventanas en sistemas X Window.

Capítulo 4
Localización probabilística con mallas de probabilidad

La localización, como mencionamos anteriormente, es uno de los problemas más comunes que abarca la robótica móvil. La localización se resuelve de varias maneras como ya indicamos en el capítulo 1. Nuestro problema se centra en una técnica concreta que es la localización probabilística, consistente en determinar progresivamente una estimación de la posición del robot tras una sucesión de observaciones tanto odométricas como de visión. La eficiencia de este método dependerá del tamaño del mapa en el que se realiza la localización. Todo esto nos servirá para localizar al robot, incluso desconociendo su posición inicial.

En este capítulo veremos los fundamentos teóricos en los que se basa *modelo de mallas de probabilidad*, el diagrama de bloques en el que se estructura el algoritmo, los modelos de observación diseñados y cómo se ha programado este método.

4.1. Fundamentos teóricos de la localización con mallas de probabilidad

La localización se puede presentar de varias formas ante una situación real. El problema se puede plantear conociendo la posición inicial, lo cual nos daría una ligera idea inicial del recorrido que el robot puede ir tomando y hacer más probables las posiciones más cercanas a ese punto. Pero en nuestro caso desconocemos la posición inicial del robot, con lo cual inicialmente cualquier posición en el mundo es susceptible de ser la posición real del robot. Por esta razón, se han elegido métodos probabilísticos ya que históricamente son los que mejores soluciones han dado a este tipo de problema con el que nos enfrentamos.

La idea fundamental de la localización probabilística es la estimación y representación de las densidades de probabilidad asociadas a las posibles posiciones del robot en el mundo real. A cada posible posición en el espacio (x,y,Θ) se le asocia una probabilidad que refleja la verosimilitud de que sea la posición en la que realmente se encuentre

el robot.

Como información de entrada nuestro algoritmo dispone de lecturas sensoriales recibidas por el ojo del robot, la cámara (visión monocular), que es el modelo de observación visual y lecturas sensoriales de los actuadores del robot, el modelo de movimiento.

Contamos también como fuente de información con un mapa balizado del entorno. Al disponer de éste mapa, intuitivamente, si el robot observa por la cámara una puerta, las posiciones situadas frente a las puertas en ese mapa serán las que acumulen mayor probabilidad, mientras que las demás posiciones acumularán menor probabilidad.

El algoritmo de localización por mallas de probabilidad se divide en dos etapas que se ejecutan constantemente:

- *Modelo de movimiento*, se captura la información de movimiento medida por los encoders del robot.
- *Modelo de observación visual*, se captura toda la información que proporcionan las nuevas lecturas sensoriales provenientes de la cámara del robot y se calculan las nuevas densidades de probabilidad para las posiciones del espacio. Una vez capturada toda la información anterior se actualizan todas las evidencias acumuladas. A medida que el robot recibe nuevas observaciones sensoriales se acumula la información y se actualizan las probabilidades haciéndolas evolucionar. Las posiciones compatibles con la observación suben la acumulación y las incompatibles la bajan.

Para la actualización de las evidencias se utilizarán técnicas basadas en el *Teorema de Bayes* que nos permitirán efectuar la fusión de la información sensorial recibida para la acumulación de evidencias.

El conjunto de posibles posiciones se representa con una rejilla regular tridimensional que alberga las densidades de probabilidad calculándolas como:

$$p(x,y,\Theta)=p([obs1,obs2...obsN],\ [obs1,obs2...obs(N\text{-}1)]).$$

Más adelante se indicará más en profundidad el uso que se hace del *Teorema de Bayes* en nuestro problema, para la incorporación de la información sensorial.

4.2. Diseño general y algoritmo

El diseño de la implementación de la localización se resume en un esquema perceptivo que implementa el comportamiento general de la localización. Este esquema es apoyado por otro esquema de servicio que ayuda a la visualización gráfica de los

resultados obtenidos en cada momento del proceso mediante una interfaz gráfica.

El problema de la localización se ha resuelto con dos esquemas fundamentales: *guivisualloc y visualloc*. El primero se dedica a crear e ir alimentando y refrescando la interfaz gráfica como mencionamos en el párrafo anterior. El esquema *visualloc* de mallas de probabilidad se encarga primeramente de recoger toda la información del escenario y almacenarla en una rejilla manteniéndola fresca durante el desarrollo del algoritmo. Después de este primer paso se encarga de materializar el algoritmo que resuelve la localización. Utiliza las variables proporcionadas por la plataforma JDE para el desarrollo del algoritmo como se indica en la figura 4.1. Particularmente utiliza la información de los encoders (x,y,Θ) y la información proveniente de la cámara. Este esquema es lanzado por el usuario desde la interfaz gráfica.

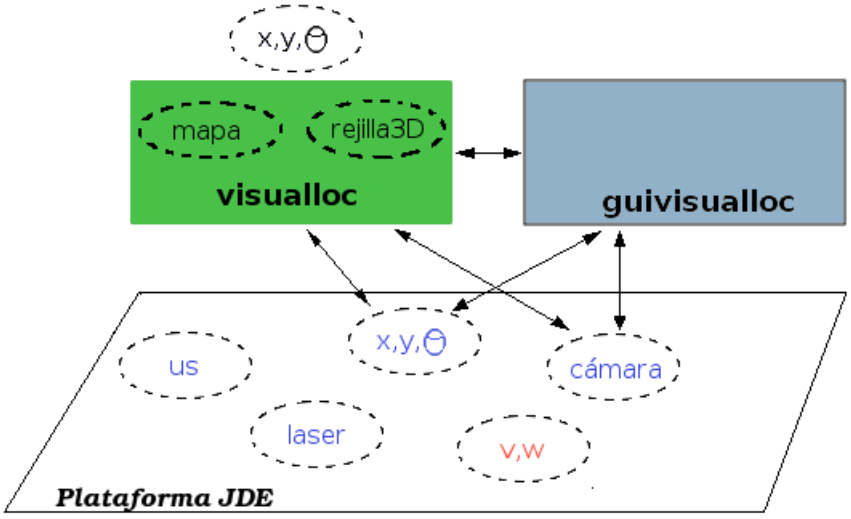


Figura 4.1: Implementación con esquemas de la localización con mallas de probabilidad

En la figura vemos como el esquema *visualloc* contiene el mapa visual del mundo que hemos implementado con una rejilla bidimensional para albergar la información del escenario y una rejilla tridimensional de probabilidad que sirve para almacenar las densidades de probabilidad por cada una de las posiciones válidas a partir de los modelos de observación dando pie a la fusión con la regla de Bayes que combina muchas probabilidades.

La finalidad de la localización es dar la posición (x,y,Θ) estimada de dónde se encuentra el robot. Siempre daremos como posición estimada la posición con mayor probabilidad en el cubo en cada iteración. Al principio de los tiempos habrá muchas posiciones posibles pero a medida que el algoritmo avanza se van descartando posiciones y va convergiendo en una única posición, a través de la regla de Bayes, que es donde se encuentre el robot.

Aquí, en este punto, es donde utilizaremos la *rejilla tridimensional de probabilidad* que representa todas las posibles posiciones y orientaciones dentro del mapa albergando la probabilidad de cada una de esas posiciones. Esta rejilla es visualizada junto con la nube de probabilidad que contiene alrededor en las posiciones más probables y que se va desplazando y rotando en función del modelo de movimiento del robot.

4.2.1. Esquema "visualloc"

Las funciones de este esquema *visualloc* para el *modelo de mallas de probabilidad* son:

1. Captura de la información.
2. Generación de la rejilla donde se almacena el mundo.
3. Localización.
   - Modelo de movimiento.
   - Modelo de observación visual y actualización de evidencias.

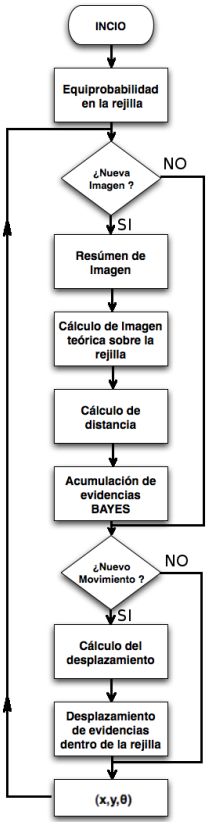


Figura 4.2: Pseudocódigo de localización con mallas de probabilidad

La información a capturar es la del mundo que es un mapa conocido del entorno, el cual cuenta con balizas visuales reales (figura 4.3). En nuestro caso las balizas que utilizaremos serán las puertas, una papelera azul en medio de uno de los pasillos y los extintores. Estas balizas están representadas en el mapa tal cual se encuentran en el mundo real. Estas balizas son representadas por colores dentro del mapa. Por ello el algoritmo recibe la información del mapa a través de una imagen en formato PNM y a color.

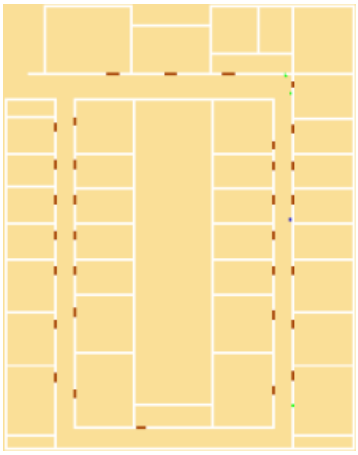


Figura 4.3: Mapa de balizas

Para la captura de la información del mapa se utiliza un analizador sintáctico que va leyendo linea a linea los siguientes archivos, almacenando la información necesaria:

- Imagen PNM a color del mapa (figura 4.3): este fichero contiene en las primeras cinco lineas, entre otras cosas, el *ancho* y el *alto* que ocupa la imagen. En las siguientes líneas contiene el chorro de bytes que componen la imagen.

- Fichero de configuración del mundo: este fichero contiene la *ruta* donde se encuentra la imagen, la *resolución* que tiene cada pixel dentro de esa imagen. Adicionalmente, como datos concretos para los simuladores, este fichero también contiene el *puerto* donde se lanzará el simulador y la posición inicial (x,y,Θ) del robot en ese mundo. Pero en esta primera parte nos centraremos únicamente en la *ruta* y la *resolución*.

Toda esta información es almacenada en la siguiente estructura de datos:

```
typedef struct {
   int ancho;
   int alto;
   int dimension;
```

```
   float resolucion;
   float x_robot;
   float y_robot;
   float theta_robot;
}Tmundo;
```

Con la información de la imagen del mapa y el fichero del mundo hacemos uso de la biblioteca *gridslib.h* ya comentada en el capítulo 3, para que el esquema se construya un mapa interno en forma de rejilla bidimensional almacenando el color de cada pixel de la imagen por cada posición el color que le corresponde dentro de la rejilla.

- -90: para el color de las paredes.
- 110: para el color de las puertas.
- 120: para el color de los extintores.
- 60: para el color de la papelera.

Para tener el mapa almacenado es necesario anclar la imagen a través de los datos del fichero del mundo realizando conversiones entre tamaños de rejilla y el de los píxeles.

Más a bajo nivel, creamos un puntero a la estructura de datos *Tgrid* que es nuestra rejilla de ocupación. Dentro de esta estructura tenemos una gran cantidad de campos que albergarán distintos tipo de información para la rejilla.

Se procede a rellenar la rejilla con la información de ocupación del mapa. Este proceso lo realiza una función, dentro ya de nuestro algoritmo, llamada *relleno_grid* a la cual se le pasa la estructura *Tmundo* con toda la información recogida en la parte de captura del escenario y un puntero a la estructura *Tgrid*. Lo que hace el algoritmo en esta función es ir rellenando en el campo *map* una serie de valores en función del valor de cada pixel de la imagen que estamos leyendo. Esta tarea se apoya fundamentalmente en las ecuaciones anteriores.

Una vez que tenemos toda la información del escenario almacenada en la rejilla, pasamos a interactuar con el robot y sus lecturas sensoriales. Estas lecturas sensoriales nos van a ir aportando la información a tratar mediante nuestro algoritmo y conseguir una buena localización mediante.

Se utiliza una *rejilla tridimensional* para almacenar esta información en forma de evidencias para cada una de las posibles posiciones y orientaciones dentro del marco

del mundo en el que estamos. Esta afirmación no es del todo cierta ya que la rejilla tridimensional de probabilidad abarca todas las posiciones (x,y) que representan el entorno pero albergará para cada posición sólo dieciséis orientaciones distintas en el espacio angular de una circunferencia ($0..2\pi$).

Se han diseñado dos modelos de observación: visual y de movimiento. El primero nos da las lecturas provenientes de la cámara de la que está provista el robot y el segundo nos da las lecturas provenientes de los encoders del robot.

4.3. Modelo de observación visual

Este modelo es el que captura toda la información de posición desde la cámara dentro del marco probabilístico. Calculamos las probabilidades de estar en cada posición a la luz de la imagen observada. Este cálculo consiste en obtener la distancia entre la imagen real resumida y la imagen teórica calculada para cada posición. Esta distancia es el número de píxeles diferentes entre las imágenes, por lo tanto, si la distancia es baja la probabilidad de ser iguales será alta.

Se realizan varios pasos fundamentales hasta llegar al cálculo de las probabilidades: resumen de imagen observada, cálculo de imagen teórica para cada posición posible y cálculo de la distancia entre imágenes.

La cámara proporciona una imagen cruda de 320x240 píxeles que la plataforma nos presenta en una estructura de tipo array. El primer paso que realizamos es resumir esa imagen y obtener de ella los datos más relevantes. Lo que hacemos es realizar un filtrado por color de la imagen cruda. Este filtrado nos permite obtener los colores de las balizas visuales que utilizaremos, las puertas, la papelera y los extintores.

La imagen cruda está compuesta por colores en formato RGB (Red Green Blue) y los colores se forman a partir de la cantidad de rojo, verde y azul, respectivamente. El filtro que se aplica realiza una conversión a través de las ecuaciones XXX de este formato al modelo HSI ([Baena, 2003] [Barrera *et al.*, 2005]) que realiza un filtrado más robusto pudiendo obviar o no la iluminación de cada escena en cada momento ya que separa la iluminación de la información del color. Se establecen umbrales mínimos y máximos por cada componente (Hmin, Hmax, Smin, Smax, Imin e Imax) para la distinción de los colores. La conversión de componentes se realiza mediante las ecuaciones siguientes:

$$H = cos^{-1}\left(\frac{\frac{1}{2}\left[(R-G)+(R-B)\right]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right) \tag{4.1}$$

$$S = 1 - \frac{3}{(R+G+B)} min\,(R,G,B) \tag{4.2}$$

$$I = \frac{1}{3}(R+G+B) \tag{4.3}$$

Una vez realizado el filtrado (figura 4.4) obtenemos la imagen real filtrada del mismo tamaño que la imagen cruda con los colores que nos interesan. Este filtrado se almacena en una estructura del lenguaje C.

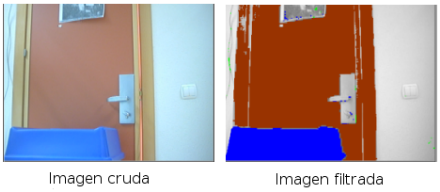


Figura 4.4: Filtrado de imagen 320x240

El siguiente paso es resumir la imagen filtrada a un espacio *unidimensional*. Este proceso consiste en ir recorriendo la imagen filtrada por columnas. Es decir, por cada columna vamos contando los píxeles de cada color, determinando si son suficientes o no como para que representen una baliza del mundo del que se están obteniendo las imágenes. Esta información de color se almacena en una nueva imagen, ahora de 320x1 píxeles que es nuestra imagen resumida (figura 4.5).

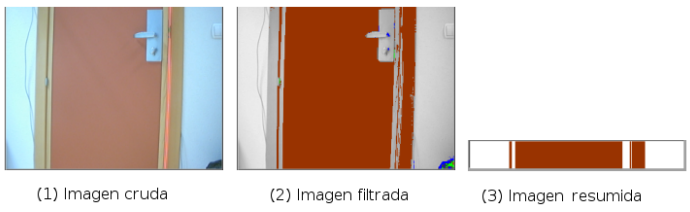


Figura 4.5: Filtrado de puerta

En el caso en que en una misma columna de la imagen se encuentren varios colores relevantes, se establecen las *prioridades de color* para determinar qué color es el relevante en la imagen. En la figura 4.6 podemos ver como se almacena en la imagen unidimensional el color azul que es más prioritario debido a que proporciona más información para nuestro algoritmo.

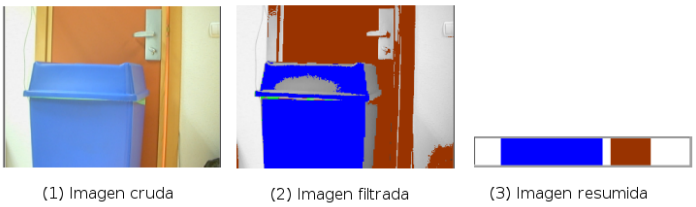


Figura 4.6: Filtrado de puerta y papelera (2)

Una vez realizado el resumen de la imagen real que obtenemos de la cámara, es necesario poder relacionar esta imagen simplificada con las posiciones. Para ello usamos el mapa extrayendo por cada posición una imagen teórica *unidimensional*. Esta imagen teórica se compara con la que hemos obtenido como resumen de la imagen de la cámara para determinar el parecido que tienen conformando así la probabilidad de ser o no posiciones válidas a la luz de la imagen real. Aquellas posiciones en las que la imagen real y su imagen teórica sean muy parecidas tendrán alta probabilidad, las que no sean parecidas tendrán baja probabilidad.

El algoritmo se encarga de calcular para cada posición una imagen teórica con un cono de visión determinado por el ángulo de visión de la cámara (en nuestro caso 48 grados), recorriendo la rejilla en la que está almacenado el mundo mediante la ecuación de la recta (*rectas de búsqueda*).

Inicialmente se calculan 48 puntos destino (B), uno por cada grado de los 48 posibles, a través de la ecuación:

$$B_x = A_x + RADIO\_VISION \cdot cos(\Theta) \tag{4.4}$$

$$B_y = A_y + RADIO\_VISION \cdot sin(\Theta) \tag{4.5}$$

siendo B el punto destino, A el punto origen, $RADIO\_VISION$ la longitud de alcance del cono de visión (se establecen *7 metros* conforme a la cámara utilizada) y Θ cada uno de los 48 grados que abarca el ángulo de visión de la cámara.

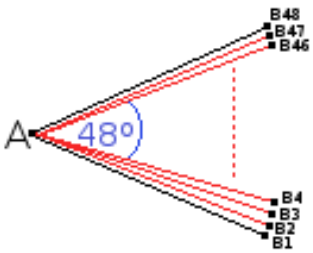


Figura 4.7: Creación de *rectas de búsqueda* dentro del cono de visión

$$P_x = A_x + \lambda(B_x - A_x) \tag{4.6}$$

$$P_y = A_y + \lambda(B_y - A_y) \tag{4.7}$$

Se calcula una recta desde el punto origen (A) donde queremos calcular la probabilidad, hasta el punto destino (B) por cada ángulo dentro de los 48 posibles que nos da la cámara, con una longitud igual al alcance de visión que tiene la cámara. El algoritmo irá recorriendo el cono de visión en el mapa punto por punto y recta por recta almacenando los colores relevantes. El color del suelo se considera transparente y los colores distintos de suelo son los colores relevantes.

Con estos datos se va construyendo una imagen teórica del mismo tamaño que la imagen real simplificada (320x1 píxeles). Cómo solo tenemos información de 48 ángulos se realiza un escalado de 48 a 320 que es el tamaño de la imagen.

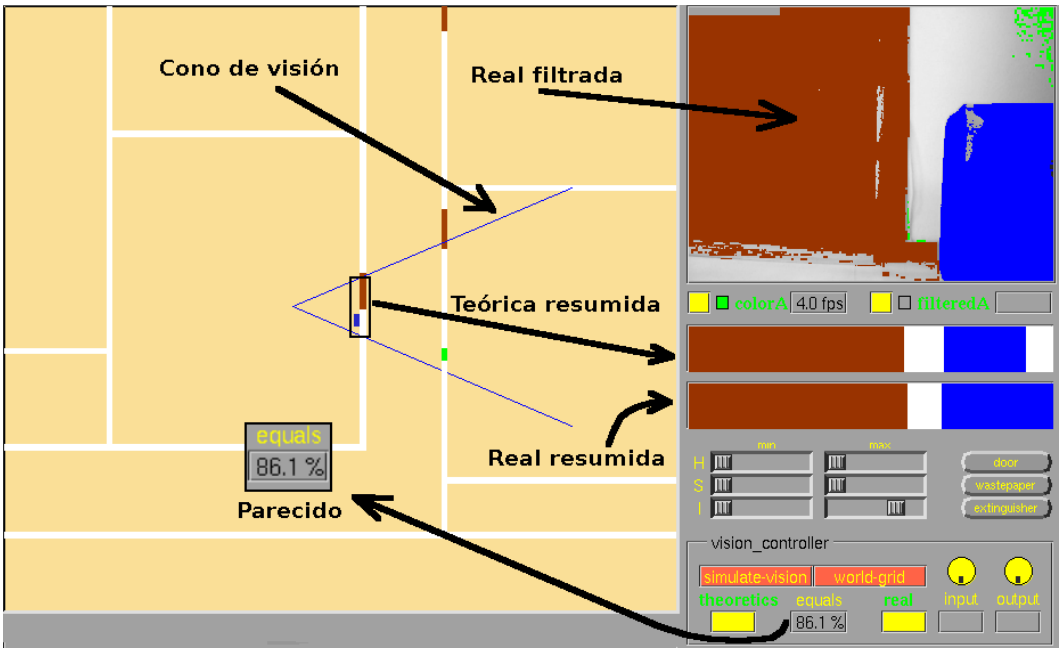


Figura 4.8: Cono de visión teórica. Comparación imágenes teórica y real

Una vez obtenida la imagen teórica unidimensional por cada posición se compara con la imagen real resumida obtenida anteriormente para obtener la probabilidad en cada posición. Esta comparación se realiza pixel a pixel. Se establece así una relación de distancia (número de píxeles diferentes) entre las imágenes y se calcula la probabilidad a través de la siguiente función:

$$p\left(x, y, \Theta / img_{real}\right) = \frac{1}{dist\left(img_{real}, img_{teorica}(x, y, \Theta)\right)} \tag{4.8}$$

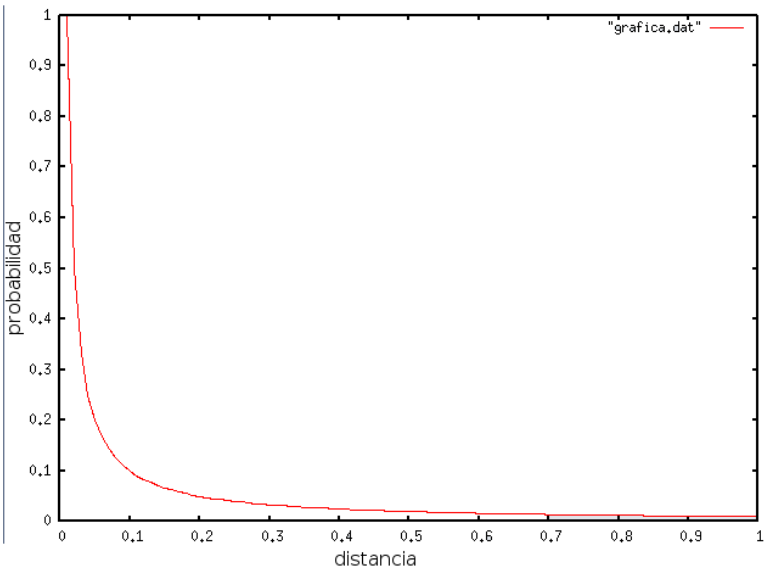


Figura 4.9: Verosimilitud de la posición en función de la distancia entre observaciones

La figura 4.10 muestra un ejemplo de la incorporación de observaciones visuales donde las posiciones más compatibles con la observación desde 0 grados ganan en probabilidad.

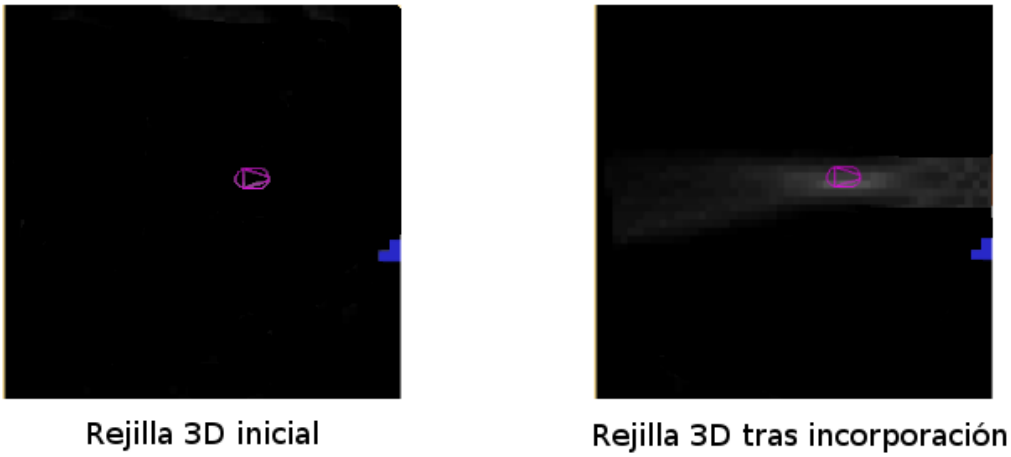


Figura 4.10: Incorporación de observación visual en 0 grados

Las incorporaciones de nuevas observaciones han de ser independientes. Una observación se considera independiente cuando se obtiene una nueva imagen y ésta es lo suficientemente distinta a la incorporada anteriormente como para aportar nuevas informaciones. Se considera también una observación independiente cuando la nueva imagen se ha tomado desde una posición lo suficientemente distinta a la posición donde se tomó la anterior imagen. En la figura 4.2 se puede apreciar como se introduce la independencia preguntando por separado por las observaciones visuales y odométricas.

4.4. Modelo de movimiento

Este modelo es el que captura la información de posición que proporcionan los sensores de odometría del robot que se usan para medir desplazamientos y giros relativos. Este modelo se materializa desplazando la nube de probabilidad de la rejilla con un movimiento homólogo al medido para el robot por los sensores de odometría.

Generalmente, en los algoritmos de localización se realiza un planteamiento clásico del modelo de movimiento para el cálculo de nuevas observaciones donde la probabilidad viene determinada por el estado anterior, la observación en el instante actual y las órdenes que se mandan a los actuadores del robot. Sin embargo, nuestro modelo de movimiento no ordena a los actuadores si no que toma observaciones de los encoders del robot en cada momento para poder calcular los desplazamientos que el robot realiza en cada iteración del algoritmo. Por este motivo, nuestro localizador es capaz de funcionar en paralelo con cualquier programa que sea capaz de mover el robot.

Este modelo nos permite incorporar en la malla de probabilidad los movimientos ejercidos por el robot. Los desplazamientos y rotaciones que ejerce el robot son deterministas y discretos debido a que no se aplica ningún tipo de ruido gaussiano. Si el robot se desplaza una determinada distancia o rota un determinado ángulo, las evidencias realizarán exactamente el mismo desplazamiento y/o rotación dentro de la rejilla tridimensional de probabilidad.

Por ejemplo, si anteriormente el robot estaba en la posición (x_0,y_0,Θ_0) y posteriormente el robot se encuentra en la posición (x_1,y_1,Θ_1), de lo que se trata es de incorporar a nuestra rejilla tridimensional ese propio desplazamiento, es decir, la nube de probabilidad de las posiciones se desplaza dentro de la rejilla en función de (x_0-x_1, y_0-y_1, Θ_0-Θ_1) desde la posición anterior.

Una vez calculado el desplazamiento en coordenadas absolutas que ha ejercido el robot, el cálculo de cuántas celdillas hay que desplazar se realiza a través de las siguientes ecuaciones:

$$\Delta r_x = \frac{\Delta m_x + \Delta m_x(\frac{Grid.resolucion}{2})}{Grid.resolucion} \tag{4.9}$$

$$\Delta r_y = \frac{\Delta m_y + \Delta m_y(\frac{Grid.resolucion}{2})}{Grid.resolucion} \tag{4.10}$$

$$\Delta r_\Theta = \frac{\Delta m_\Theta}{(\frac{umbral}{2})} \tag{4.11}$$

siendo Δr el número de celdillas que tenemos que desplazar, Δm el desplazamiento

en el mundo absoluto y *umbral* el giro mínimo que ha de efectuar el robot para incorporar un desplazamiento angular.

Veamos la intuición de estos cálculos. Como nosotros tenemos una rejilla tridimensional compuesta de celdillas que ocupan un determinado espacio en el mundo absoluto, esto es, la resolución, debemos utilizar ésta para saber si el desplazamiento ejercido por el robot realmente es mayor que una celdilla. Los movimientos menores que el tamaño de una celdilla no provocan desplazamiento de la nube de probabilidad, sin embargo, solo se desplaza el contenido cuando el movimiento es mayor que el tamaño de una celdilla.

Nuestra rejilla tridimensional es un array de celdillas. El algoritmo irá recorriendo este array por filas y por columnas y por cada celdilla calculamos el desplazamiento que debe sufrir el contenido. Estos desplazamientos no alteran la forma de la nube, solo la desplazan. Los desplazamientos que caen fuera de la malla se desechan y los nuevos se introducen con un valor inicial por defecto (probabilidad a priori). Para esto utilizamos la técnica del *double buffering* que consiste en tener siempre una rejilla actualizada mientras en una segunda rejilla se incorpora el movimiento y luego se intercambian. Esto se realiza así para no machacar el contenido de las celdillas.

Veamos el pseudocódigo que realiza esta función:

```
for(i=0;i<size);i++)
  for(j=0;j<size);j++)
    {
     new_i=i+A·_x;
     new_j=j+A_y;
     if ((new_i>=0)&&(new_i<size)
        &&(new_j>=0)&&(new_j<size))
        {/*La nueva posicion cae dentro de la rejilla tridimensional*/
          old_cell=j*size+i;
          new_cell=new_j*size+new_i;
          new_cell.probability=old_cell.probability;
        }
     else ; /* la nueva posicion cae fuera de la rejilla tridimensional */
```

Así una vez recorrida toda la rejilla se habrá terminado de incorporar dentro de ella el movimiento realizado por el robot.

Las capturas de observaciones a través del modelo de movimiento, como ocurría en el modelo de observación visual, también han de ser independientes. Para ello se

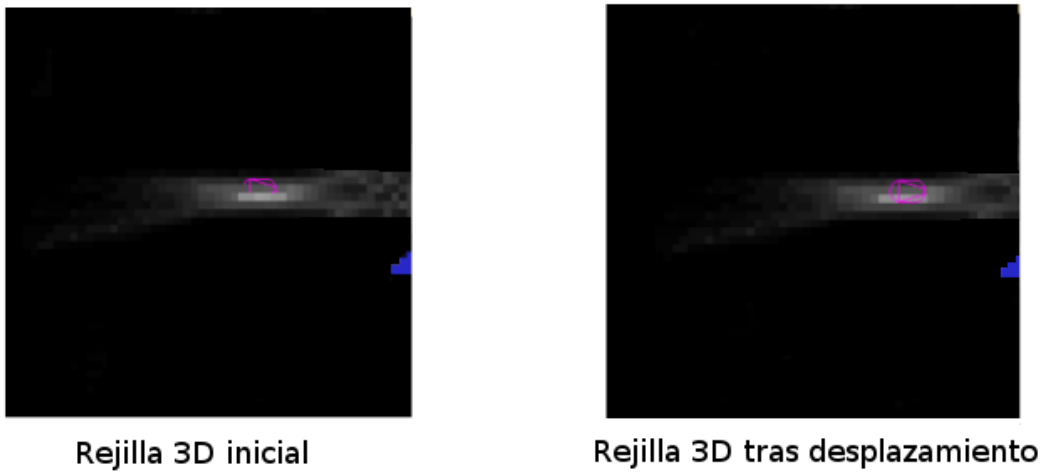


Figura 4.11: Incorporación de observación odométrica

determina un umbral de distancia recorrida por el robot, cuando el robot supera esa distancia se incorpora la nueva observación odométrica.

4.5. Acumulación de evidencias. Regla de Bayes

Con una sola observación no es suficiente para averiguar la posición. Es necesario realizar la acumulación de evidencias ya que con una sola observación muchas posiciones compatibles tendrán alta probabilidad y tenemos que ir eliminando la ambigüedad de las evidencias. Las posiciones candidatas malas van bajando al incorporar nuevas observaciones y las candidatas buenas van acumulando mayor probabilidad ya que son las posiciones compatibles con el flujo de observaciones.

El paso de *acumulación* de probabilidades se realiza apoyándonos en la regla de Bayes que se comporta de una forma robusta y sirve para fusionar todas las evidencias sensoriales que va recogiendo el robot a través de la cámara y la odometría. Esas evidencias se expresan como probabilidad siguiendo los modelos que hemos descrito. En las posiciones compatibles con las observaciones la probabilidad va a ir creciendo, en las incompatibles va decreciendo. Con la incorporación de varias observaciones y odométricas la esperanza es que la posición real sea la única compatible con ese conjunto de observaciones.

En la malla de probabilidad, inicialmente, se asigna para cada terna (x,y,Θ) en el espacio absoluto, una probabilidad 0.5 ya que en el primer instante el robot desconoce cuál es su ubicación y optamos por tener equiprobabilidad para todas las posiciones. A medida que se van obteniendo imágenes por la cámara, se irán calculando las nuevas

probabilidades.

Para realizar la fusión de evidencias se sigue el desarrollo matemático completo de la regla de Bayes [Thrun, 1997]. A través de este desarrollo se llega a una fórmula incremental (ecuación 4.12) que nos permite manejar la probabilidad acumulada a través de la última observación sensorial y de una forma muy sencilla. Para calcular la nueva probabilidad acumulada en el instante (t) fusionaremos los ratios de la probabilidad acumulada en el instante (t-1) y de la nueva probabilidad que nos da la observación de la cámara en el instante (t) mediante la relación 4.15 que se deduce del desarrollo de la regla de Bayes:

$$P(x, y, \Theta/obs_1, obs_2...obs_N) \sim P(x, y, \Theta/obs_1, obs_2...obs_{N-1}) \cdot P(x, y, \Theta/obs_N) \quad (4.12)$$

$$Pac(x_t) \sim Pac_{t-1} \cdot Pobs_t \quad (4.13)$$

$$\rho ac(x_t) = \rho ac_{t-1} \cdot \rho obs_t \quad (4.14)$$

$$\ln \rho ac(x_t) = \ln \rho ac_{t-1} + \ln \rho obs_t \quad (4.15)$$

Donde:

- P es la probabilidad de esa posición.
- ρ es el ratio de la probabilidad determinado por $\frac{P}{1-P}$.
- x_t representa la posición en el instante actual.

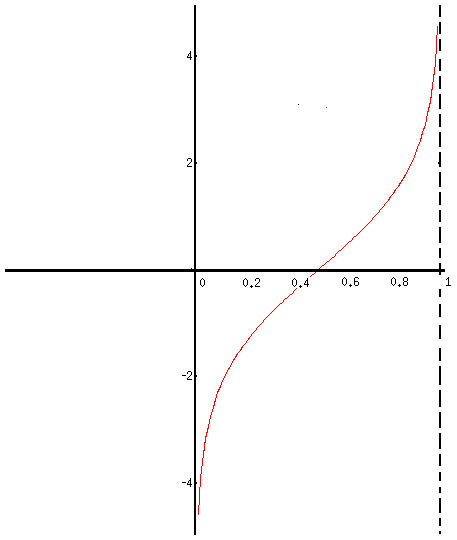


Figura 4.12: y = ln (x/(1-x))

Nos interesa manejar los logaritmos de los ratios ya que por la naturaleza de la función $\ln\frac{x}{1-x}$ en [0..1] aseguramos que si la probabilidad es mayor que 0.5 (valor a priori) su ratio es mayor que 1, su logaritmo positivo y por tanto la probabilidad acumulada sube con esta incorporación. Por el contrario si, para una observación, la probabilidad de una posición es menor que 0.5 su ratio es menor que 1 y su logaritmo negativo con lo que la probabilidad acumulada baja. Una vez aplicada la fusión con la regla de Bayes podemos asegurar que tras varias iteraciones del algoritmo se llega a estimar una buena localización. En la figura 4.14 podemos ver un ejemplo de todo el proceso de incorporación y acumulación de observaciones en el entorno expresado en la figura 4.13.

Figura 4.13: Imagen y situación de la puerta

La imagen inicial capturada corresponde a la puerta que está a la derecha de la rejilla tridimensional en el mapa. Teniendo en cuenta esta situación del entorno, veamos un ejemplo típico de evolución del algoritmo con un conjunto de observaciones y movimientos. En esta ejecución se visualiza toda la malla de probabilidad con los 16 ángulos posibles siendo el contenido de cada celdilla el máximo de probabilidad en cada corte.

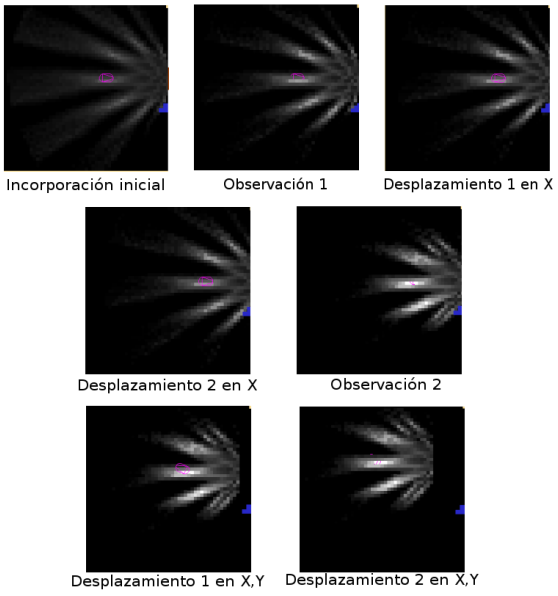


Figura 4.14: Ejecución tipica

Vemos en la primera parte de la secuencia la captura de una incorporación de la observación inicial. En color blanco se representan las posiciones más probables. Al visualizar todo el cubo de probabilidad vemos como se visualizan en un color más claro las posiciones con mayor probabilidad para todos los cortes del ángulo (recordemos que discretizamos en 16 orientaciones distintas). La visualización nos demuestra que la nube de probabilidad es congruente porque, teniendo en cuenta la observación que hemos incorporado, vemos cómo situándonos en esas posiciones probables y mirando en dirección a la orientación correspondiente estaríamos viendo una imagen muy parecida a la que hemos incorporado.

En la segunda captura se realiza una nueva incorporación de la misma imagen. Vemos como se incrementa la intensidad de color blanco, indicándonos la acumulación de evidencias. La nube va convergiendo hacia los puntos más probables, descartando posibles posiciones en el estado anterior que ahora no han aportado una buena probabilidad haciéndolas más oscuras en la visualización.

En las siguientes capturas el robot ejerce un desplazamiento positivo en X de 20cm respectivamente y se observa como la nube de probabilidad se ha desplazado también en esa dirección el número de celdillas correspondiente a 20cm en el mundo absoluto.

La siguiente captura se realiza una nueva incorporación de la observación de la cámara, que ahora será un poco distinta a las anteriores ya que el robot se ha desplazado. Vemos como al incorporar, la nube converge aún más y se hacen más claras las posiciones que van siendo más probables.

Las últimas dos capturas muestran un desplazamiento negativo en X y positivo en Y a la vez. El robot se ha movido en diagonal. Observemos cómo la nube de probabilidad también se desplaza en la misma dirección dentro de la rejilla tridimensional.

Como ampliaremos en el capítulo 6 este algoritmo resulta muy pesado computacionalmente. Debido a ello se han restringido los experimentos a la superficie de una habitación aproximadamente de $10m^2$. En esta superficie la rejilla tridimensional de probabilidad tarda en actualizarse aproximadamente 300 segundos con 16 orientaciones distintas. Si la superficie de la rejilla de probabilidad ocupara todo el mapa ese tiempo crecería exponencialmente.

4.6. Visualización.

Una vez tenemos toda la descripción informática del algoritmo de localización, hemos desarrollado un esquema de servicio (*guivisualloc*) para depurar y observar la evolución del algoritmo a partir de la visualización sus estructuras internas y la modificación de parámetros de funcionamiento.

Para ello la plataforma JDE proporciona un esquema de servicio *guixforms* que proporciona una interfaz gráfica utilizando la biblioteca "XForms". Para nuestra aplicación este esquema se ha modificado para ajustarlo a nuestras necesidades, quitando componentes y funciones que no son necesarios y agregando a la interfaz de componentes genuinos para nuestra aplicación.

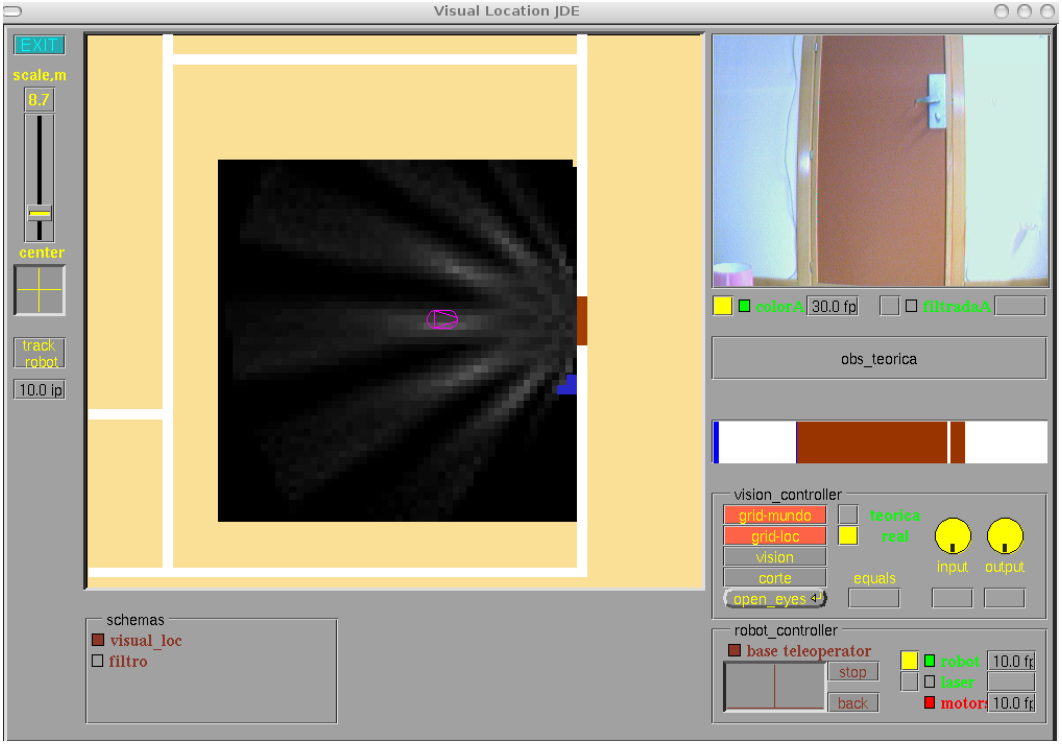


Figura 4.15: Interfaz gráfica ofrecida por *guvisualloc*.

Este esquema de servicio se ha implementado como una hebra iterativa que chequea los objetos de esta interfaz por si han sido pulsados por el usuario y en relación a ello realizar diferentes funciones. Entre las funciones esenciales se encuentran:

- Visualización de los datos sensoriales (provenientes del robot y la cámara).
- Visualización del mapa.

- Botones para permitir o no la visualización de los elementos anteriores (parte inferior derecha de la figura 4.15.
- Botón de activación del esquema perceptivo *visualloc* en la parte inferior izquierda de la figura 4.15.
- Joystick para poder teleoperar al robot dentro del entorno. "base teleoperator" en la figura 4.15
- Visualización de un cono de visión teórico el cual permite la posibilidad de elegir un punto del mapa y visualizar al instante la imagen teórica que se ve desde ese punto.
- Visualización de la imagen real resumida.
- Visualización del cubo de probabilidad por cortes (figuras 4.10 y 4.11) o entero visualizando el máximo valor de cada corte (figura 4.14).

Capítulo 5
Localización probabilística con filtro de partículas

Hasta ahora, para abordar el problema de la localización hemos utilizado la técnica probabilística basada en mallas de probabilidad. Esta técnica se encarga de hacer una estimación de la localización expresada como una función de densidad de probabilidad sobre todo el espacio de posibles estados. Esta técnica es muy robusta en espacios pequeños. Sin embargo, estas técnicas no muestran robustez cuando el espacio de posibles posiciones es demasiado grande ya que el coste computacional para mantener una estimación de probabilidad para todos los posibles estados dentro de ese espacio es demasiado grande e inmanejable.

Una técnica que busca solucionar ese crecimiento exponencial del coste computacional es la basada en métodos de *MonteCarlo*, también llamado *filtro de partículas*, para muestrear la distribución de probabilidad de todo el espacio de posibles localizaciones pero disminuyendo sustancialmente el coste computacional y sin perder representatividad.

A continuación se introducen los fundamentos teóricos en los que se basa el método de *MonteCarlo*, cómo se ha diseñado la implementación del algoritmo y el modelo de evolución y cómo se ha programado dentro de este proyecto.

5.1. Fundamentos teóricos del filtro de partículas

La idea de los filtros de partículas es la de mantener una representación muestreada de la distribución de probabilidad de manera no uniforme de las partículas. La estimación probabilística independiente de cada partícula viene determinada como la verosimilitud de ese estado a raíz de las observaciones sensoriales. Una muestra es una partícula, siendo ésta un punto (x,y,Θ) del espacio con un determinado peso en función de su probabilidad (p_i) determinada por la comparación de la imagen real con la imagen teórica asociada a cada partícula. El conjunto muestral representa las posibles posiciones del robot en el marco de referencia [Fox *et al.*, 1999] [Mackay, 1999] [Barrera

et al., 2005]

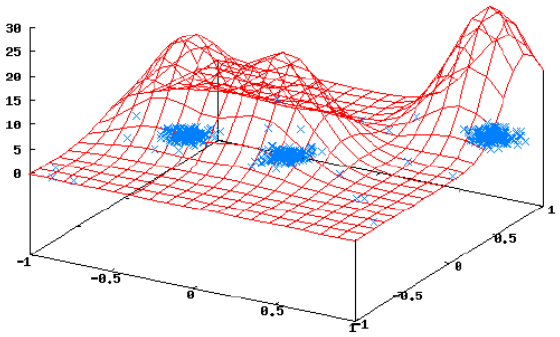


Figura 5.1: Muestreo de la función densidad de probabilidad

Resumidamente, se tiene una población de N muestras, cada una con un peso, que representan las posibles posiciones del robot. Se van generando nuevas poblaciones a partir de la población anterior junto con las imágenes que obtenemos por la cámara (modelo de observación visual) y los movimientos en (x,y,Θ) producidos por el robot (modelo de movimiento). Inicialmente el conjunto de muestras se distribuye uniformemente y de manera aleatoria por todo el espacio tridimensional abarcando las posibles posiciones del robot. Posteriormente los pesos y las posiciones de las partículas van variando a medida que se incorporan las observaciones visuales y odométricas. Las observaciones visuales suben el peso a las partículas compatibles con la observación, no su posición. Las observaciones odométricas trasladan las partículas siguiendo el desplazamiento y giro medido por los encoders. Gracias al paso de remuestreo, donde se utiliza el algoritmo de la ruleta, las partículas con más peso en la antigua población generan más hijos para la siguiente y eso lleva a un desplazamiento de las partículas hacia posiciones compatibles con las observaciones sensoriales.

El algoritmo realiza iteraciones continuas, cada una de ellas con tres pasos fundamentales en los que se basa el filtro de partículas (figura 5.3):

- *Modelo de movimiento*, se trasladan las muestras según el desplazamiento y giro ejercido llevado a cabo por el robot(Δ_r, Δ_Θ). La posición de las nuevas partículas vendrán determinadas por este desplazamiento.

- *Modelo de observación*, se calculan las nuevas probabilidades a posteriori de las muestras a partir de la última imagen captada.

- *Remuestreo*, se genera un nuevo conjunto muestral a partir de las verosimilitudes anteriores.

Con éste método no se guarda explícitamente la historia de las verosimilitudes del conjunto muestral en instantes anteriores, sino que va vinculada a las nuevas distribuciones de las muestras, las cuales se van concentrando alrededor de las posiciones más probables.

5.2. Diseño general y algoritmo

El diseño de la implementación del algoritmo es muy parecido al diseño que utilizábamos para el modelo de mallas de probabilidad. Este algoritmo se ha diseñado con dos esquemas JDE como se indica en la figura 5.2, uno perceptivo apoyado por otro de servicio que ayuda a depurar y visualizar resultados.

La diferencia entre éste método y el de mallas de probabilidad únicamente está en el algoritmo de localización. La infraestructura sensorial y el anclaje con el sistema JDE es el mismo. La distribución global en ambos métodos es análoga.

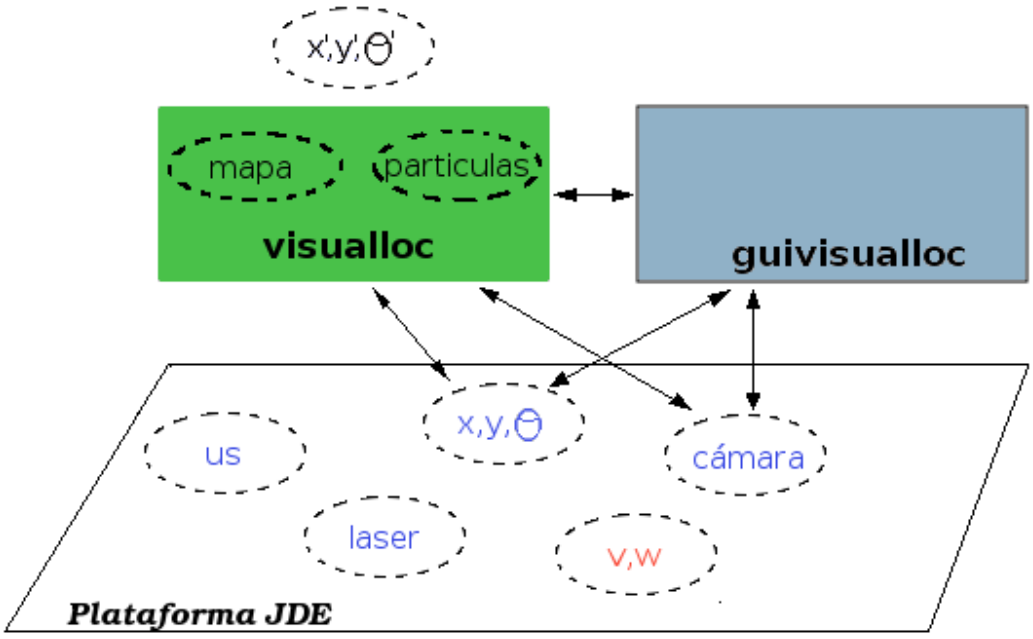


Figura 5.2: Esquemas. Filtro de partículas

Comparando esta figura con la del capítulo anterior observamos que el diseño en los esquemas es el mismo. El esquema *quivisualloc* sigue ofreciendo las mismas funcionalidades leyendo de los mismos sensores (encoders, cámaras) al igual que el esquema *visualloc*. Vemos cómo éste último sigue exportando las mismas variables (*x'*,*y'*,Θ*'*) y sigue teniendo un mapa visual del mundo. Fijémonos que la única diferencia es que este esquema en vez de utilizar una rejilla tridimensional para el cálculo de la posición

utilizará las partículas.

5.2.1. Esquema "visualloc"

Al igual que ocurría con el modelo de mallas de probabilidad el esquema *visualloc* se encarga de realizar las funciones del esquema global de la figura 5.3 que conforman el algoritmo de localización a través del filtro de partículas. Estas funciones son:

1. Captura de la información.
2. Generación de la rejilla bidimensional donde se almacena el mapa del mundo.
3. Localización con filtro de partículas.
   - *a*) Modelo de observación.
   - *b*) Modelo de movimiento.
   - *c*) Remuestreo.

La parte de captura de información y generación del mapa visual del mundo es idéntica a cómo se hacía para el modelo *de mallas de probabilidad*. Por ello haremos más hincapié en explicar el desarrollo del algoritmo del filtro de partículas y explicar las diferencias respecto al modelo *de mallas de probabilidad*. Veamos el pseudocódigo utilizado para la implementación de éste método:

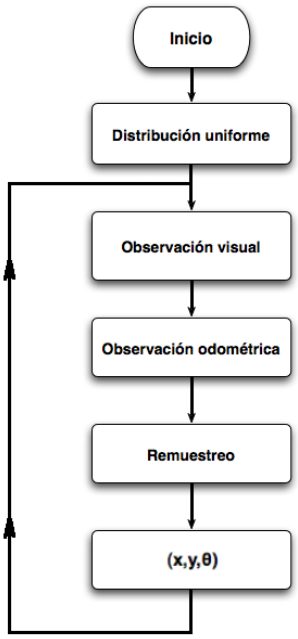


Figura 5.3: Pseudocódigo de localización con filtro de partículas

Como ocurría con el método basado en mallas de probabilidad, el algoritmo da continuamente una estimación de la posición de donde cree que el robot se encuentra

e incorporando nuevas observaciones y movimientos. La desviación típica que presenta la población de partículas es la que nos indica si la población converge alrededor de alguna posición concreta.

5.3. Modelo de observación

Nuevamente debemos capturar información de posición de la misma manera que hacemos para el modelo de mallas de probabilidad. Para ello, recurrimos de nuevo a la imagen que se obtiene de la cámara del robot. Contamos de igual manera con una imagen cruda (320x240 píxeles) en formato RGB. Convirtiendo la imagen a un espacio de colores HSI filtraremos la imagen para obtener los colores de los objetos relevantes de nuestro mundo que son las balizas visuales del entorno.

El siguiente paso, de nuevo, es resumir en una dimensión la imagen filtrada. Recordemos que este proceso consiste en recorrer la imagen por columnas contando los píxeles que son de colores relevantes. Si el número de píxeles supera un cierto umbral, lo incorporamos a la imagen resumida. Una ilustración de un filtrado simple viene detallada en las figuras 4.5 y 4.6. Una vez realizado el procesado de imagen el siguiente paso es el cálculo de probabilidad de cada partícula. Este cálculo se realizaba a través de la función de relación de la figura 4.9. Con el método del filtro de partículas se ha utilizado un proceso de cálculo de la probabilidad totalmente distinto.

La decisión de realizar un nuevo mecanismo de comparación entre imágenes se ha tomado debido a que la función de comparación pixel a pixel que se utilizaba para el modelo de mallas de probabilidad no funciona del todo bien ya que la influencia de un objeto de la imagen en la probabilidad de una posición es proporcional a su tamaño. Por ejemplo, los extintores que dan mucha información de posición, son muy estrechos en la imagen, pero sin embargo el modelo utilizado premiaba los estímulos anchos en la imagen más de lo que nos interesa. En este caso nos interesa premiar también los estímulos más estrechos que dan más información de posición.

Este nuevo mecanismo consiste en la segmentación en objetos de las imágenes. En primer lugar para que este mecanismo no sea muy pesado la segmentación se realiza únicamente sobre la imagen real. Esto se debe a que al disponer de N partículas se tienen N imágenes teóricas, una por cada partícula. Por ello se segmenta la imagen real y se compara con todas las imágenes teóricas.

La segmentación consiste en segmentar la imagen por colores, esto es, cada color

que encontremos en la imagen representa un objeto con ciertas características. Pueden existir objetos del mismo color en la imagen como se puede observar en la imagen real unidimensional de la figura 5.4:

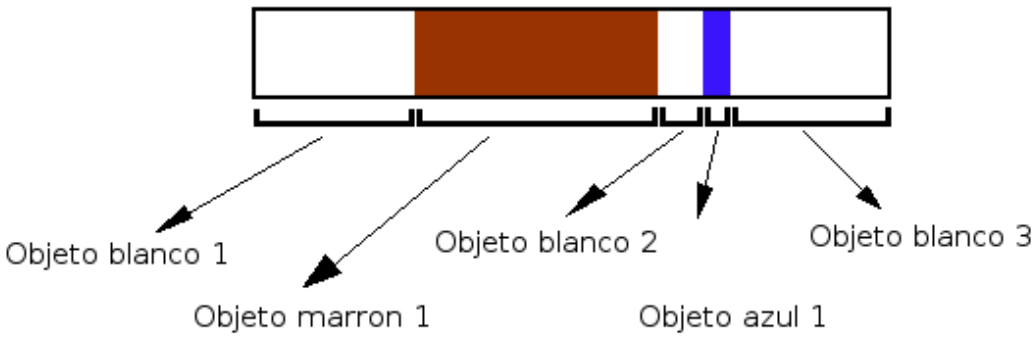


Figura 5.4: Segmentación en objetos de una imagen.

Estos objetos tienen una serie de características tales como:

- *Color del objeto.*
- *Pixel inicial.* Indica en qué columna de la imagen comienza el objeto
- *Pixel final.* Indica en qué columna de la imagen termina el objeto
- *Tamaño.* Nos indica el número de píxeles que este objeto ocupa dentro de la imagen (pixel final - pixel inicial).
- *Peso.* Nos indica el parecido que tiene el objeto en la imagen teórica.

Al no saber nunca cuántos objetos pueden formar una imagen utilizamos un array dinámico para albergar los objetos con sus características. La estructura de los objetos en lenguaje C es la siguiente:

```
typedef struct{
  char colour; //color del objeto
  int max, min; //Píxel inicial y final
  float weight; //Peso parcial
  float total_weight; //Peso total
}object_type;
```

Una vez obtenidos los objetos en los que se segmenta la imagen con todas sus características debemos calcular el grado de solape en la imagen teórica. Para contemplar pequeños desplazamientos de los objetos entre objetos se van a utilizar “colas” en la

comparación. Por ejemplo, como se ilustra en la figura 5.5, si el tamaño de la cola es de 5 píxeles y el tamaño de un objeto es de 50 píxeles que van desde el pixel 25 hasta el pixel 75, lo que haremos con las çolas"es calcular ese parecido en la imagen teórica con los 50 píxeles que ocupa el objeto en la imagen real más dos veces el tamaño de la cola (desde el pixel 20 al 80).

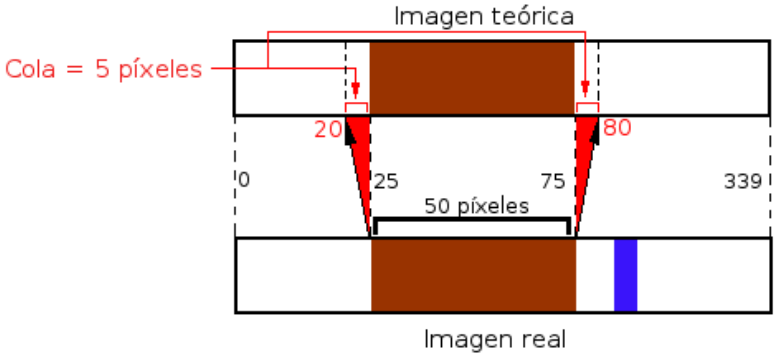


Figura 5.5: Comparación de objetos con colas

Con esto conseguimos que el cálculo de distancia permita breves desplazamientos de los objetos entre las imágenes, evitando que por un desplazamiento mínimo de píxeles de un cierto color pasemos de tener una muy buena probabilidad a tener una muy mala probabilidad. Este mecanismo aporta flexibilidad a la comparación de imágenes en el modelo de observación visual.

Debido a lo discriminantes o no que pueden ser las balizas de ciertos colores, como por ejemplo la papelera azul o los extintores (debido a su escasez en el entorno), no todos los objetos aportarán de la misma manera al peso final de la imagen. Por ejemplo, en nuestro entorno de localización nos interesaría divisar un extintor desde una distancia lejana. El color filtrado del extintor ocuparía muy pocos píxeles en la imagen debido a la lejanía con la que la cámara capta el extintor, pero sabemos que los extintores son escasos en el entorno y nos aportan bastante información de posibles localizaciones del robot. Por ello, la semejanza total entre las imágenes viene determinada *ponderadamente* por las semejanzas individuales de cada objeto. Por ejemplo, en condiciones normales, los extintores aportan su similitud a la estimación total (100 %) en un peso del 40 %, la papelera un 30 %, las puertas un 20 % y los objetos blancos el 10 % restante. Así aseguramos que los extintores y la papelera que puedan aparecer en la imagen y que aportan mucha información de posición, eleven la probabilidad de las partículas compatibles con las observaciones visuales.

$$p_{x,y,\Theta/imagen} = \frac{1}{N} \sum_{i=0}^{N-1} p_i \cdot Peso_i \tag{5.1}$$

En la ecuación, N es el número de objetos en la imagen, p_i es el grado de solape en magnitud de probabilidad de cada objeto y $peso_i$ es la parte proporcional del objeto en la imagen ($\frac{1}{N}$).

En el caso de los objetos blancos no siempre aportan su solape en la misma medida. Esto se debe a que hay varias situaciones en que es necesario penalizar los solapes de los demás colores. En la figuras 5.6 y 5.7 se puede apreciar gráficamente varias de estas situaciones:

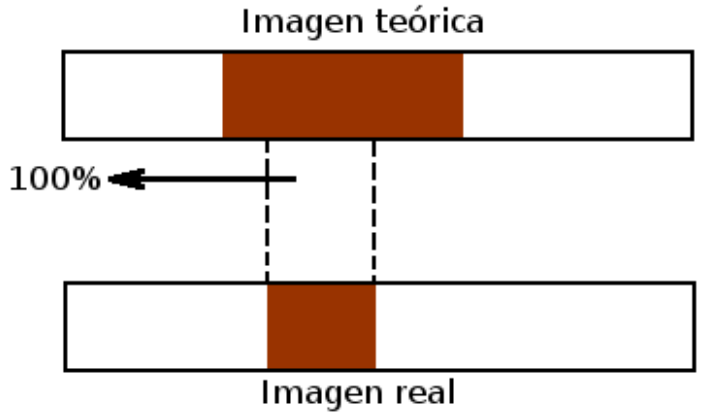


Figura 5.6: Caso excepcional de objetos blancos (1)

Realizando todo el proceso de segmentación normal en la figura 5.6 observamos como el objeto marrón de la imagen real solapa en un 100 % con el objeto marrón de la imagen teórica porque todos los píxeles que ocupa el objeto real solapan en la teórica. Pero vemos como las dos imágenes no son iguales, por ello hay que penalizar ese 100 % de solape teniendo en cuenta los objetos blancos adyacentes.

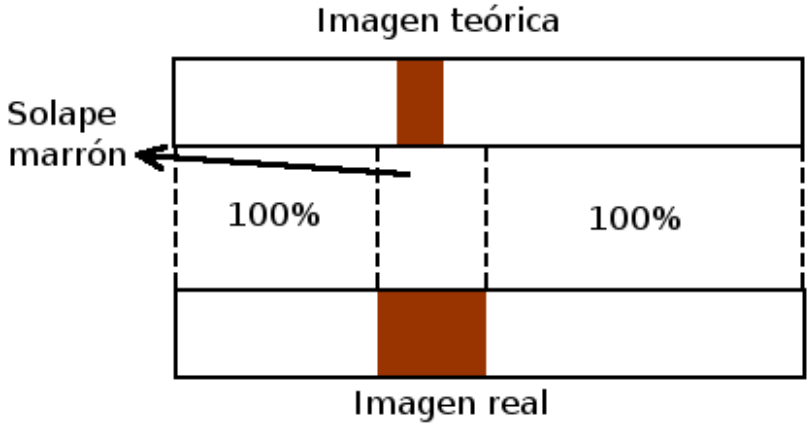


Figura 5.7: Caso excepcional de objetos blancos (2)

En la figura 5.7 observamos que si el grado de solape de los objetos blancos es del 100 % no se tiene en cuenta. Con lo cual la probabilidad total será determinada por el

grado de solape de los demás objetos a través de la ecuación 5.1.

Como veremos en los experimentos, esta función de comparación proporciona mejores resultados que la función de comparación pixel a pixel.

5.4. Modelo de movimiento

Al igual que en el capítulo 4 este modelo depende de las lecturas sensoriales del robot y no de las órdenes que se envíen a los encoders.

La incorporación del movimiento consiste en calcular el desplazamiento que ha ejercido el robot desde la última incorporación y aplicar ese desplazamiento a las partículas. En el caso de las partículas la incorporación del desplazamiento es directa, se desplazan y rotan de la misma manera que lo hace el robo. Este modelo es un modelo aleatorio al cual se le aplica también un ruido gaussiano evitando así que una partícula genere partículas en la siguiente población exactamente en la misma posición.

Se calcula el desplazamiento $(\Delta r, \Delta\Theta)$ del robot a través de las siguientes ecuaciones:

$$\Delta r = Mov_{obs} + Rudio_{lineal} \tag{5.2}$$

$$\Delta\Theta = Giro_{obs} + Ruido_{angular} \tag{5.3}$$

$$Mov_{obs} = \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2} \tag{5.4}$$

$$Giro_{obs} = (\Theta_t - \Theta_{t-1}) \tag{5.5}$$

$$Ruido_{lineal} = \Delta_r \tag{5.6}$$

$$Ruido_{angular} = \Delta_\Theta \tag{5.7}$$

y a cada posición de la partícula se le aplica el desplazamiento en pequeños incrementos para ganar suavidad en el movimiento de la población. Esto se consigue añadiendo independencia entre observaciones. Es muy importante recalcar que cada partícula se desplaza en la dirección que indica su orientación ya que cada partícula representa la posición y la orientación de dónde el robot puede estar. La secuencia es muy importante: se calcula el desplazamiento lineal y se aplica para después incorporar el desplazamiento angular. Así, con este modelo, aseguramos que las partículas se moverán conforme a la creencia que tiene cada una independientemente del resto.

En la figura 5.9 tenemos en rojo las partículas “1” y “2” en las posiciones (x_1,y_1,Θ_1) y (x_2,y_2,Θ_2) respectivamente en (t-1). Una vez calculado el desplazamiento del robot,

en primer lugar la partícula se desplaza Δr en la orientación que tenía en (t-1). En segundo lugar se realiza el desplazamiento angular sumando .Θ a la orientación en (t-1). Las nuevas posiciones en (t) serán (x_1,y_1,Θ_1) y (x_2,y_2,Θ_2) que se marcan en color azul.

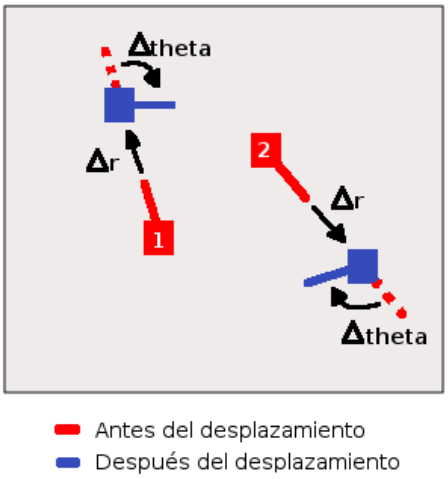


Figura 5.8: Ejemplo de desplazamiento de partículas

La información de movimiento ha de ser independiente de la información de observación visual. Esto quiere decir que cada vez que haya una nueva observación odométrica se incorpora sin tener en cuenta si existe o no una nueva imagen que incorporar. Contamos con un umbral de distancia máxima para incorporar una observación de odometría. Cuando se supera ese umbral se incorpora el movimiento. Por ejemplo, si el umbral es de 20cm y el robot se ha movido esa distancia, las partículas se desplazan también 20cm. Mientras no se supere ese umbral, no se incorpora el desplazamiento. En cada iteración del algoritmo se pregunta si se ha superado ese umbral incorporando el desplazamiento a la población de partículas cuando se haya superado.

5.5. Remuestreo

Para que la localización sea correcta con éste método, es necesario que la nube de partículas vaya convergiendo a las posiciones más probables. Para conseguir esto es necesario, a parte de el modelo de observación y el de movimiento, el paso de remuestreo. Este paso es el que aporta la inteligencia a nuestro algoritmo generando poblaciones de partículas cada vez más concentradas alrededor de las posiciones compatibles.

La idea del remuestreo es la de generar nuevas poblaciones a partir de poblaciones anteriores tal que las partículas que acumulan mayor probabilidad tengan más opciones de propagarse a la siguiente población. Esto se implementa con el algoritmo de la ruleta. La implementación se ha realizado mediante el muestreo aleatorio y uniforme en el eje Y de la acumulación de probabilidad de las partículas, seleccionando como partícula para la siguiente generación la correspondiente en el eje X (figura 5.9). Una misma partícula puede generar muchos hijos iguales en la población siguiente. En este caso, es la aplicación del modelo de movimiento con ruido gaussiano lo que hace que esos hijos se separan un poco unos de otros y exploren más zonas del espacio.

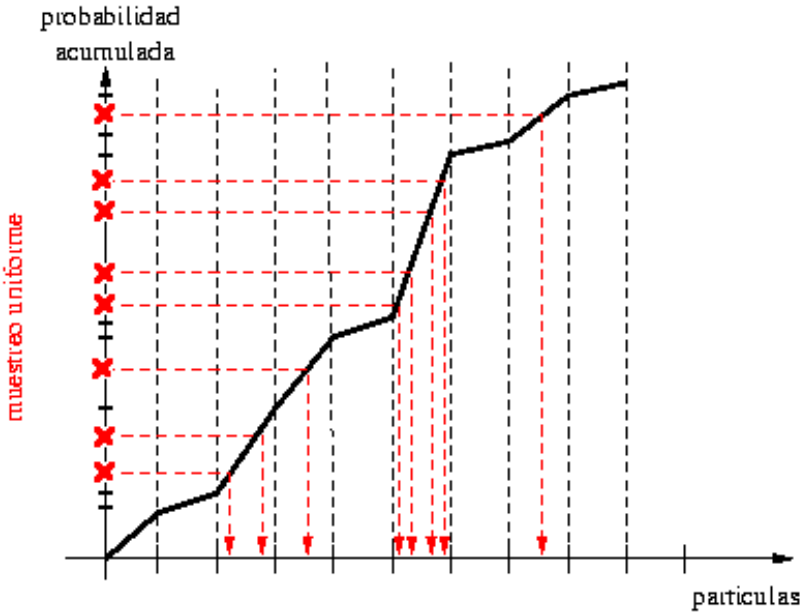


Figura 5.9: Remuestreo de las partículas

A nivel de código se tienen dos estructuras de tipo array que almacenan la población actual de partículas y la acumulación de esa población respectivamente. Este último array (ordenado) de acumulación almacena en cada posición, junto con las partículas que componen la población actual, la suma de todas las probabilidades de las posiciones anteriores. En el paso de remuestreo, al realizar el muestro aleatorio sobre el eje de acumulación, se realiza una búsqueda binaria en el array ordenado de acumulación para ver que partícula debe propagarse a la siguiente población. Estas partículas se almacenan de nuevo en el array de partículas actual representante de la nueva población.

Inicialmente la población de partículas se distribuye de manera aleatoria y uniforme dentro del espacio asignando equiprobabilidad a todas las partículas. A medida que el algoritmo va iterando, se van realizando los pasos de observación y remuestreo. Las observaciones influyen en varias iteraciones del algoritmo, no como ocurría con

las mallas de probabilidad. En cada paso de remuestreo se va generando una nueva población de partículas cada vez más concentrada alrededor de las posiciones que son más compatibles con la imagen.

En el caso de que la distribución de probabilidad de toda la población sea muy baja, se dice que la población ha degenerado. En tal caso se volverá, como inicialmente, a distribuir aleatoria y uniformemente una población de partículas sobre el espacio.

Se tiene continuamente una estimación de posición que mejora con cada iteración. Esta estimación se realiza o bien a través de la media aritmética de las posiciones de las partículas, o bien a través de la posición de la partícula con probabilidad más alta. Para matizar la validez de esta estimación utilizamos la desviación típica (δ) que presenta la población de partículas.

$$\delta^2 = \sum_{i=0}^{N-1} (X_i - X)^2 + \sum_{i=0}^{N-1} (Y_i - Y)^2 \tag{5.8}$$

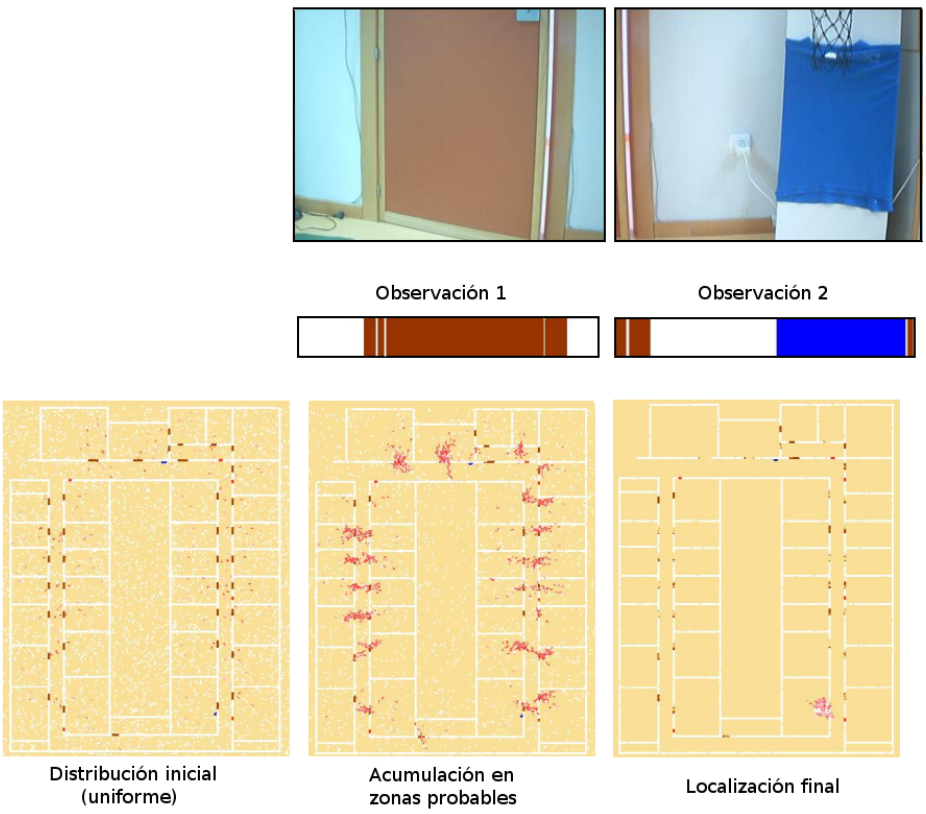


Figura 5.10: Ejecución típica del filtro de partículas

Visto esto veamos una ejecución típica del algoritmo (figura 5.10). Inicialmente se distribuyen uniformemente las partículas por todo el espacio presentando distintos colores en función de la probabilidad tienen. Tras incorporar la *observación 1* (con una puerta en la imagen) vemos como las nubes de partículas se posicionan frente a las puertas que según la observación son las zonas más probables. Cómo no tenemos más información que esa observación, cualquier posición frente a las puertas es probable. Cuando se incorpora la *observación 2* (objeto azul simulado con una camiseta) las partículas se concentran en una sola nube alrededor de la posición más probable. En este caso como la observación ha sido de la papelera azul, que es muy discriminante, todas las partículas se concentran alrededor de la posición de la papelera que, tras las observaciones anteriores, es la más probable.

Con dos observaciones muy discriminantes el robot se localiza enseguida, sin embargo, cuando no son tan discriminantes se quedan pequeñas nubes alrededor de las posiciones compatibles como veremos en el capítulo 6.

El paso de remuestreo, junto con los modelos de observación se ejecutan en cada iteración. Nuestro esquema iterativo trabaja a 10 iteraciones por segundo. Ésto, junto con el bajo coste computacional que tiene el algoritmo, hace que la localización sea bastante rápida y por tanto escalable a grandes entornos.

5.6. Visualización

Igual que en el modelo de mallas de probabilidad, necesitamos depurar y observar la evolución de las partículas. Para ello utilizamos el esquema de servicio *gui-visualloc* utilizado en la localización a través de mallas de probabilidad. El esquema funciona de la misma manera que en el capítulo 4. En este caso se han añadido funcionalidades específicas para el filtro de partículas.

Los elementos comunes tales como visualización del mapa, visualización de sensores, visualización del cono de visión teórico y joystick se mantienen en la interfaz. Se mantienen también los objetos que se utilizaban para el modelo de mallas de probabilidad para poder elegir el algoritmo a utilizar para la localización.

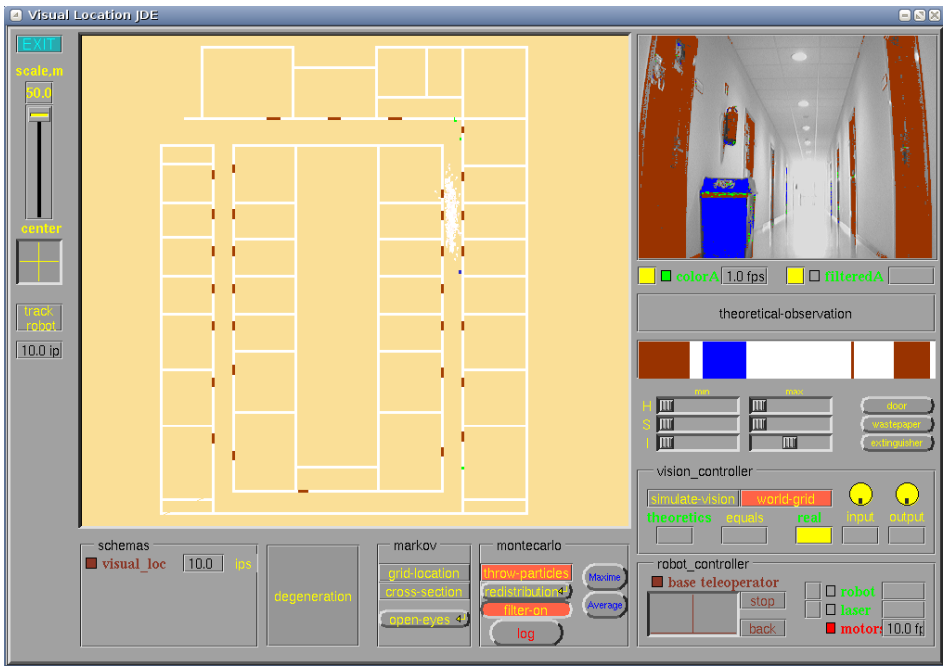


Figura 5.11: Interfaz gráfica de la aplicación. Filtro de partículas

Entre las funciones más importantes añadidas para el filtro de partículas se encuentran:

- Botón "throw part" que activa el pintado de la población de partículas.
- Botón "filter on" que pone en marcha el algoritmo del filtro de partículas.
- Botón "redistribution" que permite al usuario generar una nueva distribución aleatoria y uniforme de la población de partículas.
- Botones "average" y "maxime" para expresar la localización como la media aritmética de la posición de las partículas y también como la posición de la partícula con mayor probabilidad, dibujando un robot estimado dentro del mapa según esos botones.
- Botón luminoso "degeneration", situado en la parte inferior central de la figura 5.11, que indica que el sistema ha detectado degeneración en la población de partículas.
- Se han añadido también diales (sliders), en la parte inferior de las imágenes, que regulan los valores HSI para el control del filtro de las imágenes a través de la interfaz
- Y un botón "log" que permite la creación de un fichero con los datos de la varianza que presenta la población de partículas en cada iteración para estudiar la evolución temporal de la población.

Capítulo 6
Experimentos

En esta sección se explicarán los experimentos realizados con las dos técnicas de localización ante diferentes situaciones reales. Se realizan estos experimentos para intentar sacar el mayor rendimiento y eficacia de los algoritmos de localización utilizados, así como para comprender mejor el funcionamiento de los algoritmos y realizar los ajustes pertinentes que éstos necesiten. En la localización con mallas de probabilidad se experimenta con el tamaño de las mallas, la resolución de las celdillas y el efecto de la simetría. Para la localización con filtro de partículas se estudia, entre otras cosas, el número de partículas de la población, el ruido gaussiano aplicado al movimiento de las partículas y la simetría.

Los experimentos realizados se han realizado en un primer momento con la imagen de la figura 6.1 captada por la cámara del robot.

(a) Imagen real de la cámara

(b) Imagen real aplanada

Figura 6.1: Imagen utilizada en los experimentos

La magnitud de error de los algoritmos se mide en el orden de centímetros. El tamaño del área de localización del robot influye en la precisión del algoritmo.

6.1. Experimentos con modelo de mallas de probabilidad

En estos experimentos se realiza la ejecución típica que realiza el algoritmo de mallas de probabilidad consistente en la incorporación de evidencias en una malla tridimensional de probabilidad a través de modelos de observación visual (cámara) y de movimiento (encoders). Un ejemplo de esta ejecución se puede observar en la figura 4.14 del capítulo 4.

6.1.1. Efecto del número de orientaciones

Como se mencionó anteriormente los cubos de probabilidad, finalmente, abarcan 16 orientaciones en todo el espacio de 2π, esto es, cada tramo es de 22,5 grados. Inicialmente se ha probado con 8 y 32 orientaciones distintas:

Figura 6.2: Posibles orientaciones dentro de la malla de probabilidad

Los cubos de probabilidades son arrays de estructuras de tipo *Tgrid* y por ello son estructuras muy pesadas y sobrecargadas. La ventaja de tener mallas de probabilidad de 32 orientaciones es que nos dan mucha más información acerca de la posible localización del robot pero el inconveniente es que supone 32 posiciones para representar la malla en el algoritmo, lo cual es más costoso. Por otro lado, la ventaja de tener cubos de probabilidad de 8 orientaciones supone menos coste de cómputo, pero el inconveniente es que da muy poca información acerca de la localización del robot porque abarca muy pocas posibilidades.

Por todo esto se establece una solución de compromiso eligiendo 16 orientaciones que tiene un coste de cómputo medio (dentro de lo esperado) y nos da suficiente información para llegar a estimar una buena localización. Esta discretización es lo que

le da a la rejilla el aspecto de concha tras la incorporación de la observación (figura 6.2).

6.1.2. Efecto de la simetría

La simetría del entorno es uno de las principales características que afecta a nuestro algoritmo de localización. En mapas que presentan mucha simetría, como es nuestro caso, es difícil localizar al robot. En este experimento se estudia y se intenta disminuir el efecto negativo de la simetría. En la figura 6.3 vemos un ejemplo de simetría en la incorporación de la observación de la figura 6.1.

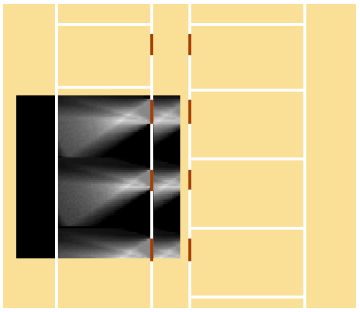


Figura 6.3: Simetría en el mapa

En la figura 6.3 observamos como al no existir ningún objeto discriminante en las habitaciones, existe una nube de probabilidad en cada habitación enfrente de cada puerta.

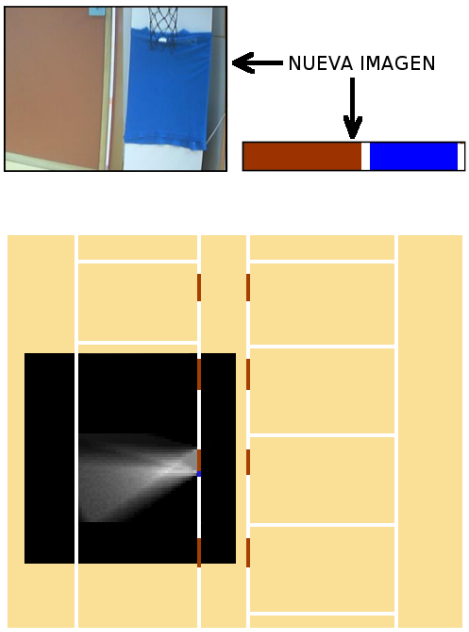


Figura 6.4: Rotura de la simetría en el mapa

Una papelera en la imagen rompe la simetría y se termina agrupando la nube de probabilidad en la habitación del medio alrededor de la puerta que tiene al lado la papelera, cosa que las otras habitaciones no tienen (figura 6.4).

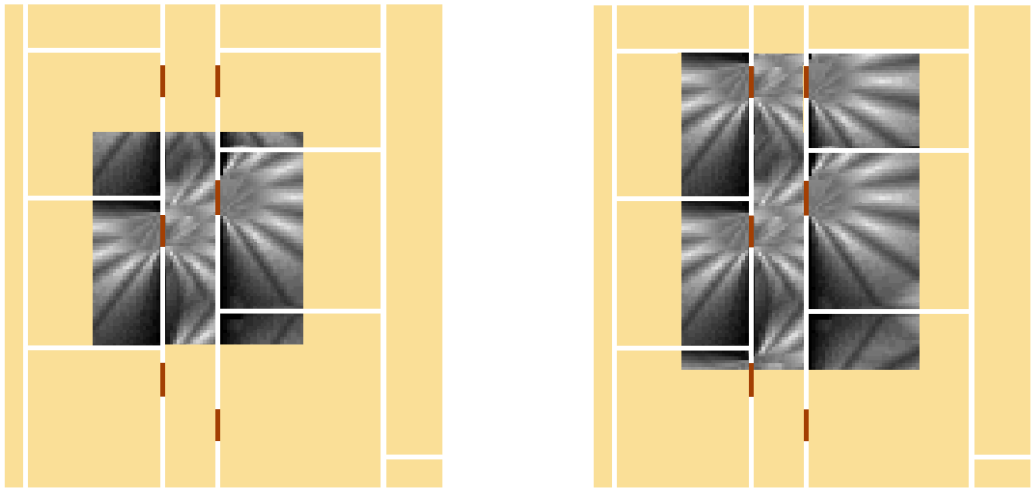
En resumen, en mapas donde no exista simetría o la simetría sea muy baja, la localización del robot se solucionará en muy pocas iteraciones. Se necesitan pocas observaciones para que la nube de probabilidad converja alrededor de alguna posición. Mientras tanto, en mapas donde exista una simetría alta como es el caso que nos ocupa, se necesitan muchas más observaciones para lograr la convergencia alrededor de una posición. Sólo con observaciones discriminantes se disminuye el efecto negativo de la simetría.

6.1.3. Efecto de la resolución de las celdillas y dimensión de la malla

La resolución de las celdillas es importante para establecer un compromiso entre el tiempo de incorporación de observaciones y la precisión con que el algoritmo realiza la localización.

La ventaja de tener una resolución de celdilla relativamente grande implica que el tiempo de incorporación es mucho menor pero existe el inconveniente de que la precisión con la que el algoritmo determina la posición del robot es bastante baja. En el caso de tener una resolución muy baja implica mucha precisión en la localización del robot pero supone un aumento del tiempo de incorporación.

La dimensión de la malla de probabilidad junto con la resolución de sus celdillas es muy importante para realizar un algoritmo mínimamente rápido dentro de lo que cabe esperar. Sabemos que el algoritmo de malla de probabilidad debe incorporar observación y actualizar probabilidades para todas las posibles posiciones y orientaciones posibles. Hemos ratificado con estos experimentos que esto es muy costoso. El tiempo de incorporación crece exponencialmente con el tamaño del área en el cual se tiene que localizar el robot y su resolución. Para los experimentos se han creado mallas de probabilidad de 8x8 y 10x9 metros y se ha puesto en marcha el algoritmo.



(a) 8m x 8m (b) 10m x 9m

Figura 6.5: Efecto de la dimensión de las mallas

En la malla de la figura 6.5(a) se tarda en incorporar una única observación alrededor de 250 segundos (más de 4 minutos). En la malla de la figura 6.5(b) ante la misma y única observación se tarda alrededor 400 segundos (aproximadamente 7 minutos). Esto quiere decir que si utilizamos una malla del tamaño del mapa, que en este caso es de aproximadamente 150 m^2, el tiempo estimado de incorporación de una única observación rondaría los 80 minutos.

Por esta razón se ha decidido implementar el algoritmo del filtro de partículas que es mucho menos costoso para resolver la localización en robots móviles.

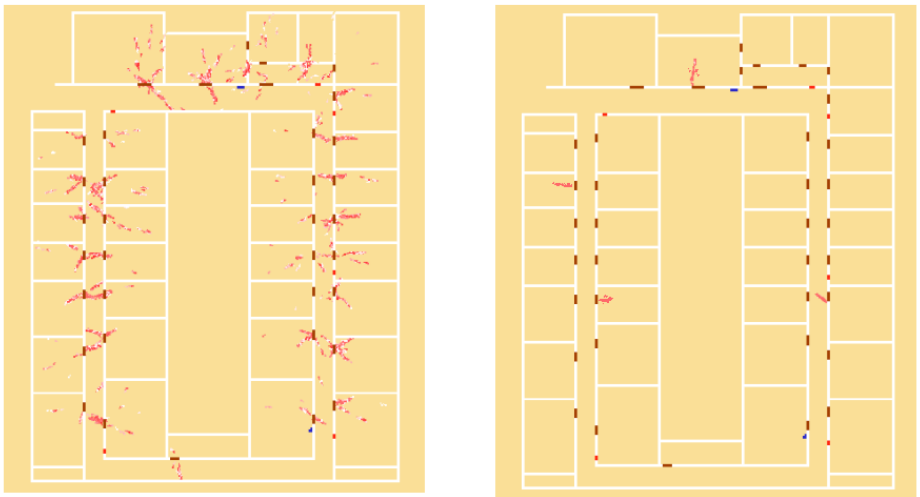
6.2. Experimentos con Filtro de Partículas

Los experimentos realizados con esta técnica han sido muy variopintos. Inicialmente se han establecido 3000 partículas para la población. Este número se ha ido variando a lo largo de los experimentos para establecer un número de partículas adecuado para el correcto funcionamiento del algoritmo. Los valores de la desviación típica que utilizamos para aplicar el ruido gaussiano en el modelo de movimiento también se han ido variando a lo largo de los experimentos. Es necesario establecer una relación entre el número de partículas y el ruido gaussiano para obtener buenos resultados.

Con este algoritmo también se han realizado experimentos en condiciones normales. Esta ejecución típica consiste en la generación continua de poblaciones de partículas cada vez más agrupadas en las posiciones compatibles con la observación. Un ejemplo de esta ejecución lo podemos observar en la figura 5.10 del capítulo 5.

6.2.1. Efecto del número de partículas

El número de partículas debe ser lo suficientemente grande como para abarcar casi todas las posibles posiciones del entorno. En los experimentos vemos como un número pequeño de partículas (por debajo de 1000) incapacita muchas posibilidades. De nuevo hay que establecer una solución de compromiso entre el coste del algoritmo y la precisión de la localización.



(a) 6000 partículas (b) 300 partículas

Figura 6.6: Efecto del número de partículas

En la figura 6.6(a) se aprecia que un número relativamente grande de partículas facilita que, tras la observación de la figura 6.1, las partículas se sitúen prácticamente en todas las posiciones que se encuentran enfrente de las puertas. Sin embargo, un número bajo de partículas como es el caso de la figura 6.6(b) solamente se contemplan posiciones de tres o cuatro zonas enfrente de las puertas. Como solución de compromiso en nuestro algoritmo, finalmente tenemos una población de 8000 partículas, que permite una buena precisión con un coste algorítmico normal.

6.2.2. Efecto de la simetría

Como ocurría con el modelo de mallas de probabilidad, la simetría es una de las características negativas que presenta nuestro entorno.

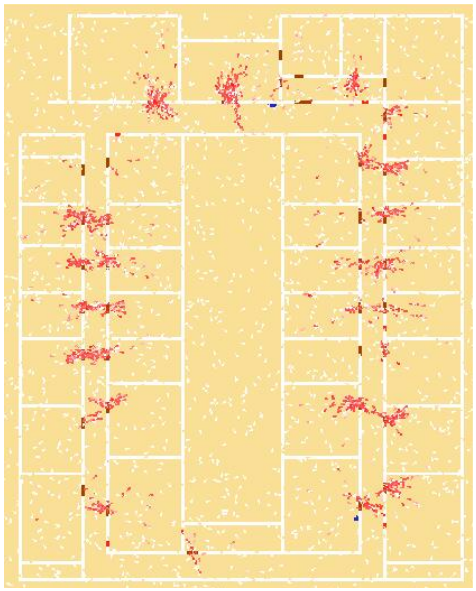


Figura 6.7: Simetría en el mapa

En la figura 6.7 podemos observar como tras la observación de la figura 6.1 las partículas se sitúan alrededor de las posiciones que tienen enfrente una puerta. Al existir muchas posiciones con una puerta de frente vemos como el robot podría estar enfrente de cualquier puerta. De nuevo hacemos uso de las balizas discriminantes para que el robot se localice. En la figura 6.8 podemos ver como tras la observación simulada de una papelera azul como la de la figura 6.4 la población converge al lugar donde está la papelera, lo que quiere decir es que la puerta que estaba viendo al principio era justo la que hay al lado de la papelera y no otra.

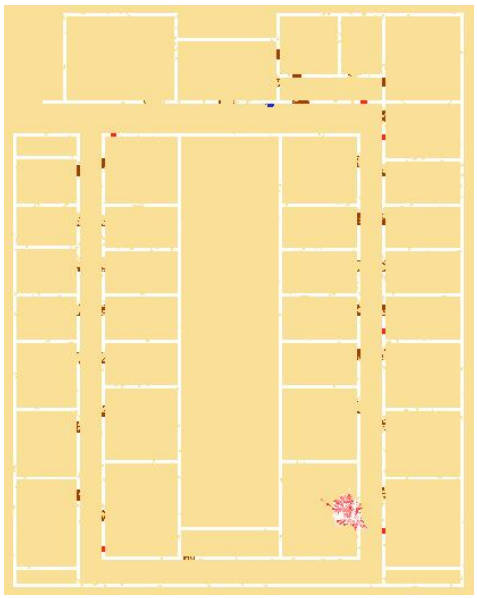


Figura 6.8: Rotura de simetría

6.2.3. Efecto del ruido gaussiano

El hecho de aplicar un ruido gaussiano en el modelo de movimiento nos servía para dar "holgura" a las posiciones de las partículas. La desviación típica aplicada en este ruido gaussiando sirve para que esa holgura sea más o menos grande.

En la figura 6.9 podemos observar que la desviación típica utilizada es importante para lograr una buena localización.

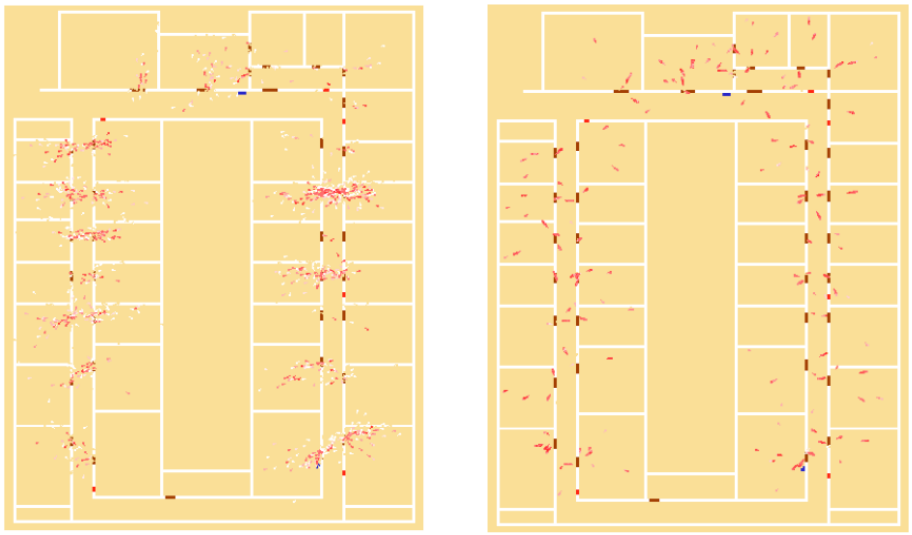


Figura 6.9: Efecto del ruido gaussiano

En la figura 6.9 (a), donde la desviación típica es bastante grande (1500 mm), vemos como las nubes partículas se concentran alrededor de las puertas pero las partículas quedan muy separadas unas de otras. Esto está bien para desambiguar cuando una nueva observación es incompatible con la posición donde se encuentran las partículas concentradas. Al estar más separadas, se necesitan muy pocas iteraciones posteriores para que las partículas ocupen de nuevo la mayor parte del entorno.

Sin embargo en la figura 6.9 (b), donde la desviación es de (100 mm), observamos como las nubes partículas quedan pequeñas y muy concentradas alrededor de las posiciones compatibles con la imagen de la figura 6.1. Esto es bueno porque la localización del robot es mucho más precisa. Pero a la hora de tener que desambiguar porque las siguientes observaciones sean incompatibles se necesitan muchas iteraciones para lograr que las partículas vuelvan a ocupar la mayor parte de las posiciones.

Si no utilizáramos ruido gaussiano todas las partículas terminarían concentrándose en una única posición sin permitir que la población albergue más posiciones posibles.

El ruido gaussiano es importante para que cada partícula pueda "observar" a su alrededor dentro de la nube de partículas, permitiendo una búsqueda local.

Por esta razón, hay que establecer un compromiso entre la posibilidad de desambiguar ante imágenes incompatibles y la precisión de la localización. Se establece por tanto un término medio. Los valores de la desviación típica establecidos finalmente para Δr son de 800mm y para $\Delta\Theta$ de 15 grados. Así conseguimos una buena capacidad de desambiguar ante imágenes incompatibles y una buena precisión en la localización del robot.

6.3. Experimentos con el modelo de observación visual.

Tras realizar los experimentos con el filtro de partículas hemos visto que el modelo de comparación de imágenes utilizado en mallas de probabilidad no era el más adecuado. Esto se debe a lo estricto que es este modelo. Recordemos que la función calculaba la distancia entre las imágenes comparándolas pixel a pixel. Es decir, la probabilidad de ser iguales era 0 ó 1. Este modelo es muy bueno para estímulos anchos dentro de la imagen. Por ejemplo, en la figura 6.10 vemos como una pequeña desviación en la imagen provoca una disminución enorme en el parecido y por tanto en la de la probabilidad de esa posición a la luz de esa imagen.

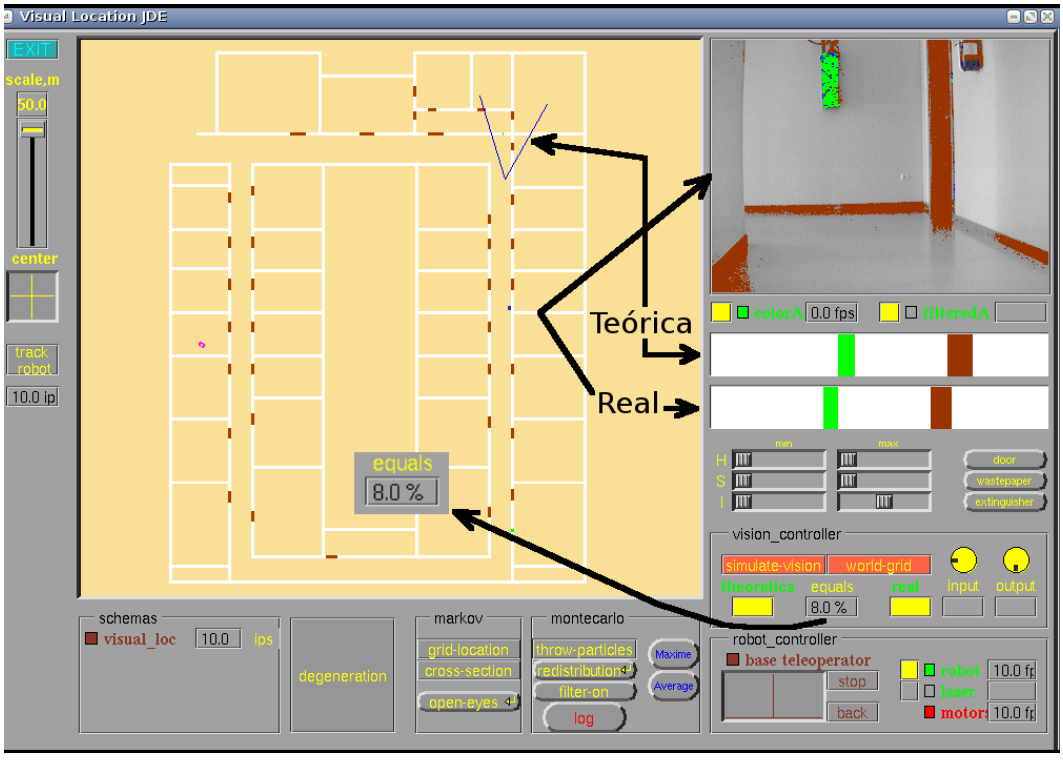


Figura 6.10: Antiguo modelo de comparación

En la figura 6.10 observamos que las imágenes son muy parecidas. La única diferencia es una pequeña desviación. Con el modelo antiguo, esto suponía una probabilidad demasiado baja. Esta situación no era muy correcta debido a que esta probabilidad haría bajar en gran proporción la acumulación y no permitía localizar al robot.

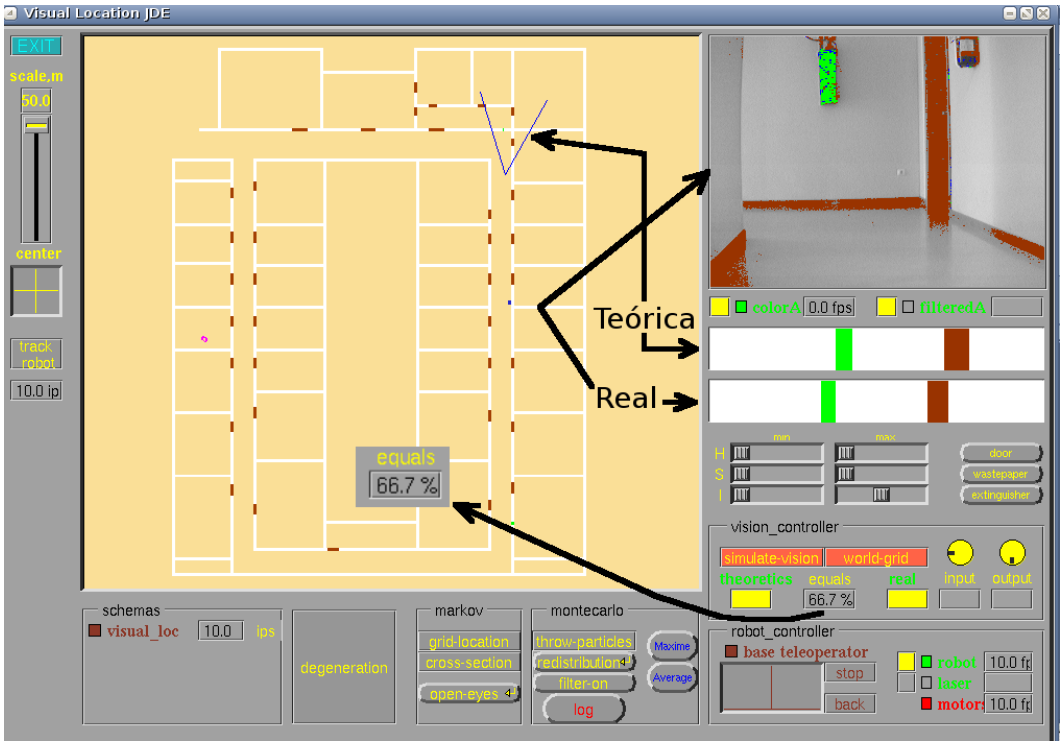


Figura 6.11: Nuevo modelo de comparación

Sin embargo en la figura 6.11 observamos que ante la misma situación la probabilidad ya es algo más razonable. Este nuevo modelo utilizado nos permite ser más flexibles en la comparación de imágenes permitiendo breves desplazamientos entre imágenes. Este modelo, como ya se explicó en el capitulo 5, consiste en la segmentación de la imagen y la utilización de "colas" que permiten contemplar los posibles desplazamientos premiando también la aparición de los extintores.

Capítulo 7
Conclusiones y trabajos futuros

Una vez descritas las dos soluciones dadas al problema de localización visual, en este capítulo se resumen las conclusiones sacadas tras el diseño, implementación y experimentos realizados, indicando los aportes genuinos de este proyecto y las posibles líneas futuras.

En los tres primeros capítulos se da una reseña histórica acerca de la robótica en general, se presenta el entorno de desarrollo con el cual se trabaja y por otro lado el contexto en el que está encuadrado el proyecto y los objetivos marcados para el mismo. En los siguientes se explican los métodos utilizados en este proyecto. En un primer lugar se estudia el método de mallas de probabilidad y en segundo lugar el filtro de partículas. A continuación se describen los pertinentes experimentos realizados con ambos métodos.

7.1. Conclusiones

Repasando los objetivos marcados en el capítulo 2 y teniendo en cuenta el principal objetivo marcado para este proyecto vamos a repasar punto por punto la medida en que se han satisfecho dichos objetivos. El objetivo principal se ha resuelto satisfactoriamente con los algoritmos utilizados y el robot real se localiza correctamente en la planta de la universidad empleando información sensorial de visión y odometría, dando de manera continua una estimación de posición más precisa que la odometría y sin acumulación de errores.

El objetivo principal estaba articulado en tres etapas principales: el diseño e implementación de un algoritmo de localización basado en las técnicas de *mallas de probabilidad* mediante el esquema perceptivo *visualloc-mallas.c*, un algoritmo de localización basado en *filtros de partículas* implementado en el esquema *visualloc-particulas.c* y los experimentos realizados con ellos a través de la interfaz proporcionada por nuestro esquema de servicio implementado en *guivisualloc.c*.

El primer subobjetivo, la técnica probabilística basada en mallas de probabilidad, se ha descrito en el capítulo 4. Después del estudio de la teoría del algoritmo se ha planteado un diseño organizado en esquemas para su implementación. Esta implementación utiliza dos modelos de observación que nos proporcionan información sensorial del robot: visual y odométrico. Para el modelo visual se ha programado un filtro de imágenes para sacar información relevante de las imágenes captadas. Después se realiza un resumen de la imagen real y el cálculo de imágenes teóricas para cada posición posible dentro de la malla. Estas imágenes teóricas se comparan pixel a pixel con la imagen real. Para el modelo odométrico se realiza el cálculo del desplazamiento ejercido por el robot para introducirlo dentro de la malla desplazando de manera análoga el contenido de la misma. Tras la incorporación de las observaciones en cada iteración se realiza una acumulación a partir de una formulación de la regla de Bayes donde las posiciones compatibles con las observaciones tendrán la probabilidad más alta y las no compatibles las más bajas.

El segundo subobjetivo, el filtro de partículas, se ha descrito en el capítulo 5. Este algoritmo está basado en el muestreo de la probabilidad de ocupación con una población de partículas que tras varias iteraciones se concentran alrededor de las posiciones compatibles con las observaciones. Después del estudio de la teoría del algoritmo se ha diseñado mediante esquemas una implementación. Esta implementación utiliza también dos modelos de observación y un paso añadido de remuestreo basado en el algoritmo de la ruleta para la generación de nuevas poblaciones de partículas.

El modelo visual funciona de igual manera con la única excepción de la función de comparación de las imágenes que ha sido cambiada porque la utilizada en el algoritmo anterior no funcionaba bien tras comprobarlo mediante los experimentos. Esta nueva función busca solucionar uno de los requisitos de este proyecto: la robustez, pudiendo digerir mejor las condiciones de iluminación y pequeños desplazamientos laterales en la comparación de imágenes. Esta función consiste en la segmentación de las imágenes. El modelo odométrico se aplica directamente a las partículas añadiendo un ruido gaussiano. El paso de remuestreo consiste en la generación de nuevas poblaciones a través del algoritmo de la ruleta en el que las partículas con mayor probabilidad generan hijos para la población siguiente. Es el ruido gaussiano aplicado en el modelo de movimiento el que evita tener hijos exactamente iguales en poblaciones siguientes.

En los experimentos descritos en el capítulo 6 se puede observar que los objetivos propuestos se han satisfecho. De los experimentos realizados con el primer algoritmo de localización sacamos la conclusión de la dependencia que tiene el coste del algoritmo

con el tamaño y la resolución de la malla escogida. Esta técnica permite representar bien las situaciones ambiguas y con ella el robot real se localiza. Los inconvenientes que presenta la utilización de este algoritmo es la "no escabilidad" a grandes entornos debido al alto coste computacional que necesita la incorporación de observaciones. Esto impide satisfacer uno de los requisitos que presentaba este proyecto: el funcionamiento en tiempo real para que la localización sea aplicable al robot real.

Sin embargo, esto no ocurre con el segundo algoritmo que es capaz de incorporar observaciones con un coste computacional muy bajo y con mayor rapidez, consiguiendo así cumplir el requisito de ejecución en tiempo real para la localización comentado en el capítulo 2, trabajando con 8000 partículas y un orden de magnitud de error expresado en centímetros.

En ambos casos, existe la característica de la simetría que en muchos casos dificulta la localización del robot. En los experimentos se han afinado los algoritmos para que funcionen en el robot real y se ha suavizado la repercusión negativa que ofrece la simetría intentando introducir mayor robustez a los algoritmos.

La localización a través del filtro de partículas es más precisa que en el modelo de mallas de probabilidad debido a la escalabilidad que presenta y a que el orden de magnitud de error en ambos métodos es el mismo.

7.2. Trabajos futuros

Una de las líneas futuras más inmediata y parte de la motivación que presenta este proyecto en relación a otros comportamientos es la posibilidad de portar los algoritmos de navegación global [Isado, 2005],[López, 2005] al robot real. Estos algoritmos de navegación se han realizado a través de simuladores ya que era necesario una estimación continua de la posición del robot que no estaba resuelta para el robot real. Los posibles trabajos futuros serán portar los algoritmos de navegación al robot real ya que ahora es posible gracias a los algoritmos de localización descritos e implementados en este proyecto.

Otra posible línea futura de continuación de este proyecto es la resolución de la localización cuando existen obstáculos dinámicos (obstáculos que no están en el mapa). Este proyecto se ha resuelto sin tener en cuenta la posibilidad de que haya gente vagando por el entorno. Lo más inmediato sería probar los algoritmos con obstáculos dinámicos. Quizá probando a suavizar los modelos de observación para que el ruido que se añade a las lecturas sensoriales fuera el más suave posible y no altere en demasía la estimación.

Bibliografía

[ActivMedia, 2002] ActivMedia. Aria reference manual. *Technical Report version 1.1.10, ActiveMedia Robotics*, 2002.

[Arulampalam *et al.*, 2002] Sanjeev Arulampalam, Simon Maskell, Neil Gordon, y Tim Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002.

[Baena, 2003] Alfonso Matute Baena. Filtro de color configurable. *Proyecto Fin de Carrera, URJC*, 2003.

[Barrera *et al.*, 2005] Pablo Barrera, José María Cañas, y Vicente Matellán. Visual object tracking in 3d with color based particle filter. *Int. Journal of Information Technology*, 2005.

[Benítez, 2004] Raúl Benítez. Sistema de localización basado en la detección de segmentos para robot móviles. *Proyecto fin de carrera, URJC*, 2004.

[Brian P. Gerkey, 2003] Andrew Howard Brian P. Gerkey, Richard T. Vaughan. The player/stage project: Tools for multi-robot and distributed sensor systems. *Proceedings of the international conference on Advanced Robotics.*, pages 317–323, 2003.

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

[Crespo, 2003] María Angeles Crespo. Localización probabilística en un robot con visión local. *Proyecto fin de carrera, Universidad Politécnica de Madrid*, 2003.

[Fox *et al.*, 1999] Dieter Fox, Wolfram Burgard, Frank Dellaert, y Sebastian Thrun. Monte Carlo localization: efficient position estimation for mobile robots. In *Proceedings of the 16th AAAI National Conference on Artificial Intelligence*, pages 343–349, Orlando (Florida, USA), July 1999.

[Fox *et al.*, 2000] Dieter Fox, Wolgram Burgard, Frank Dellaert, y Sebastian Thrun. Particle filters for mobile robot localization. In *In A. Doucet, N. de Freitas and N. Gordon, editors, Sequential Monte Carlo Methods in Practice.*, Springer Verlag, New York, 2000.

[Garcia, 2002] Esther Garcia. Construcción de un teleoperador para el robot eyebot. *Proyecto fin de carrera, URJC*, 2002.

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

[Isado, 2005] José Raul Isado. Navegación global por el método del gradiente. *Proyecto Fin de Carrera, URJC*, 2005.

[Isard y Blake, 1998] M. Isard y A. Blake. Condensation-conditional density propagation for visual tracking. *Int. J. Computer Vision, in press*, 1998.

[Kachach, 2005] Redouane Kachach. Localización probabilística con laser en el robot pionner. *Proyecto fin de carrera, URJC*, 2005.

[Lobato, 2003] David Lobato. Evitación de obstáculos basada en ventana dinámica. *Proyecto fin de carrera, URJC*, 2003.

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

[Mackay, 1999] D.J.C. Mackay. Introduction to Monte Carlo methods. In M. Jordan, editor, *Learning in Graphical Models*, pages 175–204. MIT Press, Cambridge (MA, USA), 1999.

[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.

[Siddiqi *et al.*, 2003] Sajid M. Siddiqi, Gaurav S. Sukhatme, y Andrew Howard. Experiments in Monte-Carlo localization using WiFi signal strength. In *Proceedings of the 11th International Conference on Advanced Robotics ICAR'2003*, pages 471–476, Coimbra (Portugal), June 2003.

[Sáez y Escolano, 2002] J.M. Sáez y F. Escolano. Localización global en mapas 3D basada en filtros de partículas. In *Actas del 2do Workshop de Agentes Físicos, WAF'2002*, pages 29–40, Universidad de Murcia, March 2002.

[Thrun, 1997] S. Thrun. Bayesian landmark learning for mobile robot localization. In *To appear in Machine Learning*, April 1997.

[Thrun, 2000] S. Thrun. Probabilistic algorithms in robotics. *AI Magazine*, 2000.