UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

MÁSTER UNIVERSITARIO EN INICIACIÓN A LA INVESTIGACIÓN EN TECNOLOGÍA (MUIT)

ESPECIALIDAD EN:
TECNOLOGÍAS INFORMÁTICAS
Y DE LAS COMUNICACIONES (TINC)

Trabajo Fin de Máster MUIT-TINC

Grid-localization for autonomous robot working in indoor environment.

Luisa F. Sánchez Peralta

Octubre 2010

UNIVERSIDAD DE EXTREMADURA
Escuela Politécnica

MÁSTER UNIVERSITARIO EN INICIACIÓN A LA INVESTIGACIÓN EN TECNOLOGÍA (MUIT)

Trabajo Fin de Máster MUIT-TINC

Grid-localization for autonomous robot working in indoor environment.

Autor: Luisa F. Sánchez Peralta
Fdo:

Director: Pedro Núñez Trujillo
Fdo:

Tribunal Calificador

Presidente: Pablo García Rodríguez
Fdo:

Secretario: José Vicente Crespo
Fdo.:

Vocal: José Manuel Taboada Varela
Fdo.:

CALIFICACIÓN:
FECHA:

Índice general

Índice de figuras

Índice de cuadros

Resumen

Introducción

La localización es una problemática a resolver dentro del campo de la robótica autónoma móvil y consiste en identificar la posición y orientación del robot con respecto al entorno en el que se mueve. Para afrontar este problema existen distintas metodologías.

Objetivos

El objetivo general de este Trabajo Fin de Máster es el estudio y desarrollo de un sistema de localización probabilístico basado en celdas de ocupación y el uso de un sensor láser que permita la determinación de la posición del robot en un entorno cerrado cuyo mapa es conocido.

Metodología

La metodología empleada en el presente trabajo de investigación se ajusta al método científico clásico. Se ha diseñado e implementado un algoritmo de localización haciendo uso de medidas de similitud (coeficientes de correlación) entre un mapa global del entorno y un mapa local creado a partir de las medidas del entorno realizadas con el sensor láser.

Resultados

Se obtiene un error en la localización de $287{,}58 \pm 169{,}73$ milímetros en el eje X, $135{,}93 \pm 82{,}56$ milímentros en el eje Z y $1{,}76 \pm 1{,}26$ en el ángulo de rotación.

Conclusiones

Las pruebas realizadas demuestran que el robot puede localizarse correctamente dentro de un entorno cerrado, definido mediante mapas de celdas de ocupación y basándose en medidas de similitud. Gracias al diseño basado en componentes de la implementación, que ha sido incorporada a RoboComp, es fácil de ampliar o acoplar con otros componentes.

Palabras clave

robótica autónoma, localización, láser

Abstract

Introduction

Localization is a problem to be solved in the field of mobile robotics and it aims to identify the position and orientation of the robot in the wnvironment it is moving in. There are several approaches to address this problem.

Objetives

The general objective of this Masther Thesis is the study and development of a probabilistic localization system based on grid-maps and using a laser sensor which allows identifying the robot pose in a indoor environment with a known map.

Methodology

Followed methodolgy adjusts to the classic scientific method. A localization algorithm has been designed and implemented. It is based on the use of similarity measures (correlation coefficients) between a global map of the environment and a local map created with the measures of the laser sensor.

Resuls

Localization error is equal to $287{,}58 \pm 169{,}73$ mm in X-axis; $135{,}93 \pm 82{,}56$ mm in Z-axis and $1{,}76 \pm 1{,}26$ in rotation angle.

Conclusions

Tests show that the robot can localizate itself inside an indoor environment, defined by ocuppation grid maps and based on similarity measures. It has been implemented using component-based design and has been included in the RoboComp framework. Thus, it is easy to extend and connect with other components.

Las pruebas realizadas demuestran que el robot puede localizarse correctamente dentro de un entorno cerrado, definido mediante mapas de celdas de ocupación y basándose en medidas de similitud. Gracias al diseño basado en componentes de la

implementación, que ha sido incorporada a RoboComp, es fácil de ampliar o acoplar con otros componentes.

Palabras clave

autonomous robotics, localization, laser, correlation coefficients

Capítulo 1

Introducción

Un *robot* es una máquina controlada por ordenador y programada para moverse, manipular objetos y realizar trabajos a la vez que interacciona con su entorno. Para considerar a un robot *autónomo*, el sistema de navegación ha de residir en la misma máquina y ha de ser capaz de operar sin conexiones físicas a equipos externos. Finalmente, la *movilidad* de un robot es el grado con el que el robot es capaz de moverse libremente en el mundo. Estos tres conceptos dan lugar a los **robots autónomos móviles**, campo de aplicación de este Trabajo Fin de Máster.

La **localización** es una problemática a resolver dentro del campo de la robótica móvil y consiste en identificar la posición y orientación del robot con respecto al entorno en el que se mueve. Para afrontar este problema existen distintas metodologías. Este Trabajo Fin de Máster abordará la localización desde la perspectiva de robótica probabilística para poder identificar, a partir de los datos de un sensor láser, la posición y orientación del robot.

La presente memoria se estructura en 7 capítulos. Los dos primeros capítulos, de caracter más general, describen los objetivos de trabajo y la metodología científica seguida para su consecución. El capítulo 3 presenta el entorno de trabajo para el cual se va a desarrollar un componente que permita la localización. Tras esta introducción, en el siguiente capítulo se describen los conceptos en los que se ha basado el desarrollo del componente *localizationComp* para pasar en el capítulo 4 a comentar el diseño

del componente. Finalmente, la memoria concluye con un capítulo de resultados y otro de conclusiones.

1.1. Objetivos

El **objetivo general** de este Trabajo Fin de Máster es el estudio y desarrollo de un sistema de localizacion probabilístico basado en celdas de ocupación y el uso de un sensor láser que permita la determinación de la posición del robot en un entorno cerrado cuyo mapa es conocido.

Este objetivo general puede particularizarse en los siguientes *objetivos específicos*:

- Familiarización con la robótica probabilística
- Desarrollo de un algoritmo que permita calcular la posición actual del robot en base a los datos del láser y de un mapa del entorno conocido a priori
- Implementación de un componente de RoboComp que obtenga la posición del robot mediante el algoritmo planteado en el punto anterior
- Implementación de una interfaz de usuario que permita visualizar los resultados del componente desarrollado

Capítulo 2

Metodología

La metodología empleada en el presente trabajo de investigación se ajusta al método científico clásico [2], [3], de tal modo que se puedan obtener resultados válidos de una forma fiable.

El método científico puede adapatarse a cada una de las necesidades y ámbitos de actuación que puedan surgir, por lo que existen multitud de variantes del mismo. No obstante, puede decirse que en el método científico (MC-14) están presentes los 14 puntos siguientes, agrupados en 4 fases:

Fase I. Observación, generalización e hipótesis

1. *Observación curiosa.* Este paso es el comienzo del proceso inductivo, por el que se descubren nuevos problemas, ideas, teorías... a partir del uso de los sentidos así como diferentes instrumentos y herramientas. Se debe estar alerta porque los problemas e idas no tienen un origen concreto y pueden surgir en diversas circunstancias: una necesidad no cubierta, la resolución de otro problema, la experiencia previa...

   Tras realizar el trabajo de la asignatura *Introducción a la Investigación en Teoría de la Señal y las Comunicaciones* dentro del área de visión artificial y robótica autónoma móvil, se decidió continuar la línea con el Trabajo Fin de Máster.

   Al participar en las tareas de investigación de un grupo consolidado como es el Laboratorio de Robótica y Visión Artificial (RoboLab), se contó con

una experiencia previa y un amplio bagaje de gran valor durante esta primera fase del método científico.

2. *Detección del problema.* Una vez que se ha detectado un problema, idea, necesidad...es conveniente formularlo como pregunta de tal forma que la solución persiga su respuesta.

   La pregunta inicial que se definió fue: ¿Cómo se puede conseguir una lozalización autónoma para robots móviles en entornos cerrados?. Al ser una pregunta bastante general permite la obtención de diferentes soluciones. En base a la experiencia, se particularizó la pregunta inicial al campo de las celdas de ocupación, por lo que finalmente la pregunta a resolver es ¿cómo se puede localizar un robot móvil basándose en celdas de ocupación y los datos ofrecidos por un sensor láser?

3. *Objetivos y planificación.* Tras definir el problema, se extraen los objetivos que deben cumplirse con la solución obtenida mediante el método científico. Estos objetivos deben ser realistas y dentro de una planficación flexible.

   Los objetivos del presente trabajo se exponen en el apartado 1.1. La planificación temporal ha sido flexible de tal forma que ha permitido compaginar la actividad laboral con la realización del Trabajo Fin de Máster.

4. *Búsqueda, exploración y recopilación de pruebas.* Este paso es el centro de la resolución de problemas y contribuye en gran manera a la importancia del método científico. Se debe hacer una búsqueda exhaustiva de información y reunir todas las pruebas y documentación que sirvan para poder resolver el problema, sin dejar de lado la creatividad e innovación. En lo que respecta a este Trabajo Fin de Máster, se realizó una lectura de literatura relacionada con el tema particular así como una revisión de conceptos teóricos previamente adquiridos.

5. *Generación de soluciones creativas y alternativas lógicas.* Tras la fase de observación, se plantean posibles soluciones, bien mediante el método de prueba y error o bien siguiendo un proceso sistemático, gradual y análiti-

co de razonamiento lógico.

Estas soluciones surgen apoyadas por la experiencia del grupo RoboLab, de donde se han obtenido opiniones que han permitido combinar diferentes soluciones e ideas previas para obtener diferentes soluciones que respondan a la pregunta planteada en el punto 2.

6. *Evaluación de las evidencias.* En este punto se tiene ya un conjunto de posibles soluciones que deben analizarse para elegir la hipótesis de trabajo.

   En el presente Trabajo Fin de Máster se optó por la hipótesis tras una valoración en términos de adecuación técnica, adecuación a las necesidades del grupo de trabajo, disponibilidad y tiempo para llevarlo a cabo.

7. *Formulación de la hipótesis.* La hipótesis es la solución propuesta al problema definido, tras descartar otras posibles soluciones en el apartado anterior.

   La hipótesis de este Trabajo es la posibilidad de basar en medidas de simulitud la localización autónoma de robots móviles en entornos cerrados. La solución adoptada en este caso es la que se describe en el capítulo 5.

Fase II. Pruebas, análisis y conclusiones

8. *Experimentación y prueba de la hipótesis.* En este punto influye el tipo de problema y su importancia en cuanto al tipo de prueba y experimentación al que debe someterse la hipótesis.

   Las pruebas realizadas en este trabajo arrojan los resultados reseñados en el capítulo 6.

9. *Extracción de conclusiones.* Para extraer conclusiones es recomendable empezar revisando las anotaciones del punto 6 y las resultados obtenidos tras las pruebas de la hipótesis. Si esta resultara parcialmente falsa, se debe modificar algún punto de la misma y volver a evaluarla, mientras que si es completamente falsa es conveniente retomar alguna otra solución descartada en el punto 5.

Tras realizar las pruebas pertinentes, las conclusiones extraídas se recogen en el capítulo 7.

10. *Emisión del juicio de valor.* Gracias a una mente abierta y actitud escéptica a lo largo del proyecto, el investigador debe mantener su hipótesis hasta que esta se compruebe como falsa, sin dejar de lado la posibilidad de aceptar nuevas propuestas y opiniones que puedan reajustar la hipótesis con el fin de mejorarla.

Fase III. Teoría científica

11. *Desarrollo de la teoría.* Una vez que la hipótesis se ha comprobado que es válida, suele ser necesario un informe o memoria que recoja el trabajo realizado. Para hacer público el trabajo realizado puede optarse por enviar artículos a revistas, comunicaciones a congresos, publicaciones de libros, patentes... Se espera escribir una comunicación que recoja los resultados obtenidos a algún congreso nacional como por ejemplo al Workshop de Agentes Físicos (WAF2011).

Fase IV. Ingredientes extra

12. *Métodos creativos, no lógicos, lógicos y técnicos.* Deben conocerse los métodos de trabajo adecuados para obtener resultados mejores que los obtenidos de la improvisación.

13. *Objetivos del método.* El método científico tiene como objetivo principal refinar, extender y aplicar el conocimiento y buscar la verdad aunque a veces no pueda determinarse con total precisión. Hay que tener en cuenta que el principio metodológico principal es llevar a cabo los 11 pasos expuestos anteriormente.

14. *Aptitudes y habilidades personales.* En cuanto a las aptitudes personales que debe presentar un investigador, cabe señalar, entre muchas otras, una curiosidad innata, honestidad, flexibilidad, capacidad de comunicación y organización, creatividad o una mente abierta.

Capítulo 3

Robocomp

El entorno en el que se desarrolla el proyecto queda caracterizado por el robot sobre el que será integrado (familia RobEx) y por el sistema de componentes (RoboComp). El robot en el que se llevarán a cabo las pruebas es un robot de la serie RobEx. Estos son robots móviles con conectividad wireless y/o Ethernet, disponen de un ordenador de a bordo en el que se ejecutan algunos componentes y desde el que pueden interactuar con otros componentes ejecutados de forma remota. En general las tareas del ordenador de a bordo se centran en el control del hardware del robot: movimiento de la base y captura de las señales de los dispositivos incorporados, láser, cámaras, micrófonos, etc.

El sistema de componentes llamado RoboComp puede definirse como un grafo de componentes o procesos que pueden ser distribuidos en varios procesadores proporcionando un sistema de comunicación que permite el intercambio de información entre sí mediante una síntesis muy simple.

Tanto RobEx como RoboComp son proyectos desarrollados en el Laboratorio de Robótica y Vision Artificial de la Universidad de Extremadura. Son libres y se puede acceder a ellos mediante sus correspondientes páginas web [4],[5]. Dado que los resultados del trabajo se han incluido dentro de RoboComp, el software desarrollado en él se distribuye bajo la misma licencia.

En este capítulo se describen los elementos con los que se ha trabajado. En primer lugar, se explica el concepto de la programación orientada a componentes. Posteriormente se hace una revisión de RoboComp, sistema para el cual se ha implementado el componente de localización. En tercer lugar, se describe el proyecto Player, simulador de robots sobre el que se ha desarrollado el componente. Para acabar el capítulo, se exponen brevemente otras herramientas software que también se han empleado.

3.1. Programación orientada a componentes

A la hora de desarrollar software hay que tener muy presentes los conceptos de escalabilidad y reusabilidad, especialmente cuando el software va a emplearse en aplicaciones robóticas. La **programación orientada a componentes** permite:

- Construir sistemas rápida y eficazmente, incorporando componentes ya existentes
- Construir sistemas fiables, incorporando componentes probados en múltiples proyectos software
- Obtener sistemas modulares
- Construir sistemas flexibles cuyos componentes puedan ser reemplazados

Un *componente* es una unidad de abstracción estática con interfaz de acceso a su funcionalidad. Los componentes son unidades ejecutables con dependencias contextuales explícitas y que pueden ser utilizados independientemente de su implementación. Para llevar a cabo una tarea concreta, pueden usarse diferentes componentes de forma conjunta. Cada componente se implementa con programación orientada a objetos. La estructura genérica de un componente se muestra en la figura 3.1.

Con la programación orientada a componentes obtenemos *modularidad*, ya que el diseño del sistema se reduce a la composición de unidades funcionales; *flexibilidad*,

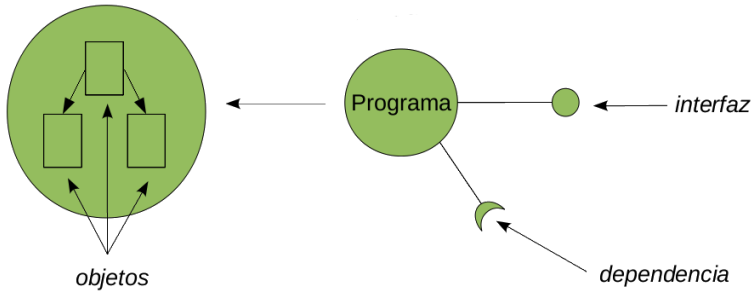


Figura 3.1: Estructura de un componente [1]

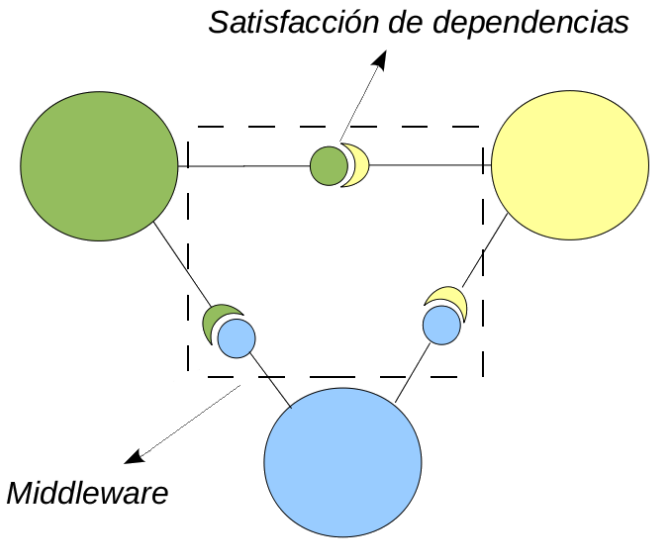


Figura 3.2: Sistema compuesto por tres componentes [1]

puesto un componente puede sustituirse por otro que tenga la misma funcionalidad; y *ejecución distribuida*, eliminando las limitaciones asociadas a la capacidad de procesamiento.

Para la comunicación entre componentes es necesario un middleware que la lleve a cabo. La figura 3.2 muestra un sistema formado por tres componentes. El componente amarillo es dependiente de los componentes verde y azul, mientras que el verde sólo lo es del azul y éste último no presenta ninguna dependencia.

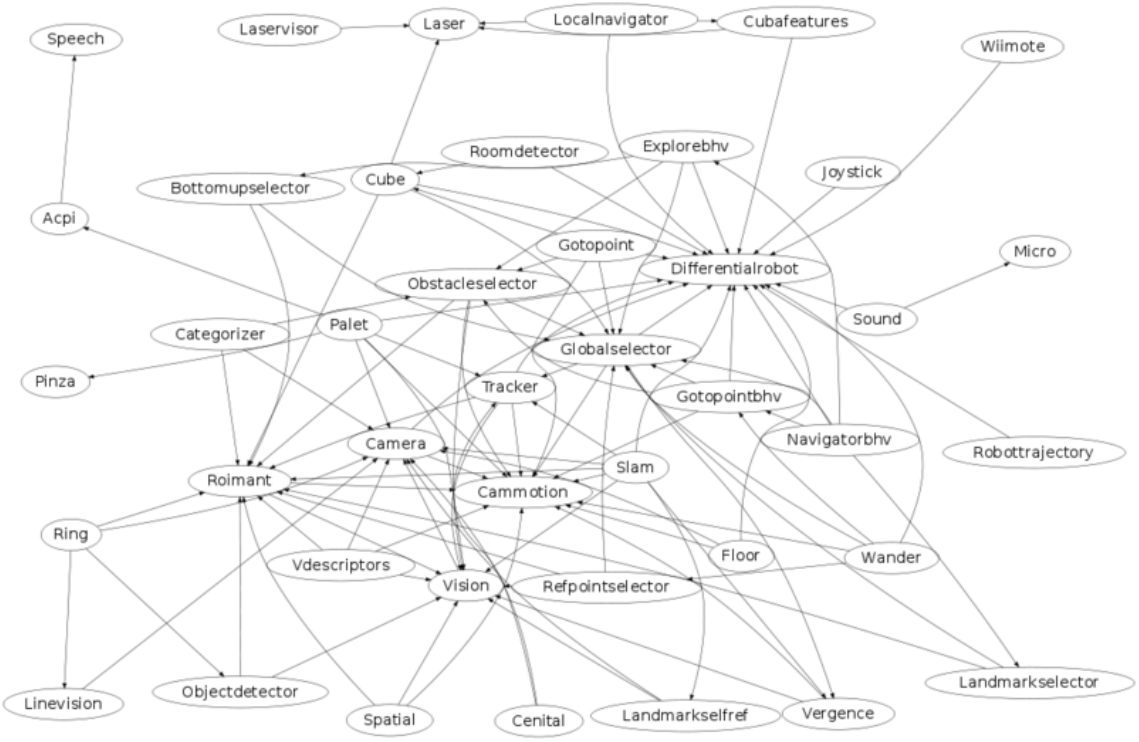


Figura 3.3: Grafo de componentes de RoboComp

3.2. RoboComp

RoboComp es un repositorio de componentes basados en Ice con aplicaciones en robótica y visión artificial. RoboComp se comenzó a desarrollar en Robolab en 2005. Actualmente el proyecto ha sido migrado a SourceForge, donde, además de encontrarse la página del proyecto [5], tiene un wiki con documentación y un repositorio. Dispone de componentes para captura y visualización de vídeo, control del robot RobEx, detección y mantenimiento de regiones de interés (ROI), lectura y visualización de láser, lectura de joystick o navegación, entre otros muchos. La figura 3.3 muestra el grafo de componentes de RoboComp: los componentes y las relaciones entre ellos.

Además de los componentes, RoboComp tiene una serie de herramientas, como son un generador automático de componentes nuevos y un programa gráfico de manipulación y control de componentes (managerComp).

3.3. Proyecto Player

El **proyecto Player** [6] auna esfuerzos para el desarrollo de herramientas que faciliten la investigación con robots y sensores. Se compone de tres elementos:

- **Player.** Proporciona una interfaz para diferentes robots y sensores. El modelo cliente/servidor de Player permite que el programa de control del robot pueda escribirse en cualquier lenguaje y ejecutarse en cualquier ordenador con conexión de red con el robot. Player soporta varias conexiones cliente simultáneas, creando nuevas posibilidades para control y detección colaborativo y distribuido. Además, Player permite trabajar con una amplia variedad de robots y accesorios.

- **Stage.** Simula una población de robots en un entorno bidimensional definido en una imagen. Existen distintos tipos de sensores: láser y odometría, entre otros. Normalmente, Stage presenta una interfaz Player por lo que es fácil pasar de la simulación al hardware. Muchos controladores diseñados en Stage se han comprobado que funcionan correctamente en robots reales.

- **Gazebo.** Es un simulador 3D de robots para exteriores. Al igual que Stage, puede simular una población de robots, sensores y objetos, pero lo hace en un mundo tridimensional. Genera una retroalimentación de los sensores realista asi como interacciones físicas entre objetos, ya que incluye simulación de cuerpos rígidos. Además de su propia interfaz, Gazebo también dispone de una interfaz Player. Controladores escritos para el simulador Stage pueden emplearse en Gazebo y viceversa.

En el presente Trabajo Fin de Máster se ha empleado el simulador Stage. Como ya se ha comentado, Stage es un simulador para aplicaciones robóticas que proporciona un mundo virtual para robots móviles, sensores y varios objetos que los robots pueden detectar y manipular. Este programa normalmente se usa junto con el servidor Player para servir como mundo simulado a los programas de control. Stage proporciona modelos simples, con poca carga computacional de diferentes disposi-

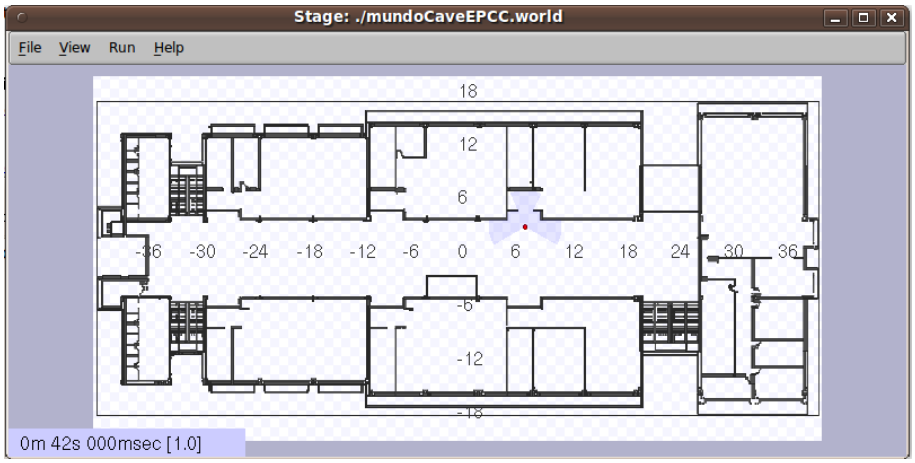


Figura 3.4: Pantalla de Player/Stage

tivos, en lugar de intentar recrear cada uno de ellos con gran fidelidad, ya que se he comprobado que esta aproximación es útil en este ámbito.

La figura 3.4 muestra una pantalla del simulador Stage. Mediante ficheros de configuración, se le indica a Stage la imagen que debe utilizar como mapa del entorno, la escala y las dimesiones reales del entorno simulado. Así mismo, se indica la posición y orientación iniciales del robot dentro del mundo.

3.4. Otras herramientas software

IPP

IPP es una biblioteca empleada para procesar señales de una y dos dimensiones. IPP es software privativo, y aunque es gratuito para el uso personal bajo GNU/Linux, su código fuente no es público. Ofrece una gran variedad de funciones para trabajar con imágenes que se han usado en múltiples componentes relacionados con el tratamiento de imágenes.

Qt4

Qt4 es un framework de desarrollo cuyo objeto principal es la creación de interfaces gráficas. A pesar de que sea este su principal uso, engloba una gran cantidad de

funcionalidades distintas: interfaz multiplataforma con el sistema operativo (sistema de ficheros, procesos e hilos, entre otros), comunicación por red mediante sockets, interfaz con OpenGL, conexiones SQL, renderizado de HTML, y módulos de reproducción y streaming multimedia. Atendiendo al lenguaje de programación, Qt está escrita en C++, pero existen multitud de bindings que hacen posible su uso desde otros lenguajes como Java o Python.

Qt4 designer

Qt4 designer es una herramienta de la empresa Trolltecha para diseñar y construir interfaces gráficos de usuario desde los componentes Qt. Permite diseñar y construir widgets y dialogs de una forma sencilla y eficaz. Una característica esencial de este diseñador es que permite aprovechar las señales y slots de Qt lo que facilita la tarea de conexión del interfaz con el código interno de la aplicación

Ice

Ice es un middleware que permite crear componentes software que pueden funcionar de forma distribuida sobre plataformas heterogéneas, de forma que puedan comunicarse entre sí y llevar a cabo un trabajo conjunto. Es el principal producto de la empresa estadounidense ZeroC, y se distribuye bajo una doble licencia GPL+ privativa para habilitar que las empresas desarrollen software privativo con Ice, a cambio del pago de una licencia. Las principales características de Ice frente a otras tecnologías similares es su rapidez, baja latencia y escalabilidad.

CMake

Cmake es una aplicación que permite generar automáticamente ficheros Makefile y ficheros de proyecto de varios IDE como Kdevelop. Cmake permite delegar la creación de ficheros Makefile, consiguiendo un resultado multiplataforma y robusto, sin llegar a perder el control del proceso de compilación. Cmake es una iniciativa libre que nació como respuesta a la ausencia de una alternativa suficientemente buena durante el desarrollo de una librería llamada ITK.

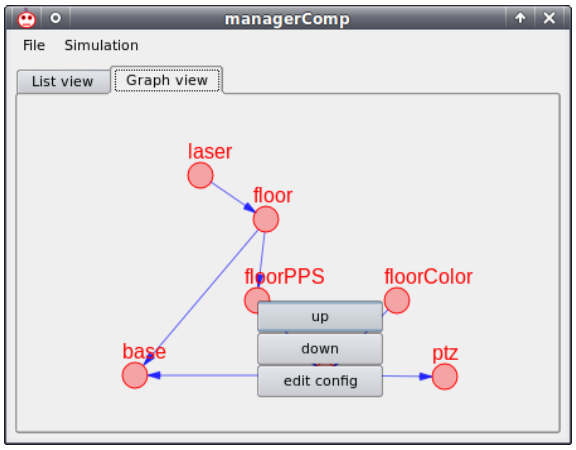


Figura 3.5: Grafo en managerComp

Kdevelop

Es un entorno de desarrollo integrado para sistemas GNU/Linux y otros sistemas Unix, publicado bajo licencia GPL, orientado al uso bajo el entorno gráfico KDE, aunque también funciona con otros entornos. El mismo nombre alude a su perfil: Kdevelop – KDE Development Environment (Entorno de Desarrollo para KDE).

managerComp

La aplicación managerComp permite visualizar, tanto gráficamente como en una lista, el estado de los componentes configurados en tiempo real. A pesar de estar integrado en RoboComp, se detalla independientemente por no ser un componente propiamente dicho. Además de la visualización del estado de los componentes, también permite "arrancarlos" y "pararlos", como puede observarse en la figura 3.5.

Capítulo 4

Antecedentes

4.1. Localización autónoma de robots móviles

La **localización** [7]-[9] es un proceso que consiste en alinear el sistema de coordenadas de un elemento móvil con el del mundo exterior, es decir, permite al móvil conocer su posición dentro del mundo exterior. El problema general de la navegación puede formularse en tres preguntas: ¿dónde estoy?; ¿dónde están las cosas con respecto a mí? y ¿cómo llego hasta mi objetivo? [10]. La localización responde a la primera de ellas y es una cuestión muy importante en un robot autónomo, puesto que si este no conoce su posición con respeco al entorno no puede actuar y decidir qué hacer de forma satisfactoria.

La navegación es habitualmente una tarea complicada debido a diferentes factores, entre ellos pueden mencionarse los límites en el poder computacional, dificultades a la hora de detectar y reconocer objetos, dificultades para evitar colisiones y dificultades a la hora de usar la información proporcionada por los sensores [11].

Para la localización, el robot puede valerse de mapas concretos que conoce a priori y datos obtenidos de sus sensores o bien mediante técnicas más actuales permiten llevar a cabo la localización y creación de mapas simultáneamente (SLAM: Simultaneous Localization and Mapping).

Si el robot parte de una posición inicial conocida, el problema de la localización se resuelve con el *seguimiento*. La posición final del robot se obtiene corrigiendo pequeñas desviaciones en la posición del robot durante el movimiento del mismo debido a rozamiento, errores en los sensores... En este punto se engloba la odometría, entre otros. La odometría estima la posición del robot a partir de una posición inicial teniendo en cuenta el movimiento de las ruedas. Es una localización incremental que presenta errores sistemáticos (debidos a diferentes diámetros de ruedas, falta de alineamiento de las mismas, resolución limitada de los encoders, velocidad limitada de muestreo de los encoders) y no sistemáticos (patinaje de las ruedas, rugosidad del suelo, objetos en el suelo) que hacen que el error en la estimación vaya creciendo y por tanto llegue un momento en el que no se tenga certeza de la posición del robot.

En cambio, si el robot desconoce su posición inicial, nos enfrentamos a un problema de *localización global*. Para ser capaz de situarse en el marco de referencia, el robot ha de valerse de los datos obtenidos por sus sensores.

Por último, también existe el problema de la localización de un grupo de robots, siendo especialmente interesante la situación en la que pueden detectase entre sí puesto que requiere añadir dependencias estadísticas en las estimaciones locales de cada robot.

Existen multitud de algoritmos centrados en resolver el problema del seguimiento. En este ámbito, los filtros de Kalman representan una muy buena solución al problema, de forma que estiman posibles posiciones posteriores del robot condicionadas a los datos proporcionados por los sensores. No obstante, la necesidad de conocimiento inicial hace despreciable estos filtros para resolver el problema de la localización global.

Aunque la mayor parte de la literatura se centre en el problema del seguimiento, podemos encontrar otros métodos igualmente válidos para resolver el problema de la localización global. Por ejemplo, combinando los filtros de Kalman junto con los algoritmos de localización de Markov podemos conseguir buenas aproximaciones de

la posición del robot de forma paramétrica. También es posible usar el algoritmo de Monte Carlo para resolver la misma tarea, de forma que se representan las posibles posiciones posteriores del robot en forma de partículas ponderadas, estimando el estado de forma recursiva utilizando un filtro de Bayes sobre dichas partículas. Esta idea la podemos encontrar en la literatura como filtros de partículas o algoritmos de condensación.

En la localización, como ya hemos comentado, intervienen por un lado el mapa del entorno y por otro los sensores que proporcionan información sobre el mundo exterior al robot. Estos conceptos, junto con los sistemas de referencia, se explican en los siguientes apartados.

4.1.1. Mapas

El entorno de un robot se modela mediante **mapas**, por ejemplo mapas de ocupación, que proporcionan información al robot para tomar decisiones autónomas de movimiento:orientación, giros, velocidad... Para poder construir y/o utilizar estos mapas de forma adecuada, el robot debe estar correctamente localizado.

En función de la zona que se incluya en los mapas, podemos identificar los *mapas globales*, que engloban la totalidad del mundo en el que el robot se mueve; y los *mapas locales*, que sólo recogen el entorno cercano a la posición del robot o región de interés (ROI).

En cuanto al tipo de información que recogen, existen distinos tipos de mapas:

- *Mapas de ocupación:* dividen el espacio en celdas que se identifican con libres u ocupadas
- *Mapas topológicos:* generan nodos en puntos especificos del entorno y si es posible navegar entre ellos se unen mediante un arco, dando lugar a un grafo
- *Mapas métricos:* representan un entorno a una determinada escala

- *Mapas de elementos geométricos:* extraen rasgos geométricos (segmentos rectos, esquinas...) que permitan identificar los datos de los sensores

4.1.2. Sensores

La percepción es un factor crítico para el problema de la localización. Los **sensores** permiten adquirir datos del entorno en el que se mueve el robot. Existe una gran diversidad de sensores que miden diferentes magnitudes físicas: sensores de visión, que permiten captar imágenes a través de una o varias cámaras de video; sensores de posición, como los encoders; o sensores de rango, entre otros. Para elegir un sensor u otro, es conveniente tener en cuenta catacterísticas como precisión, rango de funcionamiento, velocidad de respuesta o calibración [12].

Los sensores de rango permiten al robot detectar posibles obstáculos del entorno en el que debe operar, por lo que también pueden ser usados para la medición de distancias. Existen diferentes tipos de sensores de rango atendiendo al tipo de señal que utilizan:

- basados en ondas acústicas: SONAR
- basados en señales electromagnéticas: infrarrojos, RADAR
- basados en luz coherente: laser rangefinder

El principio de funcionamiento más habitual de los sensores laser rangefinder para medir distancias es el denominado principio de tiempo de vuelo pulsional, que consiste en la medida del tiempo que necesita el un pulso para ir y volver del objetivo. Dado que la electrónica necesaria para medir tiempos del orden de picosegundos (10-12) es muy cara, se han desarrollado otras tecnologías como la detección de desplazamiento de fase o desplazamiento de frecuencia, que emplea la medición de la diferencia de frecuencia entre la onda emitida y la recibida (efecto Doppler) o por triangulación que se basa en relaciones trigonométricas entre ondas emitidas y recibidas.

El uso de estos sensores esta muy extendido en aplicaciones robóticas porque presenta una serie de ventajas. En primer lugar, un láser tiene una alta resolución, acompañado también de una alta precisión. Ello permite utilizarlos de una forma fiable en la navegación de robots; por ejemplo, en entornos dinámicos, es decir, con presencia de seres humanos u otros robots, o en tareas más complejas como la construcción de mapas (mapping) o en la localización del robot en el entorno [13],[14]. Otra ventaja de este tipo de sensores es que presentan un ángulo de detección del orden de 270° por lo que no es necesaria incluir un dispositivo de rotación.

4.1.3. Sistemas de coordenadas

La localización de un robot móvil [7] puede verse como un problema de transformación de coordenadas. Los mapas se describen con respecto a un sistema de coordenadas global que es independiente de la pose del robot. La localización es el proceso de establecer la correspondencia entre el sistema de coordenadas del mapa y el sistema de coordenadas local del robot. Al conocer esta transformación, se permite al robot expresar la localización de los objetos de interés con respecto a su propio marco de referencia, que es un requisito necesario para la navegación.

La posición del robot está determinada por una tupla de coordenadas que sitúan de forma global e unívoca al robot en el mundo por el que se desplaza. La orientación del robot vendrá dada por un ángulo que indicará hacia dónde está mirando el robot.

Centrándonos en el estudio de robots móviles que se desplazan por un terreno plano, consideraremos que el robot se mueve únicamente por el plano XZ, de forma que su posición vendrá dada por una tupla de 2 componentes. Denotaremos como A, al sistema de referencia fijo correspondiente al mundo exterior, donde las coordenadas P_x y P_x determinarán la posición del robot con respecto a ese sistema de referencia.

Por otro lado, para conocer la orientación del robot en dicho plano, necesitaremos otro sistema de referencia centrado siempre en el robot, denotado por B, de forma

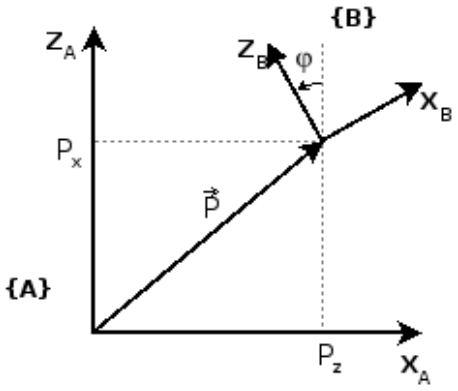


Figura 4.1: Sistemas de referencia para la orientación del robot

que la orientación corresponde al ángulo,φ, formado entre los sistemas de referencias A y B, como puede verse en la figura 4.1.

4.2. Robótica probabilística

La **robótica probabilística** [15] es un enfoque de la robótica en el que se tiene en cuenta la incertidumbre en la percepción y acción del robot. La idea clave de la robótica probabilística es representar explícitamente la incertidumbre usando teoría de la probabilidad. El problema de la localización por tanto se traduce en determinar la probabilidad de que el robot se encuentre en una determinada posición teniendo en cuenta la información proporcionada por sus sensores y los movimientos realizados por el robot. Esto no permite poder localizar el robot incluso desconociendo la posición inicial, es decir, enfrentarnos al problema de la localización global.

En la localización probabilistíca, la información se presenta en términos de densidades de probabilidad. Cada *pose* (posición y orientación) tiene asociada una probabilidad que representa el grado de certeza de que esa sea la posición real del robot. Esta probabilidad se actualiza tras obtener nuevos datos a través de los sensores o de nuevos movimientos del robot.

Por *estado* entenderemos el conjunto de aspectos relacionados con el robot y su entorno. En el estado puede haber variables que cambien a lo largo del tiempo,

como la situación de una persona, y otras que sean estáticas, como la posición de una pared. Denotaremos el estado en el instante de tiempo t con x_t. Como ya hemos comentado, el robot puede adquirir información sobre su entorno mediante sus sensores. La medida tomada en el instante de tiempo t la denotaremos con z_t. Finalmente, denotaremos con u_t a los cambios de estado en el intervalo de tiempo $(t-1;t]$, es decir, información de control. En todas estas variables, un subindice $t_1 : t_2$ indica datos para un intervalo de tiempo, siendo $t_1 \leq t_2$. Un estado se dice que es completo si es el mejor predictor del futuro, es decir, el conocimiento de estados, medidas o datos de control pasados no aportan información que modifique la estimación del nuevo estado. Los procesos que cumplen esta condición se conocen como *cadenas de Markov*.

La evolución del estado, así como de las medidas, está gobernado por leyes probabilísticas. En general, el estado x_t se genera estocásticamente a partir del estado x_{t-1}, por tanto, tiene sentido especificar la distribución de probabilidad con la que x_t se genera. Es lógico pensar que la probabilidad del estado x_t vendrá condicionada por estados pasados, medidas y datos de control, por lo que la distribución de probabilidad tendrá la siguiente expresión: $p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t})$. Si consideramos el estado x_{t-1} completo, la información futura y pasada son independientes y por tanto tenemos la siguiente igualdad:

$$p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t \mid x_{t-1}, u_t) \tag{4.1}$$

Esta probabilidad se conoce como probabilidad de transición de estado y especifica cómo cambia el entorno a lo largo del tiempo en función de u_t.

Si lo que queremos es estimar el estado en el que se generan los datos, suponiendo de nuevo el estado x_t, tenemos que:

$$p(z_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t}) = p(z_t \mid x_t) \tag{4.2}$$

Dicho de otra manera, el estado x_t es suficiente para predecir las medidas z_t. Si el estado x_t es completo, el conocimiento de cualquier otra variable no aporta ninguna información adiccional. A esta probabilidad se la denota como probabilidad de

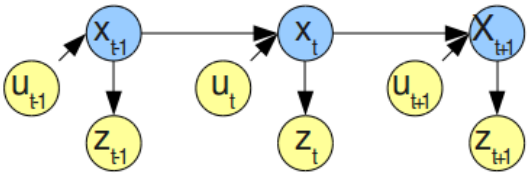


Figura 4.2: Red de Bayes que caracteriza la evolución de los datos de control, estados y medidas

medida y establece la ley probabilística que determina cómo la medida z se obtiene del estado x.

Estas dos probabilidades describen el sistema estocástico dinámico del robot y de su entorno. La figura 4.2 muestra la evolución de estados y medidas según las probabilidades 5.7 y 4.2. El estado en el tiempo t es estocásticamente dependiente del estado $t-1$ y de u_t. La medida z_t depende estocásticamente del estado en el instante de tiempo t. A este modelo se le conoce como *modelo oculto de Markov* (HMM - hidden Markov model).

4.3. Registrado de imágenes

El *registrado de imágenes* [16]-[17] es el proceso de superponer dos o más imágenes de la misma escena y que se han tomado en distintos momentos de tiempo, desde distintos puntos de vista y/o mediante distintos sensores. El registrado de imágenes tiene aplicación en diversos campos, como pueden ser la imagen médica, la cartografía o la visión artificial.

Para llevar a cabo el registrado de imágenes pueden usarse marcadores externos incluidos en el espacio de la imagen o bien basarse en información extraída directamente de la imagen. A su vez, existen distintos métodos según el tipo de información extraída de la imagen que se emplee:

- *Métodos basados en características.* Estos métodos extraen características de la imegan para estimar los parámetros de registrado. Mayoritariamente, las

características que se extraen son puntos, líneas o curvas y regiones.

- *Métodos basados en área.* Estos métodos hacen uso del contenido completo de la imagen sin intentar detectar objetos o regiones característicos.

En el trabajo que nos ocupa, nos centraremos en métodos que emplean la información global de la imagen para establecer el registro entre ellas y en particular emplearemos la intensidad de los píxeles de las imágenes. Estos métodos pueden clasificarse en tres grupos:

- *Basados en correlación.* Comparan directamente las intensidades de las dos imágenes sin hacer ningún análisis de estructuras. Generalmente se emplea la correlación cruzada normalizada y sus variantes.

- *Basados en Fourier.* Explotan la transformada de Fourier de las imágenes en el dominio de la frecuencia.

- *Basados en información mutua.* Se basan en maximizar la medida de información mutua, un concepto derivado de la teoría de la información.

La elección de uno u otro metodo debe hacerse de acuerdo a la naturaleza de las imágenes que deben registrarse.

4.3.1. Template Matching

Dada una imagen, el *template matching* consiste en encontrar los lugares de una imagen que coinciden con una subimagen (normalmente llamada máscara o plantilla -*template* en inglés) por lo general menor que la imagen original [18],[19]. Por tanto, es necesaria una medida de simulitud entre la máscara y la región de la imagen que coincide con la posición actual de la máscara.

Esta medida de simulitud se calcula para parejas de ventanas de la máscara y de la imagen original. Una vez calculadas las medidas de similitud en las distintas posiciones, se busca el valor máximo. La pareja de ventanas para la que la medida de

similitud es máxima son los puntos de correspondencia y en esa posición la máscara tiene la mayor similitud con la imagen original [16].

Medidas de similitud basadas en correlación

La *correlación cruzada* y sus variantes son medidas de similitud, es decir, dan una medida del grado de parecido entre una imagen y una máscara. Este tipo de métodos son generalmente útiles en imagenes que desplazadas entre si por transformaciones afines o rígidas [18].

Una manera de encontrar los puntos en los que máscara e imagen original se superponen de mejor manera es tratar la máscara como un filtro y calcular la suma de los productos (o una versión normalizada) para cada posición de la máscara en la imagen original. Las posiciones en las que el parecido entre las dos imágenes es mayor son aquellas en las que el valor de la correlación es máximo. Una opción alternativa es trabajar en el dominio de la frecuencia, haciendo uso del teorema de la correlación.

Definiremos $f(x, y)$ como una imagen que contiene objetos o regiones. Si queremos determinar si f contiene un objeto determinado o una región en la que estamos interesados, definiremos $w(x, y)$ como ese objeto o región (ésta será la máscara). Por tanto, si existe un emparejamiento entre ambas, la correlación cruzada entre ambas imágenes será máxima en la posición en la que w se corresponde mejor con f. Matemáticamente, la correlación cruzada [19] entre dos imágenes$f(x, y)$ y $w(x, y)$ se define como

$$f(x, y) \circ w(x, y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f^*(m, n) w(x + m, y + n) \tag{4.3}$$

donde f^* es el complejo conjugado de f. En nuestro caso, trataremos con imágenes reales, por lo tanto $f = f^*$.

La figura 4.3 muestra la disposición de las dos imágenes para calcular la correlación. Para cada posición (s, t) dentro de $f(x, y)$, aplicar la ecuación 4.3 da lugar a un valor de correlación. Conforme s y t varían, $w(x, y)$ se desplaza por la imagen,

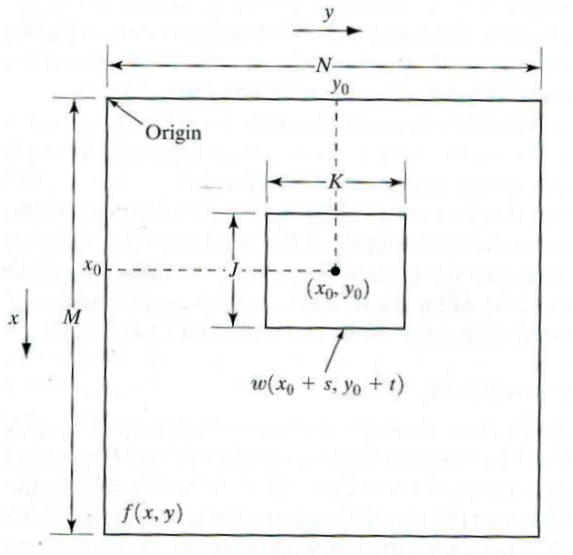


Figura 4.3: Correlación entre las imágenes f y w en el punto (x_0, y_0)

dando la función $c(s,t)$. El valor máximo de ésta última indica la posición donde $w(x,y)$ se corresponde mejor con $f(x,y)$.

La desventaja de la correlación definida en 4.3 es sensible a los cambios de amplitud de $f(x,y)$ y de $w(x,y)$. Para evitar estos efectos, la otra medida de simulitud que suele emplearse son los *coeficientes de correlación*, definidos como:

$$\gamma(x,y) = \frac{\sum_s \sum_t \left[f(s,t) - \overline{f}(s,t)\right]\left[w(x+s,y+t) - \overline{w}\right]}{\sqrt{\sum_s \sum_t \left[f(s,t) - \overline{f}(s,t)\right]^2 \sum_s \sum_t \left[w(x+s,y+t) - \overline{w}\right]^2}} \tag{4.4}$$

donde $s = 0, 1, 2, \ldots, M-1$, $t = 0, 1, 2, \ldots, N-1$, $\overline{w}$ es la media de los valores de w (se calcula una única vez) y $\overline{f}(x,y)$ es la media de $f(x,y)$ en la zona en la que coincide con la posición concreta de w, y los sumatorios incluyen los coordenadas que son comunes a f y w. El coeficiente de correlación $\gamma(x,y)$ estará en el rango de -1 a 1, independientemente de cambios en la amplitud de $f(x,y)$ y $w(x,y)$.

Los coeficientes de correlación pueden calcularse en el dominio de la frecuencia, pero esto suele ser más dificil de implementar y por tanto se calculan directamente

aplicando la ecuación 4.4.

4.4. Modelos de medida basados en correlación

Un modelo de medidas del entorno describe el proceso de formación por el cual las medidas del sensor se generan en el mundo físico. Dependiendo del sensor se utiliza un modelo u otro: los sensores de imagen se modelan mejor mediante geometría proyectiva mientras que por ejemplo un sensor SONAR es mejor modelarlo describiendo la onda acústica y sus reflexiones en las superficies del entorno.

Formalmente, el modelo de la medida se expresa como la distribución de probabilidad condicional $p(z_t|x_t, m)$ donde x_t es la *pose* del robot, z_t es la medida en el instante de tiempo y m es el mapa del entorno. Este mapa es un listado de objetos en el entorno y sus localizaciones: $m = m_1, m_2, \ldots, m_N$, donde N es el número total de objetos en el entorno y cada m_n con $1 \leq n \leq N$ especifica una propiedad.

Los mapas se pueden indexar de dos formas: los basados en características y los basados en posición. En los mapas basados en características, m_n contiene, junto a la posición en cartesianas de la característica, que puede ser un objeto, esquina o una intersección, las propiedades de ésta. En los mapas basados en posición, m_n sólo contiene una posición. En mapas en los que la altura no se tiene en cuenta, es habitual denotar $m_{x,y}$ en lugar de m_n para hacer mención explícita que $m_{x,y}$ es una propiedad de la coordenada (x, y) del mundo. Los mapas de celdas de ocupación son mapas de localización que asignan a cada posición (x, y) un valor de ocupación binario que especifica si esa posición está ocupada o no por un objeto.

En cualquier caso, resulta importante destacar que el entorno físico real y el mapa que lo representa no son la misma cosa. Así, las medidas que obtienen los sensores se asocian a elementos del entorno, no a la representación en el mapa de estos elementos. Sin embargo, el modelado de los sensores se condiciona, tradicionalmente, al mapa m.

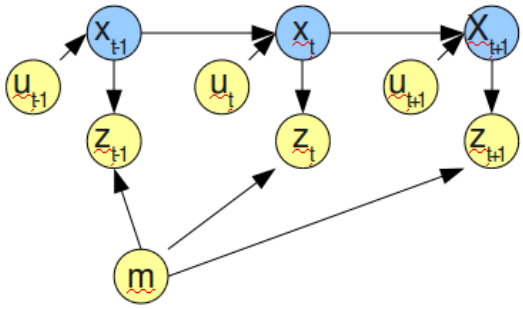


Figura 4.4: Grafo de la localización del robot.

En este caso, la figura 4.2 se ve modificada por el conocimiento del mapa, resultado el gráfico mostrado en 4.4.Los nodos en amarillo son conocidos: el mapa m, las medidas z y los datos de control u. El objetivo es inferir la *pose* del robot. El robot conoce el mapa de su entorno y su objetivo es determinar su posición relativa al mapa dadas las percepciones del entorno y sus movimientos.

Por tanto, combinando lo expuesto en los apartados 4.2 y 4.3, podemos enfrentar el problema de la localización desde un punto de vista probabilístico haciendo uso de medidas de similitud.

Denotaremos por m_{local} el mapa local que obtenemos a partir de cada *scan* que realiza el láser. Este mapa será un mapa de celdas de ocupación. En el modelo de medida compara el mapa local m_{local} con el mapa global m, de tal forma que cuanto mayor sea el parecido entre uno y otro, mayor es la probabilidad $p(m_{local} \mid x_t, m)$. Puesto que el mapa local se crea con respecto al sistema de referencia del robot, es necesario transformarlo al sistema de referencia global. Para llevar a cabo esta transformación, podemos aplicar una traslación junto con una rotación que mapee z_t^k en el sistema de coordenadas global:

$$\begin{pmatrix} x_{z_t^k} \\ y_{z_t^k} \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_{k,sens} \\ y_{k,sens} \end{pmatrix} + z_t^k \begin{pmatrix} \cos(\theta + \theta_{k,sens}) \\ \sin(\theta + \theta_{k,sens}) \end{pmatrix}$$

donde $x_t = (x, y, \theta)^T$ es la *pose* del robot en el instante de tiempo t; $(x_{k,sens}, y_{k,sens})^T$

posición del sensor en el sistema de coordenadas local del robot y $\theta_{k,sens}$ es la orientación angular del rayo del sensor, con respecto al sistema de referencia local. Las coordenadas $(x_{z_t^k}, y_{z_t^k})$ tienen sentido cuando el sensor detecta un obstáculo. Si el sensor toma su valor máximo $z_t^k = z_{max}$, estas coordenadas no tienen sentido en el mundo físico, por lo que las lecturas iguales al máximo deben ser descartadas.

Una vez que tenemos ambos mapas en el mismo sistema de referencia, podemos compararlos haciendo uso de 4.4 para obtener el coeficiente de correlación entre el mapa global y el local en el instante de tiempo t: γ_{m,m_{local},x_t}. Como se comentó previamente, el valor de estos coeficientes de correlación está acotado en un rango de valores igual a $[-1, 1]$. El valor $p(m_{local} \mid x_t, m) = max\{\gamma_{m,m_{local},x_t}, 0\}$ se interpreta como la probabilidad del mapa local condicionado al mapa global y la *pose* x_t del robot. Puesto que el mapa local se genera a partir de un único *scan*, esta última probabilidad sustituye a la probabilidad de medida $p(z_t \mid x_t, m)$. Para mejorar los resultados que se obtienen con este modelo de medida, se suele convolucionar el mapa m con un filtro Gaussiano, que suavice las probabilidades calculadas.

Capítulo 5

Algoritmo de localización basado en celdas de ocupación

En este capítulo se describe el método implementado para poder llevar a cabo la localización de robots autónomos en entornos cerrados cuyo mapa es conocido. El resultado es el desarrollo del componente *localizationComp*. En primer lugar, se expondrán los datos de partida con los que se cuenta. Posteriormente, se describirá el algoritmo para finalizar con una descripción del componente.

5.1. Datos de partida

El método planteado se basa en imágenes para establecer la posición del robot dentro de las coordenadas del mundo. Por tanto, son fundamentales dos imágenes: una que se corresponda con un mapa del entorno cerrado y otra que refleje la información obtenida por los sensores del robot.

5.1.1. Mapa global

Esta imagen es un mapa de la planta del entorno cerrado en el que el robot navegará y dentro del cual queremos que sea capaz de localizarse. Es un mapa métrico hecho a una escala determinada que es fundamental conocer. Al tratarse de una imagen digital, cada pixel puede considerarse una celda de ocupación, puesto

que permite discretizar el entorno. Por tanto, el conjunto de todas las celdas de ocupación, $\mathbf{x}_{i,t}$, conforman una partición del espacio de todas las posiciones posibles, dominio(X_t):

$$\text{dominio}(X_t) = \mathbf{x}_{1,t} \cup \mathbf{x}_{2,t} \cup \ldots \mathbf{x}_{K,t}$$

Para cada par de celdas $\mathbf{x}_{i,t}$, $\mathbf{x}_{k,t}$ con $i \neq k$, tenemos que $\mathbf{x}_{i,t} \cap \mathbf{x}_{k,t} = \emptyset$, es decir, las celdas de ocupación no se superponen. Se indicará si la celda está libre u ocupada en función del valor que presente. Las celdas ocupadas tendrán valor 0 mientras que a las libres se les asignará el valor 1. Para obtener mejores resultados, se aplicará un filtro gaussiano que suavizará los valores iniciales. El filtrado lo haremos con una matriz 3x3:

$$\begin{pmatrix} \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \\ \frac{2}{16} & \frac{4}{16} & \frac{2}{16} \\ \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \end{pmatrix} \tag{5.1}$$

5.1.2. Mapa local

Es una imagen sintética creada a partir de un scan del láser. La información suministrada por el sensor láser en un escaneo es normalmente muy densa y tiene una precisión angular bastante buena. Las imágenes de rango suministradas por los sensores de rango tienen normalmente la forma $(\rho, \theta)_{l|l=1\ldots N_R}$, en la cual $(\rho, \theta)_l$ son las coordenadas polares de un punto de la lectura de rango (ρ_l es la medida de distancia de un obstáculo al eje de rotación del sensor en la dirección θ_l), y N_R es el número de lecturas de rango. Por tanto, las medidas escaneadas son adquiridas por el láser con una resolución angular $\Delta\theta = \theta_l - \theta_{l-1}$. El rango angular de la medida, R; la resolución angular del láser, $\Delta\theta$; y el número de lecturas de rango, N_R, se relacionan con la resolución angular por la siguiente ecuación:

$$N_R = \frac{R}{\Delta\theta} \tag{5.2}$$

Los puntos $(\rho, \theta)_l$ que proporciona el láser se transforman a coordenadas cartesianas dentro del sistema de referencia del robot. Las coordenadas $(x, z)_l$ con respecto

al sistema de referencia local son:

$$x_l = \rho_l \sin\theta_l \tag{5.3}$$

$$z_l = \rho_l \cos\theta_l \tag{5.4}$$

Esta representación del mundo obtenida mediante la lectura del láser se transforma a celdas de ocupacion y se define su valor, con el mismo criterio que en el mapa global, para obtener la imagen binaria. Para llevar a cabo correctamente esta transformación es necesario tener en cuenta la escala del mapa global, para poder crear una imagen del láser cuyas dimensiones sean acordes a las dimensiones que tendría su representación en el mapa global. Sea (m, n) la posición en la imagen del mapa local del punto $(x, z)_{local}$. La transformación a aplicar es:

$$m = \left\lfloor \rho_{max}\text{escala} \ \left(1 + \frac{x}{\rho_{max}}\right) \right\rfloor \tag{5.5}$$

$$n = \left\lfloor \rho_{max}\text{escala} \ \left(1 - \frac{x}{\rho_{max}}\right) \right\rfloor \tag{5.6}$$

A este mapa local se le aplicará un filtro gaussiano con la misma máscara que la definida para el mapa global en eq:filtro.

5.2. localizationComp

El componente ha sido implementado en el lenguaje de programación de alto nivel C++ y en cuanto a la estructura del programa hemos utilizado programación orientada a componentes. Para obtener una visión global de como interactúa nuestro componente con los ya existentes se muestra el grafo de componentes en managerComp (ver figura

A continuación se describen brevemente los componentes con los que se relaciona *localizationComp*:

- *joystickComp* permite controlar el desplazamientos del robot, hecho fundamental para poder llevar a cabo la localización del mismo

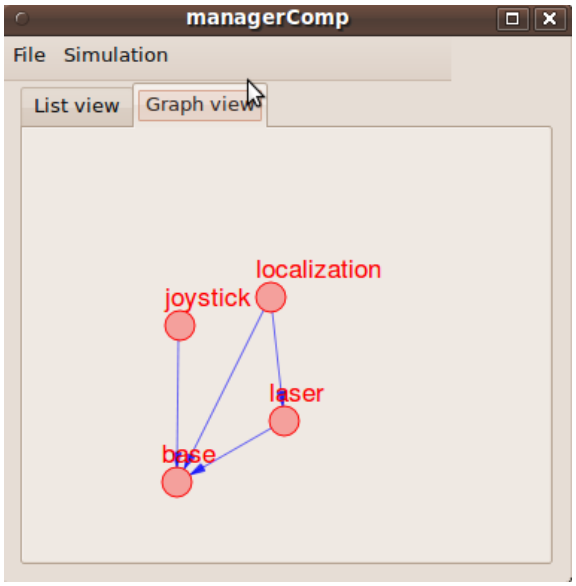


Figura 5.1: Relaciones del componente *localizationComp* con otros componentes de RoboComp

- *laserComp* establece la conexión con el sensor láser para obtener los datos referentes a las mediciones que realiza en cada barrido: la distancia a cada punto del scan y el ángulo correspondiente, $(\rho, \theta)_{l|l=1...N_R}$
- *differentialrobotComp* controla la base robótica lo que permite desplazar el robot por el mundo

Antes de pasar a describir el algoritmo de localización, expondremos brevemente el funcionamiento global del componente *localizationComp*, para saber en qué momento se hace uso del algoritmo. El diagrama de flujo del componente se muestra en la figura 5.2.

En primer lugar se lleva a cabo la creación e inicialización de las variables para pasar después a cargar el mapa global del entorno (5.1.1). Tras esto, se adquieren los datos de la odometría, que servirán de *ground truth* o datos fiables con los que posteriormente comparar la estimación realizada. Hay que mencionar que estos datos de la odometría no presentan errores debido a que estamos trabajando con simulación y no con un entorno real, por lo que podemos considerar que la medida de

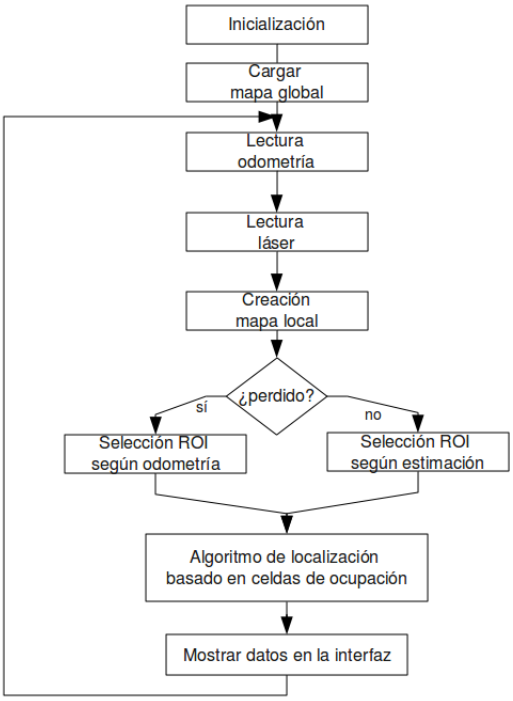


Figura 5.2: Diagrama de flujo del componente *localizationComp*

la odometría es la medida real de la posición del robot. En siguiente lugar, se realiza una percepción del entorno mediante una lectura del láser. Con este scan, se crea la imagen del mapa local (5.1.2). Una vez que tenemos las dos imágenes, se define la ROI del mapa global con la que vamos a calcular los coeficientes de correlación. En este punto, aplicamos el algoritmo de localización para acabar presentando los resultados en la interfaz gráfica de usuario (5.4).

5.3. Algoritmo

El algoritmo implementado basa la localización en medidas de similitud entre las dos imágenes presentadas en el apartado 5.1. Para ello, hace uso de la correlación cruzada (ver apartado 4.3.1). El objetivo es determinar la probabilidad de estado: $p(x_t \mid z_t, u_t)$. Como ya se comentó, el mapa local se crea con respecto al sistema de referencia del robot, que no tiene por qué estar alineado con el sistema de referencia global. La posición y orientación para las que el grado de similitud entre ambos mapas sea mayor será considerada la *pose* del robot, $x_t = (x, y, \theta)^T$.

Con el fin de reducir la incertidumbre inicial y evitar posibles ambigüedades a la hora de detectar la posición inicial del robot, se tomará como dato de partida los datos de la odometría. Asumir que el robot se encuentra inicialmente en una posición determinada nos permite reducir el campo de búsqueda inicial a una región centrada en dicha posición, pero en ningún caso se toma la posición inicial de la odometría como primera posición estimada. Posteriormente, los datos de la odometría del robot no se consideran en el algoritmo.

El algoritmo que se sigue en cada lleva a cabo en cada iteración es el siguiente:

1. adquisición de datos del sensor láser, z_t
2. creación del mapa local, m_{local}
3. $\forall\theta$ definido
   - *a*) rotar m_{local}
   - *b*) calcular correlación cruzada entre m_{local} y m
   - *c*) detección del máximo, $\gamma^{max}_{x,z,\theta}$
   - *d*) si $\gamma^{max}_{x,z,\theta} \geq \gamma_{max}$
     - almacenar variables
4. actualización de la *pose*, $x_t = (x, y, \theta)^T$

En cada iteración, se realizará una lectura de los datos del sensor y se procederá a su tratamiento para crear el mapa local. Este dato se corresponde con la medida z_t, es decir, la observación real del entorno del robot. Una vez obtenido el mapa local, se analizará el grado de similitud de éste con los distintos estados del robot, x_t. Cada estado x_t estará definido por un par de coordenadas en el sistema de referencia global (x, z) junto con un ángulo de rotación, θ. La medida de similitud,$\gamma_{x,z,\theta}$ se calcula la correlación en todas las posiciones (x, z) en las que el mapa global y el local rotado θ se superponen.

A continuación se describen con detalle los pasos que componen el método propuesto. Los siguientes pasos se realizan para cada uno de los ángulos definidos.

Paso 1. Adquisición de los datos del láser. En este primer paso se hace una lectura del sensor, obteniendo un listado de pares de distancias y ángulos, correspondientes al scan hecho.

Paso 2. Creación del mapa local. En este paso se transforman los datos del láser en una imagen binaria. Cada punto del scan se tranforma a coordenadas cartesianas en el sistema de referencia local. Teniendo en cuenta la escala del mapa global, se crea una imagen del tamaño adecuado y se discretiza el entorno local del robot, identificando qué píxeles están ocupados. Los datos en los que el sensor devuelve su valor máximo no son tenidos en cuenta puesto que no implica la detección de un obstáculo. Una vez obtenida esta imagen binaria, se le aplica un filtrado gaussiano.

Paso 3. Determinar ángulos de rotación. Se define un número determinado de ángulos de rotación con los que se calculará la medida de similitud entre el mapa global y el mapa local. Es necesario llegar a un compromiso entre la granularidad deseada y el coste computacional que requiere explorar todos los ángulos.

Paso 3*a*. Rotación. Se toma el mapa local comocomo punto de partida y se rota con respecto al punto central del mismo para obtener el mapa láser rotado. Hay que tener en cuenta que el mapa rotado contenga integramente la información del mapa local original, por ello será siempre de igual o mayor tamaño, lo que dependerá del ángulo de rotación. Si las dimesiones del mapa local original son $W_l \times H_l$ y hemos de rotarlo un ángulo θ, las dimensiones del mapa local rotado $W_r \times H_r$ serán:

$$\begin{pmatrix} W_r \\ H_r \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} W_l \\ H_l \end{pmatrix} \tag{5.7}$$

Paso 3*b*. Correlación. En este paso es donde se realiza la medida de similitud entre las dos imágenes mediante el cálculo de los coeficientes de correlación según la ecuación 4.3 . Esta correlación se calcula en todos los puntos en los que el mapa

local rotado se superpone completamente en el mapa, por lo tanto para cada ángulo θ obtenemos una matriz $\gamma_{x,z,\theta}$. Sean $W_g \times H_g$ las dimesiones del mapa global, luego las dimesiones de la matriz de los coeficientes de correlación serán

$$W_c = W_g - W_r + 1 \tag{5.8}$$

$$H_c = H_g - H_r + 1 \tag{5.9}$$

Cada uno de los elementos de esta matriz es un estado x_t, candidato a determinar la *pose* del robot.

Paso 3*c*. Máximo. Una vez calculados los coeficientes de correlación, identificamos la posición del máximo $\gamma^{max}_{x,z,\theta}$ que corresponderá con la posición (x, z) del mapa global donde ambos mapas tienen mayor grado de similitud. De este modo, seleccionamos el mejor estado x_t de entre todos los posibles para un determinado θ.

Paso 3*d* El valor máximo obtenido se compara con el máximo que hubiera almacenado de iteraciones anteriores para determinar si el estado identificado en esta iteración es mejor. En caso de que sea así, se almacena la información correspondiente al estado x_t.

Paso 4. Actualizar posición. En este último paso, actualizaremos la *pose* del robot con los valores correspondientes a γ_{max}: $x_t = (x, y, \theta)^T$.

5.4. Interfaz

Para mostrar los resultados del obtenidos en tiempo real, se ha implementado una interfaz gráfica de usuario que se muestra en la figura 5.3. Se muestra el mapa del entorno en el que se desea navegar y a la derecha de este, la *pose* real (datos obtenidos del simulador Player/Stage) y la *pose* estimada por el sistema. Se incluye también un botón de reset que permite inicializar el algoritmo al estado inicial en el que se desconoce la posición estimada mediante los coeficientes de correlación y se toma como punto de partida una región de interés (ROI) centrada en la posición dada por la odometría.

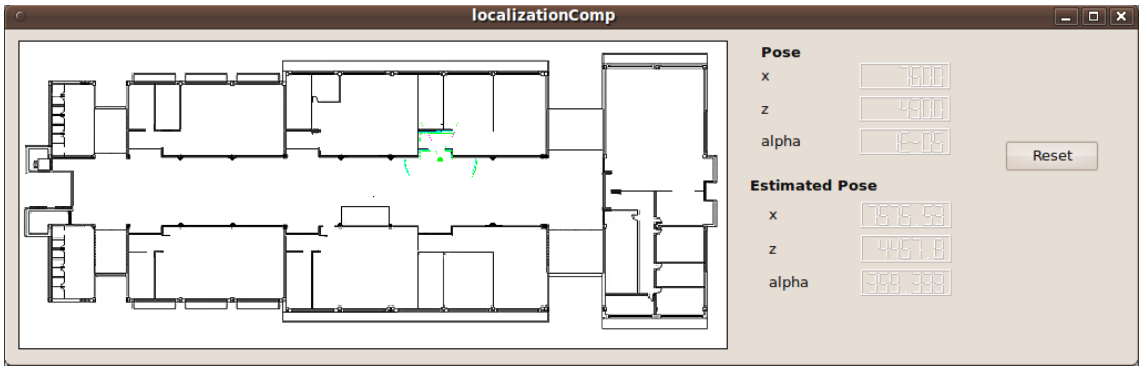


Figura 5.3: Interfaz gráfica de usuario del componente *localizationComp*

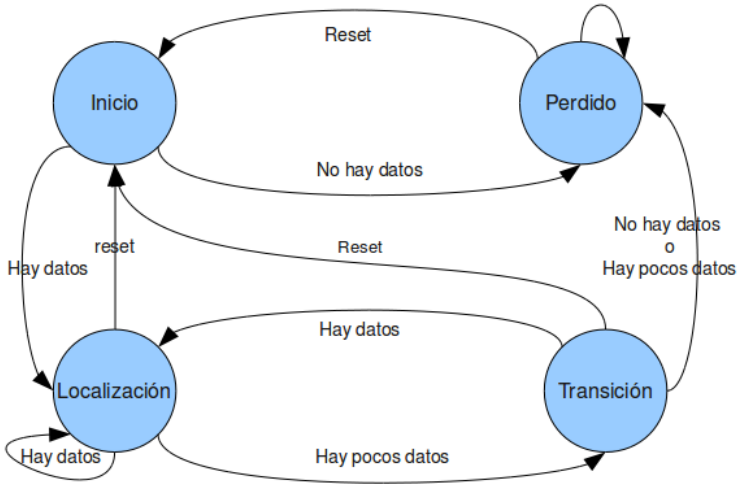


Figura 5.4: Diagrama de estados

Para concluir esta sección se presenta un diagrama de estados que recoge el funcionamiento del sistema según la cantidad de datos de los que se disponga, con la posibilidad siempre de volver al estado inicial mediante el botón de reset.

Capítulo 6

Resultados

En este capítulo se presentan los experimentos realizados a partir del desarrollo del componente *localizationComp*. El entorno seleccionado para llevar a cabo las pruebas ha sido el pabellón de informática de la Escuela Politécnica, puesto que permite llevar a cabo las pruebas en dos fases, una inicial haciendo uso del simulador Player/Stage y una vez que se hayan obtenido resultados adecuados, pasar a realizar una prueba real con un robot de la familia RobEx. En este trabajo sólo se recogen resultados de la fase de simulación debido a limitaciones temporales.

6.1. Datos iniciales para la estimación

El mapa con el que se ha trabajado se muestra en en la figura 6.1. Este es un mapa métrico con una escala de 7,8947 píxeles por metro y dimensiones de 640×278 píxeles, lo que da lugar a un entorno real de 81067×35213 mm. Teniendo en cuenta estos datos, la resolución en las coordenadas X y Z será:

$$\text{resolución} = \frac{1 \text{ pixel}}{\text{escala (pix/mm)}} = 126{,}66 \text{ mm} \tag{6.1}$$

El componente *laserComp* proporciona un listado de distancias y ángulos de las medidas del sensor láser, según lo expuesto en la sección 5.1. El láser simulado nos da un rango de acción $\rho_{max} = 4096$ mm y un rango angular de 240°. En cada scan, obtenemos datos de 768 rayos, por lo que, aplicando la ecuación 5.4, la resolución

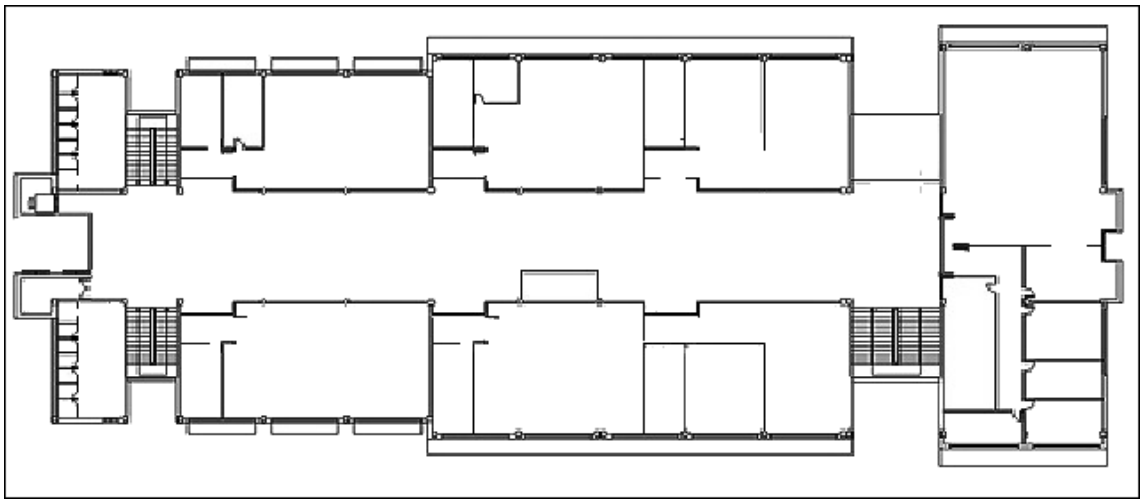
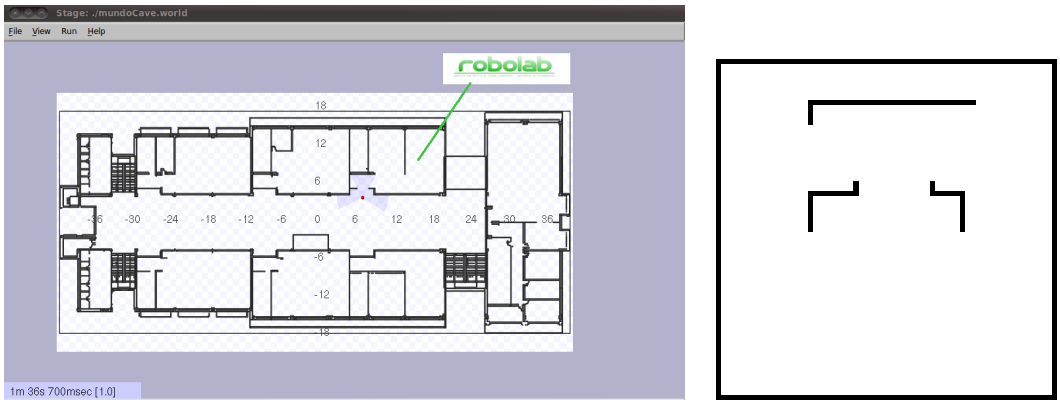


Figura 6.1: Mapa del entorno

angular del láser simulado es 0,3125°. Transformar el rango total que cubre el sensor, empleando la escala del mapa global, resulta en una imagen de 67 × 67 píxeles. Suponiendo que, por ejemplo, el robot se encuentre a la entrada del labotorio (ver 6.2(a)), la imagen que se crea como mapa local puede verse en la figura 6.2(b)[1].

El efecto de filtrado gaussiano aplicado las imágenes de partida junto con la selección de una ROI permite mejorar la detección del máximo, ya que reducimos el número de máximos locales ala vez que aumentamos el valor de estos. Estos resultados se exponen en la figura 6.3, donde puede observarse de forma gráfica el resultado de los coeficientes de correlación cuando se usa el mapa global completo y las imágenes sin filtras (6.3(a)) y esos mismos coefientes cuando se usa una ROI y ambas imágenes han sido previamente filtradas (6.3(b)). Es evidente que la detección en el segundo caso es mucho más fácil que en el primero.

[1]El borde negro mostrado en la figura 6.2(b) tiene como único propósito definir los límites de la imagen, pero dicho borde no está presente en la imagen usada en el algoritmo



(a) Mapa global - Robot situado a la entrada de Robolab

(b) Mapa local - Imagen del láser

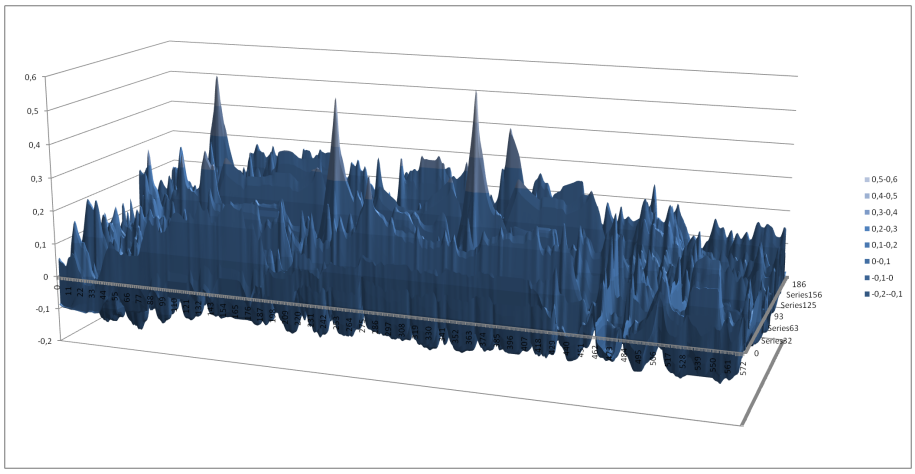
Figura 6.2: Mapas

6.1.1. Resolución angular

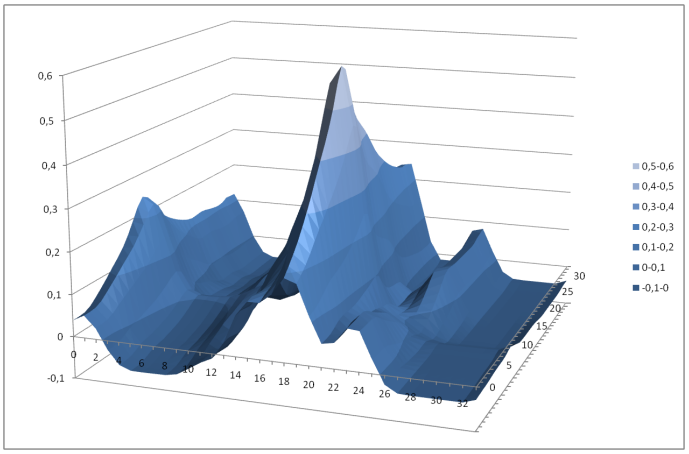
La resolución angular con la que trabajamos viene determinada por el número de ángulos que se rote el mapa local para posteriormente calcular la correlación. En la iteración inicial (en la que no se ha realizado una estimación anteriormente) se hace un barrido completo con 36 ángulos, lo que nos da una resolución inicial de 10°. En las iteraciones sucesivas, puesto que ya se ha hecho una estimación, se reduce el rango de búsqueda a un cuadrante y también se reduce el número de ángulos a 16. Esto nos permite incrementar la resolución a 5,625°.

6.1.2. Cantidad de datos necesarios para la estimación

Para poder realizar una estimación de la posición, el sensor láser ha de detectar una cierta cantidad de obstáculos; esto es, al menos un número mínimo de haces deben presentar distancias con $\rho < \rho_{max}$. En caso de que no se obtengan suficientes datos, los resultados de los coeficientes de correlación, γ_{max}, aunque correctos, no pueden considerarse válidos. Por tanto, se hace necesario definir este número mínimo de datos para poder considerar como correcta la estimación. Para ilustrar este razonamiento, supongamos que realizamos un giro de 360° en la entrada del laboratorio (mostrada anteriormente en 6.2(a)). Los resultados obtenidos cada 45° pueden observarse en la figura 6.4. En verde se muestran los datos del láser pintados te-



(a) Correlación mapa local con mapa global sin filtrado



(b) Correlación mapa local con ROI con filtrado

Figura 6.3: Coeficientes de correlación

θ_{real}	γ_{max}	N	Figura
0°	0,615757	591	6.4(a)
45°	0,460299	536	6.4(b)
90°	0,539685	384	6.4(c)
135°	0,377090	310	6.4(d)
180°	0,287832	208	6.4(e)
225°	0,360723	250	6.4(f)
270°	0,580412	397	6.4(g)
315°	0,532621	532	6.4(h)

Cuadro 6.1: Valores del coeficiente de correlación máximo y número de datos del láser para distintos ángulos de rotación

niendo en cuenta la *pose* real, mientras que en cyan o rojo se muestran los datos del láser con la *pose* estimada. En esta secuencia de imágenes podemos observar que conforme el láser detecta menos obstáculos (es decir, se gira hacia el pasillo), el ángulo estimado se aleja más del real.

La tabla 6.1 recoge el valor del coeficiente de correlación máximo, γ_{max}, y el número de datos del láser, N, para los distintos ángulos de rotación θ seleccionados. Como puede observarse, los ángulos para los que $\gamma_{max} < 0{,}4$ son los ángulos en los que el ángulo estimado no se corresponde con el ángulo real. En estos ángulos, el número de datos válidos es inferior a 300. Si realizamos varias rotaciones consecutivas, el diagrama de dispersión que obtenemos es el mostrado en la figura 6.5. A la vista de estos datos, corroborados por otras pruebas experimentales como la presentada pero en otras posiciones del mapa, estableceremos de forma empírica como umbral a partir del cual considerar que hay suficientes datos. Será ncesario disponer de al menos el 40 % de N_R, en nuestro caso, 307 datos. Siguiendo este criterio, los datos pintados en rojo en la figura 6.4 son aquellos en los que $N < 0{,}4N_R$, mientras que en cyan se dibujan el resto.

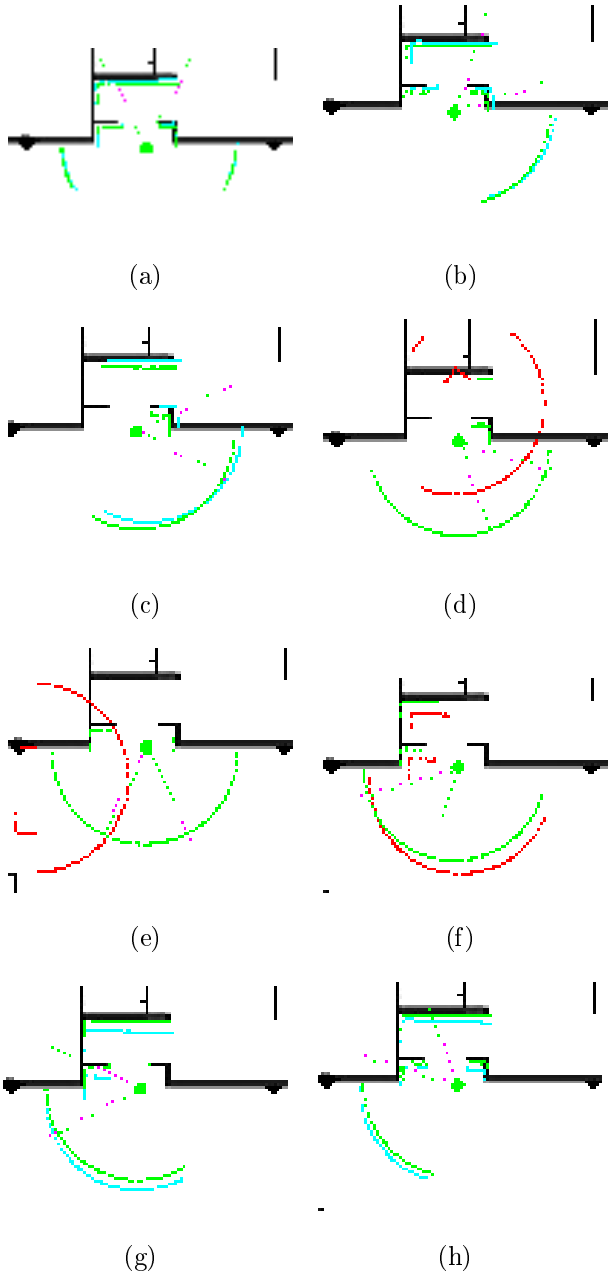


Figura 6.4: Localización durante un giro de 360°

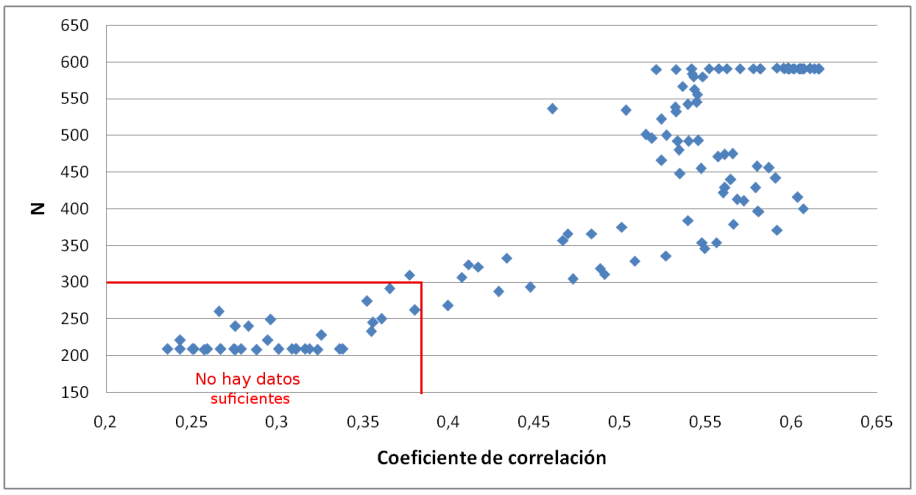


Figura 6.5: Relación entre los coeficientes de correlación y el número de datos obtenidos con el laser

6.2. Errores en la estimación

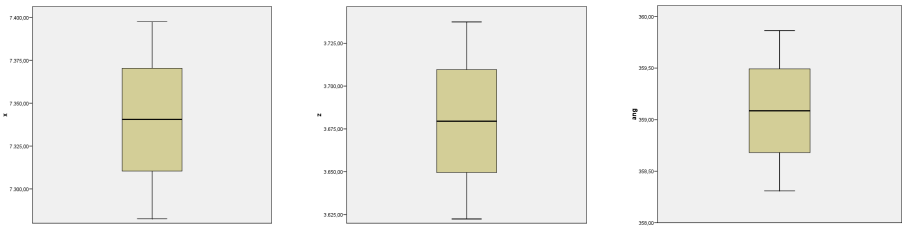
Puesto que lo que tenemos es una medida real y una medida estimada, hemos de medir el error que comete el algoritomo diseñado. Tendremos en cuenta, en primer lugar, el error que cometemos en la medida cuando el robot no se desplaza, para identificar el error de medida. Posteriormente, veremos el error cometido en la estimación de la posición alo largo de un recorrido con el robot.

6.2.1. Error en comportamiento estático

Para calcular el error de medida, dejaremos el robot en una posición determinada y obtendremos diferentes estimaciones de esa *pose*. Los resultados obtenidos son los mostrados en la tabla 6.2 y los diagramas de caja de la figura 6.6. Cabe destacar que como la desviación en las coordenadas X y Z es inferior a la resolución calculada en la ecuación 6.1, estamos obteniendo siempre el máximo en el mismo píxel. Del mismo modo, la desviación en el ángulo de rotación es tambien inferior a la resolución expuesta en 6.1.1, lo que determina que siempre estamos eligiendo el mismo intervalo angular.

Variable	valor real	valor medio	desviación
Eje X (mm)	7216, 02	7340,35	35,36
Eje Z (mm)	3646, 2	3679,65	35,36
Ángulo de rotación (°)	1,4	359,08	0,48

Cuadro 6.2: Errores de medida de la posición estimada



(a) Coordenada X (mm) (b) Coordenada Z (mm) (c) Ángulo de rotación (°)

Figura 6.6: Error de medidad - diagramas de caja

6.2.2. Error en comportamiento dinámico

En esta sección vamos a calcular el error que se produce en la estimación de la *pose* del robot cuando éste se está moviendo. Para ello, se ha llevado a cabo un recorrido con el robot, mostrado en la figura 6.7. Definiremos el error absoluto de estimación en el instante de tiempo k, e_k, como

$$e_k^T = \mid x_{real}^T - x_{real}^T \mid \tag{6.2}$$

donde x_{real}^T y x_{real}^T son la *pose* real y estimada, respectivamente. Esto nos da el error absoluto en cada una de las componentes.

La figura 6.8 muestra la planta del recorrido real realizado, en rojo, y el estimado, en azul. Puede observarse que conforme nos desplazamos a la derecha, la posición estimada se va separando de la real. Este hecho se debe a la disposición simétrica de las dos habitaciones del laboratorio, que da lugar a máximos locales que espacialmente son mejor estimación de la posición que el máximo global, que es el que detecta el algoritmo. La situación a la que nos referimos se muestra en el figura 6.10. Una posible solución a esta problemática es emplear la trayectoria para llevar a cabo la desambiguación entre máximos locales. Con este enfoque, la probabilidad

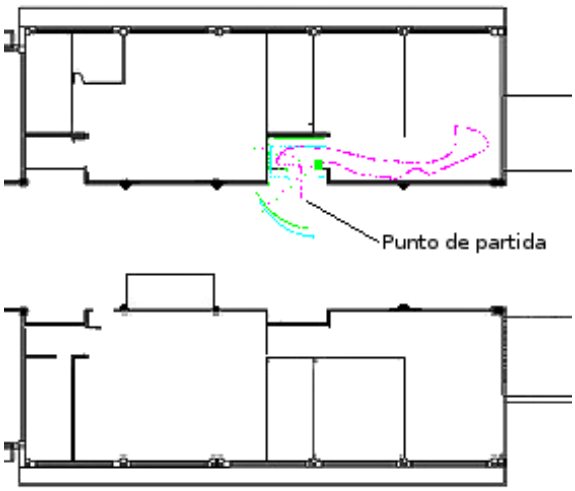


Figura 6.7: Recorrido realizado

Variable	error max	error medio	desviación
Eje X (mm)	6115,2	1227,95	1920,47
Eje Z (mm)	1977,33	287,50	280,28
Ángulo de rotación (°)	18,78	1,88	1,62

Cuadro 6.3: Errores de la posición estimada con respecto a la posición real durante el recorrido de la figura 6.7

del estado x_t, $p(x_t)$, será función de la posición en la que nos encontremos en el estado anterior, x_{t-1}. Puede observarse en la figura 6.8 que el arco de la derecha se perfila, pero debido a que la mejor posición es un máximo local y no global, la posición final estimada sitúa al robot en la habitación de la izquierda en lugar de en la derecha.

El error medio para cada componente de la *pose* cometido a lo largo del desplazamiento del robot de la figura 6.7 se muestra gráficamente en la figura , mientras que la tabla 6.3 expone los valores estadísticos calculados. En estas gráficas puede observerse un pico acusado en el error que da lugar a errores máximos de gran valor, fruto de la existencia de ese máximo local no detectado.

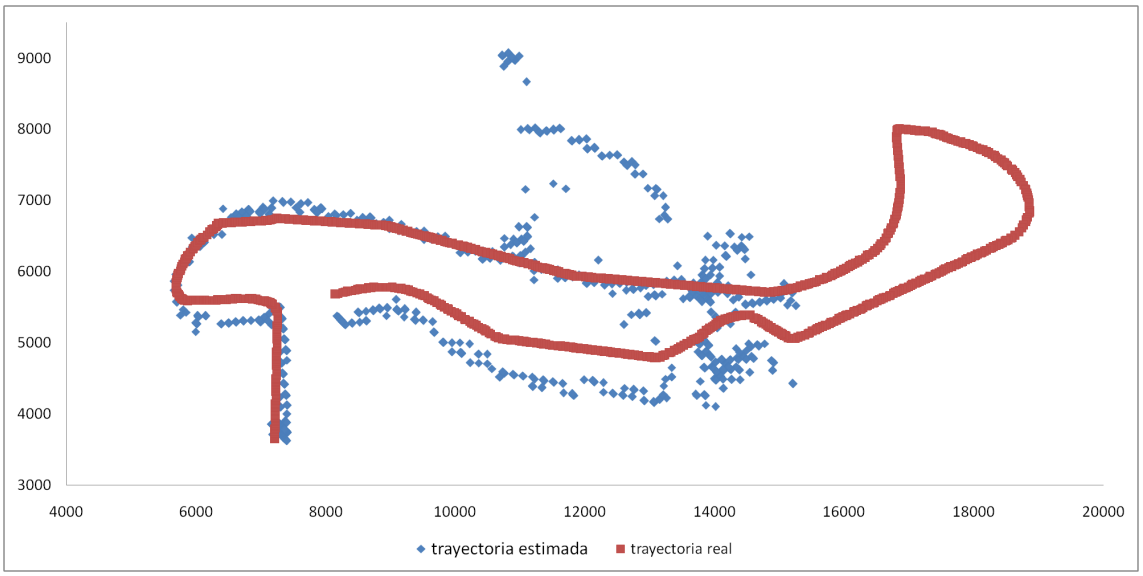


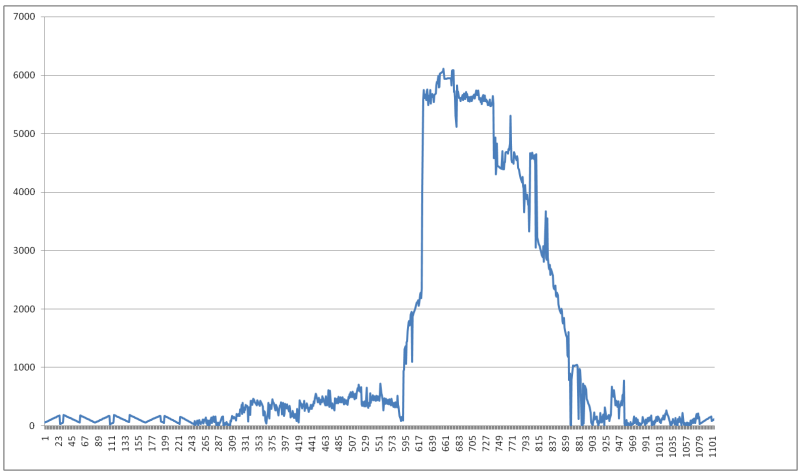
Figura 6.8: Recorrido real y recorrido estimado

Variable	error max	error medio	desviación
Eje X (mm)	729,60	287,58	169,73
Eje Z (mm)	436,28	135,93	82,56
Ángulo de rotación (°)	5,61	1,76	1,26

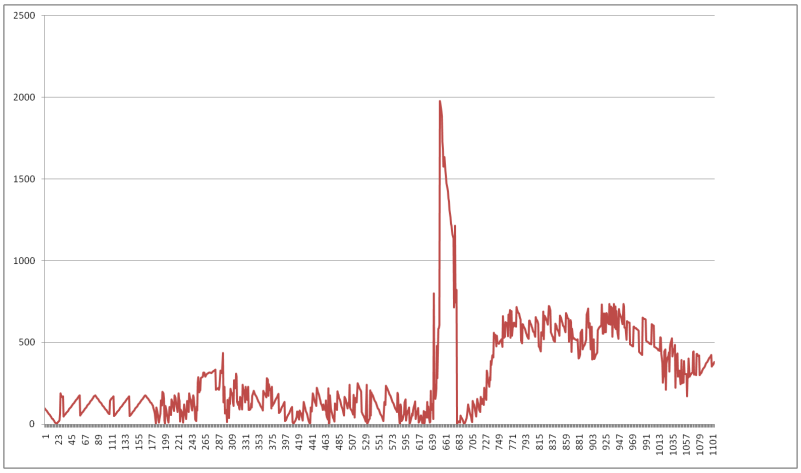
Cuadro 6.4: Errores de la posición estimada con respecto a la posición real durante las primeras posiciones del recorrido de la figura 6.7

En caso de que no existan máximos locales, como es en el caso de las primeras posiciones del recorrido de ejemplo, los valores de la tabla 6.3 se mejoran notablemente, sobre todo en las coordenadas X y Z, como recoje la tabla 6.4.

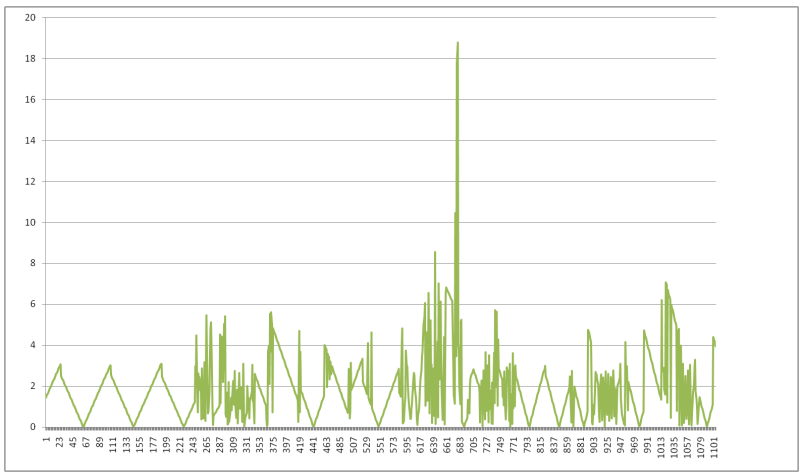
Estos resultados llevan a la conclusión de que es posible estimar de forma adecuada la *pose* del robot dentro de un entorno cerrado utilizando un sistema de localización probabilístico basado en celdas ocupación, haciendo uso de un sensor láser y empleando medidas de similitud entre imágenes siempre y cuando dispongamos de una cantidad mínima de información y la presencia de simetría en el entorno no de lugar a muchos máximos locales.



(a) Error absoluto en la coordenada X (mm)



(b) Error absoluto en la coordenada Z (mm)



(c) Error absoluto en el ángulo de rotación (°)

Figura 6.9: Error medio a lo largo del tiempo para el recorrido de la figura 6.7

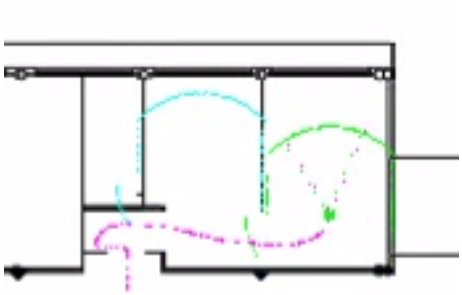


Figura 6.10: Error en la localización debido a la existencia de máximos locales

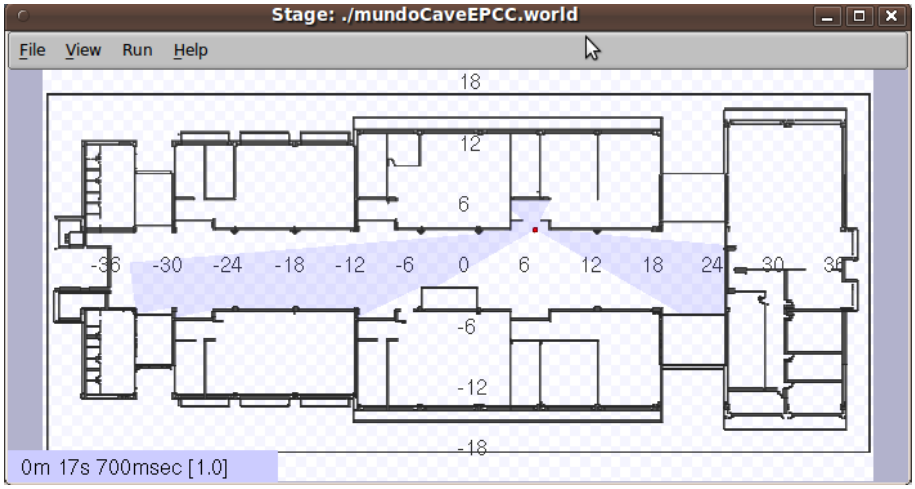


Figura 6.11: Simulación de láser de 40 metros en Player/Stage

El rango del láser limita la localización a entornos con obstáculos lo suficientemente cerca para poder detectarlos en cantidad suficiente. Para poder hacer un recorrido por el pasillo del pabellón de informática, por ejemplo, un láser de 4 metros no es suficiente puesto que no puede detectar suficientes obstáculos a uno y otro lado del pasillo como para poder estimar la posición de forma correcta. Si el láser empleado hubiera tenido un rango de 40 metros, se tendría mucha más información para poder llevar a cabo la localización, como muestra la simulación del Player/Stage (figura 6.11).

Capítulo 7

Conclusiones

El trabajo desarrollado en el presente Trabajo Fin de Máster ha tratado de resolver la problemática de la localización de un robot móvil autónomo en un entorno cerrado desde la perspectiva de la robótica probabilística con el fin de poder identificar, a partir de los datos de un sensor láser, la posición y orientación del robot.

El entorno se ha modelado mediante mapas de ocupación, lo cual ha permitido poder aplicar técnicas de registrado de imágenes para detectar el grado de similitud entre la medida del entorno obtenida por el sensor láser y el mapa del entorno.

En la localización global, el robot ha de ser capaz de determinar su posición con respecto a su entorno sin conocer su posición anterior. En un entorno altamente simétrico, como puede ser la planta de un edificio, este problema se complica puesto que es probable que existan varias localizaciones en las que la medida del sensor y el mapa sean parecidos. Para solventar este problema, se ha tomado la solución de compromiso de reducir el área de búsqueda inicial a una región de interés (ROI) centrada en la posición dada por la odometría.

Gracias a la utilización de un diseño basado en componentes, será fácil ampliarlo o acoplarlo a otros componentes existentes para que funcionen conjuntamente, extendiendo así las capacidades del robot.

Las pruebas realizadas demuestran que el robot puede localizarse correctamente dentro de un entorno cerrado, definido mediante mapas de celdas de ocupación y basándose en medidas de similitud.

7.1. Líneas de trabajo futuras

Algunas de las líneas de trabajo futuras para ampliar y mejorar el funcionamiento del componente sería:

- Dotar al componete de mayor flexibilidad mediante la mejora de la interfaz gráfica de usuario, de tal modo que se pueda cargar el mapa global, cambiar la resolución angular...

- Incluir la trayectoria estimada en la decisión de la estimación actual con el fin de escoger la mejor *pose* aunque esta se de en un máximo local y no global.

- Incluir la idea de filtros de partículas en la solución de la localización. Los filtros de partículas permiten estimar el estado de un sistema que cambia a lo largo del tiempo. En nuestro caso, tendríamos una población finita de robots (las partículas) en diferentes estados, xt, y una medida del entorno adquirida con el sensor real, zt. La idea del filtro de partículas radica en asignar una probabilidad a cada uno de esos robots sintéticos de acuerdo con el grado de similitud que tenga la medida sintética con la medida real, para, en la siguiente iteración, replicar con mayor probabilidad aquellos estados con valor más alto. Este método suele converger a la solución real transcurridas unas iteraciones. Esta aproximación evitaría el problema de la localización inicial y haría que no fuera necesario estimar un area inicial de dimensiones inferiores al mapa global completo.

- Optimizar el algoritmo de búsqueda restringiendo aún más los ángulos en los que se realiza la exploración, intentando obtener al mismo tiempo, la mayor granularidad posible.

- Dar el salto de simulación a aplicación real para estimar la localización de un robot real en un entorno real. Para ello será necesario considerar los obstáculos que no están presentes en un mapa de la planta de un edificio pero que son detectables por el sensor láser, como pueden ser bancos, maceteros o la propia presencia de personas.

Bibliografía

[1] Pilar Bachiller. Robots móviles: Arquitecturas software. Apuntes de la asignatura Sistemas informáticos y telemáticos avanzados.

[2] Norman W. Edmund. *The General Pattern of the Scientific Method (SM-14). Second Student Edition.* Norman W. Edmund (Retired Founder of Edmund Scientific Co.) 407 Northeast Third Ave., Ft. Lauderdale, FL 33301-3233, 1994.

[3] Norman W. Edmund. The scientific method today. http://www.scientificmethod.com/.

[4] Home page: Robex arena.

[5] Souceforge.net: robocomp.

[6] The player project. http://playerstage.sourceforge.net/.

[7] Rudy Negenborn. *Robot Localization and Kalman Filters.* PhD thesis, Utrecht University, 2003.

[8] Burgard Wolfram Fox Dieter, Thrun Sebastian and Dellaert Frank. *Sequential Monte Carlo Methods in Practice.*, chapter Particle filters for mobile robot localization. Springer Verlag, New York, 2000.

[9] José Manuel Pérez Lorenzo. *Localización de Robots Móviles en Interiores con Mapas de Segmentos y Regiones Visuales Curvilíneas.* PhD thesis, Universidad de Málaga - Escuela Técnica Superior de Ingeniería de Telecomunicación, 2010.

[10] Durrant-Whyte H Leonard J. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376–82, 1991.

[11] Amit Singhal. Issues in autonomous mobile robot navigation. Technical report, University of Rochester, 1997.

[12] Marcial Martín Román. Utm-30lx and urg 04lx laser range sensors characterization for autonomous robot navigation". Master's thesis, Universidad de Extremadura - Escuela Politécnica, 2010.

[13] Wernersson Ake Fosberg Johan, Larsson Ulf. Mobile robot navigation using the range-weighted hough transform. *IEEE Robotics and Automation Magazine*, 2(1):18 – 26, 1995.

[14] Vassilis Varveropoulos. Robot localization and map construction using sonar data. *The Rossum Project – http://rossum.sourceforge.net.*

[15] Dieter Fox Thrun Sebastian, Burgard Wolfram. *Probabilistic robotics.* The MIT Press, 2005.

[16] Barbara Zitova and Jan Flusser. Image registration methods: a survey. *Image and Vision Computing*, 21:977 – 1000, 2003.

[17] Xiaoming Peng, Mingyue Ding, Chengping Zhou, and Qian Ma. A practical two-step image registration method for two-dimensional images. *Information Fusion*, 5:283–298, 2004.

[18] Lisa Gottesfeld Brown. A survey of image registration techniques. *ACM Comput. Surv.*, 24(4):325–376, 1992.

[19] Rafael C. Gonzalez and Richard Eugene Woods. *Digital image processing.* Prentice Hall, 2002.