Automatización de los recorridos del robot en un sistema de exploración basado en comportamientos

Proyecto de Fin de Carrera

Facultad de Informática de la Universidad del País Vasco

9 de mayo de 2013

Eneko Laskurain

Supervisores
Elena Lazkano y Ekaitz Jauregi

Agradecimientos

Por una lado, me gustaría agradecer a Ekaitz Jauregi la ayuda que me ha brindado en todos esos momentos de la fase de desarrollo en los que no entendía por qué las cosas funcionaban como funcionaban y, por otro lado, a Elena Lazkano que en la última fase me ha puesto las pilas. Gracias a los dos por el tiempo invertido conmigo.

También tengo que agradecer a toda esa gente ajena al proyecto, como familia y amigos, que han facilitado que siguiese adelante en los malos momentos que he tenido a lo largo de este tiempo.

Resumen

Este proyecto de fin de carrera se basa en el trabajo realizado en la tesis *Advances towards behaviour-based indoor robotic exploration*, donde se presenta un sistema de navegación robótica basada en comportamientos.

En el desarrollo técnico se muestra la automatización de los recorridos del robot para facilitar la posterior evaluación de los resultados simulados. También, con el mismo objetivo, se muestran los cambios referentes a la carga de un mapa topológico en la memoria del robot. Pero primero se muestran los fundamentos de los robots móviles junto con los diferentes paradigmas de control existentes, enfatizando el control basado en comportamientos. También se aborda brevemente el sistema utilizado para la estimación de la localización. Todo ello para concluir con la muestra de resultados y las valoraciones de las pruebas simuladas en el simulador Stage.

Laburpena

Advances towards behaviour-based indoor robotic exploration tesian egindako lanean oinarritzen da karrera amaierako proiektu hau, non roboten nabigaziorako portaeratan oinarritutako sistema bat aurkezten den.

Garapen teknikoan, simulazioan lortutako emaitzen ebaluazioa errazteko, robotaren ibilbideen automatizazioa erakusten da. Helburu berarekin, mapa topologiko bat memorian kargatzearen inguruko aldaketak adierazten dira. Baina lehenik, robotika mugikorraren oinarriak azaltzen dira, gehiegi sakondu gabe, eta kontrol robotikoaren eredu ezberdinak ikusten dira, portaeretan oinarritutako kontrol sistemari garrantzia emanez. Kasu konkretu honetarako erabilitako lokalizazio sistemaren oinarriak ere aipatzen dira. Guzti hau dena, Stage simulagailuan lortutako emaitza simulatuak erakusteko eta hauen balorazioa adierazteko.

Índice general

Índice de figuras

Índice de cuadros

Parte I

INTRODUCCIÓN

Capítulo 1

INTRODUCCIÓN

Contenido

El ser humano siempre ha mostrado fascinación por cuerpos y mentes artificiales. La creación de estas máquinas o artefactos capaces de imitarnos o de realizar tareas de forma autónoma ha estado ligada al conocimiento tecnocientífico del momento. A estas maquinas, los griegos, las denominaban *automatos* de donde deriva la palabra *autómata*. Se puede contemplar a largo de la historia los diversos intentos de alcanzar estos sueños.

Por ejemplo, Jacques Vaucanson (1709-1782), construyó varios muñecos animados, como un flautista capaz de tocar varias melodías y un pato (1739) capaz de graznar, beber, comer, digerir y evacuar la comida. El científico eslovaco Wolfgang von Kempelen construyó un autómata conocido como *el turco* (1769) que jugaba al ajedrez. Años más tarde se reveló su funcionamiento, donde la inteligencia del autómata provenía de una persona escondida en el mismo artefacto. En 1805, Henri Maillardert construyó una muñeca mecánica que era capaz de hacer dibujos.

Todos ellos comparten una característica, son simples artefactos dotados de un sistema mecánico capaces de realizar una tarea concreta. Simples autómatas. A diferencia de éstos, en 1948 y 1949 William Grey Walter construyó sus primeros robots autónomos electrónicos. Elmer y Elsie, también conocidas como tortugas debido a su apariencia, disponían de la capacidad de dirigirse a la luz, de manera que podían encontrar el camino a la base de carga cuando se encontrasen con las baterías prácticamente descargadas. Este último ejemplo muestra el salto de autómata a robot.

Vivimos en un mundo donde la tecnología avanza cada vez más deprisa, introduciendo nuevos conceptos y términos en nuestra vida diaria. Centrando el interés en el ámbito de la robótica, el cual lleva varias décadas presente en numerosos entornos, el concepto o término más sonado es *robot*. La noción popular de robot hace referencia a un dispositivo humanoide con cierto grado de inteligencia, que substituye a las personas en la realización de diferentes tareas. Esta visión está en gran medida influenciada por numerosos relatos de la literatura de ciencia ficción y obras cinematográficas.

El termino *robot* se utilizó por primera vez en el año 1921, cuando un dramaturgo checo llamado Karel Čapek estreno en el teatro nacional de Praga su obra titulada *Rossum's Universal Robots* (R.U.R.). Su origen es la palabra eslava *robota*, que significa literalmente trabajo o labor y figuradamente trabajo realizado de manera forzada. A partir de entonces aparecieron nuevos derivados como la palabra *robótica*, creada por Isaac Asimov para describir el campo de estudio que concentra la mecánica, el control automático, la electrónica, la

informática, la física y las matemáticas como ciencias básicas.

Isaac Asimov contribuyó con varias narraciones relativas a robots desde 1939. La imagen de robot que aparece en su obra es el de una máquina bien diseñada y con una seguridad garantizada que actúa de acuerdo con tres principios. Estos principios fueron denominados por Asimov como las Tres Leyes de la Robótica, y son:

1. Un robot no puede actuar contra un ser humano o, mediante la inacción, no puede hacer que un ser humano sufra daños.

2. Un robot debe obedecer las ordenes dadas por los seres humanos, salvo que estén en conflicto con la primera ley.

3. Un robot debe proteger su propia existencia, a no ser que esté en conflicto con las dos primeras leyes.

Consecuentemente todos los robots de Asimov son fieles sirvientes del ser humano a diferencia de los robots que aparecen en la obra de Čapek, en un principio sirvientes al ser humano que después se revelan a éstos.

Los robots actuales no tienen mucho que ver con los humanoides que aparecen en la literatura de ciencia ficción. Además, no existe una definición de robot que satisfaga a todos, e incluso muchas personas tienen su propia definición. Expertos en la materia como A. Mackworth define que un robot es *una máquina que puede sentir, actuar y reaccionar en un entorno y, posiblemente, implica un razonamiento para llevar a cabo estas acciones de manera autónoma*. Para R. Brooks *un robot es algo que tiene algún efecto físico en el entorno, basándose en cómo percibe el entorno y cómo el entorno cambia a su alrededor*.

Distintas organizaciones y asociaciones internacionales ofrecen definiciones diferentes aunque, obviamente, próximas entre sí. Por ejemplo, la Organización Internacional para la Estandarización lo define como *un manipulador automáticamente controlado, reprogramable, multiuso, programable en tres o más ejes, que pueden estar fijos en un lugar o movilizarse para ser usado en aplicaciones de automatización industrial*. El Robot Institute of America, después denominado como Robot Industries Association (RIA) lo define como *manipulador funcional reprogramable, capaz de mover material, piezas, herramientas o dispositivos especializados mediante movimientos variables programados, con el fin de realizar diversas tareas*.

Cada experto, organización o asociación tiene su particular definición pero la respuesta más común a la pregunta de ¿Qué es un robot? probablemente sea la que dio J. Engelberger, pionero de la robótica industrial, cuando respondió: *no puedo definir un robot, pero sé reconocer uno cuando lo veo.*

1.1. Evolución de la robótica

La robótica es la herramienta perfecta para saciar el afán del ser humano de crear artefactos semejantes al ser humano con el objetivo de ahorrar trabajo. Actualmente, la robótica se define como la ciencia y tecnología de los robots. Se ocupa del diseño, manufactura y aplicaciones de éstos y combina diversas disciplinas como la mecánica, electrónica, tratamiento de datos, inteligencia artificial e ingeniería de control.

La robótica no lleva muchos años en activo si se compara con otras ramas de la tecnología, pero a pesar de todo su evolución no ha sido insignificante. A la hora de explicar la evolución de la robótica se tiende a hablar de generaciones. Aunque más que generaciones también se podrían denominar pasos o avances en tipos de robots. A continuación se presenta una aproximación de las diferentes generaciones.

Primera generación

La primera generación de robots, la cual supone el inicio de la robótica, se conoce como *manipuladores*. Éstos sistemas mecánicos multifuncionales de un sencillo sistema de control, permiten gobernar el movimiento de sus elementos de diferentes maneras. Por un lado está el modo *manual*, cuando el operario controla directamente la tarea del manipulador. Por otro lado, el modo de *secuencia fija* cuando se repite, de forma invariable, el proceso de trabajo preparado previamente. Por último está la *secuencia variable*, cuando se pueden alterar algunas características de los ciclos de trabajo. Muchas aplicaciones requieren operaciones básicas que pueden ser realizadas óptimamente mediante manipuladores, sobre todo cuando las tareas son sencillas y repetitivas, de ahí el uso de estos dispositivos.

Segunda generación

La segunda generación o los *robots de aprendizaje*, son una mejora de los

robots de primera generación. Son manipuladores que se limitan a repetir una secuencia de movimientos, previamente ejecutada por un operador humano. Dicho movimiento se recoge haciendo uso de un controlador manual o un dispositivo auxiliar. En este tipo de robots, en la fase de enseñanza, el operario se vale de una pistola de programación con diversos pulsadores, teclas, joystics, o bien utiliza un maniquí. A veces incluso el operario desplaza directamente la mano del robot, lo que se denomina *programación gestual*. Utiliza una estructura de control de ciclo abierto, pero en lugar de interruptores y botones mecánicos utiliza una secuencia numérica de control de movimientos almacenados. Estos robots son los más empleados en el ámbito industrial.

Tercera generación

Los robots de tercera generación se consideran *robots con control sensorizado* o *robots controlados por computador*. Como el propio nombre indica, son manipuladores o sistemas mecánicos multifuncionales controlados por un computador que habitualmente suele ser un microordenador/microprocesador. En este tipo de robots, el operario no necesita mover físicamente el elemento de la maquina cuando la prepara para realizar un trabajo. El control por computador dispone de un lenguaje especifico compuesto por varias instrucciones adaptadas al robot. Con estas instrucciones se crea un programa de aplicación utilizando solo el terminal del ordenador y no el brazo. A esta programación se le denomina *textual* y se crea sin la intervención del manipulador. Dado que se utilizan computadoras para la estrategia de control y se tiene conocimiento del entorno local gracias a sensores, es posible modificar dichas estrategias. En estos robots la estructura de control es de ciclo cerrado.

Cuarta generación

La denominada generación de *robots inteligentes* son similares a las del grupo anterior, pero además, son capaces de relacionarse con el entorno a través de sensores y tomar decisiones en tiempo real. Presentan más extensiones sensoriales y mejores. Gracias al conocimiento del entorno y el procesamiento dirigido por expectativas mejoran el desempeño de las tareas. El control de los sistemas no es tan simple como en las anteriores generaciones ya que en esta etapa la variedad es más amplia. Se puede decir que actualmente se trabaja en esta generación con el fin de avanzar a próximas generaciones y de dar un salto cada vez mayor que los anteriores.

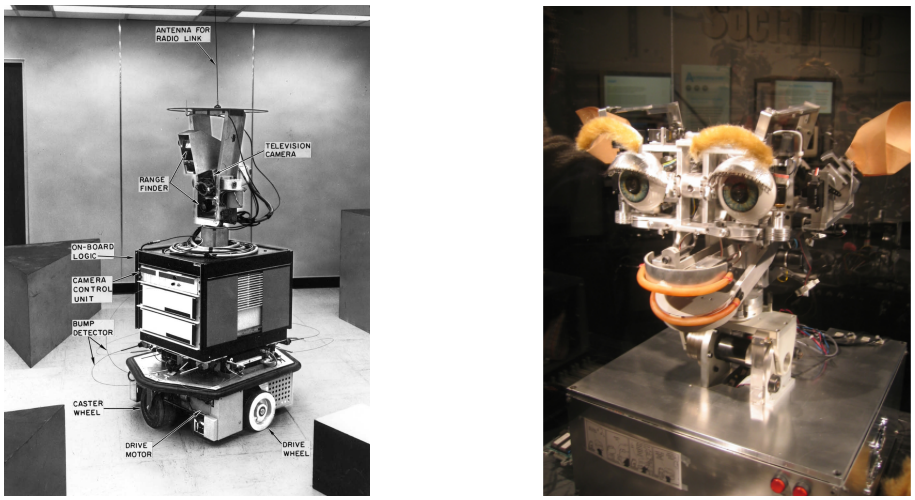
Próximas generaciones

En cuanto las próximas generaciones robóticas, probablemente el desarrollo no sea tan lineal como ha sido hasta ahora. Las diferentes lineas de investigación abiertas en paralelo favorecerán la aparición de nuevas generaciones prácticamente al mismo tiempo. Lo cierto es que todas ella comparten el mismo objetivo, crear robots inteligentes y autónomos, capaces de estar situados en un entorno, adoptar diferentes comportamientos, tomar decisiones o razonar. Resumiendo, se busca que éstas entidades autónomas actúen en semejanza al ser humano. Los campos de estudio o investigación como la robótica situada, la robótica basada en el comportamiento o la robótica evolutiva ayudarán a alcanzar el tan ansiado objetivo.

1.2. Tipos de robots

La clasificación de robots es una tarea difícil ya que se deben tener en cuenta distintos factores, como por ejemplo el ámbito de trabajo o la morfología. Dependiendo del criterio de clasificación pueden surgir más o menos grupos. En la sección 1.1 se han descrito diferentes generaciones de robots, las cuales pueden valer para clasificar robots. Pero dado que la clasificación se centra en los sistemas de control que se han ido desarrollando no es adecuada para percibir el alcance de la robótica. A continuación se muestran algunos tipos de robots clasificados según el ámbito de trabajo al que pertenecen:

Figura 1.1: Robot manipulador de la empresa KUKA Robots.



a) b)

Figura 1.2: a) Robot Shakey, SRI. b) Robot Kismet, MIT

- **Robots industriales**. Éstos son robots que funcionan en el ámbito de la industria. Mayormente son manipuladores o brazos robóticos parte de una cadena de montaje. El primero de este tipo fue construido por Harold Roselund en 1938. Su tarea era pintar con spray. Debido a su complicada programación no tuvo gran éxito. En 1946, el considerado fundador del primer robot industrial, George Devol inventó un aparato mecánico que permitía repetir una secuencia de movimientos en una máquina herramienta. A partir de entonces aparecieron patentes importantes y en 1961 se instaló el primer *Unimate*, de George Devol, en una de sus plantas General Motors. En la actualidad se ha generalizado el uso de este tipo de robots, concretamente en la industria automovilística [MI02]. En la figura 1.1 se muestra un robot manipulador.

- **Robots de investigación**. En este grupo se pueden englobar una gran cantidad de robots de distintos tipos como pueden ser brazos robóticos o robots sociales, pero al hablar de robots de investigación se refiere a los que no han sido comercializados en el mercado y ayudan a diferentes instituciones a avanzar en diferentes tecnologías. En 1968 apareció Shakey (ver figura 1.2.a) desarrollado por el SRI (Standford Research Institute) [Nil84]. Este robot fue el primer robot móvil de propósito general capaz de razonar sobre sus propias acciones. En cuanto al ámbito de la interacción persona computador, se debe mencionar al robot Kismet (figura 1.2.b) creado a finales de los 90 por el Instituto Tecnológico de Massachu-

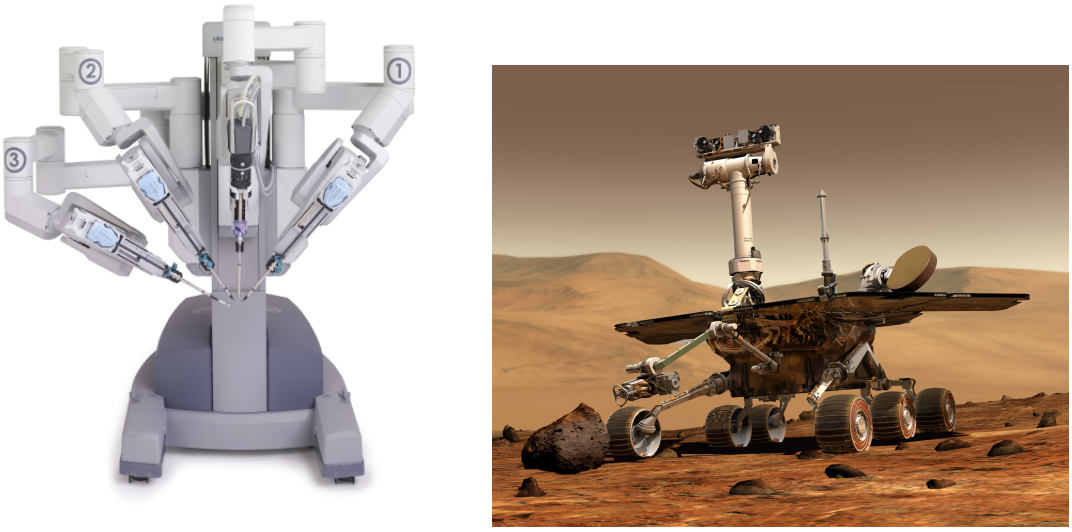
a) b) c)

Figura 1.3: a) Robot Aibo, Sony Corporation. b) Robot iCat, Philips Electronics. c) Robot Wakamaru, Mitsubishi Heavy Industries.

setts. Kismet es capaz simular emociones a través de diversas expresiones faciales, vocalizaciones o movimiento. Su objetivo ha sido el estudio de la interacción con seres humanos [BS99]. Cada uno de estos robots es pionero en su respectivo campo.

- **Robots sociales**. Un robot social es un robot autónomo capaz de interactuar y comunicar con usuarios, entenderlos e incluso relacionarse con seres humanos, de un modo personal y en términos sociales. Las personas, en cambio, deben ser capaces de entenderlo en los mismos términos [Bre04]. En los últimos años ha habido un significativo esfuerzo en desarrollar robots que actúen con naturalidad. Estos robots socialmente interactivos están diseñados para enganchar de forma afectiva y también para entretener. Además, deben funcionar en un entorno cercano a las personas. Para las empresas este es un nuevo modelo de negocio en auge. En la figura 1.3 se muestra una variedad de robots sociales desarrollados por diferentes empresas.

- **Robots aplicados a la medicina**. Dentro de este grupo se pueden englobar varios tipos de robot como robots quirúrgicos o robots para la rehabilitación. El uso de estos robots tiene como objetivo mejorar los procedimientos quirúrgicos y los procedimientos de rehabilitación. Básicamente son una herramienta más, pero inteligente, ya que ayudan a compensar las deficiencias y limitaciones que pueda tener el cirujano para realizar ciertas actuaciones. De ese modo posibilitan la implantación de algunas técnicas de cirugía mínimamente invasivas gracias a la utilización de soportes de ayuda robotizados, consiguiendo minimizar la herida,



a) b)

Figura 1.4: a) Robot quirúrgico DaVinci desarrollado por Intuitive Surgical. b) Imagen generada por ordenador de un *rover* de la NASA.

reducir el tiempo de intervención y el de posterior recuperación. Uno de los robots mas conocidos actualmente es el robot DaVinci (figura 1.4.a) desarrollado por Intuitive Surgical.

- **Robots de exploración**. Se consideran exploradores los robots que son capaces de moverse de forma autónoma a través de un medio (tierra, mar o aire) en un entorno desconocido. Muchos comparten las mismas características pero con alguna diferencia ya que las condiciones ambientales no son las mismas. Básicamente son robots móviles (ver Capítulo 3) con la particularidad de que tienen que recoger información de un entorno desconocido de manera autónoma. Un robot de exploración de entornos cerrados (edificios), un robot de exploración submarina o un robot de exploración espacial (figura 1.4.b) son algunos ejemplos de robots de exploración.

- **Otros**. La variedad de los ámbitos o aplicaciones a las que se dedican los robots es muy amplia y extensa. Otros ejemplos en tipos de robots pueden ser: robots militares, robots domésticos como el conocido Roomba de iRobot, robots educativos como el Lego Mindstorm o robots de seguridad.

Día a día la lista se va ampliando ya que los robots desarrollan unas habilidades y capacidades que les permiten estar más presente en la vida de las personas.

Capítulo 2

DOCUMENTO DE OBJETIVOS DEL PROYECTO

2.1. Contexto del proyecto

El desarrollo técnico de este PFC se basa en una parte del trabajo *Advances towards behaviour-based indoor robotic exploration* [JI11], concretamente en la localización y mapeado de un sistema basado en comportamientos, siendo el punto de partida la documentación y todas las líneas de código de dicha tesis doctoral. Este proyecto se realiza durante un curso en Alemania como estudiante ERASMUS en la universidad Albert Lüdwig en Freiburg im Breisgau. Se mantiene el contacto mediante correo electrónico y reuniones presenciales cuando sea posible.

2.2. Objetivos

El desarrollo del PFC tiene varios objetivos tanto personales como académicos. Todo comienza con la iniciación en el ámbito de la robótica móvil, seguido de la aplicación de los conocimientos obtenidos durante los años de carrera, además de los obtenidos de forma autodidacta a lo largo del PFC. Los conocimientos adquiridos durante el curso en el extranjero ayudarán a entender diferentes conceptos.

En cuanto al desarrollo técnico, se pretende reforzar los resultados del sistema localización de robot móviles propuesto en *Advances towards behaviour-based indoor robotic exploration* [JI11] como un sistema basado en comportamientos para la navegación robótica. Para ello, se realizarán varios experimentos, se automatizarán las estrategias de mapeado y localización para su posterior evaluación. Además, se adaptarán los procedimientos de carga de mapas para poder valorar conjuntamente los resultados con diferentes tipos de mediciones.

2.3. Fases del proyecto

El inicio de un PFC está marcado por la identificación de las fases que éste tendrá. Como muestra la experiencia, el desarrollo de las fases no siempre es de manera lineal, no hay transición entre fases paso a paso y muchas veces se tiene que conciliar con varias fases en su desarrollo e incluso retomar fases previamente finalizadas.

A continuación se muestran las diferentes fases identificadas al inicio de este PFC:

- Puesta a punto del sistema.
- Entendimiento del trabajo base del cual parte el proyecto.
- Aprendizaje del funcionamiento del sistema.
- Desarrollo y pruebas en el sistema.
- Edición de memoria.
- Preparación de presentación.

En esta lista no se mencionan las diferentes reuniones llevadas a cabo ya que su realización ha dependido mucho de la disponibilidad del autor.

2.3.1. Cronología

En los siguientes diagramas de Gantt se pueden apreciar las diferentes fases del proyecto a lo largo de la ejecución del mismo. En la figura 2.1 se muestra una primera estimación realizadas al inicio del proyecto.

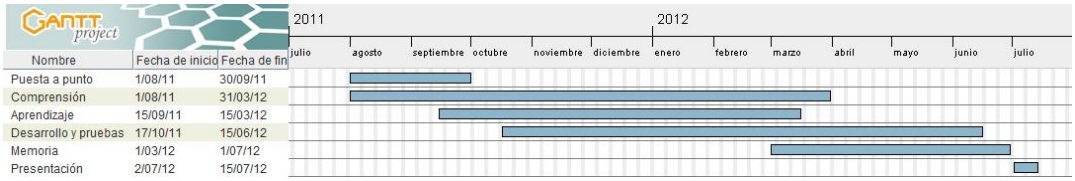


Figura 2.1: Primera estimación de las fases del proyecto.

Como se ha mencionado anteriormente, este proyecto de fin de carrera se realiza conjuntamente con un año como ERASMUS y debido a diferencias en el calendario tanto de clases, exámenes y vacaciones respecto a la universidad de origen, no ha sido posible ceñirse a la primera estimación. Por ello, al ver que no se podía alcanzar las fechas fijadas en un principio se realiza una segunda estimación (figura 2.2).

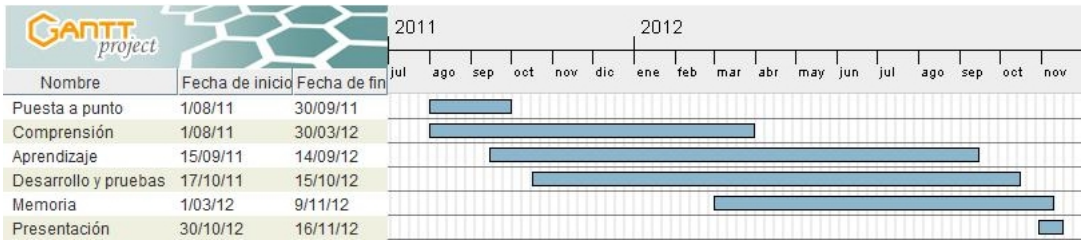


Figura 2.2: Segunda estimación de las fases del proyecto.

A primera vista se aprecia que la duración de las fases es muy larga pero dado que en este tipo de diagramas se marcan las fechas de inicio y fin, no se pueden representar los diversos parones que habrá a lo largo del desarrollo del proyecto.

2.4. Problemas

En líneas generales, los problemas encontrados a lo largo de la realización de este proyecto se pueden clasificar en varios tipos:

- Problemas relacionados con la implantación del sistema (Player/Stage). No ha sido fácil obtener el funcionamiento correcto del sistema. Al depender de muchas librerías y un montón de ficheros, se han encontrado diversos problemas que se han ido solucionando poco a poco.
- Problemas con la naturaleza del sistema basado en comportamientos. Muchos errores de los resultados de las pruebas puede que no aparezcan de manera regular ya que dependen de demasiadas variables. Por esta razón ha sido bastante común creer en el funcionamiento correcto y de repente topar con errores que no han sido fáciles de encontrar.
- Problemas con el material personal. Como en todo proyecto o trabajo llega un momento el que por diferentes razones se pierden los documentos y se ha echado de menos tener más actualizadas las copias de seguridad. También se han tenido problemas con la refrigeración del ordenador personal, que han dado lugar a fallos en pruebas largas con el simulador.
- Problemas relacionadas con prioridades. Debido a el curso que se ha realizado en el extranjero y las dificultades que se han encontrado en las diferentes asignaturas que se han tenido que aprobar, no se ha podido prestar máxima atención a este proyecto. Los diferentes parones que ha tenido han llevado a alargar más de lo estimado su finalización.

Parte II

ROBÓTICA MÓVIL BASADA EN COMPORTAMIENTOS

Capítulo 3

FUNDAMENTOS DE LA ROBÓTICA MÓVIL

Contenido

Los características que definen un robot móvil se pueden resumir en dos aspectos principales: locomoción y percepción. Como su propio nombre indica el robot es móvil, por ello, debe estar dotado de un sistema de locomoción. Este aspecto juega un papel importante ya que el sistema de locomoción puede diferir del entorno en el que se utiliza. Además, las opciones pueden acarrear dificultades a la hora de la implementación. El cuanto a la percepción, dado que el robot se mueve en un entorno debe ser capaz de adquirir conocimiento de él. Es una tarea importante en un robot autónomo puesto que las decisiones que debe tomar a la hora de moverse dependen de ello.

3.1. Locomoción

Un robot móvil necesita diversos mecanismos que le permitan moverse sin problemas ni límites en ciertos entornos. Para ello, existe una gran variedad de posibilidades. La mayoría de los mecanismos tienen inspiraciones biológicas. Sin embargo, la rueda es elemento que mayor rendimiento obtiene en superficies planas.

Los sistemas biológicos pueden ofrecer una locomoción exitosa en una gran variedad de entornos complicados. Por lo tanto, es normal querer copiar dichos mecanismos de locomoción, sin embargo, son extremadamente complicados de modelar. Por una parte, es muy costoso económicamente lograr alcanzar modelos biológicos precisos. Por otra parte, es muy difícil obtener el mismo nivel de tiempo de respuesta y eficiencia de sus homólogos biológicos. Debido a estas limitaciones, los robots móviles generalmente disponen de ruedas o también a veces están dispuestos de un pequeño numero de articulaciones, puesto que es el método biológico de locomoción más sencillo. La locomoción mediante patas articuladas tiene en su contra el requerimiento de más grados de libertad y por tanto una mayor complejidad mecánica que la locomoción mediante ruedas.

La eficiencia de la locomoción mediante ruedas depende en gran parte de las cualidades ambientales, particularmente en lo llano y duro que es el suelo, mientras que la eficiencia en la locomoción mediante articulaciones depende de una gran variedad de características, como el peso de las articulaciones, el cuerpo, etc.

En las próximas líneas se describe más a fondo la locomoción mediante ruedas, dado que este trabajo se centra en robots móviles con este tipo de

mecanismo.

3.1.1. Robots móviles con ruedas

La rueda es el mecanismo de locomoción más popular en la robótica móvil y en general en los vehículos fabricados por el ser humano. La eficiencia obtenida con este medio es muy alta gracias a una ejecución mecánica relativamente simple. Además, el equilibrio no es un problema a tener en cuenta a la hora de diseñar robots con ruedas, puesto que las ruedas están casi siempre en contacto con el suelo en todo momento. De esta manera, la investigación tiende a centrarse en los problemas de tracción, estabilidad, maniobrabilidad y control. El problema del equilibrio, en cambio, está mas presente en los robots con locomoción mediante patas articuladas.

Existe una gran variedad de configuraciones posibles en cuanto a tipos de ruedas y técnicas de locomoción. Cada cual obtiene diferentes resultados acondicionados para cada tipo de aplicación.

Tipos de ruedas

La elección del tipo de rueda tiene un gran efecto sobre la cinemática general del robot. En la figura 3.1 se pueden apreciar cuatro tipos de rueda generalmente utilizadas. Todos ellos difieren ampliamente en sus características.

La rueda estándar y la rueda castor o loca tienen un eje primario de rotación y por tanto son altamente direccionales. Para obtener el movimiento en una dirección diferente, la rueda debe ser dirigida primero a través de su eje vertical. La diferencia fundamental entre estas dos ruedas es que la rueda estándar puede lograr este movimiento de dirección sin efectos secundarios, ya que el centro de rotación/direccionamiento se encuentra en la zona de contacto con el suelo, mientras que la rueda castor o loca gira alrededor del eje de desplazamiento causando pequeños desvíos de dirección.

La rueda sueca y la rueda esférica o de bola comparten un diseño que no está acondicionado por la direccionalidad que, por ejemplo, limita a la rueda estándar. La rueda sueca funciona como una rueda estándar pero además ofrece una baja resistencia en otras direcciones. Los pequeños rodillos que están unidos alrededor de la circunferencia de la rueda son pasivos y el eje primario de la rueda sirve como el conjunto alimentado activamente. La principal venta-

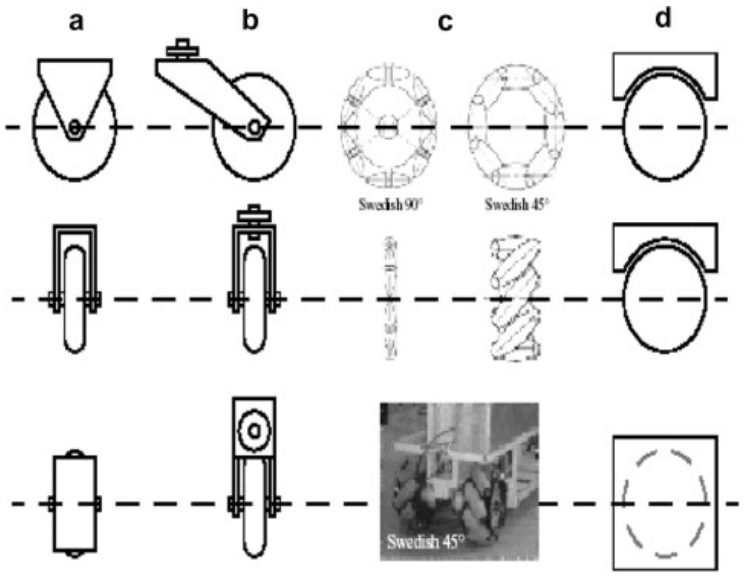


Figura 3.1: Cuatro tipos de rueda. (a) rueda estándar. (b) rueda castor o loca. (c) rueda sueca. (d) rueda de bolas

ja de este diseño es que, aunque la rotación de la rueda es alimentada sólo a lo largo del eje principal, la rueda se puede mover con muy poca fricción a lo largo de muchas trayectorias posibles, no sólo hacia delante y hacia atrás. La rueda esférica, en cambio, es una rueda verdaderamente omnidireccional, está diseñada de modo que pueda ser accionada para girar en cualquier dirección. Este mecanismo imita al diseño esférico del ratón de ordenador, comúnmente utilizado hace varios años.

Independientemente de qué rueda de las anteriores se utilice, en los robots que están diseñados para todo tipo de entornos y terrenos con mas de tres ruedas hay que tener en cuenta que para mantener la estabilidad las ruedas deben estar en contacto con el suelo. Por tanto se requiere un sistema de suspensión para ello y así obtener una flexibilidad en la propia rueda.

Configuraciones cinemáticas

Dependiendo la aplicación a la que está dirigido el robot, se opta por una u otra configuración cinemática. En la figura 3.2 se pueden apreciar las configuraciones cinemáticas más utilizadas. Generalmente la configuración más utilizada es de tracción diferencial con ruedas estándar como ruedas motrices y una o más ruedas de cualquier tipo para aportar estabilidad [OSS+07].

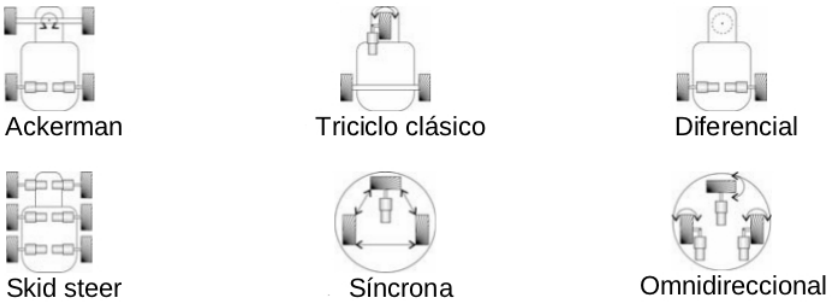


Figura 3.2: Diferentes configuraciones de locomoción.

Para facilitar el diseño, a la hora de elegir la configuración cinemática más adecuada se establecen algunas suposiciones. Por ejemplo, se consideran insignificantes las partes dinámicas del robot, se supone que la rueda tenga un único eslabón de dirección o que todos los ejes sean perpendiculares a la superficie.

Un ejemplo de configuración cinemática económica y de complejidad reducida es la que ofrece el robot Shakey [Nil84]. Cuenta con dos ruedas motrices que necesitan un par de actuadores para lograr movimiento. También incorpora dos ruedas de bola diametralmente opuestas para brindar de estabilidad al robot. De esta manera se simplifica el movimiento de giro con sólo invertir el sentido de giro de las ruedas motrices, por lo que no es necesaria la rotación, facilitando así el movimiento.

3.2. Percepción

Una de las tareas más importantes de un sistema autónomo de cualquier tipo es adquirir conocimiento sobre su entorno. Esto se hace tomando mediciones utilizando diversos sensores, para luego extraer información significativa. Un sensor es un elemento que permite al robot conocer ciertas características del entorno por el cual se desplaza. En otras palabras, los sensores son los dispositivos que permiten a un robot percibir su entorno.

Hay una amplia variedad de sensores utilizados en robots móviles. Algunos sensores se utilizan para medir valores simples como la temperatura interna de la electrónica del robot o la velocidad. Otros sensores más sofisticados pueden

ser utilizados para adquirir información sobre el entorno del robot, o incluso para medir directamente la posición de un robot.

3.2.1. Clasificación sensorial

Los sensores se pueden clasificar de varias maneras pero comúnmente se clasifican gracias a dos ejes funcionales importantes: Propioceptivos/extereoceptivos y activos/pasivos.

Sensores proprioceptivos y extereoceptivos es la clasificación más utilizada en robótica móvil. La propiocepción se refiere a la percepción del estado interno del robot, por ejemplo, velocidad del motor, carga de la batería, ángulo de algún componente, etc. La extereocepción se refiere a la percepción de aspectos externos al robot como temperatura, intensidad de la luz o distancia a objetos. Este último tipo se usa para interpretar las características ambientales significativas.

Los sensores pasivos miden la energía ambiental del entorno. Por ejemplo, los sensores de temperatura, micrófonos y cámaras CCD o CMOS. Los sensores activos emiten energía en el entorno para después medir su reacción. Debido a que los sensores activos pueden manejar más las interacciones controladas con el entorno, a menudo logran un rendimiento superior. Por el contrario, un sensor activo puede sufrir de interferencia entre la señal que controla y la que está fuera de su control. Las señales emitidas por otros robots cercanos, o los sensores similares en el mismo robot pueden influir en las mediciones resultantes.

A la hora de llevar a cabo cualquier producto robótico se deben tener en cuenta diferentes aspectos de los sensores:

- *La velocidad de operación.* La velocidad a la que el sensor obtiene nuevas mediciones. Unos sensores son más apropiados para trabajar en tiempo real y otros sólo en determinados momentos.
- *El coste.* Es la mayor barrera a la hora de fabricar un robot, ya que repercute directamente en el coste final del producto. La variedad en los precios de los sensores es muy dispar. Cuanto más calidad de medición se desee más caro es el sensor.

- *Tasa de error*. Se refiere a la mediciones no válidas de un sensor.
- *Robustez*. Es la tolerancia que tiene un sensor a cambios en el medio de funcionamiento.
- *Requerimientos computacionales*. Este aspecto es otra barrera a la hora de fabricar un robot, ya que algunos sensores que requieren gran capacidad computacional obligan a unas prestaciones mínimas en el robot. Este aspecto suele ir unido al coste y a la velocidad de operación.
- *Potencia, peso y tamaño*. Estos tres son aspectos muy importantes a tener en cuenta en el diseño, ya que influyen de diversas maneras, como por ejemplo la autonomía.

3.2.2. Sensores

En las próximas líneas se clasifican los tipos de sensores utilizados centrándose principalmente en los sensores para extraer información del entorno.

3.2.2.1. Sensores de tiempo de vuelo

El tiempo de vuelo (*time of flight*, TOF) es un método de medición activa. Se emite un pulso de energía, éste refleja en un objeto y vuelve al receptor. Los impulsos medidos proceden, por ejemplo, de una fuente de ultrasonido u óptico. Por lo tanto, los parámetros relevantes que intervienen en el cálculo son la velocidad del sonido en el aire y la velocidad de la luz. La distancia d al objeto que causa el reflejo se puede calcular basándose en la velocidad de propagación del pulso v y el tiempo de vuelo t.

Las ventajas de los sistemas TOF surgen de su naturaleza de detección activa en líneas rectas. La señal de retorno sigue esencialmente el mismo camino de vuelta a un receptor. En muchos casos la transmisión y recepción ocurre en el mismo dispositivo. El rango absoluto a un punto observado se obtiene sin un análisis complicado, la técnica no se basa en ninguna hipótesis sobre las propiedades planas o la orientación de la superficie del objetivo [BEF96].

Las fuentes potenciales de error para sensores basados en el tiempo de vuelo son:

- Las variaciones en la velocidad de propagación, en particular en el caso de los sistemas acústicos.
- Las incertidumbres en la determinación del momento exacto de llegada del pulso reflejado.
- Las imprecisiones en el circuito de temporización utilizado para medir el tiempo de ida y vuelta de vuelo.
- La interacción de la onda incidente con la superficie del objeto.

Ultrasonidos

Hoy en día, los sensores de ultrasonidos son muy utilizados en sistemas robóticos debido principalmente a la disponibilidad de sistemas de bajo coste y su facilidad de uso. Sus principales usos son, por ejemplo, la evitación de obstáculos o detección de movimiento. El principio básico de un sensor ultrasónico es transmitir un señal de ondas (ultrasonidos) y medir el tiempo necesario para que las ondas vuelvan al receptor.

Este tipo de sensores tienen varias limitaciones como el denominado *tiempo en blanco* o *blanking time*. Para no influenciar la señal de llegada al receptor con la onda que sale por el emisor, es necesario guardar un pequeño tiempo desde que se emite la señal. A esto se le llama tiempo en blanco.

Otra limitación de estos sensores es la atenuación. La onda ultrasónica que sale del emisor se va dispersando y atenuando según avanza en el medio, de manera que cuanto más tarda el eco o reflejo en llegar al receptor, más débil se espera que sea (figura 3.3). La superficie del objeto también influye en la amplitud de la señal reflejada. Esta limitación solo influye en la distancia máxima que se quiere llegar a medir.

A la hora de detectar la posición de un objeto a partir del eco que se recibe, existen varios impedimentos o limitaciones. Las *reflexiones* o *reflejos* existen cuando los objetos reflejan la onda emitida incidiendo de forma muy oblicua en un objeto, éste se refleja en una dirección diferente a la de retorno (figura 3.4.a). No se puede asegurar que dicha señal no sea resultado de diferentes direcciones del entorno. Un error parecido ocurre cuando una señal emitida

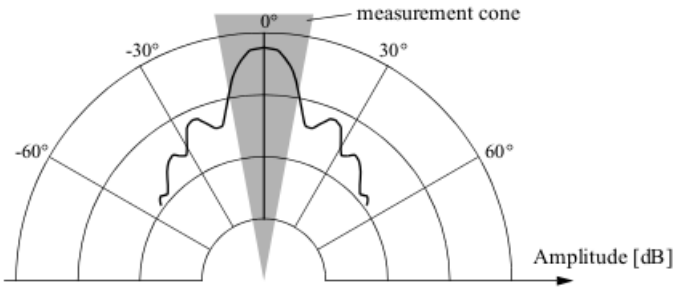


Figura 3.3: Intensidad típica de distribución de un sensor de ultrasonido.

por un sensor es recibida por otro sensor del robot. Esto se denomina como *cross talk* (figura 3.4.b). Este problema se evita dejando un tiempo entre las medidas de dos sensores de ultrasonido diferentes o garantizando el orden de disparo.

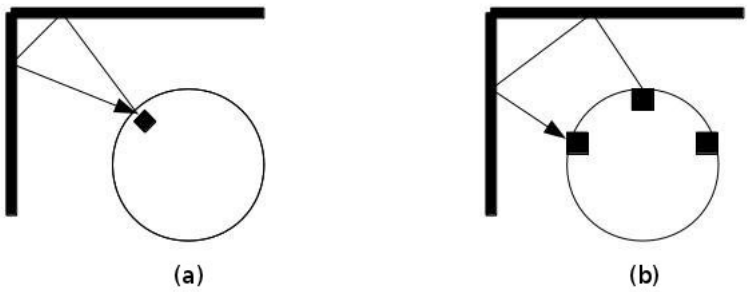


Figura 3.4: (a) Efecto de reflexión y (b) efecto *cross-talk*.

Telémetro láser

Los sensores de tipo láser o telémetros láser son dispositivos TOF utilizados para medir las distancias a objetos del entorno. Pero en lugar de enviar un pulso de sonido, estos dispositivos emiten luz a través de un proceso de amplificación óptica basada en la emisión estimulada de fotones. La energía láser es emitida en una rápida secuencia de ráfagas cortas dirigidas directamente al objeto dentro del rango [BEF96].

La base del cálculo reside en la medición del tiempo que transcurre desde que sale el rayo de luz hasta que se recibe, después de haber rebotado en un

objeto. Un método consiste en medir la frecuencia entre una onda continua de frecuencia modulada y su reflexión recibida. Otro método, aún más fácil, es medir el desplazamiento de fase de la luz reflejada [SN04].

Estos sensores se han convertido, hoy en día, en dispositivos imprescindibles en la navegación de robots para interiores y exteriores, aunque también presentan algunos defectos. A diferencia de los sensores de ultrasonidos el láser no puede detectar la presencia de materiales ópticamente transparentes como el cristal, ya que no reflejan la luz, y esto puede ser un obstáculo importante en ciertos entornos.

Otro tipo de sensores láser se basan en la triangulación como método para medir la distancia, pero están fuera de la categoría TOF.

3.2.2.2. Odometría

El sistema de odometría es el conjunto de sensores y hardware necesario para medir el desplazamiento horizontal de las ruedas de accionamiento, además del cambio en la orientación del robot. Es el método más utilizado para el posicionamiento del robot móvil, dado que proporciona una buena precisión a corto plazo y es barato. La odometría se basa en la suposición de que las revoluciones de las ruedas se pueden traducir en desplazamiento lineal con respecto al suelo. Sin embargo, en caso de deslizamiento de las ruedas o algunas otras causas, las revoluciones de la rueda no se pueden traducir proporcionalmente en movimiento lineal. Los errores resultantes se pueden clasificar en dos grupos: errores sistemáticos y errores no sistemáticos [BEF96].

Los errores sistemáticos son los que resultan de imperfecciones cinemáticas del robot. Es decir, valores inexactos de los parámetros del robot como el diámetro de la rueda o la distancia entre ejes. Estos errores se pueden medir y calibrar. Los errores no sistemáticos, por otro lado, son aquellos que resultan de la interacción del suelo con las ruedas. Por ejemplo, deslizamiento de las ruedas, golpes o grietas. Estos efectos se producen al azar y, por tanto, no se pueden corregir.

Una manera de medir la distancia recorrida por el robot es mediante un codificador. El codificador es un transductor rotativo que transforma un movimiento angular en una serie de impulsos digitales. Estos impulsos generados pueden ser utilizados para controlar los desplazamientos de tipo angular o li-

neal. Se utilizan para medir el estado interno y la dinámica de un robot móvil. Dado que estos sensores tienen aplicaciones amplias fuera de la robótica móvil, son sensores de alta calidad y bajo coste. Los codificadores ópticos o encoders incrementales, en concreto, se han convertido en el dispositivo más popular para medir la velocidad angular y la posición dentro de una unidad de motor dentro del eje de un mecanismo de rueda o dirección. En robótica móvil, los codificadores se utilizan para controlar la posición o la velocidad de las ruedas y otras articulaciones accionadas por motor. Dado que estos sensores son propioceptivos, la estimación de la posición respecto al robot es correcta pero a la hora de aplicar a los problemas de localización son necesarios diferentes métodos de corrección [SN04].

3.2.2.3. Sensores de orientación

Los sensores de orientación son de particular importancia para el posicionamiento de un robot móvil ya que pueden ayudar a compensar los errores surgidos en un método de posicionamiento basado en odometría, ya que cualquier pequeño error de orientación momentánea hará que un error de posición lateral vaya en constante crecimiento. Por esta razón, es de gran ayuda poder detectar los errores de orientación y corregirlos inmediatamente. Los sensores más utilizados para determinar la orientación del robot son el giroscopio y la brújula. Éstos son propioceptivos como el giroscopio, o extereoceptivos como la brújula.

La brújula es un sensor que proporciona una medida de orientación absoluta, necesidad importante en la navegación de plataformas autónomas. Este sensor obtiene la orientación angular con respecto al campo magnético de la tierra. El sensor más comúnmente conocido de este tipo es probablemente la brújula magnética. Pese a que las brújulas están diseñadas para evitar las variaciones que sufren en sus lecturas causadas por campos eléctricos o estructuras metálicas que se encuentran en el entorno, los errores en las mediciones que proporcionan son aún significativas.

El giroscopio es una brújula de medida incremental que mide cambios en la orientación del robot. Preservan su orientación en relación con un marco de referencia fija. Por tanto, proporcionan una medida absoluta sobre un sistema móvil. En su contra tiene una limitación importante, ya que con el tiempo va acumulando un error.

3.2.2.4. Sensores de proximidad

Normalmente los sensores de proximidad tienen una salida binaria, la cual indica la presencia de un objeto dentro de un intervalo de distancia especificado. Este tipo de sensores se suelen utilizar, por ejemplo, para agarrar o evitar un objeto. Existen varios tipos de sensores de proximidad dependiendo del método de medición. Entre ellos se encuentras los sensores inductivos, sensores de efecto hall y los sensores capacitivos.

Los sensores inductivos son los de uso más frecuente en la industria. Se basan en un cambio de inductancia debido a la presencia de un objeto metálico. Los sensores de efecto hall, se basan en la modificación de un campo magnético por presencia de un objeto metálico. A diferencia de los sensores inductivos y de efecto hall que detectan solamente materiales ferromagnéticos, los sensores capacitivos, en cambio, son potencialmente capaces (con diversos grados de sensibilidad) de detectar materiales sólidos y líquidos.

3.2.2.5. Sensores de contacto

Los sensores de contacto (*bumpers*) son sensores pasivos que no necesitan ningún estimulo para ser activados. Como su propio nombre indica, se utilizan para obtener información asociada con el contacto. Por ejemplo, para obtener la información del contacto entre un robot móvil y un objeto del entorno. Los sensores de contacto se clasifican en dos categorías dependiendo de la señal de salida que ofrecen: binarios o analógicos. Los sensores binarios básicamente conmutan ante la presencia o ausencia de un objeto. Por el contrario, los sensores analógicos proporcionan a la salida una señal proporcional a una fuerza local.

3.2.2.6. Sensores de visión

Las cámaras de vídeo son sensores muy utilizados en percepción gracias a las numerosas ventajas que presentan, sobre todo la resolución, velocidad en la adquisición de la información y bajo costo. En su contra presentan una fuerte dependencia de las condiciones ambientales de iluminación, así como la complejidad computacional requerida a la hora de obtener información [JB96].

Las cámaras PTZ (Pan, Tilt, Zoom), usadas comúnmente en robótica móvil, son sensores monoculares que capturan vídeo o imágenes. A diferencia de las cámaras fijas, estas cámaras con tres grados de libertad permiten ajustar la visión al objeto de interés. El ángulo *pan* se refiere al desplazamiento de la cámara que permite girar sobre su propio eje horizontal. El ángulo *tilt*, en cambio, se refiere a la inclinación que permite ajustar la cámara respecto a su eje vertical. Por último, el zoom.

Últimamente se está incrementando el uso de cámaras Kinect de Microsoft, gracias a las amplias posibilidades que puede ofrecer. El sensor Kinect es una barra horizontal conectada a una base con un pivote motorizado con un rango de movimiento de aproximadamente 30°. Consta de una cámara RGB, que funciona utilizando unos sensores capaces de convertir la señal recibida en forma de fotones a una señal electrónica digital que descompone la luz capturada en tres componentes: rojo, verde y azul. Además, dispone de un sensor de profundidad formado por dos componentes: un generador de láser de infrarrojos y un sensor CMOS monocromo de luz infrarroja. Esta composición permite a la cámara capturar vídeo con información en tres dimensiones bajo cualquier condición de iluminación [RMDC11].

Capítulo 4

PARADIGMAS DE CONTROL ROBÓTICO

Contenido

Las funciones de un robot inteligente se pueden dividir en tres categorías. Estas categorías son *percepción*, *planificación* y *actuación*. Cuando una función obtiene datos procedentes de los sensores del robot y produce una salida útil para otras funciones, la función pertenece a la categoría de percepción. Cuando la función requiere de información (ya sea de los sensores o de su propio conocimiento simbólico sobre el entorno) y produce una o más tareas a realizar por el robot (alcanzar la luz, seguir 10 metros o parar) pertenece a la categoría de planificación. Las funciones que producen comandos de salida a los actuadores, por ejemplo los motores pertenecen a la categoría de actuación. En la tabla 4.1 se presentas estas tres categorías en términos de entrada y salida.

Primitiva	**Entrada**	**Salida**
Percibir	datos del sensor	información sensada
Planificar	información (sensada y/o cognitiva)	directivas
Actuar	información sensada o directivas	comandos para actuadores

Cuadro 4.1: Primitivas de robot en términos de entrada y salida

El principal objetivo de las arquitecturas de control es conseguir que un robot se comporte de un modo inteligente. El robot debe ser capaz de adaptarse al entorno y lograr una integración natural. Para ello se emplean diferentes tipos de arquitecturas de control. La arquitectura se refiere a la manera en el que se coordina la percepción y actuación de un robot.

Hay diferentes maneras de definir qué es una arquitectura de control y todas ellas son válidas. Walter Jacobs describe que el comportamiento depende de la selección de una estrategia donde ésta es emitida por un componente del sistema llamado control [Jac72]. Maja Mataric explica que una arquitectura de control es una manera de organizar un sistema de control y que la arquitectura impone restricciones a la manera en el que el control debe ser resuelto [Mat92]. Ronald Arkin define con un nivel de abstracción superior, indicando que la arquitectura de control son las especificaciones y sistemas programados que proveen las herramientas y lenguajes para la construcción de sistemas basados en el conocimiento [Ark98].

Existen muchas maneras de controlar un robot pero todos se basan en un espectro bien definido. Existen cuatro enfoques básicos que se utilizan hoy en día: el control deliberativo, el control reactivo, el control híbrido y el control basado en comportamientos. En el control deliberativo primero se piensa bien lo que se pretende hacer y después se actúa en consecuencia. En el control reactivo en cambio, directamente se actúa o reacciona en función a los estímulos

percibidos, sin pensar. El control híbrido es una mezcla de los anteriores donde se piensa y se actúa de manera paralela. Por último, el control basado en comportamientos se define como "pensar en la manera de actuar" [Mat01].

No existe un enfoque mejor que otro. Cada cual tiene sus puntos fuertes y debilidades. Deliberar es lento y la reacción debe ser rápida. La ventaja de deliberar es que se puede planificar para poder evitar acciones poco apropiadas, pero por contra, mientras el robot piensa pueden ocurrir diversos problemas. Además, deliberar requiere mucha información precisa. El entorno, al ser dinámico, cambia mientras el robot delibera, por lo que cuanto más piensa el robot, más inexacta es su planificación. Todo depende de la tarea del robot y su entorno. Si la tarea y el entorno requieren que el robot se mueva y reaccione muy rápidamente, generalmente no hay tiempo para pensar. Por otro lado, si el entorno no cambia mucho, y el robot tiene tiempo suficiente, se puede planificar de antemano para encontrar la mejor acción.

La figura 4.1 describe cuáles han sido las bases para las diversas técnicas desarrolladas que a continuación se especificarán más a fondo.

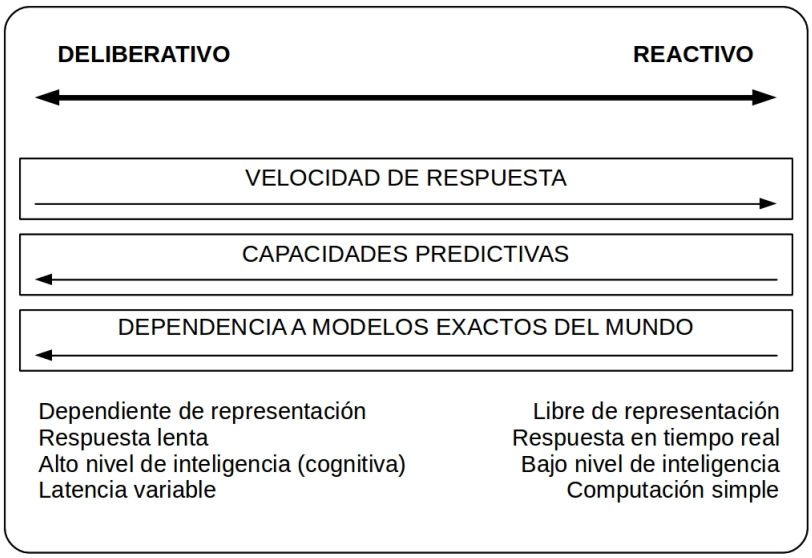


Figura 4.1: Espectro de sistema de control de robot.

4.1. Control deliberativo

El control deliberativo tiene su origen en las primeras aplicaciones de la robótica. Estos intentos de construir robots autónomos se basaron fundamentalmente en métodos y técnicas de la Inteligencia Artificial clásica. Este tipo de control dominó las implementaciones robóticas desde 1966 con la aparición de Shakey [Nil84], principal exponente del paradigma deliberativo.

Esta arquitectura de control consta de tres pasos que deben realizarse en la secuencia de percibir, planificar y actuar (ver Figura 4.2). El módulo responsable de la percepción, tiene que obtener una representación altamente fiable del estado actual del sistema basándose en las lecturas de los sensores y el modelo interno del mundo. Después de extraer a partir de las lecturas la situación actual del entorno, la tarea del módulo de planificación es encontrar la secuencia de acciones que, dado el estado actual y la meta, tratará de llevar al robot hasta su objetivo. Por último, el módulo de la actuación es el encargado de transformar la secuencia de acciones en señales que los actuadores entienden para así poder avanzar [JI11]. Este ciclo se repite una y otra vez.

El robot requiere un conocimiento completo sobre su entorno, además, este tipo de control a menudo requiere asumir la validez del modelo que se posee sobre el entorno. El modelo sobre el cual se calculan las acciones debe ser consistente, confiable y seguro. Si la información utilizada es inexacta o ha variado a lo largo del tiempo, los resultado de las acciones pueden no ser válidas. Estos modelos que representan el entorno se construyen a partir del conocimiento previo del entorno y la información recopilada por los sensores [Mur00].

Una de las características más destacadas es que son sistemas estructurados secuencialmente con una clara división entre las diferentes funcionalidades. Por el contrario, se precisa gran cantidad de tiempo de procesamiento y capacidad de almacenamiento, y esto puede presentar algunos problemas en ciertas aplicaciones.

4.2. Control reactivo

El paradigma reactivo surgió de la insatisfacción con el paradigma deliberativo. Este tipo de control se caracteriza por la fuerte dependencia existente

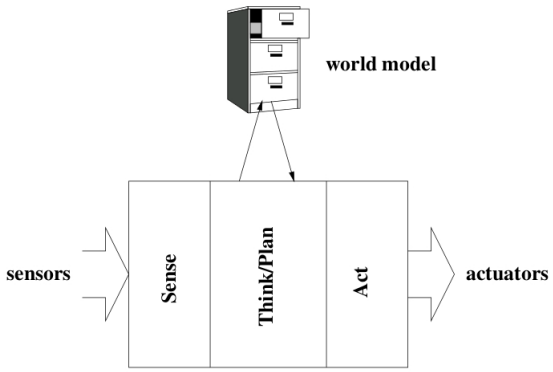


Figura 4.2: Esquema de control deliberativo.

entre la información obtenida a través del sistema de percepción y las acciones que se generan. De esta manera el robot puede responder rápidamente a los cambios de un entorno no estructurado. El control reactivo se resume como "estimulo-respuesta" donde no existe el modulo de planificación. Dado que no se debe planificar sobre la próxima tarea, la respuesta es más rápida. El control, por lo general, consiste en un conjunto de reglas donde actúan como reflejos simples. Desde una perspectiva matemática, los módulos reactivos son simplemente una función de transferencia, la transformación de los estímulos sensoriales en los comandos de actuación [Mur00].

El control deliberativo se basa en gran parte en la planificación. La acción del robot se resuelve mediante el razonamiento realizado sobre el modelo del entorno una vez deducido el estado actual a partir de las entradas sensoriales obtenidas. En los sistemas de control de reactivos ocurre lo contrario, se basan en reflejos, por lo que el proceso de razonamiento del control reactivo es mucho más rápido. Prevalece la rapidez de respuesta sobre el coste computacional del razonamiento y se ejecuta un conjunto de reglas precompiladas activadas por la información sensorial recibida. El control reactivo se puede resumir como la colección de condiciones y acciones, donde las condiciones pueden estar sujetos a las lecturas sensoriales, o de los estados internos [JI11].

La desventaja de este tipo de control deriva de su falta de memoria, además de la falta de representación interna del entorno. Como consecuencia, no puede adaptarse ni aprender. Este hecho representa un grave inconveniente en el nivel de complejidad que los comportamientos que este tipo de sistemas pueden mostrar.

4.3. Control híbrido

Los sistemas híbridos pueden ser considerados como la evolución de los sistemas de deliberación. En la década de los 80 hubo una gran tendencia a desarrollar robots utilizando el modelo reactivo. De esta manera, los robots operaban en tiempo real y sus componentes podían ser de bajo coste, tanto económico como computacional. Pero el obtener una respuesta rápida implica la eliminación de la planificación o las funciones que intervienen a la hora de recordar o razonar sobre el estado global del robot en relación a su entorno. En entornos complejos y dinámicos los sistemas reactivos producen un resultado robusto. Sin embargo, las decisiones que deben tomar este tipo de controles a menudo pueden ser una desventaja. Cuando un entorno puede modelarse de manera cercana a la exacta, se reduce la incertidumbre y se crea garantía de que el entorno es estructurado, el control deliberativo es una mejor opción. Pero en el mundo real no existen condiciones ideales que favorezcan el uso de métodos deliberativos [JI11].

Con el objetivo de obtener las ventajas de ambos controles se crean los sistemas híbridos, añadiendo funciones cognitivas a los sistemas reactivos. Estos son capaces de incorporar el razonamiento deliberativo y la ejecución puramente reactiva. Por ello, son capaces de obtener un gran potencial. La suma de diversas formas de conocimiento en una arquitectura puede hacer que en el desempeño de las tareas se obtengan mejores resultados.

La idea de *planificar* y después *percibir-actuar* se desarrolló a partir de dos suposiciones del paradigma híbrido. En primer lugar, la planificación implica un largo plazo y requiere de un conocimiento global por lo que no se puede asociar a la ejecución en tiempo real. En segundo lugar, la planificación y los algoritmos de modelado del entorno son computacionalmente costosos por lo que tampoco se pueden asociar a la ejecución a tiempo real, ya que podrían ralentizar la velocidad de reacción. La organización de la percepción en los sistemas híbridos es mas compleja. La percepción es puramente híbrida, puesto que se aplica tanto a la planificación, como a las acciones reactivas. En los módulos reactivos, la detección sigue siendo como lo fue para el paradigma reactivo: local y específico. Pero la planificación y la deliberación requieren modelos globales del entorno [Mur00].

4.4. Control basado en comportamientos

Los sistemas basados en comportamientos están fundamentados en los sistemas reactivos, pero con unas capacidades más complejas. A menudo se confunden con ellos. Básicamente estos sistemas son arquitecturas de control distribuidas en paralelo donde cada unidad de procesamiento produce un comportamiento en el robot.

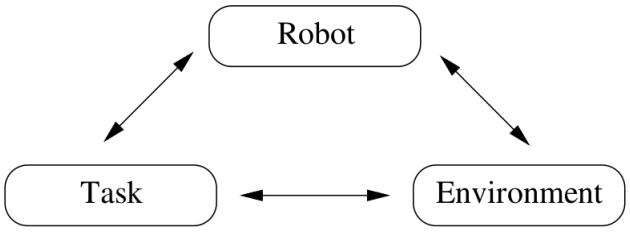


Figura 4.3: Elementos fundamentales para un comportamiento.

Un comportamiento se compone de tres elementos fundamentales, la tarea, el entorno y el robot en sí (ver figura 4.3). Estos tres elementos están estrechamente unidos y no pueden ser considerados de forma independiente [Neh03]. El comportamiento surge de la interacción entre estos tres componentes, y si uno de estos componentes cambia, también lo hace el comportamiento. Por lo tanto, un comportamiento es un hilo de ejecución que sólo tiene un objetivo generalmente específico para el entorno. Al igual que las reglas de control reactivo, los comportamientos se ejecutan en paralelo, para conseguir una respuesta rápida. Una de las mayores diferencias entre los sistemas reactivos y los sistemas basados en el comportamiento es la imposibilidad de adquirir suficiente conocimiento del entorno utilizando exclusivamente reglas. Sin embargo, es posible construir una representación del entorno usando una red apropiada de comportamientos [JI11].

Uniendo los comportamientos se puede obtener una máquina inteligente, pero el control es necesario para coordinar todos los comportamientos. Este tipo de sistema está compuesto por módulos de comportamiento (ver figura 4.5) que funcionan en paralelo, donde el problema de la fusión de sensores de las arquitecturas deliberativas se transfiere a la fusión de la acción en arquitecturas basadas en comportamientos. Los módulos ofrecen diferentes salidas a los actuadores, pero sólo una acción cada vez puede ser ejecutada por los actuadores y por lo tanto, las propuestas están coordinadas. En términos generales, hay

dos enfoques para la selección de acciones: arbitraje y fusión. El arbitraje es el proceso de seleccionar una acción entre todos los candidatos posibles, también llamado estrategia competitiva. La fusión o enfoque cooperativo, en cambio, es el proceso de combinar todas las salidas de candidatos en una acción como única salida (figura 4.4).

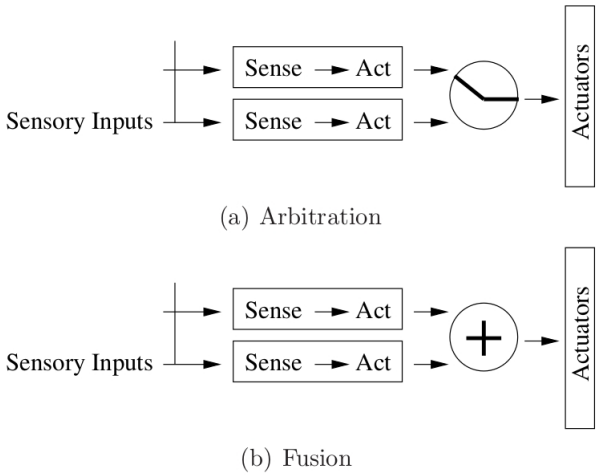


Figura 4.4: Elementos fundamentales para un comportamiento.

La diferencia con los sistemas reactivos reside en el nivel de representación permitido y en las unidades de procesamiento. En lugar de reglas precompiladas, las unidades básicas de procesamiento de los sistemas basados en comportamiento tienen diferentes características:

- Cada comportamiento tiene su propio objetivo.
- Todos los comportamientos se ejecutan continuamente. Los comportamientos reciben entradas y producen salidas en un bucle continuo.
- Un comportamiento es capaz de comunicarse con otros comportamientos, de esa manera puede recibir información de otras unidades.
- Las reglas precompiladas de los sistemas reactivos producen acciones simples y con la unión de las salidas de las diferentes reglas se consigue un comportamiento simple.

Al igual que en el control reactivo, los comportamientos en estos sistemas se ejecutan en paralelo para conseguir una respuesta rápida. En los sistemas

reactivos es imposible adquirir el conocimiento del entorno solo con reglas, sin embargo, es posible construir una representación utilizando una red adecuada de comportamientos.

Una de las arquitecturas de este campo es la arquitectura de subsunción desarrollada por R. Brooks en 1986. Su enfoque, basado en comportamientos, iba en contra de la tradicional investigación de la Inteligencia Artificial del momento. Brooks argumentó que el paradigma *percibir-planificar-actuar* utilizado en los primeros robots autónomos era, de hecho, perjudicial para la construcción de robots de trabajo reales. Además, sostuvo que la construcción de modelos globales y el razonamiento mediante la utilización del conocimiento explícito de la representación simbólica era un impedimento para una respuesta robótica oportuna y que en realidad llevaba a los investigadores en la dirección equivocada [Bro99].

La base de la robótica basada en el comportamiento es comprender que el acoplamiento de la percepción y la acción da lugar a todo el poder de la inteligencia y que la cognición es sólo el punto de vista de un observador.

4.4.1. Arquitectura de subsunción

La arquitectura de subsunción propuesta por Brooks (1986) es la arquitectura basada en comportamientos más conocida. La palabra subsunción viene del verbo subsumir, que se podría definir como "pensar en un objeto como parte de un grupo" [RAE01]. En el contexto de la robótica basada en comportamientos, el nombre de subsunción surge del proceso de coordinación utilizado entre las capas de comportamientos dentro de la arquitectura. Las actuaciones complejas subsumen de comportamientos más simples.

La arquitectura de subsunción está compuesta por capas que están dispuestas en orden creciente de complejidad. Cada capa implementa un nivel de comportamiento. Cada capa está compuesta por varios módulos de comportamiento que trabajan en paralelo. Los módulos de comportamiento forman una arquitectura de control basada en la prioridad, donde la capa inferior tiene menor prioridad. Las capas de módulo de nivel superior pueden subsumir o inhibir las salidas/entradas de los módulos en los niveles inferiores. En algunos casos se utiliza una señal de reinicio para devolver al comportamiento a su estado inicial. Cada módulo de comportamiento lleva a cabo una acción y es responsable de su propia percepción del mundo.

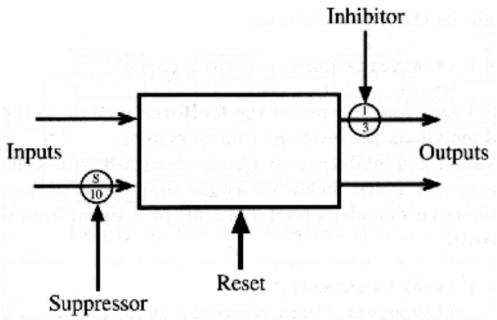


Figura 4.5: Estructura del módulo de comportamiento.

De esta manera, las capas de comportamientos se desarrollan gradualmente y, por tanto, el sistema muestra robustez si una capa falla (figura 4.6). No es evidente cómo debe ser establecido la prioridad de las capas. Dependiendo de las condiciones ambientales, el orden de prioridad seleccionado puede llevar al robot a su meta o al fracaso.

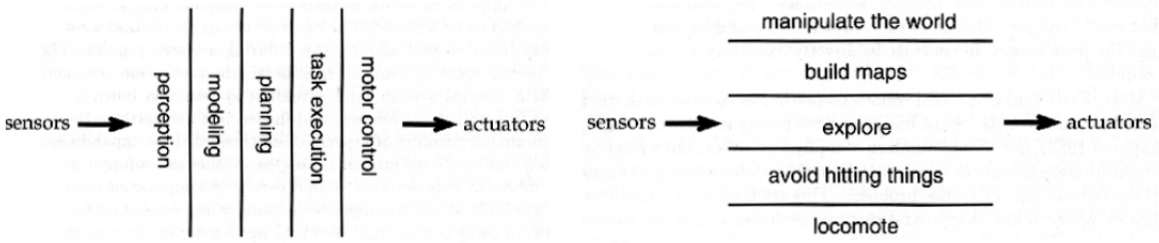


Figura 4.6: Diferencia entre descomposición tradicional (izquierda) y basada en comportamientos (derecha).

Parte III

DESARROLLO TÉCNICO

Capítulo 5

HERRAMIENTAS SOFTWARE

Contenido

Cuando las tareas a realizar por los robots son simples, es suficiente una solución desarrollada por herramientas de bajo nivel como pueden ser los microcontroladores. A medida que las tareas se vuelven más complejas, más potencia es requerida, por ello, un ordenador es más adecuado que un microcontolador. Para ello existen varias herramientas para abstraer las capas de bajo nivel de los dispositivos, de tal manera conseguir un código modular, portable a otras arquitecturas, otros hardware.

En este proyecto se ha utilizado la herramienta *Player/Stage*, junto con SORGIN, un marco software diseñado especialmente para el desarrollo de arquitecturas de control basadas en comportamientos [ALR+03]. En las próximas líneas se profundizan sus características más a fondo.

5.1. Player/Stage

Player/Stage es un proyecto software de código abierto creado y mantenido por la comunidad académica internacional de robótica, disponible a través de SourceForge. El objetivo es permitir la investigación en robots y sistemas de sensores. El proyecto está compuesto por dos tipos de componentes:

- **Player** es un servidor de dispositivos robóticos.
- **Stage** y **Gazebo** son simuladores robóticos.

5.1.1. Player

Player es un servidor en red para el control de robots, que se ejecuta en un robot y proporciona una interfaz simple y sencilla para comunicarse con los sensores y actuadores del robot a través de la red IP. El programa cliente se comunica con Player a través de un socket TCP, leyendo datos de los sensores, escribiendo órdenes en los actuadores y configurando dispositivos durante la ejecución.

Player respalda una amplia variedad de hardware robótico. La plataforma original de Player es la familia ActivMedia Pioneer 2, pero muchos otros robots y muchos sensores comunes tienen soporte actualmente. La arquitectura

modular de Player hace que sea muy fácil añadir soporte para nuevo hardware, además existe una comunidad de usuarios y desarrolladores que contribuyen con nuevos drivers.

Player está diseñado para ser independiente del lenguaje de programación y de la plataforma. El programa cliente se puede ejecutar en cualquier máquina que tenga una conexión de red a su robot, en el que se ejecuta Player y puede ser escrito en cualquier lenguaje que soporte sockets TCP. Actualmente existen clientes disponibles en C, C++, Tcl, Java y Python. Además, Player no impone restricciones sobre como estructurar sus programas de control del robot.

Player permite a varios dispositivos utilizar la misma interfaz. Por ejemplo, los drivers del robot Pioneer 2 (direccionamiento diferencial) y del robot B21 de RWI (direccionamiento síncrono) utilizan la interfaz de *position* de Player para permitir el control de movimiento del robot. De esta manera el mismo código podría servir para los dos robots. Esta característica es muy útil cuando se combina con el simulador Stage.

Otra característica de Player es que tiene capacidad para soportar virtualmente cualquier número de clientes, siempre dependiendo de las capacidades de la máquina en la que se ejecute. Además, permite que varios clientes accedan simultáneamente a dispositivos comunes de uno o varios robots.

Player es un software libre bajo la Licencia Pública GNU.

5.1.1.1. Arquitectura cliente-servidor

Player es el módulo que define la arquitectura del sistema. Se puede descomponer en dos partes. Por un lado, hay un entorno servidor que se ejecuta sobre el robot (real o simulado) que recibe peticiones TCP sobre el protocolo IP.

Por otro lado, hay uno o más clientes que acceden a ese robot, para lo que existen unas librerías cliente que evitan trabajar a bajo nivel. El programa cliente se comunica con el robot a través de sockets TCP, permitiendo la lectura de los sensores, accionar los mecanismos del robot, así como la configuración de ciertos parámetros y dispositivos del robot.

De esta manera, el desarrollador sólo necesita escribir un programa cliente

que haga uso de las interfaces proporcionadas por las librerías de Player. Las librerías se encargan de contactar con el servidor y enviar mensajes, de forma que sólo se necesita usar dichas interfaces para controlar el robot. Con esta arquitectura, un mismo cliente que use unas interfaces concretas, puede controlar diferentes robots que tengan los drivers apropiados para implementar las mismas interfaces (incluso aunque el hardware sea diferente). En definitiva, los drivers comunican el hardware del robot con los programas cliente a través de las interfaces estándar.

5.1.1.2. Interfaces, drivers y dispositivos

A continuación se procede a la definición de algunos términos utilizados en el ámbito de Player/Stage.

- **Interfaz**. Una interfaz ofrece un conjunto de operaciones para acceder y controlar un sensor. Permite controlar un dispositivo a alto nivel y de forma genérica, dejando de lado el lenguaje específico del propio hardware del dispositivo.
- **Driver**. Un driver es el software encargado de controlar los dispositivos de forma directa. Se comunica con el hardware y a su vez implementa una o varias interfaces para que otros puedan acceder al dispositivo de forma genérica, abstrayéndose del lenguaje específico del hardware.
- **Dispositivo**. Es la pieza hardware en sí, un sensor o un actuador.

Estos elementos están relacionados entre sí, de forma que un programa cualquiera utiliza las interfaces para enviar comandos a los sensores y actuadores del robot. A su vez, las interfaces se comunican con los drivers y son éstos los que se comunican de forma directa con los dispositivos.

Como ya se ha visto, Player posee dos partes bien diferenciadas: un servidor que se ejecuta en el robot y se comunica con su hardware y un programa cliente que se encarga de enviar peticiones al servidor. Esta comunicación se realiza a través de unas interfaces estándar que proporcionan las librerías de Player. Dichas interfaces proveen al desarrollador funciones de alto nivel para comunicarse con el servidor que se está ejecutando en el robot, evitando la parte engorrosa de la comunicación entre un servidor y un cliente mediante

sockets TCP. Del mismo modo, el servidor implementa las mismas interfaces que el cliente, de forma que la comunicación sea posible.

A parte de las interfaces para la comunicación entre cliente y servidor, existen interfaces para comunicarse con los dispositivos del robot (láser, motor, GPS, etc.). Gracias a estas interfaces, el hardware específico de cada robot (el tipo de sensores y actuadores) es indiferente para el desarrollo de los programas clientes.

5.1.2. Stage

Stage es un entorno de simulación para la plataforma Player. Ofrece una interfaz gráfica en dos o tres dimensiones que imita el mundo real gracias al uso de una imagen que sirve de mapa o plano del entorno del robot. Stage está diseñado para soportar la investigación en sistemas autónomos multi-agente, de modo que proporciona modelos computacionalmente baratos y simples de una gran cantidad de dispositivos en lugar de intentar emular cualquier dispositivo con alta fidelidad.

Stage se usa a menudo como plugin de Player y proporciona poblaciones de dispositivos virtuales para Player. Los usuarios escriben programas clientes para el servidor Player. La idea es que los clientes no noten la diferencia entre robots reales y sus equivalentes simulados. Stage permite desarrollar prototipos rápidos destinados para robots reales, así como experimentar con dispositivos reales que el usuario no tiene en su posesión, o no dispone de ellos en un momento dado. Además, proporciona la simulación de una amplia variedad de sensores y actuadores, tales como sónar, medidores de distancia por láser, dispositivos de detección de color por visión, cámaras 3D de profundidad u odometría (con modelo de error de desplazamiento).

5.1.2.1. Simulación

Internamente, el driver Stage no accede a los dispositivos hardware, sino que simula dichos dispositivos. Esto es posible gracias a la arquitectura de Player, la cual permite una abstracción total a la hora de desarrollar programas, dado que no es necesario comunicarse con el hardware de forma directa, sino a través del driver. Por esto, las peticiones se envían al driver y las respuestas se reciben

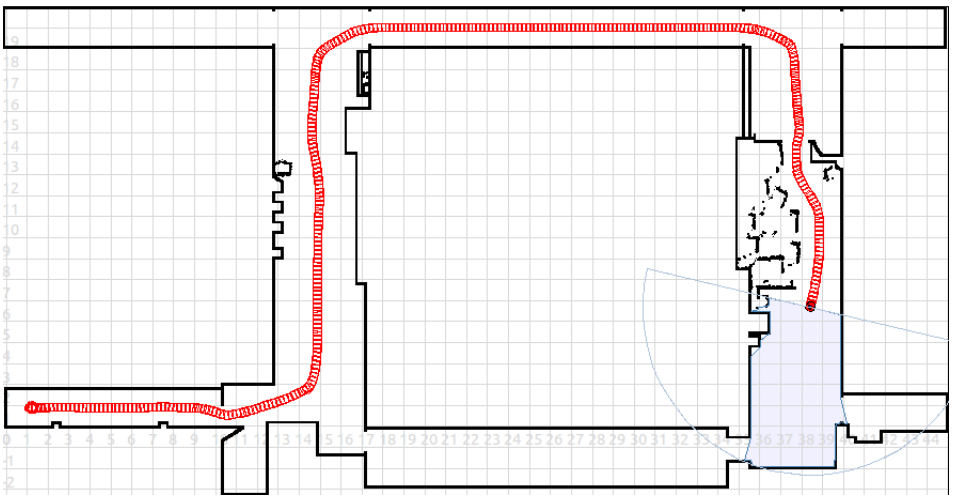


Figura 5.1: Simulación de Stage.

del propio driver. En un caso real, el driver retransmitiría las peticiones a los dispositivos y recibiría datos de ellos, pero en simulación este paso no existe. Las respuestas de los dispositivos son simuladas y no reales.

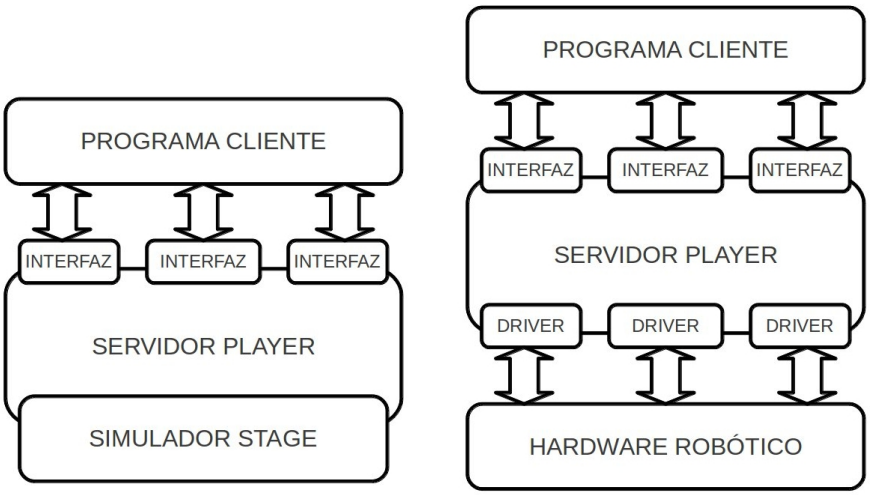
En la figura 5.1 se puede apreciar la ejecución de una simulación en Stage.

5.1.3. Arquitectura general

Una vez conocidos los elementos de Player/Stage en la figura 5.2 se puede apreciar la arquitectura básica con la utilización de Stage y en la figura 5.2 con la utilización de un robot real.

5.2. SORGIN

SORGIN es un framework para el desarrollo de arquitecturas de control basados en comportamientos [ALR+03]. Para implementar una arquitectura de control, es esencial desarrollar herramientas necesarias para crear y depurar los diferentes comportamientos. SORGIN es un conjunto de estructuras de datos y bibliotecas de funciones asociadas. Desde un punto de vista genérico,



a) b)

Figura 5.2: a) Arquitectura de simulación de Player/Stage. b) Arquitectura utilizando player como servidor.

el comportamiento global puede ser considerado como una red de entidades concurrentes activas (hilos) que interactúan de manera asíncrona entre ellas. SORGIN identifica estas entidades activas con componentes software y define su interacción creando métodos base de modularidad y bibliotecas de comportamiento. Ha sido desarrollado utilizando el lenguaje de programación C e hilos POSIX, ambos elementos estándares y portables.

Los comportamientos de SORGIN consisten en:

- Un conjunto de entradas y salidas que son enlaces a una clase específica de datos.
- Un enlace a una función de inicialización (*start*) en el que se llevará a cabo todo lo necesario antes del inicio del bucle principal. Esta función es especialmente importante cuando el módulo trata directamente con un dispositivo físico.
- Un enlace a la función a realizar en el bucle principal, que se ejecuta como un proceso independiente (*calculate*).
- Un enlace a una función que se lleva a cabo cuando el comportamiento o módulo interrumpe su ejecución (*stop*).

No hay un componente central que dirige el controlador global. Cuando los comportamientos se ponen en marcha, son plenamente responsables de su propia ejecución, cada uno de ellos funciona a su respectivo ritmo. Después de iniciar y lanzar el módulo de comportamiento, se crea un hilo donde se ejecutará hasta que termine o sea detenido.

La comunicación entre los módulos se realiza mediante el establecimiento de conexiones entre los enlaces de las entradas y salidas de los módulos y los datos. Estas estructuras de datos son más que simples mensajes. Cada elemento contiene toda la información necesaria para leer y escribir con seguridad los datos a través de la red cuando es necesario, o para evitar conflictos cuando la información es requerida a nivel local. De esta manera, durante la ejecución de una instancia de la función *calculate* el diseñador de la arquitectura de control no necesita preocuparse por esos detalles, sólo deben ser llamadas las funciones estándar ofrecidas para leer/escribir las entradas/salidas. Dado que es inherentemente modular, facilita al diseñador del sistema robótico ampliar las competencias del robot añadiendo nuevas características sin la necesidad de rediseñar o descartar las antiguas, permitiendo la construcción de sistemas robóticos cada vez más complejos.

Uno de los aspectos positivos de este framework es que permite la reutilización del código. Promueve la reutilización automática de comportamientos a través de las diferentes tareas, y por tanto, la generación automática de bibliotecas de comportamientos. La abstracción del comportamiento se proporciona gracias a los objetos **behavior** y **io_data**. En la figura 5.3 se muestra la base principal de SORGIN.

La figura 5.4 muestra cómo Player y SORGIN se combinan. Los comportamientos (**behavior**) de SORGIN se ejecutan como programas cliente de Player y a través de los drivers de éste último acceden a los dispositivos del robot.

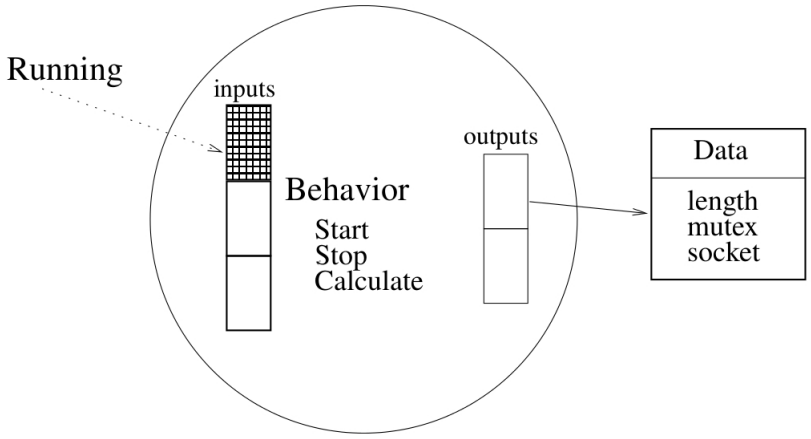


Figura 5.3: Estructura principal de SORGIN.

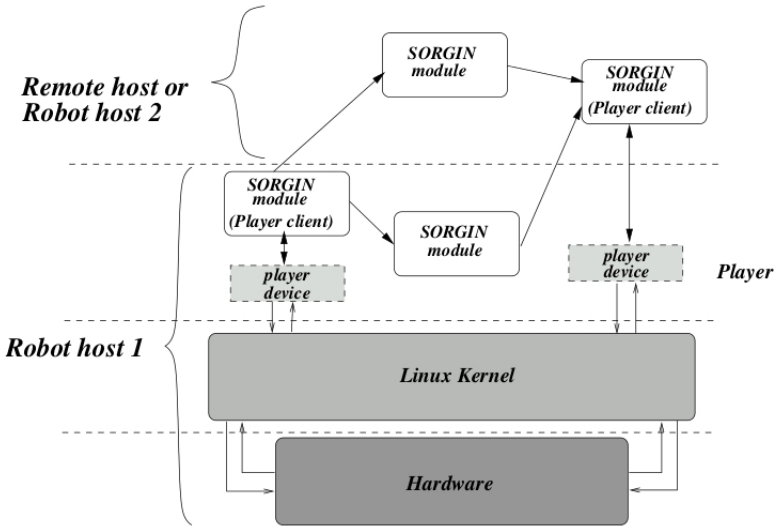


Figura 5.4: Ejemplo de cómo los comportamientos se implementan usando el framework SORGIN.

Capítulo 6

SISTEMA BASADO EN COMPORTAMIENTOS PARA LA EXPLORACIÓN DEL ENTORNO

Contenido

El problema de la navegación en la robótica móvil se puede resumir en tres preguntas [BEF96]:

- ¿Dónde estoy?
- ¿A dónde voy?
- ¿Cómo puedo llegar allí?

A día de hoy no existe una solución concreta ni sencilla acerca de la localización de los robots móviles. Una manera de mantener la localización es mediante el uso de la odometría. Es uno de los métodos más usados para estimar la posición del robot, ya que proporciona una buena precisión a corto plazo y es barato de implantar. Sin embargo, la idea de la robótica móvil es tener el robot localizado a lo largo del tiempo, y la odometría presenta una inevitable acumulación de errores (figura 6.1). En concreto, la acumulación de errores de orientación causa grandes errores en la estimación de la posición.

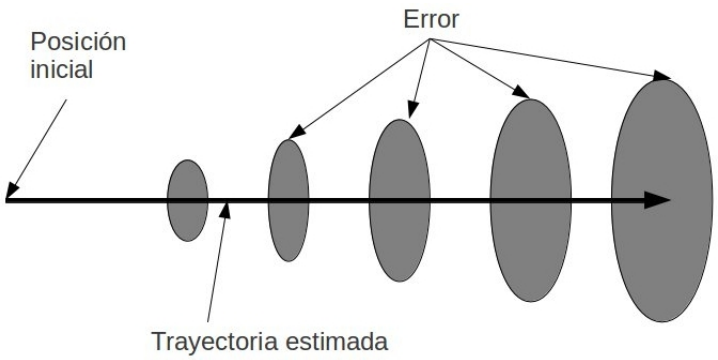


Figura 6.1: A lo largo de la trayectoria del robot el error de estimación de posición va aumentando. Crece la incertidumbre sobre cual es la posición real.

Para hacer frente a estos inconvenientes, en métodos de odometría o cualquier otro, el campo de la robótica probabilística ofrece un conjunto de algoritmos que proporcionan herramientas necesarias para mantener los estados concretos del espacio [Thr02]. Por el contrario, los robots desarrollados bajo el paradigma de comportamientos sólo requieren un subconjunto de propiedades ambientales para la localización, además, muestran un mayor grado de adaptabilidad a los cambios dinámicos del entorno.

6.1. Navegación en los sistemas basados en comportamientos

Para los robots móviles la capacidad de interactuar con un entorno dinámico y cambiante es de gran importancia. Una buena opción para hacer frente a los problemas que se puedan presentar es estructurar el sistema de control en comportamientos. El principal objetivo de esta arquitectura de control es identificar los diferentes controladores y las respuestas a entradas de sensor con los comportamientos deseados del robot. De esta manera, el sistema de control se estructura en conductas o comportamientos independientes, cada una dedicada a realizar una tarea diferente. La ventaja es que el sistema es modular y simplifica el proceso de diseño, así como la posibilidad de añadir nuevos comportamientos sin causar un aumento importante en la complejidad.

Las acciones de salida deseados se fusionan o se arbitran por un mecanismo de control (ver figura 6.2), donde se extiende una solución al problema de la fusión de las acciones. Este elemento se encarga de decidir la salida concreta y necesaria en cada momento. De esta manera se pueden tener activos al mismo tiempo a diferentes comportamientos, ya que es necesaria en numerosas situaciones. Se requiere un control coordinado.

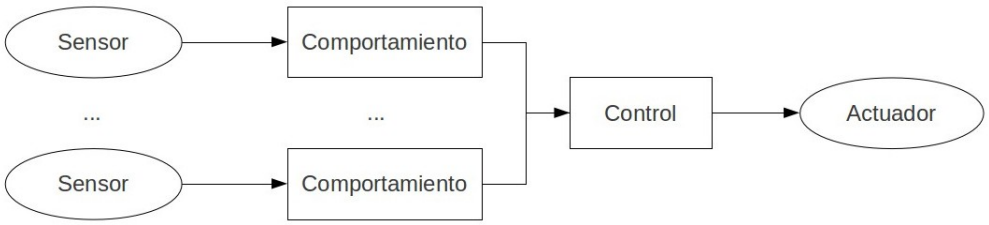


Figura 6.2: Diagrama de bloques de una arquitectura de control basada en comportamientos.

Anteriormente se ha mencionado que los problemas de la navegación en la robótica móvil se resumen en tres preguntas, pero numerosos estudios de la etología (rama de la biología y psicología que estudia el comportamiento de los animales) muestran que muchos animales e incluso personas son capaces de navegar sin la necesidad de responder a todas las preguntas. Por ejemplo, la pregunta *¿Dónde estoy?* no es necesariamente la primera pregunta que hacer. Es más importante preguntar *¿Cómo puedo llegar a mi destino?*, ya que muchas veces no se requiere saber la posición actual.

Moverse en un espacio y determinar si se ha alcanzado el objetivo o no son las capacidades mínimas para la navegación. Esto no implica que deba conocerse la posición inicial ni que sea necesaria una representación precisa del entorno. La referencia a un objetivo distingue la navegación de otras formas de comportamiento espacial, como la exploración, evitar obstáculos, orientación o la estabilización de la trayectoria.

La navegación inspirada en la biología se clasifica en dos grupos generales: navegación local y búsqueda de caminos [FM00]. *La navegación local* sólo requiere el reconocimiento de un único lugar, conocido como el objetivo. El agente elige sus acciones en función de la información sensorial o interna, sin necesidad de representar los objetos o lugares fuera del alcance sensorial actual. Los métodos de navegación local también son conocidas como tácticas o estrategias locales de control. *La búsqueda de caminos* implica el reconocimiento de varios lugares y la representación de las relaciones entre los lugares que pueden estar fuera del rango actual de percepción. La búsqueda de caminos se basa en las capacidades de la navegación local para moverse de un lado a otro. Pero, además, permite al agente encontrar lugares que no pueden ser encontrados por la navegación local.

6.2. INCA para la localización

El sistema que se ha utilizado para este proyecto combina los métodos estadísticos de INCA [IA08] con las propiedades topológicas del entorno. Para ello se genera una serie de datos para después tras un test de tipicidad, decidir si la localización es conocida o se trata de un nuevo área en el entorno.

Los datos son clasificados en k clases, $C_1, C_2, \cdots, C_k$, representados por k vectores aleatorios $Y_1, Y_2, \cdots, Y_k$. Estos vectores corresponden a:

Pasillos (*corridors*), vestíbulos (*halls*):

$$Y = (sin(\theta_0), cos(\theta_0), sin(\theta_{mean}), cos(\theta_{mean}), (x_0, y_0), (x_f, y_f), d) \tag{6.1}$$

Cruces (*junctions*):

$$Y = (sin(\theta_0), cos(\theta_0), sin(\theta_{mean}), cos(\theta_{mean}), \theta_{w_1}, \cdots, \theta_{w_{num\_ways}}, (x_0, y_0)) \tag{6.2}$$

Para poder identificar los diferentes lugares del entorno se ofrece una serie de información a los test de INCA:

- Orientación inicial y orientación media, θ_0 y θ_{mean}.
- Posición inicial y final dados por el subsistema de odometría. Las posiciones corresponden a los valores recogidos en la activación o desactivación de los diferentes nodos del entorno: (x_0, y_0), (x_f, y_f).
- La longitud d del área calculada mediante la posición inicial y final.
- Número de caminos alternativos y la orientación asociada: *num_ways* y $\theta_{w_1}, \cdots, \theta_{w_{num\_ways}}$.

Suponiendo que y_0 sea una unidad pendiente de clasificación, éste puede pertenecer a una clase C_j, $j = 1, \cdots, k$ o a ninguna clase existente y pertenece a una clase nueva. Se considera el test de INCA,

$$H_0 : y_0 \text{ pertenece a una clase existente.}$$
$$H_1 : y_0 \text{ no pertenece a una clase existente.}$$

Se minimiza la suma ponderada de las proximidades de y_0 a las diferentes clases y si éste valor resulta significativo entonces se decide que proviene de una clase diferente. En caso contrario se añade el valor a la clase C_i correspondiente.

6.3. Estructura topológica del entorno

En el enfoque basado en el comportamiento donde se usan mapas topológicos, el entorno es representado mediante una discretización más gruesa basada de los lugares distinguibles. Los mapas topológicos deben estar compuestos de comportamientos específicos firmemente acoplados a esos lugares significativos.

En el entorno utilizado para los experimentos se definen tres tipos de localizaciones que son convertidos a nodos: *pasillos, vestíbulos y cruces*. Todos ellos son identificables midiendo las características geométricas con sensores como un láser. Además, también se identifica el *callejón sin salida* (dead end)

para llevar a cabo diferentes acciones como puede ser cambiar de orientación o empezar con el proceso de mapeado. El objetivo del proceso de creación de mapas (mapeado) es obtener la información de los diferentes nodos existentes en el entorno.

Las localizaciones que el robot debe identificar no son solamente puntos del entorno sino áreas alrededor de estos puntos. Para ello, se generan una serie de datos para después testear y decidir si la localización es conocida o se trata de un nuevo área en el entorno. Estos test se realizan gracias a los métodos estadísticos de INCA [IA08], donde primero se estima el número real de grupos en un conjunto de datos y segundo se decide si una nueva unidad pertenece a uno de estos grupos previamente identificados o se trata de una unidad de valores atípicos. Cada localización o nodo se representa por un vector de observación constituido por diferentes mediciones: variables del tipo coordenadas en metros y variables del tipo orientación (ecuaciones 6.1 y 6.2) que las estadísticas de INCA utilizan para clasificar.

6.3.1. Mapa del entorno utilizado en simulación

Como ya se ha mencionado anteriormente, los experimentos simulados se llevan a cabo en el simulador *Stage*. Concretamente en un mapa basado en la tercera planta de la Facultad de Informática. Se trata de un entorno semi estructurado de tipo oficina con una geometría regular. En la figura 6.3 se aprecia el mapa.

El entorno se modela como un mapa topológico donde se han definido tres tipos de lugares distinguibles: pasillos, vestíbulos y cruces. La figura 6.4 muestra la distribución de los nodos en el entorno como resultado de la localización mediante INCA y con una odometría ideal (en el capitulo 7 se describen más los resultados). Cada nodo está representado en su punto medio en referencia a su posición inicial y final.

Los pasillos, vestíbulos y cruces pueden diferir en su orientación (el valor medio de la brújula que mantiene el robot al pasar a través de ellos en su ruta canónica). Ésta es la razón por la que cada lugar físico se corresponde a dos o más nodos diferentes en el mapa topológico.

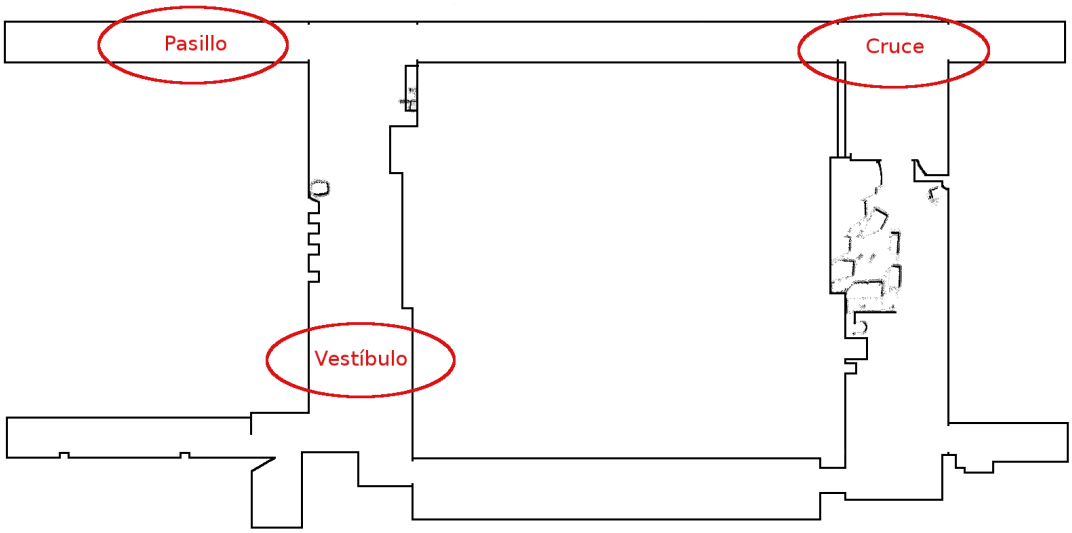


Figura 6.3: Entorno semi-estructurado de tipo oficina.

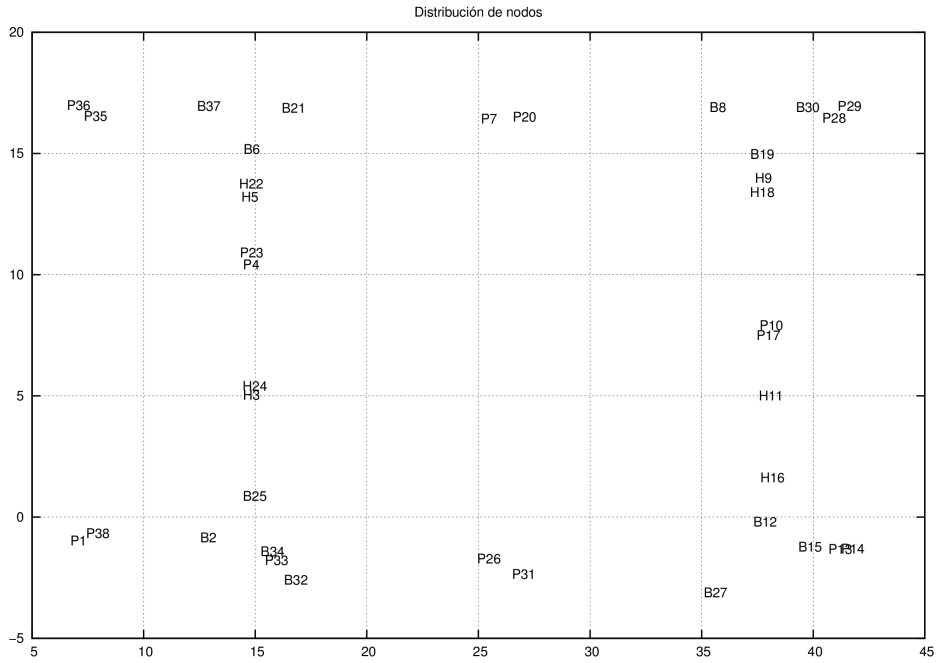


Figura 6.4: Distribución de los nodos en el entorno.

6.4. Exploración del entorno

La exploración puede definirse como el proceso mediante el cual un robot móvil recorre el entorno detectando lugares de referencia y transiciones entre ellos, para construir así de forma autónoma un modelo del entorno lo más completo posible.

En muchas aplicaciones, el proceso de exploración no se ejecuta de forma independiente, construyéndose el modelo del entorno a medida que avanza el robot. De esta manera, cada misión lleva consigo la búsqueda de un camino que una la salida con la meta. Dependiendo de cual sea la estrategia, se puede derivar en un modelo del entorno incompleto incluyendo solamente las zonas del entorno por las que se ha pasado durante la búsqueda del objetivo, sin tener en cuenta todas las alternativas desde cada lugar de referencia [dLMZA]. La creación de un modelo poco útil puede suponer un enorme contratiempo, por ello, suele ser importante escoger una estrategia de exploración adecuada. En el caso concreto de este proyecto no se efectúa una búsqueda de camino mientras se explora el entorno. La estrategia se centra en explorar el entorno para comprobar la correcta identificación de los lugares distinguibles.

6.4.1. Comportamiento de exploración

El proceso de mapeado requiere una estrategia de exploración para guiar al robot en la inspección del entorno. En la estrategia utilizada en este proyecto de fin de carrera, el comportamiento de exploración (figura 6.5) es una coordinación de las estrategias locales de navegación y subsistemas de identificación de puntos o lugares de referencia del cual el robot está dotado. Éstas son:

- Dos estrategias locales de navegación que se combinan de una manera cooperativa (suma ponderada): equilibrar el espacio libre a ambos lados del robot y seguir una orientación de brújula deseada (θ_d).

- El subsistema de identificación de puntos o lugares de referencia (*landmark*) permite al robot reconocer pasillos, pared izquierda/derecha, vestíbulos, cruces y callejones sin salida. Estos puntos o lugares de referencia se utilizan para cambiar la orientación deseada del robot.

Aunque el robot se puede posicionar en cualquier punto de partida, el mapa permanece vacío hasta que el robot encuentre un callejón sin salida. Por lo tanto, la construcción del mapa se inicia después de que un callejón sin salida se haya identificado. Gracias a esto se obtiene la medida correcta de la longitud de las localizaciones de referencia (nodos). Posteriormente, el primer pasillo, el primer cruce y el primer vestíbulo se identifican siempre como nuevos nodos ya que no hay ninguna instancia del mismo tipo almacenado en el mapa.

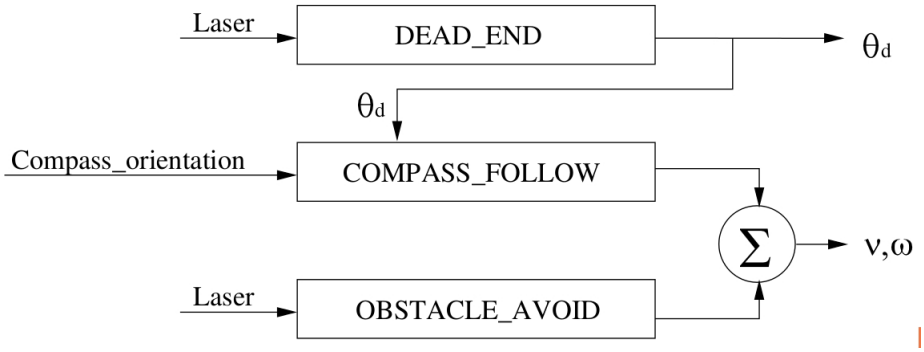


Figura 6.5: Diagrama del módulo del comportamiento de exploración (v: velocidad translacional, w: velocidad angular).

Una vez que el proceso de construcción del mapa se inicia, cada vez que el robot identifica un pasillo, un vestíbulo o un cruce, se registra la información geométrica de la ubicación identificada, para después realizar un test de tipicidad con los métodos estadísticos de INCA y decidir si la localización es conocida o se trata de un nuevo área en el entorno. Por ejemplo, cuando la ubicación corresponde a un cruce, las distintas orientaciones alternativas que el robot puede elegir se almacenan. Si un nodo de tipo cruce se visita de nuevo, ahora, se escoge el camino (orientación) no utilizado anteriormente. De esta manera, el robot tiene la posibilidad de cubrir todo el entorno.

El robot termina el proceso de exploración cuando todas las alternativas de los nodos de cruce han sido tomadas o, en el caso de las simulaciones con odometría ideal, cuando se alcance el número de nodos al cual convergen los resultados llevados a cabo. En el caso de la odometría corregida mediante láser, en cambio, cuando el error de odometría provoca un crecimiento continuo y es necesario detener las pruebas. Opciones utilizadas en las pruebas presentadas en el capítulo 7.

Capítulo 7

AUTOMATIZACIÓN DE LOS RECORRIDOS DEL ROBOT

Contenido

Este capítulo se centra en la realización de pruebas en simulación del sistema basado en comportamientos presentado en *Advances towards behaviour-based indoor robotic exploration* [JI11]. El sistema demuestra la validez de los métodos estadísticos de INCA [IA08] para la localización robótica. Para la realización de pruebas se ha modificado la estrategia de exploración original para dejar de lado la aleatoriedad que muestra el robot a la hora de tomar decisiones en cuanto a qué camino (orientación) debe tomar cuando alcanza un cruce. Mientras el robot navega por el entorno identifica diferentes lugares distinguibles que son activados gracias a comportamientos de identificación. El problema se presenta a la hora de evaluar los resultados de diferentes pruebas, ya que la aleatoriedad en la toma de decisiones lleva a obtener diferentes resultados. Por ello, el objetivo de los cambios realizados es poder evaluar más fácilmente los resultados de las diferentes pruebas simuladas. Las herramientas utilizadas para simular los resultados han sido el servidor *Player*, el marco software SORGIN y el simulador *Stage*. En cuanto al robot utilizado, se trata de una versión simulada del robot *Pioneer 3-DX* llamado **Galtxagorri**.

7.1. Estrategia de exploración

En el sistema utilizado para el desarrollo de este proyecto, al igual que la tesis en el que se basa, se divide en dos procesos: *mapeado* que se refiera a la creación de mapas cuando el entorno es totalmente desconocido y *localización* cuando se conoce el entorno y simplemente se quiere obtener la localización del robot mientras navega.

Cuando no se tiene conocimiento previo del entorno (mapa topológico), el robot comienza con el proceso de mapeado. En este punto, el robot móvil desconoce su localización con respecto al entorno global y comienza a construir un mapa mientras navega por el entorno e identifica los diferentes lugares distinguibles. En la sección 6.3 del capítulo 6 se presentaron los diferentes tipos de nodos y herramientas que se utilizan para su clasificación. El proceso de mapeado comienza justo después de un callejón sin salida (dead-end) y termina cuando se alcanzan los n nodos del que dispone el entorno ($n = 38$ en el caso de las pruebas con una odometría ideal). A la hora de decidir qué camino escoger la primera vez que se visita un cruce, siempre se opta por el camino que tiene más fácil el robot para navegar (pasillo o vestíbulo que presenta mayor anchura). La segunda vez que se pasa por un nodo de tipo cruce se utiliza el mismo criterio que al principio pero esta vez teniendo en cuenta sólo

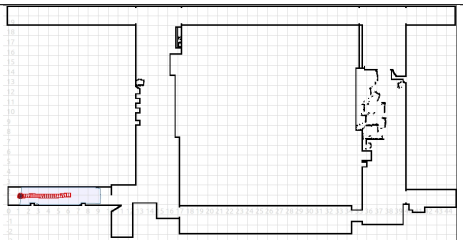
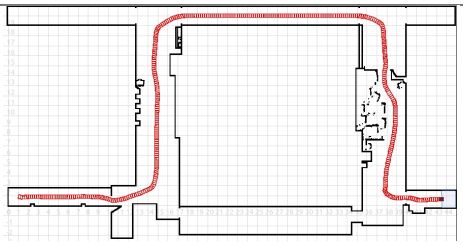
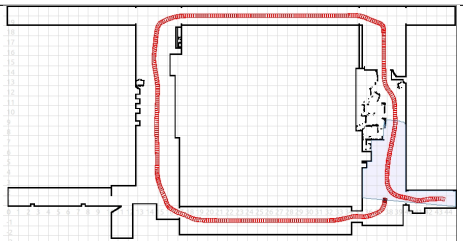
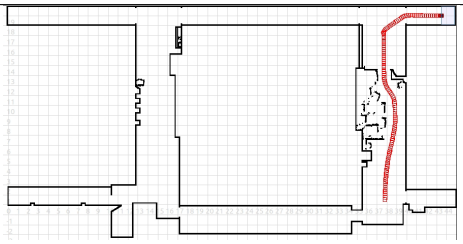
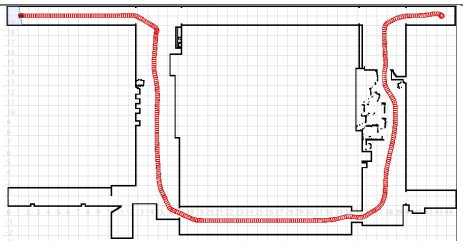
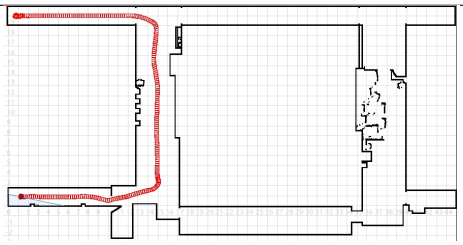
los caminos que no se han tomado anteriormente. La idea es optar a todos los caminos disponibles para así poder recorrer el entorno completo en menor tiempo.

El proceso de localización, en cambio, se inicia después de terminar el proceso de mapeado o cuando se comienza con un mapa previamente cargado. El objetivo es mantener al robot navegando por el entorno el tiempo requerido siendo capaz de saber cual es su posición actual en todo momento. La estrategia a la hora de decidir cual será el camino a tomar en cada cruce es idéntico al del proceso de mapeado. Originalmente, las decisiones de qué camino escoger cuando todas las opciones, o diferentes caminos, estaban tomadas se decidía de manera aleatoria dificultando la posterior comparación de diferentes pruebas. Por ello, actualmente se ha optado por seguir la estrategia del proceso de mapeado, es decir, tomar las decisiones en el mismo orden. Además, para que el proceso de mapeado y localización pudieran devolver patrones similares en sus resultados, ya que el ciclo de mapeado termina cuando se alcanzan n nodos del entorno y el ciclo de localización no acaba hasta que se decide parar la simulación, se ha optado por inicializar las variables que contienen la información de los caminos escogidos. De esta manera, el ciclo de localización vuelve a iniciarse cuando se alcanzan los n nodos y así el proceso de localización presenta una repetición cíclica con los mismos patrones que el proceso de mapeado.

Todas estas decisiones se han tomado para facilitar el análisis de las diferentes pruebas realizadas.

7.1.1. Exploración del entorno

En el capítulo 6 se ha analizado la estructura del entorno en el que se han realizado las pruebas del robot. Como ya se ha mencionado, se trata de la tercera planta de la Facultad de Informática. Un entorno de tipo oficina que el sistema de adquisición de mapa, en condiciones ideales sin error de odometría, modela con 38 nodos de los cuales 17 son pasillos, 8 vestíbulos y 13 cruces. En la tabla 7.1 se muestra un ciclo completo dividido en diferentes segmentos para facilitar la comprensión y que no haya confusiones en nodos visitados por segunda vez. La trayectoria del segmento termina donde se encuentra el robot. La figura 7.1 muestra la distribución exacta de cada nodo.

Trayectoria del robot	**Secuencia de nodos**
	El robot no comienza el proceso de mapeado hasta encontrar un callejón sin salida (*dead end*).
	P1 - B2 - H3 - P4 - H5 - B6 - P7 - B8 - H9 - P10 - H11 - B12 - P13
	P14 - B15 - H16 - P17 - H18 - B19 - P20 - B21 - H22 - P23 - H24 - B25 - P26 - B27
	H16 - P17 - H18 - B19 - P28
	P29 - B30 - H9 - P10 - H11 - B12 - P31 - B32 - P33 - B34 - H3 - P4 - H5 - B6 - P35
	P36 - B37 - H22 - P23 - H24 - B25 - P38

Cuadro 7.1: Trayectoria completa del robot en el ciclo de 38 nodos dividido en segmentos, donde P=pasillo, H=vestíbulo, B=cruce y el número que sigue define el identificador de nodo.

7.2. Experimentos en simulación

Se han llevado a cabo dos tipos de experimentos dependiendo del método de corrección del error de odometría. En un principio se han realizado pruebas relacionadas con una odometría ideal, basándose en el GPS para corregir el error. Después, para la obtención de unos resultados que se ciñen más a la realidad, se ha utilizado el método de corrección de odometría mediante láser (Laser corrected odometry, LODO). Al mencionar resultados más ceñidos a la realidad, se refiere a que los experimentos se llevarían a cabo en un entorno cerrado, de interior, donde las ventajas del sistema GPS estándar no aportan mucha ayuda. En este caso ambos métodos de corrección son simulados.

7.2.1. Odometría ideal (GPS)

Cuando el error de odometría es corregido mediante GPS se obtiene una odometría ideal. La posición de los nodos detectados por el robot coincide con su posición real. En la figura 7.1 se puede apreciar la distribución espacial de cada nodo del entorno que corresponde a los datos obtenidos en el proceso de mapeado.

En la estrategia de exploración original, la toma de decisiones en cuanto a qué cruce tomar cuando todas las opciones han sido visitadas, radica en la aleatoriedad y los resultados obtenidos en diferentes pruebas pueden no coincidir. La figura 7.2 muestra la activación de los diferentes nodos del entorno con una toma decisión de recorridos aleatorio.

La figura 7.3, en cambio, muestra la activación de cada nodo a lo largo del tiempo aplicando la nueva estrategia no aleatoria. La sección 7.1 resume las decisiones optadas para la facilitar el análisis de la localización en las pruebas de exploración a largo plazo. Dado que el robot repite la trayectoria, se repite la secuencia de nodos identificados. La activación de los primeros 38 nodos corresponde al proceso de mapeado, donde el robot identifica nuevos nodos durante la exploración (ver sección 7.1.1). Cuando se alcanzan todos los nodos del entorno (38 en el escenario de estas simulaciones) comienza el proceso de localización.

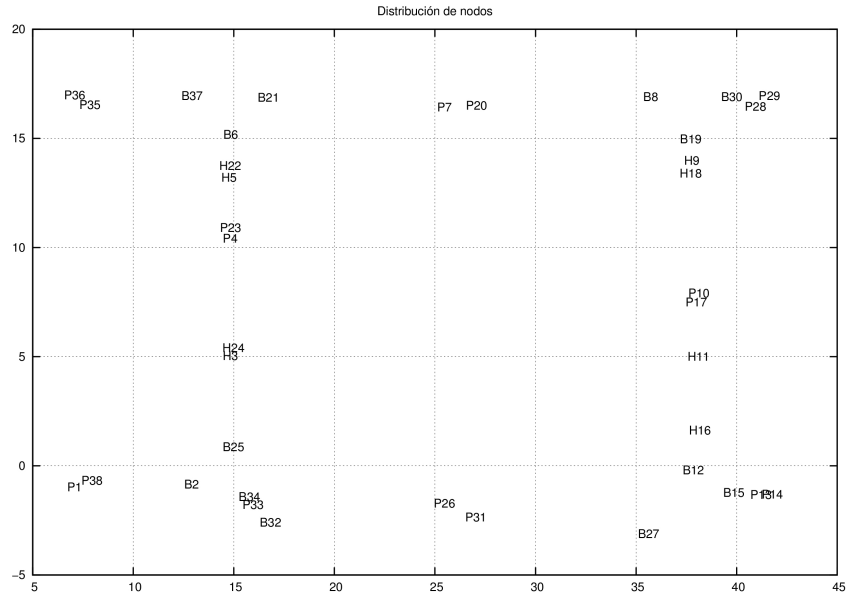


Figura 7.1: Distribución de los nodos en el entorno mediante corrección de odometría por GPS.

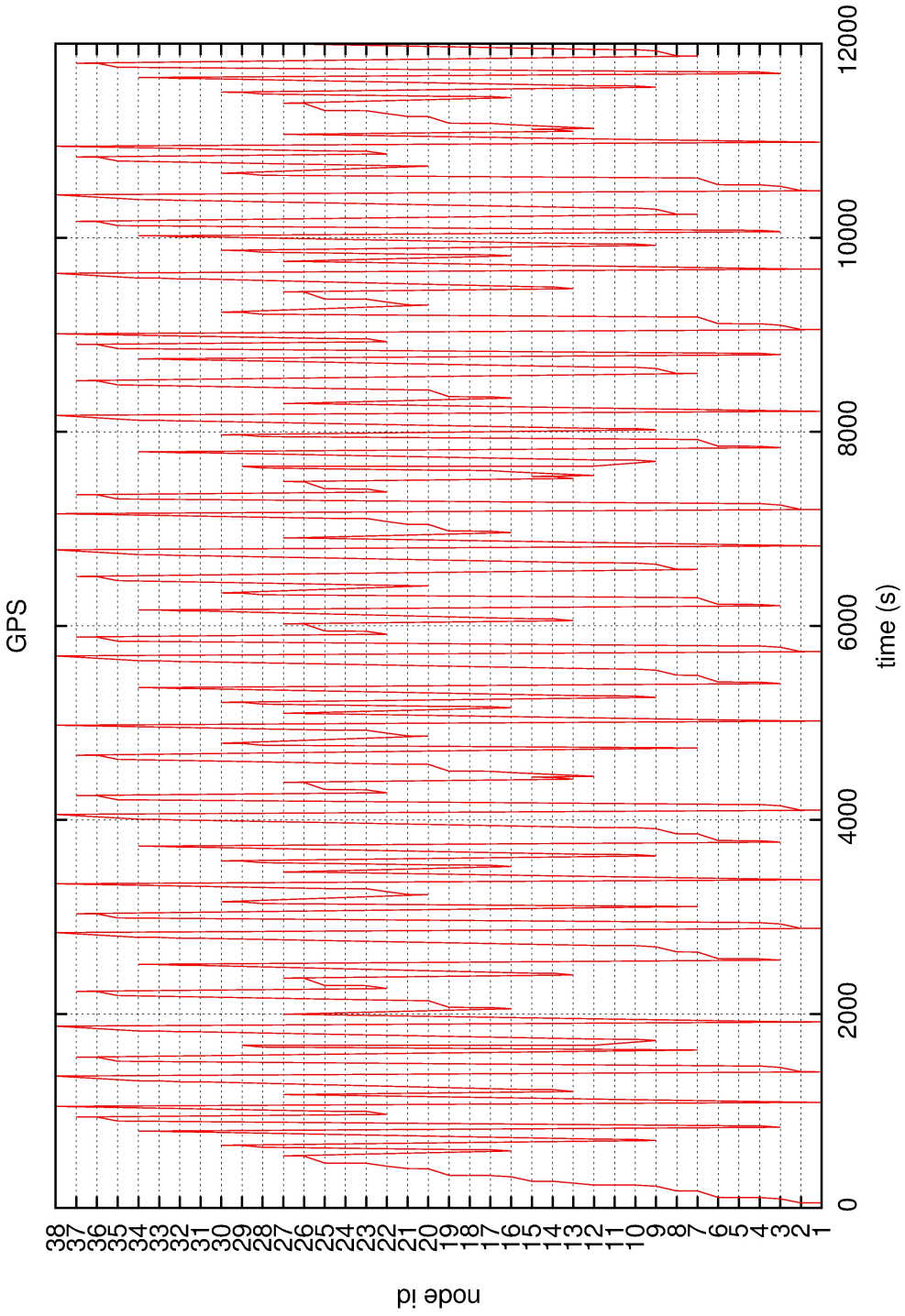


Figura 7.2: Activación de los nodos a lo largo del tiempo con corrección de odometría por GPS con estrategia de toma de decisión de cruce aleatoria.

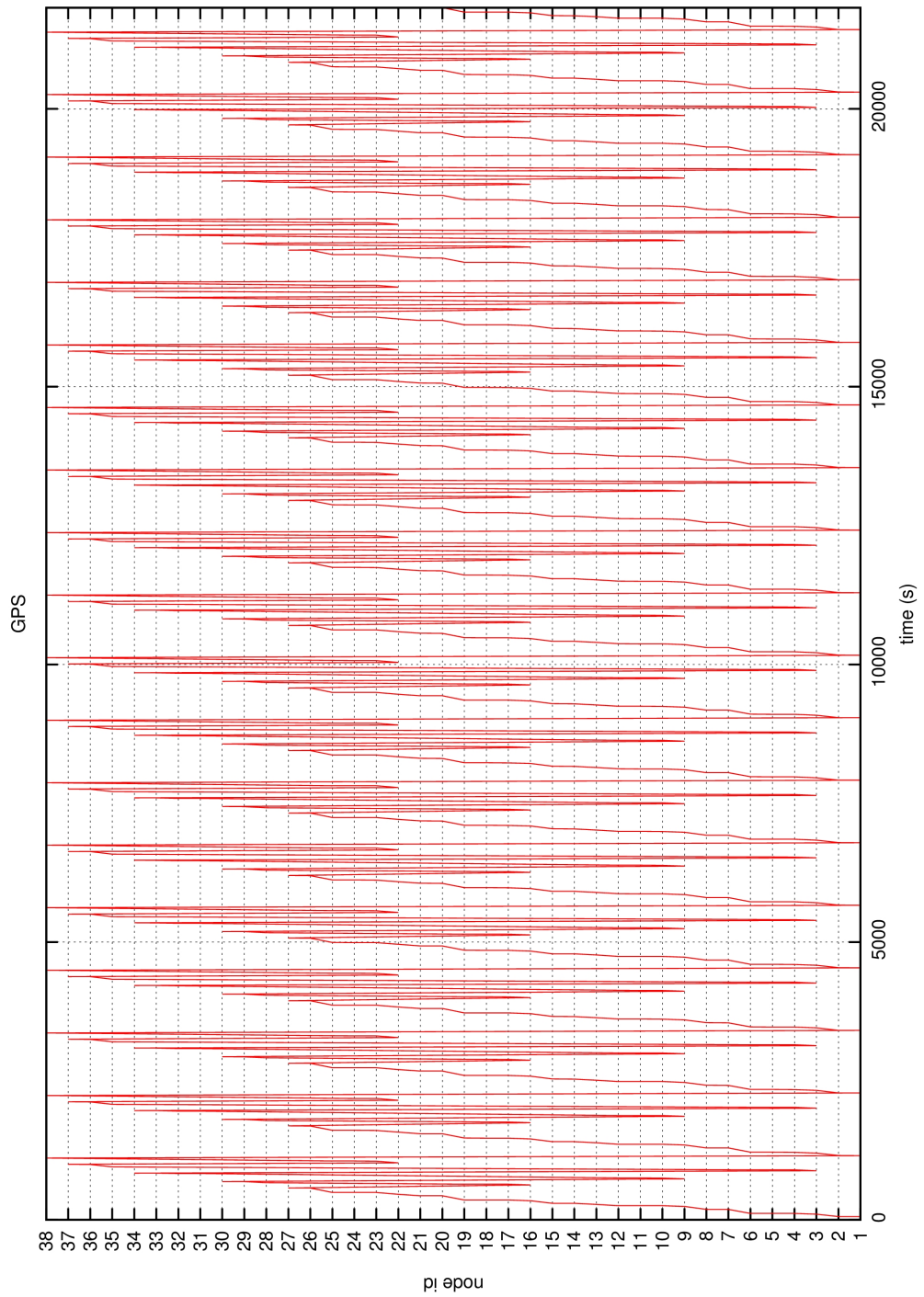


Figura 7.3: Activación de los nodos a lo largo del tiempo con corrección de odometría por GPS con estrategia de toma de decisión no aleatoria.

Al tratarse de una odometría ideal al plasmar las posiciones que recoge el robot a lo largo del tiempo en un gráfico, se obtiene como resultado posiciones superpuestas. En la figura 7.4 se muestra el camino recorrido por el robot durante un breve periodo de tiempo. Se aprecia con facilidad cómo los pasillos y vestíbulos se encuentran en las mismas posiciones. En el paso de los cruces en cambio, puede diferir ya que depende de cómo salga orientado del anterior nodo.

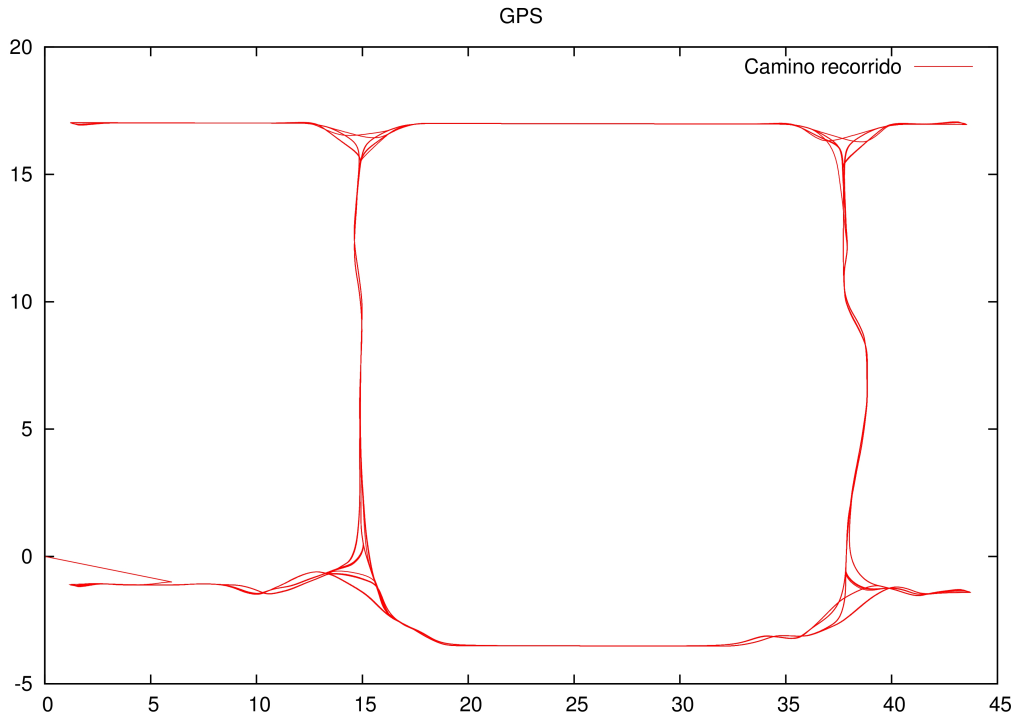


Figura 7.4: Camino recorrido por el robot a lo largo del tiempo con corrección de odometría por GPS.

7.2.2. Odometría corregida mediante láser (LODO)

La estimación de las posiciones se hace a través del driver LODO esta vez. Este controlador tiene como entradas los datos de odometría sin tratar y los valores recogidos por el láser. Las salidas son las posiciones estimadas corregidas y escaneos de láser sincronizados. Este método de corrección mejora las estimaciones de odometría pero a la larga también va acumulando error.

En la figura 7.5 se puede apreciar la localización de cada nodo del entorno que corresponde a los datos obtenidos en el proceso de mapeado.

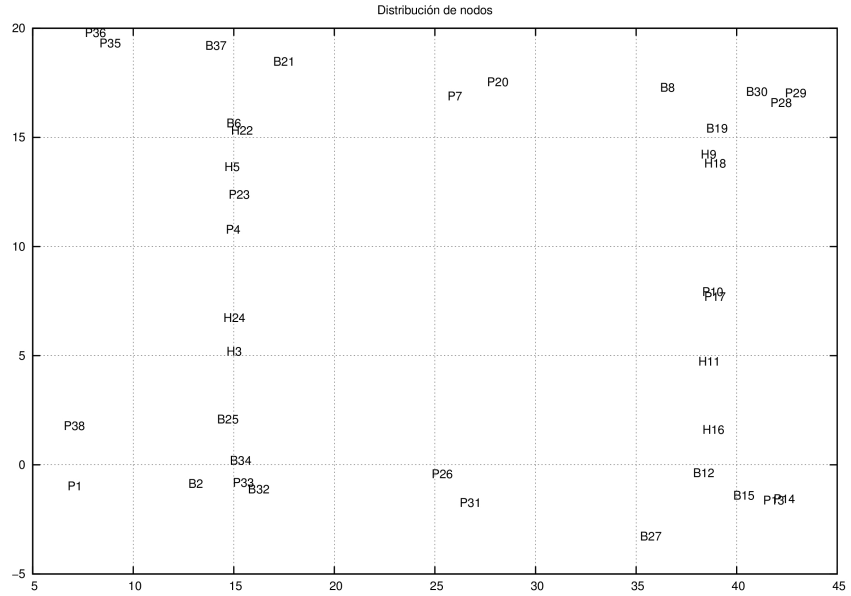


Figura 7.5: Distribución de los nodos en el entorno mediante corrección de odometría mediante LODO.

Las pruebas realizadas con LODO no han sido de la misma duración que las pruebas con GPS, ya que el error en este caso se va acumulando y al final no deja ver con claridad qué resultados ofrece el proceso de localización. Por ello, las pruebas realizadas se han detenido en 5000 segundos, momento en el que se aprecia que el error acumulado de odometría no permitía localizar adecuadamente los nodos correspondientes del entorno. En la figura 7.6 se puede ver la activación de los nodos comenzando con el proceso de mapeado y continuando con la localización. Alrededor del segundo 3000 (simbolizado con un punto negro), el robot comienza a confundir las localizaciones durante la exploración.

En la figura 7.7, en cambio, se muestra la activación de los nodos del entorno que corresponde a una prueba realizada cargando previamente el mapa obtenido gracias al proceso de mapeado. En este caso, se completan con éxito 3 ciclos de los 38 nodos correspondientes y el robot localiza adecuadamente hasta aproximadamente al segundo 4200 (simbolizado con un punto negro) cuando comienza a identificar incorrectamente las localizaciones.

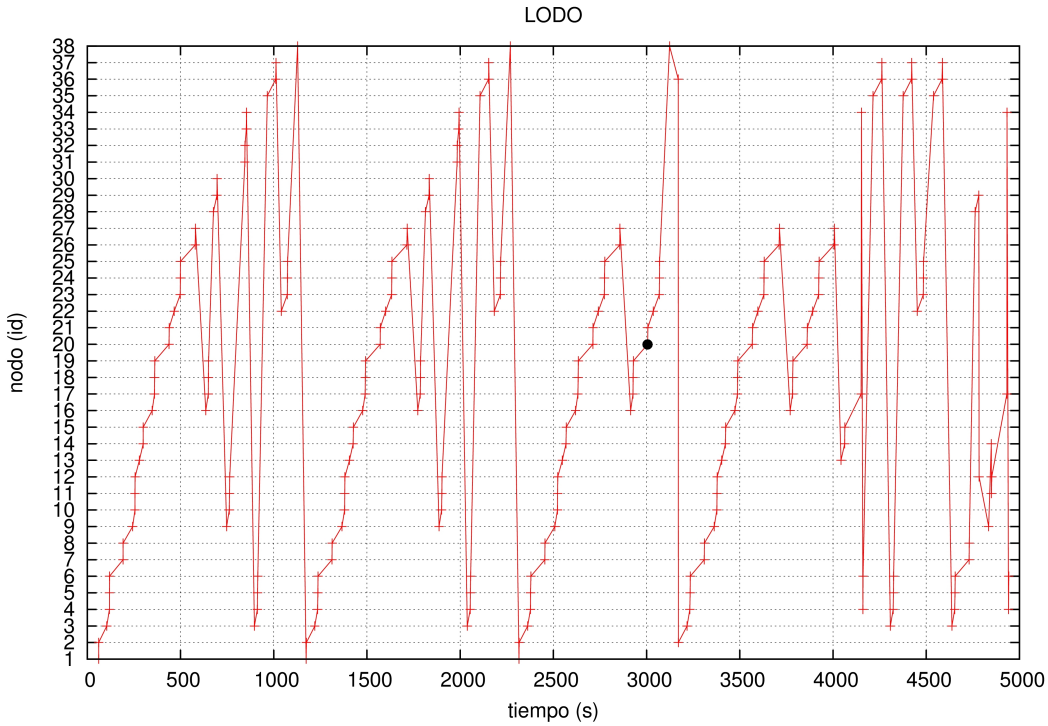


Figura 7.6: Activación de los nodos a lo largo del tiempo con corrección de odometría por LODO.

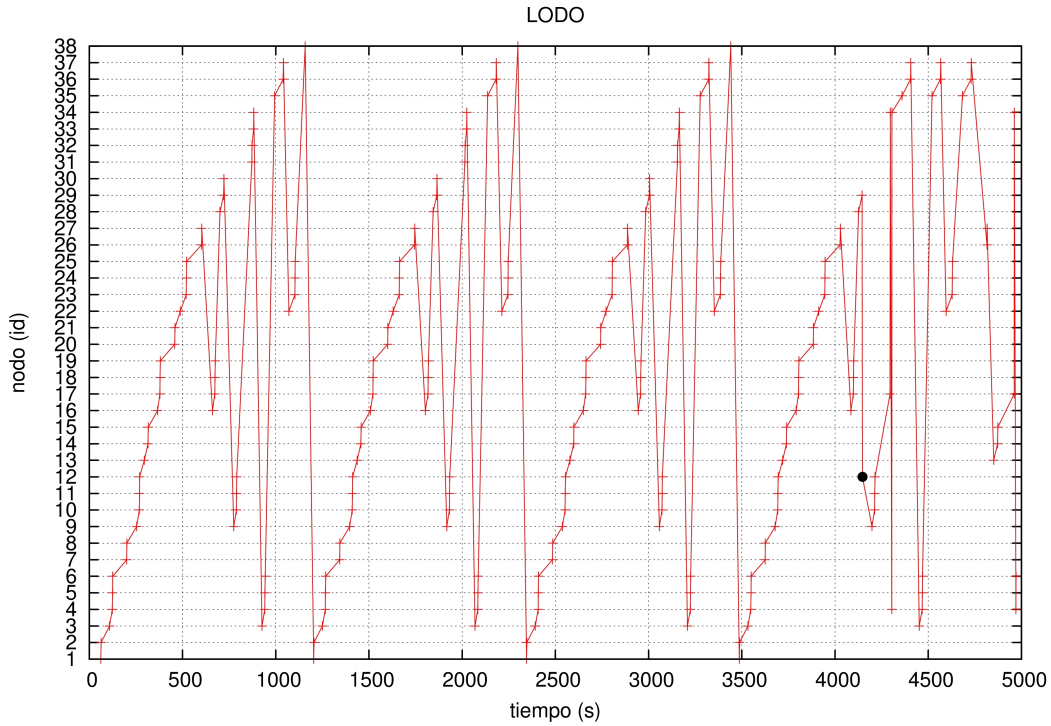


Figura 7.7: Activación de los nodos a lo largo del tiempo con corrección de odometría por LODO sin proceso de mapeado.

El error resultante después de la corrección de LODO produce una rotación en la trayectoria del robot (ver figura 7.8).

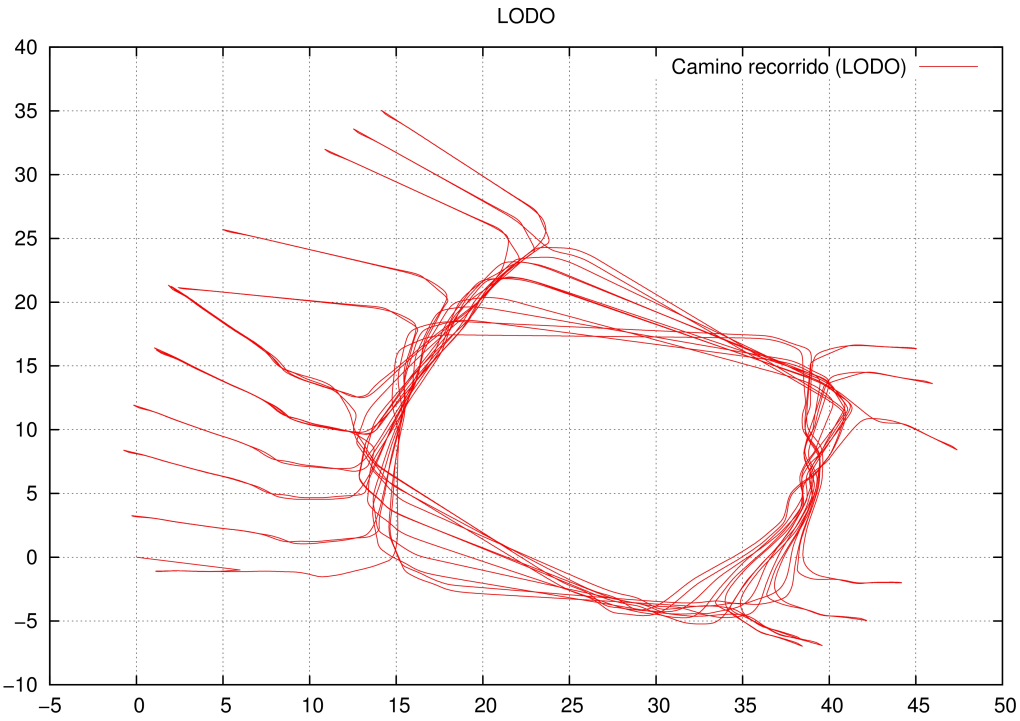


Figura 7.8: Camino recorrido por el robot a lo largo del tiempo con corrección de odometría por LODO.

7.3. Carga automática del mapa del entorno

A la hora de desarrollar cualquier aplicación o nuevo sistema, las pruebas que verifican su validez muchas veces suelen ejecutarse sobre el mismo escenario. Por ello, en muchos casos ayuda, o es conveniente, tener un proceso que simplifique la realización de pruebas. En este caso, el entorno por el que navega el robot durante la exploración es siempre el mismo y dado que se quiere comprobar la validez del sistema basado en INCA para la localización, es una buena ayuda disponer de un proceso para cargar automáticamente el modelo del entorno.

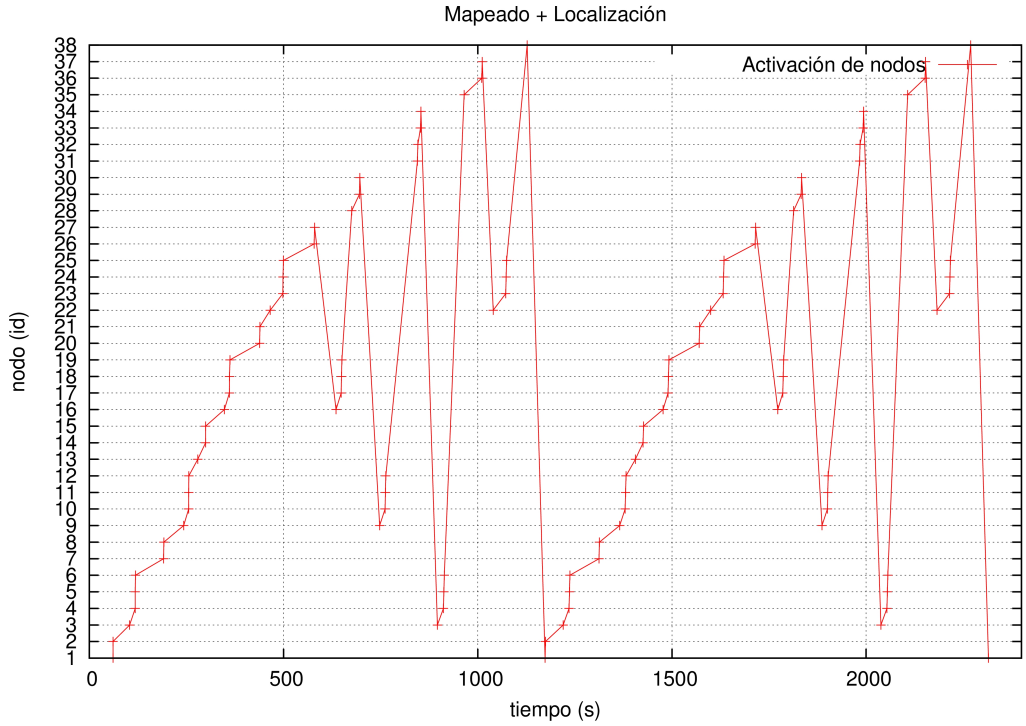


Figura 7.9: Activación de nodos con los procesos de mapeado y localización.

Originalmente, la carga de los nodos se ejecuta en función a los diferentes tipos de nodo que existen comenzando primero con los pasillos con orientación de 0°, 90°, 180°y 270°sucesivamente. De la misma manera se hace con vestíbulos y por último, con los cruces, siempre de menor a mayor grados en cuanto a orientación. Todo ello, utilizando los mismos ficheros que utilizan los métodos estadísticos de INCA para identificar si se corresponde a una nuevo localización o es una anteriormente visitada. Se puede deducir que el orden de carga de los diferentes nodos no corresponde al orden en el que el robot identifica los nodos durante el proceso de mapeado. Este detalle puede llevar a una confusión si se comparan datos de pruebas con diferentes simulaciones como pueden ser mapeado + localización y carga de mapa + localización. Aunque los nodos sean los mismos los identificadores no lo son, la numeración en el mapa topológico es distinta. En la figura 7.9 se pueden apreciar los resultados obtenidos con los procesos de mapeado + localización, un ciclo por cada proceso, donde el robot comienza una exploración con total desconocimiento sobre el entorno. Partiendo de estos datos recogidos en el proceso de mapeado, al realizar una simulación con una carga automática del mapa y posterior proceso de localiza-

ción se obtienen los resultados de la figura 7.10. El resultado se muestra con dos ciclos de localización. Las diferencias en ambos resultados son evidentes y se pueden malinterpretar unos resultados correctos por el hecho de representarlos en un orden distinto.

Con el objetivo de evitar confusiones derivadas de una representación diferente de las pruebas, se ha optado por actualizar el método de carga de mapa en memoria para que la numeración de nodos obtenida corresponda con en el proceso de mapeado. De esta manera, los posteriores ciclos de localización mostrarán una orden comparable fácilmente con el proceso de mapeado siempre que se utilice un método de corrección de odometría ideal o el error de odometría no se acumule demasiado. En la figura 7.11 se muestra los resultados del nuevo método de carga de mapa, donde se realizan dos ciclos de localización. Al comparar con los resultados de la figura 7.9 se comprueba que los nodos son identificados en el mismo orden.

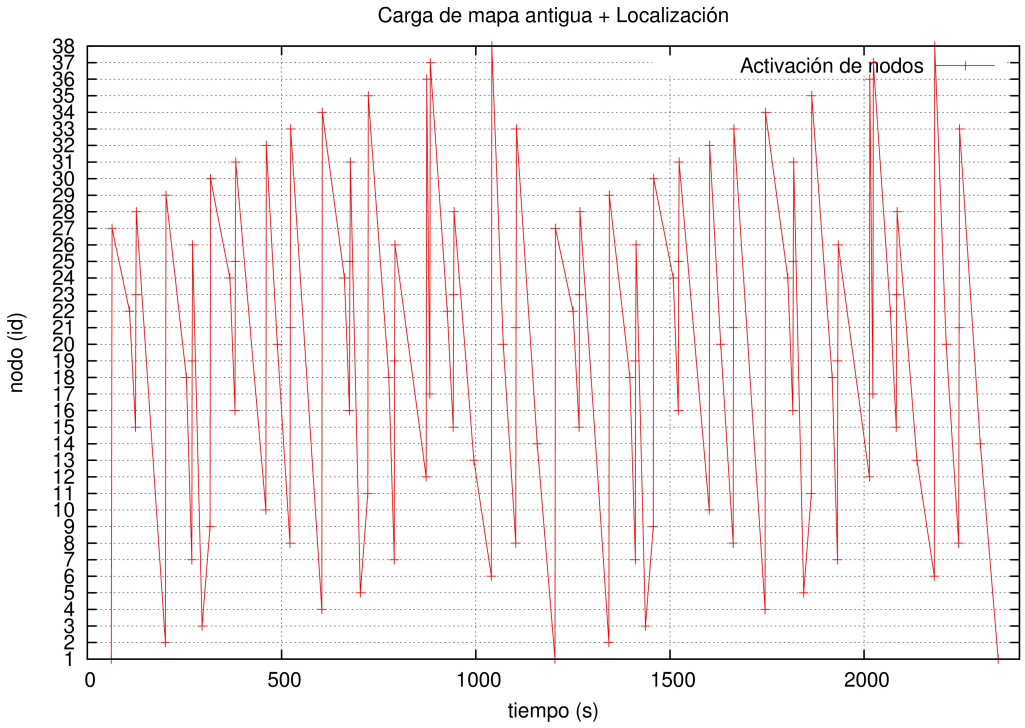


Figura 7.10: Activación de nodos con los procesos de carga de mapa original y localización.

Al comparar las figuras 7.10 y 7.11 se aprecian las diferencias que se obtienen al cargar los mismos datos recogidos del proceso de mapeado. Los nuevos resultados se asemejan, de manera adecuada, a los obtenidos cuando el robot no tiene conocimiento previo de los diferentes nodos del entorno.

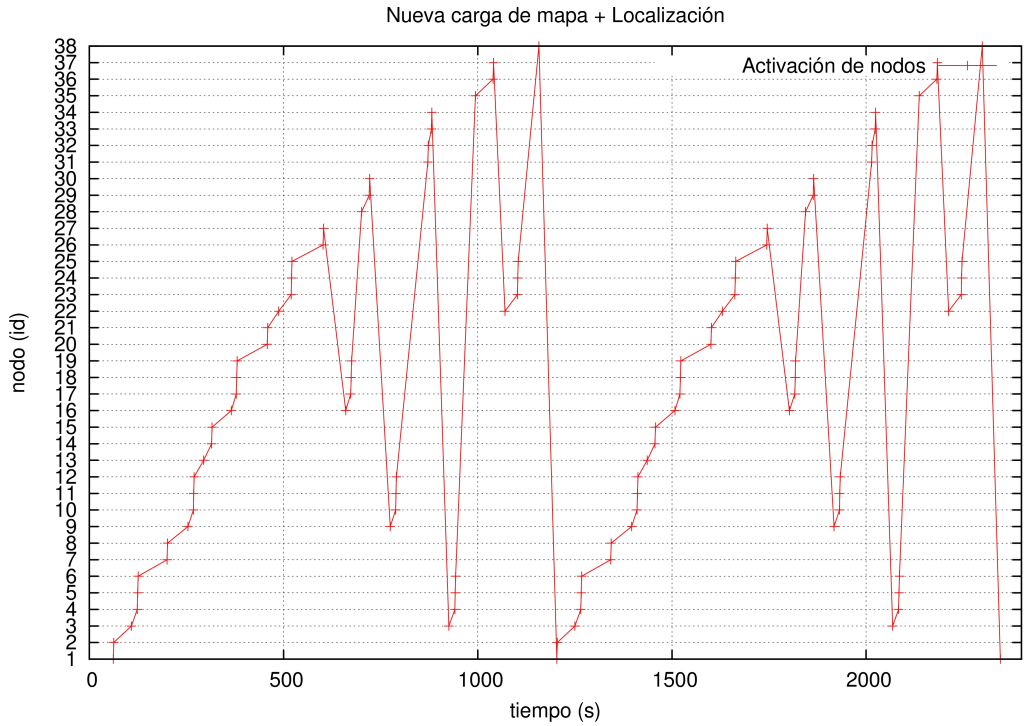


Figura 7.11: Activación de nodos con los procesos carga de mapa actualizada y localización.

7.4. Valoración de resultados

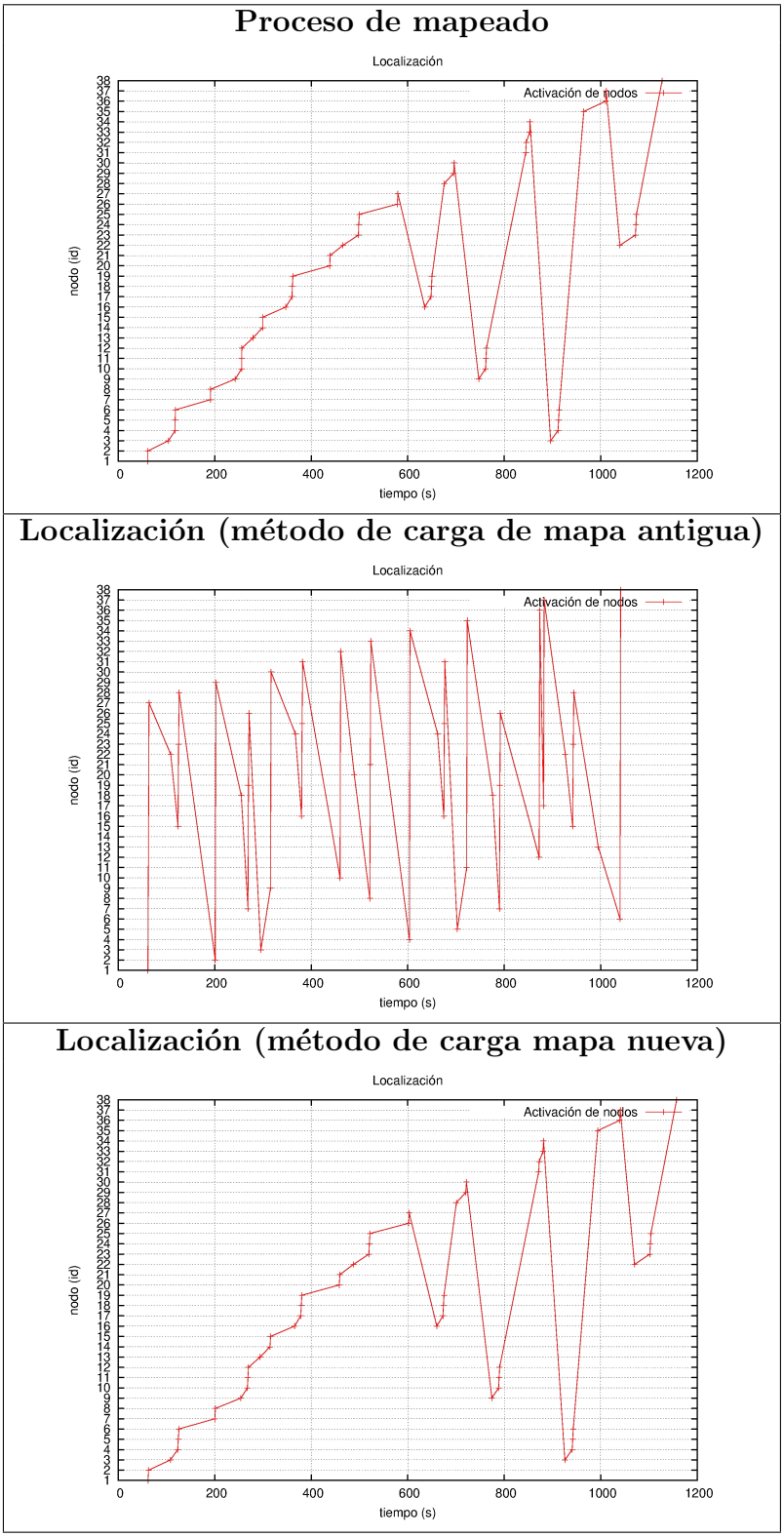
Las pruebas realizadas tanto con GPS o LODO nuevamente confirman que INCA puede utilizarse tanto para mapeado del entorno como para la localización. En la odometría ideal conseguida gracias a la corrección de GPS, todos los nodos visitados a lo largo del tiempo han sido correctamente clasificados y ya que después de visitar los 38 nodos correspondientes del entorno se repite el ciclo de exploración. En la figura 7.3 se aprecia visualmente que el recorrido del robot carece de errores y siempre realiza el mismo recorrido en cada ciclo de 38

nodos. La antigua estrategia para la toma de decisiones, en cambio, dificultaba la apreciación de la validez del sistema a simple vista (figura 7.2), por ello, los nuevos resultados obtenidos son satisfactorios.

A diferencia del método de corrección de GPS, en el caso de la corrección mediante LODO, el error se va acumulando poco a poco en las coordenadas x, y y θ. Mientras el error se mantiene en un ángulo de rotación relativamente pequeño, el proceso de localización se lleva a cabo correctamente. Cuando se alcanza un nivel de rotación suficientemente grande para que INCA no pueda definir el nodo como visitado, comienza a aparecer nuevos nodos que en realidad corresponden a nodos ya conocidos. La figura 7.7 muestra una prueba simulada con corrección de odometría mediante LODO donde se aprecia cómo con el tiempo la identificación de los diferentes nodos no se efectúa correctamente. La simulación mostrada finaliza antes de que INCA empiece a detectar nuevos nodos y empezar a empeorar la identificación de los nodos existentes.

Al igual que en los experimentos realizados en *Advances towards behaviour-based indoor robotic exploration* [JI11] se comprueba la validez de un sistema de mapeado y localización basado en INCA como sistema basado en comportamientos. Todo ello con los correspondientes cambios en la toma de decisión sobre la elección de las posibles opciones de los cruces, automatizando para mostrar resultados más intuitivos.

Del mismo modo, también se confirma que el método de adquisición de mapas funciona correctamente y los resultados obtenidos con la carga de mapa y el proceso de localización son los esperados. La tabla 7.2 resume las diferencias que se encuentran en el proceso de mapeado y el resultado de localización después de la carga de de mapa en la memoria, tanto antigua como nueva.



Cuadro 7.2: Diferencias entre el proceso de mapeado y localización una vez cargado el mapa.

Parte IV

CONCLUSIONES

Capítulo 8

CONCLUSIONES

Un robot debe ser capaz de adaptarse al entorno y lograr una integración natural. Para ello se emplean diferentes tipos de arquitecturas de control, donde el principal objetivo de las arquitecturas de control es conseguir que un robot se comporte de un modo inteligente. Existen diferentes tipos de arquitecturas de control pero analizando los problemas al que se tiene que enfrentar un robot como puede ser un entorno dinámico y poco estructurado, el enfoque basado en comportamientos parece el más adecuado. Este enfoque para la navegación robótica se basa en la idea de que el problema de control se resuelve mejor con un diseño de enfoque ascendente y con la suma incremental de procesos simples, llamados comportamientos. Diferentes resultados en este campo muestran favorables resultados para seguir esta línea de desarrollo.

En cuanto a la localización, la principal fuente de error en los sistemas de localización odométrica deriva del modelado de los parámetros que describen la cinemática del robot móvil. Estos valores odométricos están basados en codificadores rotativos acoplados a las ruedas motrices. El error no surge de la identificación de los valores iniciales sino que surgen con los cambios a lo largo del tiempo. Los avances en el hardware de los sistemas de localización odométrica no son suficiente para la conseguir la solución al problema de los errores de localización. Por ello, se recurre a diferentes métodos de estimación de localización como los métodos probabilísticos. En el caso concreto de este trabajo se utilizan los métodos estadísticos de INCA como métodos de estimación de la localización, los cuales presentan resultados favorables en el caso de que el método de corrección odométrica sea precisa a lo largo del tiempo.

Respecto a la automatización de los recorridos del robot, la odometría ideal (corrección mediante GPS) muestra que a lo largo del tiempo el robot realiza los mismos ciclos dejando de lado la aleatoriedad presentada inicialmente. Las pruebas simuladas en la corrección de odometría mediante LODO, se puede llegar a la misma conclusión que en las pruebas con GPS, pero esta vez al acumular el error a lo largo del tiempo las muestras de la automatización pueden no ser tan perceptibles a simple vista. Además, el nuevo método creado para cargar el mapa en la memoria del robot facilita la valoración de los diferentes resultados simulados tanto con la utilización del proceso de mapeado o directamente ejecutando el proceso de localización previa carga de mapa.

8.1. Trabajo futuro

En el capítulo 7 se han realizado diversas pruebas mostrando los resultados de los diferentes cambios realizados al sistema presentado en el sistema base utilizado en el proyecto [JI11], concretamente en el apartado de mapeado y localización. Todas las pruebas muestran resultados favorables, cumpliendo con los objetivos fijados. Como bien se ha podido comprobar todas las pruebas han sido realizados en el simulador Stage. Por ello, como trabajo futuro se propone extender las pruebas realizadas en un entorno real y con un robot real para así poder contrastar y validar los resultados.

El próximos paso después de comprobar la validez de la automatización de los recorridos en un entorno real, sería integrar un sistema de planificación de caminos. Dotar el robot con la inteligencia para decidir cual es la mejor ruta para llegar a una localización concreta, con diferentes métodos de búsqueda.

Referencias

[ALR+03] A. Astigarraga, E. Lazkano, I. Ranó, B. Sierra, and I. Zarautz. Sorgin: a software framework for behavior control implementation. *CSCS14*, 1:243–248, 2003.

[Ark98] Ronald C Arkin. *Behavior-based robotics*. MIT press, 1998.

[BEF96] J. Borenstein, HR Everett, and L. Feng. Where am i? sensors and methods for mobile robot positioning. *University of Michigan*, 119:120, 1996.

[Bre04] C. Breazeal. Social interactions in hri: The robot view. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 34:181–186, 2004.

[Bro99] Rodney A Brooks. *Cambrian intelligence: the early history of the new AI*. Mit Press, 1999.

[BS99] C. Breazeal and B. Scassellati. A context-dependent attention system for a social robot. *rn*, 255:3, 1999.

[dLMZA] J de Lope, D Maravall, JG Zato, and Dpto de Sistemas Inteligentes Aplicados. Estrategias de exploración para la construcción autónoma de modelos topológicos en robots móviles. In *Conf. de la Asociacion Espanola de Inteligencia Artificial, CAEPIA*, volume 99, pages 44–51.

[FM00] M.O. Franz and H.A. Mallot. Biomimetic robot navigation. *Robotics and autonomous Systems*, 30(1):133–153, 2000.

[IA08] I. Irigoien and C. Arenas. Inca: new statistic for estimating the number of clusters and identifying atypical units. *Statistics in medicine*, 27(15):2948–2973, 2008.

[Jac72] W. Jacobs. Control systems in robots. In *ACMâ72: Proceedings of the ACM annual conference. ACM Press. New York, NY, USA.*, pages 110–117., 1972.

[JB96] Javier González Jiménez and Anibal Ollero Baturone. Estimación de la posición de un robot móvil. *Automática*, (29):3–18, 1996.

[JI11] E. Jauregui Izturta. *Advances towards behaviour-based indoor robotic exploration.* PhD thesis, Universidad del País Vasco, 2011.

[Mat92] M.J. Mataric. Behavior-based control: Main properties and implications. In *Proceedings, IEEE International Conference on Robotics and Automation, Workshop on Architectures for Intelligent Control Systems*, pages 46–54. Citeseer, 1992.

[Mat01] Maja J Matarić. Learning in behavior-based multi-robot systems: Policies, models, and other agents. *Cognitive Systems Research*, 2(1):81–93, 2001.

[MI02] R.I. Madrigal and E.V. Idiarte. *Robots industriales manipuladores*, volume 130. Edicions UPC, 2002.

[Mur00] R. Murphy. *Introduction to AI robotics.* The MIT Press, 2000.

[Neh03] Ulrich Nehmzow. *Mobile robotics: a practical introduction.* Springer, 2003.

[Nil84] N.J. Nilsson. Shakey the robot. Technical report, DTIC Document, 1984.

[OSS+07] R Silva Ortigoza, JR García Sánchez, VR Barrientos Sotelo, MA Molina, VM Hernández Guzmán, and G Silva Ortigoza. Una panorámica de los robots móviles. *Télématique*, (003):1–14, 2007.

[RAE01] RAE. Diccionario de la lengua española (22.a ed.), 2001.

[RMDC11] Cesar Augusto Romero Molano and César Augusto Díaz Celis. Navegación de robot móvil usando kinect y opencv. In *Congreso Internacional de Ingeniería Mecatrónica-UNAB*, volume 2, 2011.

[SN04] Roland Siegwart and Illah R Nourbakhsh. *Introduction to autonomous mobile robots.* MIT press, 2004.

[Thr02] S. Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.