INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

Construcción de Mapas y Localización Simultánea con Robots Móviles

Autor:

Víctor Manuel Jáquez Leal

Sometido al Programa de Graduados en Informática y Computación en cumplimiento parcial con los requerimientos para obtener el grado de:

Maestro en Ciencias de la Computación

Asesor:

Dr. Eduardo Morales Manzanares

Cuernavaca, Morelos., Octubre 2005

Construcción de Mapas y Localización Simultánea con Robots Móviles

Presentada por:

Víctor Manuel Jáquez Leal

Aprobada por:

Dr. Eduardo Morales Manzanares

Profesor-Investigador del Departamento de Computación

ITESM Campus Cuernavaca

Asesor de Tesis

Dr. Luis Enrique Sucar Succar

Profesor-Investigador del Departamento de Computación

ITESM Campus Ciudad de México

Sinodal

Dr. Leonardo Romero Muñoz

Profesor-Investigador de la Facultad de Ingeniería Eléctrica

Universidad Michoacana de San Nicolás de Hidalgo

Sinodal

Agradecimientos

A mi padres Rebeca y Carlos. A mi hermano Ricardo.
A mis amigos de toda la vida: Noberto, Chuy, Julio, Tomás y Ulises.
No hubera sido nada sin ustedes.

A la gente de Cuernavaca, en especial a Jacqueline, por ayudarme a hacer la vida más llevadera; a Héctor por guiarme en el intrincando mundo de la investigación; a Sergio por las discusiones técnicas; a Verónica por las charlas personales.

A mis asesores y maestros, con especial énfasis a Eduardo Morales y Luis Enrique Sucar.
Al ITESM Campus Cuernavaca y la Cátedra de Robótica Móvil por permitirme jugar con los robots.

Y tanta otra gente. No quiero dejar de mencionar a Aurora, Chapis, Mónica, Vanesa, Blanca, Alberto, José Luis, Víctor Hugo, Adriana y a todos los que han estado conmigo a lo largo de los años. Estoy muy agradecido con todos.

Resumen

En este trabajo de tesis se hace un análisis del problema de la construcción de mapas y localización simultánea con robots móviles, dentro de ambientes interiores. Se modela el problema como una red bayesiana dinámica y se analizan los modelos gráficos probabilistas capaces de resolver el problema. Con el fin de obtener inferencias de esta red bayesiana dinámica, se estudia el algoritmo de Filtros de Partículas, el cual es el método actualmente utilizado en la comunidad de la robótica móvil para obtener un mapa del ambiente haciendo hipótesis de la posición del robot. La representación del ambiente de tipo de rejilla de ocupación.

Las principales aportaciones hechas en este trabajo apuntan en tres direcciones: la primera es una arquitectura de software modular y con procesamiento en línea, que toma ventaja de algunas características del funcionamiento de los robots móviles, tal como la asincronía en la recepción de los datos de los diferentes sensores del robot y disponibilidad del procesamiento concurrente en diferentes hilos. En segundo lugar una estrategia de exploración del ambiente que intente obtener una representación del mismo de manera autónoma, que permita el movimiento continuo del robot, basado en una navegación local. Por último, un modo de añadir al mapa, construido principalmente con el telémetro láser, las mediciones percibidas por el sonar para agregar más elementos de ayuda en la navegación segura del robot en el ambiente.

Se muestra cómo el modelo propuesto es capaz de construir mapas de ambientes de interiores, tipo oficina, con buena precisión. La arquitectura propuesta es útil como marco de referencia para posteriores desarrollos, que por su naturaleza modular puede ser modificado fácilmente. En presencia de obstáculos transparentes, como paredes y puertas de vidrio, la fusión sensorial ayudó a corregir los errores en el mapa, ofreciendo información más útil al momento de la planeación de trayectorias, dando también la capacidad de generar distintos mapas según el sensor. Finalmente la exploración autónoma, basada en información local del espacio de configuraciones, navegando a la región visible más lejana dentro de una política ϵ-greedy, capaz de navegar por completo el ambiente, recorriendo zonas ya exploradas para facilitar la localización, y navegando a zonas desconocidas. Este esquema es completo pero no eficiente en tiempo, además a pesar de que funcionó bien en simulación, pero no tuvo los mismos resultados en ambientes reales.

Se realizaron varios experimentos, tanto en ambientes reales como en simulados. En los ambientes simulados se probó mayormente el algoritmo de exploración, mientras que en los reales se trabajó más con la construcción del mapa guiando el robot con la ayuda de una palanca de comandos virtual. Los ambientes reales fueron de interiores dentro del Campus, con bajo control del dinamismo interior (hubo personas moviéndose en ocasiones), probando la robustez de los algoritmos para modelar los sensores. Los ambientes simulados fueron dentro de *Stage* con diversos ambientes.

Índice general

Índice de Figuras

Índice de Tablas

Índice de Algoritmos

Capítulo 1

Introducción

En este capítulo se presenta una introducción al tema de la robótica móvil y la importancia de su estudio en la actualidad. Se continúa con una descripción inicial del problema a resolver, junto con las razones que motivaron a la presente tesis. Se justifica el trabajo con una breve descripción del problema a solucionar, sus dificultades y avances teóricos y prácticos.

1.1. Antecedentes

Los mecánicos fueron los pioneros creando máquinas capaces de tener movimientos repetidos, elegantes, controlados, calculados y predecibles. Luego llegaron los electrónicos, quienes dieron a esos movimientos un significado de automatización, ahora las máquinas podían interactuar en ambientes más complejos, recibiendo retroalimentación, manipulando más variables. Finalmente llega la gente de la computación, tratando de dotar de cierto grado de inteligencia a esas máquinas; en una palabra, autonomía. Se intenta pasar de la automatización y teleoperación a la autonomía [Nehmzow, 2003].

La robótica móvil estudia a los robots que realizan tareas desplazándose de forma autónoma en ambientes dinámicos y complejos, desde espacios interiores como oficinas y casas, exteriores como espacios terrestres, submarinos, aéreos, y, en últimos años, hasta extraterrestres, como en el planeta marte. Estas habilidades llevan implícita la capacidad de razonar sobre una representación interna del mundo.

El problema más abarcado en la literatura de la robótica móvil, y el enfoque de este trabajo, son los robots que interactúan en espacios interiores, pensados para interactuar en ambientes dinámicos, haciendo tareas de servicio para los humanos: entrega de mensajes o paquetería, guía de turistas, recolector de basura, vigilante, etcétera. También en trabajos

demasiado peligrosos que pongan en riesgo la vida humana, en ambientes inestables como basureros de desechos tóxicos, o de acceso difícil.

El paso entre un robot que haga tareas eficientemente y un robot que sólo puede moverse y percibir su ambiente es grande y no trivial, ya que implica el razonamiento sobre un ambiente para realizar una planeación inteligente de sus movimientos [Dudek and Jenkin, 2000]. Para poder realizar este razonamiento de su espacio de operación, el robot debe tener una representación del mismo, es decir un mapa de su ambiente. Con el mapa del ambiente, el robot podrá realizar tareas de localización y planeación de movimientos. La localización ofrecerá el punto de partida, la tarea a realizar será su meta y el cálculo de los movimientos necesarios se hará a partir de las restricciones expuestas en el mapa.

Inicialmente los investigadores daban al robot una representación del ambiente. Estos mapas eran hechos de manera externa al robot y por lo mismo eran difíciles de empatar con la perspectiva del robot. Nuestra percepción es distinta a la del robot. Por lo que la tendencia es que el robot genere su propia representación del ambiente, utilizando sus mismos sensores, tomando en cuenta sus limitaciones intrínsecas y alcances. Además, si nuestro fin es la autonomía, la habilidad de un robot para construir mapas confiables es un requisito clave [Montemerlo *et al.*, 2002a].

Antes de entrar en materia, es importante declarar la diferencia entre cartografía y exploración. Este trabajo usa la palabra cartografía para la traducción de la palabra inglesa *mapping*, que representa la labor de generar un mapa del ambiente a partir de la información disponible. No le interesa cómo ésta se recabó, o cuánto tiempo tardó en obtenerse. En cambio la exploración es la tarea de recorrer un ambiente desconocido con el fin de obtener los datos necesarios para su cartografía. Se distinguen estas dos palabras porque la investigación sobre el tema ha separado estos problemas.

1.2. Justificación

El problema de la cartografía y localización simultanea, mejor conocido en la literatura como SLAM debido a sus siglas en inglés (*Simultaneous Localization and Mapping*), es un tema con bastante trabajo realizado [Thrun, 2002b], sin embargo aún no se llega a un algoritmo que sea aceptado ampliamente por la comunidad científica, aunque las soluciones con enfoques probabilistas son las más utilizadas.

El objetivo del problema de SLAM es crear una representación del ambiente donde trabajará

el robot, en otras palabras, un mapa. Para hacer esto de manera autónoma, el robot se vale de sus sensores para percibir el mundo. Una vez que el robot se encuentra en un punto y percibe su ambiente, el robot debe cambiar su ubicación para descubrir el entorno observable desde el nuevo punto y finalmente integrar ambas representaciones en un mapa general. A esta forma de generar las representaciones del ambiente se les llama incrementales, ya que el procesamiento del mapa ocurre cada vez que se tiene un nuevo punto de vista del ambiente.

Sin embargo, se presenta el problema de que la información de los sensores contiene mucho ruido que conduce a errores e inexactitudes en las mediciones. Esto es más notorio al tratarse de la odometría, ya que el error odométrico aumenta con el tiempo y la distancia recorrida, por lo que sus lecturas no son confiables. Entonces, si se desea integrar dos representaciones vistas por el robot en tiempos distintos, se debe conocer exactamente la posición del robot donde hizo las mediciones. No obstante, el trabajo de localización requiere un mapa previo para relacionar las mediciones con la representación. Una suposición base para este problema es que el robot no cuenta con sensores de localización global como un *GPS*, que no funciona en ambientes interiores, ni tiene la precisión suficiente, sino que se intentará localizar usando únicamente sus sensores de distancia y odometría.

Por estas razones el problema obliga resolver simultáneamente el cálculo de la posición del robot, ya que esta no se puede conocer de manera exacta con simples mediciones. A la vez que el robot explora su ambiente debe localizarse. Nuevas técnicas de localización y cartografía utilizan filtros bayesianos, echando mano de métodos de inferencia como los filtros Kalman y hoy en día los filtros de partículas [Thrun, 2002b; Thrun, 2002a].

Pero a pesar de todo el trabajo teórico realizado, actualmente existen pocos sistemas de software dedicados al problema. En los comienzos de la programación de robots móviles, el software eran soluciones parciales, restringidas en su alcance y limitados en el hardware soportado. Proyectos como, por ejemplo, *Carmen* [Montemerlo *et al.*, 2002b] o *Scene* [Davison, 2001], orientados casi exclusivamente a la cartografía, están elaborados para trabajar en equipos específicos, con sensores concretos, lo que limita su alcance y capacidad de extensión. Actualmente se realizan esfuerzos para homogeneizar el control de distintos tipos robots móviles con proyectos como *Player/Stage* [Gerkey *et al.*, 2004] o *DRDOS* [Austin and Overett, 2004]. A partir de estos esfuerzos se han comenzado a trabajar en sistemas que realizan tareas más complejas, reutilizando mucho del trabajo anterior. Ejemplo de este es el trabajo de Andrew Howard que se publicó recientemente sobre la plataforma de Player/Stage llamado *Simple Mapping Utilities* [Howard, 2004]. Este

desarrollo es la piedra angular para este trabajo. Recientemente Eliazar y Parr [2003] liberaron el código de su algoritmo *DP-SLAM* [Eliazar and Parr, 2005], el cual no se revisó en este trabajo de tesis, pero valdría la pena hacerlo para trabajos futuros.

Por otro lado, existen esfuerzos por unificar diversos sistemas de software para robots móviles disponibles con la intención de reutilizar el mayor código posible, muestra de esto es el proyecto *MARIE* [Côté *et al.*, 2004]. Finalmente, de los proyectos más ambiciosos en cuanto a un marco de trabajo de software genérico para cualquier sistema de robots, tanto móviles como manipuladores, es *Orocos* [Network, 2000], y *Miró* [Kraetzschmar *et al.*, 2002], que son sistemas para el control de robots basado en componentes distribuidos utilizando *CORBA* [OMG, 2005].

Este trabajo propone hacer un sistema de software que resuelve el problema del SLAM, tratando de reutilizar los componentes de software disponibles, haciendo un sistema modular y que genere mapas de manera incremental y en línea, que sirva de referencia para posteriores desarrollos en robótica móvil. En la implementación se ha tomado como referencia la Arquitectura 3+ [Pacheco Sánchez, 2004], tratando de seguir buenas prácticas de programación y documentación.

También se propone un método de fusión sensorial de distintos sensores de distancia, aprovechando los resultados del filtro de partículas, generando mapas por cada sensor para luego unirlos en un mapa completo. El mapa final contendrá una representación más útil del ambiente, al agregar información complementaria entre sensores: lo que uno no puede percibir, el otro lo hará.

Por último, se presenta un algoritmo de exploración, basado en la búsqueda local de zonas libres dentro del espacio de configuraciones, tratando de cubrir todo el ambiente disponible, sin necesidad de esperar el procesamiento del mapa.

1.3. Definición del Problema

El trabajo de esta tesis cubre dos problemas principalmente: el problema del SLAM y una arquitectura de software que pueda ser reutilizable para proyectos futuros. En seguida se describen de manera introductoria ambos problemas.

1.3.1. El problema de la cartografía y localización simultánea

El problema de la cartografía es la adquisición de un modelo espacial del ambiente de un robot. Para adquirir este mapa, el robot hace uso de sus sensores de distancia. Sin embargo estos sensores tiene una percepción de la realidad restringida y limitada, además están sujetos a mucho ruido que genera errores de medición. Debido a las limitaciones de alcance de los sensores de distancia, el robot debe moverse en su ambiente para registrar áreas aún desconocidas, para continuar con la construcción del mapa de manera incremental.

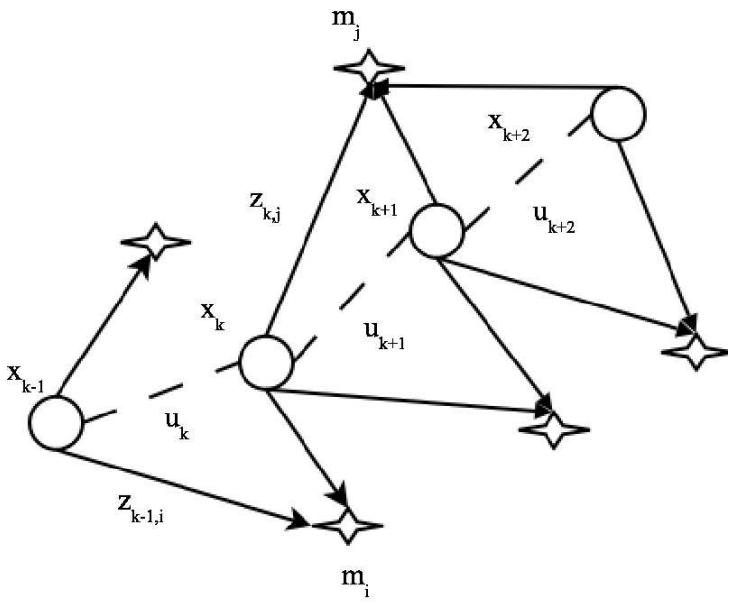


Figura 1.1: El robot se viaja de la posición x_{k-1} a la posición x_k, gracias al comando u_k. Las observaciones a las marcas m_i están dadas por las mediciones z_k.

Variable	Descripción
x_k	Posición del robot al tiempo k.
u_k	Comando de movimiento aplicado al tiempo $k-1$ para desplazar al robot de x_{k-1} a x_k en el tiempo k.
m_i	Posición de la marca o elemento del mapa m_i
$z_{k,i}$	Observación (medición) de la marca m_i desde la posición x_k al tiempo k.

Tabla 1.1: Notación usada en la descripción del SLAM.

En la Figura 1.1 se observa un esquema que refleja de manera más comprensible las operaciones involucradas en el problema del SLAM, mientras que en la Tabla 1.1 se describen las variables utilizadas en el esquema.

Para el problema incremental del SLAM, al robot que se encuentra en la posición x_{k-1} se le aplica un comando de movimiento u_k, que normalmente consiste de un vector que contiene la velocidad de desplazamiento y la velocidad rotacional, en el tiempo $k-1$. El tiempo k se considera discreto $(1, 2, \cdots, 3)$. En la nueva posición x_k, el robot, utilizando sus sensores de distancia, toma mediciones $z_{k,i}$ a las marcas m_i que componen el ambiente visible para el robot.

En la Sección 2.3 se enumeran los tipos de representación del ambiente actualmente usados. Su función es representar los obstáculos visibles para el robot. Cada tipo de mapa puede representar ciertos tipos o características de los obstáculos observados. De manera general se le llama marca o hito a la observación del tipo o característica a representar del obstáculo.

La tarea cartográfica tiene múltiples complicaciones que han convertido a este problema en un tema muy atractivo para la comunidad científica. Estas complicaciones se pueden agrupar en cinco categorías [Thrun, 2002b]:

- *El ruido de medición*. Los sensores de distancia son inexactos, susceptibles a ruido y muy dependientes de sus capacidades físicas. La odometría es más compleja ya que su ruido es estadísticamente dependiente, debido a que su error es acumulativo en el tiempo.

- *Dimensionalidad del espacio*. Describir un espacio requiere mucha información para definir cada elemento dentro del ambiente y mientras más detalle se le quiere agregar para tener una planeación de movimientos más exactas, la dimensionalidad crece.

- *Asociación de correspondencias*. Este es el problema de mayor dificultad y trata de determinar si diferentes mediciones, hechos en tiempos distintos, corresponden a un mismo objeto físico.

- *Ambientes dinámicos*. Una de las restricciones aplicadas al problema de la exploración es que el ambiente debe estar estacionario al momento de hacer esta tarea.

- *Selección de ruta para la exploración*. Determinar los movimientos óptimos para la exploración en un tiempo y movimientos mínimos.

En el Capítulo 2 se detallará la descripción del problema del SLAM, dándole claridad y formalidad.

1.3.2. El problema de la arquitectura de software

La Cátedra de Robótica Móvil del ITESM Campus Cuernavaca [ITESM, 2005] es un grupo de investigación interdisciplinario dedicado a enfrentar diversos problemas sobre el uso de robots móviles dentro de la vida diaria, entre estos ellos están la navegación, exploración, arquitectura, laboratorios virtuales, etcétera.

Dentro de la Cátedra se han hecho varios desarrollos individuales, cada uno construyendo desde cero sus aplicaciones y pruebas de concepto, haciendo extremadamente complicada la integración de un prototipo completo. Por lo que se decidió trabajar en formas de integrar éstos y futuros trabajos e ir subiendo en abstracciones, sin tener que rehacer los cimientos constantemente.

Uno de estos trabajos, que inicialmente se estudiaron para el propósito de esta investigación, es la tesis doctoral de Romero Muñoz [2001], que realizó un sistema para la exploración con robots móviles. Durante el primer semestre de la maestría se intentó utilizar el software sin éxito. La falta de prácticas de programación y documentación estándares, y la falta de tiempo del Doctor Romero para asesorías, hizo imposible hacer funcionar el sistema para utilizarlo como base para un trabajo posterior.

Este sistema de exploración seguía una arquitectura de control tradicional [Brooks, 1986], donde se descompone el problema en unidades funcionales de Percepción-Planeación-Acción (Figura 1.2).

Figura 1.2: Arquitectura de software tradicional compuesta de módulos funcionales.

El último año, un tesista de maestría, que trabajaba en la Cátedra, propuso una arquitectura de tres gradas [Pacheco Sánchez, 2004], comunicadas a través de un protocolo TCP/IP. La grada de más bajo nivel es la *funcional*, donde se interactúa directamente con los sensores y actuadores; sigue la grada de *ejecución* que es el servidor de aplicaciones robóticas (navegación, exploración, interacción, etc.), donde cada aplicación se instala para ser utilizada en cualquier momentos. La grada de mayor abstracción es la de *decisión* que coordina la ejecución de tareas (Figura 1.3).

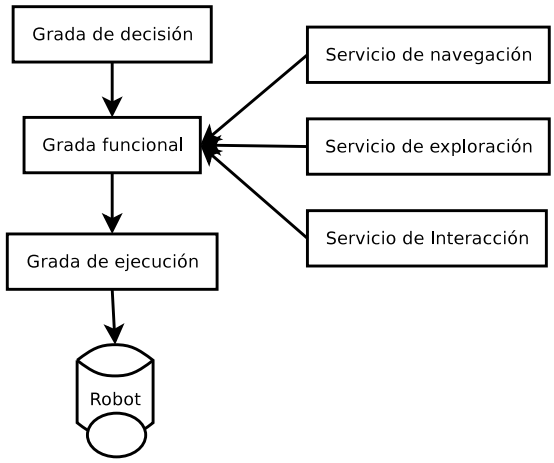


Figura 1.3: Arquitectura de software 3+ para robots móviles.

Cada grada es un problema a resolver, aunque la *funcional* está, al menos parcialmente, resuelta con el proyecto *Player/Stage* [Vaughan *et al.*, 2003]. La de *ejecución* en cambio ofrece retos que Orocos piensa resolver con un protocolo alambrado como *CORBA*, otros con paso de mensajes al estilo *XML-RPC* [Winer, 2005], otros más se evitan problemas y no usan un esquema distribuido, dejando el sistema en una forma monolítica. Finalmente, uno de los problemas más recientes y de mayor complicación es el coordinador de tareas. Cuando se busca que un robot interactúe entre humanos, se quiere que haga un conjunto de tareas interrelacionadas de manera concurrente; estas tareas pueden ser modeladas como un *Proceso de Decisión de Markov* (MDP por sus siglas en inglés) [Elinas *et al.*, 2004].

En este trabajo se presenta una arquitectura similar a la 3+, donde la capa funcional es el servidor *Player*, mientras que la capa de ejecución son una serie de hilos de procesamiento, que representan módulos independientes y atacan un problema específico: exploración, cartografía, supervisión, etc. El sistema genera de manera incremental y en línea el mapa del ambiente que esta recorriendo el robot, superando así unas de las limitaciones del *Simple Mapping Utilities*, que trabaja únicamente fuera de linea. Además se provee de un método

para controlar y supervisar el sistema a través del protocolo HTTP.

El sistema en general es una máquina de estados finitos que alberga otras máquinas de estado finito (una por cada hilo de procesamiento).

1.3.3. El problema de la fusión sensorial y la exploración

Mucho del trabajo realizado en la generación de mapas, se ha realizado utilizando un sólo sensor, ya que se simplifica el trabajo. Sin embargo el mapa resultante puede ser impreciso debido a las limitaciones físicas del sensor. El ejemplo característico son los obstáculos de vidrio, que no pueden ser detectados por el sensor más comúnmente utilizado, el telémetro láser. Mientras que por el otro lado, el sonar, capaz de detectar casi cualquier superficie, sufre de mucha especularidad, y tiene una resolución muy gruesa, por lo que lo mapas hechos con sonar lucen muy toscos, poco delineados. Es por eso, que para tener una mejor consistencia espacial, es mejor fusionar la información de sensores de diferente naturaleza, buscando obtener lo mejor de ambos sensores.

En este trabajo se propone utilizar el resultado del SLAM, que es la trayectoria más probable seguida por el robot dentro del ambiente, para generar, procesando las mediciones de un tipo de sensor de distancia a cada punto de la trayectoria, un mapa. Así, se puede generar un mapa por cada tipo de sensor registrado en el sistema, y luego fusionar dichos mapas con un operador *OR* probabilístico.

El último tema que abarca esta tesis es la exploración autónoma. Para moverse en un ambiente de manera óptima, el robot debe tener una representación de dicho ambiente y calcular una trayectoria. Sin embargo no se tiene tal mapa, y de acuerdo a la arquitectura propuesta y el costo computacional de la cartografía robótica, sería muy difícil tener el mapa incremental procesado al mismo tiempo que el movimiento del robot, por lo que nos resultaría ineficiente implementar un algoritmo que vaya a las zonas desconocidas del mapa incremental y se detenga a esperar el procesamiento de la cartografía, además está la necesidad que la cartografía tiene de volver a visitar zonas ya exploradas para mejorar su localización. Para tratar de solventar estas restricciones y necesidades se presenta un algoritmo que realiza movimientos dentro del espacio de configuraciones local del robot, sin una planeación compleja de trayectorias, limitándose a la operación secuencial: "giro, avance", tratando de cubrir todo el espacio libre disponible en el largo plazo. Sin embargo, debido a la carencia de un criterio de paro que indique cuando un ambiente se exploró por completo, el robot se moverá continuamente en el ambiente hasta que el usuario le indique.

1.4. Objetivos

Objetivo General

Desarrollar un sistema de software para la cartografía y localización simultanea de robots móviles, con procesamiento incremental en línea, bajo una arquitectura de software de tres gradas, multihilos y asíncrona. Este sistema proveerá de una estrategia de exploración autónoma y hará fusión sensorial del telémetro láser y los sonares del robot para hacer un mapa que provea información para una navegación más segura.

Objetivos particulares

- Implementar un algoritmo probabilista para el problema de SLAM de manera eficiente, autónoma y en línea.
- Utilizar una arquitectura de software de tres gradas.
- Proponer un esqueleto para el desarrollo de las nuevas aplicaciones robóticas de la Cátedra de Robótica Móvil, alentando la reutilización de código.
- Hacer fusión sensorial del telémetro láser y los sonares para tener representaciones de mapas más útiles en la planeación de trayectorias.
- Proponer e implementar un algoritmo de exploración.

1.4.1. Modelo Propuesto

El modelo a exponer en este trabajo se basa en la reutilización del mayor número de ideas y software disponibles, para ofrecer una prueba de concepto de lo que se puede hacer en el área de robótica móvil. Para poder embonar cada una de estas ideas se deben utilizar estrategias innovadoras, algunas de éstas se enumeran:

- Procesamiento en línea del mapa.
- Fijado de los parámetros del sistema a través de un archivo XML.
- Cliente delgado para la supervisión del proceso.

- Comunicación con el monitor del sistema a través del protocolo HTTP.
- Procesamiento asíncrono de los datos producidos por los sensores.
- Cada subproceso de la construcción del mapa se ejecutará en un flujo de procesamiento propio.
- Algoritmos para utilizar la información del sonar cuando las mediciones del láser sean incongruentes, como en el caso de paredes de vidrio.
- Una estrategia de exploración *glotona* que permita un recorrido los más completo posible del ambiente con movimiento continuos del robot.

En el diagrama de la Figura 1.4, se esboza una vista de despliegue[1] de la arquitectura del proyecto. El servidor que actúa como *middleware* será el responsable de ejecutar los componentes necesarios para realizar la tarea. Como se observa en este trabajo no existe un coordinador, tal como lo especifica la Arquitectura 3+, ya que no se realizan múltiples tareas.

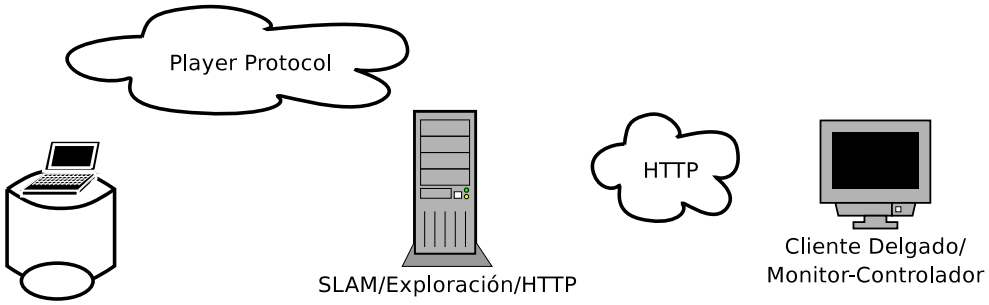


Figura 1.4: Vista de despliegue del sistema de software realizado en este trabajo de tesis.

Para el problema de la fusión sensorial se propone un algoritmo que agregará la información del sonar a los mapas creados con el telémetro láser, para facilitar una navegación y planeación de rutas más seguras. Esta unión se hará de manera adicional al proceso de SLAM. El principal motivo para la fusión es en el caso cuando el robot se topa con un obstáculo transparente para el láser (paredes de vidrio normalmente), entonces será cuando la información del sonar refuerce la hipótesis del obstáculo, ya que éste percibe una distancia menor que la láser, siendo más conforme a la realidad. Esta información se suma al mapa, generando una representación más exacta del ambiente.

[1]Vista UML que encapsula a los nodos de la topología del sistema de hardware donde el sistema se ejecuta.

Por el lado de exploración autónoma, se considera que la estrategia glotona (*greedy*) es, si no óptima, sí es muy cercana a ésta [Koenig *et al.*, 2001]. Esta estrategia tiene como premisa ir a la zona inexplorada (frontera) más cercana. Buscar una exploración óptima es un problema de complejidad exponencial, mientras que la exploración con heurística glotona se acerca mucho al óptimo, sin tanto costo computacional. Por otro lado se tiene que, con los algoritmos de SLAM, los mapas convergen más rápidamente cuando se cierran ciclos, y por ello hay que decidir entre cerrar ciclos o ir a fronteras.

Para decidir la ruta a la frontera, se debe tener un mapa, y éste se está construyendo. El procesamiento necesario para la construcción es grande y nuestro robot pasaría mucho tiempo detenido, esperando a que el procesamiento del mapa llegue a la posición actual y última observación del robot. Para evitar que el robot permanezca estático y haga lenta la exploración del ambiente, se propone movimientos ciegos en relación al mapa, utilizando sólo planeación y navegación local (dentro de la observación actual del robot). Con esto se logra de manera colateral, revisitar lugares ya observados, lo que facilita la localización del robot, además de que sea capaz de observar todo el ambiente.

1.5. Hipótesis

Se obtendrá un sistema de software que permita experimentar con la cartografía y exploración autónoma de ambientes, así como un esqueleto para extender a nuevas aplicaciones robóticas. Este sistema estará basado en la arquitectura de tres gradas, donde la grada de ejecución y de decisión estarán en un mismo nivel y los servicios robóticos se diseñarán bajo un modelo de hilos asíncronos.

Este sistema utilizará un sistema de fusión sensorial tanto de odometría, láser y sonares. El algoritmo para láser y sonar será novedoso y simple. Finalmente se probará que una estrategia simple de exploración es suficiente para una exploración autónoma y completa.

1.6. Organización del documento

Los siguientes capítulos están organizados de la siguiente manera: se hará una descripción detallada del problema de la cartografía y localización simultanea en el Capítulo 2, haciendo antes una revisión de la redes dinámicas bayesianas, necesarias para la formalización del problema.

El Capítulo 3 hará un estudio profundo de la técnica probabilista a utilizar para la solución del problema del SLAM, así como su implementación a nivel general.

En el Capítulo 4 se cubrirán las aportaciones hechas en este trabajo al problema del SLAM que son: la arquitectura de software empleada, la fusión sensorial entre el sonar y el láser y finalmente la estrategia de exploración autónoma.

Se hablará de los experimentos realizados en el Capítulo 5 y finalmente las conclusiones y trabajo futuro en el Capítulo 6. En el Apéndice A se expone la documentación del sistema de software que se realizó.

Capítulo 2

Cartografía y Localización Simultánea

En este capítulo se hace una introducción al problema a desarrollar en este trabajo. Comienza con una breve reseña histórica del problema cartográfico en la robótica móvil, justificando el uso de técnicas probabilistas para describir su operación. A modo de introducción se hablará de las redes bayesianas dinámicas, que son esenciales para describir formalmente el problema de la cartografía en robótica móvil. Inmediatamente después se hace una revisión de las representaciones del ambiente disponibles en la literatura, para terminar con la descripción detallada del problema cartográfico en los robots.

2.1. Robótica móvil y la probabilidad

La robótica móvil, que persigue obtener robots que se desplacen en ambientes de uso común para los seres humanos de manera autónoma, se ha venido desarrollando desde la segunda mitad del siglo XX. Su intención es buscar la manera de que máquinas, dotadas de un nivel de inteligencia artificial, puedan realizar tareas dentro de un ámbito de interacción humano.

Inicialmente los investigadores se dedicaron a los problemas de planeación y control, dando por sentado que las operaciones del robot eran perfectas [Thrun, 2002a], así como la suposición de una fiel y exacta representación del ambiente, que les permitiera hacer la planeación y ejecución de movimientos de manera ideal. Sin embargo, este enfoque cambió radicalmente en la década de 1980 al aparecer la robótica reactiva, donde el control del robot dependía directamente de los sensores de distancia. Se rechazaron los modelos o representaciones del ambiente, argumentando que el mundo es el modelo. Pero bajo este enfoque también hay una suposición poco realista: la percepción del robot es perfecta.

Diez años después se retomó la idea de modelar comportamiento del robot y del ambiente, pero ahora con una fuerte influencia probabilista. Actualmente no sólo el ambiente y el comportamiento de robot se modela con el uso de las leyes de la probabilidad, sino también casi todos los niveles de percepción, comportamiento y decisión [Thrun, 2002a].

Se ha echado mano de las probabilidades por la sencilla razón de que se ha observado que los resultados obtenidos, en la interacción con el mundo real, son mucho mejores [Thrun, 2000] al considerar a los sensores, actuadores y al mundo, no como variables exactas, sino como distribuciones de probabilidad, ya que se toma en consideración la incertidumbre asociada.

Más precisamente, las ideas de la probabilidad se aplican en el modelado de la *percepción* de la realidad por parte del robot y las *acciones* realizadas por el robot en ese ambiente. Sin embargo, la aplicación de la probabilidad en la robótica no está exenta de inconvenientes, ya que por más detallado que sea el modelo probabilístico, aun seguirá siendo erróneo, en particular al hacer falsas suposiciones de independencia [Thrun, 2002a].

2.2. Redes Bayesianas Dinámicas

Antes de intentar comprender más formalmente lo que son las redes bayesianas dinámicas, primero hay que revisar lo que es una red bayesiana. A continuación se presenta, de manera sucinta, una revisión a las redes bayesianas, para continuar con las redes bayesianas dinámicas.

2.2.1. Redes Bayesianas

Una red bayesiana es un grafo acíclico dirigido de nodos que representan variables aleatorias y los arcos, relaciones de dependencia entre las variables [Charniak, 1991]. Si hay un arco del nodo A a otro nodo B, se dice entonces que A es un *padre de* B. *Si un nodo tiene un valor conocido, se dice que es un nodo* evidencia. Un nodo puede representar cualquier tipo de variable aleatoria, como una medición observada, un parámetro, una variable latente o una hipótesis.

Un red bayesiana es también la representación de una distribución conjunta de todas las variables simbolizadas como nodos en el grafo. Sean las variables $X_1, \cdots, X_n$ y sean $padres(A)$ los padres del nodo A. Luego la distribución conjunta para X_1 hasta X_n es

representado como $\prod_{i=1}^{n} p(X_i \mid padres(X_i))$. Si X no tiene padres, su distribución de probabilidad se dice que es *a priori*, y de otra manera es *condicional*. En la Figura 2.1 se muestra un ejemplo de una red bayesiana.

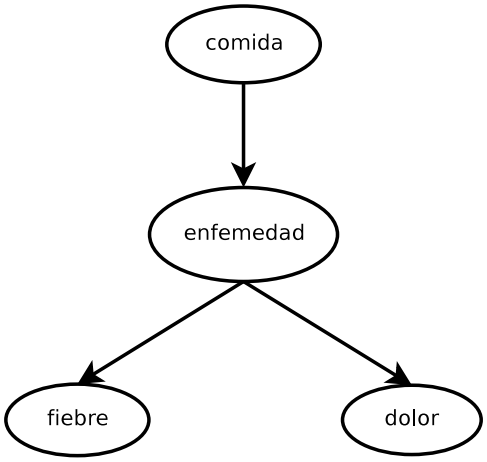


Figura 2.1: Ejemplo de una red bayesiana.

Para poder hacer cálculos es necesario especificar para cada nodo X la distribución de probabilidad dados sus padres. La distribución de X dados sus padres puede ser de cualquier forma, aunque es común utilizar distribuciones discretas o gaussianas para simplificar las operaciones.

2.2.2. Redes Bayesianas Dinámicas

Es posible ampliar el concepto de redes bayesianas para modelar sistemas dinámicos, los cuales reciben información a instantes continuos de tiempo (datos temporales) y esta información cambia el estado del sistema. Para modelar sistemas que manejan datos en series de tiempo, es natural utilizar modelos gráficos dirigidos, que capturan el hecho de que el tiempo fluye hacia adelante. Por lo tanto, si un modelo gráfico tiene todos su arcos dirigidos, tanto los que ocurren dentro de los intervalos de tiempo, como lo que ocurren en el cambio de intervalos, se le conoce como red bayesiana dinámica o DBN por sus siglas en inglés (Dynamic Bayesian Network) [Murphy, 2002].

En la Figura 2.2 se muestra un ejemplo muy sencillo de una red bayesiana dinámica. La variable S es el estado no observable en el tiempo y las variables X y Y son variables aleatorias que representan las observaciones dentro del estado.

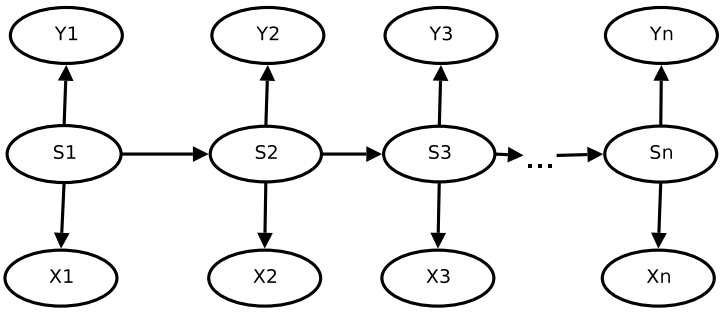


Figura 2.2: Ejemplo de una red bayesiana dinámica.

El resto de esta sección ha sido tomada del trabajo de Kevin Murphy [2002], al cual puede referirse para ampliar la discusión del tema.

Para representar el estado no observable u oculto en una DBN se utiliza un conjunto de N_h variables aleatorias, $S_t^{(i)}, i \in \{1, \cdots, N_h\}$, cada una de las cuales puede ser discreta o continua. De manera similar las observaciones pueden ser representadas en términos de N_o variables aleatorias, las cuales también pueden ser discretas o continuas.

Hay que recalcar que en estos modelos el tiempo se hace discreto en instantes que se consideran de igual duración, así es posible ver a las redes bayesianas dinámicas como un conjunto de redes estáticas unidas a lo largo de una secuencia de tiempo.

Para representar el modelo de transición en una DBN se definen las distribuciones condicionales correspondientes usando dos porciones de una red bayesiana temporal (2TBN por sus siglas en inglés -*2 slice Temporal Bayes Net*-), al que se le llamará $B_{\rightarrow}$. Los modelos de transición y de observación son entonces definidos como el producto de las distribuciones de probabilidad condicionales en la 2TBN:

$$P(Z_t \mid Z_{t-1}) = \prod_{i=1}^{N} P(Z_t^{(i)} \mid Pa(Z_t^{(i)}))$$

Donde $Z_t^{(i)}$ es el i-ésimo nodo de la red en el tiempo t (que puede ser oculto u observado; por lo que $N = N_h + N_o$), y $Pa(Z_t^{(i)})$ son los padres de $Z_t^{(i)}$, los cuales pueden estar en el mismo periodo de tiempo o el anterior (asumiendo que se está restringido a un modelo de Markov de primer orden). Ahora, es posible representar la distribución de estado inicial, $P(Z_1^{(1:N)})$, usando una red bayesiana estándar (una sola instancia en el tiempo), la que se denotará B_1. Juntas, B_1 y $B_{\rightarrow}$ definen la DBN. La distribución conjunta para una secuencia de tamaño T puede ser obtenida "desenrollando" la red hasta que se tengan T instantes de

tiempo, y luego multiplicando todas las distribuciones de probabilidad condicional juntas:

$$P(Z_{1:T}^{(1:N)}) = \prod_{i=1}^{N} P_{B_1}(Z_1^{(i)} \mid Pa(Z_t^{(i)})) \times \prod_{t=2}^{T} \prod_{i=1}^{N} P_{B_\rightarrow}(Z_t^{(i)} \mid Pa(Z_t^{(i)}))$$

Los modelos ocultos de Markov (HMM) forman parte de las DBN, y su especificación simplifica su representación y cálculos de inferencia [Murphy, 2002]. También hay otros modelos como los de estado espacio (SSM), modelos de Markov de memoria mixta, modelos ocultos de Markov de duración variable, modelos segmentados, modelos ocultos de Markov jerarquizados, etcétera, y todos ellos son casos específicos de redes bayesianas dinámicas. Cada uno de estos casos, debido a sus suposiciones de modelado, simplifican descripción gráfica probabilista y por ende sus operaciones para hacer trabajos de inferencia [Arulampalam *et al.*, 2002], en comparación del caso general.

Hasta aquí hemos hecho un repaso general de la redes bayesianas dinámicas, tópico relevante para comprender cómo se modela el problema del SLAM con incertidumbre asociada. Sin embargo, antes de entrar de lleno a dicho modelado, es importante revisar las representaciones del ambientes disponibles en la literatura. De la representación del ambiente deriva en mucho la manera de visualizar e implementar la solución del problema. La siguiente sección hablará exclusivamente de los tipos de mapas usados comúnmente.

2.3. Tipos de mapas

Existen varias razones para tener una representación del ambiente del robot móvil. La ponderación de estas razones inclina la balanza en la selección de un tipo mapa sobre otro. Los propósitos de tener un mapa se enumeran a continuación [Georgiev, 1999]:

1. Localización. Haciendo una correspondencia entre el modelo y la observación del robot en el ambiente, se puede ubicar la posición del robot.

2. Planeación de movimiento. Una vez localizado el robot y asignado una posición destino, se puede calcular el conjunto de movimientos necesarios en el mapa para alcanzar dicho destino de manera cercana a la óptima.

3. Evitar obstáculos. Uno de los puntos más importantes en la planeación de movimientos es el cálculo del espacio de configuraciones, dónde los obstáculos

son restricciones dentro de este espacio. Con un mapa se pueden obtener estas restricciones fijas del ambiente, para luego construir el espacio de configuraciones.

4. Uso humano. El mapa construido por el robot puede ser usado por el hombre en tareas de exploración cartográfica en zonas potencialmente peligrosas o para actividades en el área de la realidad virtual.

5. Una combinación de cualquiera de los anteriores.

Uno de los temas pendientes en el problema del SLAM es una forma unificada de representación de ambientes. Sin embargo actualmente, dependiendo del tipo de ambiente a cartografiar, se hace uso de un tipo de representación.

A grandes rasgos se pueden categorizar los ambientes en dos grupos: [Georgiev, 1999]: 1) interiores y exteriores; 2) estructurados y no estructurados.

	Interiores	**Exteriores**
Estructurados	oficinas, casas	bodegas, parques
No estructurados	almacenes desordenados	bosques

Tabla 2.1: Relación de las categorías en que se agrupan los tipos de ambientes.

Los ambientes interiores son relativamente pequeños, con muchas estructuras rectangulares; el piso es plano y su dinámica es ligeramente controlable y predecible. En cambio los ambientes exteriores son espacios grandes con obstáculos muy dispersos, de formas irregulares; el terreno es disparejo y resulta muy difícil controlarlo y/o predecirlo.

Los ambientes estructurados tiene regiones claramente distinguibles (cuartos, oficinas, estancia, etc.). En los ambientes no estructurados no se pueden hacer suposiciones previas sobre el espacio.

Tomando en cuenta las combinaciones de las dos categorías expuestas, se puede elegir el tipo de representación, considerando las restricciones de memoria y velocidad de sistema de cómputo a usar.

De manera general se pueden agrupar las diferentes de representaciones del ambiente en tres grandes grupos:

- *Geométricos*: Representan el ambiente por medio de primitivas geométricas, tal como líneas, celdas, puntos o polígonos. Estos tipo de mapas son los más detallados y ocupan más recursos de memoria.

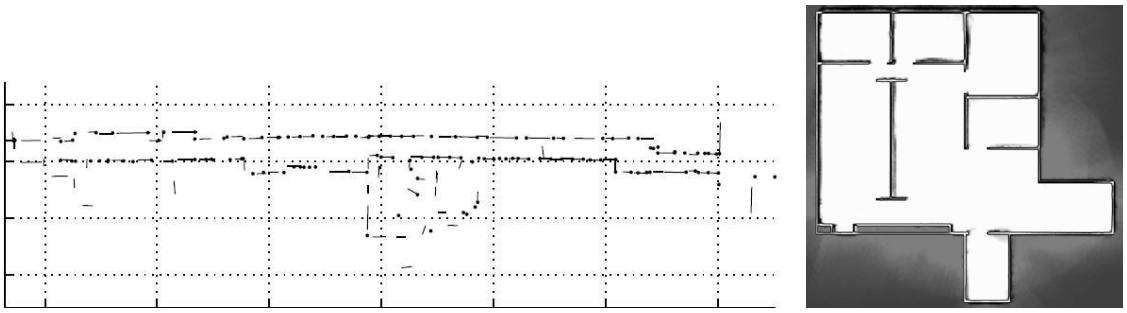
- *Topológicos*: Representa al espacio como un conjunto de grafos, donde cada marca natural es un vértice y sus arcos son conexiones entre ellos. Son más compactos pero eliminan mucha información métrica que puede ser importante.
- *Híbridos*: Mezclan las representaciones topográficas con información métrica en cada nodo.

La tendencia general en la literatura del SLAM es utilizar representaciones geométricas. No obstante, este tipo de representación tiene varios estilos de implementación, dependiendo de la categorización del ambiente:

- *Descripción explícita del espacio libre*. También conocida como *modelo de rejilla de ocupación*, donde las celdas tienen una probabilidad de estar ocupadas por obstáculos, y los sensores tienen un modelo probabilista, además, esta representación facilita la fusión sensorial. En la Figura 2.3(b) se observa un ejemplo de un mapa utilizando el modelo de rejilla de ocupación. Debido a que todo es espacio es discretizado en celdas sólo se recomienda para ambientes de interiores y estructurados.
- *Descripción explícita de objetos*. Son elementos geométricos dentro del espacio compuestos de líneas, arcos y polígonos que representan características con posición probabilista. En la Figura 2.3(a) se muestra un ejemplo de un mapa creado con líneas rectas como primitivas. Este modelo sólo almacena las relaciones geométricas del ambiente, se ha usado con éxito en ambientes interiores y exteriores, pero estructurados.
- *Mapa del terreno*. Expone la altitud de los puntos observados, además de marcar oclusiones existentes. Este tipo de mapa se emplea para ambientes no estructurados y generalmente en exteriores.

Cada tipo de mapas conlleva su propia solución al problema de la asociación de datos. La asociación de datos, como se verá más adelante, es la manera en que el robot distingue un objeto dentro del ambiente de los demás, a pesar de observarlo desde otro punto de vista o con otro tipo de sensor. Mientras que unos echan mano de los filtros Kalman para fusionar las diferentes mediciones en el tiempo como por tipo de sensor, otros utilizan otras variantes de filtros bayesianos.

En este trabajo, que esta orientado hacia la cartografía de ambientes interiores y estructurados, utilizará el enfoque de rejilla de ocupación probabilista. En el Capítulo 4 se hablará sobre la solución al problema de la asociación de datos.



(a) características (b) rejilla

Figura 2.3: En 2.3(a) se muestra un mapa de características, utilizando líneas rectas como primitivas [Diosi and Kleeman, 2004]. En 2.3(b) se muestra un mapa de rejilla de celdas con probabilidad de ocupación.

2.4. Cartografía Robótica

El problema de la cartografía es la adquisición de un modelo espacial del ambiente de un robot [Thrun, 2002b]. Para adquirir este mapa, el robot únicamente hace uso de sus sensores disponibles.

Esta tendencia a que el robot genere sus propios mapas, en lugar de que el humano le provea uno, viene de la idea de que la percepción humana y la del robot son diferentes, y así, un mapa generado a través de los limitados sensores del robot será mucho más útil que un mapa generado por un individuo con sentidos distintos. Además, si se persigue una verdadera autonomía en el robot, es mejor que éste pueda generar una representación de su ambiente sin ayuda exterior.

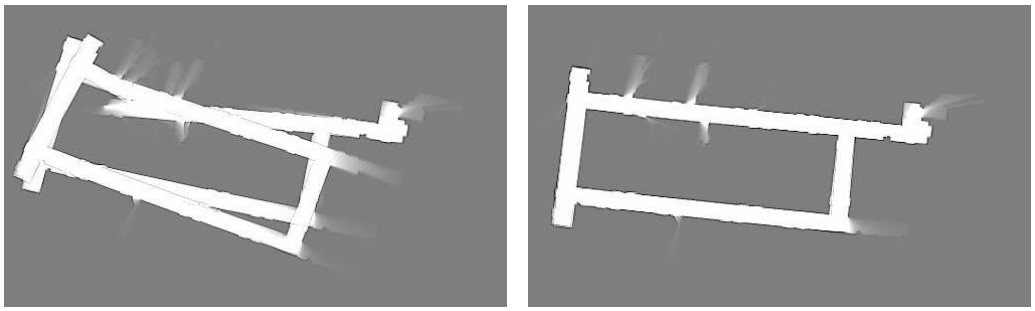
Sin embargo los sensores dentro del robot tienen una percepción de la realidad limitada a cierta cantidad de espacio y restringida por obstáculos, además están sujetos a ruido que genera errores de medición. Dadas estas limitaciones de alcance, el robot debe moverse en su ambiente para registrar áreas aun desconocidas y continuar con la construcción del mapa de manera incremental.

La construcción incremental de mapas impone tres problemas fundamentales: 1) localizar al robot, únicamente utilizando los sensores de odometría y de distancia; 2) construir un mapa global dados los puntos de observación; 3) fusionar la información de los sensores de distancia, tanto en tiempo como de tipo de sensor, para distinguir cada objeto que forma

parte del ambiente.

Los sensores de movimiento propio del robot obtienen información importante para la construcción del mapa, al ofrecer la posición desde el punto de vista de observación; pero el movimiento del robot también está sujeto a errores, por lo que la simple información odométrica es insuficiente para determinar la posición y orientación del robot. Es por esta razón que el trabajo cartográfico exige una constante localización del robot.

En la Figura 2.4(a) se observa un mapa generado sin el trabajo de localización, utilizando únicamente los datos arrojados por el sensor odométrico, los cuales, como ya se dijo, son susceptibles a error. Para empeorar la situación el error odométrico es estadísticamente dependiente, es decir, es acumulativo: mientras más se desplaza el robot, el error registrado es mayor. En cambio en la Figura 2.4(b) se aprecia un mapa donde se calcula la localización del robot.



(a) Sólo Odometría (b) Filtro de Partículas

Figura 2.4: La figura 2.4(a) es un mapa extraído utilizando sólo la odometría del robot. La figura 2.4(b) es un mapa aprendido con seguimiento de la posición del robot [Howard, 2004].

Por lo tanto se puede deducir que de manera general, el problema de la cartografía incremental se puede esquematizar como en la Figura 2.5. La percepción de los sensores de distancia generan un mapa local en el instante de observación. En la asociación de datos, se tratan de identificar los puntos que ya se habían visto previamente para correlacionarlos en el mapa global. Así mismo se tiene que hacer un trabajo de corrección en la posición del robot, ya que como se mencionó, el uso de la pura información odométrica es insuficiente. Posteriormente se selecciona un nuevo punto de observación, y se controla al robot para dirigirse a esa posición, y el ciclo se repite.

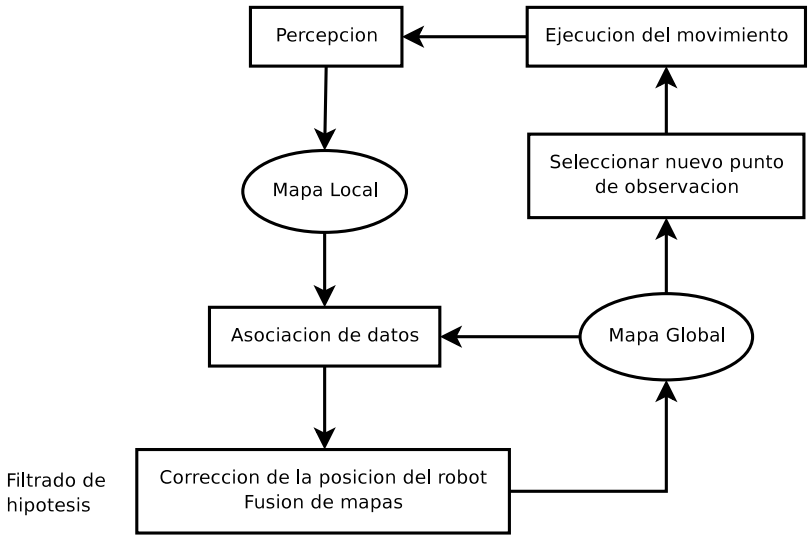


Figura 2.5: Procedimiento General del SLAM Incremental [Chatila, 2005].

El problema cartográfico no se detiene con los problemas generados por la naturaleza ruidosa de los sensores, existen también otras cuestiones difíciles de resolver, que aunque ya se mencionaron someramente en el capítulo anterior, a continuación se detallan [Thrun, 2002b]:

- *Dimensionalidad del espacio*. Describir un espacio requiere mucha información ya que hay que definir cada elemento dentro del ambiente. Desde el punto vista topológico, una simple casa, tiene una docena de elementos como corredores, habitaciones, intersecciones, puertas, y así. Al pasar a una representación geométrica en dos dimensiones, el número de elementos llega a miles, donde cada primitiva registrada es un elemento a describir. En un ambiente de tres dimensiones estos elementos llegan a ser millones.

- *Asociación de correspondencias*. Este puede ser el problema de mayor dificultad si se usa una representación explícita de los objetos. Este problema consiste en determinar si diferentes mediciones, realizadas en tiempos distintos, corresponden a un mismo objeto o son dos objetos distintos y distinguibles.

- *Ambientes dinámicos*. Una de las restricciones aplicadas al problema de la exploración es que el ambiente debe estar estacionario al momento de realizar esta tarea. Esto se debe a que la eliminación de ruido en los sensores se complica, ya que un objeto trasladándose dentro del lugar produce un error de medición a eliminar dentro del

mapa. Muy poco trabajo se ha realizado al problema cartográfico dentro de ambientes dinámicos. Uno de los trabajos revisado propone la existencia de dos mapas de celdas: uno donde el ambiente observado se registre en un mapa y los elementos estáticos se van filtrando al segundo mapa [Wolf and Sukhatme, 2004].

- *Exploración*. Determinar los movimientos exactos para la exploración en un tiempo y desplazamientos mínimos es un problema considerado como NP duro y aún sin solución óptima, sin embargo se tienen evidencias teóricas de que la exploración *glotona* tiene una complejidad espacial aceptable [Koenig *et al.*, 2001].

El trabajo científico ha puesto especial interés en el primer problema descrito: cómo generar un mapa lidiando con los errores de los sensores de distancia y más que nada con el odómetro. Para poder actualizar la parte del mapa visible por sus sensores de distancia, debe saber exactamente su posición, sin embargo, para poder localizarse el robot requiere un mapa previo [Murphy, 1999], ya que no se puede confiar en la información odométrica registrada. Pero el mapa previo no existe, porque se esta haciendo en ese momento. Esto hace un problema con variables en abrazo mortal: no puede hacerse un mapa sin localización, no se puede localizar el robot sin un mapa. Para resolver este dilema es necesario resolver ambas incógnitas de manera simultánea, por esto, en la literatura científica, este problema se le conoce como Cartografía y Localización Simultáneas, SLAM por sus siglas en inglés (Simultaneous Localization And Mapping).

2.5. Descripción formal del SLAM

Teniendo claro el problema de la Cartografía y Localización Simultáneas y sus distintas complicaciones inherentes, ahora se puede formular el problema de manera probabilista, para luego buscar mecanismos de solución.

2.5.1. SLAM como una red bayesiana dinámica

A finales de los noventas se modeló el problema como una red bayesiana dinámica [Murphy, 1999]. Este enfoque fue afinado [Montemerlo *et al.*, 2002a] tal como se observa en la Figura 2.6. En ella se observa que el robot se mueve de la posición s_1 utilizando de una secuencia de instrucciones de control $u_1, u_2, \cdots, u_t$. Se considera u_1 inexistente porque se da por sentado que no hubo ningún comando de movimiento para llegar a la posición inicial del

robot. Al moverse cambia de posición y cada posición donde realiza una observación del ambiente se considera un estado. Las observaciones del ambiente son medidas de distancia $z_1, z_2, \cdots$ a los objetos dentro del ambiente obtenidas por medio de sus sensores. Estos objetos, representados como puntos en un plano, representan las marcas $l_1, l_2, \cdots$, que deben ser asociados con las marcas vistos anteriormente.

Para decirlo más puntualmente, z_t es la medición de un sensor en el tiempo t y u_t es el comando de control ejecutado por el robot en el intervalo de tiempo $[t-1, t)$.

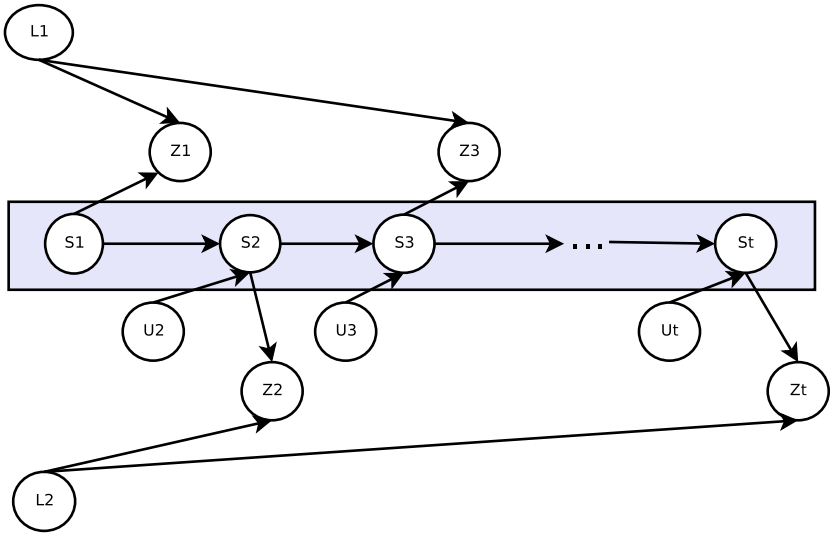


Figura 2.6: Modelo gráfico probabilista del problema del SLAM [Montemerlo *et al.*, 2002a].

2.5.2. Formulación del problema del SLAM

Con la visualización del SLAM como una red bayesiana dinámica, ahora se formula analíticamente el problema del SLAM: El problema de la Cartografía y Localización Simultánea es determinar el conjunto de todas las marcas l[1] que forman el mapa m y las posiciones del robot s_t dadas las mediciones de los sensores y las instrucciones de control.

La formulación que a continuación se define fue establecida por [Thrun, 2002b] donde, de manera general, se modela el problema del SLAM como una regla de Bayes:

$$p(x \mid d) = \eta \, p(d \mid x) p(x)$$

[1]Como ya se mencionó, las marcas son los puntos que conforman un mapa.

Donde se desea aprender acerca de una cantidad x (por ejemplo un mapa), basado en datos medición d (por ejemplo un barrido láser u odometría). Entonces la regla de Bayes indica que el problema puede ser resuelto multiplicando $p(d \mid x)$ y $p(x)$. El término $p(d \mid x)$ es el modelo *generativo*, que describe el proceso de generar mediciones de los sensores bajo diferentes mundos x. El término $p(x)$ es llamado *antecedente* y especifica el deseo de asumir que x es la instancia en el mundo antes de que lleguen datos. Finalmente η es un normalizador para asegurar que el lado izquierdo de la regla de Bayes es una distribución de probabilidad válida.

En la cartografía robótica, los datos de los sensores van llegando secuencialmente. Como ya se explicó en la sección anterior, existen dos tipos de datos que se reciben de los sensores: de medición de distancia y de control. Se denotan a las mediciones de los sensores de distancia como z y a los comandos de movimiento como u. Por conveniencia se asume que los datos son recibidos alternadamente:

$$z_1, u_1, z_2, u_2, \dots$$

donde los subíndices indican el instante de tiempo.

Entonces, exponiendo la regla de Bayes anterior con la notación descrita y asumiendo que el comando de movimiento del robot es independiente de la posición, se obtiene:

$$p(x \mid z, u) = \eta \, p(z \mid x, u) p(x \mid u) \tag{2.1}$$

Ahora, también se asume que los comandos de movimiento son independientes de la mediciones de distancia, por lo que la Ecuación 2.1 se puede reformular así:

$$p(x \mid z, u) = \eta \, p(z \mid x) p(x \mid u) \tag{2.2}$$

Como se vio ya, el problema del SLAM se puede modelar como una red bayesiana dinámica, razón para extender la regla de Bayes a un problema de estimación temporal utilizando un filtro bayesiano, que es un estimador recursivo para calcular una secuencia posterior de distribuciones de probabilidad sobre unas cantidades que no pueden ser observadas directamente.

$$p(x \mid u) = \sum p(x_t \mid u_t, x_{t-1}) p(x_{t-1}) \tag{2.3}$$

Es posible suponer que $p(x_{t-1}) = p(x_{t-1} \mid z^{t-1}, u^{t-1})$ considerando que el exponente t se refiere a todos los datos que existen antes del tiempo t, es decir:

$$z^t = \{z_1, z_2, \cdots, z_t\}$$
$$u^t = \{u_1, u_2, \cdots, u_t\}$$

Por lo que finalmente, si se sustituye 2.3 en 2.2, se obtiene que el problema de la cartografía robótica se puede representar con la siguiente formulación dentro de un espacio continuo:

$$p(x_t \mid z^t, u^t) = \eta p(z_t \mid x_t) \int p(x_t \mid u_t, x_{t-1}) p(x_{t-1} \mid z^{t-1}, u^{t-1})\, dx_{t-1}$$

Como se mencionó, los filtros bayesianos al ser recursivos, la probabilidad posterior $p(x_t \mid z^t, u^t)$ es calculada utilizando la probabilidad obtenida en el tiempo anterior (propiedad Markoviana). La probabilidad inicial al tiempo $t = 0$ es $p(x_t \mid z^t, u^t) = p(x_0)$.

Es requisito indispensable para un filtro bayesiano que el estado x_t contenga todas las cantidades desconocidas, esto es el mapa y la posición del robot. Ahora, usando m para designar el mapa y s para describir la posición del robot, se tiene el siguiente filtro de Bayes:

$$p(s_t, m_t \mid z^t, u^t) = \eta p(z_t \mid s_t, m_t) \int\int p(s_t, m_t \mid u_t, s_{t-1}, m_{t-1}) p(s_{t-1}, m_{t-1} \mid z^{t-1}, u^{t-1})\, ds_{t-1}\, dm_{t-1} \quad (2.4)$$

Se advierte que para este estimador no se requiere una integración del mapa m, como se muestra en la Ecuación (2.4). Tal integración es complicada debido a la alta dimensionalidad de espacio para todos los mapas posibles. Es por esto que la suposición de un mundo estático es de mucha importancia práctica. Esta suposición implica que el índice de tiempo en el mapa m no es necesario. Además, la mayoría de las propuestas de solución asumen que el movimiento del robot es independiente del mapa. Esto conduce a una forma más conveniente del filtro:

$$p(s_t, m \mid z^t, u^t) = \eta p(z_t \mid s_t, m) \int p(s_t \mid u_t, s_{t-1}) p(s_{t-1}, m \mid z^{t-1}, u^{t-1})\, ds_{t-1} \quad (2.5)$$

Para poner el estimador (2.5) a trabajar, se tienen que especificar dos probabilidades generativas: $p(s_t \mid u_t, s_{t-1})$ y $p(z_t \mid s_t, m)$. Estas distribuciones se asumen usualmente

como invariantes en el tiempo, por lo que son escritas comúnmente como $p(s \mid u, s')$ y $p(z \mid s, m)$. Ambas son modelos generativos del robot y su ambiente.

La probabilidad $p(z \mid s, m)$ se le conoce como el *modelo de percepción* y describe, en términos de probabilidad, cómo las mediciones z se generan en estados s y mapas m. En otras palabras el modelo de percepción describe el funcionamiento de los sensores del robot.

La probabilidad $p(s \mid u, s')$ describe que al ejecutar el control u en el estado s', el robot se conduce al estado s, y a esto se le llama el *modelo del movimiento*. Generalmente el modelo probabilista de movimiento es un generalización de la cinemática del robot en el ambiente [Thrun *et al.*, 2000].

Hay que hacer incapié en que la Ecuación (2.5) no puede ser implementada en una computadora tal como está expuesta, ya que el espacio de la distribución de probabilidad de los mapas y las posiciones del robot son una distribución continua, y por ende infinita. Es por esto que deben tenerse más suposiciones para simplificar la tarea. Estas suposiciones hacen la diferencia entre los diferentes algoritmos de SLAM existentes.

2.5.3. Resumen del capítulo

El objetivo de este capítulo fue poder formular el problema del SLAM como una red bayesiana dinámica, cuya solución incremental está dada por un filtro bayesiano. Para poder hacer esta formalización, se revisó inicialmente los conceptos de las redes bayesianas como modelos gráficos probabilistas. Enseguida se extendió el concepto de redes bayesianas a redes bayesianas dinámicas, para modelar procesos estocásticos que varían en el tiempo.

Después de exponer los fundamentos del razonamiento con incertidumbre, se hace un breve paréntesis, con el fin de redondear los temas introductorios, y se describen de los tipos de representación del ambiente, es decir los mapas, y se comenta su utilidad específica. Una vez dados los tópicos necesarios para formular el problema del SLAM, se presenta de manera más detallada el problema de la cartografía robótica, para finalmente describirlo de manera formal.

En el siguiente capítulo se expondrán de manera los mecanismos de inferencia para redes bayesianas dinámicas. De estos mecanismos se expone a detalle la herramienta matemática a utilizar para resolver la red bayesiana dinámica que describe el problema tratado.

Capítulo 3

Filtros de Partículas

Como ya se definió en el capítulo anterior, el problema de la cartografía y localización simultánea se modela como una red bayesiana dinámica, donde el objetivo es obtener, al mismo tiempo, tanto el mapa como la posición del robot, razón por la que se hará un repaso de los algoritmos de inferencia disponibles para la solución de redes bayesianas dinámicas. Entre ellos están los analíticos que dan soluciones exactas; sin embargo, dada la alta dimensionalidad[1] del problema, o su cantidad de variables aleatorias a considerar, vuelven intratable su cálculo. Es en este caso donde es preferible utilizar otros métodos como los basados en simulaciones de *Monte Carlo*.

Finalmente se estudiarán más específicamente los filtros de partículas en su modalidad de *Rao-Blackwell*, lo cuales, según la evidencia, dan mejores resultados específicamente para el problema del SLAM en tiempos aceptables.

Antes de entrar a las técnicas de solución del problema del SLAM, sería de utilidad ver a grandes rasgos la aplicación de la técnica: Como ya se mencionó que el problema se modela como una red bayesiana dinámica, donde nuestras variables ocultas a es la posición del robot y el mapa del ambiente. El propósito entonces es filtrar una distribución de probabilidad que nos de las posiciones del robot a lo largo de la exploración del ambiente y el mapa asociado a esa trayectoria.

Con la técnica de filtro de partículas, cada partícula nos representa una trayectoria seguida por el robot y un mapa asociado a esa trayectoria. Mediante el teorema de *Rao-Blackwell*, podemos olvidarnos temporalmente de la distribución de probabilidad de la variable oculta que representa a los mapas del ambiente, y fijarnos únicamente en la variable de la posición del robot. Entonces la distribución propuesta, con la cual se muestrean las partículas es

[1] Dimensionalidad: número mínimo de coordenadas independientes requeridas para especificar de manera única los puntos en un espacio.

una distribución normal parametrizada con la cinemática del robot, y la ponderación de la partícula se hace con en nivel de correlación entre las celdas marcadas como obstáculos dentro del mapa asociado a la partícula. Al final del proceso, obtenida la trayectoria más probable, se calcula analíticamente la variable oculta del mapa.

3.1. Soluciones a las redes bayesianas dinámicas

Como en todo modelo gráfico probabilista, se pueden hacer varios tipos de inferencia para obtener información de la red bayesiana dinámica utilizada. La inferencia a utilizar dependerá de la clase de solución buscada. De entre las inferencias disponibles más importantes se encuentran las listadas en la Tabla 3.1 [Murphy, 2002].

Inferencia	Descripción
Filtrado	Calcular $P(X_t \mid y_{1:t})$, por ejemplo, seguimiento del estado del sistema en el tiempo.
Predicción	Calcular $P(X_{t+h} \mid y_{1:t})$ para algún horizonte $h > 0$ en el futuro
Clasificación	Calcular $P(y_{1:y}) = \sum_{x_{1:y}} P(x_{1:t}, y_{1:t})$. Esto puede ser usado para calcular la proximidad de una secuencia bajo diferentes modelos.

Tabla 3.1: Tipos de inferencia en redes bayesianas dinámicas

Para realizar las inferencias descritas sucintamente en la Tabla 3.1, se disponen de varios algoritmos. En un inicio están aquellos algoritmos que realizan inferencias exactas sobre modelos cuyas variables ocultas son discretas, sin importar la distribución de probabilidad de los nodos observados. La Tabla 3.2 enumera algunos de estos algoritmos y sus propiedades [Murphy, 2002].

Cooper [1987] probó que una de las restricciones principales para el uso de redes bayesianas con distribuciones discretas, es el hecho de que, en general, su solución es NP-Dura. Al ser más compleja la topología de la red o al crecer la cantidad de variables y el tamaño de las distribuciones conjuntas, el tiempo de requerido para la inferencia aumenta exponencialmente. Más aun, cuando se tienen modelos cuyos estados ocultos tienen distribuciones continuas o mixtas (discretas en unas variables, continuas en otras), la

Nombre	Descripción
Algoritmo de avance-retroceso (forwards-backwards algorithm)	Se basa en convertir la DBN en un Modelo Oculto de Markov y aplicar el algoritmo clásico de avance-retroceso.
Árboles conjuntos abiertos (unrolled junction tree)	Se basa en desenrollar la DBN en T instantes (donde T es la longitud de la secuencia) y luego aplicar cualquier algoritmo de inferencia para una red bayesiana estática.
Algoritmo de frontera (frontier algorithm)	Calcula la DBN como un Modelo Oculto de Markov considerando únicamente los nodos hasta el instante presente, dejando las relaciones futuras.
Algoritmo de interfaz (interface algorithm)	Es una modificación al algoritmo de frontera.

Tabla 3.2: Algoritmos de inferencia exacta

inferencia exacta del estado prácticamente no existe [Murphy, 2002]. Entonces se necesitan algoritmos que busquen aproximaciones a la distribución de probabilidad condicional.

Actualmente existen varios algoritmos que buscan aproximaciones a las probabilidades condicionales, unos están basados en hacer discreto el espacio mientras que otros utilizan técnicas de muestreo. No obstante, la elección del mejor algoritmo depende de la naturaleza del problema a tratar. En este momento no existe una única técnica, ya sea aproximada o exacta, que funcione bien en cualquier tipo de red [Charniak, 1991].

3.1.1. Filtros Kalman

Una de las primeras formas de resolver el problema del SLAM fue utilizando filtros Kalman [Thrun, 2002b], ya que una vez modelado como una red bayesiana dinámica, parecía ser la técnica obvia a utilizar, debido a que es la que tiene una inferencia más exacta y con más tiempo de existencia y utilización en las comunidades probabilistas. Sin embargo, como se verá en la Subsección 3.1.2, no se obtiene los resultados esperados. Por tanto, con el motivo de mostrar estas razones, daremos una revisión rápida de los filtros Kalman. Esta

revisión esta basada en el trabajo de [Welch and Bishop, 1995].

Uno de los casos típicos de redes bayesianas continuas es cuando el estado oculto es modelado con gaussianas. En 1960 R. E. Kalman publicó un método con una solución recursiva, el cual tuvo gran éxito con la aparición de la computación.

Los filtros Kalman, como se le llama a dicha técnica, son la herramienta con mayor fundamento matemático, evitando el uso de heurísticas y de búsquedas de soluciones no determinísticas, llegando así a resultados más exactos a la hora de hacer un seguimiento del estado en una DBN, lo que los convierten en la técnica a utilizar para este tipo de problemas, si se ajustan las suposiciones que exige Kalman, tal como el uso de distribuciones normales.

Viendo el algoritmo a "alto nivel", los filtros de Kalman estiman un proceso usando un forma de control de retroalimentación: el filtro estima el estado del proceso en un instante dado y luego obtiene retroalimentación en forma de mediciones ruidosas.

Las ecuaciones para el filtro de Kalman se clasifican en dos grupos: a) ecuaciones de *actualización de estado* y b) ecuaciones de *actualización de medición*.

Las ecuaciones de actualización de estado son responsables de proyectar al futuro la estimación del estado actual y la estimación de la covarianza del error para obtener el estimado *a priori* del siguiente instante. Las ecuaciones de actualización de medición son responsables de la retroalimentación, al incorporar a la estimación *a priori* una nueva medición para obtener un estimado *a posteriori* mejorado.

Las ecuaciones de actualización de estado pueden ser visualizadas como ecuaciones *predictoras, mientras que las actualizaciones de medición pueden ser vistas como ecuaciones* correctoras. Esta relación *predictor-corrector* se puede observar gráficamente en la Figura 3.1, donde la entrada de los datos de uno corresponde a la salida del resultado del otro.

Una crítica a los filtros de Kalman es que sólo pueden estimar el estado de un proceso ocurre en tiempos discretos, y es dirigido por una ecuación lineal. Entonces ¿qué ocurre si el proceso a estimar y/o la relación de medición con el proceso es no-lineal? Un filtro de Kalman que linealiza sobre la actual media y covarianza se conoce como un *filtro extendido de Kalman* o EKF por sus siglas en inglés.

En algo análogo a una serie de Taylor, se linealiza la estimación alrededor del actual valor estimado usando las derivadas parciales de las funciones del proceso y la medición para calcular estimados aún frente a relaciones no lineales.

Es importante destacar que una falla fundamental de los EKF es que las distribuciones

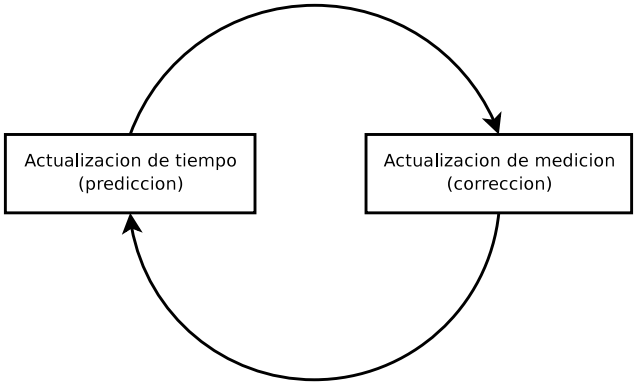


Figura 3.1: Ciclo del filtro de Kalman. La actualización del estado proyecta el estimado actual del estado hacia adelante en el tiempo. La actualización de medición ajusta el estimado proyectado con una medición en ese tiempo.

(o densidades en el caso continuo) de varias variables aleatorias no son normales después de llevar a cabo su respectiva transformación no lineal. Los EKF son simplemente una estimación *ad hoc* del estado que sólo aproxima a la optimalidad de la regla de Bayes por linealización.

3.1.2. Filtros de Kalman aplicados en SLAM

El enfoque más común para resolver el problema de la Cartografía y Localización Simultanea es utilizar filtros Kalman [Thrun, 2002b]. Es más, la literatura designa a los algoritmos SLAM como aquellos que están basado en filtros Kalman, aunque SLAM es el nombre de un problema y no de una solución. La Figura 3.2 muestra el filtro Kalman aplicado a la localización, la cual es parte del problema de la cartografía.

La principal ventaja del enfoque con filtro Kalman es el hecho de que estima la distribución posterior completa de los mapas en línea. Además los filtros Kalman contienen toda la incertidumbre de las medicines en el mapa, lo cual puede ser benéfico para la navegación. Adicionalmente el enfoque puede demostrar que converge con probabilidad de uno al verdadero mapa y posición del robot, sobre una distribución de incertidumbre residual que se origina en gran medida de una fluctuación original aleatoria [Thrun, 2002b].

A pesar de todas estas ventajas, la solución al problema cartográfico utilizando EKF sufre de tres importantes limitaciones [Thrun, 2002a]:

- La complejidad para cada actualización es de $O(K^2)$, donde K es el número de

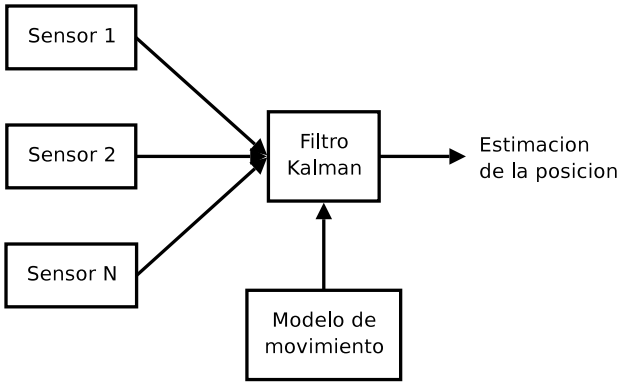


Figura 3.2: Esquema del uso del filtro de Kalman para el problema de la estimación de la posición de un robot [Chatila, 2005].

características que describen el mapa, aún con la ausencia del problema de asociación de datos. Esta limitación impone una importante restricción al escalar.

- Los EKF no pueden incorporar información negativa, es decir, no pueden usar el hecho de que un robot pueda no ver una característica aunque esta sea esperada. La razón de esta incapacidad es que las mediciones negativas producen distribuciones posteriores no gaussianas, que no pueden representarse en los EKF.
- Los EKF no proveen soluciones sólidas al problema de la asociación de datos. La falsa asociación de datos conduce frecuentemente a errores catastróficos en la cartografía.

Analizando con mayor detalle las desventajas se tendrá una idea más clara de ellas: La más costosa operación en la actualización del filtro de Kalman son las multiplicaciones de matrices, las cuales pueden ser implementada en $O(K^2)$ [Thrun, 2002b]. En la práctica el número de características no se conoce *a priori*, por lo que si el tamaño del ambiente a modelar es grande, a medida que la ruta del robot es mayor, la actualización se va complicando cada vez más.

Además, la limitación más importante de los filtros Kalman es la suposición del ruido gaussiano. En particular la suposición de que el ruido de medición debe ser independiente y gaussiano impone una limitación clave, que tiene importantes implicaciones al momento de implementarlo. Por ejemplo, en un ambiente con dos marcas diferentes exactamente iguales, la identificación tales marcas conducirá a una distribución multimodal sobre las posibles posiciones del robot. Dicho esto de manera más general, los enfoques con filtros Kalman son incapaces de lidiar con el problema de la asociación de correspondencias, el

cual es la dificultad de asociar mediciones individuales de un sensor con características en el mapa.

3.2. Métodos Monte Carlo

En el caso de las técnicas de muestreo, esencialmente los algoritmos de aproximación para la inferencia de redes bayesianas, se basan en asumir aleatoriamente la existencia de valores para algunos nodos y luego usan estos valores para inferir la cantidad de los otros nodos. Simultáneamente mantienen estadísticas de estos valores que los nodos toman, y finalmente estas estadísticas dan la respuesta [Charniak, 1991]. A estos métodos se les llama técnicas de Monte Carlo.

En algunas aplicaciones de modelos gráficos, la distribución posterior de una variable no observada es el principal punto de interés. Para la mayoría de las situaciones la distribución posterior se obtiene con el propósito de obtener el valor esperado para hacer, por ejemplo, predicciones [Jordan, 2002]. Cuando las variables de los nodos son distribuciones continuas, no monotónicas o muy complejas para ser evaluadas usando técnicas analíticas, se deberán utilizar las técnicas Monte Carlo.

La presente explicación de los métodos Monte Carlo para la inferencia de redes bayesianas dinámicas es tomada del trabajo de Jordan [2002].

El problema fundamental que aquí se enfrenta es encontrar el valor esperado de alguna función $f(x)$ con respecto a una distribución de probabilidad $p(x)$. Aquí, x puede estar formada por variables discretas, continuas, o alguna combinación de ambos. En el caso de variables continuas se quiere calcular el valor esperado $\langle f \rangle$ descrito en la Ecuación 3.1.

$$\langle f \rangle = \int f(x)p(x)dx \tag{3.1}$$

Esto se ilustra esquemáticamente para una distribución de una variable en la Figura 3.3.

La idea general detrás de los métodos de muestreo es la de obtener un conjunto de muestras $x^{(m)}$ (donde $m = 1, \cdots, M$) a partir de la distribución $p(x)$. Esto permite que el valor esperado en la Ecuación 3.1 se aproxime a la suma finita

$$\hat{f} = \frac{1}{M} \sum_{m=1}^{M} f(x^{(m)})$$

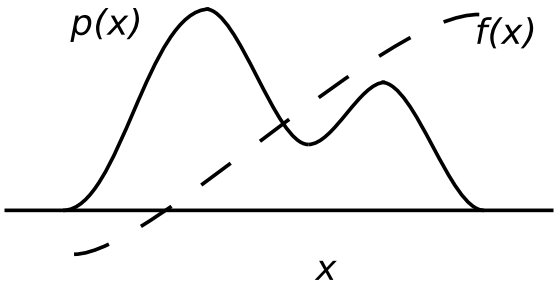


Figura 3.3: Función $f(x)$ cuyo valor esperado será evaluado con respecto a una distribución $p(x)$.

La precisión de tal aproximación dependerá de una variedad de factores. De inicio hay que advertir que mientras las muestras $x^{(m)}$ sean esbozadas a partir de la distribución $p(x)$ entonces $\langle \hat{f} \rangle \simeq \langle f \rangle$ y el estimador tiene el promedio correcto. La varianza del estimador se observa fácilmente como $\frac{\sigma^2}{M}$, donde

$$\sigma^2 = (f - \langle f \rangle)^2$$

es la varianza de la función $f(x)$ bajo la distribución $p(x)$. Valga aquí enfatizar que la precisión del estimador no depende de la dimensionalidad de x, y con esto puede alcanzar una alta precisión con un número relativamente pequeño de muestras $x^{(m)}$. Además, la varianza del estimador decrecerá con el incremento del número M de muestras.

Mientras que los métodos de muestreo tienen un amplio espectro de aplicación, para este trabajo sólo interesa el caso en donde la distribución $p(x)$ se especifica en términos de un modelo gráfico. En el caso de un grafo dirigido con variables ocultas, resulta obvio que las muestras vienen de la distribución conjunta (asumiendo que es posible muestrear de una distribución condicional en cada nodo) usando el siguiente enfoque de muestreo: La distribución conjunta esta especificada por

$$p(x) = \prod_{i=1}^{d} p(x_i \mid Pa(x_i))$$

donde $Pa(x_i)$ denota un conjunto de variables asociadas con los padres de x_i. Para obtener una muestra de la distribución conjunta se hace un recorrido a través del conjunto de variables en el orden $x_1, \cdots, x_d$ muestreando a partir de las distribuciones condicionales $p(x_i \mid Pa(x_i))$. Esto es siempre posible ya que a cada instante todos los valores de los padres

serán instanciados. Después de un recorrido a través del grafo se obtendrá un conjunto de la distribución conjunta.

En el caso de un grafo dirigido, donde algunos de los nodos son instanciados con valores observados, es el mismo principio que el caso anterior: a cada instante, cuando un valor muestreado es obtenido de una variable x_i, cuyo valor es observado, el valor muestreado es comparado con el valor observado y si concuerdan, el valor de la muestra se retiene y el algoritmo procede a la siguiente variable en turno [Jordan, 2002]. Sin embargo, si el valor muestreado y el valor observado no concuerdan, la muestra completa se descarta y el algoritmo comienza de nuevo con el primer nodo del grafo. Este algoritmo muestrea correctamente una distribución posterior ya que ésta corresponde al hecho de esbozar muestras de la distribución conjunta de las variables ocultas y de las variables de datos y descarta aquellas muestras que no concuerdan con los datos observados. La manera de descartar las muestras que no corresponden a las observaciones define las variantes de los métodos Monte Carlo.

3.3. Filtros de partículas

El algoritmo de filtros de partículas es un método Monte Carlo que conforma la base para la mayoría de los filtros Monte Carlo desarrollados. A los filtros de partículas también se les conoce como secuencias Monte Carlo, muestreo secuencial con importancia (SIS por sus siglas en inglés -Sequencial Importance Sampling-), *bootstrap filter*, algoritmo de condensación, supervivencia por adaptación, etc [Murphy, 2002].

El tema explicado a lo largo de esta sección se recoge en el artículo de Simon Maskell y Neil Gordon [2002].

La idea fundamental es representar la función de distribución posterior buscada con un conjunto de muestras aleatorias con pesos asociados y calcular estimaciones basadas en estas muestras y pesos. Al aumentar el número de muestras, esta caracterización Monte Carlo se vuelve una representación equivalente de la descripción funcional usual de la distribución posterior, y el filtro SIS se acerca al estimador bayesiano óptimo.

Para detallar estas ideas se denota $\{x^i_{0:k}, w^i_k\}^{N_s}_{i=1}$ como *mediciones aleatorias* que caracterizan la función de distribución posterior $p(x_{0:k} \mid z_{1:k})$, donde $\{x^i_{0:k}, i = 0, \cdots, N_s\}$ es un conjunto de muestras con pesos asociados $\{w^i_k, i = 1, \cdots, N_s\}$ y $x_{0:k} = \{x_j, j = 0, \cdots, k\}$ es el conjunto de todos los estados hasta el tiempo k. Los pesos están

normalizados de tal forma que $\sum_i w_k^i = 1$. La densidad posterior en k puede ser aproximada como en la Ecuación 3.2.

$$p(x_{0:k} \mid z_{1:k}) \approx \sum_{i=1}^{N_s} w_k^i \delta(x_{0:k} - x_{0:k}^i) \tag{3.2}$$

donde la función δ es la función de Dirac.

En otras palabras, se seleccionan aleatoriamente un conjunto de muestras $x_{0:k}^i$ obtenidas de una distribución fácilmente muestreable. Las muestras seleccionadas son aceptadas si son parte de la distribución posterior buscada (razón del uso de la función de Dirac) en base a la ponderación de la muestra.

Este es un enfoque no paramétrico, y por tanto puede manejar distribuciones no lineales, multimodales, etc. La ventaja sobre la discretización es que el método es adaptativo, colocando más partículas (correspondiendo a una discretización más fina) en lugares donde la densidad probabilista es mayor [Murphy, 2002].

Los pesos son elegidos utilizando el principio de *Importancia del Muestreo* [Maskell and Gordon, 2002]. Este principio se fundamenta en lo siguiente: Suponiendo $p(x) \propto \pi(x)$, donde $p(x)$ es una densidad de probabilidad difícil de muestrear, pero para la cual $\pi(x)$ puede ser fácilmente evaluada (y por tanto $p(x)$ por proporcionalidad). También sea $x^i \sim q(x), \{i = 1, \cdots, N_s\}$ muestras que son fácilmente generadas por una distribución propuesta $q(x)$, llamada una *densidad de importancia*. Entonces, una aproximación ponderada a la densidad $p(x)$ esta dada por:

$$p(x) \approx \sum_{i=1}^{N_s} w^i \delta(x - x^i) \tag{3.3}$$

donde

$$w^i \propto \frac{\pi(x^i)}{q(x^i)} \tag{3.4}$$

es el peso normalizado de la i-ésima partícula.

Por lo tanto, si las muestras, $x_{0:k}^i$, son esbozadas a partir de una densidad de importancia, $q(x_{0:k} \mid z_{1:k})$, entonces los pesos en 3.2 definidos en 3.4 serán

$$w_k^i \propto \frac{p(x_{0:k}^i \mid z_{i:k})}{q(x_{0:k}^i \mid z_{1:k})} \tag{3.5}$$

La Figura 3.4 muestra lo anterior de manera gráfica. La curva gaussiana representa la distribución propuesta $q(x)$, con la que se obtendrán las muestras. La curva multimodal es la distribución que $\pi(x)$ que es proporcional a la distribución buscada, y con ella se evalúan las muestras obtenidas. Las flechas son las partículas muestreadas y su longitud representan el peso que se le ha dada a cada una. Se puede apreciar que se han generado más partículas en el centro de la distribución propuesta, pero con bajo peso, mientras que en los extremos son más escasas pero con mayor ponderación.

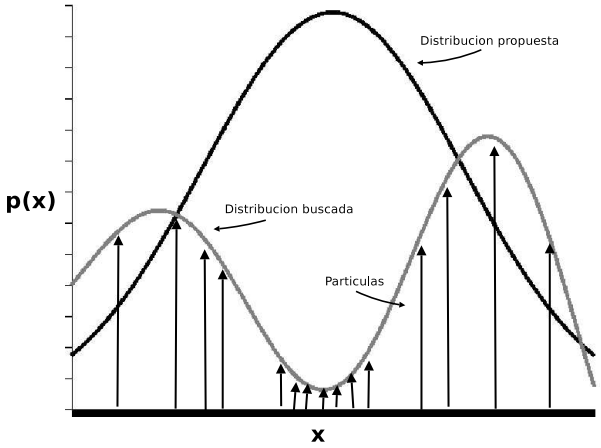


Figura 3.4: Funcionamiento general del filtro de partículas en una distribución conjunta no secuencial.

Para el caso de un cálculo de densidad secuencial, como un filtro bayesiano, a cada iteración se deberán tener muestras que constituyen una aproximación a $p(x_{0:k-1} \mid z_{1:k-1})$, y buscar una aproximación $p(x_{0:k} \mid z_{1:k})$ utilizando un nuevo conjunto de muestras. Si la densidad de importancia se selecciona para factorizarse de tal manera que:

$$q(x_{0:k} \mid z_{1:k}) = q(x_k \mid x_{0:k-1}, z_{1:k}) q(x_{0:k-1} \mid z_{1:k-1}) \tag{3.6}$$

entonces se pueden obtener muestras $x^i_{0:k} \sim q(x_{0:k} \mid z_{1:k})$ aumentando cada una de las muestras existentes, $x^1_{0:k-1} \approx q(x_{0:k-1} \mid z_{1:k-1})$, con el nuevo estado $x^i_k \approx q(x_k \mid k_{0:k-1}, z_{1:k})$.

Para poder deducir la ecuación de actualización de pesos, la distribución $p(x_{0:k} \mid z_{1:k})$ debe expresarse primero en términos de $p(x_{o:k-1} \mid z_{1:k-1})$, $p(z_k \mid x_k)$ y $p(x_k \mid x_{k-1})$:

Se puede proponer $p(x_{0:k} \mid z_{1:k})$ utilizando la regla de Bayes para actualizar su estado a partir de estado anterior de la siguiente manera:

$$p(x_{0:k} \mid z_{1:k}) = \frac{p(z_k \mid x_{0:k}, z_{1:k-1})p(x_{0:k-1} \mid z_{1:k-1})}{p(z_k \mid z_{1:k-1})} \tag{3.7}$$

donde la constante de normalización es

$$p(z_k \mid z_{1:k-1}) = \int p(z_k \mid x_k)p(x_k \mid x_{1:k-1})dx_k$$

Suponiendo que $p(x_{0:k-1} \mid z_{1:k-1})$ se puede factorizar igual que en 3.6, entonces

$$p(x_{0:k} \mid z_{1:k}) = \frac{p(z_k \mid x_{0:k}, z_{1:k-1})p(x_k \mid x_{0:k-1}, z_{1:k-1})p(x_{0:k-1} \mid z_{1:k-1})}{p(z_k \mid x_{0:k}, z_{1:k-1})} \tag{3.8}$$

Debido a que las mediciones z son independientes dado el estado x y expresando el filtro de manera recursiva, entonces

$$\begin{aligned} p(x_{0:k} \mid z_{1:k}) &= \frac{p(z_k \mid x_k)p(x_k \mid x_{k-1})}{p(z_k \mid z_{1:k-1})}p(x_{0:k-1} \mid z_{1:k-1}) \\ &\propto p(z_k \mid x_k)p(x_k \mid x_{k-1})p(x_{0:k-1} \mid z_{1:k-1}) \end{aligned} \tag{3.9}$$

Sustituyendo 3.6 y 3.9 en 3.5, la ecuación de actualización de pesos puede ser mostrada como:

$$\begin{aligned} w_k^i &\propto \frac{p(z_k \mid x_k^i)p(x_k^i \mid x_{k-1}^i)p(x_{0:k-1}^i \mid z_{1:k-1})}{q(x_k^i \mid x_{0:k-1}^i, z_{1:k})q(x_{0:k-1}^i \mid z_{1:k-1})} \\ &= w_{k-1}^i \frac{p(z_k \mid x_k^i)p(x_k^i \mid x_{k-1}^i)}{q(x_k^i \mid x_{0:k-1}^i, z_{1:k})} \end{aligned} \tag{3.10}$$

Además, si $q(x_k \mid x_{0:k-1}, z_{1:k}) = q(x_k \mid x_{k-1}, z_k)$, entonces la densidad de importancia se vuelve sólo dependiente en la x_{k-1} y z_k. Esto es particularmente útil en el caso común cuando sólo una estimación filtrada de $p(x_k \mid z_{1:k})$ es requerida a cada instante. A partir de este punto se asume tal caso. El peso modificado es entonces:

$$\begin{aligned} w_k^i &\propto w_{k-1}^i \frac{p(z_k \mid x_k^i) p(x_k^i \mid x_{k-1}^i)}{q(x_k^i \mid x_{k-1}^i, z_k)} \\ &\stackrel{\text{def}}{=} \hat{w}_t^i \times w_{t-1}^i \end{aligned} \tag{3.11}$$

donde $\hat{w}_t^i$ se define como el peso incremental.

La densidad posterior filtrada $p(x_k \mid z_{1:k})$ puede ser aproximada como:

$$p(x_k \mid z_{1:k}) \approx \sum_{i=1}^{N_s} w_k^i \delta(x_k - x_k^i) \tag{3.12}$$

donde los pesos están definidos en 3.11. Puede demostrarse que si el número de muestras $N_s \to \infty$ la aproximación 3.12 se acerca a la verdadera densidad posterior $p(x_k \mid z_{1:k})$.

El algoritmo SIS, por tanto, consiste en una propagación recursiva de los pesos y las muestras tal como cada medición es recibida de manera secuencial. Una descripción en pseudo-código se ve en el Algoritmo 3.1.

Algoritmo 3.1 Algoritmo del filtro de partículas SIS

for $i = 1$ hasta N_s **do**
  Proponer partícula $x_k^i \sim q(x_k \mid x_{k-1}^i, z_k)$
  Ponderar partícula, w_k^i de acuerdo a 3.11
end for

3.3.1. El problema del empobrecimiento de la partícula

Un problema común con el filtro de partículas SIS es el fenómeno de la degeneración o empobrecimiento. Después de varias iteraciones todas las partículas, excepto una, tendrán un peso despreciable. Se ha demostrado que la varianza de las ponderaciones de importancia pueden sólo incrementarse con el tiempo, y por tanto es imposible evitar este fenómeno. Esta empobrecimiento implica que un gran esfuerzo computacional, dedicado a la actualización de partículas cuya contribución es la aproximación de siguiente estado $p(x_k \mid z_{1:k})$, es casi inútil. Una medición aceptable de la degeneración del algoritmo es el tamaño efectivo de la muestra N_{eff}, y se define como:

$$N_{eff} = \frac{N_s}{1 + Var(w_k^{*i})} \tag{3.13}$$

donde $w_k^{*i} = p(x_k^i \mid z_{1:k})/q(x_k^i \mid x_{k-1}^i, z_k)$ es conocida como "el verdadero peso". Como esto no puede ser evaluado exactamente, se hace una estimación $\widehat{N_{eff}}$ de N_{eff} que se obtiene con

$$\widehat{N_{eff}} = \frac{1}{\Sigma_{i=1}^{N_s}(w_k^i)^2} \tag{3.14}$$

donde w_k^i es el peso normalizado obtenido con 3.10. Se advierte que $N_{eff} \leq N_s$, y un N_{eff} pequeño indica un empobrecimiento severo. Claramente, el problema de la degeneración es un efecto indeseado de los filtros de partículas. Hay varios enfoques para evitar este problema:

- Utilizar un N_s muy grande (fuerza bruta). Es muy poco práctico.
- La elección de una buena densidad de importancia que minimice $Var(w_k^{*i})$. En [Grisetti *et al.*, 2005] hacen uso de este enfoque para reducir el número de partículas necesarias en el problema del SLAM.
- El remuestreo, que es la técnica que se utilizará en esta tesis.

3.3.2. Remuestreo

Este método para reducir los efecto del empobrecimiento se basa en usar remuestreo siempre que una degeneración significativa es observada (cuando N_{eff} llega a ser menor que un umbral, N_T). La idea básica del remuestreo es eliminar partículas que tienen pesos bajos y concentrarse en las partículas con pesos altos. El paso del remuestreo involucra la generación de un nuevo conjunto $\{x_k^{i*}\}_{i=1}^{N_s}$ al remuestrear (con reemplazo) N_s veces a partir de una representación discreta aproximada de $p(x_k \mid z_{1:k})$ dada por

$$p(x_k \mid z_{1:k}) \approx \sum_{i=1}^{N_s} w_k^i \delta(x_k - x_k^i) \tag{3.15}$$

de tal forma que $Pr(x_k^{i*} = x_k^j) = w_k^j$. La muestra resultante es de hecho una distribución muestra independiente e idéntica de la distribución discreta 3.15, y por lo tanto los pesos ahora se reasignan a $w_k^i = 1/N_s$. Es posible implementar este procedimiento de remuestreo en $O(N_s)$ operaciones usando una variedad de métodos [Murphy, 2002].

Uno de los más conocidos es el remuestreo sistemático [Maskell and Gordon, 2002], y su operación se describe en el Algoritmo 3.2, donde $\mathcal{U}[a,b]$ es la distribución uniforme en el intervalo $[a,b]$. Por cada partícula remuestreada x_k^{j*}, este algoritmo de remuestreo también almacena el índice de su padre, denotado por i^j.

Algoritmo 3.2 Algoritmo de remuestreo sistemático

Inicializar la función de distribución conjunta: $c_1 = 0$
for $i = 2$ hasta N_s **do**
  Construye la función de distribución conjunta: $c_i = c_{i-1} + w_k^i$
end for
Inicia en el fondo de la función de distribución conjunta: $i = 1$
Proponer un punto de inicio: $u_1 \approx \mathcal{U}[0, N_s^{-1}]$
for $j = 1$ hasta N_s **do**
  Moverse a través de la función de distribución conjunta: $u_j = u_1 + N_s^{-1}(j-1)$
  while $u_j < c_i$ **do**
    $*i = i + 1$
  end while
  Asignar muestra: $x_k^{j*} = x_k^i$
  Ponderar: $w_k^j = N_s^{-1}$
  Asignar padre: $i^j = i$
end for

Aunque el paso de remuestreo reduce los efectos del problema de degeneración, introduce otros problemas prácticos:

1. Limita la oportunidad de paralelizar el proceso ya que todas las partículas deben ser combinadas.

2. Las partículas que tiene altos pesos w_k^i son estadísticamente seleccionadas muchas veces. Esto conduce a una pérdida de diversidad entre las partículas y la muestra resultante contendrá muchos puntos repetidos. Este problema, conocido como *empobrecimiento de la muestra*, se agrava en el caso cuando hay ruido ligero en el proceso. Esta es la razón por la cual no se hace el remuestreo a cada iteración.

3. Ya que la diversidad en las rutas de los estados seguidos por las partículas se reduce, cualquier estimación basada en dicha ruta se degenera.

Existen esquemas para contrarrestar este efecto, el más utilizado es manejar una distribución propuesta que ofrezca una buena diversidad en las muestras. Un filtro de partículas genérico con remuestreo se describe en el Algoritmo 3.3.

Algoritmo 3.3 Algoritmo genérico del filtro de partículas SISR

for $i = 1$ hasta N_s **do**
  Proponer una partícula $x_k^i \sim q(x_k \mid x_{k-1}^i, z_k)$
  Ponderar partícula, w_k^i de acuerdo a 3.11
end for
Calcular el peso total: $t = \sum_{i=1}^{N_s} w_k^i$
for $i = 1$ hasta N_s **do**
  Normalizar: $w_k^i = w_k^t / t$
end for
Calcular $\widehat{N_{eff}}$ usando 3.14
if $\widehat{N_{eff}} < N_T$ **then**
  Remuestrear usando el algoritmo 3.2
end if

En resumen, se pueden ver los filtros de partículas como un proceso de dos partes [Jordan, 2002]: Al instante k se hace una representación de la distribución posterior $p(x_k \mid z_{1:k})$ expresada como muestras $\{x_k^i\}$ con sus pesos correspondientes $\{w_k^i\}$. Para obtener la representación correspondiente al siguiente instante, se esbozan N_s muestras de la distribución propuesta, y luego por cada muestra se usa la nueva observación z_{k+1} para evaluar los pesos correspondientes $w_{k+1}^i \propto p(z_{k+1} \mid x_{k+1}^i)$. Esto se puede observar de manera esquemática en la Figura 3.5.

3.3.3. Filtro de partículas Rao-Blackwellizados

Los filtros de partículas pueden ser muy ineficientes en espacios de alta dimensionalidad. Una técnica estándar para incrementar la eficiencia de los métodos de muestreo es reducir el tamaño del espacio de estados marginalizando alguna de las variables analíticamente; a esto se le llama Rao-Blackwellización [Murphy and Russell, 2001]. Combinando ambas técnicas se obtiene un filtro de partículas Rao-Blackwellizado (RBPF por sus siglas en inglés).

El teorema de Rao-Blackwell demuestra cómo mejorar cualquier estimador dado bajo cualquier función convexa [Murphy, 2002]. Su núcleo es la siguiente identidad:

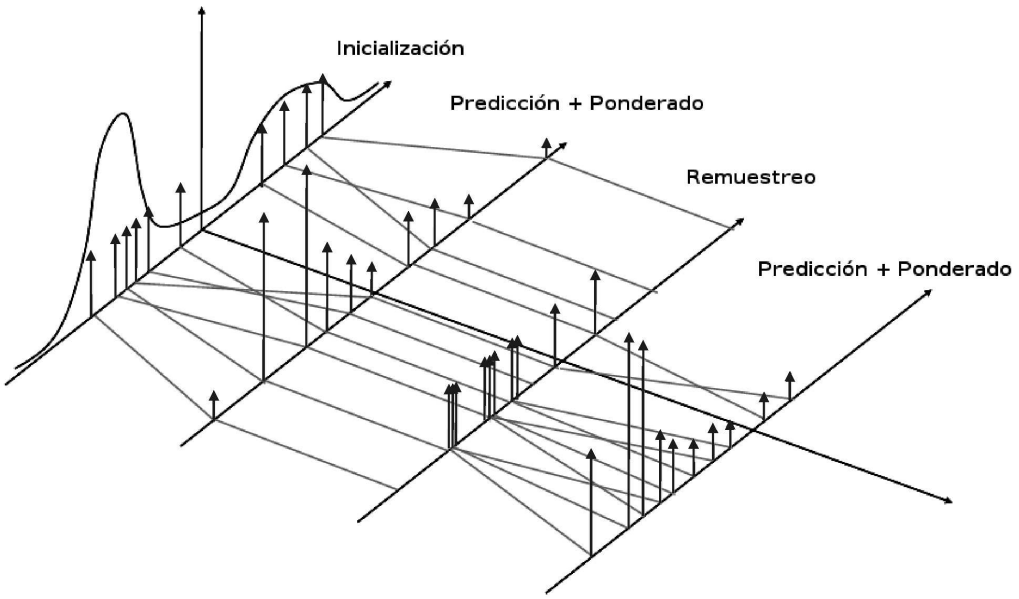


Figura 3.5: Esquema general de la operación de un filtro de partículas secuencial [Chatila, 2005].

$$Var[\tau(X,R)] = Var[E(\tau(X,R) \mid R)] + E[Var(\tau(X,R) \mid R)]$$

donde $\tau(X,R)$ es algún estimador de X y R. Por lo tanto $Var[E(\tau(X,R) \mid R)] \leq Var[\tau(X,R)]$, así que $\tau'(X,R) = E(\tau(X,R) \mid R)$ es un estimador de varianza menor. Por lo que si es posible muestrear R y calcular el valor esperado de X dado R analíticamente, resultan necesarias menos muestras (para una mayor precisión). Desde luego, menos muestras no implican necesariamente menor tiempo, eso depende de si es posible calcular el valor esperado condicional o no.

La idea principal es dividir el espacio de estados Z_k en dos subespacios, R_k y X_k, de tal manera que la distribución $P(X_k \mid R_{1:k}, y_{1:k})$ pueda ser actualizada analíticamente y eficientemente; la distribución $P(R_{1:k} \mid y_{1:k})$ es actualizada usando filtros de partículas. La justificación para esta descomposición proviene de la regla de la cadena:

$$P(X_{1:k}, R_{1:k} \mid y_{1:k}) = P(X_{1:k} \mid R_{1:k}, y_{1:k}) P(R_{1:k} \mid y_{1:k})$$

Muestreando únicamente R_t generalmente requerirá un mucho menor número de partículas (para alcanzar un umbral de precisión fija) que el filtrado de partículas genérico, el cual muestreará de manera simultánea R_k y X_k [Murphy and Russell, 2001].

Los RBPF son muy similares a los filtros de partículas genéricos, a excepción de que cada partícula ahora mantiene no sólo una muestra de $P(R_{1:k} \mid y_{1:k})$, la cual se denota como $r^i_{1:k}$, sino también una representación paramétrica de $P(X_k \mid r^i_{1:k}, y_{1:k})$, la que se llamará α^i_k. (La representación paramétrica será un vector de promedio y una matriz de covarianza, por ejemplo.) Las muestras R_k son actualizados como un filtro de partículas estándar y luego las distribuciones X_k serán actualizadas usando un filtro exacto, condicional en R_k. El proceso general se muestra en el Algoritmo 3.4.

Algoritmo 3.4 Algoritmo Genérico de un Filtro de Partículas Rao-Blackwellizado

Muestreo secuencial con importancia

for $i = 1$ hasta N **do**

Muestrea $(\hat{r}^i_k) \sim q(r_k \mid r^i_{1:k-1}, y_{1:k})$

Asigna $(\hat{r}'^i_{1:k}) \stackrel{\text{def}}{=} (\hat{r}^i_k, r^i_{1:k-1})$

end for

Evaluar los pesos de importancia

Normalizar los pesos

Remuestrea N muestras de $(\hat{r}'^i_{1:k})$ de acuerdo a la distribución de importancia $\tilde{w}^i_k$ para obtener N muestras aleatorias $(r^i_{1:k})$ distribuidas aproximadamente a $p(r^i_{1:k} \mid y_{1:k})$

Cálculo analítico $\rightarrow$ Actualizar α^i_k dado α^i_{k-1}, r^i_k, r^i_{k-1} y y_k.

3.4. Filtros de partículas aplicados en SLAM

Como ya se analizó, el algoritmo del filtro de partículas es recursivo y opera en dos fases: la *predicción* y la *actualización*. En este aspecto se asemeja mucho a los filtros Kalman estudiados en la Sección 3.1.1. La predicción está en el modelo de movimiento del robot, generando las N partículas. La actualización está en la ponderación de las partículas generadas, utilizando para ello el modelo de observación.

Para resolver el problema del SLAM, cada partícula en un filtro de partículas Rao-Blackwellizado representa una trayectoria posible del robot y un mapa, y no únicamente la posición actual, como en los EKFs [Thrun, 2002a]. La idea principal es estimar un distribución posterior $p(s_{1:t} \mid z_{1:t}, u_{0:t})$ sobre las trayectorias potenciales $s_{1:t}$ de un robot dado su observaciones $z_{1:t}$ y sus mediciones de odometría $u_{0:t}$. Se utiliza esta distribución posterior para calcular una distribución posterior de mapas y trayectorias [Grisetti *et al.*, 2005]:

$$p(s_{1:t}, m \mid z_{1:t}, u_{0:t}) = p(m \mid s_{1:t}, z_{1:t}) p(s_{1:t} \mid z_{1:t}, u_{0:t})$$

Lo anterior puede hacerse de manera eficiente ya que la distribución posterior sobre mapas $p(m \mid s_{1:t}, z_{1:t})$ puede calcularse analíticamente, dado por conocidos $s_{1:t}$ y $z_{1:t}$. Además esto puede hacerse gracias a que las trayectorias y los mapas son condicionalmente independientes [Thrun, 2002a].

Para estimar la distribución posterior $p(s_{1:t} \mid z_{1:t}, u_{0:t})$ sobre las trayectorias potenciales, se utiliza un filtro de partículas en donde un mapa individual se asocia a cada muestra. Cada mapa es construido dadas las observaciones $z_{1:t}$ y las trayectorias $s_{1:t}$ representadas por la partícula correspondiente. Dicho de otra manera, se aproximará la distribución posterior en el tiempo t usando un conjunto de partículas ponderadas, donde cada partícula especifica una trayectoria $S_{1:t}$ y la correspondiente representación condicionalmente factorizada de $P(m_t)$ [Murphy, 1999].

A continuación se presenta una revisión de cómo trabajan las fases de predicción y actualización con filtros de partículas en la resolución del problema de la cartografía con robots móviles. Primero se revisa de manera genérica, y en seguida se muestra un ejemplo simplificado siguiendo la implementación de *Simple Mapping Utilities* (PMAP) [Howard, 2004], la cual usa una representación del ambiente de rejilla de ocupación. El modelo de observación se hace con la información del telémetro láser y su modelo de movimiento está parametrizado para robots diferenciales como el Pioneer 2AT.

3.4.1. El modelo de movimiento (Predicción)

En la fase de *predicción*, después de cada acción de movimiento del robot, se reproduce el conjunto de partículas utilizando una distribución de fácil muestreo, que se le conoce como *distribución propuesta*, la cual se construye a partir del *modelo de movimiento* del robot.

Se comienza con el algoritmo del filtro de partículas estimando la trayectoria posterior $p(s_t \mid s_{t-1}, u^t)$ del modelo de movimiento, manteniendo un conjunto de partículas que representan la distribución. Cada partícula $s_t^m \in S_t$ representan trayectorias supuestas del robot [Montemerlo *et al.*, 2002a]. Estas partículas resultan ser hipótesis de las distintas posibles nuevas posiciones de observación durante una trayectoria recorrida. Se puede suponer que el conjunto de partículas S_{t-1} está distribuida de acuerdo a $p(s_{1:t-1} \mid z_{1:t-1}, u_{0:t-1})$, lo que es una aproximación asintóticamente correcta.

En el sistema PMAP, el modelo de movimiento esta dado por lo que reporta la odometría, como el cambio de posición de pos_i a pos_{i+1}, dado el comando de movimiento u_i. Este cambio de posición es un Δpos. La distribución propuesta esta basada en distribuciones normales, una por cada grado de libertad, con media igual a cero y varianzas proporcionales a Δpos. Así, para cada partícula, la distribución de probabilidad $p(s_t \mid s_{t-1}, u^t)$ esta dada por

$$\begin{aligned} \Delta pos_t^j.x &= \mathcal{N}(0, k_x * \Delta pos_{t-1}^j.x) \\ \Delta pos_t^j.y &= \mathcal{N}(0, k_y * \Delta pos_{t-1}^j.y) \\ \Delta pos_t^j.\theta &= \mathcal{N}(0, k_\theta * \Delta pos_{t-1}^j.\theta) \end{aligned}$$

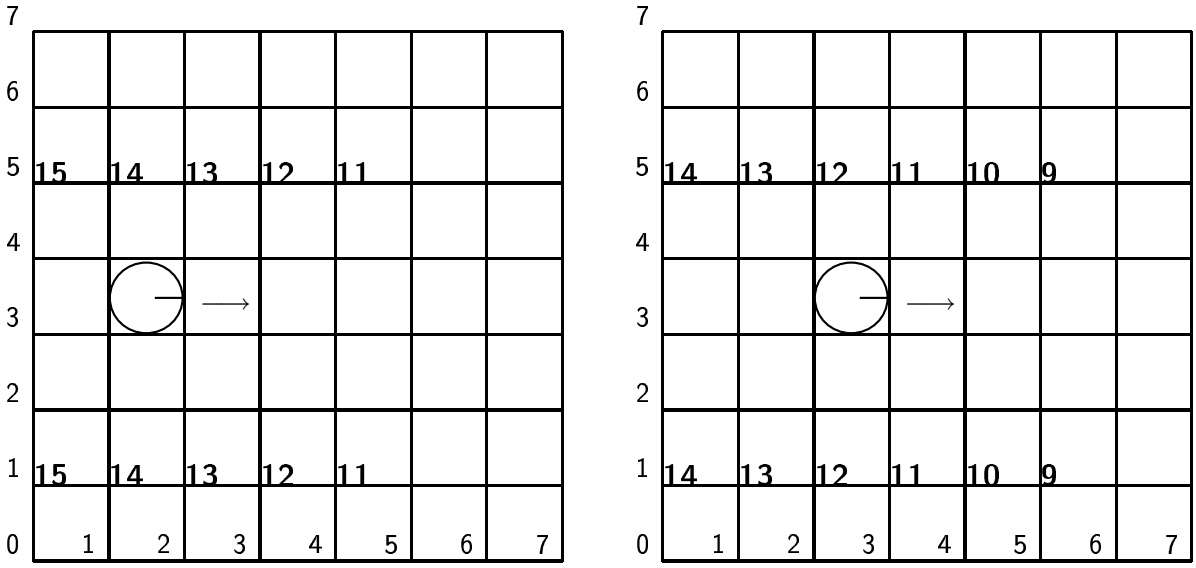
donde las k_i son coeficientes *ad hoc* para la distribución de acuerdo al comportamiento observable del robot.

El Δpos_t^j se le suma a la posición actual (en el tiempo t) del robot dentro de la partícula j, dando la nueva posición en la partícula. La posición anterior ($t-1$) pasa a ser parte de la trayectoria seguida por dicha partícula.

Un ejemplo simplificado puede verse en la Figuras 3.6 y 3.7. Es un ambiente, donde las celdas son del mismo tamaño que el robot, el cual cuenta con un sensor láser que le permite ver a 180°hacia el frente. El robot recorre lo que típicamente sería un pasillo. En este ejemplo se utilizan únicamente dos partículas.

En la Figura 3.6 se muestran las dos partículas que se mantienen en el tiempo actual: $Pos_t^1 = [2, 3, 0°]$ y $Pos_t^2 = [3, 3, 0°]$. Se aplica entonces un comando de movimiento y el robot se desplaza hacia adelante. Utilizando la diferencia entre la posición actual en la partícula y la distancia reportada por la odometría, se genera una nueva hipótesis por cada partícula utilizando la distribución propuesta: $Pos_{t+1}^1 = [4, 3, 0°]$ y $Pos_{t+1}^2 = [4, 4, 0°]$ (Figura 3.7).

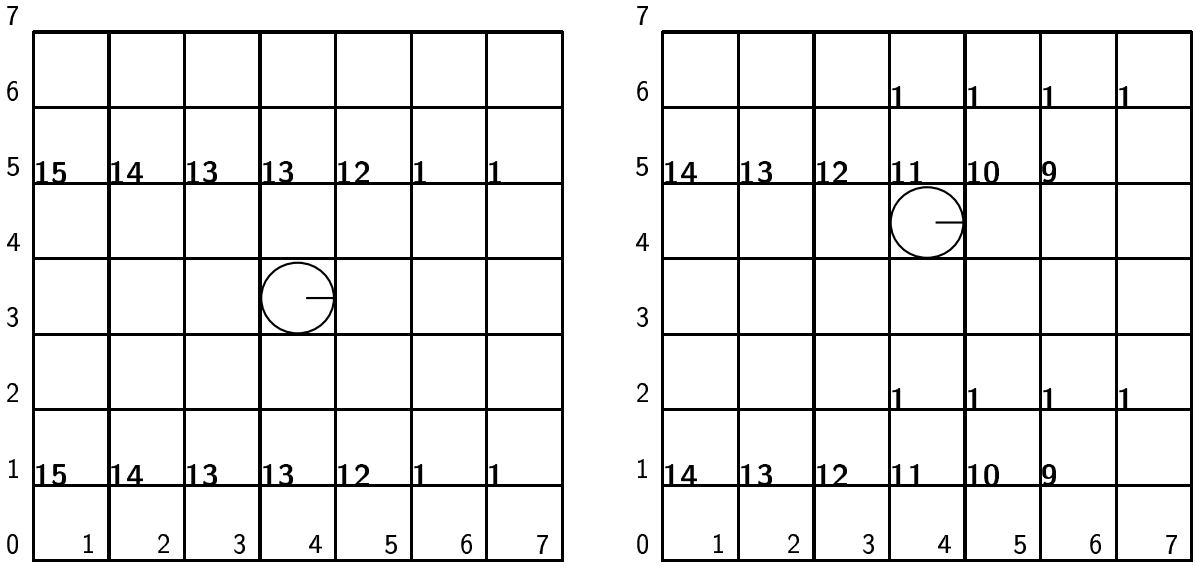
Después de renovar la posición en las N partículas, el nuevo conjunto S_t se obtiene del conjunto temporal de partículas. Cada partícula s_t^n se esboza (con reemplazo) con una probabilidad proporcional al llamado *factor de importancia* w_t^n. Este factor de importancia es calculado en base al modelo de observación.



(a) Partícula #1 en tiempo t

(b) Partícula #2 en tiempo t

Figura 3.6: Dos partículas con la posición en tiempo t del robot y sus percepciones previas.



(a) Partícula #1 en tiempo $t+1$

(b) Partícula #2 en tiempo $t+1$

Figura 3.7: Dos partículas en tiempo $t+1$ dadas la distribución propuesta y sus percepciones actuales.

3.4.2. El modelo de observación (Actualización)

En la fase de *actualización*, cada partícula se pondera de acuerdo al *modelo de observación*. Esta ponderación esta definida por:

$$w_t^m = \frac{\text{distribución objetivo}}{\text{distribución propuesta}} = \frac{p(s_{1:t}^m \mid z_{1:t}, u_{0:t})}{q(s_{1:t}^m \mid z_{1:t}, u_{0:t})} \tag{3.16}$$

Como ya se dijo, el denominador está dado por una aproximación del modelo de movimiento hecho a partir el modelo de observación, mientras que el numerador es la distribución buscada. La división anterior es intratable directamente, pero, como se probó anteriormente, el cálculo del peso de las partículas puede hacerse de manera recursiva [Murphy, 1999]:

$$w_{t+1}^j = v_{t+1}^j + w_t^j$$

La ecuación anterior indica que el nuevo peso de la j-ésima partícula en el nuevo instante es igual a su peso en el instante anterior más el peso incremental $v_{t+1}^j \propto p(z_{t+1} \mid s_{t+1}, m_t^j)$, que es una función de vecindad [Grisetti *et al.*, 2005], donde se obtiene la probabilidad de las observaciones dado el mapa anterior contenido en la j-ésima partícula y la nueva posición, es decir la probabilidad donde concuerden las lecturas con el mapa propuesto. Por la evaluación de la función de vecindad, cada partícula debe tener una representación del ambiente de manera independiente, haciendo que este algoritmo pueda requerir grandes cantidades de memoria.

En la práctica, PMAP implementa la ponderación de las partículas de la siguiente manera: cada haz de luz del telémetro láser marca una celda como ocupada, dado que su distancia de respuesta fue menor a su máximo creíble, suma una unidad al valor de dicha celda. Razón por la que se observa números en las celdas ocupadas en las Figuras 3.6 y 3.7. Mientras más cercanas las celdas están del robot, más veces se han registrado como ocupadas. PMAP mantiene un umbral superior a 127 en el número de veces de observada una celda como ocupada; cuando una celda ha sido observada como ocupada más de 127 veces, ya no se marcan aumentos de observación.

Al inicio del programa, inicializa una tabla de búsqueda de los vecinos cercanos a la celda a evaluar. El tamaño de la tabla está en función de la razón entre el tamaño de la celda (resolución del mapa) y una constante de error máximo permitido en el desplazamiento de las lecturas, que resulta ser del tamaño del diámetro del robot (en el caso de PMAP es constante a 50 centímetros). La tabla de búsqueda de vecinos entonces cubre una área de

-0.50 a +0.50 metros. El contenido de dicha tabla sería la distancia del centro, a la celda dentro de la área marcada. En el caso del ejemplo mostrado en las Figuras 3.6 y 3.7, donde la resolución de la celda es igual al tamaño del robot (0.50 metros), la tabla resulta ser de 3 columnas por 3 renglones de un metro distancia cada elemento de la búsqueda (0.50 de resolución / 0.50 de error máximo), y su contenido es:

$$\begin{array}{r|ccc} & -1 & 0 & +1 \\ \hline -1 & 1.41 & 1 & 1.41 \\ 0 & 1 & 0 & 1 \\ +1 & 1.41 & 1 & 1.41 \end{array} \tag{3.17}$$

Después se ordena la tabla como si fuera un vector de la menor distancia a la mayor, obteniendo:

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1.41 & 1.41 & 1.41 & 1.41 \end{pmatrix} \tag{3.18}$$

Finalmente se podan todos los vecinos que no cumplan la condición $\text{distancia} * \text{tamaño de celda} < \text{error máximo}$. En el ejemplo resulta entonces un arreglo de un sólo elemento: $\begin{pmatrix} 0 \end{pmatrix}$. Esto indica que sólo se evaluará la celda donde incide el haz de luz del láser. Sin embargo, con fines demostrativos, se cambia su valor de distancia de 0 a 1 para poder evaluarlo.

Cuando se recibe una lectura del láser que corresponde a la nueva posición del robot, por cada haz de luz, cuya distancia reportada no sea mayor a la máxima permitida, se recorre la lista vecinos más cercanos y se va analizando de las celdas vecinas más cercanas a las más lejanas. Si una de esas celdas tiene un valor de ocupación con un umbral mayor a 8, se almacena temporalmente su distancia reportada en la tabla de búsqueda y se continúa con el siguiente vecino. Si ninguna celda vecina tiene un valor mayor de 8, entonces el valor por defecto de la distancia es 1000.

Terminada de recorrer la tabla de vecinos, se tiene la distancia a la celda con un valor de ocupación mayor a 8 más lejana de la celda de incidencia del haz de luz del láser. Dicha distancia se multiplicará por el tamaño de la celda y se suma en un acumulador que lleva el error total de la partícula en el tiempo actual $(t+1)$. La operación se repite para cada celda marcada por cada haz de luz de la lectura del láser. El peso de la partícula es igual al cuadrado de los errores calculados en la partícula más su valor en el tiempo anterior. El algoritmo 3.5 esquematiza lo antes dicho.

Algoritmo 3.5 Algoritmo del ponderado de la partícula en PMAP

```
error = 0
error máximo = 0.5
for all haz de láser do
  P = Celda sobre la que incide el haz
  mdist = 1000;
  for all Vecinos de P según 3.18 do
    V = Celda vecina de P
    if valor de ocupación de V > 8 then
      mdist = distancia de V a P
      break (termina ciclo)
    end if
  end for
  error = error + mdist * tamaño de celda
  if error > error máximo then
    error = error máximo
  end if
end for
peso de la partícula = peso de la partícula + error * error
```

En el caso de las dos partículas nuevas de ejemplo, si se considera que el error previo es de 0 en ambos casos, las partículas se ponderarían así de acuerdo al Algoritmo 3.5, suponiendo que se tiene 8 láseres incidentes en obstáculos, el resto al no incidir en obstáculos no se analizan:

$$\begin{aligned} Err^1_{t+1} &= 0*0.5+0*0.5+1*0.5+1000*0.5+ \\ &\quad 0*0.5+0*0.5+1*0.5+1000*0.5 \\ &= 0+0+0+0.5+0.5+0+0+0.5+0.5 \\ &= 2.0 \\ Err^2_{t+1} &= 1*0.5+1*0.5+1*0.5+1.41*0.5+ \\ &\quad 1*0.5+1*0.5+1*0.5+1.41*0.5 \\ &= 0+0.5+0.5+0.5+0.705+0.5+0.5+0.5+0.705 \\ &= 4.41 \end{aligned}$$

Como se observa, el error en la medición del partícula 2 es mayor que la partícula 1, por lo que tiene mejor ponderación es la partícula 1, siendo entonces que la trayectoria por la partícula 1 es la más probable.

Básicamente la ponderación de partículas es la medida de correlación de las celdas ocupadas dentro de una ventana cuyo tamaño es dos veces el error factible (en este caso, el diámetro del robot). Esta correlación esta dada por el histograma de ocupación de una celda y sus vecinos dentro de la ventana. Es por esto que el error puede luego ser convertido a una probabilidad normalizando sus valores.

3.4.3. Remuestreo

En ciertos momentos las partículas con pesos muy pequeños son eliminados, que es el proceso de remuestreo [Rekleitis, 2004], y se reproducen las partículas con altos pesos. Para elegir el momento de hacer un remuestreo se puede calcular, tal como se dijo en el punto 3.3.2, el tamaño efectivo de la muestra, observando la varianza entre las partículas; aunque también se pueden tomar otras estrategias, como utilizar un umbral de iteraciones o una combinación de ambas.

En el caso del PMAP, se hace una combinación de valores para saber si se debe remuestrear o no: si el tamaño efectivo de la muestra es menor a un umbral arbitrario (0.8), o si han transcurrido un número fijo de pasos en la trayectoria sin remuestreado (10 o algún otro fijado por el usuario).

Para obtener el tamaño efectivo de la muestra, se convierten los pesos de todas las partículas en probabilidades normalizadas, se observa su varianza y se aplica la Ecuación 3.14. Cuando se ha decidido remuestrear, se toman entonces las partículas con las mejores ponderaciones vistas como probabilidades normalizadas y sus trayectorias y mapas internos se copian a las que tienen menor ponderación, no así sus posiciones actuales, estas se mantienen.

3.4.4. Actualización del Mapa

Obtenidas las trayectorias dado el modelo de movimiento y de percepción, es posible calcular analíticamente el mapa del ambiente explorado, debido a que, como ya se ha dicho, se utiliza un filtro de partículas Rao-Blackwellizado. Aquí se usan los algoritmos necesarios para construir la representación del ambiente, dependiendo del tipo de mapa utilizado. En el caso del enfoque empleado en esta tesis, la de rejillas de ocupación, estos algoritmos se

verán en el Capítulo 4, en la sección de Fusión Sensorial.

Aunque el algoritmo del filtro de partículas es común entre diversos algoritmos para resolver el problema del SLAM, cada implementación maneja su representación del ambiente de la manera más conveniente para sus autores, esto conlleva a que el manejo de la ponderación de las partículas dado el modelo de observación varíe entre las implementaciones. Sin embargo, es posible establecer que, de manera general, los algoritmos de SLAM como PF siguen la estructura mostrada en el Algoritmo 3.6 [Murphy, 1999].

Algoritmo 3.6 Algoritmo General del SLAM con filtros de partículas

for all Comando de movimiento **do**
  Muestreo de s_t con la distribución propuesta del modelo de movimiento.
  Ponderar cada partícula dado s_t^i dada correlación de la nuevas mediciones con las ya hechas.
  Actualizar cada componente del mapa dentro de cada partícula usando s_t^i y las mediciones z_t.
  Calcular el número efectivo de muestras.
  if $N_{eff} < Umbral$ ó contador último remuestreo $> Umbral$ **then**
    Remuestrear las partículas
  end if
end for
Generar mapa con las trayectorias S y las mediciones Z.

3.4.5. Problemas del enfoque con filtro de partículas

El problema del enfoque presentado es determinar el número de partículas a utilizar para obtener la representación de un ambiente. El espacio de mapas posibles es vasto, y el conjunto de partículas necesariamente debe representar un muestreo muy disperso de este espacio. Entonces se debe especificar un número de partículas que represente este espacio de mapas posibles. Mientras más intrincado sea el ambiente, con ciclos, pasillos, más grande será el número necesario de partículas.

Mantener un filtro de partículas sobre todos los mapas posibles hace uso muy intensivo de la memoria, ya que cada partícula almacena un mapa completo. Hay que tener mucho cuidado en la relación número de partículas / tamaño del ambiente, porque fácilmente se puede exceder en la memoria física de la computadora.

El algoritmo actualiza constantemente las lecturas nuevas del sensor de distancia y esto crece linealmente con el número de partículas definidas en el filtro. Por lo que también hay que definir un correcto número de partículas, para que el algoritmo no se vuelva de una lentitud inaceptable para trabajar en línea.

Si el modelo de movimiento tiene una distribución de probabilidad propuesta con una varianza muy amplia o poco representativa del movimiento real del robot, o el ambiente es muy propenso a derrapes, se necesitarán muchas partículas para poder encontrar el mapa que represente el espacio de manera correcta.

Se han propuesto varios métodos para aliviar el trabajo de los filtros de partículas al generar la distribución propuesta de manera adaptativa, con aprendizaje del modelo de movimiento [Eliazar and Parr, 2004], mejores métodos de remuestreo y preproceso del la información odométrica cruda para corregirla.

3.5. Resumen del capítulo

El objetivo de este capítulo fue explicar detallamente el algoritmo de los filtros de partículas, desde su fundamento teórico, así como sus pasos de ejecución: generación de partículas dado el modelo de predicción, ponderado de las partículas dado el modelo de observación, y el algoritmo de remuestreo para evitar la degeneración de las partículas. Además se explica una variante de los filtros de partículas llamados *rao-blackwellizados*, donde en una probabilidad conjunta, un estimador se calcula de manera analítica para reducir el espacio de búsqueda en el filtro de partículas.

Después de haber explicado de manera general el algoritmo de los filtros de partículas, se hace la aplicación del algoritmo para el problema de la cartografía robótica en el sistema de PMAP, el cual usa una representación tipo rejilla de ocupación del ambiente.

Comprendido ya el funcionamiento de la aplicación de los filtros de partículas en el problema del SLAM, en el siguiente capítulo se detallarán las aportaciones hechas en el problema de cartografía con robots móviles, las cuales intentan sacar provecho de las propiedades de este tipo de soluciones para obtener mejores resultados en un robot de servicios. Estas aportaciones están dentro del marco de la arquitectura de software, la fusión sensorial y los algoritmos de planeación de movimientos en ambientes desconocidos.

Capítulo 4

Contribuciones a la cartografía autónoma

En este capítulo se describen las aportaciones hechas en esta tesis. Se comienza con una descripción de la arquitectura del sistema de cartografía en línea, su modo de empleo y lineamientos en general. Se comenta el trabajo realizado para la exploración autónoma del ambiente. Finalmente se habla del trabajo realizado en la fusión sensorial con dos sensores de naturaleza distinta (telémetro láser y sonar).

Dentro de el tópico de arquitectura de software se hablará de como se diseñó el sistema para obtener una aplicación que procesa el mapa de un ambiente en línea, buscando que el robot se mantenga siempre en movimiento y no se detenga cada vez que tiene un nuevo punto de observación, sino que cada registro de datos, sea una posición a procesar. Para eso hay que enfrentar problemas de sincronía entre los sensores del sistema robótico.

Dentro del problema de la exploración autónoma se desarrolla un algoritmo de exploración ciego al mapa global. Debido al retraso entre la navegación del robot y el procesamiento del mapa, se trabajó en un algoritmo de navegación local, que no depende del mapa procesado, que, sin la hipotética restricción del tiempo, es capaz de explorar todo el ambiente.

Sobre el trabajo hecho en fusión sensorial, se hablará de esquema de fusión de un sensor en el tiempo empleado y el modelado del sonar utilizado con este mismo esquema. Posteriormente, y gracias al teorema de *Rao-Blackwell*, que permite procesar analíticamente la representación del ambiente, se generan distintos mapas, cada uno con el sensor registrado. Luego se fusionan ambos mapas con un operador OR probabilístico, celda por celda.

4.1. Arquitectura del sistema de software

El sistema que se desarrolló es un software para la cartografía autónoma. Está basado en un PMAP [Howard, 2004], que fue modificado con la intención de trabajar en línea, y no fuera de línea analizando una bitácora de movimientos, como estaba diseñado originalmente por el autor. Las modificaciones hechas fueron sometidas a Andrew Howard y las aceptó dentro de la versión de desarrollo de PMAP. Estos cambios se orientaron a eliminar dependencias de software y la generación de una biblioteca de carga dinámica del PMAP.

Otra modificación importante que se hizo sobre PMAP, fue la adición de los sonares al momento de actualizar la posición del robot. De esta manera, al generar el mapa una vez obtenida la trayectoria, se pueden generar tres mapas distintos: uno con láser, otro con sonar y la fusión de ambos. Esta modificación aún no se somete al auto original del PMAP.

Se diseñó arquitectura de software para el desarrollo de dicho sistema teniendo en consideración una estructura de tres gradas, el procesamiento asíncrono de la información sensorial, el procesamiento paralelo e independiente y una supervisión y control a través del protocolo HTTP. Otra decisión de diseño tomada fue que el sistema fuera lo más modular posible, ofreciendo la capacidad de cambiar un módulo por otro sin muchos problemas debido al bajo acoplamiento entre ellos.

4.1.1. Arquitectura de tres gradas vs. Arquitectura de tres capas

A mediados de los años 80 Brooks [1986] revolucionó la concepción de los sistemas robóticos móviles autónomos al introducir su arquitectura de subsunción, al romper con el enfoque tradicional de Percepción-Planeación-Acción (SPA por sus siglas en inglés - Sensing/Planning/Action-), donde la planeación se volvía tan compleja que simplemente no podía afrontar ambientes con incertidumbre e impredecibles, ya que la planeación hecha resultaba de poca utilidad. No obstante, la arquitectura de subsunción, al ser de naturaleza meramente reactiva, no cumplía con las expectativas a largo plazo de una tarea compleja.

Según Gat [1997] estos problemas pueden verse como el resultado del método usado para administrar la información del estado interno del robot. El cómputo con alto consumo de tiempo, como la planeación o el modelado del ambiente generan estados internos cuya semántica refleja estados en el ambiente. Es por estas circunstancias que los robotistas al

paso de los años propusieron nuevas arquitecturas que pudieran mezclar las virtudes del SPA como de la arquitectura de subsunción, eliminando lo más posible sus desventajas. El resultado fue una convergencia entre los distintos proyectos que se realizaron de manera independiente y con fines distintos: La arquitectura de tres capas.

Las arquitecturas de tres capas organizan los algoritmos robóticos de acuerdo al manejo de su estado interno:

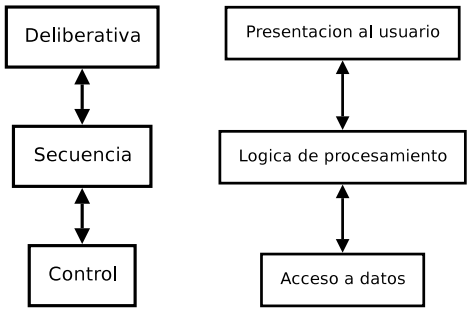
- **Capa de Control**: no mantiene un manejo de su estado.
- **Capa de secuencia**: contiene un estado reflejando memorias sobre el pasado.
- **Capa deliberativa**: mantiene su estado que refleja predicciones sobre el futuro.

Los algoritmos sin estado, basados en sensores, habitan en la *capa de control* o funcional; ahí se implementan los ciclos de retroalimentación y el acoplamiento entre sensores y actuadores, utilizando primitivas de comportamiento tales como seguimiento de paredes, movimiento a un destino con evasión de obstáculos, etcétera. Los algoritmos que contienen memoria sobre el pasado habitan en la *capa de secuencia* o de ejecución; esta capa realiza la ejecución del plan recibido y recuerda lo que ya se ejecutó para seleccionar la primitiva de comportamiento a solicitar. Los algoritmos que hacen predicciones sobre el futuro habitan en la *capa deliberativa* o de planeación; son los que más tiempo de cómputo requieren y están orientados a la planeación y búsqueda.

Otro problema muy diferente, y que puede prestar a confusiones, es la cuestión de la arquitectura de software para el procesamiento robótico, que se enfrenta a planteamientos tales como desempeño, flexibilidad, mantenimiento, reusabilidad y escalabilidad. Tal vez ambas "arquitecturas" tengan puntos de contacto comunes, pero esto es más por coincidencia que alguna razón deducible. Hasta en el nombre se encuentran las diferencias: mientras que una es arquitectura de capas, la otra es arquitectura de gradas.

La arquitectura de software distribuida cliente/servidor de tres gradas [Institute, 2005], incluye un sistema de interfaz de usuario en la grada superior, donde radican los servicios al usuario tales como diálogo y despliegues de información. La tercera grada provee la administración del acceso a datos; este componente se encarga de asegurar la consistencia de las datos a procesar. La grada intermedia procesa la administración de servicios que son compartidos entre múltiples aplicaciones.

Para visualizar mejor las diferencias la Figura 4.1, se observa la diferencia conceptual entre la arquitectura de tres capas y la arquitectura de tres gradas.



(a) Arquitectura de capas

(b) Arquitectura de gradas

Figura 4.1: En 4.1(a) se muestra arquitectura de capas en un robot, donde cada capa está en función del manejo del estado interno del robot. En 4.1(b) se observa la arquitectura de software genérica de tres gradas, donde se separa el procesamiento de acuerdo a la tarea del flujo de información.

En esta arquitectura de gradas de la figura 4.1(b), la grada de interfaz con el usuario se expone a través de páginas Web en HTML, donde el usuario puede controlar y supervisar la operación del robot. La grada de acceso a datos esta provista por el software *Player/Stage* que envía la información de los sensores y envía comandos de control a los actuadores. Por último la grada de lógica de procesamiento provee de los módulos de exploración y cartografía. Dichos módulos mantienen un bajo acoplamiento, permitiendo intercambiarlos por nuevos módulos donde se implementen otros algoritmos. Todo lo anterior dicho se esquematiza en la Figura 4.2.

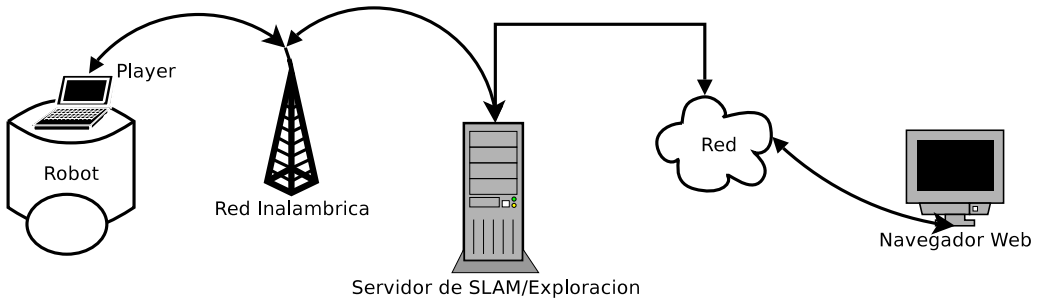


Figura 4.2: Vista de despliegue del sistema. Se observa la disposición física de los distintas partes del software.

4.1.2. Asincronía de información de los sensores

Uno de los principales problemas a los que se enfrenta la robótica móvil aplicada es, como ya se mencionó, el acoplamiento entre actuadores y sensores. Dicho de otra manera, la sincronía entre ellos es que, cuando se le pida un movimiento no planeado a los actuadores, sea en el momento en que los sensores detectaron la necesidad de ese movimiento imprevisto. El problema de la asincronía se vuelve más complejo cuando los diferentes componentes del robot son heterogéneos, de fabricantes distintos, y se unen en una unidad de procesamiento también diferente.

Antes de explicar la solución al problema de la asincronía, expliquemos un recurso de programación clásico llamado retrollamada o *callback* en inglés. En el lenguaje de programación C se pueden tener punteros a funciones, con esto se puede asociar algún evento con la ejecución de una función. Este concepto es utilizado y ampliado por la biblioteca *GObject* [Lacage, 2004], en lo que denominaron señales o *signals* (diferente al concepto de *signal* clásico de *Unix*). Por tanto se elaboró un envoltorio a los *proxys* de los dispositivos de *Player/Stage* para obtener una representación de *GObject* de ellos y pudieran asociarse funciones cuando ocurriera el evento de actualización de datos del sensor.

Entonces, para intentar zanjar el problema de la asincronía en la actualización de datos de los sensores, se decidió abordar el problema aprovechando el uso de retrollamadas que se ejecutaran cuando la información se renovara. Como se visualiza en la Figura 4.3, cuando se actualizan las mediciones de los sensores, cada dispositivo, que representa a un sensor, ejecuta una función que se asoció al evento, que mete esa información a una estructura de cola (primeras entradas / primeras salidas). Esta estructura es asíncrona, debido a que el proceso productor añade datos y el proceso consumidor extrae datos, cada uno a tiempos distintos. En este caso el consumidor de datos es el proceso del SLAM el cual tiene un tiempo de procesamiento de los datos mayor a su velocidad de arribo. Por otro lado la estructura de cola asíncrona es segura en el manejo de hilos de procesamiento ya que maneja bloqueos automáticos al meter y sacar datos de ella.

Esta asincronía es factible, debido a que el proceso de la cartografía robótica, que consume muchos recursos de cómputo, es de lento procesamiento y no requiere un acoplamiento entre sensores y actuadores para describir un comportamiento en el robot, aunque sí necesita una constante y exacta descripción de los movimientos realizados y descripciones de ambiente capturados.

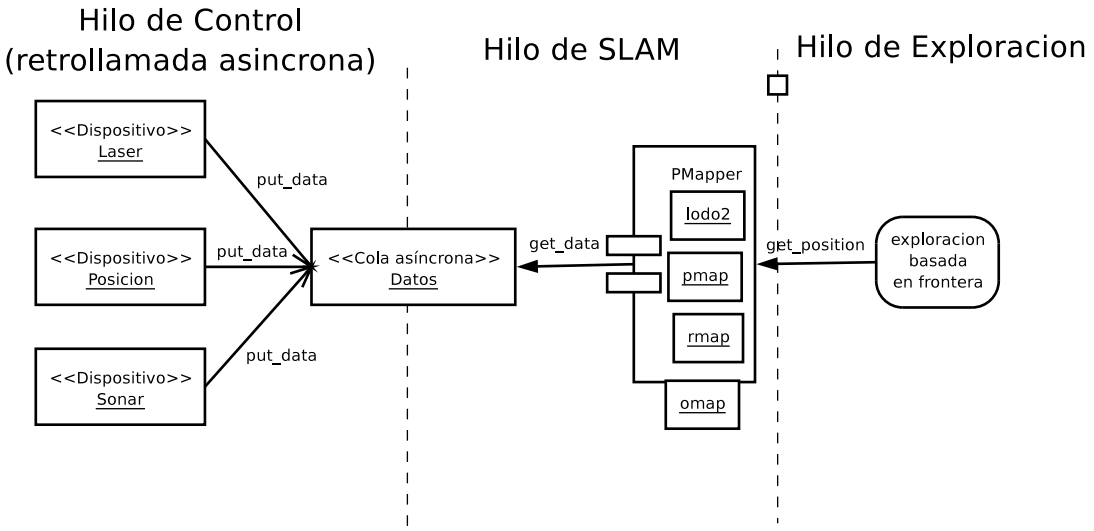


Figura 4.3: Vista del manejo asíncrono de la información de los sensores. Los objetos que representan los sensores al recibir nueva información, realizan una retrollamada, la cual se encarga de encolar la información nueva que sera procesada por PMAP.

4.1.3. Procesamiento multihilos

Antes de hablar sobre el procesamiento multihilos, hay que comentar las principales clases que componen el sistema. En la Figura 4.4 se observan la composición de agregación en la cual interactúan las clases principales. La vista de clases evidencia las relaciones en el código de los distintos objetos que conforman el sistema.

La clase *PCoordinator* es la principal, encargada de controlar el estado global del sistema, de iniciar y finalizar los hilos de procesamiento involucrados, de controlar el ciclo de vida de los objetos que componen el sistema.

La clase *PMonitor* es la encargada de levantar un servicio Web, que servirá para supervisar el desempeño del sistema a través del protocolo HTTP. Las respuestas son construidas a través de plantillas, lo cual ofrece gran flexibilidad, al ser capaz de generar HTML, VXML o algún otro formato de XML.

La clase *PRobot* es una composición de todos los dispositivos que se encuentran en el robot, y cada uno de estos están representados en la clase *PDevice*, quien es la encargada de administrar el ciclo de vida del dispositivo y el manejo de las retrollamadas cuando hay una actualización en su información. La clase *PRobot*, además de ser una colección de los servicios, también se encarga del descubrimiento de los dispositivos configurados y activos

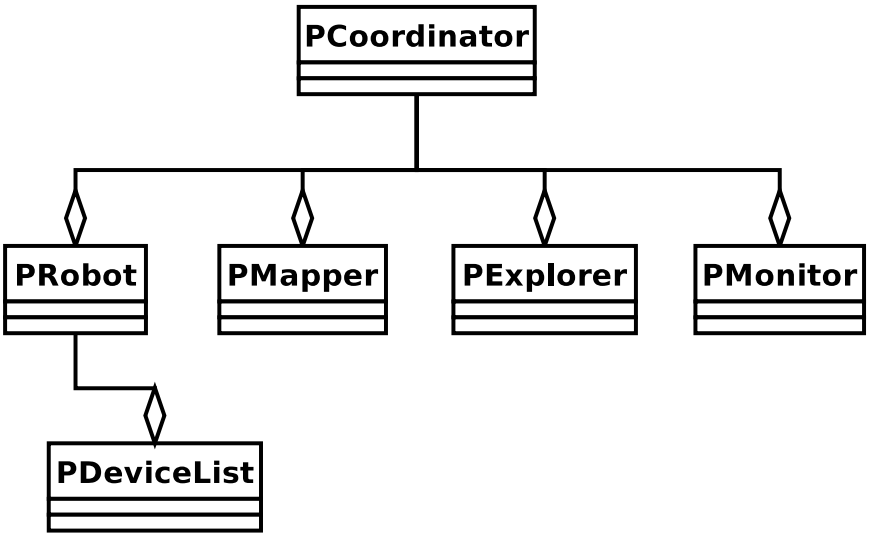


Figura 4.4: Vista de clases. Muestra la composición de agregación de las clases que forman parte del sistema.

en el robot, así como la conexión y desconexión del sistema con el robot.

PExplorer es la clase encargada de realizar los movimiento necesarios para explorar el ambiente que es desconocido para generar su mapa. El algoritmo implementado se verá más adelante.

Por último, la clase *PMapper* coordina las operación del *Simple Mapping Utilities*, las cuales fueron modificadas para que funcionaran como bibliotecas y no como una aplicación en sí, y que operaran en línea, y no fuera de línea con la lectura de una bitácora. Esta clase también implementa las retrollamadas de los dispositivos involucrados en la cartografía e implementa la cola asíncrona de procesamiento.

Las operación de los distintos objetos, instanciados de las clases descritas, trabajan en distintos hilos de ejecución, de acuerdo a la tarea desempeñada. Este diseño permite aislar las tareas, hacerlas independientes unas de otras, y que sean capaces de trabajar en paralelo, si se cuenta con un sistema de cómputo con varios procesadores. La Figura 4.5, muestra la descomposición de los hilos de procesamiento en el sistema. Esto también ofrece la posibilidad de reemplazar una clase, utilizando mecanismos de herencia en GObject [Lacage, 2004].

El **hilo principal**, que contiene la instancia de la clase *PCoordinator*, quien es la encargada de controlar los demás hilos, hacer el análisis léxico y sintáctico del archivo de configuración, además de manejar la instancia de la clase *PMonitor*, que es el servidor Web incrustado en

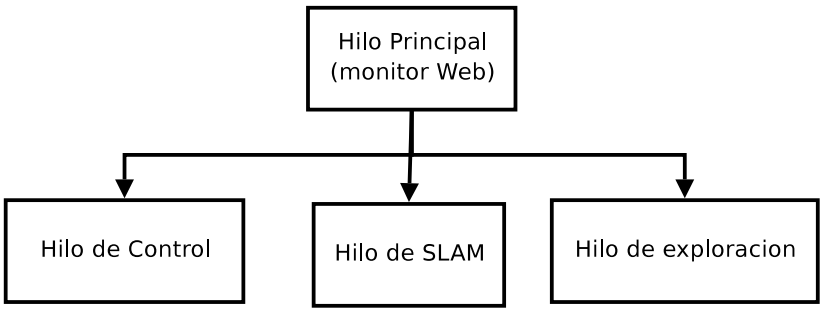


Figura 4.5: Vista de la jerarquía y control del los hilos de procesamiento en el sistema.

la aplicación y también coordinar el estado global del sistema. El primer hilo que levanta es el **hilo de control** encargado de actualizar constantemente la información de los sensores del robot y de enviar los comandos de operación sobre los actuadores, a través de la función **player_read**. El hilo de control ejecutará algunos ciclos de actualización de datos espúreos para asegurar una buena conexión con el robot.

El hilo principal levanta después el **hilo del SLAM**, quien conecta las retrollamadas a los dispositivos seleccionados (láser, sonar y odometría), instancia la cola asíncrona e inicializa las estructuras necesarias para el proceso del SLAM en la clase *PMapper*, así como su ciclo de procesamiento con respecto van llegado los datos en la cola, con la función **process_coarse**.

Por último levanta el **hilo de exploración** donde la instancia del *PExploration* ejecuta la estrategia de movimientos necesaria para la exploración autónoma del ambiente desconocido, utilizando la función **explore**.

El proceso descrito hasta ahora puede visualizarse en el diagrama UML de actividades de la Figure 4.6. Los diagramas de actividad son una variación de una máquina de estado, en donde los estados representan la ejecución de las acciones o subactividades y las transiciones que son disparadas al término de las acciones subactividades [OMG, 2003].

Desde cierta perspectiva, la arquitectura expuesta tiene resemblanza a la arquitectura de tres capas, descrita al inicio de esta sección. El hilo de control sería la capa de control, la capa de secuencia sería el hilo de exploración y la capa de deliberación estaría representada por el hilo del SLAM.

Esta arquitectura tiene la gran de ventaja del bajo de nivel de acoplamiento entre las clases, pudiendo sustituir el algoritmo de SLAM tan sólo modificando la clase *PMapper*, o el algoritmo de exploración cambiando la clase *PExplorer*, también el comportamiento del

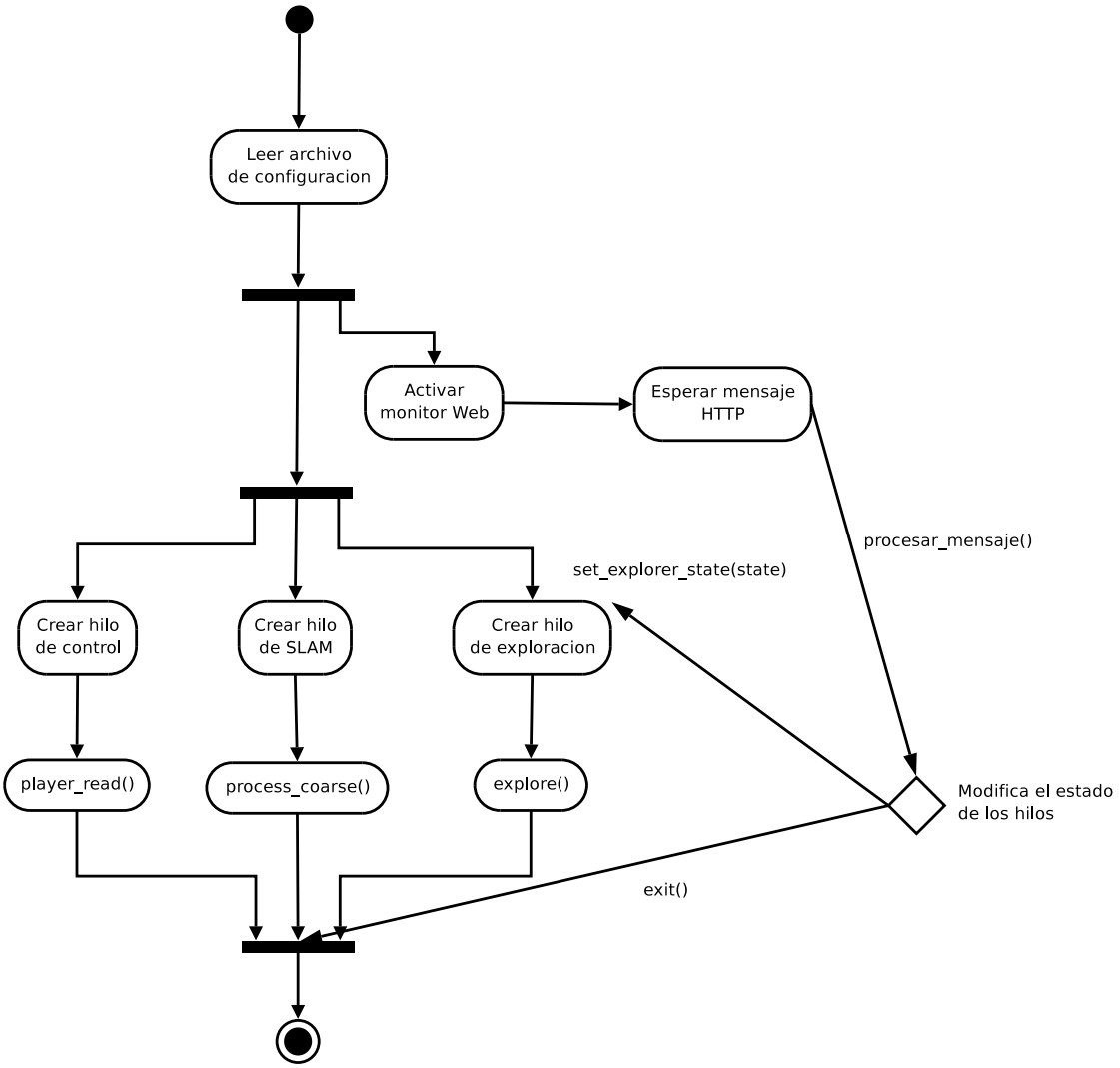


Figura 4.6: Diagrama de actividades del sistema durante su ciclo de vida.

monitor con retoques en la clase *PMonitor*.

Sin embargo, no todo resultó como lo esperado en el periodo de diseño del sistema: Se observó que los errores y retrasos de transmisión datos por la red entre el robot (servidor *Player*) y el sistema de construcción de mapas, la asincronía de la información de los sensores, los errores intrínsecos del hardware del robot y defectos en el controlador del robot en *Player*, hacían que en la exploración autónoma, donde era necesaria la ejecución de primitivas de movimiento, hubiera comportamientos no deseados, al no existir un buen acoplamiento entre la información de los sensores y la reacción de los actuadores, tal como la colisión del robot con los obstáculos. Se tiene la impresión de que es mejor implementar los comportamientos primitivos como controladores dentro del servidor *Player*, y que no

tengan que viajar por la red inalámbrica de ida y vuelta.

4.2. Exploración Autónoma

El problema de la Cartografía con robots móviles se limita a obtener la posición del robot y a observar lo que le rodea, tomando en consideración lo que observó anteriormente [Newman *et al.*, 2003]. La forma cómo se desplazó el robot en el ambiente no forma parte de su objetivo; bien puede haberse teleoperado a través de un joystick, o el robot recorrió el ambiente de manera autónoma, esto no importa en el problema del SLAM.

Sin embargo, si lo que se busca es la autonomía completa, el robot deberá entonces ser capaz de explorar el ambiente desconocido y generar la representación interna de él por sí mismo, sin intervención de alguna clase. A esta tarea se le conoce como exploración.

La exploración es el acto de moverse dentro de un ambiente desconocido mientras se construye un mapa del mismo [Yamauchi, 1997]. Una buena estrategia de exploración es aquella que genera un mapa completo, o casi completo, en un tiempo razonable de tiempo.

Al no contar con una representación previa de ambiente, es imposible hacer planeación de movimientos y por ende es imposible hacer el recorrido del ambiente completo en un tiempo mínimo. Por esto es necesario contar con estrategias y heurísticas para hacer dicha tarea.

Intuitivamente podría pensarse en un algoritmo que desplace al robot hacia las zonas desconocidas más cercanas dentro del mapa construido incrementalmente, y lo mantenga alejado de las regiones ya exploradas. Esta estrategia, aunque no es óptima en tiempo, sí se acerca mucho [Koenig *et al.*, 2001]. Sin embargo, esta estrategia ya no está tan cerca de la optimalidad cuando la construcción de mapa es bajo el contexto del SLAM [Stachniss and Burgard, 2004], ya que típicamente el robot debe volver a visitar los lugares que ya observó para poderse localizar más fácilmente, en especial en el momento de cerrar circuitos grandes dentro del ambiente.

Otra restricción a considerar, específica del sistema construido, es que, como se vio en la sección anterior, la construcción del mapa se hace en un hilo independiente, lo cual aisla la percepción del mapa del hilo de exploración. Y más aún, ya que el procesamiento del mapa es costoso, casi siempre la construcción incremental del mapa está un tiempo atrás de la posición actual de robot. Habría que esperar a que el procesamiento del mapa llegara hasta la última información recibida, para que se planeara una nueva zona de exploración y preparar la planeación de movimientos para llegar ahí. Hacerlo así iría en detrimento de

las virtudes de la arquitectura planteada, tal como el bajo nivel de acoplamiento entre los objetos.

Es por estas razones, que se decidió alejarse de la estrategia común de la frontera (región inexplorada) más próxima, se optó por una estrategia local, basada en el espacio de configuraciones observable por el telémetro láser de robot.

4.2.1. Espacio de configuraciones

La presente explicación ha sido tomada del libro de Dudek [2000].

El espacio de configuraciones (o C-Space) es un formalismo para la planeación de movimientos. Una configuración q del robot A es una especificación del estado físico de A con respecto a un marco fijo del ambiente F_w.

Si se imagina un robot móvil A cuya configuración puede representarse completamente como $q = (x, y, \theta)$. El espacio de configuraciones de A es el espacio $\mathcal{C}$ de todas las posibles configuraciones válidas de A dentro de su ambiente.

La construcción física del robot, su cinemática y la presencia de obstáculos en el ambiente puede prohibir ciertas configuraciones.

Para simplificar el uso del espacio de configuraciones, usualmente se dilatan los obstáculos presentes en el ambiente la longitud del radio del robot, de manera que el robot pueda suponerse como un punto. A esta dilatación se le llama también como la *suma Minkowski*. Esta dilatación puede apreciarse en la Figura 4.7, donde el obstáculo B_i en el marco W, al dilatarse el radio del robot A, en el espacio de configuraciones se vuelve un obstáculo de dimensiones diferentes CB_i.

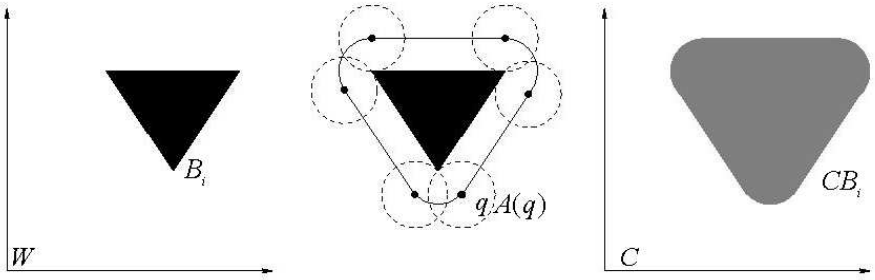
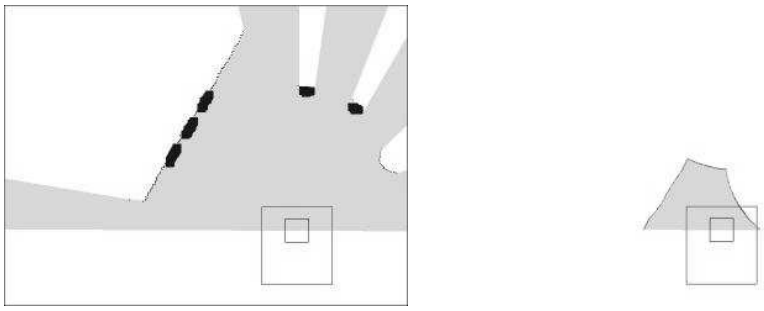


Figura 4.7: Dilatación de obstáculos en el espacio de configuraciones [Ahuactzin and Fraichard, 2005].

Es posible no tener la representación completa del ambiente, porque ésta se obtendrá al final de la exploración, pero sí del lugar observado en la posición actual del robot. Esta representación se obtiene a través de sus sensores. A esta representación local se le pueden dilatar los objetos y obtener el espacio de configuraciones dentro de la región observable.

El proyecto de *Player* tiene un controlador para obtener el espacio de configuraciones dentro de la región observable por el telémetro láser. Hace esto acortando la distancia observable por el láser, además de recortar las mediciones que no entran dentro del espacio libre al dilatarse los obstáculos, delimitando la porción libre de obstáculos. En la Figura 4.8 se observa como una lectura del láser, se acorta a una región totalmente navegable.



(a) láser (b) láser C-Space

Figura 4.8: En 4.8(a) se observa la lectura cruda de un telémetro láser. En 4.8(b) se observa la misma lectura pero ahora procesada para extraer el espacio de configuraciones posible en ese espacio local. Tomado del proyecto Player/Stage.

4.2.2. Algoritmo de exploración ciega

El algoritmo de exploración ciega fue hecho para explorar un ambiente, sin tener una representación incremental de lo ya cartografiado, utilizando sólo búsqueda dentro de espacio de configuraciones percibible por el láser. El fundamento del algoritmo está basado en la etapa de búsqueda del algoritmo del Hilo de Ariadna [Mazer *et al.*, 1992]. Dicho algoritmo tiene sus fundamentos en la planeación probabilística de trayectorias, el cual se aplica en situaciones con robots con muchos grados de libertad o ambientes muy complejos. Genéricamente este tipo de algoritmos trabajan en dos etapas: exploración y explotación. En la etapa de exploración generan grafos aleatorios que permita llevar de un estado a otro al robot dentro del espacio de configuraciones. Al conjunto de grafos obtenidos se aplica

la etapa de explotación, donde se busca el mejor grafo que permita el cambio de estado [Dudek and Jenkin, 2000].

Básicamente el algoritmo utiliza el espacio de configuraciones local, extraído de la lectura del láser para elegir la posición más lejana visible, entre la posición anterior del robot y la actual. Para llegar a la nueva posición, el robot gira el ángulo especificado por el espacio configuraciones y avanza la distancia reportada también por el espacio de configuraciones. No obstante, el ángulo a girar se selecciona con una política ϵ-greedy, ya que con una probabilidad ϵ se selecciona un ángulo al azar. Acto seguido el robot girará hacia esa dirección y avanzará lo indicado por el espacio de configuraciones o hasta que se detecte un obstáculo cercano por los sonares. Y el ciclo se repite. Lo anterior se expresa en el Algoritmo 4.1.

Algoritmo 4.1 Exploración ciega

loop
  $P \leftarrow$ valor aleatorio uniforme $[0, 1)$
  if $P < \epsilon$ **then**
    $Angulo \leftarrow$ valor aleatorio entre $[0, \pi/2]$
    $Distancia \leftarrow$ obtener distancia en el $Angulo$
  else
    $MaxDist \leftarrow 0$
    for all Laser en el LaserCSpace **do**
      $Dist \leftarrow \sqrt{X_{k-1} - X_k)^2 + (Y_{k-1} - Y_k)^2} + Laser.Distancia$
      if $Dist > MaxDist$ **then**
        $Angulo = Laser.Angulo$
        $Distancia = Laser.Distancia$
        $MaxDist = Distancia$
      end if
    end for
  end if
  ejecutar giro al $Angulo$
  while obstáculos estén lejanos **do**
    ejecutar avance con $Distancia$
  end while
end loop

En el área de aprendizaje por refuerzo, que intenta aprender un modelo a partir de un

proceso exploratorio [Morales Manzanares, 2004]. Existen varias técnicas para explorar el proceso que intenta modelar, utilizando distintas heurísticas. La más básica es la exploración aleatoria, la cual, ejecutándola hasta el infinito, es capaz de explorar todo el ambiente. Por otro lado tenemos la heurística de búsqueda glotona (*greedy*), la cual trata de buscar localmente cuál es el resultado óptimo. Esta heurística permite explorar el modelo en menor tiempo, sin embargo se tiene la posibilidad de quedar atrapado en un mínimo local o un ciclo. Para intentar romper esta situación, se puede hacer una mezcla de la exploración aleatoria y la glotona obteniendo una política ϵ-*greedy*, donde, como ya se explicó, se selecciona una acción aleatoria con una probabilidad pequeña de ϵ o una acción glotona con una probabilidad $1 - \epsilon$.

Por lo tanto, el algoritmo presentado es completo, gracias a la política ϵ-greedy, asegurando que el modelo del ambiente será obtenido en su totalidad, además permite que una región del mapa sea visitado más de una vez. No obstante la optimalidad del tiempo de construcción del mapa está perdida.

Una de las ventajas de este algoritmo de exploración ciega, es que el robot no necesita de una representación para planear sus movimientos, por lo que no hay que llevar una correspondencia entre el proceso del SLAM y la exploración, por lo tanto el robot estará en constante movimiento. Sin embargo sí se asegura que, sin restricciones de tiempo, el espacio será explorado por completo, cumpliendo también con la expectativa de volver a áreas ya exploradas con el propósito de mejorar la localización del robot. Además el costo computacional es muy bajo, a diferencia de las búsquedas globales necesarias para encontrar las fronteras más próximas.

Sin embargo, sabemos que por razones de consumo de energía de la baterías, no es posible ignorar la restricción de tiempo de exploración, por lo que queda como trabajo futuro buscar mejorar las heurísticas del algoritmo de exploración.

Los resultados obtenidos con este algoritmo de exploración son extraños, ya que en el simulador *Stage* la operación sobre el espacio de configuraciones del láser trabaja estupendamente, pero en ambientes reales, con pasillos estrechos, al dilatarse los obstáculos, el espacio de configuraciones se vuelve nulo, dejando al robot girando tratando de encontrar una ruta para desplazarse. Para solventar tal caso, se detenía la exploración, se operaba con joystick el robot hasta la zona no explorada y se activaba de nuevo la exploración autónoma.

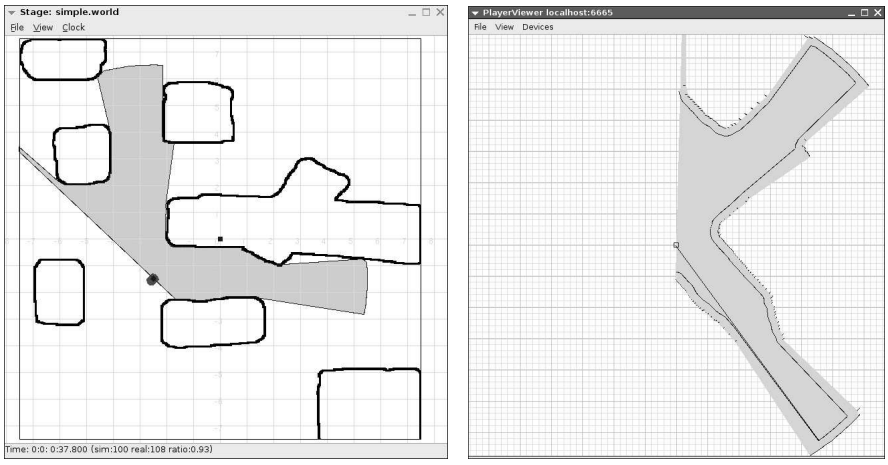
Es apreciable que el problema principal en la exploración ciega es el procesamiento del

espacio de configuraciones local, ya que, como se acaba de mencionar, los espacios estrechos vuelven nulos los espacios de configuraciones. Así que la solución es revisar el algoritmo que genera el espacio de configuraciones, para evitar caer en este caso.

4.2.3. Ejemplo de la exploración

Para aclarar el funcionamiento del la exploración ciega propuesta se presenta el siguiente ejemplo. Las capturas son del simulador *Stage*.

En la Figura 4.9(a) se observa a un robot que percibe su ambiente con el telémetro láser. En la Figura 4.9(b) se visualiza el espacio de configuraciones local obtenido de las mediciones del láser.



(a) Stage (b) Player Viewer

Figura 4.9: En 4.9(a) se observa al robot en un ambiente simulado y la medición del sensor láser. En 4.9(b) se observa la lectura calculada por el LaserC-Space. De ella obtenemos el ángulo a la distancia más larga recorrible.

El primer paso es entonces buscar la medición láser más lejana más la posición anterior del robot y la actual, dada su distancia euclidiana ($d = \sqrt{(x_{k-1} - x_k)^2 + (y_{k-1} - y_k)^2} + distancialaser$). Si la posición del robot es la inicial, entonces la distancia euclidiana es 0 y simplemente cuenta la distancia láser más lejana. Este el caso del ejemplo.

Por espacio de configuraciones se sabe que la distancia más lejana al punto anterior y al actual está a $-54°$ con una distancia de 7.75 metros, lo que forma una línea recta expresada

el la Figura 4.9(b). Entonces, el robot ejecuta el giro y luego trata de recorrer la distancia obtenida.

La distancia es recorrida con una aceleración proporcional al porcentaje de la distancia recorrida, con aceleración hasta la mitad del recorrido y desaceleración después de ésta, manteniendo como límite la velocidad especificada en la configuración del sistema (Ver apéndice A.2.4).

Mientras se recorre la distancia, se supervisa constantemente las nuevas lecturas del láser y el sonar. Si alguna lectura frontal es menor a una distancia de seguridad al obstáculo más cercano, entonces el robot detiene su movimiento y reinicia el ciclo.

Esto ocurre si un número aleatorio de distribución uniforme es mayor a ϵ, de lo contrario, el giro se elige de manera aleatoria, también de distribución uniforme, y se avanza la distancia reportada en espacio de configuración a ese ángulo.

4.3. Fusión sensorial

Para el trabajo de cartografía con robots móviles, el robot debe percibir el mundo a representar a través de sus sensores de distancia. Mucho del trabajo realizado en esta área se ha limitado a procesar la información de un sólo sensor, en busca de una simplificación del problema de por sí ya complejo. Sin embargo, modelar el mundo de operación del robot utilizando sólo una fuente de percepción presenta varios problemas [Aboshosha, 2003]:

- Consistencia espacial limitada: Normalmente, los sensores individuales cubren únicamente una región restringida. Consecuentemente, los mapas de mediciones físicas pueden resultar inconsistentes.
- Tiempo de procesamiento limitado: Cada sensor individual tiene un tiempo limitado de procesamiento que limita la velocidad del robot para que este sea capaz de desplazarse con una constante observación del ambiente.
- Imprecisión: Debido a varias limitaciones y restricciones, las lecturas de sensores individuales son imprecisas.
- Incertidumbre: A pesar del uso de complicados algoritmos para asegurar las mediciones de sensores individuales y homogéneos, el resultado de dichos sistemas sigue teniendo incertidumbre.

Es por estas razones que muchos investigadores han trabajado en la fusión de varios sensores diferentes, lo que permitiría obtener mejores representaciones del ambiente, combinando las ventajas que ofrecen por separado [Romero Muñoz, 2001]. Con tal esfuerzo se obtienen varios beneficios [Aboshosha, 2003]:

- Mayor robustez: La fusión de señales de varios sensores permite al sistema seguir operando aún en el caso fallar un sensor individual.
- Mayor confiabilidad: Utilizando la integración de sensores, las mediciones pueden ser aseguradas y las limitaciones de los sensores individuales pueden ser superadas. Por lo tanto, la integración de sensores es esencial para una navegación segura en terrenos tanto conocidos como desconocidos.
- Una mejor consistencia espacial: Los sensores individuales tiene diferentes posiciones y espacios de operación, y sus lecturas son inconsistentes. Consecuentemente, la fusión de las lecturas de sensores distribuidos conducirán a una mejora en toda la consistencia de las mediciones.
- Reducción del la ambigüedad y la vaguedad: usando múltiples sensores ayuda en la eliminación de las distintas interpretaciones de la percepción de las propiedades del ambiente circundante.
- Resolución mejorada: Cuando múltiples e independientes mediciones hechos con con la misma propiedad están fusionados, la resolución de este valor es mejor que con la medición de un sólo sensor.
- Confianza mejorada: Normalmente la confianza en un sistema con múltiples sensores es mayor que con un único sensor. El paralelismo y la redundancia son empleadas para asegurar el resultado final.

No obstante, la integración de sensores exhibe varias debilidades que hay que considerar al momento de buscar mecanismo que aborden semejante fusión [Aboshosha, 2003]:

- Más complejidad: Tratar con las propiedades físicas heterogéneas de sensores distribuidos incrementa la complejidad de los algoritmos aplicados.
- Probable inestabilidad: El uso masivo de fusión sensorial puede conducir a una inestabilidad del sistema, especialmente si la capacidad de transmisión y el poder

de cómputo es limitado, ya que los tiempos de respuesta pueden verse afectados perdiendo reactividad al momento de evitar obstáculos.

- Dificultad para interpretar la posición: la distribución espacial de los sensores incrementa la carga computacional necesaria para interpretar sus mediciones.
- Distintos tiempos de procesamiento: Sensores distribuidos tienen diferentes tiempos de procesamiento y la sincronización de los módulos de sistema de control se vuelve crítica.
- Más ruido: Las lecturas de los sensores están acompañadas de ruido. El ruido puede aumentar al incrementarse el número de sensores integrados.
- Carga computacional mayor: El procesamiento de los datos de múltiples sensores demanda un mayor poder computacional y por lo tanto cuesta más tiempo.

El problema de la fusión sensorial ha sido abordada desde diferentes ángulos, dependiendo principalmente de la representación del ambiente y de la técnica física de medición empleada por los sensores utilizados.

En este trabajo se realizó la fusión sensorial de un conjunto de sonares y un telémetro láser. En general, la motivación de esta fusión se sostiene un su carácter complementario. Las principal desventaja de un sistema simple de sonares es la especularidad, una amplitud muy ancha del haz e interferencia. Las paredes muy lisas actúan como espejos, haciendo detectables sólo las paredes con ángulos cercanos a 90°al haz acústico. Objetos que tienen propiedades aislantes del sonido pueden desaparecer por completo en las mediciones. En cambio los telémetros láser, pueden no percibir obstáculos que están fuera de su plano de percepción. La detección de espejos y puertas de vidrio son un problema también [Diosi and Kleeman, 2004], ya que son transparentes y por lo tanto indetectables por el láser.

4.3.1. Correlación de datos

Cuando la representación del ambiente es un mapa de características, donde los elementos del entorno se describen con sus propiedades geométricas, la fusión sensorial se trabaja en la correspondencia de datos para afirmar la existencia de un objeto en el ambiente y poder representarlo en el modelo utilizando primitivas geométricas (como puntos y líneas). Tardós y Castellanos [2001] han trabajado mucho en esta área. Sin embargo, cuando el mapa se representa con una rejilla de ocupación probabilista, la fusión sensorial resulta muy sencilla,

ya que al discretizar todo el espacio disponible, la correspondencias de datos se limita a averiguar si la medición de un sensor cae o no en una celda marcada previamente como obstáculo o zona libre.

Uno de los retos que presenta el problema del SLAM es la correlación de datos, donde se distinguen los objetos que componen el ambiente unos de otros. Esta labor es importante ya que de otra manera se pueden colocar obstáculos que no existen en la realidad y simplemente son ruido de los sensores.

Los mapas de rejilla probabilista simplifican mucho este problema, ya que el ambiente es discretizado, así como también la información de los sensores, por lo que la probabilidad de ocupación se incrementa cuando los sensores detectan un obstáculo en cierta celda de la rejilla. Sin embargo, en mapas métricos el problema se complica, ya que se debe fusionar las lecturas de los sensores con técnicas matemáticas más elaboradas como los filtros Kalman o algoritmos de agrupamiento. Aunque, como ya se ha dicho, este trabajo hecha mano de los mapas de rejilla probabilista.

El problema de la correspondencia de datos puede verse como dos tipos de fusión sensorial: la fusión sensorial de un mismo sensor (láser o sonar) en el tiempo, y la fusión sensorial de varios sensores de naturaleza distinta (láser y sonar) [Cañas and García-Pérez, 2002]. El enfoque adoptado en este trabajo fue procesar de manera separada la información de cada sensor, y al final, cuando se tiene un mapa por cada sensor, estos se fusionan utilizando un OR probabilista [Howard and Kitchen, 1996].

4.3.2. Fusión sensorial en el tiempo

El problema de construcción y mantenimiento de rejillas probabilistas de ocupación se divide en dos etapas [Cañas and García-Pérez, 2002]:

- Captura de toda la información que proporciona una nueva lectura de sensor sobre la ocupación del espacio, siguiendo un determinado modelo sensorial.
- Utilizar la información anterior para actualizar la creencia acumulada, materializando la fusión con otras medidas anteriores.

Los enfoques más representativos para representar la creencia de ocupación y de incorporar las información de nuevas observaciones sensoriales son [Cañas and García-Pérez, 2002]:

- Modelo probabilístico bayesiano.
- Teoría de la evidencia.
- Conjuntos difusos.
- Enfoque histográmico de Borenstein.
- Actualización con ecuación diferencial.
- Decisión por mayoría.

En este trabajo se ha seguido el enfoque histográmico [Borenstein and Koren, 1991], debido a su fácil implementación y eficiencia en sus resultados. En él cada celda mantiene un valor de certidumbre CV, indicando la confianza en la existencia de un obstáculo en esa posición, que se mueve entre $CV_{min} = 0$ y $CV_{max} = 255$. Para utilizar el mapa se suele binarizar la creencia de ocupación comparando el valor almacenado en cada celdilla con cierto umbral. Sólo las casillas con evidencia superior al umbral se consideran realmente ocupadas.

El modelo histográmico utilizado en [Borenstein and Koren, 1991] sólo modifica las celdas del eje central perpendicular al sensor que realizado la medida. Para la celda que se encuentra en la posición la distancia medida por el sensor, su valor de ocupación se incrementa $\Delta(t) = +1$, ya que incide sobre el obstáculo detectado. En cambio, en las celdas intermedias, entre el sensor y la distancia medida al obstáculo, su valor de ocupación disminuye $\Delta(t) = -1$.

La mezcla de información se hace empleado una regla aditiva que suma el valor del modelo sensorial al acumulado de la celda:

$$CV_{i,j}(t+1) = CV_{i,j}(t) + \Delta(t)$$

En este trabajo de Borenstein hay un estudio explícito del carácter dinámico de la representación. La regla de actualización contempla la posibilidad de que la creencia pueda cambiar completamente de sentido con un número finito de observaciones, tantas veces como se necesite e independientemente de lo confiado que se estuviera en la creencia anterior. Se considera el número crítico de medidas necesarias para dar una creencia por firme. Ese valor marca la velocidad máxima en la que se desplazan los obstáculos, ya que si su velocidad es muy alta, no se verán reflejados en el nivel de ocupación de las rejillas que ocupen; de lo contrario marcarán una estela en el mapa final.

Otra ventaja es que debido a que las observaciones sensoriales se asumen independientes, se incorporan todas mediciones sin excepción. Tampoco se hacen hipótesis sobre cómo se distribuyen las medidas del sensor dada una configuración del mundo. Es la compensación entre unas y otras la que va conformando la distribución de probabilidad del espacio.

La principal ventaja de la propuesta de Borenstein es que esta forma de fusión sensorial en el tiempo esta pensada para sistema dinámicos, para robots en movimiento, mientras que la mayoría del resto de las propuestas asumen estático el robot a la hora de registrar las percepciones sensoriales [Cañas and García-Pérez, 2002].

4.3.2.1. Modelado de sonar

En este trabajo de tesis, se modeló el sonar de la siguiente manera: Se leía la información de un sonar, se procesa su posición con respecto al robot y se hace la correspondencia a la posición del mapa. Posteriormente se podan mediciones que sean menores y mayores a un umbral inferior y superior respectivamente.

Si la medición no fue podada, se marca como máxima si la medición es mayor a un umbral máximo, definido por la distancia máxima observable por el sonar. Esto ocurre si no se observó un obstáculo y todo el espacio es libre, restándole una unidad a todas las celdas dentro del cono. En caso de que fuera menor a la distancia máxima, quiere decir que se ha reportado un obstáculo, por lo que la distancia hasta el obstáculo se reporta como libre, restando una unidad a las celdas dentro de ese espacio, y sumando tres unidades a las que están dentro del arco que incide sobre el obstáculo. Lo anterior se puede observar en la Figura 4.10.

4.3.3. Fusión sensorial de varios sensores de naturaleza distinta

Como se explicó en el Capítulo 2, el mapa tiene dos usos principalmente: localización y planeación de movimientos. En la Cátedra de Robótica Móvil del Campus Cuernavaca, el trabajo realizado en localización se basa exclusivamente en la información proporcionada por el láser; sin embargo, para la planeación de movimientos, es importante mantener al robot alejado de las zonas de peligro, sobre todo regiones de obstáculos que el láser es incapaz de observar, como espejos [Hernández, 2005]. Por esto, se mantienen dos mapas de manera simultanea: el mapa construido únicamente con la información del láser para las tareas de localización, y el mapa con la fusión sensorial del sonar y el láser, para una

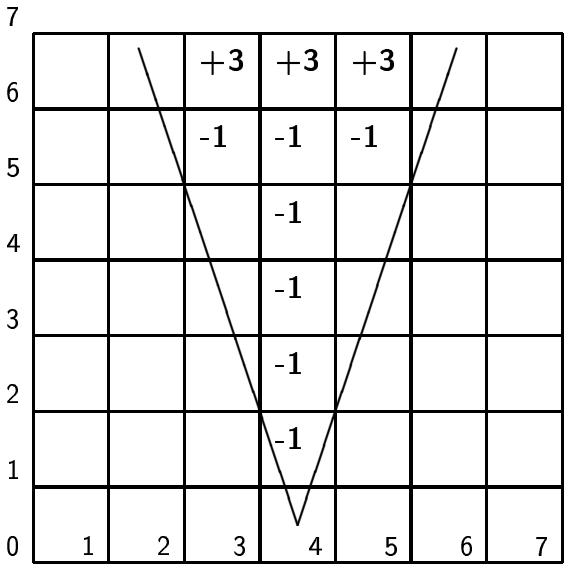


Figura 4.10: Modelado del sonar bajo el enfoque histográmico aplicado en este trabajo.

planeación de movimientos más segura.

Para realizar esta tarea se aprovechó la forma cómo trabaja el filtro de partículas para el problema cartográfico, sobre todo aprovechando las virtudes de la Rao-Blackwellización: El resultado final del filtro de partículas es un conjunto de posiciones que forman una trayectoria del robot en el ambiente, y cada posición esta asociada a las lecturas de los sensores de distancia. Entonces, en el procesamiento analítico del mapa, se toman esta secuencia de posiciones y de lecturas y se generan los mapas.

En el caso de la fusión de la información de dos sensores distintos, se decidió utilizar una versión modificada del método propuesto por [Howard and Kitchen, 1996], ya extendido previamente por [Romero Muñoz, 2001] de un operador de disyunción probabilístico:

$$O \Leftarrow S_1 \vee S_2 \cdots \vee S_n \tag{4.1}$$

Esta implicación dice que la ocupación (O) de una celda (x, y) en el mapa esta dada por la disyunción de la ocupación observada por los distintos sensores (S_i). Por lo tanto, para procesar la fusión de los diferentes sensores, el primer paso es generar el mapa de cada sensor de manera aislada utilizado el enfoque histográmico explicado anteriormente. Ya obtenidos ambos mapas se fusionan con el operador de disyunción.

La razón de utilizar la disyunción subyace en la necesidad de evitar lo más posible los obstáculos o las zonas inciertas, entonces con una disyunción, desde el punto de vista probabilístico, los obstáculos son más marcados si alguno de los dos sensores observa un

obstáculo en una misma celda.

Transformando la implicación lógica en 4.1 al mundo de las probabilidades, se obtiene:

$$p(O) = p(S_1 \vee S_2 \cdots \vee S_n) \tag{4.2}$$

Ahora, suponiendo que las lecturas de los diferentes sensores son mutuamente independientes y exclusivos, y utilizando las leyes de Morgan en 4.2, se obtiene:

$$\begin{aligned} p(O) &= p(S_1) \vee p(S_2) \cdots \vee p(S_n) && (4.3)\\ p(O) &= \neg(\neg p(S_1) \wedge \neg p(S_2) \cdots \wedge \neg p(S_n)) && (4.4)\\ &= 1 - (1 - p(S_1))(1 - p(S_2)) \cdots (1 - p(S_n)) && (4.5)\\ &= 1 - \prod_{i=1}^{n} (1 - p(S_i)) && (4.6) \end{aligned}$$

La salida en bruto de los mapas individuales de los sensores utilizando el enfoque histográmico, son valores de certidumbre (CV) que van de 0 a 255, donde 0 indica espacio ocupado y 255 espacio libre. Los valores de estas celdas son directamente convertibles en archivos de imágenes digitales formato PGM (*Portable Gray Map*), donde el blanco se identifica con el 255 y el negro con 0, y los valores intermedios son niveles de gris.

Bajo este enfoque se puede decir que se tiene la probabilidad de espacio libre si se normalizan los valores de certidumbre, esto es $\neg p(S_i)$. Por lo que la simple multiplicación de la probabilidad de espacio libre de las mismas celdas de los mapas de los sensores individuales y normalizando este valor, daría el valor de certidumbre de espacio libre fusionado.

$$\begin{aligned} \neg p(O) &= (\neg p(S_1))(\neg p(S_2)) \cdots (\neg p(S_n)) && (4.7)\\ &= \frac{CV_1}{255}\frac{CV_2}{255} \cdots \frac{CV_n}{255} && (4.8)\\ CV_{fusion} &= \neg p(O) \cdot 255 && (4.9) \end{aligned}$$

Ahora si sólo se utilizan dos sensores, láser y sonares, entonces la Ecuación 4.9 se simplifica a:

$$CV_{fusion} = \frac{CV_{sonar} \cdot CV_{laser}}{255} \tag{4.10}$$

Con la Ecuación 4.10 se hace la fusión sensorial de los mapas hechos con la información del sonar y el láser, y así obtener los dos mapas necesarios: únicamente láser para localización y la fusión de sonar y láser para la planeación de movimientos.

Una observación interesante es que en los programas de procesamiento digital de imágenes, tales como *Photoshop* o *Gimp*, hay varias maneras de fusionar dos o más imágenes apiladas en capas [Gruschel, 2005], de entre estas maneras están la modos simétricos, no simétricos, oscurecedores, abrillantadores, antisimétricos, etc. Dentro los modos oscurecedores hay un método que se llama **multiplicador**, el cual funciona de la misma manera como ésta fusión sensorial: multiplicando los pixeles coincidentes y dividiéndolos entre 255.

4.3.4. Ejemplos de fusión

A continuación se lista una serie de casos donde la fusión se realizó, observando las diferentes combinaciones de observaciones con detección de obstáculos positivos y negativos tanto del sonar como del láser y su combinación en la fusión sensorial.

El primer caso se muestra en la Figura 4.11. Esta es la amplificación de una sección del mapa elaborado en el Departamento de Matemáticas del Campus Cuernavaca. En este caso es un muro que fue observado sin complicaciones tanto por el sonar como el láser. Aquí la fusión sensorial combina ambas hipótesis dando una mayor creencia en la existencia del obstáculo. Como se puede apreciar, en la fusión sensorial, las zonas grises, que representan partes desconocidas, aumenta la creencia de ser obstáculos mostrando una tonalidad gris más oscura.

(a) sonar (b) láser (c) fusión

Figura 4.11: En 4.11(a) se muestra un caso positivo de detección de un obstáculo por el sonar. El mismo obstáculo fue detectado también por el láser (4.11(b)). Esto resulta en una fusión donde ambas lecturas se combinan para una fuerte hipótesis de obstáculo (4.11(c)).

El segundo caso, Figura 4.12, es cuando el láser no observa un obstáculo traslúcido que

el sonar sí ve, aunque con ciertos problemas de especularidad, lo que forma conos de ocupación. La fusión resultante marca una hipótesis de obstáculo con alta probabilidad, aunque no tan absoluta como ocurre con un muro. Este ejemplo también se obtuvo en el mapa de Departamento de Matemáticas. Son puertas de vidrio adyacentes, que dan entrada a los cubículos de los investigadores.

(a) sonar (b) láser (c) fusión

Figura 4.12: En 4.12(a) se muestra un caso de detección de obstáculo del sonar. El mismo obstáculo no fue detectado por el láser (4.12(b)). Esto resulta en una fusión con una alta probabilidad en la hipótesis de ocupación (4.12(c)).

La Figura 4.13, muestra el caso cuando el sonar, debido a lo grueso del ángulo de los sensores y el poco tránsito del robot por esa región, el mapa del sonar marca como ocupada una sección del mapa que el láser reporta como libre, gracias a su exactitud y fineza en su angularidad. La fusión resultante es una hipótesis, donde la probabilidad de ocupación disminuye, pero no lo suficiente para mostrar la zona como libre. Este caso es también del Departamento de Matemáticas en un espacio muy reducido, donde el robot no se desplazó, únicamente lo observó en su movimiento.

(a) sonar (b) láser (c) fusión

Figura 4.13: En 4.13(a) se muestra un caso negativo de detección con el sonar, viendo un obstáculo donde no lo hay. El láser sí pudo definir ese mismo espacio como libre (4.13(b)). Esto resulta en una fusión con visible probabilidad de ocupación. (4.13(c)).

El último caso es cuando tanto el sonar como el láser fallan en la detección del obstáculo.

Esto ocurre cuando hay una pared de cristal extensa, y el mayor número de observación con los sonares son de manera perpendicular. Es cierto que el sonar marca una delgada zona de ocupación que corresponde a las observaciones oblicuas, pero aún así es más notorio la percepción como espacio libre. El láser toma completamente esa área como espacio libre. La Figura 4.14 muestra este caso. Esta sección es del mapa que se elaboró del Centro Electrónico de Cálculo (CEC) del campus Cuernavaca, donde se tiene un extenso muro de vidrio que separa el pasillo del CEC.

(a) sonar (b) láser (c) fusión

Figura 4.14: Se muestra un caso negativo de detección, perdiendo casi de vista un obstáculo que sí existe. El láser es incapaz de percibirlo (4.14(b)) y el sonar sólo en ocasiones (4.14(a)). Esto resulta en una fusión con probabilidad marcada en la hipótesis de espacio libre (4.14(c)).

4.4. Resumen de las aportaciones

En este capítulo se han revisado las aportaciones realizadas: la propuesta de una arquitectura de software robótico; el algoritmo de exploración ciego, el cual sólo utiliza el espacio de configuración local; por último, se tiene el esquema para la fusión sensorial utilizando un OR probabilístico.

En el caso de arquitectura de software, el autor de esta tesis no tienen conocimiento de un sistema de SLAM basado en filtros de partículas que trabajen en línea, con exploración autónoma y supervisión, construido de manera modular. Además la novedad del control y supervisión por el protocolo HTTP, lo que permite que el sistema completo sea integrado a uno más grande, que contemple otras actividades del robot, donde un coordinador general pueda dirigir el comportamiento de sus distintos componentes distribuidos a través de mensajes utilizando el protocolo HTTP. La desventaja de esto es que no se contemplaron las ideas de la arquitectura de tres capas, con el detrimento de la reactividad del robot y de su capacidad de evitar a tiempo obstáculos.

Sobre la fusión sensorial, tampoco se ha encontrado literatura, que utilice la solución de SLAM con filtros de partículas, que es la trayectoria más probable seguida por el robot en el ambiente, para generar después, gracias a la Rao-Blackwellización del filtro de partículas, distintos mapas, correspondientes a cada sensor de distancia disponible. También se probó que la fusión sensorial basada en un operador OR probabilístico corresponde al algoritmo de multiplicación de imágenes, en el procesamiento digital de imágenes.

En cuanto al algoritmo de exploración, no se tiene conocimiento de algún algoritmo, aplicado a la exploración, parecido al actual, que haga búsquedas en el espacio de configuraciones locales y que trate de recorrer la mayor distancia posible en ella, agregando siempre una posibilidad ϵ de ir a cualquier otro lado. Esto permite mantener el bajo acoplamiento entre los módulos del sistema, no consume muchos recursos computacionales, y permite el volver a recorrer sitios ya visitados, facilitando la localización en la cartografía. La desventaja es que el tiempo de exploración no se asegura que sea el mínimo y que no existe un criterio de paro, es decir, un evento que le indique al robot cuándo se exploró el ambiente por completo y termine de recorrerlo, por lo que esta labor que se delega al usuario mediante el uso de la interfaz.

En el siguiente capítulo se hará una enumeración de los experimentos realizados, los resultados obtenidos y comentarios sobre los mismos.

Capítulo 5

Experimentos

En este capítulo se describen los experimentos realizados durante el desarrollo del sistema, así como las experiencias obtenidas en el uso del mismo.

Las partículas utilizadas en los ambientes reales fueron entre 20 y 50. No son necesarias más porque los entornos no son complejos ni cierran ciclos grandes. Cuando se presentan estos casos, lo recomendables es utilizar 200 o más partículas. En el caso los ambientes simulados, únicamente se empleaba 1 partícula, ya que la odometría es perfecta.

Uno de las intenciones que quedaron en tintero fue tener una simulación con ruido odométrico, sin embargo el trabajo que implica superaba al tiempo disponible. El grupo de desarrollo de *Player/Stage* sigue debatiendo la mejor manera de simular el error en los sensores, sin llegar aún a una especificación de trabajo. El error odométrico es especialmente complicado dado su cualidad de ser aditivo.

No obstante, el funcionamiento de *PMAP* está probado en ambientes grandes y con circuitos. Para probarlo existen varias bitácoras publicas de recorridos con robots en ambientes grandes y complejos. Uno de ellos es *Radish* [Howard and Roy, 2003], o el mismo sitio de *PMAP* [Howard, 2004] tiene una galería de mapas y bitácoras en ambientes igualmente difíciles de cartografiar. Muestra de esta última galería es la Figura 5.1, donde se despliegua un mapa de una sección de los laboratorio de Intel.

Para detrimento de este trabajo, las bitácoras disponibles de para *PMAP* no contienen información de los sonares, impidiendo probar el algoritmo de fusión sensorial realizado. Pero se puede observar claramente que los algoritmos que fundamentan el presente esfuerzo funcionan correctamente, con respecto a lo planteado.

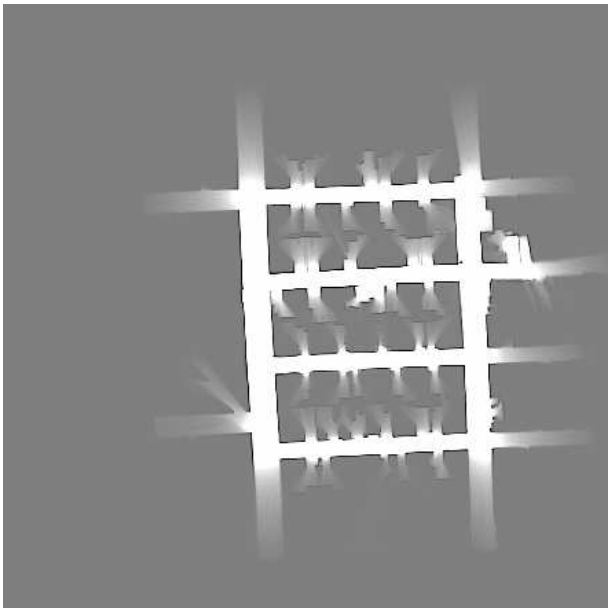


Figura 5.1: Sección de los laboratorios de Intel [Howard, 2004].

tamaño de celda	0.10 metros
robot	Pioneer 2 AT
sensores utilizados	Sick Láser
número de partículas	200
tamaño del ambiente	23x23 metros aproximadamente

Tabla 5.1: Laboratorios de Intel

5.1. Laboratorio de Sistemas Inteligentes

El primer lugar que se cartografió fue el Laboratorio de Sistemas Inteligentes, el cual es un ambiente interior muy poco estructurado, con un pequeño ciclo. El laboratorio es un lugar con muchas sillas, mesas, equipo electrónico por todos lados, y el espacio de navegación es muy reducido.

Se construyó el mapa utilizando PMAP tal como se descarga. Se teleoperó con una palanca de mando virtual (playerjoy) y se almacenó una bitácora de lo percibido por el robot. Esta bitácora se analiza por PMAP en un proceso fuera de línea generando el mapa que se observa en la Figura 5.2.

Este mapa fue el primero utilizado con éxito en algoritmos de navegación y localización

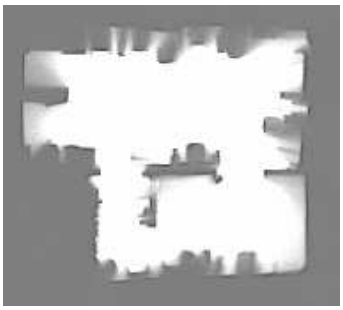


Figura 5.2: Laboratorio de Sistemas Inteligentes

tamaño de celda	0.05 metros
robot	Nomad Scout II
sensores utilizados	Sick Láser
número de partículas	200
tamaño del ambiente	8x8 metros aproximadamente

Tabla 5.2: Laboratorio de Sistemas Inteligentes

[Hernández, 2005] utilizado en la Cátedra de robótica móvil, sobre un ambiente real, pudiendo desplazar al robot de un punto a otro dentro del laboratorio.

5.2. Ambientes simulados

Los ambientes simulados fueron los usados con mayor frecuencia, con ellos se probó el sistema en línea, el supervisado por Web, la fusión sensorial y por último la exploración autónoma.

Una vez comprendido el funcionamiento del PMAP a grandes rasgos, se trabajó en modificar las bibliotecas para que funcionaran en línea. Y se trabajó en el desarrollo de un sistema bajo la arquitectura descrita en el Capítulo 4.

Cuando el código estuvo funcional, inicialmente se comenzaron pruebas sobre ambientes simulados en el *Stage*. La Figura 5.3 es prueba de ello. La operación de exploración se realizaba con un algoritmo sencillo de deambulado o con la palanca de mando virtual.

Las condiciones con las que se hizo la mayoría de los experimientos simulados fue con una computadora de escritorio de laboratorio fungiendo como robot simulado,

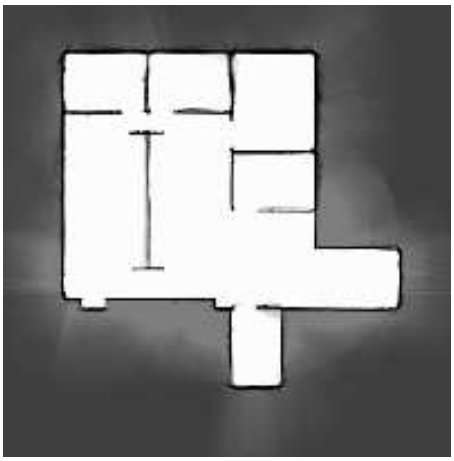


Figura 5.3: Ambiente simulado con espacios de navegación estrechos.

tamaño de celda	0.05 metros
robot	Pioneer 2 DX simulado
sensores utilizados	Sick Láser y anillo de sonares
número de partículas	1
tamaño del ambiente	16x13 metros aproximadamente

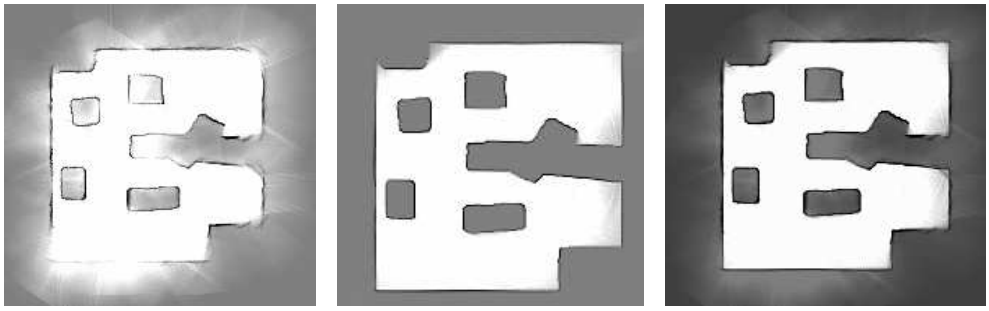
Tabla 5.3: Ambiente simulado con espacios estrechos.

ejecutando el *Stage*, mientras que otra computadora portátil servía como servidor de cartografía. Se separaban ambos procesos ya que cada uno consumía considerables recursos computacionales y correrlos en un mismo equipo de cómputo resultaba, no sólo lento, sino que también el robot simulaba chocaba con los obstáculos. La computadora de escritorio que fungía como robot simulador, también se empleaba para ejecutar la palanca de mandos virtual y con un navegador Web como monitor del proceso.

Los mapas en la Figura 5.4 representa un ambiente simulado, con el cual se hizo mucho del trabajo de pruebas y experimentación. El que se muestra en la Figura 5.4(c) se hizo cuando ya se tenía la fusión sensorial.

En cuanto a la exploración autónoma, se vio que el algoritmo respondía bastante bien en ambientes dispersos, con espacios abiertos grandes, tal como el caso de la Figura 5.4. Este mapa lo podía capturar completo, en tiempos de 10 a 20 minutos. En cambio, en ambientes con espacios libres estrechos, como el de la Figura 5.3, autónomamente nunca iba a zonas estrechas donde el espacio de configuraciones ensanchaba los obstáculos de manera que

parecía intransitable la región. Había que llegar con la palanca de mandos virtual, recorrer esas regiones estrechas y continuar luego con la exploración autónoma.



(a) sonar (b) láser (c) fusión

Figura 5.4: Ambiente simulado disperso.

tamaño de celda	0.05 metros
robot	Pioneer 2 DX simulado
sensores utilizados	Sick Láser y anillo de sonares
número de partículas	1
tamaño del ambiente	10x10 metros

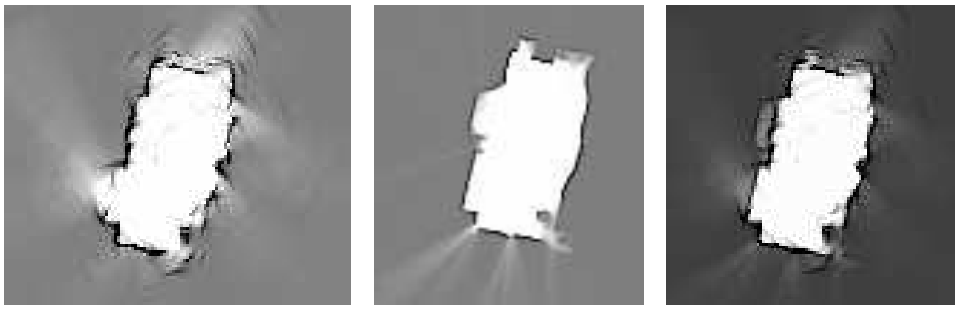
Tabla 5.4: Ambiente simulado disperso.

5.3. Laboratorio de Robots Humanoides

El Laboratorio de Robots Humanoides se cartografió utilizando la exploración autónoma. Es un espacio pequeño, cerrado, interior, se había estructurado con papel cascarón y no había espacios estrechos que el espacio de configuraciones cerrara a la navegación local.

El mapa obtenido se observa en la Figura 5.5. La exploración no tardó más de 5 minutos, ya que con un número de movimientos, recorrió todo lo necesario para generar la representación del ambiente.

Este experimento fue la primera vez que se probó el sistema de cartografía en línea, con supervisión por Web, fusión sensorial y exploración. El sistema funcionó bastante bien.



(a) sonar (b) láser (c) fusión

Figura 5.5: Mapa de Laboratorio de Robots Humanoides utilizando sonar (5.5(a)), láser (5.5(b)) y fusión (5.5(c)).

tamaño de celda	0.05 metros
robot	Nomad Scout II
sensores utilizados	Sick Láser y anillo de sonares
número de partículas	20
tamaño del ambiente	3x5 metros aproximadamente

Tabla 5.5: Laboratorio de Robots Humaniodes.

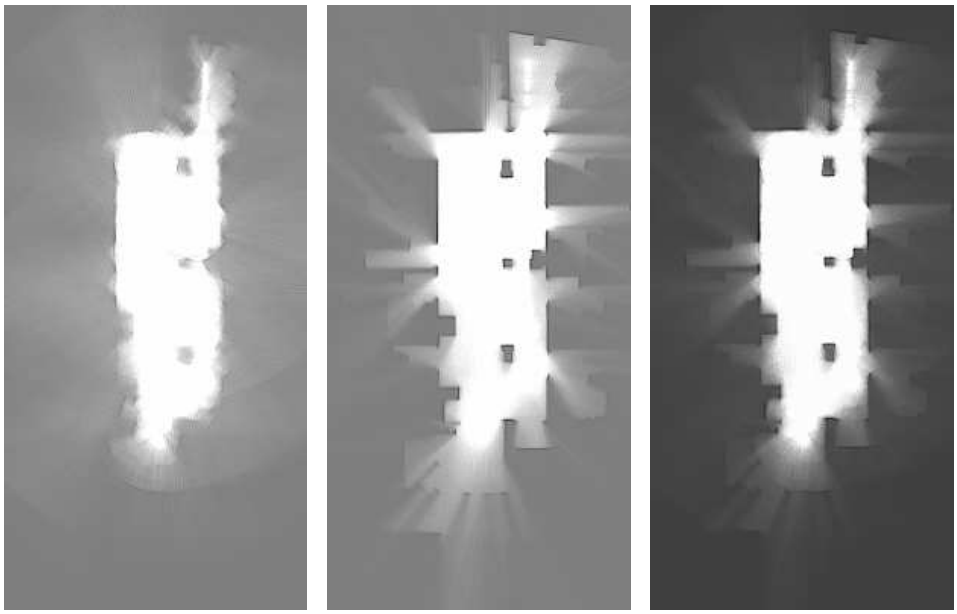
5.4. Departamento de matemáticas

El ambiente que se utilizó para desarrollar y probar la fusión sensorial fue el del Departamento de Matemáticas. El resultado se observa en la Figura 5.6.

Para este ambiente se siguió la estrategia inicial de recorrer el ambiente con la palanca de mandos virtual, guardar una bitácora de lo capturado por lo sensores, y después con el PMAP modificado para la fusión sensorial, se generaron fuera de línea los mapas.

Este ambiente, debido a sus puertas de cristal y espacios estrecho de navegación, fue muy útil para desarrollar la fusión sensorial. Es un espacio muy estructurado de interiores.

El mapa también fue utilizado frecuentemente en pruebas de los algoritmos de navegación y localización, probando la detección de obstáculos no observable por el láser y el atravesar puertas estrechas que son problemáticas cuando se usa la suma Minkowski.



(a) sonar (b) láser (c) fusión

Figura 5.6: Mapa del Departamento de Matemáticas del campus Cuernavaca. La figura 5.6(a) muestra el mapa utilizando únicamente sonar. La figura 5.6(b) muestra el mapa formado con la información del láser. La figura 5.6(c) muestra la fusión de ambos mapas.

tamaño de celda	0.05 metros
robot	Nomad Scout II
sensores utilizados	Sick Láser y anillo de sonares
número de partículas	50
tamaño del ambiente	4x15 metros aproximadamente

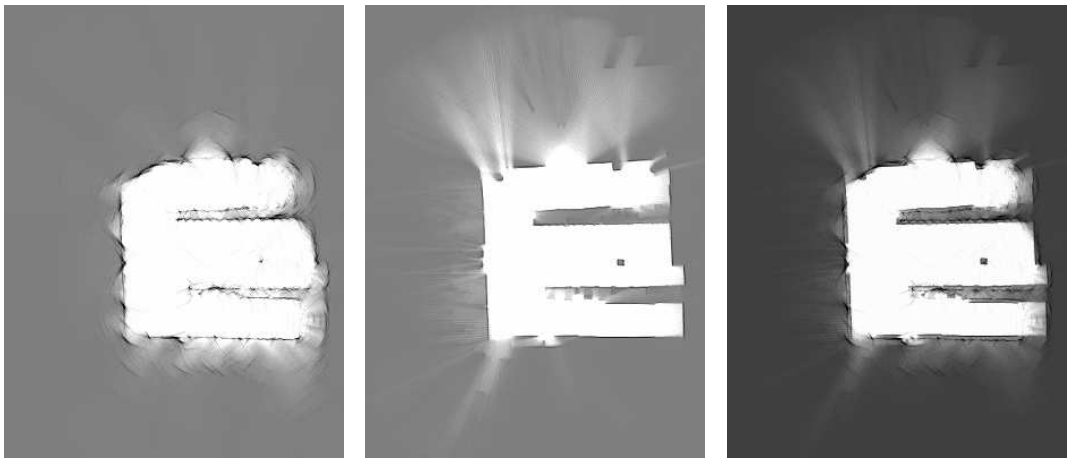
Tabla 5.6: Departamento de Matemáticas.

5.5. Centro Electrónico de Cálculo - CEC

Este fue el último ambiente cartografiado. El CEC fue arreglado con papel cascarón y moviendo mesas para obtener un ambiente grande, estructurado, interior, con zonas tanto estrechas (pasillos extremos) como dispersas (pasillo central).

Este ambiente es particularmente complicado, por los muros de cristal, de buen tamaño y sin ciclos. El problema de la carencia de ciclos cuando los ambientes son grandes es que no

hay forma de correlacionar datos y la localización no tiene mucho soporte, por lo que se puede deformar el mapa final. Los muros de cristal resultaron todo un reto para la fusión sensorial. El resultado final se puede observar en la Figura 5.7.



(a) sonar (b) láser (c) fusión

Figura 5.7: Mapa del Centro Electrónico de Cálculo (CEC) del campus Cuernavaca. En 5.7(a) se muestra el mapa construído con sonares. En 5.7(b) se muestra el mapa formado con el láser. En 5.7(c) se observa la fusión de los mapas.

tamaño de celda	0.05 metros
robot	Nomad Scout II
sensores utilizados	Sick Láser y anillo de sonares
número de partículas	50
tamaño del ambiente	9.6x7.9 metros aproximadamente

Tabla 5.7: Exploración del CEC.

Igualmente, se empleó la estrategia de recorrer manualmente el ambiente, y generar el mapa fuera de línea. En este caso ya se contaba con un modelado de sonares y de fusión sensorial, pero el resultado del mapa con sonares, en un inicio, fue totalmente inútil, sólo se observaba una región informe blanca. Lo complicado del ambiente, con su especularidades y estrecheces, hizo obvias las fallas en el modelado de los sonares. Esta situación obligó a reestructurar el modelado del cono de observación de los sonares logrando mapas más exactos. El resultado se ve en la Figura 5.7(a).

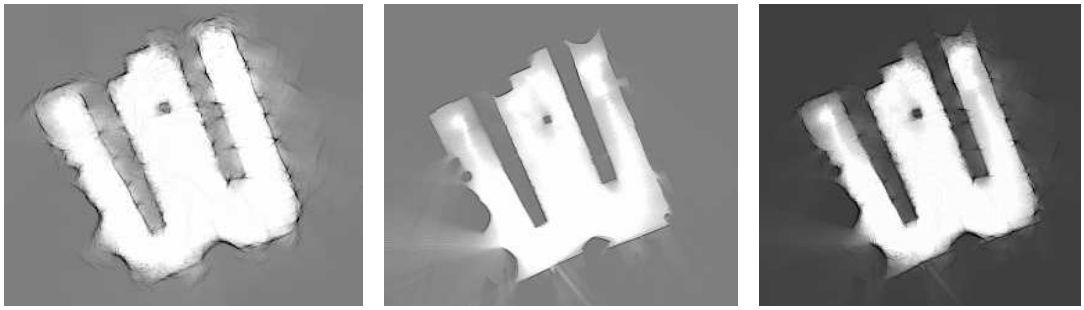
5.6. 1er Concurso Mexicano de Robótica

Dentro del marco del 1er Concurso Mexicano de Robótica, llevado a cabo en el Campus Cuernavaca la tercer semana de agosto del 2005, se llevó a cabo una demostración de avances entre el grupo de robótica móvil del ITESM Campus Cuernavaca y la Facultad de Ingeniería de la UNAM, dentro de la categoría designada como "robots de servicio".

La demostración consistió en cartografiar el mismo CEC y hacer operaciones de navegación en el mismo, tratando de utilizar interacción humano/robot a alto nivel.

Dicha participación sirvió para probar todos los sistemas realizados, además de un intento de integrarlos todos bajo un coordinador de tareas, añadiendo interacción con diálogos de voz. La integración no se pudo llevar a buen término, así que se limitó a probar los sistemas de manera independiente.

El resultado se puede ver en la Figura 5.8. Como se puede apreciar más claramente en este mapa, la orientación de la imagen resultante esta en función de la orientación inicial del robot.



(a) sonar (b) láser (c) fusión

Figura 5.8: Mapa de CEC durante el Primer Concurso Mexicano de Robótica.

En el caso de la cartografía y exploración, la situación fue más bien desfavorable: la exploración no funcionó correctamente debido a lo estrecho de los corredores; el sistema en línea falló debido a que la comunicación entre el *Player* y el robot se perdía constantemente, generando ruido en la información enviada por los sensores; entre otros infortunios de último momento. Finalmente el mapa se construyó con exploración teleoperada con palanca de mando virtual, y se proceso fuera de linea utilizando la bitácora que se manejó como respaldo.

tamaño de celda	0.05 metros
robot	Nomad Scout II
sensores utilizados	Sick Láser y anillo de sonares
número de partículas	50
tamaño del ambiente	9.6x7.9 metros aproximadamente

Tabla 5.8: CEC en el concurso de robótica.

Una vez revisados los experimentos realizados en los distintos ambientes, en el siguiente capítulo se discutirán las conclusiones obtenidas, para finalizar con un listado de áreas abiertas de investigación sobre el mismo camino de esta tesis.

5.7. Comentarios finales

En este capítulo se han descrito los experimentos realizados a lo largo del proceso de trabajo de tesis. Los mapas reales cartografiados se hicieron con el doble fin de servir para los experimentos de cartografía referentes a esta tesis, y para el trabajo de planeación/navegación de Sergio Hernández [2005]. Todos estos experimentos fueron desarrollados dentro del área del Campus Cuernavaca del ITESM, el cual no cuenta con ambientes de interiores complejos, con ciclos y de gran tamaño. Los ambientes reales cartografiados no fueron de mayor complicación y podían obtenerse con pocas partículas. Los mapas obtenidos, con apreciación subjetiva, tenían una observable semejanza con los ambientes reales. Queda como area abierta a la investigación buscar algún método de comparación entre el modelo aprendido por el robot y el ambiente real. Sin embargo, para los fines prácticos de este trabajo, la función del mapa es para la planeación de trayectorias y localización autónoma del robot, y estas tareas fueron realizadas con relativa exactitud, lo que prueba colateralmente la validez del modelo con respecto a la realidad.

Los ambientes simulados fueron utilizados básicamente para probar el funcionamiento de la arquitectura y la exploración autónoma. Su cartografía no requería mayor esfuerzo ya que al ser la odometría perfecta, utilizando una única partícula se convergía en un mapa idéntico al ambiente simulado. Esto se pudo probar sobreponiendo el mapa obtenido y la figura del ambiente en un programa de edición digital de imágenes, a una misma escala. Esto se puede observar en la anterior Figura 5.3, imagen que es resultaldo de esta sobreposición.

Como ya se mencionó al comienzo del presente capítulo, no se contempló el desarrollo de

un módulo para *Stage* capaz de inducir ruido aditivo a la odometría simulada, ya que es una tarea elaborada, y se decidió mejor esperar a que el grupo de desarrollo de *Player/Stage* se decidieran por un modelo de error genérico para sensores.

Capítulo 6

Conclusiones y trabajo futuro

Este capítulo abarca las conclusiones de la presente tesis. Se habla un poco sobre los temas abiertos en el área del SLAM utilizando filtros de partículas

De cada tópico relevante en este trabajo se hace un comentario final, a manera de conclusión, tratando también de dejar abiertas líneas de trabajo posterior.

6.1. SLAM con filtros de partículas

Los filtros de partículas, como herramienta matemática para resolver el problema de la construcción de mapas y la localización simultánea, entre otras, está teniendo gran aceptación en la comunidad de la robótica móvil, por su simplicidad de implementación, su velocidad de procesamiento y los buenos resultados que arroja.

Lo realizado en esta área se limitó a estudiar y comprender su funcionamiento y aplicación en el problema de la cartografía con robots móviles. Gran parte del tiempo dedicado a este trabajo se empleó en estudiar artículos y bibliografía relacionada.

Aunque esta novel herramienta matemática ha tenido gran popularidad y aceptación, no obstante permanecen abiertos varias dificultades, en especial sobre el tipo de representación a utilizar, que, debido al limitado tiempo disponible para hacer este trabajo de tesis, no se pudieron afrontar. La forma de representar el espacio afecta todo el proceso de cartografía, también la planeación de trayectorias y movimientos, así como la localización local y global. Cada tipo de mapa tiene sus ventajas y defectos. En el caso de las rejillas, estas son de fácil implementación pero resultan muy ineficientes en el consumo de espacio en memoria, cuando se quiere trabajar en exteriores. Si bien, de manera teórica, los filtros de partículas son el camino a seguir para generar mapas incrementales, la forma de representación del

espacio aún no es homogénea. Un buen reto sería proponer alguna representación que le sea útil a la mayoría de las personas involucradas en la robótica en general, y a la robótica móvil en particular, aunque hay que señalar que los mapas de características tiene más aceptación en la comunidad e indican la tendencia actual.

Otro problema abierto dentro de la cartografía con filtros de partículas es precisamente el número de partículas a utilizar. El número de partículas debe especificarse antes de la construcción del mapa. Este número depende de qué tanto la distribución propuesta se acerca a la buscada, así como del proceso de remuestreo, para evitar que la partícula con la hipótesis correcta sea desechada [Grisetti *et al.*, 2005]. En este trabajo se utilizó como distribución propuesta una distribución normal por cada grado de libertad ($[x, y, \theta]$), lo que constituye una distribución propuesta poco depurada. Es por esto que el número de partículas seleccionadas influye en el mapa final, concordando con la realidad o no. Mejorar la distribución propuesta bien puede quedar como trabajo futuro.

Finalmente, también se deja como un interesante trabajo futuro, desarrollar alguna de medición para verificar la exactitud del modelo obtenido por el robot y el mundo real. A jucio del autor de este trabajo de tesis este trabajo es cuestionable ya que las observaciones de un robot están limitada fundamentalmente a la precisión de sus sensores, y realizar esta comparación sería competir entre la percepción humana y la percepción robótica. Si el fin último de la representación interna del ambiente para un robot es la planeación y localización, siempre y cuando pueda hacer estas tareas correctamente, el mapa es correcto. No obstante, si el fin es obtener un mapa con fines de consumo humano, el problema de verificar la correctez del mapa es una cuestión abierta.

6.2. Arquitectura

La arquitectura de un sistema de robot de servicio es un tema que abarca de manera global todos los tópicos descritos (cartografía, exploración, supervisión y control). Para comenzar, se echó mano de la arquitectura de software clásica, analizando las distintas vistas y aplicando patrones arquitectónicos útiles, uno de estos patrones es la arquitectura de tres gradas. También, por otro lado, está la distribución de tareas robóticas en tres capas, tal cómo lo describió [Gat, 1997], que también habría que agregar.

En resumen este trabajo de tesis implementó un arquitectura de tres gradas, separando la generación de datos, de la grada de procesamiento de información, con la grada de

presentación de la información. El procesamiento de los datos, es decir, los de *PMapper* y *PExplorer*, junto con los de datos (*PRobot*) y el despliegue de información (*PMonitor*), se ejecutan de manera concurrente, en varios hilos de procesamiento. El procesamiento de los datos por parte de *PMapper* es asíncrono con respecto a la generación de los mismos en *PRobot*. El control del sistema global se hace a través de una interfaz Web, que puede adaptarse a través de un mecanismo de plantillas (ver Apéndice A.3).

Este trabajó intentó seguir un patrón arquitectónico de tres gradas, perdiendo de vista la distribución de tareas, que resulta muy recomendable. Los resultados saltaron a la vista: en nuestro sistema, la independencia de los subsistemas hacen factible el cambio de algoritmos sin necesidad de una recodificación completa, se cuentan con herramientas de supervisión, y el control del trabajo global es muy intuitivo. Sin embargo, por no seguir la distribución de tareas, el bajo nivel de acoplamiento en la capa de control, hicieron que la reactividad del robot no fuera la deseable, obteniendo resultados desagradables y extremos, como la colisión del robot con obstáculos.

Es por esto que una asignatura pendiente sería rehacer el sistema, ahora contemplando las tres capas de [Gat, 1997]. En la capa de control el *Player* no es suficiente, dentro de él habría que implementar controladores para primitivas de comportamiento como seguimiento de paredes, campos potenciales, histogramas de campos de vectores, etc.. Pudiendo así tener constantemente un comportamiento reactivo sin perder el objetivo local de movimiento. De esta manera, la siguiente capa, de secuencia, trabajaría en un nivel más alto de abstracción, olvidando comandar al robot con operación de velocidad y giro. Esta capa podría ya estar desplegada en otra computadora conectada al robot a través de una red inalámbrica. Finalmente tendríamos la capa deliberativa con los planeadores de trayectorias.

Es importante recalcar que el sistema corre en línea, con las modificaciones en PMAP. Además de ser administrable el sistema entero a través de mensajes por el protocolo HTTP, y una supervisión del sistema a través del mismo protocolo. La supervisión se hace por medio de plantillas HTML que pueden ser modificables al vuelo, si necesidad de recompilar el sistema. Que el sistema sea controlable por mensajes HTTP hace posible que se pueda integrar a un sistema más grande, donde un coordinador de tareas robóticas concurrentes pueda acceder a las operaciones de exploración y cartografía.

Este coordinador de tareas robóticas concurrentes implica varios retos: tal como buscar la manera de integrar la Interacción Humano-Robot dentro de esta arquitectura de tres capas. ¿Sería una capa distinta o se integraría a una de esas tres? El mismo dilema esta para el coordinador de tareas, que contemple al robot desde una perspectiva de alto nivel, capaz

de interactuar socialmente con los seres humanos y realizando tareas de utilidad para ellos.

No menos importante, la integración de un sistema de control de fallos, capaz de prevenir un mal funcionamiento del robot, vigilando constantemente el buen funcionamiento de sus distintos componentes. Este módulo de control de fallos es una ausencia de relevancia actualmente en el sistema.

Sin embargo, el sistema actual representa un avance con respecto a los clásicos sistemas de Percepción-Procesamiento-Acción, al trabajar de manera simultánea varias tareas y con un control más simple de robot para la teleoperación y supervisión.

6.3. Fusión sensorial

En este trabajo, para lograr la fusión, se añadió las mediciones de los sonares en el *PMAP*, utilizando el modelado de Borenstein [1991]. La propuesta de esta tesis de aprovechar la marginalización analítica del mapa con los filtros de partículas rao-blackwellizados, y obtener distintos mapas, de acuerdo al sensor utilizado y su fusión, es de relevancia, ya que así, los algoritmos de localización y planeación pueden utilizar la información de un sensor específico o de todos en conjunto.

En el caso, por ejemplo, de la localización: La localización es más fácil de calcular utilizando un sólo tipo de sensor, específicamente empleando el telémetro láser, como es el caso de lo usado actualmente en la Cátedra [Hernández, 2005]. Entonces, para comparar el entorno actual del robot con el modelo, utilizando el láser, se debe utilizar un modelo construido únicamente con el mismo sensor: láser. Por otro lado, para planear trayectorias seguras, donde el robot tenga la menor posibilidad de colisionar, se emplea el mapa con la fusión, el cual agrega obstáculos que no pudieron ser percibidos por el láser.

La situación cambiaría radicalmente si se cambia de tipo de mapa, a uno, por ejemplo, de características. La fusión tendría que hacerse mediante algoritmos de agrupamiento o con filtros Kalman, lo que impone un costo computacional mayor, y enfrentaría una situación de compromiso entre tener diversidad de información en distintos mapas con el tiempo de procesamiento.

Además de proveer varios mapas, uno por cada sensor de distancia disponible por el robot, se presenta la fusión como una multiplicación de imágenes, visto en el procesamiento digital de imágenes, lo que es muy barato computacionalmente y ofrece resultados que van de acuerdo a lo esperado: servir de complemento para tener una representación del ambiente

más precisa al momento de la planeación de movimientos.

6.4. Estrategia de exploración

El fallo de la exploración autónoma en el 1er Concurso Mexicano de Robótica no creemos que halla sido producto de un mal algoritmo, sino del bajo acoplamiento de la capa de control, además de que el cómputo del espacio de configuraciones del láser es muy restrictivo, eliminando el espacio libre para movimiento en corredores estrechos. El algoritmo en sí es prometedor, valdría el esfuerzo implementarlo añadiendo mejoras en el cálculo del espacio de configuraciones, contemplando estas excepciones.

Aún así habría que buscar mejores estrategias para el movimiento del robot en ambientes desconocidos que mejoren la rapidez con que el mapa es construido. Un buen candidato sería la Búsqueda en Espiral [Dudek and Jenkin, 2000] o el seguimiento de paredes. No obstante, en cualquier caso, faltaría un elemento de suma importancia: poder evaluar la información contenida en el mapa incremental actualmente visible, para elegir la zona a visitar.

Como ya se mencionó, seguir la estrategia de ir a la frontera más cercana es óptimo para recorrer un ambiente inexplorado rápidamente, pero no toma en cuenta las propiedades de la construcción de mapas con algoritmos de localización simultánea. Cerrar circuitos y revisitar zonas ya exploradas son elementos importantes a considerar en la estrategia.

La pregunta es cómo elegir la opción entre ir a una frontera o revisitar una zona ya explorada. La literatura nos habla del procesamiento de la entropía en la información del mapa [Sim and Roy, 2005] o tratar el mapa como un proceso oculto de Markov [Stewart *et al.*, 2003]. Para procesar el mapa con el fin de decidir los movimientos del robot, habría que lograr una sincronía entre la exploración y la construcción del mapa, lo que por el momento es difícil dado el costo computacional que conlleva la construcción del mapa. Además está el problema del criterio de paro, al no tener un procesamiento de las zonas inexploradas del mapa, no se puede saber automáticamente cuándo finalizar la exploración.

De manera práctica, se puede argumentar que el algoritmo es suficiente para la exploración, que tiene un muy bajo costo computacional, a diferencia de otras estrategias que hacen búsqueda de fronteras y planeación de movimientos; cubre, si no se tienen restricciones de tiempo o zonas muy estrechas, todo el ambiente; el robot siempre esta en movimiento y no espera a que el mapa incremental se termine de procesarse hasta la posición actual,

para que luego se haga una búsqueda del nuevo punto de observación, y seguir con la planeación del movimientos y la ejecución de los mismos. Desde este punto de vista, existe la probabilidad de que recorra el ambiente mucho más rápido que con esquema del tipo de búsqueda de frontera más cercana.

6.5. Trabajo futuro

El trabajo futuro puede extenderse en múltiples direcciones. La construcción de mapas es un área de investigación muy activa actualmente, innovaciones, hallazgos y observaciones interesantes surgen día a día. Aún estamos muy lejos de contar con el método infalible de construcción de mapas, aunque los primeros pasos en firme ya se han dado.

Si se pudiera hacer un listado de deseos para la construcción de mapas en un robot de servicios, tal vez podríamos tener algo parecido:

- Una representación del ambiente capaz de aprender información tridimensional, lo suficientemente compacto para ser viable computacionalmente y con elementos para facilitar el razonamiento sobre el ambiente.
- Encontrar una nueva forma de compactar la información en las partículas para ahorrar espacio de memoria.
- Contar con distribuciones propuestas más exactas o tener la capacidad de obtenerlas mediante aprendizaje.
- Algoritmos de remuestreo más eficaces que no eliminen hipótesis correctas.
- Agregar información de visión estereoscópica como sensor de distancia.
- Una arquitectura de software y hardware que sea capaz de lidiar con lo complicado del mundo real, reactivo pero a la vez con capacidad de razonar, resistencia a fallos y con facilidad de desarrollo y extensión.
- Generar mapas con distintos niveles de información, tal como obtener un sólo sensor, una fusión de todos, etc., aprovechando la marginalización analítica en la operación de Rao-Blackwell.
- Estrategias de exploración que se acerquen a la optimalidad en tiempo y que generen mapas completos.

Referencias

[Aboshosha, 2003] A. Aboshosha. An introduction to robot distributed sensors. Technical report, University of Tübingen, Computer Science Dept., Nov 2003.

[Ahuactzin and Fraichard, 2005] Juan Manuel Ahuactzin and Thierry Fraichard. Robotics and motion planning. Lecture slides in the Summer School of Image and Robotics, jul 2005.

[Arulampalam *et al.*, 2002] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions of Signal Processing*, 50:174–188, February 2002.

[Austin and Overett, 2004] David Austin and Gary Overett. Dave's robotic operating system. `http://dros.org/`, 2004.

[Borenstein and Koren, 1991] J. Borenstein and Y. Koren. Histogramic in-motion mapping for mobile robot obstacle avoidance. *IEEE Transactions on Robotics and Automation*, 7(4):535–539, 1991.

[Brooks, 1986] Rodney A. Brooks. A robust layered control system for a mobil robot. *IEEE Journal of Robotics and Automation*, 1986.

[Castellanos *et al.*, 2001] J.A. Castellanos, J.Ñeira, and J.D. Tardós. Multisensor fusion for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, 17(6):908–914, dec 2001.

[Cañas and García-Pérez, 2002] Jose María Cañas and Lía García-Pérez. Construcción de mapas dinámicos: comparativa. In *II Workshop On Physical Agents (Waf'2002)*, 2002.

[Charniak, 1991] Eugene Charniak. Bayesian networks without tears. *AI Magazine*, 12(4):50–63, 1991.

[Chatila, 2005] Raja Chatila. Simoultaneous localization and mapping (slam). Lecture slides in the Summer School of Image and Robotics, jul 2005.

[Cooper, 1987] G. F. Cooper. Probabilistic inference using belief networks is NP-hard. Technical Report KSL-87-27, Medical Computer Science Group, Stanford Univ., Stanford, CA, 1987.

[Côté *et al.*, 2004] Carl Côté, Francois Michaud, and et al. Mobile and autonomous robotics integration environment. `http://marie.sourceforge.net/`, 2004.

[Davison, 2001] Andrew Davison. Scene: Software for map-building and localisation. `http://www.robots.ox.ac.uk/~ajd/SceneN/`, 2001.

[Diosi and Kleeman, 2004] Albert Diosi and Lindsay Kleeman. Advanced sonar and laser range finder fusion for simultaneous localization and mapping. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, september 2004.

[Dudek and Jenkin, 2000] Gregory Dudek and Michael Jenkin. *Computational Principles of Mobile Robotics*. Cambridge University Press, The Edinburg Building, Cambridge CB2 2RU, UK, 1st edition, 2000.

[Eliazar and Parr, 2003] Austin I. Eliazar and Ronald Parr. Dp-slam: Fast, robust simultaneous localization and mapping without predetermined landmarks. In Georg Gottlob and Toby Walsh, editors, *IJCAI*, pages 1135–1142. Morgan Kaufmann, 2003.

[Eliazar and Parr, 2004] Austin I. Eliazar and Ronald Parr. Learning probabilistic motion models for mobile robots. In *ICML*, 2004.

[Eliazar and Parr, 2005] Austin I. Eliazar and Ronald Parr. Dp-slam. `http://www.cs.duke.edu/~parr/dpslam/`, 2005.

[Elinas *et al.*, 2004] Pantelis Elinas, Enrique Sucar, Alberto Reyes, and Jesse Hoey. A decision theoretic approach for task coordination in social robots. In *13th IEEE International Workshop on Robot and Human Communication (RO-MAN 2004)*, pages 159–164, September 2004.

[Gat, 1997] E. Gat. On three-layer architectures. *Artificial Intelligence and Mobile Robots. MIT/AAAI Press*, 1997.

[Georgiev, 1999] Atanas Georgiev. Exploration of unknown environments by an autonomous mobile robot. In the World Wide Web, 1999. Presentación de su examen para la candidatura al doctorado.

[Gerkey *et al.*, 2004] Brian Gerkey, Andrew Howard, and et al. The player/stage project. `http://playerstage.sourceforge.net/`, 2004.

[Grisetti *et al.*, 2005] G. Grisetti, C. Stachniss, and W. Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2443–2448, 2005.

[Gruschel, 2005] Jens Gruschel. Blend modes. `http://www.pegtop.net/delphi/blendmodes/`, Oct 2005.

[Hernández, 2005] S. Hernández. Navegación de un robot móvil en ambientes interiores usando marcas naturales del ambiente. Master's thesis, ITESM Campus Cuernavaca, 2005.

[Howard and Kitchen, 1996] A. Howard and L. Kitchen. Generating sonar maps in highly specular environments. In *Proceedings of the Fourth International Conference on Control, Automation, Robotics, and Vision*, pages 1870–1874, 1996.

[Howard and Roy, 2003] Andrew Howard and Nicholas Roy. The robotics data set repository (radish), 2003.

[Howard, 2004] Andrew Howard. Simple mapping utilities (pmap). `http://robotics.usc.edu/~ahoward/pmap/index.html`, 2004.

[Institute, 2005] Carnegie Mellon's Software Engineering Institute. Three tier software architectures. In World Wide Web, aug 2005.

[ITESM, 2005] Campus Cuernavaca ITESM. Cátedra de robótica móvil. `http://www.mor.itesm.mx/~robotica`, 2005.

[Jordan, 2002] M. Jordan. Probabilistic graphical models. 2002.

[Koenig *et al.*, 2001] S. Koenig, C. Tovey, and W. Halliburton. Greedy mapping of terrain. In *Proceedings of the International Conference on Robotics and Automation ICRA*, pages 3594–3599, 2001.

[Kraetzschmar *et al.*, 2002] Gerhard K. Kraetzschmar, Hans Utz, Stefan Sablatnög, and Stefan Enderle. Miro - middleware for cooperative robotics. *IEEE Transactions on Robotics and Automation, Special Issue on Object-Oriented Distributed Control Architectures*, 18:493–497, Aug 2002.

[Lacage, 2004] Mathieu Lacage. The glib object system. `http://www.le-hacker.org/papers/gobject/`, 2004.

[Maskell and Gordon, 2002] Simon Maskell and Neil Gordon. A tutorial on particle filters for on-line nonlinear/non-gaussian bayesian tracking. In *IEE Colloquium on Tracking*, 2002.

[Mazer *et al.*, 1992] E. Mazer, J. Ahuactzin, G. Talbi, and P. Bessiere. The ariadne's clew algorithm, 1992.

[Montemerlo *et al.*, 2002a] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002. AAAI.

[Montemerlo *et al.*, 2002b] Michael Montemerlo, Roy Nicholas, and Thrun Sebastian. Carnegie mellon robot navigation tookit. `http://www-2.cs.cmu.edu/~carmen/`, 2002.

[Morales Manzanares, 2004] Eduardo Morales Manzanares. Búsqueda, optimización y aprendizaje. `http://dns1.mor.itesm.mx/~emorales/Cursos/Busqueda04/principal.html`, 2004. Notas de curso.

[Murphy and Russell, 2001] Kevin Murphy and Stuart Russell. *Sequential Monte Carlo Methods in Practice*, chapter Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. Springer-Verlag, 2001.

[Murphy, 1999] Kevin P. Murphy. Bayesian map learning in dynamic environment. In *Advances in Neural Information Processing System (NIPS)*, volume 12, 1999.

[Murphy, 2002] Kevin P. Murphy. *Probabilistic Graphical Models*, chapter Dynamic Bayesian Networks. 2002.

[Nehmzow, 2003] Ulrich Nehmzow. *Mobile robotics : A practical introduction*. Springer, 2nd edition, 2003.

[Network, 2000] European Robotics Network. Orocos: Open robot control software. `http://www.orocos.org/`, 2000.

[Newman *et al.*, 2003] Paul M. Newman, Michael Bosse, and John J. Leonard. Autonomous feature-based exploration. In *ICRA*, pages 1234–1240. IEEE, 2003.

[OMG, 2003] OMG. *OMG Unified Modeling Language Specification m Version 1.5*, 2003.

[OMG, 2005] Object Management Group OMG. Corba basics. `http://www.omg.org/gettingstarted/corbafaq.htm`, 2005.

[Pacheco Sánchez, 2004] J. Antonio Pacheco Sánchez. Arquitectura de software para la integración de aplicaciones en robótica móvil, 2004. Artículo para el proyecto final de la materia de robótica móvil.

[Rekleitis, 2004] Ioannis M. Rekleitis. A particle filter tutorial for mobile robot localization. Technical Report TR-CIM-04-02, Centre for Intelligent Machines, McGill University, 3480 University St., Montreal, Québec, CANADA H3A 2A7, 2004.

[Romero Muñoz, 2001] Leonardo Romero Muñoz. *Construcción de mapas y localización de robots móviles: un enfoque probabilista*. PhD thesis, Instituto Tecnológico de Monterrey, Campus Cuernavaca, 2001.

[Sim and Roy, 2005] Robert Sim and Nicholas Roy. Global a-optimal robot exploration in slam. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Barcelona, Spain, 2005.

[Stachniss and Burgard, 2004] C. Stachniss and W. Burgard. Exploration with active loop-closing for fastslam. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[Stewart *et al.*, 2003] B. Stewart, J. Ko, D. Fox, and K. Konolige. The revisiting problem in mobile robot map building: A hierarchical bayesian approach. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, Siena, Italy, 2003.

[Thrun *et al.*, 2000] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, 2000.

[Thrun, 2000] S. Thrun. Probabilistic algorithms in robotics. *AI Magazine*, 21(4):93–109, 2000.

[Thrun, 2002a] S. Thrun. Particle filters in robotics. In *Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI)*, 2002.

[Thrun, 2002b] Sebastian Thrun. Robotic mapping: A survey. In G. Lakemeyer and B.Ñebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[Vaughan *et al.*, 2003] Richard T. Vaughan, Brian P. Gerkey, and Andrew Howard. On device abstractions for portable, reusable robot code. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2121–2427, Las Vegas, Nevada, U.S.A, Oct 2003. (Also Technical Report CRES-03-009).

[Welch and Bishop, 1995] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1995.

[Winer, 2005] Dave Winer. Xml-rpc specification. `http://www.xmlrpc.com/spec`, 2005.

[Wolf and Sukhatme, 2004] Denis Wolf and Gaurav S. Sukhatme. Online simultaneous localization and mapping in dynamic environments. In *Proceedings of the Intl. Conf. Robotics and Automation. ICRA*, volume 2, pages 1301– 1307, 2004.

[Yamauchi, 1997] B. Yamauchi. A frontier based approach for autonomous exploration. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 1997.

Apéndice A

Documentación del sistema

FSlammer es un sistema de cartografía robótica en línea autónoma. Está basado en *Player/Stage* y *PMAP Utilities*. Implementa una exploración glotona, bajo una arquitectura de software en tres gradas, con procesamiento asincrónico y concurrente. La aplicación se supervisa y se controla a través de una interfaz Web.

FSlammer está escrito en el lenguaje de programación C, utilizando en gran medida la biblioteca `GLib`.

A.1. Guía rápida de ejecución

El servidor de *FSlammer* se ejecuta de la siguiente forma:

```
$ fslammer <configfile.xml>
```

Si el nombre de archivo de configuración no se le especifica, intentará abrir el archivo `config.xml`.

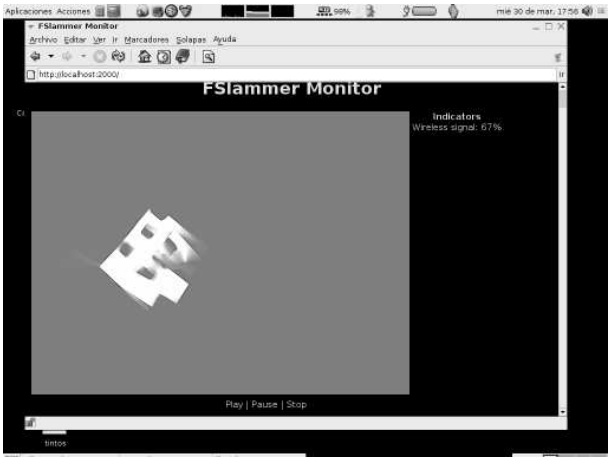


Figura A.1: Captura de pantalla de la aplicación

A.2. Guía de Instalación

Para instalar y ejecutar *fslammer* se proveen las siguientes instrucciones.

A.2.1. Dependencias de Software

El sistema *fslammer* tiene varias dependencias de software las cuales deben satisfacerse para la correcta compilación del sistema. La mayoría de estas bibliotecas son parte de los proyectos *GTK+/GNOME*. Recuerde que si utiliza una distribución de *GNU/Linux* basada en binarios, es necesario que tenga instalados los paquetes de desarrollo.

- Player (libplayerc client library)
- Glib-2.0 (GObject and GThread)
- lib-XML-2.0
- libsoup-2.2
- glut (optional pmap's dependency)

A.2.2. Obteniendo las fuentes del sistema

Los códigos fuente del sistema está disponible en el CVS del servidor `laplace.cva.itesm.mx`.

Es importante hacer notar que *fslammer* y *pmap* son dos paquetes distintos y hay que descargarlos de manera separada. El pmap que utilizada no es la versión oficial de Andrew Howard, sino una versión modificada para la fusión sensorial, es por eso necesario descargar la versión del CVS de `laplace.cva.itesm.mx`.

Después de descargar el *fslammer*, se necesita tener dentro del subdirectorio `pmap` la versión modificada del *pmap*. Esto se logra con el siguiente conjunto de instrucciones:

```
$ cvs -d ":pserver:anonymous@laplace.cva.itesm.mx:/lsi" login
$ cvs -d ":pserver:anonymous@laplace.cva.itesm.mx:/lsi" co fslammer
$ cd fslammer
$ rm -rf pmap
$ cvs co -d pmap ceyusa/pmap
```

A.2.3. Compilación

Si las dependencias están satisfechas correctamente, la compilación del programa deberá ejecutarse sin problemas. Tal vez sea necesario fijar la variable de entorno PKG_CONFIG_PATH a /usr/local/lib/pkg-config si es que se instaló el servidor `player` en /usr/local.

```
$ make
```

A.2.4. Configuración y parametrización

El sistema se configura a través de un archivo de configuración en XML que se le especifique en la línea de comandos, o `config.xml` por defecto.

El archivo de configuración de ejemplo se explica por sí mismo: la sección de robot describe cómo conectarse con el servidor `player` que controla al robot. Luego hay una sección por cada módulo del sistema, donde cada subsistema es parametrizado, por ejemplo, *pmapper*, donde se configuran los parámetros del subsistema de cartografía con filtros de partículas. Si alguno de estos parámetros no se especifican, el sistema tomará unos por defecto.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<robman>
    <robot>
        <address>127.0.0.1</address>
        <!-- address>192.168.100.10</address -->
        <!-- address>10.49.154.224</address -->
        <!-- address>10.49.147.139</address -->
        <port>6665</port>
        <id>myrobot</id>
        <!-- timeout to write a map graph on disk -->
        <mapcreation>5</mapcreation>
        <!-- maximum turn rate (grad/s) -->
        <!-- turnrate>0.25</turnrate -->
        <turnrate>0.08</turnrate>
        <!-- maximum velocity (m/s) -->
        <!-- maxspeed>0.3</maxspeed -->
        <maxspeed>0.2</maxspeed>
    </robot>

    <module>
        <name>mapper</name>
        <params>
            <!-- Maximum useable range value (e.g., 8.00 or 16.00). -->
            <param name="range_max">8.0</param>
            <!-- Angular step size for each successive range reading (should be M_PI / 180). -->
            <param name="range_step">0.0174532925</param>
            <!-- Number of samples in filter -->
            <!-- param name="num_samples">200</param -->
            <param name="num_samples">20</param>
            <!-- Width of occupancy grid (cells). -->
            <param name="grid_width">64.0</param>
            <!-- Height of occupancy grid (cells). -->
            <param name="grid_height">48.0</param>
            <!-- Size of each grid cell (m). -->
            <param name="grid_scale">0.050</param>
            <!-- Interval (map update steps) between resampling. -->
            <param name="resample_interval">-1</param>
            <!-- Standard deviation (sigma) value for resampling. -->
            <param name="resample_sigma">-1.0</param>
            <!-- use of lodo2. Avoid it in simulation -->
            <param name="lodo">1</param>
        </params>
    </module>
```

```
    <module>
        <name>monitor</name>
        <params>
            <!-- The TCP port to bind the http server -->
            <param name="port">2000</param>
            <!-- The directory where the docs are -->
            <param name="directory">./web</param>
        </params>
    </module>
</robman>
```

A.2.5. Organizar el ambiente robótico

Justo antes o después de configurar el servicio fslammer, hay que configurar un servidor *Player* que nos permita controlar un robot. Este robot puede ser simulado (*Stage*) o real.

A.2.6. Ejecución y operación del programa

```
$ fslammer <configfile.xml>
```

Para supervisar y operar el servidor de *fslammer*, se necesita un navegador del Web y abrir la URL `http://localhost:2000`, si *fslammer* está en la misma computadora que el navegador. El puerto TCP también puede variar según lo especificado en el archivo de configuración.

Una precaución importante es verificar que algún *firewall* no esté bloqueando el puerto TCP utilizado por el *fslammer*.

La página que muestra el servidor *fslammer* contiene el mapa construido hasta el momento por el sistema cartográfico, así como tres opciones:

- *Play* - Comienza el movimiento de exploración autónomo.
- *Pause* - Detiene el movimiento de exploración autónomo, pero sigue la tarea de cartografía. Esto es por si se quiere guíar al robot con algún tipo de palanca de mando.
- *Stop* - Detiene la ejecución de todo el sistema.

Estas tres opciones corresponden a los comandos que se pueden ejecutar a través de métodos HTTP:

- `http://localhost:2000/play`
- `http://localhost:2000/pause`
- `http://localhost:2000/stop`

Además se puede acceder a los mapas generados incrementalmente se utilizan los siguientes métodos HTTP:

- `http://localhost:2000/coarse.png` - para obtener la imagen en formato PNG del mapa fusionado.
- `http://localhost:2000/coarse-laser.pgm` - para obtener la imagen en formato PGM del mapa generado únicamente con láser.
- `http://localhost:2000/coarse-sonar.pgm` - para obtener la imagen en formato PGM del mapa generado únicamente con sonar.
- `http://localhost:2000/coarse-fusion.pgm` - para obtener la imagen en formato PGM del mapa fusionado.

A.3. Configuración de plantillas Web

Una de las opciones de *fslammer* es la capacidad de cambiar el aspecto de la interfaz Web a través de plantillas. Estas plantillas están dentro del subdirectorio `Web`, de las fuentes de *fslammer*, con terminación `tpl`.

La variable interna @INDICATORS@ contiene los indicadores de supervisión manejados por el sistema.

Estas plantillas en HTML pueden cambiarse por otras, que pueden estar en XML-RPC, texto plano o VXML, dependiendo de la aplicación deseada del robot.

A.4. Módulos del sistema

A continuación se presenta una sucinta descripción de cada módulo del sistema.

A.4.1. pconfig.h

Este módulo hace en análisis sintáctico del archivo de configuración, obteniendo las estructuras de datos que contendrán los parámetros necesarios para los demás módulos del sistema. Para hacer el análisis sintáctico se utiliza *libxml2*.

A.4.2. pcoordinator.h

El coordinador es el administrador de cada subsistema (*PRobot*, *PMapper*, *PExplorer*, *PMonitor*). Es responsable de administrar los hilos de procesamiento, asigna recursos y fija el estado del sistema de manera global, haciendo que los subsistemas tomen sus correspondientes subestados.

Las responsabilidades del coordinador son:

- Inicializar cada subsistema.
- Fijar los hilos de PMapper, PExplorer y lecturas del servidor player.
- Fija las llamadas asíncronas para PMonitor.
- Coordina los subestados de cada subsistema basados en el estado general.

A.4.3. pdevice.h

Es la representación de cada dispositivo del robot.

Cada dispositivo reconocido es inicializado en el constructor y desuscrito en el destructor de cada clase. Se fija un *signal* por cada dispositivo inicializado capaz de procesar cada ves que recibe nueva información de manera asíncrona.

A.4.4. perror.h

Este módulo contiene solamente una base de datos de los posibles errores del sistema. Esta información es utilizada por las estructuras GError.

A.4.5. pexplorer.h

Es la implementación de la exploración glotona.

A.4.6. pmapper.h

En este módulo se procesa las entradas de la odometría, láser y sonares para generar el mapa del ambiente utilizando las *PMAP Utilities*.

El módulo será inicializado en el hilo de procesamiento principal, donde registramos tres retrollamadas: una para la odometría, para el láser y para el sonar. Estas mediciones son la entrada de un objeto g_async_queue. La función **p_mapper_process_coarse** es ejecutada en otro hilo de procesamiento donde las pasadas mediciones son extraídas del objeto cola y procesados por los algoritmos de *pmap*.

A.4.7. pmonitor.h

Es la implementación del monitor Web. Abre un servidor Web en el puerto especificado en la configuración, y llama, de manera asíncrona, una función específica cuando una petición HTTP GET llega. El módulo muestra el mapa incremental actual, algunos indicadores de dispositivos y un control para establecer el estado global del sistema.

A.4.8. probot.h

En un envoltorio para la conexión a *Player* además de contener una lista enlazada de todos los dispositivos (pdevices) encontrados en el robot.