INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

TECNOLÓGICO
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 Computacionales

Asesor:

Dr. Eduardo Morales Manzanares

Coasesor:

Dr. Roberto A. Valdivia Beutelspacher

Cuernavaca, Morelos., Mayo 2005

Construcción de Mapas y Localización Simultánea con Robots Móviles

Presentada por:

Víctor Manuel Jáquez Leal

Aprobada por:

Dr. Eduardo Morales Manzanares

Profesor del Departamento de Computación

ITESM Campus Cuernavaca

Asesor de Tesis

Dr. Roberto A. Valdivia Beutelspacher

Profesor del Departamento de Computación

ITESM Campus Ciudad de México

Coasesor de Tesis

Dr. Sinodal Uno

Profesor del Departamento de Computación

ITESM Campus Cuernavaca

Sinodal

Dr. Sinodal Dos

Profesor del Departamento de Computación

ITESM Campus Cuernavaca

Sinodal

Resumen

En este trabajo de tesis se hace un análisis del problema de la construcción de mapas y localización simultánea con robots móviles. Se modela el problema como una red Bayesiana dinámica y se analizan los modelos gráficos probabilistas capaces de resolver el problema. Se abunda en el algorítmo de Filtros de Partículas, que es el método actualmente utilizado en la comunidad de la robótica móvil.

Las aportaciones hechas en este trabajo apuntan en tres direcciones: Una arquitectura de sofware que intenta explotar las características físicas de los robots móviles, como la asincronía de la información y la disponibilidad de procesamiento distribuido. En segundo lugar una estrategia de exploración del ambiente que intente obtener una representación del mismo siguiendo las restricciones del modelado del problema y sin exigir rutas precisas a las zonas desconocidas debido a la disparidad entre la construcción del mapa y la velocidad de movimiento del robot. Por último un modo de integrar al mapa, construido principalmente con el telémetro láser, información del sonar para agregar más elementos de ayuda en la navegación segura del robot en el ambiente.

Los resultados dieron una correcta construcción de mapas. La arquitectura propuesta es útil como marco de referencia para posteriores desarrollos, sin embargo no se modeló correctamentamente la arquitectura de capas en deterioro resultante en la reactividad del robot. La fusión sensorial tuvo resultados buenos, ofrenciendo mapas más útiles al momento de la planeación de trayectorías, dando también la capacidad de generar distintos mapas según el sensor. Finalmente la exploración autónoma, que funcionó bastante bien en simulación, pero que sin embargo no tuvo los mismos resultados en ambientes reales.

Índice general

Índice de Figuras

Índice de Tablas

Índice de Algoritmos

Capítulo 1

Introducción

En este se capítulo 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 llegan 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 autónomamente en ambientes dinámicos y complejos, es decir, desde espacios interiores como oficinas y casas, hasta exteriores como espacios terrestres, submarinos y sustentados sobre el aire. 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., 2002b].

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 correctamente 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. Distinguimos estas dos palabras porque la investigación sobre el tema ha dividido ambos problemas como se verá mas

adelante.

1.2. Justificación

El problema de la cartografía y localización y 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 utilizadas.

Para comenzar a describir el problema de SLAM, como ya se ha dicho, el objetivo 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 sensa su ambiente, el robot debe cambiar su ubicación para descubrir la topología del 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. Este problema 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 en lo absoluto confiables. Entonces, si deseamos integrar dos representaciones vistas por el robot en tiempos distintos, debemos 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.

Por estas razones el problema obliga resolver simultáneamente el problema 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 se utilizan, comenzando con filtros Bayesianos, pasando luego a los Kalman y

ahora los filtros de partículas.

Pero a pesar de todo el trabajo teórico realizado, actualmente existen pocos sistemas de software dedicados al problema. Anteriormente el software para la programación de robots móviles fueron soluciones parciales, restringidas en su alcance y en su hardware soportado, proyectos como por ejemplo Carmen [Montemerlo et al., 2002a] o Scene [Davison, 2001], los cuales están orientados casi exclusivamente a la cartografía. Actualmente se realizan esfuerzos para homogeneizar el acceso físico del robot 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. Por ejemplo, se publicó recientemente un desarrollo muy prometedor sobre la plataforma de Player/Stage llamado *Simple Mapping Utilites* [Howard, 2004]. Este desarrollo es la piedra de torque para este trabajo.

Por otro lado, existen esfuerzos por unificar diversos sistema 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.

Este trabajo se propone hacer un sistema de software para resolver el problema del SLAM, reutilizando los componentes de software que existen actualmente, pero no limitándose a ello, ya que también también propone un esqueleto para el desarrollo de sistema de robótica móvil de otras índoles, como navegación, interacción humano-robot, etc. Para ello se ha implementado la arquitectura de software 3+ [Pacheco Sánchez, 2004], tratando de ser lo más simple posible, así como buenas prácticas de programación y documentación.

En lo subsecuente este documento aborda la definición de los problemas a tratar, así como los objetivos e hipótesis.

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.

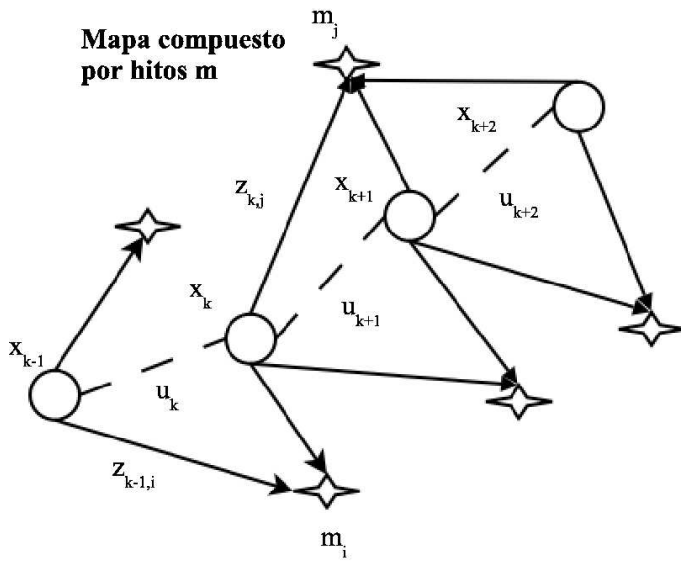


Figura 1.1: Descripción del problema del SLAM

Variable	Descripción
x_k	Posición del robot al tiempo k.
u_k	Vector de control aplicado al tiempo $k-1$ para mover al robot de x_{k-1} a x_k en el tiempo k.
m_i	Posición del hito (elemento del mapa) m_i
$z_{k,i}$	observación (medición) del hito m_i desde la posición x_k al tiempo k.

Tabla 1.1: Notación general 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 vector de control u_k 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 los objetos (m_i) que componen el ambiente visible para el robot.

Esta labor 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:

- *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.
- *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 más la descripción del problema del SLAM.

1.3.2. El problema de la arquitectura de software

Dentro de la Cátedra de Robótica Móvil del ITESM Campus Cuernavaca, se han hecho varios desarrollos individuales sobre esta área, cada uno construyendo desde cero sus aplicaciones y pruebas de concepto. Por lo que se decidió trabajar en formas de integrar éstos y futuros trabajos e ir subiendo en abstracciones, sin tener que rehacer lo cimientos constantemente.

Uno de estos trabajos, que inicialmente se estudiaron para el propósito de esta investigación, es la tesis doctoral [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 que se subsumen (Figura 1.2). Sin embargo el enfoque que actualmente se usa es el orientado a tareas, donde los módulos que descomponen el problema son las subtareas o servicios a realizar.

La arquitectura orientada a servicios se topa con nuevos retos, ya que no intuye la separación de abstracciones en gradas, sino que supone que cada tarea trabaja

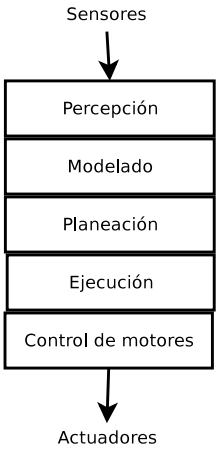


Figura 1.2: Descomposición tradicional en módulos funcionales.

independientemente con los sensores y actuadores, y no menciona la manera de resolver conflictos cuando dos servicios quieren trabajar en el mismo momento.

En el último año, uno de los tesistas de maestría que trabajan 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).

Así 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, 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 queremos que un robot interactúe entre humanos, se quiere que haga un conjunto

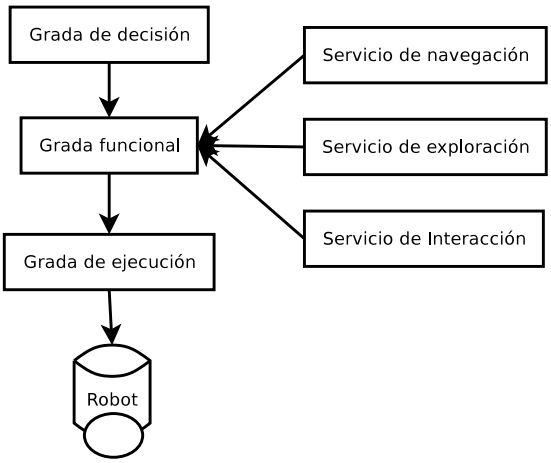


Figura 1.3: Arquitectura 3+

de tareas interrelacionadas de manera concurrente, que vistas como un Proceso de Decisión de Markov (MDP por sus siglas en inglés).

Este trabajo de tesis se limitará a proponer un simple autómata determinista para la coordinación de las subtareas involucradas en la exploración.

1.4. Objetivos

Objetivo General

Desarrollar un sistema de software para la cartografía y localización simultanea de robots móviles, 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.

- Proponer un mecanismo de fusión sensorial entre la información del telémetro láser y el sonar.
- 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.
- Implementar un algortimo de exploración que no haga uso del mapa incremental.

1.4.1. Modelo Particular

El objetivo de la ciencia no es responder a todas las preguntas, sino escoger las respuestas más importantes. Hay que tener las respuestas y probarlas. El objetivo de la tesis no esta en generar otro método para construir mapas y explorar, sino probar alguno ya existente y proveer las bases para probar otros más eficientes. No sólo nuevas ideas son necesarias para hacer ciencia, también someter las ya existentes lo es.

El modelo expuesto 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:

- Parametrización del sistema por un archivo XML.
- Cliente delgado para el monitoreo del proceso.
- Coordinador con comunicación con el monitor a través del protocolo HTTP.
- Cada subproceso de la construcción del mapa se ejecutará en una 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.
- Un método de exploración *greedy* que permita un recorrido los más completo posible del ambiente en un tiempo aceptable.

En el siguiente diagrama (Figura 1.4) se esboza una vista de despliegue 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.

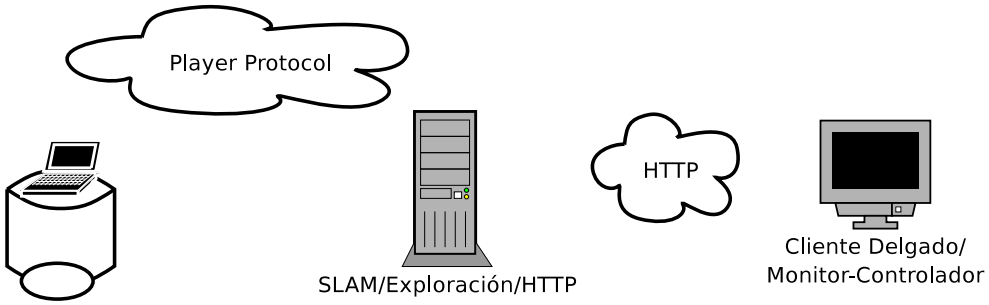


Figura 1.4: Vista de despliegue del sistema

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, agregando información adicional para facilitar una navegación y planeación de rutas más seguras. Este agregado se hará de manera adicional al proceso de SLAM ya provisto que utiliza telémetro láser y el mapa de rejilla de ocupación. 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 se usa la información del sonar, ya que éste percibe una distancia menor que la láser, siendo más conforme a la realidad. Esta información se agrega al mapa, generando una más exacta representación del ambiente.

Por el lado de exploración autónoma, se considera que la estrategia glotona (*greedy*) es, si no óptima, sí es muy cercana a ésta. 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 computación. Por otro lado tenemos que, con los algoritmos de SLAM, los mapas convergen más rápidamente cuando se cierran ciclos, y por ello tenemos que decidir entre cerrar ciclos o ir a fronteras.

Ahora, para decidir la ruta a la frontera, se debe tener un mapa, y este se esta construyendo. El procesamiento necesario para la construcción es grande y nuestro robot pasaría muchos tiempos muertos esperando a que el mapa corresponda a su posición actual. Es por ello que proponemos movimientos más o menos ciegos con relación al mapa y hacer eficiente el uso de energía del robot y el tiempo de exploración.

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 de exploración que será los 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.

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 del mundo probabilista para describir su operación. A modo de introducción a este mundo probabilista, 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 rematar 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 con interacción humana.

Inicialmente los investigadores se dedicaron a los problemas de planeación y control, suponiendo que las operaciones del robot eran perfectas [Thrun, 2002a], así como el completo modelado del ambiente. 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 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, si no también en casi todos los niveles de percepción, comportamiento y decisión.

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 percibir a los sensores, actuadores y al mundo, no como variables exactas, sino como distribuciones de probabilidad, ya que tomamos en consideración la incertidumbre asociada.

Más precisamente, las ideas de probabilidad se encuentran en 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 carece 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.

2.2.1. Redes Bayesianas

Una red Bayesiana es un grafo acíclico dirigido de nodos que representan variables 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, 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 sean $X_1, \cdots, X_n$ y sea $padres(A)$ 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.

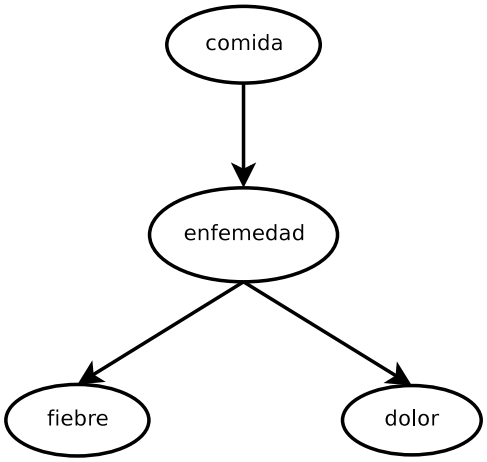


Figura 2.1: Ejemplo de una red Bayesiana

Para poder hacer cálculos es necesario especificar para cada nodo X la distribución de probabilidades de 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, lo cuales reciben información a continuos instantes 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 intervalo 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.

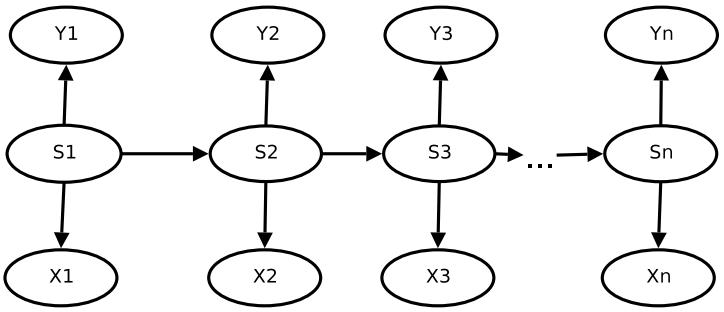


Figura 2.2: Ejemplo de una red Bayesiana dinámica

Para representar el estado no observable u oculto en una DBN se utiliza un conjunto de N_h variables aleatorias, $Q_t^{(i)}, i \in \{1, \cdots, N_h\}$, 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 podemos ver a las redes Bayesianas dinámicas, como un conjunto de red estáticas unidas por 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 Bayesina temporal (2TBN por sus siglas en inglés), al que llamaremos $B_{\rightarrow}$. Los modelos de transición y de observación son entonces definidos como el producto de los distribuciones de probabilidad condiciones 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 estamos restringidos 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 denotaremos 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 tengamos 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. 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 y por ende sus operaciones para inferir [Arulampalam et al., 2002], en comparación del caso general.

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 (mapa) 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 pueden 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 podemos 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.

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 es imposible de controlar y predecir.

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.

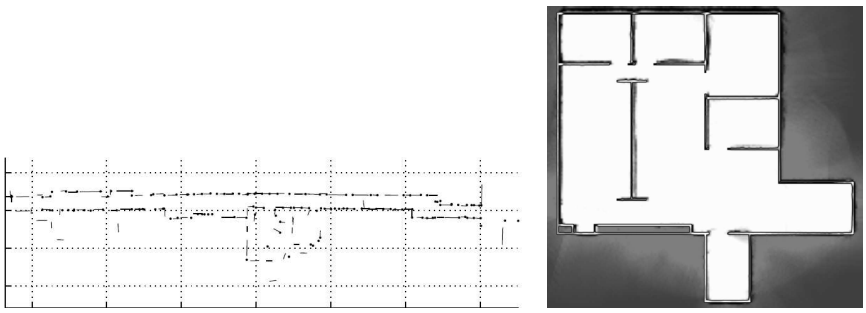
La tendencia general en la literatura del SLAM es utilizar representaciones geométricas. No obstante, este tipo de representación tiene varios estilos de implementación, dependiendo de la categorización del ambiente:

- *Descripción explícita del espacio libre*. También conocida como *modelo de rejilla*, donde las celdas tienen una probabilidad de ocupación, y los sensores tienen un modelo probabilista, además de facilitar 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.



(a) características (b) rejilla

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

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.

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.

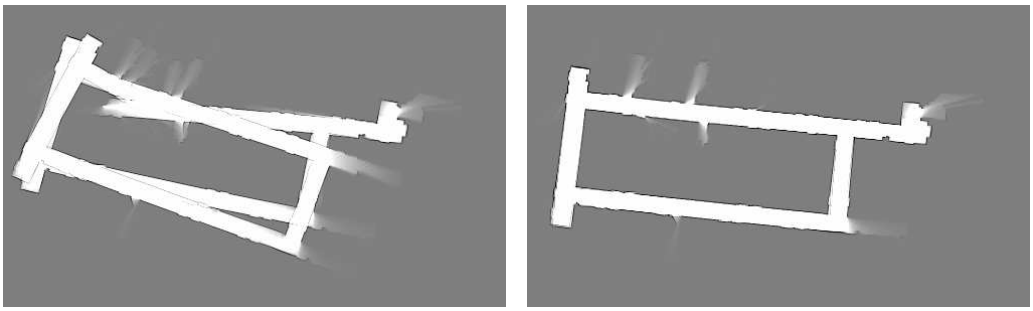
Sin embargo los sensores dentro del robot tienen una percepción de la realidad limitada a cierta cantidad de espacio y restringida por obstáculos, además están sujetos a ruido que genera errores de medición. Dadas estas limitaciones de alcance, el robot debe moverse en su ambiente para registrar áreas aún desconocida 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 únicamente la información odométrica del robot. La figura 2.4(b) es un mapa aprendido utilizando filtro de partículas.

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 asocación de datos, se tratan de identificar los puntos que ya se habían visto previamente para correlaccionarlos 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.

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

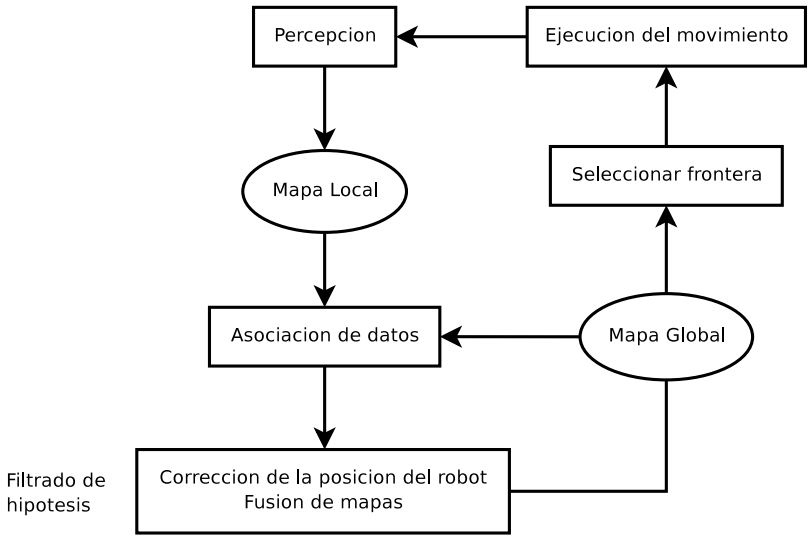


Figura 2.5: Procedimiento General del SLAM Incremental [Chatila, 2005]

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 un ambiente de dos dimensiones, el número de elementos llega a miles, donde cada punto registrado en un elemento a describir. En un ambiente de tres dimensiones estos elementos llegan a ser de millones.

- *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 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 es 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, donde el

ambiente 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, hay que 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., 2002b] a 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 $u_1, u_2, \cdots, u_t$. 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 los hitos $l_1, l_2, \cdots$, que deben ser asociados con hitos 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 $[t-1, t)$.

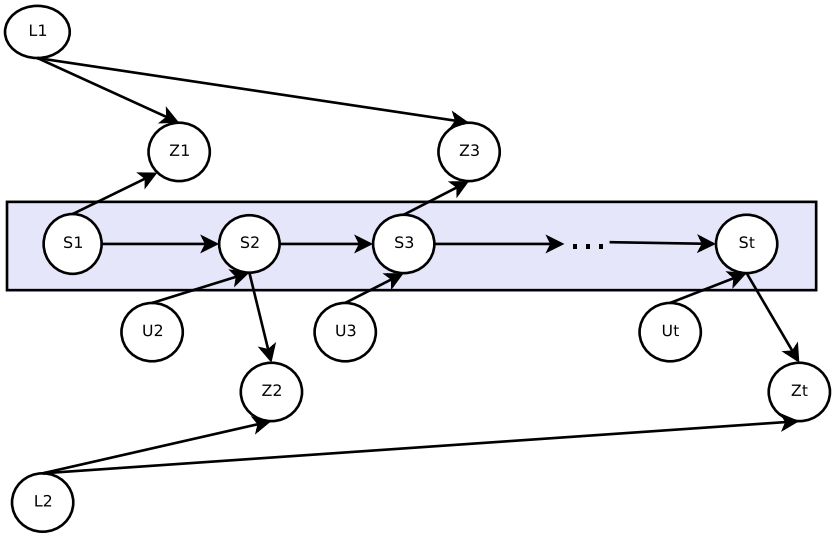


Figura 2.6: Modelo gráfico probabilista del problema del SLAM

2.5.2. Formulación del problema del SLAM

Con la visualización del SLAM como una red Bayesiana dinámica, ahora podemos tiene que formular analíticamente el problema del SLAM: El problema de la Cartografía

y Localización Simultánea es determinar el conjunto de todos los hitos l [1] que forman el mapa m y las posiciones del robot s_t dados las mediciones de los sensores y las instrucciones de control.

[Thrun, 2002b] establece que de manera general vemos el problema como una regla de Bayes:

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

Donde se desea aprender acerca de una cantidad x (v.gr. un mapa), basado en datos medición d (v.gr., barrido láser, 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.

Como se ya vio, 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

[1] Como ya se mencionó, los hitos son puntos en un plano

observadas directamente.

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, tenemos:

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

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:

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

Haciendo uso de la propiedad Markoviana propia del problema, podemos saber la posición del robot utilizando la posición en el instante anterior:

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

Podemos suponer que $p(x_{t-1}) = p(x_{t-1} \mid z^{t-1}, u^{t-1})$ considerando que exponente ^{t} 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 sustituimos 2.3 en 2.2, tenemos 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, obtenemos el siguiente filtro de Bayes:

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

La mayoría de los algoritmos cartográficos asumen un mundo estático, lo que 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:

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

Se advierte que para este estimador no se requiere una integración del mapa m, como se mostró en (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.

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 de movimiento es un generalización probabilística invariante 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 son las diferencias entre los diferentes algoritmos de SLAM existentes.

En el siguiente capítulo se expondrán de manera genérica los mecanismos de inferencia para redes Bayesianas dinámicas y las herramientas matemáticas a utilizar.

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, 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. Sin embargo, este algoritmo tiene algunas desventajas que también se revisarán.

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 que estemos utilizando. La inferencia a utilizar dependerá de la clase de solución que estemos buscando. De entre las inferencias

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

disponibles podemos encontrar las más importantes listadas en la Tabla 3.1 [Murphy, 2002].

Inferencia	Descripción
Filtrado	Calcular $P(X_t \mid y_{1:t})$ por ejemplo supervisado o 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 modelos gráficos probabilistas

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 [Murphy, 2002]. La Tabla 3.2 enumera algunos de estos algoritmos y sus propiedades.

En 1987 Cooper 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-Duro [Charniak, 1991]. Al crecer la cantidad de datos manejadas en las distribuciones de probabilidad, el tiempo de ejecución crece exponencialmente. Así que cuando se tienen modelos cuyos estados ocultos que tienen distribuciones continuas o mixtas (discretas en unas variables, continuas en otras), la 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

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

de la red a tratar. En este momento no existe una única técnica, ya sea aproximada o exacta, que funcione bien en cualquier tipo de red.

3.1.1. Filtros Kalman

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 [Welch and Bishop, 1995].

Los filtros Kalman, como se le llama a dicha técnica, son la herramienta con más fundamento matemático y con resultados más exactos a la hora de hacer un seguimiento del estado en una DBN, lo que los convierten en la técnica a utilizar en este tipo de

problemas.

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.

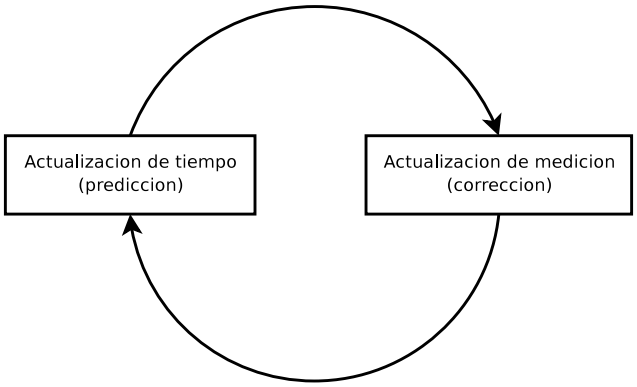


Figura 3.1: Ciclo del filtro de Kalman. La actualización del estado proyecta el estimado actual del estado hacia adelante en el tiempo. La actualización de medición ajusta el estimado proyectado con una medición en ese tiempo.

Una crítica a los filtros de Kalman es que sólo pueden estimar el estado de un proceso

controlado en tiempos discretos dirigidos por una ecuación lineal estocástica. 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 (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 se 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 mantienen toda la incertidumbre en el mapa el 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.

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

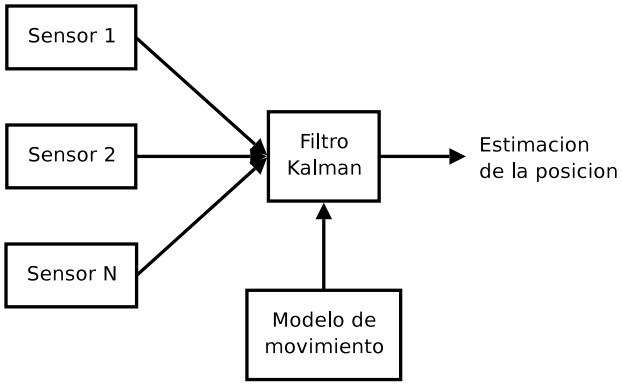


Figura 3.2: Algoritmo del filtro de Kalman para el problema de la estimación de la posición de un robot [Chatila, 2005]

- La complejidad para cada actualización es de $O(K^2)$, donde K es el número de 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 $O(K^2)$, donde K es el número de características en el mapa [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 hitos exactamente iguales. Medir tales hitos 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 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. Uno luego mantiene 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 nuestros nodos son distribuciones continuas, no monotónicas o muy complejas para ser evaluadas usando técnicas analíticas, tendremos que echar mano de las técnicas Monte Carlo.

El problema fundamental que aquí se enfrenta es encontrar el valor esperado de alguna función $f(x)$ con respecto a una distribución de probabilidad $p(x)$ [Jordan, 2002]. 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 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.

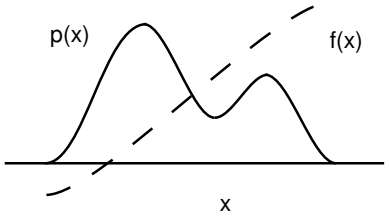


Figura 3.3: Función $f(x)$ cuya valor esperado será evaluada con respecto a una distribución $p(x)$

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

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

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

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

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 relativo número 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 nos 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 hacemos 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. 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 las variables de datos y descarta aquellas muestras que no concuerdan con los datos observados.

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.

La idea fundamental es representar la función de distribución posterior buscada con un conjunto de muestras aleatorias con pesos asociados y calcular estimados basados en estas muestras y pesos [Maskell and Gordon, 2002]. Al aumentar el número de muestras, esta caracterización Monte Carlo se vuelve una representación equivalente de la descripción funcional usual de la función de 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 puntos de soporte 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 $\sum_i w^i_k = 1$. La densidad posterior en k puede ser aproximada como en la Ecuación 3.2.

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

donde la función δ es la función de Dirac.

En otras palabras, se seleccionan aleatoriamente un conjunto de muestras $x^i_{0:k}$ obtenidas de superconjunto $x_{0:k}$ que representan todos los estados posibles en una distribución fácilmente muestreable. Las muestras seleccionadas son aceptadas o no como 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 del *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, $x^i_{0:k}$, 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

$$w^i_k \propto \frac{p(x^i_{0:k} \mid z_{i:k})}{q(x^i_{0:k} \mid z_{1:k})} \tag{3.5}$$

Para poder calcular la densidad de manera secuencial y manejarlo como un filtro con propiedad Markoviana, a cada iteración, se deberían tener muestras que constituyen una aproximación a $p(x_{0:k-1} \mid z_{1:k-1})$, y desear una aproximación $p(x_{0:k} \mid z_{1:k})$ con un nuevo conjunto de muestras. Si la densidad de importancia es seleccionada para factorizar tal causalidad de 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 $p(x_k \mid x_{k-1})$:

$$
\begin{aligned}
p(x_{0:k} \mid z_{1:k}) &= \frac{p(z_k \mid x_{0:k}, z_{1:k-1}) p(x_{0:k-1} \mid z_{1:k-1})}{p(z_k \mid z_{1:k-1})} \\
&= \frac{p(z_k \mid x_{0:k}, z_{1:k-1}) p(x_k \mid x_{0:k-1}, z_{1:k-1}) p(x_{0:k-1} \mid z_{1:k-1})}{p(z_k \mid x_{0:k}, z_{1:k-1})} \\
&= \frac{p(z_k \mid x_k) p(x_k \mid x_{k-1})}{p(z_k \mid z_{1:k-1})} p(x_{0:k-1} \mid z_{1:k-1}) \\
&\propto p(z_k \mid x_k) p(x_k \mid x_{k-1}) p(x_{0:k-1} \mid z_{1:k-1}) \qquad (3.7)
\end{aligned}
$$

Sustituyendo 3.6 y 3.7 en 3.5, la ecuación de actualización de pesos puede ser mostrada como:

$$
\begin{aligned}
w_k^i &\propto \frac{p(z_k \mid x_k^i) p(x_k^i \mid x_{k-1}^1) p(x_{0:k-1}^i \mid z_{1:k-1})}{q(x_k^i \mid x_{0:k-1}^i, z_{1:k}) q(x_{0:k-1}^i \mid z_{1:k-1})} \\
&= w_{k-1}^i \frac{p(z_k \mid x_k^i) p(x_k^i \mid x_{k-1}^i)}{q(x_k^i \mid x_{0:k-1}^i, z_{1:k})} \qquad (3.8)
\end{aligned}
$$

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:

$$
w_k^i \propto w_{k-1}^i \frac{p(z_k \mid x_k^i) p(x_k^i \mid x_{k-1}^i)}{q(x_k^i \mid x_{k-1}^i, z_k)} \qquad (3.9)
$$

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

$$
p(x_k \mid z_{1:k}) \approx \sum_{i=1}^{N_s} w_k^i \delta(x_k - x_k^i) \qquad (3.10)
$$

donde los pesos están definidos en 3.9. Puede demostrarse que si $N_s \to \infty$ la aproximación 3.10 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 1.

Algoritmo 1 Algoritmo del filtro de partículas SIS

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

Esboza $x_k^i \sim q(x_k \mid x_{k-1}^i, z_k)$

Pondera partícula, w_k^i de acuerdo a 3.9

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 excepto una partícula 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 $p(x_k \mid z_{1:k})$, es casi cero. Una medición aceptable de la degeneración del algoritmo es el tamaño efectivo de la muestra N_{eff}, y se define como

$$N_{eff} = \frac{N_s}{1 + Var(w_k^{*i})'} \tag{3.11}$$

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

$$\widehat{N_{eff}} = \frac{1}{\Sigma_{i=1}^{N_s}(w_k^i)^2} \tag{3.12}$$

donde w_k^i es el peso normalizado obtenido con 3.8. 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 que utilizaremos.

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

$$p(x_k \mid z_{1:k}) \approx \sum_{i=1}^{N_s} w_k^i \delta(x_k - x_k^i) \tag{3.13}$$

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 3.13, 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 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.

Aunque el paso de remuestreo reduce los efectos del problema de degeneración, introduce otros problemas prácticos. Primero, limita la oportunidad de paralelizar el proceso ya que todas las partículas deben ser combinadas. Segundo, las partículas que tiene altos pesos w_k^i son estadísticamente seleccionadas muchas veces. Esto conduce a una pérdida de diversidad entre las partículas así como la muestra resultante contendrá muchos puntos

Algoritmo 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

repetidos. Este problema, conocido como *empobrecimiento de la muestra*, se agrava en el caso cuando hay ruido ligero en el proceso. De hecho, para el caso de ruidos muy ligeros todas las partículas colapsarán a un sólo punto después de pocas iteraciones. Tercero, ya que la diversidad en las rutas de las partículas se reduce, cualquier estimación suavizada basada en la ruta de las partículas se degenera. Existen esquemas para contrarrestar este efecto, que no son consideradas para el trabajo actual.

Un filtro de partículas genérico se describe luego en el Algoritmo 3.

Algoritmo 3 Algoritmo genérico del filtro de partículas SISR

for $i = 1$ hasta N_s **do**
  Esboza $x_k^i \sim q(x_k \mid x_{k-1}^i, z_k)$
  Pondera partícula, w_k^i de acuerdo a 3.9
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 3.12
if $\widehat{N_{eff}} < N_T$ **then**
  Remuestrear usando el algoritmo 2
end if

En resumen, se pueden ver los filtros de partículas como un proceso de dos partes [Jordan, 2002]: Al un 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 3.4.

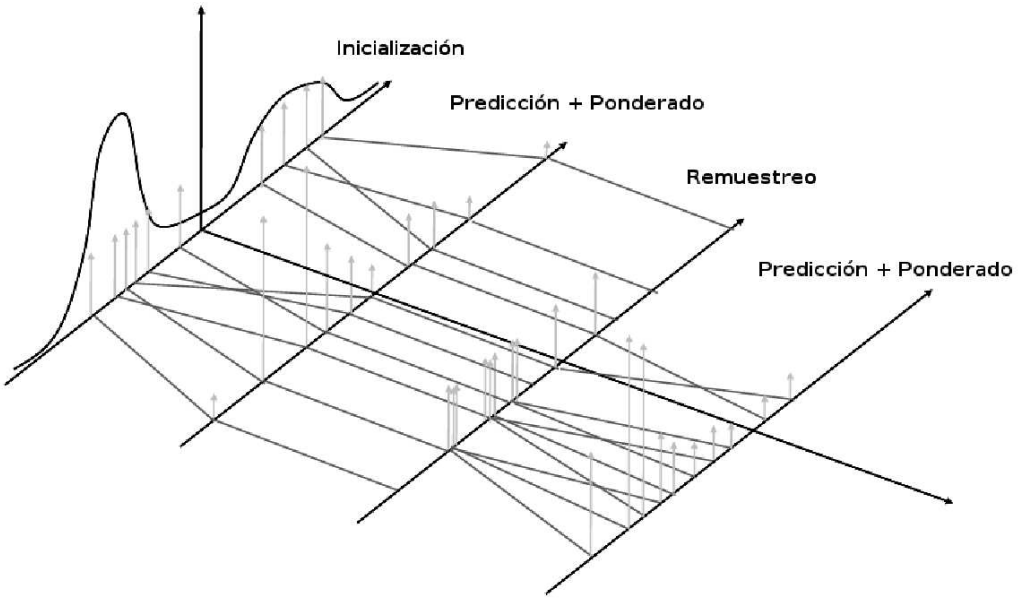


Figura 3.4: Esquema general de la operación de un filtro de partículas [Chatila, 2005]

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 prueba cómo mejorar cualquier estimador dado bajo cualquier función convexa [Murphy, 2002]. Su núcleo es la siguiente identidad:

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

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 podemos muestrear R y calcular el valor esperado de X dado R analíticamente,

necesitaremos menos muestras (para una mayor precisión). Desde luego, menos muestras no implican necesariamente menor tiempo, eso depende de si podemos 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 denotaremos como $r^i_{1:k}$, sino también una representación paramétrica de $P(X_k \mid r^i_{1:k}, y_{1:k})$, la que nombraremos 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 4.

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 esto aspecto se asemeja mucho los filtros Kalman estudiados en la Sección 3.1.1. La predicción esta en el modelo de movimiento del robot, generando las N partículas. La actualización esta 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

Algoritmo 4 Algoritmo Genérico de un Filtro de Partículas Rao-Blackwellizado

Muestreo secuencial con importancia

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

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

Asigna $(\hat{r}_{1:k}^{\prime i}) \stackrel{\text{def}}{=} (\hat{r}_k^i, r_{1:k-1}^i)$

end for

Evaluar los pesos de importancia

Normalizar los pesos

Remuestrea N muestras de $(\hat{r}_{1:k}^{\prime i})$ de acuerdo a la distribución de importancia $\tilde{w}_k^i$ para obtener N muestras aleatorias $(r_{1:k}^i)$ distribuidas aproximadamente a $p(r_{1:k}^i \mid y_{1:k})$

Cálculo analítico $\rightarrow$ Actualizar α_k^i dado α_{k-1}^i, r_k^i, r_{k-1}^i y y_k.

como posiciones presentes, a diferencia de 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 $u_{0:t}$ y utilizar esta distribución posterior para calcular una distribución posterior de mapas y trayectorias [Grisetti et al., 2005]:

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

Esto 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 $z_{1:t}$. Y puede hacerse esto 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.

A continuación se presenta una revisión de cómo trabajan las faces 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 genera un conjunto de partículas utilizando una distribución de fácil muestreo que se le conoce como *distribución propuesta* y se construye a partir del *modelo de movimiento*.

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., 2002b].

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} esta 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 es 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

$$\begin{aligned} \Delta npos.x &= \mathcal{N}(0, k_x * \Delta pos.x) \\ \Delta npos.y &= \mathcal{N}(0, k_y * \Delta pos.y) \\ \Delta npos.\theta &= \mathcal{N}(0, k_\theta * \Delta pos.\theta) \end{aligned}$$

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

El $\Delta npos$ se le suma a la posición actual del robot dentro de la partícula, dando la nueva posición en la partícula. La posición anterior pasa a ser parte de la trayectoria seguida por dicha partícula.

Un ejemplo simplificado puede verse en la Figuras 3.5 y 3.6. 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.

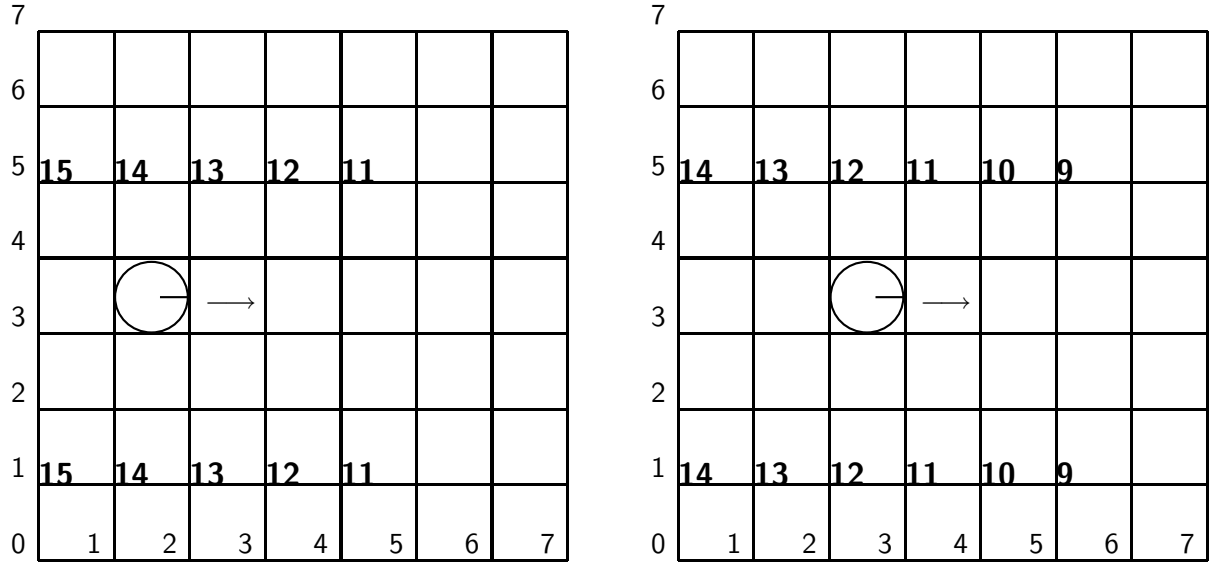


Figura 3.5: Posición inicial del robot en dos partículas y sus percepciones previas.

En la Figura 3.5 se muestran las dos partículas que se mantienen en el tiempo actual: $Pos_1 = [2, 3, 0°]$ y $Pos_2 = [3, 3, 0°]$. 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: $Pos_1 = [3, 3, 0°]$ y $Pos_2 = [4, 4, 0°]$ (Figura 3.6).

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.

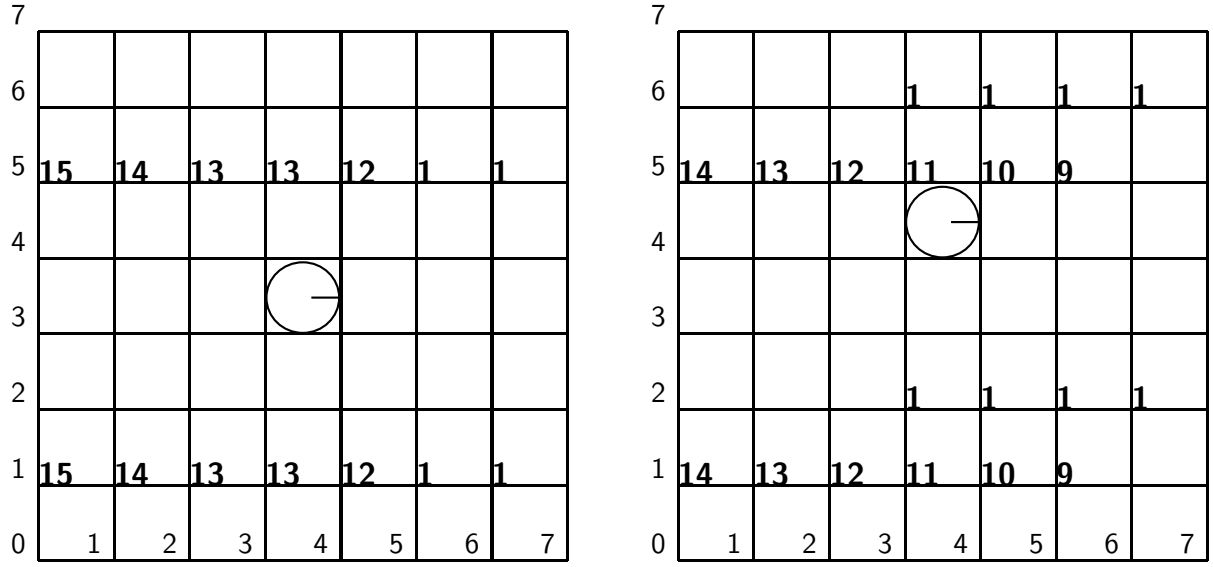


Figura 3.6: Dos partículas nuevas dado la distribución propuesta.

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:

$$w_t^m = \frac{\text{distribución objetivo}}{\text{distribución propuesta}} = \frac{p(s_{1:t}^m \mid z_{1:t}, u_{0:t})}{q(s_{1:t}^m \mid z_{1:t}, u_{0:t})} \tag{3.14}$$

Como ya dijimos 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-iésima 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 obtenemos la probabilidad de las observaciones dado el mapa anterior y la nueva posición, es decir la probabilidad donde concuerden nuestras 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 dicha celda. Razón por la que se observa números en las celdas ocupadas en las Figuras 3.5 y 3.6. 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.

Al inicio del programa, inicializa una tabla de búsqueda de los vecinos cercanos a la celda a evaluar. El tamaño de la tabla esta en función de la relación 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 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 la tabla sería de 3 columnas por 3 renglones, y su contiendo sería:

	-1	0	$+1$
-1	1,41	1	1,41
0	1	0	1
$+1$	1,41	1	1,41

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 pone la tabla de búsqueda como una ventana deslizante sobre el mapa, con la celda marcada como centro de la tabla de búsqueda y se va analizando de las celdas vecinas más cercanas a las más lejanas según la tabla. Si una de esas celdas tiene un valor de ocupación con un umbral mayor a 8, se toma su distancia reportada en la tabla de búsqueda y se multiplica por el tamaño de la celda y se suma en un acumulador que lleva el error total en el análisis de la celda en toda la tabla de búsqueda de vecinos.

La operación se repite para cada celda marcada por cada haz de luz de la lectura del láser. El peso de la partícula evaluada es la sumatoria cuadrada de los errores reportadas por cada celda evaluada en la partícula más su valor anterior.

En el caso de las dos partículas nuevas de ejemplo, si consideramos que el error previo es de 0 en ambos casos, la primera tiene un error de $Err_1 = 0{,}25$ y la segunda partícula un error de $Err_2 = 18{,}66$. Obviamente la partícula con menor error sería la tenga mejor ponderación.

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, como se verá en el proceso de remuestreo.

3.4.3. Remuestreo

En ciertos momentos las partículas con pesos muy pequeños son eliminados, que es el proceso de remuestreado [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 tamaños 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 uno fijo 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 3.12.

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, como ya se ha dicho, que se utiliza un filtro de partículas Rao-Blackwellizado.

Aquí hacemos uso de los algoritmos necesarios para construir nuestra representación del ambiente, dependiendo de nuestra representación utilizada. En el caso del enfoque empleado en esta tesis, la de rejillas de ocupación, estos algoritmos se ven 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 la representación del ambiente de la manera más conveniente para sus autores, sin embargo podemos establecer que de manera general los algoritmos de SLAM como PF siguen el ordenamiento del Algoritmo 5

[Murphy, 1999].

Algoritmo 5 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 $contador_ultimo_remuestreo > Umbral$ **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 nuestro 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, necesitaremos 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 Ronald, 2004], mejores métodos de remuestreo y preproceso del la información odométrica cruda para corregirla.

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).

4.1. Arquitectura del sistema de software

El sistema que se desarrolló es un software para la cartografía autónoma. Esta basado en un PMAP [Howard, 2004] modificado para funcionar en línea, además de integrar la fusión sensorial entre láser y sonar.

Se pensó en una 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.

4.1.1. Arquitectura de tres gradas

A mediados de los años 80 Brooks [Brooks, 1986] revolucionó la concepción de los sistemas robóticos móviles autónomos al introducir su arquitectura de subsunción, al romper con el enfoque tradicional de Sensado-Planeación-Acción (SPA), 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.

Estos problemas pueden verse como el resultado del método usado para administrar la información del estado interno del robot. El cómputo de alto consumo de tiempo, como la planeación o el modelado del ambiente generan estados internos cuya semántica refleja estados en el ambiente [Gat, 1997]. 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 a manejo de su estado interno: si no contienen estado, contienen estado reflejando memorias sobre el pasado, o contiene estados reflejando 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 evitado 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 deliberadora* o de planeación; son los que más tiempo de cómputo requieren y están orientados a la planeación y búsqueda [Gat, 1997].

Otro problema muy diferente, y que puede prestar a confusiones, es el problema de arquitectura de software para el procesamiento robótico, que lidia con problemas 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 encuentra la 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.

En nuestra arquitectura de gradas, 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 administración de servicios provee de los módulos de exploración y cartografía. Esto dicho se esquematiza en la Figura 4.1.

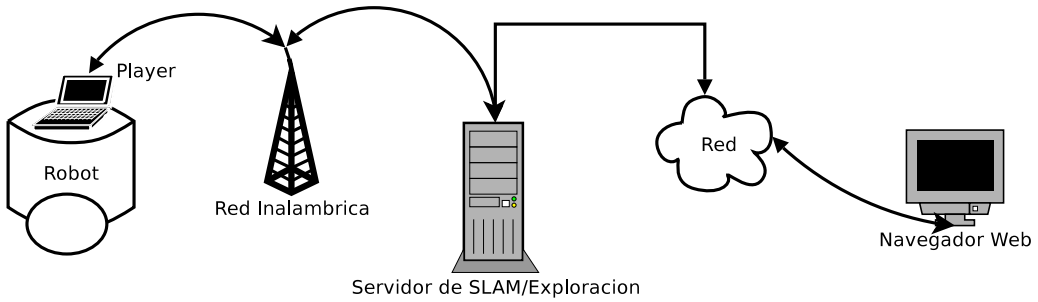


Figura 4.1: Vista de despliegue.

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, 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.

Para intentar zanjar este problema, se decidió abordar el problema aprovechando la misma asincronía de la actualización de la información en los sensores del robot. Como se visualiza en la Figura 4.2 cada vez que se actualiza la información cada dispositivo del sensor se envía 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.

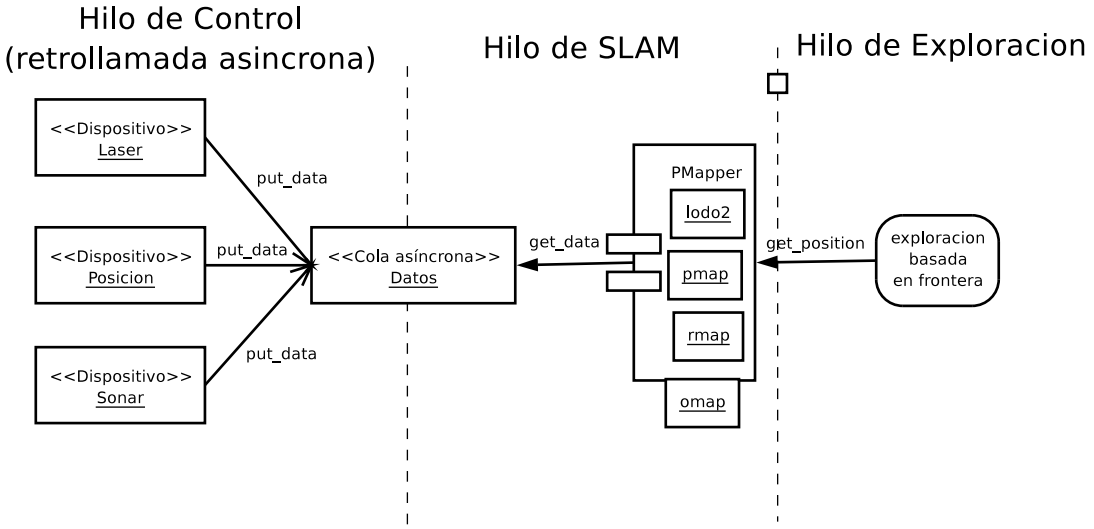


Figura 4.2: Vista de funcional.

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.

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 4.3 se observan la composición de agregación en la cual interactúan las clases principales.

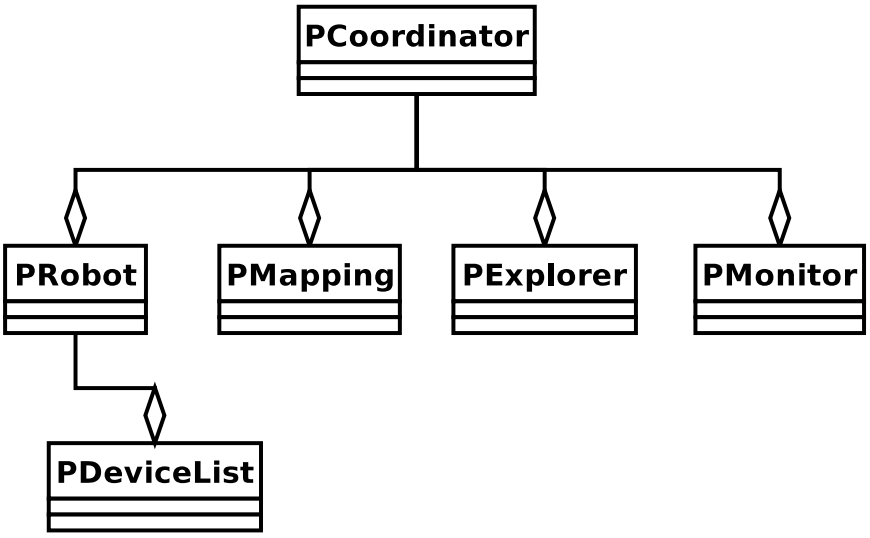


Figura 4.3: Vista de clases.

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 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 dispositivos y el ciclo de vida del robot en su conjunto.

PExploration 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 *PMapping* coordina las operación del Simple Mapping Utilities, las cuales fueron modificadas para que funcionaron como bibliotecas y trabajadas en línea. 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 4.4, nos muestra la descomposición de los hilos de procesamiento en el sistema.

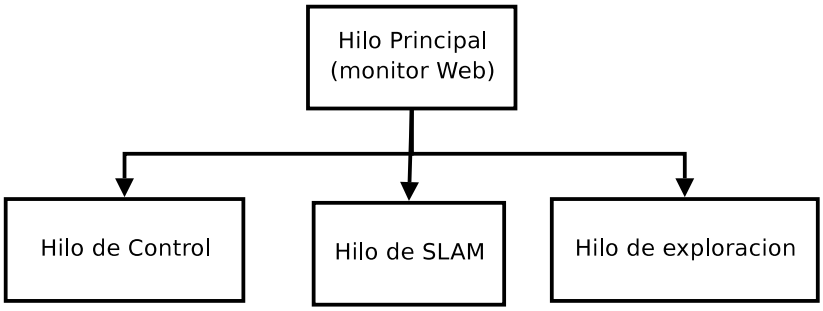


Figura 4.4: Vista de los hilos.

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 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. 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 inicializas 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.

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.

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 modificante la clase *PMapper*, o el algoritmo de exploración cambiando la clase *PExplorer*, también el comportamiento del monitor con retoques en la clase *PMonitor*.

Sin embargo no todo resultó como lo esperado en el periodo de diseño. Se observó que los errores y retrasos de transmisión datos por la red, 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 errores al no existir un buen acoplamiento entre la información de los sensores y la reacción de los actuadores. 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

El problema de la Cartografía con robots móviles se limita a obtener la posición del robot y a observar lo que le rodea, tomando a consideración lo que observó anteriormente [Newman et al., 2003]. La forma en 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 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

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 [Dudek and Jenkin, 2000].

Pensemos en un robot móvil A cuya configuración puede representarse completamente como $q = [xy\theta]$. 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*.

Estas notas puede apreciarse en la Figura 4.5. 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.

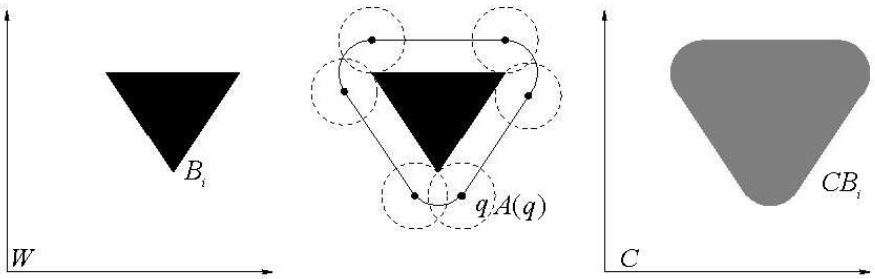


Figura 4.5: Dilatación de obstáculos en el espacio de configuraciones [Ahuactzin and Fraichard, 2005].

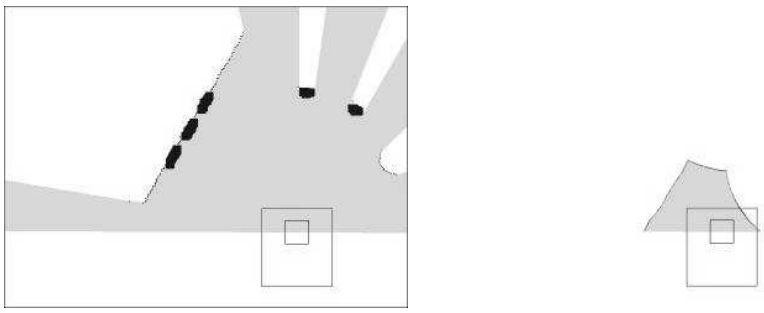
Podemos no tener la representación completa del ambiente, porque esta se obtendrá al final de la exploración, pero sí del lugar observado en la posición actual del robot. Esta representación se obtiene a través de sus sensores. A esta representación local se le pueden dilatar los objetos y obtener el 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, delimitando la porción libre de obstáculos. En la Figura 4.6 se observa como una lectura del láser, se acorta a una región totalmente navegable.

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 esta basado en la etapa de búsqueda del algoritmo del Hilo de Ariadna [Mazer et al., 1992].

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. No obstante este ángulo se selecciona con una política ϵ-greedy, ya que



(a) láser (b) láser C-Space

Figura 4.6: En la figura 4.6(a) se observa la lectura cruda de un telémetro láser. En la figura 4.6(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.

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 6.

El algoritmo asegura su completez gracias a la política ϵ-greedy. También esto permite que una región del mapa sea visitado más de una vez. No obstante la optimalidad del tiempo de construcción del mapa está perdida.

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, el espacio de configuración desaparece, dejando al robot girando tratando de encontrar un ruta para desplazarse.

Otro de los problemas, que se observaron más claramente en el simulador, fue que había regiones del ambiente que la exploración ciega ignoraba completamente. Para solventar tal caso, se detenía la exploración, se operaba con joystick el robot hasta la zona no explorada

Algoritmo 6 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**
      $Dist \leftarrow \sqrt{(\text{X anterior} - \text{X actual})^2 + (\text{Y anterior} - \text{Y anterior})^2} + Laser.Distancia$
      if $Dist > MaxDist$ **then**
        $Angulo = Laser.Angulo$
        $Distancia = Laser.Distancia$
      end if
    end for
  end if
  ejecutar giro al $Angulo$
  while obstáculos estén lejanos **do**
    ejecutar avance con $Distancia$
  end while
end loop

y se activaba de nuevo la exploración.

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 solo 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. Con tal esfuerzo se obtienen varios beneficios:

- 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 desambigüación de las propiedades del ambiente circundante.

- Resolución mejorada: Cuando múltiples e independientes mediciones hechos con con la misma propiedad esta 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, siendo realistas, la integración de sensores no es una solución óptima, ya que exhibe varias debilidades:

- 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 en el comportamiento del robot.

- 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 seguro eso 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 física de 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 no aparecer en las mediciones por completo. En cambio los telémetros láser, pueden perder de vista obstáculos que están fuera de su plano de sensado. La detección de espejos y puertas de vidrio son un problema también [Diosi and Kleeman, 2004] ya que son traslúcidos y por lo tanto indetectables.

4.3.1. Correlación de datos

Cuando la representación del ambiente es con mapas 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 [Castellanos et al., 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 ya marcada como obstáculo o zona libre.

Uno de los retos que presenta el problema del SLAM es la correlación de datos, donde distinguimos los objetos que componen el ambiente unos de otros. Esta labor es importante ya que de otra manera podemos 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 debemos fusionar las lecturas de los sensores con técnicas matemáticas más elaboradas como los filtros Kalman o algoritmos de clustering. Aunque, como ya se ha dicho, este trabajo hecha mano de los mapas de rejilla probabilista.

El problema de la correspondencia de datos podemos verlo 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). En 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.

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:

- Modelo probabilístico Bayesiano.
- Teoría de la evidencia.
- Conjuntos difusos.
- Enfoque histográmico de Borestein.
- 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 en radio medido $\Delta(t) = +1$ y en las celdas más próximas $\Delta(t) = -1$

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 no necesita que las observaciones sensoriales sean independientes, se incorporan todas. 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.

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. 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 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, nosotros decidimos extender el método propuesto por [Howard and Kitchen, 1996] y extendido 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 nuestra 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, obtenemos:

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

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

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

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 256 y el negro con 0, y los valores intermedios son niveles de gris.

Bajo este enfoque se puede decir que tenemos la probabilidad de espacio libre si normalizamos 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 podemos hacer 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.

Esta forma de fusionar dos o más mapas es idéntica a la mezcla de imágenes utilizando el algoritmo de multiplicación.

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 positivas y negativas tanto del sonar como del láser y su combinación en la fusión sensorial.

El primer caso se muestra en la Figura 4.7. 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 4.7: En la figura 4.7(a) se muestra un caso positivo de detección de un obstáculo por el sonar. Mismo obstáculo que fue detectado también positivamente por el láser (figura 4.7(b)). Esto resulta en una fusión sin complicaciones, donde ambas lecturas se combinan para dar una fuerte hipótesis de obstáculo (figura 4.7(c)).

El segundo caso, Figura 4.8, 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.

La Figura 4.9, 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

(a) sonar (b) láser (c) fusión

Figura 4.8: En la figura 4.8(a) se muestra un caso positivo de detección de un obstáculo por el sonar con especularidades, razón del cono. El mismo obstáculo no fue detectado por el láser (figura 4.8(b)). Esto resulta en una fusión con una alta probabilidad en la hipótesis de ocupación (figura 4.8(c)).

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 4.9: En la figura 4.9(a) se muestra un caso negativo de detección con el sonar, viendo un obstáculo donde no lo hay. En cambio, el láser sí pudo definir ese mismo espacio como libre (figura 4.9(b)). Esto resulta en una fusión con probabilidad macada en la hipótesis de ocupación (figura 4.9(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 4.10 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 4.10: En la figura 4.10(a) se muestra un caso negativo de detección, no viendo completamente un obstáculo que sí existe. El láser tampoco es capaz de percibirlo (figura 4.10(b)). Esto resulta en una fusión con probabilidad macada en la hipótesis de espacio libre (figura 4.10(c)).

En este Capítulo se han revisado las aportaciones realizadas que van desde la propuesta de una arquitectura de software robótico, con su prueba de concepto en un sistema de exploración en línea; luego esta el algoritmo de exploración ciego, el cual sólo utiliza el espacio de configuración local para descubrir si ambiente mientras construye el respectivo mapa; por último tenemos el esquema para la fusión sensorial utilizando un OR probabilístico.

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 presenta 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.

Por otro lado, que no afecta el número de partículas para decidir remuestrear o no, el remuestreo siempre se hace a intervalos regulares.

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 sensado por el robot. Esta bitácora se analiza por PMAP en un proceso fuera de línea generando el mapa que se observa en la Figura 5.1.

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

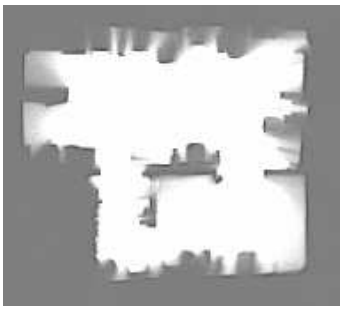


Figura 5.1: Laboratorio de Sistemas Inteligentes

sobre un ambiente real.

5.2. Ambientes simulados

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 desarrollo del 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 5.2 es prueba de ello. La operación de exploración se realizaba con un algoritmo sencillo de deambulado o con la palanca de mando virtual.

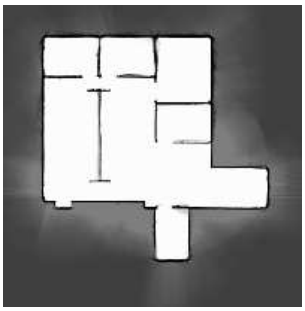
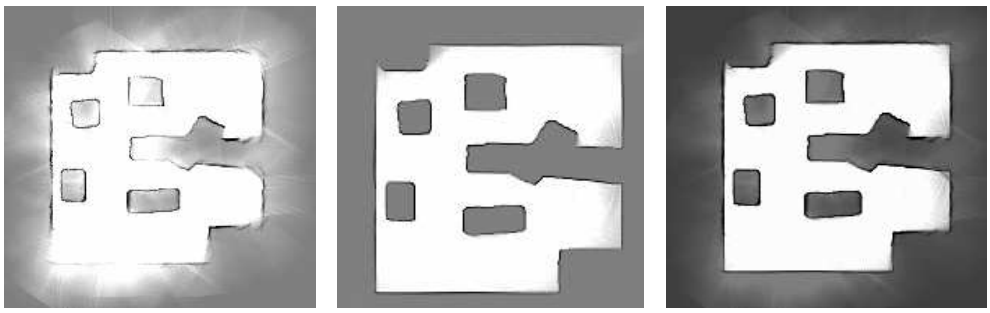


Figura 5.2: Ambiente simulado con espacios de navegación estrechos.

Los mapas en la Figura 5.3 representa un ambiente simulado con el cual se hizo muy trabajo de pruebas y experimentación. El que se muestra es uno que se hizo cuando ya se tenía la fusión sensorial.



(a) sonar (b) láser (c) fusión

Figura 5.3: Ambiente simulado disperso.

Los ambientes simulados fueron una constante de trabajo. Con ellos se probó el sistema en línea, el supervisado por Web, la fusión sensorial y por último la exploración autónoma.

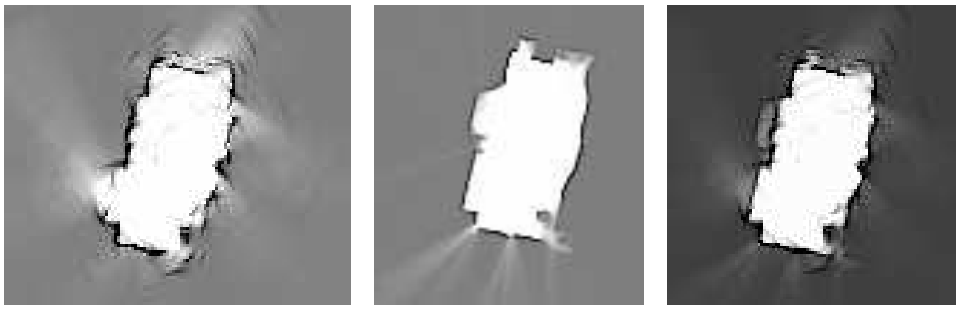
En cuanto a la exploración autónoma, se vio que el algoritmo respondía bastante bien en ambientes dispersos, con espacios abiertos grandes, tal como el caso de la Figura 5.3. Este mapa lo podía capturar completo, en tiempos de 10 a 20 minutos. En cambio, en ambientes con espacios libres estrechos, como el de la Figura 5.2, 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 esa regiones estrechas y continuar luego con la exploración autónoma.

5.3. Laboratorio de Robots Humanoides

El Laboratorio de Robots Humaniodes 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 5.4. 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.



(a) sonar (b) láser (c) fusión

Figura 5.4: Mapa de Laboratorio de Robots Humanoides utilizando sonar (5.4(a)), láser (5.4(b)) y fusión (5.4(c)).

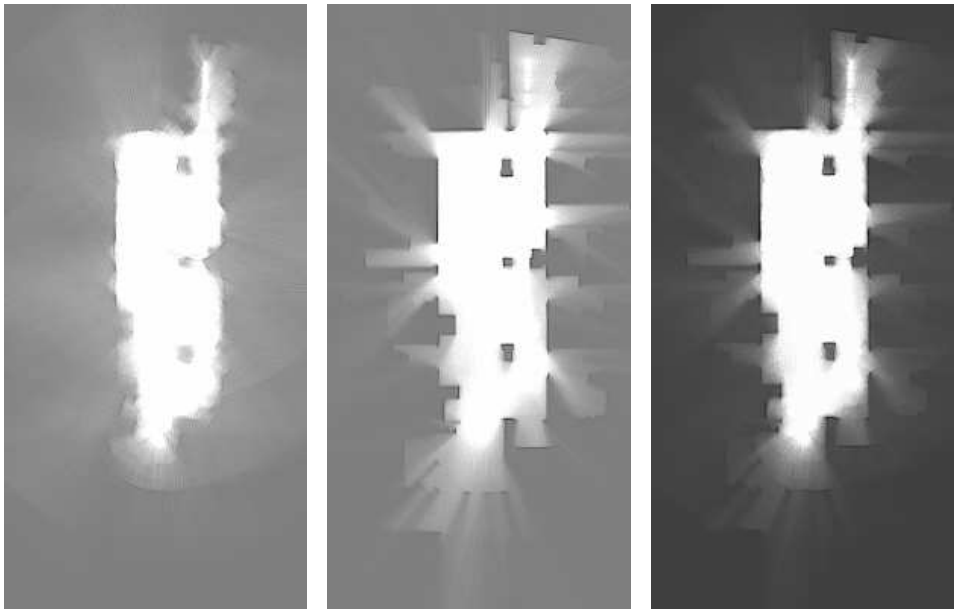
Este experimento fue la primera vez que se probó el sistema de cartografía en línea, con supervisión por Web, fusión sensorial y exploración. El sistema funcionó bastante bien.

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 5.5.

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.



(a) sonar (b) láser (c) fusión

Figura 5.5: Mapa del Departamento de Matemáticas del campus Cuernavaca. La figura 5.5(a) muestra el mapa utilizando únicamente sonar. La figura 5.5(b) muestra el mapa formado con la información del láser. La figura 5.5(c) muestra la fusión de ambos mapas.

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.

5.5. Centro Electrónico de Cálculo

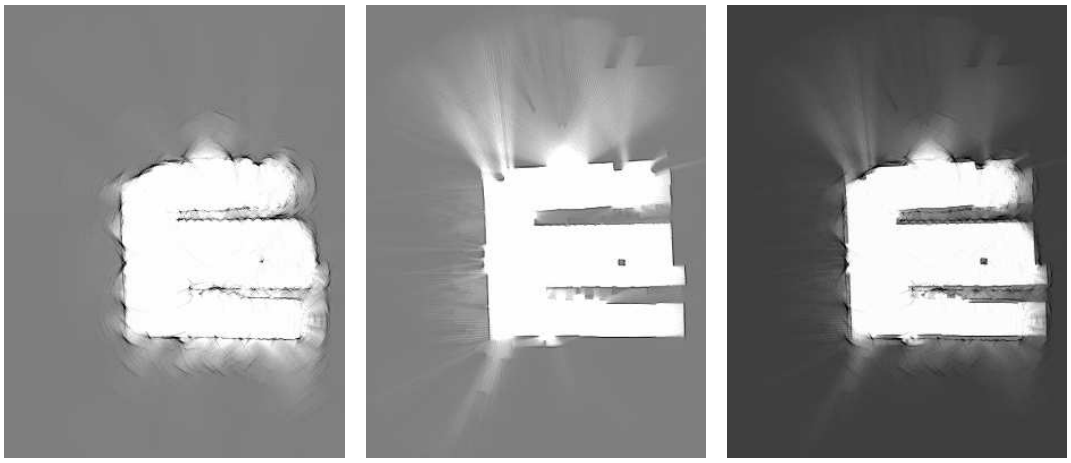
Este fue el último ambiente cartografiado. El CEC fue arreglado con papel cascarón y moviendo mesas para obtener un ambiente grande, estructurado, interior, con zonas tanto estrechas (pasillos extremos) como dispersas (pasillo central).

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 5.6.

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 reestructurar el modelado del cono de observación de los sonares logrando mapas más exactos. El resultado se ve en la Figura 5.6(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 el IIMAS de la UNAM, dentro de la categoría designada como "robots de servicio".



(a) sonar (b) láser (c) fusión

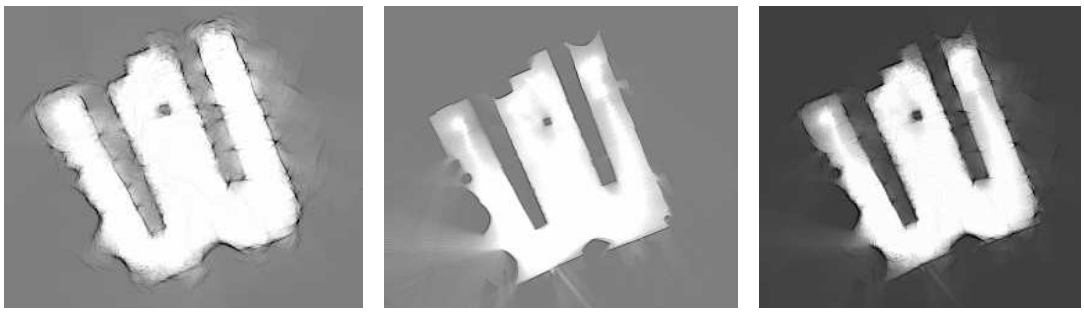
Figura 5.6: Mapa del Centro Electrónico de Cálculo (CEC) del campus Cuernavaca. En la figura 5.6(a) se muestra la generación del mapa de utilizando únicamente el sonar. En la figura 5.6(b) se muestra el mapa formado con el telémetro láser. Finalmente en la figura 5.6(c) se observa la fusión de ambos mapas.

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 ha probar los sistemas de manera independiente.

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.

El resultado se puede ver en la Figura 5.7. 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 5.7: Mapa de CEC durante el Primer Concurso Mexicano de Robótica.

Una vez revisados los experimientos realizados en los distintos ambientes, el siguiente

Capítulo se discutirán conclusiones obtenidas, para finalizar con un listado de áreas abiertas de investigación sobre el mismo camino de esta tesis.

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, esta 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, a pesar de que formalmente los filtros Kalman son los que más sustento matemático cuentan y serían la herramienta obvia a utilizar, sin embargo éstos carecen de todas las ventajas citadas, para el caso específico del SLAM.

No obstante permanecen abiertos varios problemas, en especial sobre el tipo de representación a utilizar. 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, las cuales son de fácil implementación, resultan muy ineficientes en espacio 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 debe especificarse previo a 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.

6.2. Arquitectura

La arquitectura de un sistema de robot de servicio es un tema que abarca de manera global todos los tópicos involucrados y tiene muchas perspectivas. Por un lado, la arquitectura de software clásica, analizando las distintas vistas y aplicando patrones arquitectónicos correctos. También esta distribución de tareas tal cómo lo describió [Gat, 1997].

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 no deseables 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 deliberadora con los planeadores de trayectorias.

Un tema importante aún en el tintero sería cómo 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 esta 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. Esta parte también es una ausencia de relevancia actualmente.

Sin embargo, es sistema actual representa un avance con respecto a los clásicos sistemas de Sensado-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

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 utilizando un sólo sensor, telémetro láser en el caso de la usada actualmente en la Cátedra, y utilizar el mapa con la fusión para la planeación de trayectorias, para garantizar un movimiento más seguro del robot.

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 clustering 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.

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 los movimientos en espacios estrechos. El algoritmo en sí es prometedor, valdría el esfuerzo implementarlo correctamente.

Aún así habría que buscar mejores estrategias para el movimiento del robot en ambientes desconocidos que mejores 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, 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.

6.5. Trabajo futuro

El trabajo futuro puede extenderse sin fin aparente. 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 capas de aprehender 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.

Bibliografía

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

[Ahuactzin and Fraichard, 2005] Ahuactzin, J. M. and Fraichard, T. (2005). Robotics and motion planning. Lecture slides in the Summer School of Image and Robotics.

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

[Austin and Overett, 2004] Austin, D. and Overett, G. (2004). Dave's robotic operating system. `http://dros.org/`.

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

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

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

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

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

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

[Côté et al., 2004] Côté, C., Michaud, F., and et al (2004). Mobile and autonomous robotics integration environment. `http://marie.sourceforge.net/`.

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

[Diosi and Kleeman, 2004] Diosi, A. and Kleeman, L. (2004). 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)*.

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

[Eliazar and Ronald, 2004] Eliazar, A. I. and Ronald, P. (2004). Learning probabilistic motion models for mobile robots. In *ICML*.

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

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

[Gerkey et al., 2004] Gerkey, B., Howard, A., and et al (2004). The player/stage project. `http://playerstage.sourceforge.net/`.

[Grisetti et al., 2005] Grisetti, G., Stachniss, C., and Burgard, W. (2005). 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.

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

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

[Institute, 2005] Institute, C. M. S. E. (2005). Three tier software architectures. In World Wide Web.

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

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

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

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

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

[Montemerlo et al., 2002a] Montemerlo, M., Nicholas, R., and Sebastian, T. (2002a). Carnegie mellon robot navigation tookit. `http://www-2.cs.cmu.edu/~carmen/`.

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

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

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

[Murphy, 2002] Murphy, K. P. (2002). *Probabilistic Graphical Models*, chapter Dynamic Bayesian Networks.

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

[Network, 2000] Network, E. R. (2000). Orocos: Open robot control software. `http://www.orocos.org/`.

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

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

[Rekleitis, 2004] Rekleitis, I. M. (2004). 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.

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

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

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

[Stewart et al., 2003] Stewart, B., Ko, J., Fox, D., and Konolige, K. (2003). 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.

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

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

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

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

[Vaughan et al., 2003] Vaughan, R. T., Gerkey, B. P., and Howard, A. (2003). 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. (Also Technical Report CRES-03-009).

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

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

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