SLAM
Estado del Arte

Federico Andrade y Martin Llofriu

Tutores:
Gonzalo Tejera
Facundo Benavides

Montevideo, Uruguay

Índice general

Índice de figuras

Capítulo 1

Introducción

En ese capitulo se presenta la definición de robot autónomo. Luego de presentan las motivaciones que llevan a los investigadores a trabajar en SLAM y el problema del SLAM propiamente dicho. Sobre el final del capítulo se exponen las principales dificultades que tiene esta rama de la robótica y sus diferentes clasificaciones.

1.1. Robots Autónomos

No existe una definición consensuada referente al significado de robot autónomo. A continuación citamos dos definiciones de autores conocidos:

- Funcionar autonomamente implica que un robot puede operar, autocontenido, en variadas condiciones y sin necesidad de supervisión humana. Que un robot sea autónomo significa que puede adaptarse a los cambios en el ambiente (p.e. se apagan las luces) o a problemas en si mismo (p.e. si se rompe alguna de sus partes) sin dejar de conseguir su objetivo [20]

- Un sistema es autónomo en la medida en que su comportamiento está determinado por su propia experiencia[23].

De ambas definiciones se desprende que la autonomía de un robot es necesaria para la solución de problemas planteados por la comunidad científica y la industria como el producto comercial Roomba[1], el desafío DARPA[2] y el auto sin conductor de Google[3].

De las definiciones se desprende también que la capacidad de navegar en ambientes desconocidos forma una parte importante de lo que significa un robot autónomo. En este sentido, uno de los principales problemas con los que se tiene que enfrentar un investigador

[1] http://www.irobot.com/
[2] http://archive.darpa.mil/grandchallenge/index.asp
[3] http://www.ted.com/talks/sebastian_thrun_google_s_driverless_car.html

o desarrollador de robots móviles es el problemas de la localización y confección de mapas del entorno. En la figura 1.1 que se presenta a continuación podemos ver un robot autónomo creado por la NASA, el cual se encuentra en un ambiente desconocido y sin tener información precisa sobre su ubicación. Sin embargo, la principal tarea de este robot es la navegación.

Figura 1.1: Imagen del robot Oportunity de la NASA en un entorno marciano. Imagen extraída de [24].

1.2. Motivación del Problema del SLAM

Existen ocasiones en las que se conoce de antemano el entorno en el cual se moverá el robot y es posible proporcionarle un mapa del mismo. Puede tomarse como ejemplo un agente que siempre limpia una misma habitación de un apartamento.
Por otro lado existen situaciones en las que un agente robótico debe moverse en un entorno desconocido, pero cuenta con información precisa sobre su ubicación. Para conocer su ubicación el agente puede utilizar, por ejemplo, visión global o gps, como se ve en las competencias robóticas como el sumo[4] o fútbol de robots[5]. En este caso, el robot deberá armar un mapa del entorno apoyándose constantemente en la información de su ubicación para tener mayores probabilidades de éxito. Sin embargo, existes casos, aún más complejos que los mencionados, en los que no se conoce el entorno y tampoco se tiene información sobre la ubicación exacta del robot. En este caso el robot deberá generar un mapa y mantener su ubicación en el mismo de forma concurrente. Esta tarea resulta compleja por el hecho que para poder localizarse de forma precisa se necesita un mapa, y por otro lado, para poder crear un mapa es menester estar localizado en forma precisa. Esta es la tarea que estudia el SLAM, localización y armado de mapas simultaneo. Algunos ejemplos en los cuales es importante realizar SLAM son:

- exploración espacial

[4]Por ejemplo http://www.fing.edu.uy/inco/eventos/sumo.uy/
[5]Por ejemplo www.robocup.org/

- rescate en zonas modificadas por catástrofes
- robots que realizan tareas domésticas
- autos dirigidos de forma autónoma

1.3. El problema del SLAM

El problema del SLAM (en español: Localización y armado de mapas simultaneo) aplica cuando el robot no tiene acceso a un mapa del entorno ni conoce tampoco su posición en el mismo. En la figura 1.2, por ejemplo, se observa un robot utilizado para tareas de rescate en una zona donde el humano no puede acceder debido a los elevados niveles de radioactividad, en este caso el robot no posee conocimiento de su ubicación ni del ambiente.

Luego, el robot solo dispone de la información proporcionada por las medidas obtenidas de los sensores y la noción de movimiento propio . El agente intentará obtener un mapa del entorno y simultaneamente localizarse relativo a dicho mapa[24]. En el contexto del SLAM existen diversas fuentes de incertidumbre, es decir factores que incrementan la dificultad de estimar la ubicación y mapa del entorno correctas. A continuación se incluyen algunos de estos factores:

- Ruido de los sensores: Los sensores utilizados en un robot presentan usualmente ruido en los datos que proporcionan.
- Desplazamiento impreciso del robot: El resultado de un movimiento del robot es de naturaleza no determinista. Esto se debe a que, por ejemplo, las ruedas de un robot pueden resbalar sobre el suelo. Como resultado, no es posible conocer a ciencia cierta como fue el movimiento real del robot como resultado de una orden de movimiento (p. e. avanzar, retroceder, girar) o en base a información de odometría.
- Simetrías en el entorno: El entorno sobre el que opera el robot puede presentar simetrías que dificultan la determinación de la posición actual del robot en el mapa confeccionado.
- Observabilidad parcial: La ausencia de un mecanismo de visión global del entorno dificulta la estimación de la posición y la confección del mapa.
- Entorno dinámico: Si se trabaja en un entorno dinámico, los cambios en el entorno dificultarán el proceso de estimación de la posición debido a que el mapa confeccionado puede estar desactualizado.

En pos de adaptarse a esta incertidumbre, la gran parte de las soluciones de SLAM plantea la solución de la estimación de la posición del robot y el mapa del entorno como distribuciónes de probabilidades.

Figura 1.2: Robot de la empresa IRobot realizando un rescate en un área de catástrofe. Imagen extraída de www.irobot.com.

En la actualidad existen métodos de SLAM robustos para mapear ambientes estáticos, estructurados y de tamaño limitado. Realizar SLAM en entornos dinámicos, espacialmente grandes y desestructurados sigue siendo un problema abierto a la investigación [24].

1.4. Desafíos

A continuación se presentan los principales desafíos que tiene el problema del SLAM.

1.4.1. Manejo de incertidumbre

Como se mencionó anteriormente, determinar la posición exacta del robot requiere conocimiento sobre la posición exacta del mapa. Por otro lado, la determinación del mapa del entorno requiere conocimiento de la posición exacta del agente. Dado que el agente no posee inicialmente ninguno de estos datos, y dado que la incertidumbre de su posición aumenta con su movimiento, el algoritmo de SLAM debe ser capaz de manejar cierto error en los datos que son computados. En la figura 1.3 se puede observar un robot que realiza una tarea doméstica para la cual no tiene un mapa ni tampoco su ubicación. Mientras realice la tarea de armar el mapa deberá manejar la incertidumbre de forma de mantenerse ubicado.

Esta incertidumbre debe ser manejada de forma tal que el error en las estimaciones no crezca constantemente (de manera de evitar una divergencia en la estimación de la posición del robot y del mapa).

1.4.2. Sensores

Los sensores son limitados en lo que pueden percibir y poco precisos en sus medidas. Estas limitaciones vienen dadas por muchos factores. El rango y la resolución de un sensor está sujeto a limitaciones físicas. Un claro ejemplo son las cámaras cuya calidad de imagen es limitada. Los sensores además están sujetos a ruido estocástico, lo que perturba las medidas de formas impredecibles y hace que la información extraída sea poco confiable. Otro

Figura 1.3: Imagen de un robot móvil situado en una posición inicial desconocida y en un ambiente desconocido. Imagen extraída de www.irobot.com.2

problema que incrementa el ruido en los sensores es el hecho de realizar medidas con el robot en movimiento, hecho que genera mayor error en las medidas de los sensores. Se puede apreciar en la figura 1.4 un sensor laser el cual tiene la capacidad de realizar barridos de 180° y medidas a distancias hasta de 100 metros[6].

Figura 1.4: Sensor laser comunmente utilizado para hacer SLAM. Imágen extraída de robotsinsearch.com.

[6]www.sick.com

1.4.3. Controles

En la fase de exploración del SLAM se envían ordenes de movimiento al robot, conocidos como controles. Una vez que se le envia una orden de movimiento al robot, los sensores en los motores (odometría) permiten conocer cual ha sido el movimiento que hizo el robot a partir de la orden que fue emitida. Esta información luego se procesará para actualizar la posición del robot. El problema ocurre cuando una de las ruedas del robot queda en el aire o simplemente resbala sobre una superficie con baja adeherencia, lo que implica que el robot no cambió su posición, o lo hizo pero en menor medida de la esperada, sin embargo los datos devueltos por los sensores de odometría indican que el movimiento fue completo. Esto genera un error entre la pose real del robot y su creencia de la posición. En la 1.5 se puede ver un robot con una rueda en el aire, producto de las irregularidades del entorno en el que se encuentra. Esta es una clásica situación en la que la información de odometría no se corresponde con el movimiento real del robot.

Figura 1.5: Imagen de un robot utilizado para limpieza doméstica subiendo una alfombra. Caso típico de quedar con una rueda en el aire o patinar. Imagen extraida de Wordpress.com.

1.4.4. Ciclos

Cerrar ciclos refiere a la situación en que el robot debe poder reconocer cuando pasa por un lugar que ya ha sido visitado. Realizar esta tarea con éxito se vuelve primordial en los mapas que poseen algún cruce. En la 1.6 se puede apreciar un escenario típico donde se presenta el problema de cerrar cíclos.

1.4.5. Asociación incorrecta

El agoritmo de SLAM debe ser lo suficientemente robusto como para no confundir dos lugares diferentes de forma que lo lleve a creer ques son el mismo. En caso de que el robot piense que dos lugares diferentes con características similares son efectivamente el mismo,

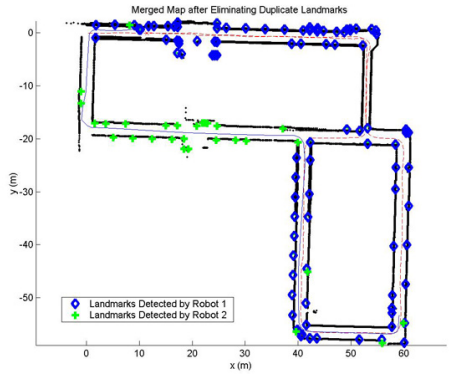


Figura 1.6: Típico escenario de pruebas donde el robot deberá tener un buen sistema para cerrar ciclos

cerrará un ciclo donde no lo hay y generará un mapa incorrecto del entorno lo cual puede llevar a errores catastróficos.

1.4.6. Capacidad de cómputo

Una gran limitación del SLAM es el procesamiento. Los algorítmos de SLAM suelen realizar tareas de procesamiento pesadas debido a la densidad de la representación del mapa y a la complejidad de los cálculos involucrados en la estimación de la posición. Esto muchas veces hace dificil el procesamiento dentro del robot y obliga a extraer los datos sensados para ser procesados afuera.

1.5. Clasificaciones de SLAM

A continuación incluimos algunas posibles clasificaciones del problema del SLAM.

1.5.1. Offline vs. Online

En el caso del SLAM online se procesa la información en el mismo robot mientras este navega en el entorno. Por otro lado, en el offline se realiza SLAM sobre un conjunto de datos que previamente fueron recuperados con algún robot, tanto de la medida de sus sensores como las medidas de odometría (movimiento).

1.5.2. Topológico vs. Métrico

Algunas técnicas de armado de mapas solamente mantienen la descripción de algunas características del entorno, que caracteriza la relación entre lugares. Estos métodos son conocidos como topológicos. Un mapa topológico se define por un conjunto de lugares diferentes y

otro conjunto que caracterizan las relaciones entre estos. Por otro lado, los métodos métricos proveen información métrica entre los lugares. En los últimos años los métodos topológicos han pasado de moda a pesar de la amplia evidencia de que los humanos utilizan a menudo información topológica[24].

1.5.3. Activo vs. Pasivo

En los algoritmos de SLAM pasivos, es otra entidad quien se encarga de controlar el robot, mientras que el algoritmo de SLAM es puramente observador.La gran mayoría de los algoritmos de SLAM son de este tipo[24]. En el caso de los algoritmos de SLAM activo, el robot explora de forma activa el entorno en busca de conseguir un mapa mas preciso en el menor tiempo posible. Existen técnicas híbridas donde el algoritmo de SLAM solo controla la dirección de los sensores y otra entidad se encarga de la dirección del movimiento del robot.

1.5.4. Estático vs. Dinámico

En el caso del SLAM estático se asume que el entorno no cambia con el tiempo, a diferencia de los métodos dinámicos que sí lo hacen. La gran mayoría de la literatura asume entornos estáticos[24].

1.5.5. Volumétrico vs. Basado en marcas

En SLAM volumétrico, el mapa es muestreado a una resolución que permite una reconstrucción fotográfica del entorno. En este caso el costo computacional es alto. Por otro lado, en el SLAM basado en marcas se extraen características de las medidas de los sensores de forma de armar el mapa en base a marcas dispersas (ver sección 2.2). Las técnicas que utilizan SLAM basado en marcas suelen ser mas eficientes ya que se descarta información de los sensores [24].

1.5.6. SLAM con un solo robot vs. Multirobot

La gran parte de los problemas están definidos para un solo robot, sin embargo, recientemente el problema de trabajar con más de un robot ha ganado mucha popularidad[24]. Los problemas para multirobots vienen de muchas maneras. En algunos casos los robots son capaces de observarse entre ellos. También se distinguen por el tipo de comunicación que utilizan. Los más realistas permiten que solamente los robots que estan más cerca se puedan comunicar.

1.5.7. Manejo de mucha incertidumbre vs. poca incertidumbre

Estos algoritmos se distinguen por la cantidad de incertidumbre que pueden manejar. Los más simples solo pueden manejar poca incertidumbre en la localización. Son útiles para los casos en los cuales un camino no se intersecta a sí mismo. En cambio para los mapas los cuales tienen lugares que se pueden alcanzar de muchas maneras (ver 1.6) es necesario que el robot pueda manejar mucha incertidumbre en su posición. La incertidumbre puede ser disminuida si el robot puede sensar información sobre su posición de forma absoluta. Un ejemplo puede ser mediante el uso sistema de posicionamiento global (GPS).

Capítulo 2

Fundamentos Teóricos

El problema del SLAM puede dividirse en varios módulos o subproblemas, siendo cada uno de estos un campo de investigación en sí. Este capítulo se organiza de la siguiente manera: se dedica una sección a la revisión de los dos grandes enfoques de la solución del SLAM, Bioinspirados y Probabilísticos, que determinan la naturaleza de la solución de cada subproblema del SLAM; luego se dedican las siguientes secciones a la discusión de los subproblemas más importantes y sus soluciones más comunes en los sistemas SLAM probabilísticos relevados.

2.1. Enfoques

Existen dos grandes enfoques para la solución del problema SLAM, Bioinspirados y Probabilísticos. Estos enfoques difieren principalmente en la forma en que procesan la información de entrada de forma de encontrar una buena estimación de la ubicación del robot y mapa del entorno.

Actualmente, el enfoque probabilístico domina el campo y ha logrado implementaciones que escalan a ambientes grandes y complejos[16]. Sin embargo, la gran capacidad de navegar que poseen mamíferos como las ratas, simios y seres humanos han motivado la investigación y desarrollo de sistemas que imitan los mecanismos biológicos de navegación de estos animales.

A continuación se incluye una descripción del enfoque bioinspirado seguida del probabilístico.

2.1.1. Bioinspirados

Desde la década de 1970 el entendimiento del funcionamiento del cerebro mamífero vinculado a las actividades de navegación y confección de mapas del entorno ha ido aumentando. Sin embargo, descubrimientos a partir del año 2005 en adelante han cambiado la forma en que se conciben los mecanismos mentales utilizados por los mamíferos para estas actividades.

Estos descubrimientos corresponden a el hallazgo de células especializadas en las actividades de navegación y reconocimiento del entorno, principalmente ubicadas en el hipotálamo. Estas células evidencian la existencia de una representación interna del entorno en el que el mamífero se desplaza.

Células de lugar

El primer gran hallazgo corresponde al descubrimiento de las células de lugar o "place cells"[21]. Las células de lugar son neuronas en el hipocampo que presentan una razón alta de disparo cuando el animal se encuentra en una posición específica del entorno que se corresponde con el campo de la célula de lugar. Es decir, que cada una de estas células estará asociada a una determinada zona del espacio, y se disparará cuando el animal se encuentre en dicha zona. La existencia de estas células plantea la hipótesis de que el hipocampo mantiene un mapa o representación interna del entorno.

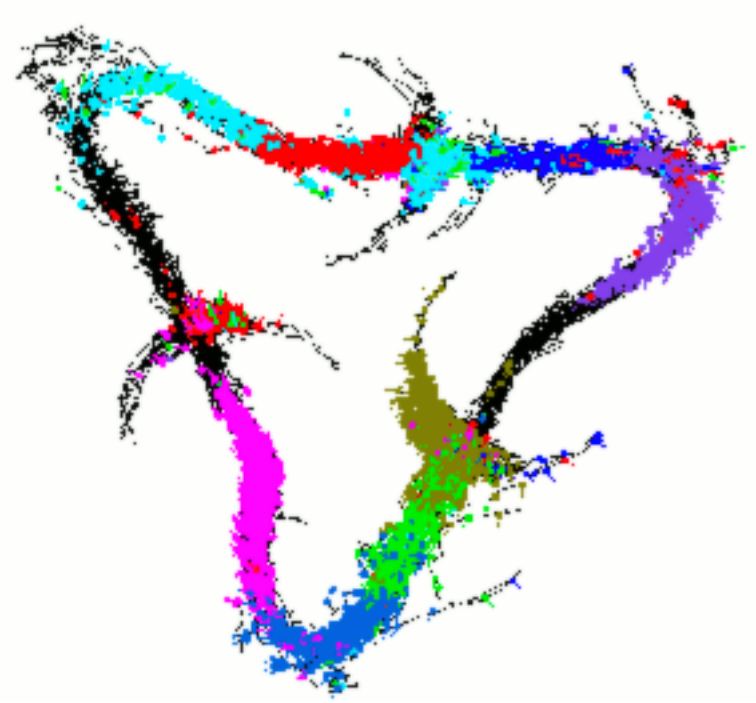


Figura 2.1: Patrones de disparo espacial. Imagen extraída de [25]

En la Figura 2.1 se muestran los patrones de disparo de siete células de lugar de una misma rata. La rata recorre varios cientos de vueltas en el sentido de las agujas del reloj alrededor de un laberinto triangular, parando en el medio de cada uno de los lados para comer una pequeña porción de comida. Los puntos coloreados indican los potenciales de acción (disparos de las neuronas), utilizando un color diferente para cada célula.

Aunque las señales visuales parecen ser el principal determinante de los disparos de las celdas de lugar, los disparos persisten en la oscuridad, lo que sugiere que la percepción u otros sentidos contribuyen también. En un entorno en el que la rata está restringida a una ruta lineal, las celdas de lugar tienen a menudo una componente direccional además de la componente de lugar.

Células Grilla

Estudios realizados en el año 2005 develaron la existencia de otro tipo de células asociadas a la representación espacial de las ratas, llamas células grilla[7]. Estas células neuronales presentan un patrón de disparo en forma de grilla regular formada por triángulos isósceles. En la Figura 2.2 se puede observar la frecuencia de disparo de tres células (derecha) en relación a la posición de la rata en el ambiente de trabajo.

Se ha demostrado que estas células mantienen su patrón de disparo tanto en presencia de estímulos externos o "allothetic cues", p. e. marcas, como en plena obscuridad. Sin embargo, la alineación de los patrones de disparo se ve afectada por cambios en la disposición de los estímulos externos. De esto se concluye:[7]:

- Es probable que existan mecanismos de integración del movimiento propio ("idiothetic cues") que mantienen el patrón en ausencia de estímulos externos.
- Si bien los patrones de disparo se mantienen en la obscuridad, la evidencia implica que es probable estos sean calibrados a partir de señales sensoriales de entrada externas de forma de corregir el error acumulado.

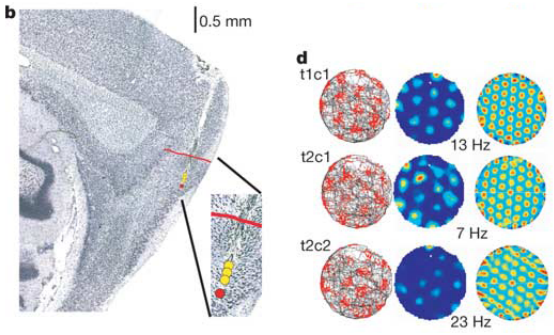


Figura 2.2: En la figura se observan patrones de disparo para células grid en el ambiente de experimentación para las células indicadas en el diagrama de la izquierda. Imagen extraída de [7].

Células de dirección de la cabeza

Varios mamíferos poseen neuronas llamadas células de dirección de la cabeza ("head direction cells") (HD)[30], las cuales están activas sólo cuando la cabeza del animal apunta

en una dirección específica, dentro del entorno en el plano horizontal. Esto significa que una neurona particular puede descargar cuando el animal apunta su cabeza en dirección noreste independiente de su posición. A diferencia de un compás, las celdas HD no dependen del campo magnético de la Tierra, sino que dependen de marcas y señales del movimiento propio.

Células Borde

La mayoría de las neuronas observadas en la corteza entorrinal presentan características de celdas de lugar o celdas grilla, pero un número pequeño de celdas (11 %) presentan un patrón inusual de disparo que no se había observado antes. Estas celdas incrementan su disparo sólo cuando el animal está rodeado de uno o varios muros, independiente del largo del borde o su relación con otros bordes del entorno, de ahí su nombre de células borde[26]. Algunos experimentos realizados muestran que cuando una arena circular con marcas visuales es rotada 90º, el campo de las celdas borde se rota de acuerdo a este cambio. También se notó que al observar simultáneamente dos celdas borde que responden a bordes opuestos, la diferencia en la orientación relativa se mantiene entre distintos escenarios.

Células Spatial View

Estas células presentan un patrón de disparo asociado a la observación de una marca por parte del animal, indiferentemente del lugar que ocupa este en el espacio[22].

SLAM Bioinspirados

Sistemas que pretenden resolver el problema del SLAM tomando como inspiración los mecanismos biológicos existentes. Estos sistemas buscan imitar la versatilidad y robustez que presentan los sistemas neurológicos de animales estudiados como ratas y simios. Sin embargo, la mayoría constan de pruebas de concepto que no son escalables como para operar en entornos reales[29].

Recientemente se ha implementado un sistema de SLAM a gran escala, capaz de confeccionar el mapa de un suburbio entero. Este sistema, denominado RatSLAM toma como inspiración las células grid y spatial view[15]. Si bien este sistema es capaz de operar con éxito a gran escala, siendo capaz de generar un mapa de todo un suburbio, su implementación no corresponde con un sistema biologicamente plausible.

Una de las implementaciones de SLAM bioinspirado es Rat-SLAM. Este sistema consta de una red neuronal "spiking network" que emula un atractor continuo. Este atractor simula un conjunto de células grilla que integran el movimiento según la información del movimiento propio. Este atractor está diseñado de forma de mantener un único núcleo de actividad que representa a ubicación del robot en cada momento.

Para el cálculo del movimiento propio se utilizan dos métodos:

- La utilización de *encoders*.

- La estimación del movimiento propio mediante la comparación de imágenes sucesivas tomadas por la cámara.

Un sistema de visión corrige la posición del núcleo de actividad del atractor mediante vínculos entre patrones visuales y núcleos de actividad aprendidos en el pasado.

En paralelo, Rat-SLAM genera un grafo conteniendo nociones métricas a modo de generar un mapa del escenario navegado. Este grafo es procesado periódicamente para eliminar nodos redundantes y facilitar el cierre de ciclos.

Neurorobótica

Los neurorobots son dispositivos robóticos que tienen sistemas de control basados en los principios del sistema nervioso. Los modelos neurorobóticos proporcionan heurísticas para desarrollar y probar las teorías de la función cerebral en un contexto real. Además, los modelos neurorobóticos podrán servir de base para el desarrollo de robots más eficaces, basados en una mejor comprensión de las bases biológicas de la conducta adaptativa.

Estos sistemas funcionan ejecutando los modelos neurológicos en un robot que opera en un entorno real. El entorno real es necesario por dos razones:

- En primer lugar, simular el entorno puede introducir sesgos no deseados y no intencionales en el modelo.
- En segundo lugar, los entornos reales son ricos, multimodales, y ruidosos, un diseño artificial de un ambiente así sería de cómputo intensivo y difícil de simular. Sin embargo, todas estas características interesantes del entorno están presentes gratuitamente cuando un neurorobot se coloca en el mundo real.

Un neurorobot tiene las siguientes propiedades que lo hacen propicio para los usos antes mencionados:

- Su actividad es una tarea de comportamiento.
- Está situado en un entorno real.
- Tiene capacidades para percibir señales ambientales y actuar sobre su entorno.
- Su comportamiento está controlado por un sistema nervioso simulado cuyo diseño refleja, en cierto nivel, la arquitectura del cerebro y su dinámica.

2.1.2. SLAM Probabilísticos

Los métodos probabilísticos se concentran en encontrar la distribución de probabilidad de la posición del robot y del mapa del entorno a través del tiempo. Para ilustrar este concepto de estimación como una distribución de probabilidad, resulta conveniente analizar la Figura 2.3 propuesta por Thrun[31] . En esta figura se puede observar un robot que

navega en un entorno unidimensional con tres puertas como únicos objetos distinguibles. En la parte inferior se despliega la estimación de la posición del robot como una distribución de probabilidad, en este caso una distribución gaussiana, donde se le asigna a cada intervalo del eje x una probabilidad de que el robot se encuentre en dicho intervalo.

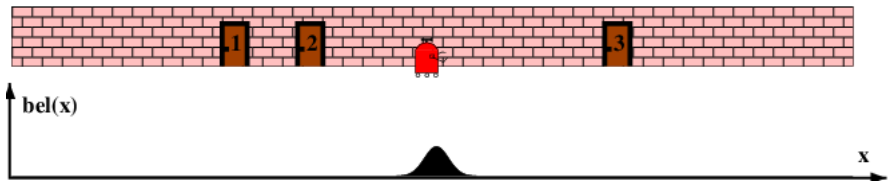


Figure 2.3: Un agente se mueve en un entorno con tres objetos distinguibles, las puertas. La estimación de su ubicación se representa como una distribución de probabilidad.

Luego, se definen tres variables estocásticas involucradas en el modelo de SLAM probabilista:

- x_i modela el valor a estimar en el instante de tiempo i. Para el caso del SLAM, esta variable puede representar la posición del robot, el mapa, o ambos. En ocasiones, esta variable se separa en la posición del robot x_i y el mapa m_i, en otras x_i involucra a ambos conceptos. Esta variable suele recibir el nombre de "estado oculto" debido a que su valor no es directamente observable y, por eso, debe estimarse en función de las restantes variables observables.

- z_i modela la observación realizada en el momento i.Esta observación puede constar de una medida, un conjunto de medidas o una observación de alto nivel, como puede ser "el objeto distinguible 1 está a 3 metros de distancia y a 30 grados".

- u_i modela los controles de movimiento emitidos en el robot al momento i. Algunos ejemplos son la velocidad del robot en un instante dado, la velocidad enviada a cada motor, o la distancia a recorrer. Otra fuente de información que es modelada utilizando esta variable es la información de sensores introspectivos como los encoders de las ruedas o sensores de inercia, que brindan nociones del movimiento propio al robot. Esta variable recibe varios nombres en la literatura, como por ejemplo información de controles o información de odometría.

El tiempo t suele tomarse como un conjunto discreto de instantes, que suelen coincidir con los momentos en los que el robot recibe información. Los subindices en las variables estocásticas x_i, z_i y u_i corresponden al instante de tiempo al que corresponden estas variables.

Tomando todo esto en cuenta, la probabilidad a estimar para solucionar el problema de SLAM puede formularse como:

$$p(x_{1:t}|z_{1:t}, u_{1:t})$$

Es decir, la distribución de probabilidad de estado $x_{1:t}$ que se adapta mejor a las observaciones $z_{1:t}$ y controles $u_{1:t}$.

Alternativamente se puede encontrar en la literatura esta probabilidad representada como

$$bel(x_{1:t}|z_{1:t}, u_{1:t})$$

para hacer énfasis en el hecho de que es la creencia (*belief*) del robot sobre el estado del sistema. Se utilizan en este texto ambas notaciones indiferentemente.

En conclusión, resolver el problema de SLAM implicará estimar la distribución de probabilidad para la variable que representa el estado del sistema x_i en cada instante de tiempo discreto del 1 al t, es decir la variable $x_{1:t}$.

Redes de Bayes

Para realizar la estimación de x_i se modela a esta variable de acuerdo a la red bayesiana incluida en la Figura2.4. En esta red se observa la relación de dependencia entre las variables involucradas en el problema de SLAM, donde una flecha de una variable A a una B indica que la primera incide en la segunda, es decir que $p(B|A)$ y $p(B)$ no son necesariamente iguales (A y B no son independientes). Las variables del estado x_i afectan solamente a la observación correspondiente z_i y al estado siguiente x_{i+1}. Los controles u_i afectan solamente al estado correspondiente x_i. Finalmente, como es coherente, todas las observaciones z_i están influenciadas también por el mapa real del entorno m.

Una conclusión interesante que se desprende de aplicar el teorema de la manta de Markov (Markov Blanket) a este modelo es que dado el estado de una de las variables de estado x_i, todas las variables correspondientes al pasado ($u_{1:i-1}$, $z_{1:i-1}$, $x_{1:i-1}$) son independientes de las variables en el futuro ($u_{i-1:t}$, $z_{i-1:t}$, $x_{i-1:t}$). En particular, implica que observaciones z_i y controles u_i anteriores a este instante no afectan el estado del sistema en instantes posteriores, conocido el estado del sistema x_i. Esto es llamado "Asunción de Markov" (Markov Assumption) en la literatura y puede verse violada por algunos factores en la práctica[31], por ejemplo por componentes no modelados en la dinámica del sistema a modelar x_i. Al existir componentes dinámicos no modelados (p. e. personas) la independencia entre medidas z_i se rompe.

De la figura 2.4 también se infiere otro hecho interesante. Existe un vínculo que relaciona las medidas z_i, a través del tiempo. Este vínculo se realiza a través de la trayectoria del robot, es decir $x_{1:t}$. En la práctica, esto implica que las diferentes partes del mapa sensadas se van correlacionando por medio de la trayectoria del robot. Esta correlación torna computacionalmente más costosa la aplicación del algoritmo, debido a que es necesario actualizar todo el mapa luego de realizar cambios en una localidad. Sin embargo, esta correlación es de hecho lo que permite que la estimación del mapa converja a pesar de la incertidumbre en la trayectoria. Esto se debe a que la correlación hace que la incertidumbre sobre la distancia

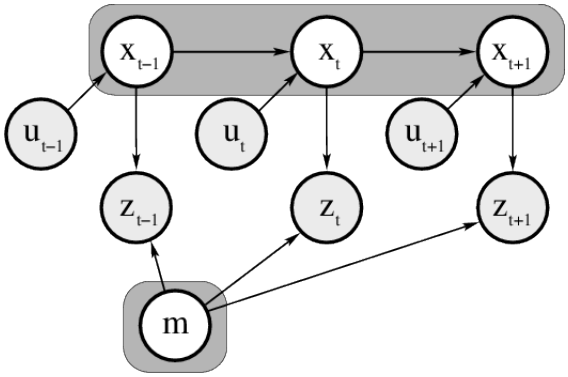


Figura 2.4: La red bayesiana que modela el problema de estimación probabilística del SLAM. En gris se muestran las variables ocultas a estimar.La figura ha sido extraída de [24].

relativa entre partes del mapa disminuya[2].

2.2. Representación del mapa

Un algoritmo de SLAM puede llevar una representación o mapa de su ambiente de diversas maneras. Cada uno de estos tipos de mapas tienen sus ventajas y desventajas asociadas. Al momento de elegir un tipo de mapa a implementar, se deben tener en cuenta las siguientes preguntas:

- ¿Es necesario mantener en el mapa nociones métricas?
- ¿Cuál es el uso que se le dará al mapa?
- ¿El entorno es dinámico?
- ¿De qué sensores se dispone?
- ¿De cuánto poder de cómputo se dispone?

A continuación se incluye una descripción de los tipos de mapas presentes en los sistemas relevados.

2.2.1. Grillas de ocupación

Las grillas de ocupación (*occupancy grids*) representan el espacio como una grilla que divide el entorno en un conjunto de celdas, donde cada celda tiene asociado un valor que representa su ocupación. Este valor puede ser absoluto (libre u ocupado) pero en la práctica suele representar una noción de probabilidad de que la celda se encuentre ocupada[19].

Estos mapas son usualmente utilizados en sistemas con sensores de distancia como sonares o sensores laser[3, 5, 8, 27].

A diferencia de otras representaciones, la salida generada por un algoritmo que utiliza grillas de ocupación es muy similar a los planos utilizados por los seres humanos. Esto convierte a esta representación en idónea cuando lo que se quiere es obtener un mapa legible de un entorno, ver Figura 2.5.

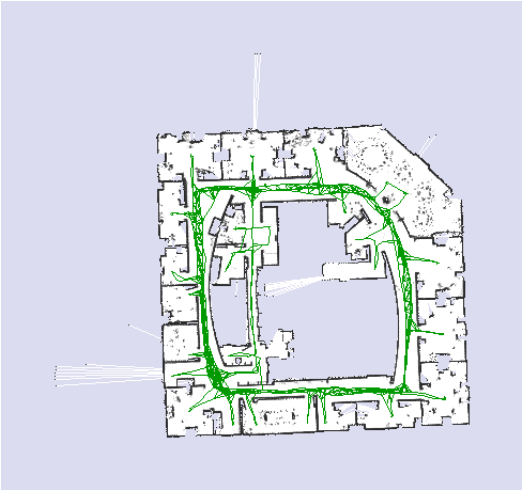


Figura 2.5: Mapa generado utilizando grillas de ocupación. Extraída de http://kaspar.informatik.uni-freiburg.de/.

Además, la naturaleza métrica de esta representación la hace adecuada para tareas de planificación de caminos (*path planning*).

Debido a que la creencia sobre la ocupación de cada celda se representa con una probabilidad, el modelo permite manejar la incertidumbre naturalmente. Esto permite incluir manejo de:

- Ruido causado por los sensores.
- Ruido causado por dinamismos no modelados, como personas moviéndose.
- Incertidumbre de la posición del robot que afecta el conocimiento sobre el mapa relativo al robot.

En general estas grillas son utilizadas para representar entornos en un plano de dos dimensiones. Sin embargo, se han realizado trabajos que extienden esta representación a tres dimensiones.

La precisión de la grilla de ocupación determina la calidad del mapa obtenido y con esta se disminuyen los errores producidos por la discretización de un ambiente continuo. Sin embargo, al aumentar el número de celdas el costo computacional de actualización del mapa aumenta, al igual que la memoria necesaria para almacenar el mapa. Al utilizar grillas de ocupación, es necesario encontrar un punto de equilibrio entre la precisión del mapa y el poder de cómputo y espacio de memoria disponible.

2.2.2. Conjunto de marcas

Otra representación ampliamente utilizada en los sistemas relevados es la de un mapa como un conjunto de marcas[10, 6, 1, 12, 9, 28, 4].

Una marca es una o más características perceptualmente distinguibles de interes, ubicadas en un objeto o un lugar de interés [20].

La mayoría de los sistemas basados en marcas usan una cámara como sensor principal. Sin embargo pueden distinguirse marcas utilizando otras fuentes de datos. Por ejemplo con múltiples medidas de un sonar o un sensor laser es posible identificar esquinas, puertas, columnas y ventanas que pueden servir como marcas.

Un mapa de marcas consta de un conjunto de coordenadas que indican la posición de cada marca en el entorno de trabajo.

Los mapas de marcas no contienen información detallada o volumétrica del entorno. Esto hace que la representación sea compacta. Por otro lado, debido a esta falta de información, no es posible realizar tareas de planificación de caminos utilizando este tipo de mapas.

Las marcas pueden ser perfectamente identificables, es decir que es posible saber de cuál marca se trata con tan solo observala. También existen sistemas donde esta correspondencia no puede ser determinada con seguridad. En los casos en los que las marcas no son perfectamente distinguibles, el algoritmo que utilice este mapa deberá tomar en cuenta esta incertidumbre.

En el contexto del SLAM, los mapas de marcas suelen representarse, en vez de con un conjunto de coordenadas, como un conjunto de gaussianas que representan la distribución de probabilidad de la ubicación de cada marca. De esta forma es posible representar la incertidumbre en la posición de las marcas, que puede ser actualizada a medida que se navega por el entorno. En este caso, para cada marca se almacenan los dos parámetros de la gaussiana correspondiente, la mediaμ y la varianza σ^2. Esto puede representarse gráficamente, para el caso de un entorno de trabajo de dos dimensiones, como un conjunto de óvalos, donde el centro indica la media de la gaussiana y los ejes los vectores propios de su matriz de covarianza, ver figura 2.6.

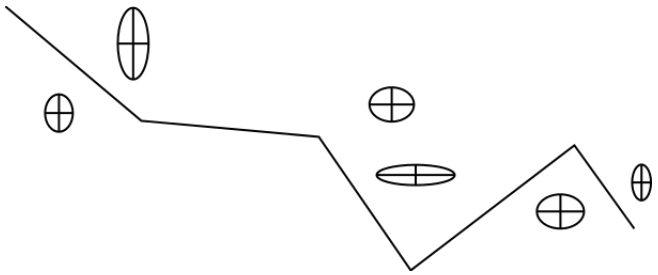


Figura 2.6: Representación gráfica de un conjunto de marcas. Cada marca es representada por una elipse. La linea marca la trayectoria del robot.

2.2.3. Mapas topológicos

Los mapas topológicos constan de una relación de conectividad entre lugares distinguibles. Su representación consta de un grafo donde los nodos son lugares distinguibles mediante sensado directo y las aristas indican representan la navegabilidad entre dos de estos lugares.

Estos mapas son propicios para sistemas de navegación asociativos según descritos en [20].

Al igual que los mapas de marcas, estos mapas son compactos en su representación debido a su falta de información métrica detallada del entorno.

2.3. Sensado

Cuando un robot sensa el ambiente, recibe información del entorno. Tanto la actualización del mapa como la localización del robot requieren relacionar la información sensada con el mapa actual. Para esto, se definen distribuciones de probabilidad de lo sensado en función de cierta configuración del entorno, o cierto mapa, y la posición del robot. Estas distribuciones de probabilidad llevan el nombre de modelo de sensado y pueden escribirse como,

$$p(z_i|x_i, m)$$

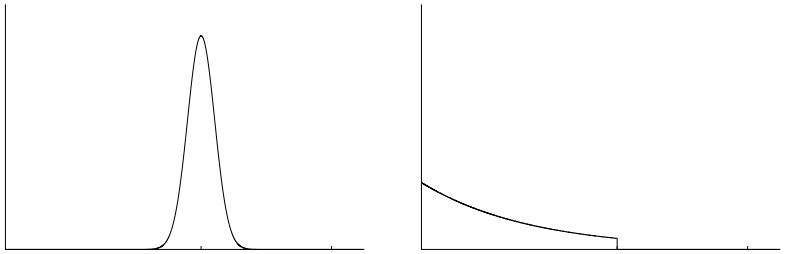
La representación de estos modelos como distribuciones de probabilidad permite modelar las fuentes de ruido que hacen que el sensor no sea un instrumento completamente fiable. De esta manera, los posibles errores cometidos por el sensor son modelados naturalmente en la solución del SLAM, tomándolos como parte de la incertidumbre a manejar en el proceso de estimación del sistema x_i.

Naturalmente, el modelo de sensado utilizado dependerá del sensor y representación del mapa utilizado. A continuación se incluye una breve descripción de cada uno de los modelos de sensado utilizado en los sistemas relevados.

2.3.1. "Beam Endpoint Model"

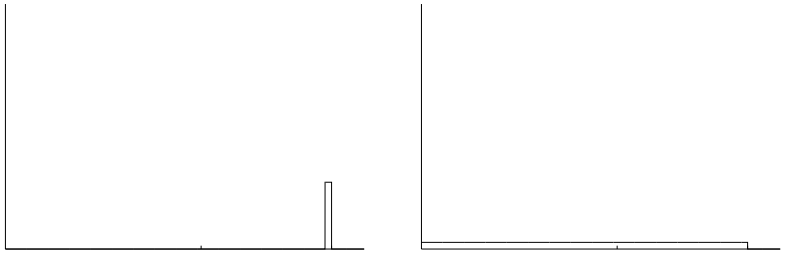
Este modelo de sensado aplica a soluciones de SLAM con representaciones métricas (ver Representación del mapa) y sensores de barrido láser o *laser range finder*s. Este modelo toma cada una de las medidas realizadas por un barrido del sensor individualmente, donde cada medida se representa z^i. Suponiendo que la medida z^i sensa un objeto o, se considera la distancia a o y se agregan cuatro componentes que modelan las diferentes potenciales fuentes de ruido:

- Un pequeño ruido blanco que puede interferir en la medida realizada, modificándola relativamente poco en torno al valor real.
- La posibilidad de sensar otro elemento no modelado, por ejemplo una persona en movimiento, ubicado entre el robot y el objeto o. En este caso se obtendría una distancia menor. La probabilidad de sensar algo entre el robot y o decrementa a medida que nos acercamos a este y se modela dicha probabilidad con una exponencial decreciente.
- La posibilidad de no obtener el reflejo del laser. En este caso se obtendría la distancia máxima del sensor.
- Error al azar uniformemente distribuido. Esta componente modela algunos errores que los sensores suelen cometer y que no tienen una explicación conocida.



(a) Pequeño error en la distancia sensada dado que se sensó el objeto.

(b) Error ocurrido por la presencia de objetos no modelados.



(c) Error producido por no sensar el objeto.

(d) Error uniforme no modelado.

Figura 2.7: Los cuatro factores de error del modelo "Beam Endopoint Model". Imagen extraída de [31].

Luego, para calcular la distribución de probabilidad, solo es necesario encontrar en el mapa el objeto más próximo a interceptar por el laser utilizando la ubicación estimada y el ángulo del laser. Con la distancia a este objeto, se calcula cada uno de los componentes de la distribución y se suman para obtener la distribución objetivo, ver figura 2.8.

Este modelo es extensivamente utilizado por su simpleza y es utilizado por dos de los sistemas relevados[5, 8].

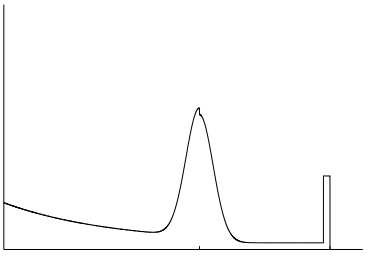


Figura 2.8: La distribución de probabilidad de sensado completa. Imagen extraída de [31]

Existen también otros modelos más simples en los que se confía en la medida del sensor y, manteniendo una grilla de ocupación probabilistica (ver sección 2.2), se asigna la probabilidad del sensado igual a la probabilidad de que el punto correspondiente en el mapa esté efectivamente ocupado[27].

2.3.2. "Pinhole Model"

Este modelo es utilizado por las soluciones de SLAM que utilizan cámaras para la extracción de marcas (ver sección 2.2). Este modelo considera a la cámara como una caja con un orificio diminuto por donde entra la luz que es interpretada como una imagen, ver figura 2.9.

El modelo Pinhole permite relacionar parámetros de la imagen proyectada y parámetros intrínsecos de la cámara con las medidas y posición de los objetos observados. Por ejemplo, realizando un análisis geométrico basado en triángulos homomórficos es posible derivar la relación:

$$d/D = f/Z$$

donde d corresponde al tamaño del objeto en la imagen, D al tamaño real, f a la distancia entre el orificio y el plano de proyección (este parámetro es conocido como "focal lenght") y Z es la distancia al objeto, ver figura 2.9.

También permite inferir el ángulo relativo de una marca con respecto al robot, aplicando derivaciones similares, basandose en la hipótesis de que la cámara consta de un orificio pequeño y un plano de proyección.

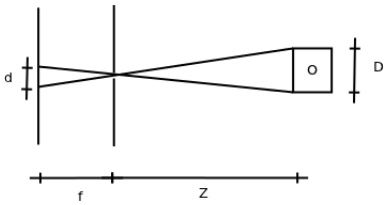


Figura 2.9: Modelo de la cámara pinhole con los parámetros f, Z, d y D asociados.

Este modelo es utilizado ampliamente por su simplicidad, aún siendo un modelo básico.

Algunas implementaciones agregan a este modelo la noción de distorsión causada por los lentes[11].

2.4. Movimiento

Para mantener un estimado de la ubicación del robot es importante mantener una noción del movimiento propio del robot. El movimiento del robot no es un proceso determinista, si las ruedas resbalan sobre el suelo el robot se desplazará menos de lo planificado. Por esto, el modelo del movimiento propio del robot se realiza, al igual que en los modelos de sensado, con distribuciones de probabilidad. Se define el modelo de movimiento o modelo de transición como:

$$p(x_t|x_{t-1}, u_t)$$

Es decir, la probabilidad de que el sistema se encuentre en determinado estado x_t, dado que se encontraba en el estado x_{t-1} y aplica los controles u_t.

Este modelo de transición también modelara como cambia el mapa a través del tiempo. En este modelado pueden incluirse dinamismos propios del sistema, inherentes a la voluntad del robot. Sin embargo, la mayoría de las implementaciones relevadas consideran el ambiente estático y su modelo de transición se remite a la porción de x_t relacionada con la posición del robot y deja estática la porción referente al mapa.

A continuación se incluye una descripción de los dos modelos de transición más utilizados.

2.4.1. Modelo de velocidad constante

Este modelo considera que el robot se mueve a una velocidad lineal y angular constante. La idea principal de este modelo es utilizar directamente los controles u_t enviados a los motores para calcular el movimiento realizado. Estas velocidades se representan con el vector

$$(v, w)$$

Es posible deducir que este movimiento de velocidad lineal y angular constante, corresponderá a un movimiento circular de radio v/w.

Para representar las estocasticidad del movimiento real del robot, se le agregan a este modelo dos fuentes de ruido:

- Un ruido$\hat{v}$ de la velocidad lineal.
- Un ruido $\hat{w}$ de la velocidad angular.

De modo que la velocidad modelada será:

$$(v + \hat{v}, w + \hat{w})$$

Adicionalmente, se agrega una fuente de ruido asociada a un movimiento de rotación γ llevado a cabo al final del movimiento. Esta fuente adicional de ruido es necesaria debido a que el movimiento circular descrito por el modelo rara vez se cumple en la realidad, ya que la hipótesis de velocidad constante tampoco suele cumplirse. El radio y la distancia recorrida del círculo pueden ser corregidas por los parámetros de ruido $\hat{v}$ y $\hat{w}$, pero el hecho de que la trayectoria puede no ser circular no puede ser corregido por estos parámetros. Esto implica que la posición final del robot puede ser corregida por los parámetros de ruido, pero no así la orientación final del robot. Por lo tanto, es necesario agregar este factor de ruido de rotación final para tomar en cuenta la posibilidad de que el movimiento no sea estrictamente circular, y la orientación final no se corresponda con la esperada al final de un movimiento de trayectoria circular.

2.4.2. Modelo de odometría

Este modelo plantea tomar los datos de odometría para calcular el movimiento inmediatamente pasado del robot. Los datos de odometría suelen obtenerse de la integración de la información de los encoders del robot, sin embargo es posible obtenerla de otras fuentes como

- Análisis de imágenes consecutivas para detectar cambios que permitan estimar el movimiento propio[15].
- Técnicas de scan-matching que permiten estimar el movimiento propio respecto a un mapa local[5, 18].
- Integración de datos provenientes de sensores de inercia.

Este modelo descompone el movimiento realizado en tres componentes (ver figura 2.10):

- Una rotación inicial de ángulo δ_{rot1}.
- Una traslación de largo δ_{trans}.
- Una rotación final de ángulo δ_{rot2}.

Al igual que el modelo de velocidad constante, para representar las estocasticidad del movimiento real del robot, se agrega una fuente de ruido por cada uno de los submovimientos.

2.5. Procesamiento de la información

El procesamiento de la información es el proceso central en un algoritmo de SLAM. Este procesamiento es el que convierte la información sensorial y de odometría entrante en un estimado de la posición del robot y el mapa del entorno.

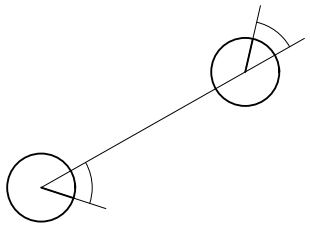


Figura 2.10: La descomposición del movimiento en tres. Imagen extraída de [31].

2.5.1. Taxonomías del procesamiento de información de SLAM

Existen diversas taxonomías para clasificar a este módulo de SLAM, a continuación se detallan algunas de ellas.

Full vs. Online

Existen dos problemas diferentes a resolver en SLAM. El primero consiste en estimar la trayectoria y mapa más verosímil dado un conjunto de observaciónes $z_{1:t}$ y un conjunto de controles $u_{1:t}$. Este problema es conocido como Full SLAM y puede formularse como:

$$p(x_{1:t}|z_{1:t}, u_{1:t}) \tag{2.1}$$

Otro planteo del SLAM probabilista propone procesar la información entrante una a una, hallando la posición más verosimil en cada instante. Este problema es conocido como Online SLAM y puede formularse como:

$$p(x_t|z_t, u_t, x_{t-1}) \tag{2.2}$$

Debido a la similitud del problema de Online SLAM con el problema de filtros de información[1], algunas soluciones al problema de SLAM toman el nombre de filtros (p.e. filtros de Kalman, filtros de partículas).

Al observar las ecuaciones 2.1 y 2.2 puede verse que la diferencia entre ellas radica en que Full SLAM toma en cuenta toda la información disponible al momento, mientras que

[1]De la literatura del procesamiento de señales con ruido es que viene el término de filtro. Estas herramientas intentan solucionar el problema de estimar una señal original x(t) cuando solo es posible observar $y(t) = x(t) + r(t)$
donde r(t) es una señal de ruido no conocida. Muchas soluciones a este problema buscan encontrar el x*(t) óptimo tal de maximizar la probabilidad
$p(x(t)|y(0) \dots y(h))$

Online SLAM calcula los cambios en el estado oculto x_t de a un paso, tomando solo la última información disponible. En términos matemáticos, esto puede expresarse como que la diferencia radica en que Online SLAM realiza una marginalización[2] de toda la información anterior de sensado en el estado oculto del sistema al momento t.

El problema de Full SLAM suele consumir más recursos. Esta idea se ampliará más adelante utilizando los conceptos presentados en la subsección Grafos.

Por otro lado, el problema de Online SLAM es más complejo debido a que debe lograrse una buena marginalización de la información.

El problema de Online SLAM goza de mayor popularidad debido a que puede ser ejecutado dentro de un robot operando en tiempo real. Sin embargo, la división entre Full y Online SLAM no es siempre tan nítida y existen sistemas cuya clasificación no es tan clara[12].

Paramétrico vs. No Paramétrico

Cuando se estima la distribución de probabilidad del estado oculto se busca encontrar una función de densidad de probabilidad, usualemente sobre un espacio de tres o más dimensiones. El mantenimiento de esta función crece con la precisión con la que se desea modelarla. Por lo tanto, es necesario realizar simplificaciones al modelo, para que sea computacionalmente factible.

La simplificación más fuerte que suele hacerse es utilizar distribuciones de probabilidad conocidas que dependan de un conjunto pequeño de parámetros. El ejemplo más común es la utilización de una distribución gaussiana que puede representarse utilizando la media μ y la varianza σ^2, como se verá más adelante en la subsección Filtros de Kalman. Al utilizar estas distribuciones paramétricas, el algoritmo solo debe trabajar con este conjunto reducido de parámetros para la actualización de su estimado a medida que llega nueva información.

La gran desventaja que presenta la utilización de distribuciones paramétricas para representar el estado del sistema radica en que solo es posible representar un conjunto reducido de funciones de densidad posibles. Por ejemplo, en el caso de las distribuciones gaussianas, se observa que estas distribuciones son unimodales, es decir, que tienen un solo máximo local. En la práctica, esto implica que el estimador solo puede representar una única creencia sobre el estado del sistema.

Otros sistemas, como los filtros de partículas (ver subsección Partículas), son capaces de representar cualquier función de densidad de probabilidad y, por lo tanto, mantener varios máximos locales en su distribución de probabilidad. Esto equivale a mantener varios estimados completamente diferente del estado del sistema de forma concurrente. En desventaja, la actualización de estas distribuciones a partir de la información entrante suele ser más costosa.

[2] Aplicado a una distribución de dos variables, el término marginalización refiere a descartar la información de una variable Y en una distribución conjunta $p(X, Y)$, usualmente integrando toda la información de aportada por Y para llegar a una distribución de una variable $p(X)$.

Existen sistemas que combinan ambos tipos de distribución, utilizando distribuciones unimodales para algunas variables y multimodales para otras[18].

2.5.2. Filtros de Kalman

Los filtros de Kalman[13] son una solución al problema de Online SLAM que propone actualizar el estado del sistema a medida que se obtiene información de odometría y sensado.

La principal característica de esta solución es el uso de distribuciones gaussianas como estimador. Esta solución se basa en el hecho que al multiplicar una variable de distribución gaussiana por un número (o matríz en el caso multivariable) se obtiene otra variable de distribución gaussiana. Luego, el algoritmo propone que la evolución del sistema puede modelarse mediante dos ecuaciones lineales.

Modelado

El modelado de la evolución del sistema se realiza mediante dos ecuaciones, la ecuación de transición y la ecuación de observación. A continuación se describen ambas ecuaciones.

Ecuación de transición La primera ecuación actualiza el estado del sistema estimado en función de una evolución independiente de factores externos y en función de la información de odometría:

$$x_t = F.x_{t-1} + B.u_t + r_t \tag{2.3}$$

donde:

- x_t refiere al estado del sistema a estimar
- F consta de una matriz que representa el cambio inherente al sistema, ajeno a factores externos. Este cambio podría representar, por ejemplo, la evolución de la posición de un objeto que viaja a velocidad constante.
- u_t representa la información de odometría.
- B consta de una matriz que representa la transformación de la información de odometría en los cambios producidos en el sistema. Esta matriz representa, de hecho, el modelo de transición del sistema explicado en sección 2.4.
- r_t es un factor de ruido que representa la naturaleza estocástica de los otros términos de la ecuación. Este ruido es también gaussiano.

Ecuación de observación La segunda ecuación propone una relación entre el estado estimado del sistema y la observación a realizar por el robot en un tiempo t. La ecuación puede expresarse como:

$$z_t = H.x_t + w_t \tag{2.4}$$

donde:

- z_t corresponde a la información de sensado.
- H consta de una matriz que transforma el estado del sistema en la observación a realizar. Por ejemplo esta matriz podría transformar las coordenadas de una marca en las coordenadas del campo visual del robot (proyección). Esta matriz representa, de hecho, el modelo de sensado del sistema explicado en sección 2.3.
- w_t representa el ruido en el otro término de la ecuación. Este término podría corresponderse con el ruido introducido por los sensores del robot, que no son totalmente precisos.

Cálculo del estado del sistema Bajo las condiciones mencionadas, la solución que minimiza el error esperado (mínimos cuadrados) del estado del sistema puede calcularse de forma cerrada utilizando un filtro de Kalman[32]. Esta actualización se realiza mediante un algoritmo que realiza solo operaciones de álgebra lineal para actualizar los estimadores de la posición del sistema (media μ y varianza σ^2). No se incluye en este trabajo una explicación detallada del algoritmo de estimación, para más información puede consultarse [32, 31].

Linealidad El algoritmo de Kalman propone la evolución de un sistema, modelado por distribuciones gaussianas, regido por dos ecuaciones lineales. La linealidad de estas ecuaciones representa una restricción importante. Supongamos que nuestras observaciones constan del sensado de la distancia a una determinada marca. La función que toma como entrada las coordenadas del robot y la marca y retorna la distancia no es una función lineal (norma euclideana). Por lo tanto, al utilizar este sistema se cometerían sistematicamente errores por utilizar una función lineal (la más aproximada posible) para la representación de la función de distancia.

Además, la utilización de una distribución gaussiana como estimador del estado del sistema implica que solo es posible mantener un único máximo local en la creencia del estado de este sistema.

Extensiones al algoritmo original

Para eliminar la restricción de modelos de sensado y transición lineales, se diseñaron extensiones al filtro original. En esta variante, conocida como Filtro de Kalman Extendido,

se reemplazan los modelos de sensado y odometría para contemplar funciones no lineales. Las ecuaciones 2.3 y 2.4 quedan:

$$x_t = f(x_{t-1}, u_t) + r_t$$

$$z_t = h(x_t) + w_t$$

donde f y h son los modelos de transición y sensado.

Esta modificación permite la utilización de filtros de Kalman con modelos no lineales. Se logra utilizando un algoritmo similar al original, pero linealizando los modelos de datos utilizando series de taylor en cada iteración.

Sin embargo, el nuevo algoritmo no corresponde a la solución cerrada del problema, es decir que la solución no es exacta y el algoritmo puede diverger por problemas vinculados a la linealización de los modelos.

2.5.3. Partículas

Los filtros de particulas intentan aproximar la distribución de probabilidad del estado del sistema x_t utilizando métodos de Montecarlo. Para esto, mantienen un conjunto de partículas que son N muestras (*samples*) de la distribución a actualizar.

Al igual que el filtro de Kalman, actualizan esta distribución a medida que se encuentra disponible nueva información de sensado u odometría.

Actualización de la distribucíon de probabilidad

La dinámica de actualización de la función de densidad del filtro de partículas es similar al del filtro de Kalman (ver subsección Filtros de Kalman). Esta actualización consta de dos pasos, el de predicción y el de actualización propiamente dicho.

Paso de predicción El paso de predicción busca modificar la función de densidad para reflejar un movimiento realizado por el robot. Para esto, se modifica la posición de cada una de las partículas según:

$$x_t = f(x_{t-1}, u_t)$$

donde f representa el modelo de movimiento a utilizar.

En la figura 2.11 se puede ver un conjunto de partículas incial (abajo a la izquierda) al que se le aplican sucesivamente los pasos de predicción.

Esta imagen ilustra como este paso del algoritmo aumenta la dispersión de las partículas, y por consiguiente la varianza de la distribución. Esto se debe a que el modelo de sensado incluye ruido para modelar la naturaleza estocástica del movimiento del robot. Este ruido se va acumulando en cada paso de predicción y aumenta la incertidumbre de la estimación.

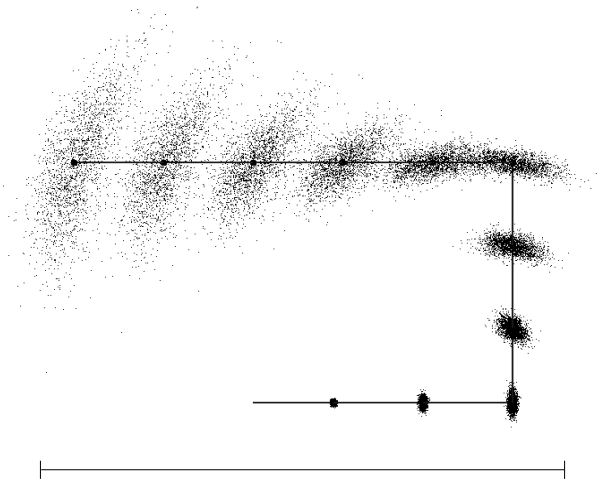


Figura 2.11: Paso de predicción aplicado sucesivas veces. Imagen extraída de [31]

Paso de actualización Este paso busca ajustar la función de densidad a la última información de sensado recibida. Para esto se le asigna a cada partícula un peso w_i proporcional a la verosimilitud de la información de sensado acorde al estado actual del sistema. Es decir

$$w_i \sim p(z_t|x_t)$$

Este peso indica, en resumen, cuán verosímil es el modelo representado por la partícula en función de la última información de sensado.

Luego, se realiza un proceso denominado remuestreo ("resampling") en el que se extraen con reposición N partículas del conjunto de partículas actuales. La probabilidad de seleccionar una partícula es proporcional a su peso w_i.

En general, el paso de actualización disminuye la incertidumbre de la estimación. En otras palabras, disminuye la dispersión de las partículas.

2.5.4. Grafos

A diferencia de las soluciones de SLAM basados en filtros, el SLAM de Grafos (*Graph SLAM*) resuelve el problema de Full SLAM.

Graph SLAM es utilizado para la resolución del SLAM basado en marcas. Para esto, la solución modela el problema de SLAM como un grafo, donde los nodos representan posiciones del robot x_i y marcas m^i (el superíndice es utilizado para no confundir con el subíndice que indica instante en el tiempo). Los datos obtenidos de odometría y sensado se procesan para transformarlos en aristas que relacionan estos nodos. Se incluye la figura 2.12 para ilustrar este concepto.

Esta formulación puede verse como un problema de optimización, donde se debe opti-

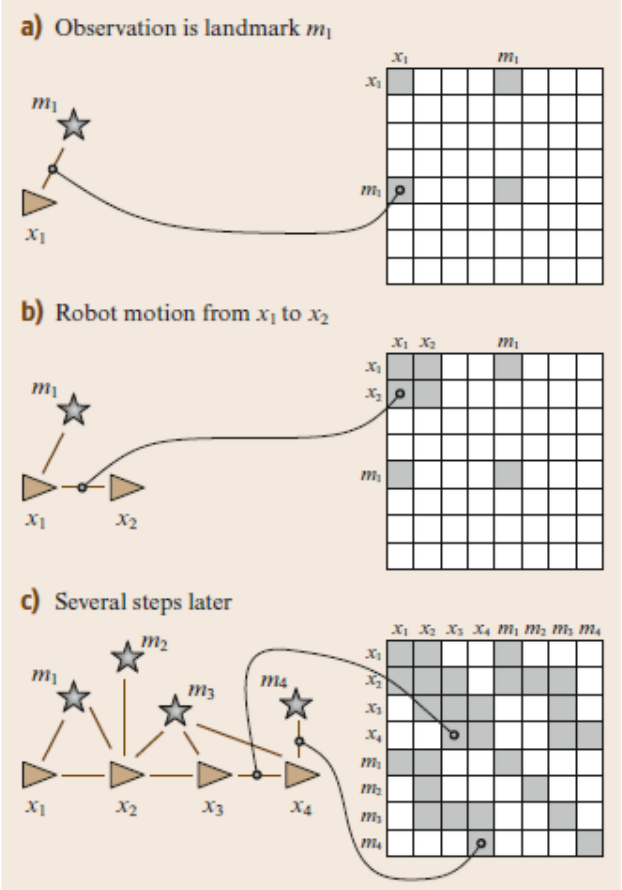


Figura 2.12: Modelo del grafo de Graph SLAM. La matriz de la izquerda corresponde a la matriz de conectividad del grafo. Figura extraída de [24].

mizar la posición de las marcas (mapa) y las posiciones sucesivas del robot (trayectoria) respetando lo mejor posible un conjunto de restricciones suaves[3] representadas por las aristas. Estas restricciones se expresan en distancias, inferidas de las observaciones de las marcas y de los datos de odometría.

Este modelo se compara con el modelo de masas y resortes (*spring-mass model*), donde un conjunto de objetos con masa se encuentran interconectados entré si por medio de resortes (ver figura 2.13). Este sistema converge a la distribución de posiciones de los objetos para minimizar la energía total contenida en cada resorte.

Es importante notar que Graph SLAM pospone los cálculos (*lazy algorithm*) de modo de procesar una gran cantidad de datos (o todos), a la vez. Para esto, se divide la solución en dos partes o módulos, el *front-end* y el *back-end*. El front-end se encarga de construir el grafo de restricciones a partir de la información de sensado y odometría. El back-end

[3]Las restricciones suaves (*soft constraints*) son restricciones que no afectan la validez de una solución, si no que afectan su costo.

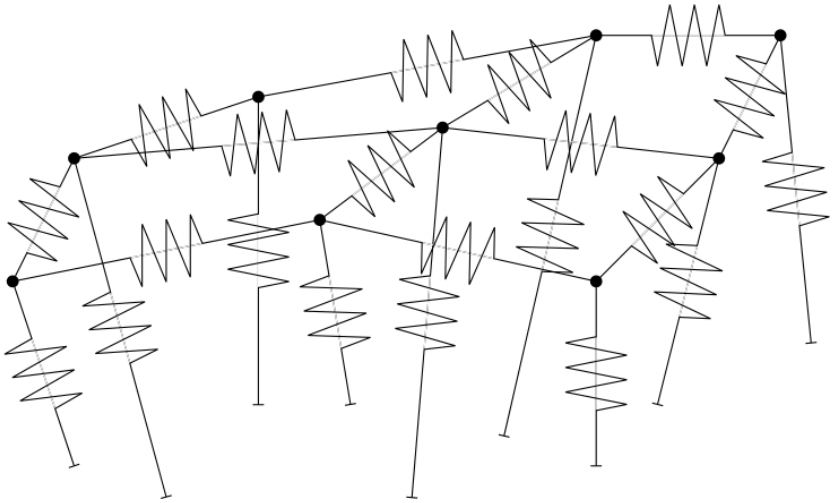


Figura 2.13: Modelo de masas y resortes. Imagen extraída del sitio Web Machine Vision and Perception Group.

realiza la optimización de este grafo de modo de obtener la solución más verosímil a los datos disponibles.

La resolución del problema por parte del back-end resulta computacionalmente costosa, cuando se lo compara con un algoritmo de Online SLAM. Esto se debe a que el algorítmo debe contemplar un mayor volumen de información, debido a que todas las conexiones derivadas de las observaciones deben ser tomadas en cuenta. En Online SLAM, en cambio, la información pasada es marginalizada en el estimado de la posición actual x_i.

2.5.5. Otros algoritmos

Existen otros algoritmos de procesamiento de la información que omitimos en este documento. Entre ellos, se puede destacar Maximización de Esperanzas (*Expectation Maximization*)[14].

Capítulo 3

Casos de estudio

3.1. FastSLAM

Fast SLAM [17, 18] es un algoritmo de SLAM basado en filtro de partículas sobre un mapa de marcas, desarrollado por Michael Montemerlo, Sebastian Thrun, Daphne Koller y Ben Wegbreit.

A continuación se detallan algunas de las particularidades de FastSLAM.

3.1.1. Partículas y Kalman

La principal característica de FastSLAM es su combinación de un filtro de partículas con un conjunto de filtros de Kalman.

Los autores se basan en el hecho que, dada la trayectoria de un robot, la posiciones de cada marca se tornan independientes entre si. Este hecho se desprende de la figura 2.4.

Basándose en esto, FastSLAM se implementa como un filtro de partículas que busca encontrar la trayectoria del robot $p(x_{1:t}|u_t, z_t)$ sin importar el mapa m. Luego, cada partícula toma como verdadera su estimación de la trayectoria y, por lo tanto, realiza una estimación independiente para cada una de las marcas del mapa.

Como resultado, cada partícula posee K filtros de Kalman de baja dimensión (igual a la del entorno de trabajo), donde K es el número de marcas del sistema.

Tomando esto en cuenta, es posible observar que una implementación sencilla de este algoritmo requiere $O(MK)$ de tiempo computacional, siendo M el número de partículas y K la cantidad de marcas. Para disminuir el tiempo computacional se implementó una estructura de datos arborecente que reduce los cálculos a $O(MlogK)$. Esto se cumple siempre y cuando la correspondencia sea conocida.

Una diferencia importante entre el modelo tradicional y FastSLAM es que utiliza gaussianas de dos dimensiones, obteniendo un orden de cómputo constante (no creciente con la

cantidad de marcas), a diferencia del tradicional, que es de orden lineal en la cantidad de marcas.

3.1.2. Optimización de la distribución propuesta

FastSLAM 2.0 implementa una mejora vinculada al paso de predicción, también conocido como paso de generación de la distribución propuesta. Esta mejora fue también encontrada en otro sistema relevado[5].

La optimización consta de utilizar la información de sensado como parte del modelo de desplazamiento, es decir que el modelo de desplazamiento queda:

$$p(x_t|x_{t-1}, u_t, z_t, m)$$

Esto implica que las partículas generadas en el paso de predicción son más verosímiles a la observación realizada. En la práctica, esto significa que menos partículas son necesarias para representar correctamente la distribución luego de incluída la información de odometría.

Es importante destacar que, si bien cada partícula corresponde a una trayectoria del robot, para generar el nuevo conjunto de partículas solo se utiliza la posición del robot más reciente x_{t-1}.

3.1.3. Inclusión de marcas nuevas

Para la inclusión de marcas nuevas detectadas se utiliza un algoritmo común en el área de SLAM sobre marcas. Este algoritmo consta en mantener un índice que relaciona:

- La cantidad de veces que se observó la marca.
- La cantidad de veces que se debería haber observado una marca (está se encontraba dentro del campo visual) y no sucedió.

Este índice se mantiene para todas las marcas y se consideran activas aquellas en las que este supera cierto umbral arbitrario.

La utilización de este índice permite al sistema incorporar nuevas marcas con una cierta seguridad de su existencia y eliminar marcas que dejan de ser observables. En definitiva, este manejo de las marcas permite contemplar cierto dinamismo en el entorno de trabajo.

3.1.4. Capacidad de escalar

Debido a que las marcas son utilizadas de manera independiente y el tiempo de ejecución de la operación más costosa (paso de actualización) se realiza en tiempo $O(M.log(K))$, este sistema escala a un número mayor de marcas que los filtros de Kalman, los cuales tienen un tiempo de actualización de $O(K^2)$.

3.1.5. Representación del mapa

Como se vió, cada partícula mantiene la estimación de las K marcas, la función de copia requiere tiempo de $O(MK)$. Sin embargo, la mayoría de las copias pueden ser evitadas.

La idea principal es que el conjunto de gaussianas en cada partícula es representada por un árbol binario de búsqueda balanceado, y que en el proceso de generación de una nueva partícula sólo se referencia a el árbol de la partícula padre incluyendo algunas modificaciones realizadas a las marcas. Solo se actualiza la estimación de la marca necesaria, el resto de las estimaciones permanecen intactas.

3.1.6. Asociación de datos

En FastSLAM se estima la asociación (identificación de la marca observada) para cada partícula. Incluso cada partícula podría poseer un número diferente de marcas en sus respectivos mapas. Las partículas que tengan errores en la asociación tienden a desaparecer, a diferencia del enfoque por EKF en el cual esto genera una falla irrecuperable.

Como consecuencia negativa, se observa que el sistema de partículas realiza una búsqueda en el espacio conjunto de las trayectorias posibles del robot y las asociaciones posibles de datos. Esto torna más grande el espacio de búsqueda, y por lo tanto aumenta el número necesario de partículas.

Bibliografía

[1] Javier Civera and Andrew J Davison. 1-point ransac for ekf filtering. application to real-time structure from motion and visual odometry. *Journal of Field Robotics*, 27(5):609–631, 2010.

[2] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localisation and mapping (slam): Part i the essential algorithms. *IEEE ROBOTICS AND AUTOMATION MAGAZINE*, 2:2006, 2006.

[3] A. Eliazar and R. Parr. DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks, 2003.

[4] Udo Frese. Closing a million-landmarks loop. In *In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing. submitted*, pages 5032–5039, 2006.

[5] G. Grisetti, C. Stachniss, and W. Burgard. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. *Robotics, IEEE Transactions on*, 23(1):34–46, February 2007.

[6] Jose Guivant and Eduardo Nebot. Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transactions on Robotics and Automation*, 17:242–257, 2001.

[7] Torkel Hafting, Marianne Fyhn, Sturla Molden, May-Britt Moser, and Edvard I. Moser. Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436(7052):801–806, June 2005.

[8] D. Hahnel, W. Burgard, D. Fox, and S. Thrun. An efficient fastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, 1:206–211 vol.1, October 2003.

[9] Christoph Hertzberg. A framework for sparse, non-linear least squares problems on manifolds, 2008.

[10] Shoudong Huang, Zhan Wang, Gamini Dissanayake, and Udo Frese. Iterated slsjf: A sparse local submap joining algorithm with improved consistency.

[11] Andrew J. Davison Javier Civera, O. Garcia and J. M. M. Montiel. *Journal of Field Robotics*, 27(5):609–631, October 2010.

[12] M. Kaess, A. Ranganathan, and F. Dellaert. isam: Incremental smoothing and mapping. *Robotics, IEEE Transactions on*, 24(6):1365 –1378, dec. 2008.

[13] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[14] S. Le Corff, G. Fort, and E. Moulines. Online expectation maximization algorithm to solve the slam problem. In *Statistical Signal Processing Workshop (SSP), 2011 IEEE*, pages 225 –228, june 2011.

[15] M J Milford and G F Wyeth. Mapping a suburb with a single camera using a biologically inspired slam system. *IEEE Transactions on Robotics*, 24(5):1038–1053, 2008.

[16] Michael Milford and Gordon Wyeth. Spatial mapping and map exploitation: A bio-inspired engineering perspective. In Stephan Winter, Matt Duckham, Lars Kulik, and Ben Kuipers, editors, *Spatial Information Theory*, volume 4736 of *Lecture Notes in Computer Science*, pages 203–221. Springer Berlin / Heidelberg, 2007. 10.1007/978-3-540-74788-8_13.

[17] Michael Montemerlo and Sebastian Thrun. FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges.

[18] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *In Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 593–598. AAAI, 2002.

[19] Hans P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2):61–74, 1988.

[20] Robin R. Murphy. *Introduction to AI Robotics*. MIT Press, Cambridge, MA, USA, 1st edition, 2000.

[21] J O'Keefe and J Dostrovsky. The hippocampus as a spatial map. preliminary evidence from unit activity in the freely-moving rat. *Brain Research*, 34(1):171–175, 1971.

[22] E. T. Rolls. Spatial view cells and the representation of place in the primate hippocampus. *Hippocampus*, 9(4):467–480, 1999.

[23] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition, 2003.

[24] Bruno Siciliano and Oussama Khatib, editors. *Springer Handbook of Robotics*. Springer, 2008.

[25] W. E. Skaggs and B. L. McNaughton. Replay of neuronal firing sequences in rat hippocampus during sleep following spatial experience. *Science (New York, N.Y.)*, 271(5257):1870–1873, March 1996.

[26] Trygve Solstad, Charlotte N. Boccara, Emilio Kropff, May-Britt Moser, and Edvard I. Moser. Representation of geometric borders in the entorhinal cortex. *Science*, 322(5909):1865–1868, 2008.

[27] Bruno Steux and O El Hamzaoui. Coreslam: a slam algorithm in less than 200 lines of c code. *Submission ICARCV*, 2010.

[28] H. Strasdat, J. M. M. Montiel, and A. Davison. Scale drift-aware large scale monocular slam. In *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, June 2010.

[29] Niko SÃ¼nderhauf and Peter Protzel. Learning from nature: Biologically inspired robot navigation and slam-aÂ review. *KI - Kanstliche Intelligenz*, 24:215–221, 2010. 10.1007/s13218-010-0038-y.

[30] J. S. Taube, R. U. Muller, and J. B. Ranck. Head-direction cells recorded from the postsubiculum in freely moving rats. I. Description and quantitative analysis. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 10(2):420–435, February 1990.

[31] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series)*. Intelligent robotics and autonomous agents. The MIT Press, August 2005.

[32] Greg Welch and Gary Bishop. An introduction to the kalman filter, 1995.