

Tópicos Selectos de Robots Móviles Inteligentes

José Gabriel Ramírez Torres

2023

Índice general

1. Robots autónomos	4
1.1. Robots aéreos	4
1.1.1. Motivación	4
1.1.2. Cuadricópteros	5
1.2. Sistema operativo robótico (ROS)	7
1.3. Arquitectura de control híbrida	7
1.4. Representación del medio ambiente	10
1.4.1. Representaciones continuas	11
1.4.2. Descomposición en celdas	12
1.4.3. Representaciones topológicas	14
1.4.4. Retos actuales en representación del medio ambiente . . .	15
1.5. Planificación de trayectorias	16
1.5.1. Espacio de configuraciones	17
1.5.2. Problema de la mudanza del piano	18
1.5.3. Árboles aleatorios de exploración rápida (RRT)	19
1.6. Usando mapas para planificación de trayectorias. Planificación global.	22
1.6.1. Planificación basada en grafos	23
1.6.2. Construcción del grafo	23

1.6.3.	Búsqueda de solución en un grafo	26
1.6.4.	Planificación basada en funciones de potencial artificial	31
1.6.5.	Conclusión	32
1.7.	Incluyendo la mecánica en la navegación. Planificación local.	33
1.7.1.	Algoritmo Bug	34
1.7.2.	Histograma del campo vectorial (VFH)	34
1.7.3.	Espacio de velocidades	36
1.7.4.	Campos de potencial artificial	37
1.8.	Conclusiones	37
2.	Robótica probabilista	39
2.1.	¿Por qué necesitamos emplear probabilidad en Robótica?	39
2.2.	Conceptos básicos de probabilidad	40
2.2.1.	Variables aleatorias discretas	41
2.2.2.	Variables aleatorias continuas	41
2.2.3.	Distribución conjunta y probabilidad condicional	42
2.2.4.	Teorema de Bayes	43
2.2.5.	Esperanza matemática de una variable aleatoria	43
2.2.6.	Estimación a partir de una serie de datos	44
2.3.	Estimación bayesiana de estado	44
2.3.1.	Ejemplo	45
2.4.	Actualizaciones bayesianas recursivas	46
2.4.1.	Ejemplo	47
2.5.	Modelo de sensores	48
2.5.1.	Ejemplo: el sensor ultrasónico	49
2.6.	Modelo de acciones	49
2.6.1.	Ejemplo	51

2.7. Filtro de Bayes y Procesos de Markov	51
2.7.1. Algoritmo de Bayes	52
2.7.2. Ejemplo de localización usando un filtro de Bayes	53
2.8. Distribuciones normales	58
2.9. Sistemas lineales y Filtro de Kalman	61
2.9.1. Ejemplo	64
2.10. Filtro de Kalman Extendido	66
2.10.1. Ejemplo	68
2.11. Filtro de partículas	70
2.11.1. Métodos de MonteCarlo	70
2.11.2. Algoritmo del filtro de partículas	72
2.11.3. Algoritmos de remuestreo	73
2.12. Conclusiones	75
3. Visión por computadora	76

Capítulo 1

Robots autónomos

1.1. Robots aéreos

Un robot aéreo o *vehículo aéreo no tripulado* (VANT) es una aeronave sin tripulación a bordo, capaz de realizar múltiples misiones (un misil de crucero no es un VANT, pues no es reutilizable). En realidad, el VANT es sólo una parte de lo que se denomina *sistema aéreo no tripulado* (SANT), que comprende el VANT, un enlace de comunicación, una estación de control y la tripulación en tierra. Es importante resaltar que, aunque el vehículo no tiene tripulación a bordo, en los SANT actuales siempre existe un piloto entrenado que opera el VANT de manera remota desde la estación de control.

1.1.1. Motivación

Al igual que a cualquier otro robot móvil, a los VANT se les confían tareas que se suponen peligrosas o aburridas para que un ser humano se vea directamente implicado en la realización de ellas. Además, se debe tomar en cuenta la ventaja estratégica que implica el contar con el punto de vista privilegiado que proporciona un vehículo aéreo sobre el terreno. Pero el gran costo económico y tecnológico que representa el desarrollo de un SANT había limitado hasta hace unos pocos años su uso en aplicaciones de tipo militar, principalmente en misiones de reconocimiento y exploración y, recientemente, localización y destrucción de blancos.

Sin embargo, la miniaturización y abaratamiento de componentes electrónicos ha permitido el desarrollo de VANTs de bajo costo, lo que ha resultado en una explosión de aplicaciones en el ámbito civil de estos vehículos, como la asistencia en zonas de desastre, misiones de intervención rápida para búsque-

da y rescate, exploración, inspección y vigilancia de instalaciones, transporte de productos e incluso agricultura de precisión, por mencionar sólo algunas de ellas.

Los desarrollos actuales de SANTS no son sistemas autónomos. El VANT sólo es capaz de ejecutar de manera automática unas pocas tareas sencillas, tales como despegue y aterrizaje automáticos, navegación entre puntos preestablecidos, vuelo estable, etc.; pero la misión propiamente dicha aún es realizada por la tripulación humana en tierra a través del enlace de comunicación con el VANT. Los SANTS actuales son incapaces (aún) de analizar el contexto actual de la tarea que realizan y de proponer o modificar un plan de acción, sin intervención humana, sobre el curso de acciones a tomar para alcanzar el objetivo planteado.

Este curso está encaminado precisamente al estudio de los mecanismos y herramientas que permitan incrementar la autonomía de los robots aéreos.

***Tarea:** Proponer una posible aplicación para un VANT. Presentar y discutir sus ventajas potenciales. Detectar cuáles son los retos tecnológicos particulares para este desarrollo.*

1.1.2. Cuadricópteros

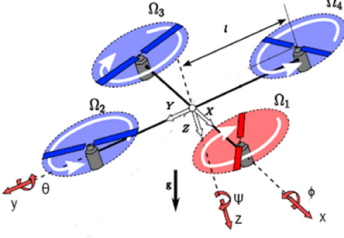
Por la manera en que se mantienen en el aire, los VANT se clasifican en aerostatos (globos y dirigibles), VANT de ala fija (tipo avión), VANT de ala rotatoria coaxial (tipo helicóptero) y VANT de alas rotatorias no coaxiales (tipo multicóptero).

Los VANT tipo multicóptero presentan ventajas muy interesantes, como son el despegue y aterrizaje verticales, capacidad de vuelo estacionario, dimensiones reducidas y facilidad de control.

En el caso de los multicópteros, cada rotor genera dos efectos sobre la aeronave: un flujo de aire que aplica una fuerza (vertical) en un punto específico y, al mismo tiempo, un par de contragiro sobre la aeronave. Es la acción combinada de todos los efectos de los rotores la que permite los movimientos del VANT.

Si bien el modelo de vuelo de un VANT multicóptero es complejo, resulta ser naturalmente desacoplado, es decir, es posible encontrar una transformación lineal de las entradas de control (velocidades de rotación de los rotores) de manera que las salidas del sistema (velocidades lineales y angulares del vehículo) se comporten de forma independiente entre ellas.

En el caso de los cuadricópteros, es posible controlar de forma independiente (desacoplada) la altura, el avance, el desplazamiento lateral y el cabeceo, por medio de una combinación intuitiva de la velocidad de giro de los rotores.

$$\begin{aligned}
\ddot{X} &= (\cos \phi \sin \theta \cos \psi + \sin \phi \sin \psi) \frac{1}{m} U_1 - 1/2 C_x A_x \rho \dot{X} |\dot{X}| & \ddot{\phi} &= \dot{\phi} \dot{\psi} \left(\frac{I_{yy} - I_{zz}}{I_{xx}} \right) + \frac{U_2}{I_{xx}} \\
\ddot{Y} &= (\cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi) \frac{1}{m} U_1 - 1/2 C_y A_y \rho \dot{Y} |\dot{Y}| & \ddot{\theta} &= \dot{\phi} \dot{\psi} \left(\frac{I_{zz} - I_{xx}}{I_{yy}} \right) + \frac{U_3}{I_{yy}} \\
\ddot{Z} &= (\cos \phi \cos \theta) \frac{1}{m} U_1 - g - 1/2 C_z A_z \rho \dot{Z} |\dot{Z}| & \ddot{\psi} &= \dot{\phi} \dot{\theta} \left(\frac{I_{xx} - I_{yy}}{I_{zz}} \right) + \frac{U_4}{I_{zz}}
\end{aligned}$$


$$\begin{aligned}
U_1 &= \sum_{i=1}^4 T_i = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\
U_2 &= (-T_2 + T_4) = b\ell(-\Omega_2^2 + \Omega_4^2) \\
U_3 &= (T_1 - T_3) = b\ell(\Omega_1^2 - \Omega_3^2) \\
U_4 &= (-1)^i \sum_{i=1}^4 M_{D_i} = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2)
\end{aligned}$$

Figura 1.1: Modelo matemático de un cuadricóptero.

Lograr que un multicóptero mantenga una posición estable en el aire no es una tarea simple. De hecho, un multicóptero es una plataforma experimental muy interesante para el desarrollo de algoritmos de control, principalmente por la alta inestabilidad del sistema completo, la fuerte interacción de las distintas entradas y la gran sensibilidad a las perturbaciones externas. En la literatura se han propuesto diferentes técnicas que buscan proporcionar un sistema de control robusto para estos vehículos, por medio de observadores robustos del estado del sistema y una buena identificación paramétrica. Sin embargo, se trata de técnicas que resultan complicadas de implementar en la vida real, por lo que los VANTs comerciales recurren al viejo y bien conocido control PI para el control de posición.

Por la complejidad natural, se aprovecha el desacople de las salidas para realizar un control en cascada, para controlar de manera sucesiva las velocidades angulares (con retroalimentación de giroscopios), después la posición angular (con información de acelerómetros), el desplazamiento horizontal (acelerómetros y flujo óptico) y finalmente la altura de vuelo (altímetro por ultrasonidos o barómetro) del robot. Se emplean únicamente correcciones proporcional e integral, para evitar las derivadas del ruido.

Tarea: Desarrollar una tabla comparativa entre los distintos tipos de VANT, mostrando el principio de sustentación, ventajas, desventajas y tipo de aplicación. Explicar claramente los mecanismos de vuelo de cada uno de ellos. Buscar y explicar al menos dos ejemplos en artículos de revista de control de cuadricópteros.

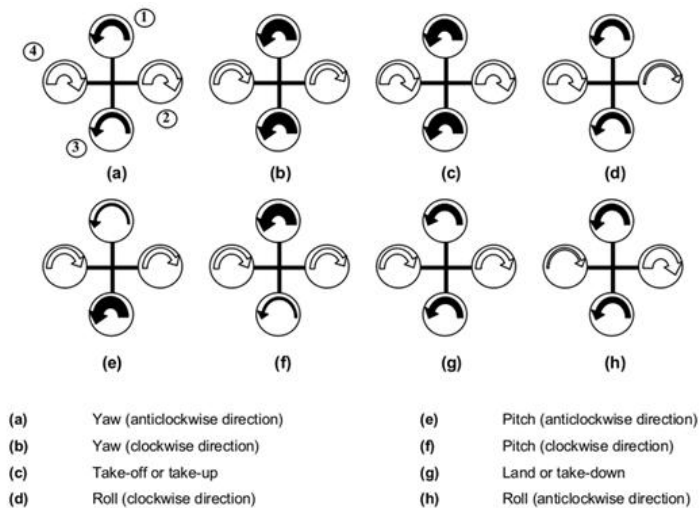


Figura 1.2: Principio de vuelo de un cuadricóptero.

1.2. Sistema operativo robótico (ROS)

- A desarrollar -

1.3. Arquitectura de control híbrida

Como se estudió en el curso de Robots Móviles Inteligentes, las estructuras de control reactivas permiten, a través de relaciones simples de tipo “estímulo - respuesta” (llamados comportamientos) y un proceso de coordinación (o arbitraje) entre respuestas, resolver problemas complejos. En este tipo de esquemas se logra separar el problema principal en un conjunto de subproblemas (alta modularidad), permitiendo al robot responder de manera rápida (reactividad) y robusta a los cambios que puedan presentarse en el medio ambiente, todo ello sin requerir de un modelo en memoria del entorno.

Sin embargo, empleando una arquitectura de control reactiva no es posible garantizar que el robot será capaz de encontrar una solución para un problema dado, ni tampoco es posible garantizar que, si existen distintas soluciones al mismo problema, el robot sea capaz de encontrar la mejor solución.

El mayor inconveniente de las estructuras de control reactivas es que no poseen memoria, lo que vuelve extremadamente difícil la coordinación de múltiples módulos que deben actuar en contextos muy parecidos (pero no iguales),

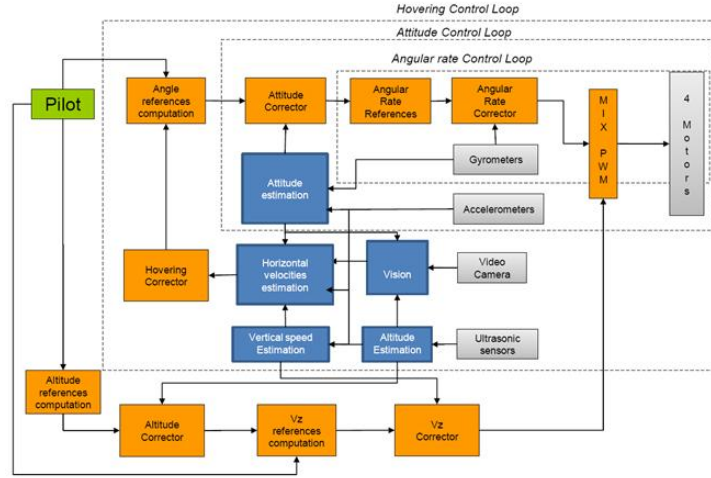


Figura 1.3: Control en cascada de un cuadricóptero.

además de que no permite resolver problemas que consideren estados¹ pasados del robot.

En la arquitectura de control híbrida, se reemplaza la “planificación reactiva” por una “ejecución reactiva”. Es decir, en lugar de decidir en el momento (por competencia, cooperación o coordinación) cuál de los módulos reactivos debe aplicarse en una situación dada, se le permite al robot emplear alguna representación en memoria para elaborar un plan previo de activación de módulos, y únicamente se vigila en tiempo real la correcta activación y ejecución de los comportamientos.

De esta manera, aparecen naturalmente (más bien, empíricamente) tres módulos o capas de control, con diferentes niveles de abstracción del problema a resolver: un planificador que elabora el plan de ejecución para alcanzar el objetivo deseado, un conjunto de mecanismos reactivos capaces de resolver situaciones bien precisas, y un dispositivo de secuenciamiento que conecta los dos anteriores.

Básicamente, el **planificador** (o deliberador, como referencia a los esquemas deliberativos clásicos) es un proceso de alto nivel que consume un gran tiempo de cómputo. Generalmente se encarga de tareas de planificación de la misión, así como la optimización y/o búsqueda de soluciones que no requieren una interacción directa con el medio ambiente. No existe límite para la complejidad computacional o los requerimientos arquitecturales de los algoritmos implantados en el deliberador. Generalmente se emplean lenguajes de alto nivel para el desarrollo del módulo deliberativo.

¹Estado = vector de variables internas y externas del robot.

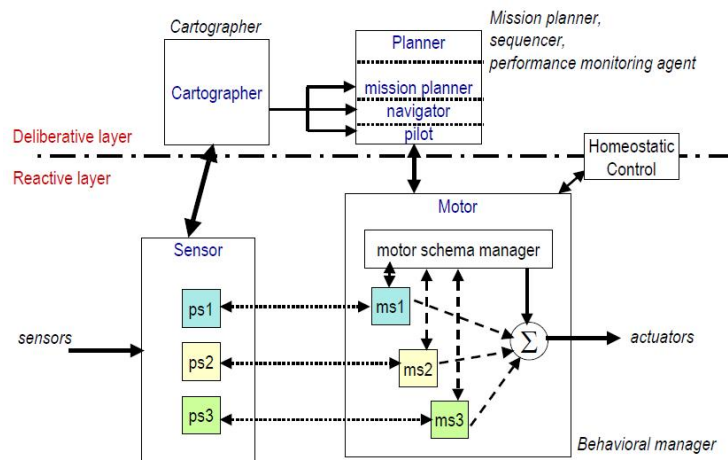


Figura 1.4: AuRA, un sistema de control híbrido.

El **control de bajo nivel** consiste en un conjunto de funciones de transferencia entre el estímulo recogido por los sensores y las acciones efectuadas por los efectores del robot. Cada función de transferencia se denomina “primitiva de control”, “comportamiento” o “habilidad” y permite al robot resolver un problema simple, como por ejemplo seguir una línea o evitar obstáculos. En general, este conjunto de primitivas se desarrolla de manera manual, explotando directamente las características físicas y mecánicas del robot. Existen algunas restricciones computacionales para cada una de las primitivas:

1. Deben de tener un tiempo de cómputo constante y predecible (tiempo real);
2. Deben ser capaces de detectar cuándo el comportamiento está fallando, para permitir a otros módulos realizar acciones correctivas;
3. Deben evitar el uso de memoria interna;
4. El comportamiento resultante debe ser continuo.

Por las restricciones citadas, el módulo de control de bajo nivel generalmente se codifica en lenguajes de bajo nivel, como ensamblador, códigos de operación (*opcodes*), códigos intermedios (*bytecode*) de máquinas virtuales o lenguaje C.

El **secuenciador** (o control homeostático, por su inspiración biológica) es el proceso que sirve de interfaz entre la capa de control de bajo nivel (en tiempo real) y el deliberador de alto nivel (lento). El secuenciador recibe el plan desarrollado por el deliberador y selecciona el comportamiento y establece los parámetros más adecuados, de acuerdo al contexto actual (tiene, por lo tanto,

acceso a la memoria y al estado interno del robot), para acercar al robot al objetivo establecido. Ante una falla de parte del comportamiento de bajo nivel actualmente empleado, el secuenciador debe ser capaz de escoger un nuevo comportamiento para corregir la situación actual, o en su caso, remitir la nueva situación al deliberador para que éste realice un nuevo plan a partir de la condición actual. En cualquier caso, la velocidad de ejecución del secuenciador debe ser tal que le permita tomar decisiones a una tasa más rápida que a la que ocurren los cambios del medio ambiente.

En la mayor parte de las implementaciones, el secuenciador está desarrollado en un lenguaje que permite la construcción de programación paralela, con múltiples hilos de ejecución activos al mismo tiempo (alarmas, interrupciones hardware y software, concurrencia), para permitir al robot tratar con situaciones y contingencias que no podrían ser tratadas en un esquema secuencial con un único hilo activo.

***Tarea:** Empleando artículos y publicaciones de reconocido prestigio científico, documentar y explicar en clase (al menos) dos ejemplos de robots móviles que empleen una arquitectura híbrida para su control. Para cada uno de ellos, identificar claramente las funcionalidades de cada capa, así como el o los lenguajes de programación empleados en su desarrollo. ¿Cuál sería, según su propio criterio, la razón principal por la que los procesos necesarios a un robot móvil se organizan naturalmente en tres capas de control?*

1.4. Representación del medio ambiente

Para poder planear tareas complejas de forma eficiente, buscando soluciones óptimas bajo algún criterio en particular, es necesario que el robot cuente con una representación en memoria del modelo del problema. En particular para los robots móviles, muchas de las tareas consisten en planificar trayectorias que lo lleven desde el punto actual a algún punto de destino de la manera más rápida, más eficiente (en consumo de energía) o más estable, por citar algunos ejemplos. El robot requiere entonces contar con una representación del medio ambiente en su memoria, con la cual poder trabajar.

Llamaremos “mapa” a cualquier representación del medio ambiente dentro de la memoria del robot. El tipo de mapa que se seleccione depende de tres características importantes:

1. La precisión de la tarea que deseamos que el robot sea capaz de realizar;
2. La precisión de los sensores y los actuadores con los que el robot dispone;

3. El costo computacional que deseamos invertir en el almacenamiento del mapa, así como la complejidad de los cálculos que deberán realizarse a partir de él para obtener los planes de ejecución de la tarea asignada.

1.4.1. Representaciones continuas

En las representaciones continuas, las características del medio ambiente que se registran en el mapa se almacenan en su posición *exacta*. En general, se le utiliza en medios ambientes planos y cerrados, es decir, no se consideran objetos tridimensionales y el mundo completo se encuentra confinado en una región bien precisa, lo que permite limitar la cantidad de memoria necesaria para su representación al número de objetos presentes en el mundo. Bajo estas consideraciones, el mapa resultante es una descripción del medio ambiente que emplea **primitivas geométricas**, tales como polígonos o líneas rectas.

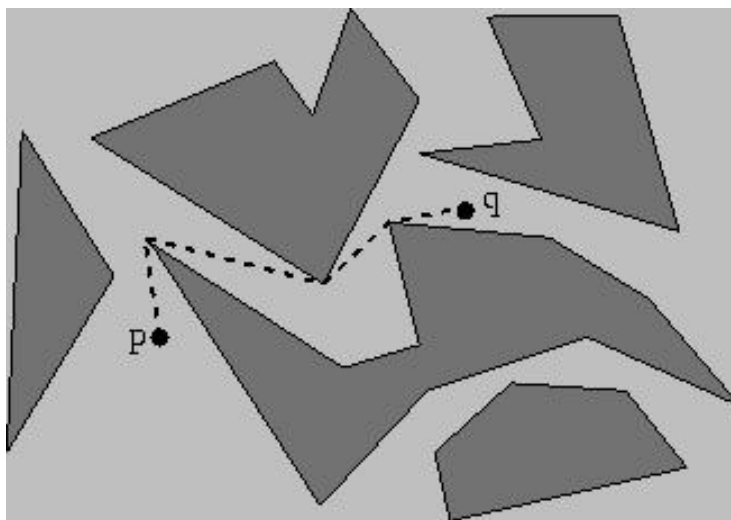


Figura 1.5: Ejemplo de un mapa geométrico, basado en polígonos.

En general, las representaciones continuas permiten un manejo de una gran precisión del medio ambiente y de la posición propia del robot dentro de éste, así como una descripción altamente detallada de los objetos que se encuentran en el mundo, empleando (bajo la consideración de un medio ambiente cerrado) una cantidad relativamente pequeña de memoria para la representación, pues el mapa contiene las características que describen a los objetos. Por otra parte, la manipulación de una representación geométrica puede resultar costosa desde el punto de vista de complejidad espacial y temporal, dado que se requieren algoritmos propios de la Geometría Computacional.

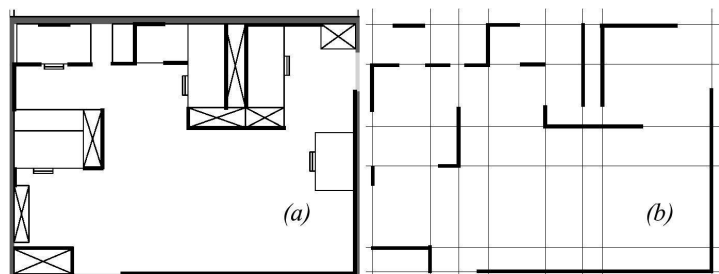


Figura 1.6: Ejemplo de un mapa geométrico, basado en líneas.

***Tarea:** Documentar las distintas maneras en que se puede **implementar** un polígono dentro de la memoria de una computadora. Hacer lo mismo para una descripción basada en líneas infinitas. Encontrar desarrollos científicos en robótica móvil que empleen este tipo de representaciones para sus mapas del medio ambiente. ¿Qué otras primitivas geométricas podrían utilizarse para construir un mapa del medio ambiente?*

1.4.2. Descomposición en celdas

Con el propósito de simplificar los algoritmos requeridos para la manipulación del mapa, es posible considerar el sacrificio de la precisión y fidelidad del mapa geométrico por una representación basada en abstracciones, es decir, capturar las características más relevantes de una región del medio ambiente, con propósitos de su manipulación durante el proceso de planificación de tareas.

Un primer enfoque posible es el denominado **descomposición en celdas exactas**. En este enfoque, el mundo (particularmente el espacio libre donde el robot puede navegar) es dividido en celdas por medio de algún criterio geométrico, de tal manera que no es relevante para el proceso de planificación saber exactamente dónde se encuentra el robot, sino únicamente en qué celda está. Así, la representación del medio ambiente es extremadamente compacta, pues se captura en un grafo de unos pocos nodos (que representan las celdas) y las aristas que unen dos celdas que están físicamente conectadas en el mundo real, denominado **mapa de adyacencias locales**.

Puede observarse claramente que la descomposición en celdas exactas depende totalmente de la geometría propia del medio ambiente, por lo que si esta información es difícil de coleccionar o no está disponible (como es el caso de un ambiente desconocido), esta representación no puede ser utilizada.

Una alternativa a esta situación es emplear una **descomposición en celdas fijas**, en la que el mundo real es aproximado por una representación discreta,

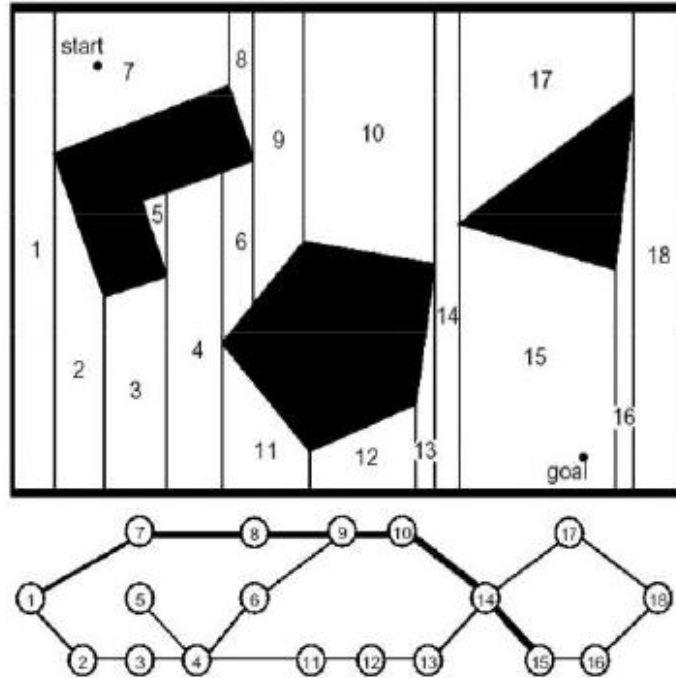


Figura 1.7: Ejemplo de un mapa por descomposición en celdas exactas.

a través de una geometría impuesta. Cada celda recibe entonces un valor que indica si el robot puede navegar a través de ella (espacio libre) o hay un obstáculo (espacio ocupado).

Este tipo de representación es muy popular en las aplicaciones reales de robots móviles actuales, en particular la denominada **rejilla de ocupación**, dado que la información obtenida a través de los sensores, junto con la posición actual del robot (localización), permiten actualizar rápidamente el mapa. En este caso, la celda contiene el número de veces que los sensores han “visto” (o han “dejado de ver”) un obstáculo en una posición determinada del mapa, lo que permite tener un estimado del grado de traversabilidad de una región del mundo.

Sin embargo, existen dos grandes inconvenientes para la representación por celdas fijas:

1. Al imponer una geometría, se pierde una gran precisión del mapa, llegando incluso al extremo de cerrar caminos que en la realidad sí son navegables; se dice que la representación pierde *completez*, al excluir posibles soluciones de navegación.

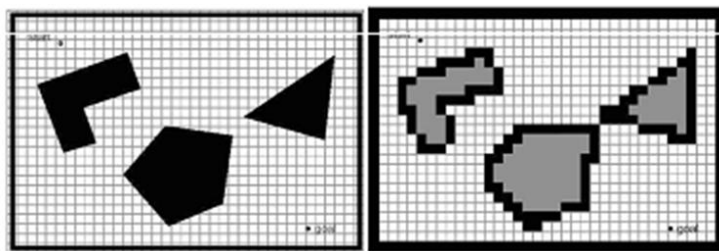


Figura 1.8: Ejemplo de un mapa por descomposición en celdas exactas.

2. El punto anterior puede remediarse aumentando la resolución del mapa (reduciendo el tamaño de las celdas), con el consiguiente incremento de memoria requerida para almacenar el mapa.

***Tarea:** Documentar (con desarrollos científicos empleados en robótica móvil) el esquema de representación del medio ambiente por medio de descomposición en celdas adaptativas, que emplea una descripción basada en **quad-trees**. ¿Cuáles son las ventajas y desventajas de este esquema? ¿Cómo puede hallarse una trayectoria desde la posición actual del robot a la posición objetivo, al emplear este tipo de representación?*

1.4.3. Representaciones topológicas

Es posible emplear representaciones del medio ambiente con un nivel de abstracción aún mayor, evitando completamente el emplear relaciones métricas (medidas) de los objetos, para concentrarse únicamente en las características contextuales del medio ambiente.

Un **mapa topológico** es una representación en grafo donde los nodos indican áreas de un tamaño no fijado *a priori* del mundo (libre y/o ocupado) que el robot es capaz de diferenciar claramente de otras, empleando sus sensores. Al igual que en un mapa de adyacencias locales, las aristas conectan dos áreas para las cuales el robot posee los medios de navegar de una a otra.

Para poder explotar adecuadamente un mapa topológico, el robot debe contar con las habilidades suficientes que le permitan reconocer el área del espacio de trabajo en la que se encuentra y distinguir claramente los accesos y salidas, así como con los comportamientos apropiados que le permitan navegar de un tipo de área al siguiente (negociar puertas, seguir líneas, identificar marcadores, transitar en pasillos, etc.).

La ventaja principal de un mapa topológico radica en su propiedad para

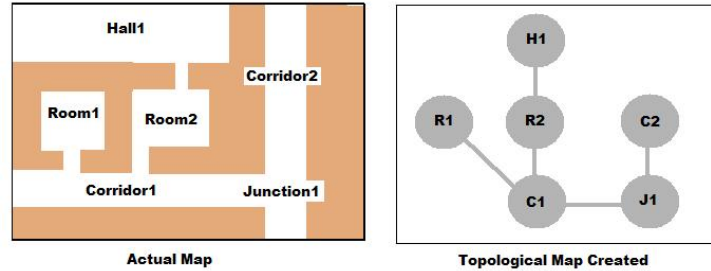


Figura 1.9: Ejemplo de un mapa topológico.

modelar características no métricas del medio ambiente, lo que permite emplear parámetros como la textura, el material, el color o la iluminación como características representativas del medio ambiente.

Tarea: El uso de mapas topológicos no es muy común en robótica móvil. Documentar algunos desarrollos científicos que empleen una representación topológica del medio ambiente. ¿Cuáles son las consideraciones particulares para cada una de estas implementaciones? ¿Emplean algún otro tipo de representación con información adicional?

1.4.4. Retos actuales en representación del medio ambiente

Las representaciones vistas anteriormente consideran que el mundo en el que evoluciona el robot móvil es estático y que puede ser representado con una estructura de datos estática. Por lo tanto, no incluyen de manera explícita información sobre objetos móviles en el escenario. Quizás el reto más importante de cualquier representación del medio ambiente es poder distinguir entre obstáculos permanentes y obstáculos transitorios, particularmente cuando el sensor se desplaza también en el medio ambiente, como es el caso de los sensores embarcados en un robot móvil.

El segundo inconveniente con las representaciones vistas anteriormente, es que tampoco se modela un grado de *traversabilidad* del medio ambiente, es decir, el grado de densidad de obstáculos en un espacio abierto. Dado que los mapas registran las distancias entre los objetos presentes en el medio ambiente, los espacios con pocos elementos presentan una dificultad importante para ser representados dentro del mapa del robot móvil.

Finalmente, otro reto igualmente importante es la fusión de información proveniente de sensores heterogéneos, con anchos de banda de información dis-

tintos y naturaleza de datos diferente. Así entonces, resulta difícil combinar en una única representación información proveniente de un sensor lento, con poco ancho de banda y una alta incertidumbre como un sensor ultrasónico, con la información proveniente de un sensor rápido, con alto ancho de banda y sin información de distancia como una cámara de video.

1.5. Planificación de trayectorias

La palabra *planificar* tiene diversas acepciones según el contexto en el que se le emplea. En los inicios de la inteligencia artificial, *un plan es un secuencia (implica orden) de operadores lógicos o de acciones, que llevan de una situación inicial a una situación final deseada*. En la actualidad, se incluyen nociones más complejas que los operadores lógicos o las acciones, tales como los procesos de decisión de Markov, los estados de información incompleta y la teoría de juegos.

En el contexto de la robótica, planificar consiste en describir una tarea de alto nivel (dada por el usuario) como una secuencia de primitivas de bajo nivel realizables por la mecánica del robot. Así, el plan puede ser empleado para ejecución, para refinamiento o como bloque de construcción para planes más complejos.

Los ingredientes básicos de un plan son los siguientes:

1. **Descripción de estados.** Un estado es el conjunto de parámetros que describen completamente la situación actual del problema (o del robot, en nuestro caso). El conjunto de todos los estados posibles define el **espacio de búsqueda** del planificador. En general, este espacio de búsqueda es demasiado grande para trabajar directamente con él.
2. **Tiempo de ejecución.** Dado que las acciones forman una secuencia temporal ordenada, el instante de tiempo t es parte del problema, aunque generalmente sea de forma implícita. Existen problemas que requieren considerar el tiempo de forma explícita, por lo que el parámetro t forma parte de la descripción de estados.
3. **Estados inicial y objetivo.** Indican los puntos inicial y final del proceso de planificación. Evidentemente, se puede incluir el parámetro t en ellos si forma parte del objetivo buscado (plazos límite o dependencias entre subtareas, por ejemplo).
4. **Acciones.** Una acción es un operador que se aplica al estado actual del problema para transformarlo en otro estado. Las acciones pueden ser continuas (modelos diferenciales), discretas (modelos de recurrencia) o abstractas. Incluso, las acciones aplicables a un estado pueden ser distintas a

las acciones aplicables a otro estado. Puesto en palabras simples, el planificador debe seleccionar, para cada estado del espacio de búsqueda, la acción (o acciones) que acerquen al agente (robot) al objetivo deseado. En muchos casos, las acciones son consideradas de manera implícita, es decir, dos estados están conectados si se sabe que el agente (robot en nuestro caso) cuenta con las habilidades necesarias para llegar de uno a otro, sin necesidad de describirlas.

5. **Criterio.** Indica cuál es la métrica que guía el proceso de búsqueda de planificación. Generalmente se consideran dos tipos de criterio: de factibilidad (encontrar una solución) y de optimización (encontrar la mejor solución según un parámetro de desempeño). Para la mayoría de los problemas en robótica móvil, el problema de factibilidad es de por sí bastante complejo, por lo que alcanzar el óptimo suele ser muy difícil. En la práctica, resulta complicado definir completamente el estado del sistema, lo que aunado a la incertidumbre en su representación, vuelve casi imposible definir un criterio de optimización. Por ello, se prefiere el desarrollo de planificadores factibles que proporcionen buenas soluciones, a planificadores óptimos incapaces de producir soluciones realizables.
6. **Completez.** La completez es un problema de computabilidad: El planificador se denomina *completo* cuando es capaz de determinar **correctamente** si existe una secuencia de acciones que lleven desde el estado inicial hasta el estado objetivo (y evidentemente entregarla como resultado), y si es capaz también de reportar **correctamente** cuando no existe tal plan. En cualquier otro caso, el planificador es *incompleto*.

1.5.1. Espacio de configuraciones

En el caso de planificación de trayectorias, el objetivo es determinar la secuencia de movimientos que lleva al robot desde una posición inicial en el *espacio de trabajo* hasta una nueva posición considerada la meta. Es necesario encontrar un conjunto de k parámetros (*i.e.* el estado) que describa de manera **unívoca** la posición del robot en el espacio de trabajo; a dicho conjunto de parámetros se le denomina **vector de configuración** o simplemente **configuración**. De esta manera, la compleja estructura tridimensional del robot queda reducida a un vector de dimensión k .

De la definición de vector de configuración, se deriva el **espacio de configuraciones** como el conjunto de todas las configuraciones posibles que puede tomar un robot en el espacio de trabajo. Este espacio de configuraciones, o **C-space**, se puede representar como un hipercubo de k dimensiones donde cada punto es una configuración posible del robot.

Es muy importante resaltar que el espacio de trabajo y el espacio de configuraciones de un robot, si bien están estrechamente relacionados, son dos des-

cripciones completamente diferentes. En particular, la dimensión del espacio de configuraciones depende directamente de la mecánica del robot, lo que no es el caso del espacio de trabajo.

Estrictamente hablando, la planificación de trayectorias se realiza utilizando el C -space como espacio de búsqueda. Bajo ciertas condiciones (en particular, cuando no existen restricciones de movimiento sobre los k parámetros que componen el vector de configuración, es decir, cuando estos parámetros son independientes entre sí) es posible dividir el espacio de configuraciones C en dos subconjuntos complementarios: el espacio de configuraciones ocupado C_{obs} que agrupa todas las configuraciones en las que el robot choca con algún objeto del espacio de trabajo, y el espacio de configuraciones libres $C_{free} = C - C_{obs}$ en las que el robot puede desplazarse sin colisiones. Si los parámetros del vector de configuración son independientes, entonces una trayectoria en el espacio libre C_{free} representará una trayectoria sin colisiones en el espacio de trabajo real.

1.5.2. Problema de la mudanza del piano

El problema anterior se denomina **Problema de la mudanza del piano** (*piano mover's problem*) y puede definirse formalmente de la manera siguiente:

1. Dado un mundo W (espacio de trabajo).
2. Dada una región $O \subseteq W$ que representa los obstáculos en el mundo.
3. Dado un robot R (cuya mecánica define el vector de configuración).
4. Determinar el espacio de configuraciones C , empleando W , O y R para definir los subespacios C_{free} y C_{obs} .
5. Dadas una configuración inicial $q_I \in C_{free}$ y una configuración final $q_F \in C_{free}$.
6. Construir un algoritmo completo capaz de encontrar una trayectoria (continua) $\tau(t) : [0, 1] \rightarrow C_{free}$, tal que $\tau(0) = q_I$ y $\tau(1) = q_F$, o reportar correctamente que dicha trayectoria no existe.

Tomando como ejemplo un robot manipulador 2R (con dos articulaciones de rotación) y base fija, podemos observar que su posición en el espacio queda totalmente determinada por los valores angulares de sus articulaciones $\theta_1 \in [0, 2\pi]$ y $\theta_2 \in [0, 2\pi]$. Así, el espacio de configuraciones es un cuadrado en el sistema (θ_1, θ_2) ². Cada obstáculo en el espacio de trabajo se representa entonces como una “sombra” en el espacio de configuraciones. La planificación de trayectorias

²En el caso de los robots manipuladores con base fija, el espacio de configuraciones es idéntico al espacio de parámetros articulares del robot.

se reduce a encontrar un camino dentro de C_{free} que lleve del punto q_I al punto q_F ; dicha trayectoria representa la secuencia de movimientos, en el mundo real, entre la configuración inicial y la configuración final deseada.

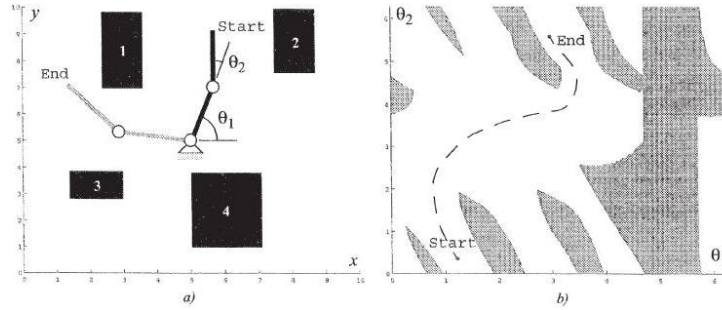


Figura 1.10: Ejemplo de planificación de trayectorias en el C-space para un robot manipulador 2R.

Desafortunadamente, es muy difícil determinar la forma exacta de la “sombra” que proyecta un obstáculo dentro del $C-space$; de hecho, está demostrado que este problema es PSPACE-duro, lo que implica un problema de tipo NP-duro. Por ello, la mayoría de los algoritmos de planificación evitan calcular directamente el espacio de configuraciones.

Tarea: Para evitar calcular la forma exacta de la frontera entre el espacio libre y el espacio ocupado, algunos planificadores utilizan una versión discreta del $C-space$. Documentar algunos desarrollos científicos que empleen una representación discreta del espacio de configuraciones. ¿Cuáles son las diferencias, ventajas y desventajas entre la representación continua del $C-space$ y su correspondiente representación discreta? ¿Cuáles son las diferencias entre esta representación y la representación de rejilla de ocupación (celdas fijas)?

1.5.3. Árboles aleatorios de exploración rápida (RRT)

Como se puede apreciar en la sección anterior, en el caso de los robots manipuladores de base fija el espacio de configuraciones es idéntico al espacio articular. La mayoría de los robots industriales poseen 5 o 6 articulaciones y, en casos como los robots de la industria automotriz, pueden llegar a tener hasta 8 o 9 articulaciones. Tratar de determinar la forma del $C-space$ de tan alta dimensión para el cálculo de trayectorias es completamente intratable, por lo que la mayoría de los robots industriales cuentan con una caja de enseñanza (*teaching box*) que permite indicar de forma manual las trayectorias que deseamos que el robot realice, pues en general no es necesario recalculer trayectorias durante la operación del robot.

Cuando es necesario resolver un problema de planificación de trayectorias desde un punto inicial hasta un punto final dentro de un espacio de configuraciones de alta dimensión (como por ejemplo, un brazo manipulador de servicios *-Robotina-* o la predicción de la estructura en el plegamiento de una proteína) es imposible de resolver de manera exhaustiva. Tampoco es posible considerar la reducción de dimensión del problema, pues se trata de parámetros articulares. Los principales métodos heurísticos requieren de una función de evaluación que permita discriminar entre diferentes soluciones, pero la complejidad del problema se refleja también en determinar dicha función.

En esta situación, los **árboles aleatorios de exploración rápida**³ suelen ser un enfoque apropiado de solución. Los algoritmos que emplean este tipo de enfoque se denominan también **métodos de mapas probabilísticos**⁴.

El funcionamiento general del algoritmo para generar un árbol de exploración aleatoria $T = (Q, E)$, formado por el conjunto Q de configuraciones y el conjunto E de aristas es el siguiente:

1. Se inicia el árbol T con $Q = \{q_I\}$ y $E = \emptyset$.
2. Se genera una configuración aleatoria q_r y, ocasionalmente, se aplica que $q_r = q_F$.
3. Se localiza en el conjunto Q el nodo q_n más cercano a q_r .
4. A partir de q_n , se aplica una acción de movimiento válida (sin colisión), para acercar el robot a la configuración q_r , y obtener la nueva configuración q_e .
5. Se agrega q_e al conjunto Q y la arista (q_n, q_e) al conjunto E .
6. Si la configuración q_F se agregó al árbol T , se detiene la construcción del árbol T . Si no fue así, se repite el proceso con una nueva configuración q_r .
7. Se explora el árbol construido para encontrar la trayectoria de q_I hasta q_F .

El resultado del algoritmo sólo considera un criterio de factibilidad, pero puede demostrarse que es *probabilísticamente completo*, es decir, que si existe una solución y si se le permite crecer infinitamente, finalmente será capaz de encontrarla.

Algunas extensiones al algoritmo, con el objetivo de acelerar la planificación, incluyen:

³*Rapid Exploring Random Tree, RRT*

⁴*Probabilistic Roadmap Method, PRM*

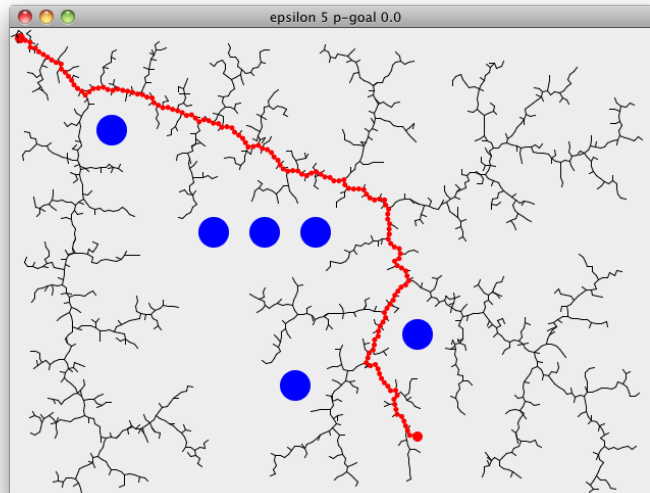


Figura 1.11: Ejemplo de un RRT en un espacio de configuraciones bidimensional.

- Crecer dos árboles al mismo tiempo, tanto desde la posición inicial como de la posición final, hasta que se encuentren;
- Sesgar la generación del nodo aleatorio q_r por medio de alguna heurística para dirigir la búsqueda (técnicas de muestreo);
- Su implementación en cómputo paralelo.

En general, el propósito de construir un RRT es únicamente encontrar una solución inicial factible al problema de planificación y, a través de un proceso de refinamiento, encontrar una mejor solución de acuerdo a un criterio de optimización.

Tarea: Documentar dos desarrollos científicos recientes que realicen la construcción de un RRT para resolver algún problema de planificación, uno para robots manipuladores y otro para robots móviles. ¿Cuáles son las consideraciones particulares para cada una de estas implementaciones? ¿Qué características comparten? ¿Qué técnicas emplean para acelerar el proceso de búsqueda de soluciones? ¿Qué algoritmo utilizan para refinar la solución?

1.6. Usando mapas para planificación de trayectorias. Planificación global.

En el caso de los robots móviles la configuración del robot, además de contener los parámetros articulares, debe contener la posición espacial del punto de referencia de la base: 3 variables para el caso 2D (dos traslaciones y una rotación) y 6 variables para el caso 3D (tres traslaciones y tres rotaciones). Esto aumenta la dimensión del C-space a considerar.

Sin embargo, cuando es posible considerar al robot móvil como un único cuerpo sólido (preferentemente un círculo -o una esfera-) de radio definido, cuyos desplazamientos sobre los ejes principales del espacio son independientes entre sí, se puede entonces trabajar directamente con una representación geométrica del espacio de trabajo, sin necesidad de construir el espacio de configuraciones, utilizando la posición del punto de referencia como vector de configuración del robot (*reducción a un único punto*). Para ello, se realiza una operación morfológica de dilatación sobre los obstáculos del medio ambiente, o se considera una distancia de seguridad entre el robot y los obstáculos durante la planificación, superior al radio del círculo que engloba al robot.

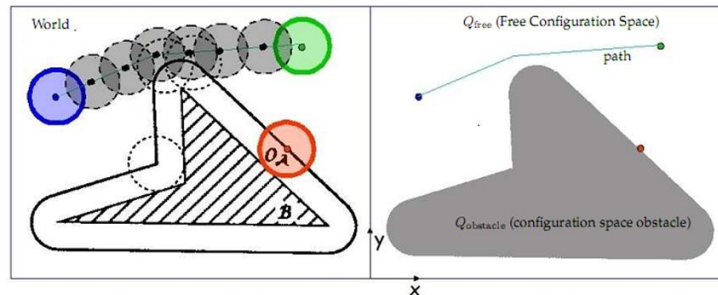


Figura 1.12: Reducción a un único punto de un robot móvil en un espacio de trabajo de dos dimensiones, con transformación morfológica del obstáculo.

Si bien existen diversas maneras de utilizar los mapas para la planificación de trayectorias, la mayoría de las propuestas pueden clasificarse en dos categorías principales:

- Las propuestas que transforman la representación del medio ambiente en un grafo cuyos vértices indican las locaciones (lugares) importantes del mapa y las aristas representan la adyacencia entre éstas. Los mapas de descomposición en celdas exactas y en celdas fijas pertenecen a esta categoría.
- Las propuestas que asignan a cada punto del espacio libre del medio ambiente alguna función matemática, denominada *función de potencial*, de

tal manera que pueda utilizarse el gradiente de esta función para dirigir el movimiento del robot móvil.

1.6.1. Planificación basada en grafos

Los grafos son entidades matemáticas ampliamente utilizadas para la representación de problemas, para las cuales existen algoritmos eficientes y robustos para la búsqueda de soluciones. La planificación basada en grafos está compuesta por dos etapas principales:

- La construcción del grafo, a partir de la representación del medio ambiente.
- La búsqueda de la solución en el grafo.

1.6.2. Construcción del grafo

Hasta ahora, hemos visto que la descomposición en celdas genera una representación en forma de grafo del medio ambiente; sin embargo, existen otras representaciones en grafo que veremos a continuación.

Grafo de visibilidad

Cuando se cuenta con una representación poligonal del medio ambiente, se emplean los vértices de los obstáculos y las posiciones inicial y final como vértices de nuestro grafo y las aristas unen dos vértices para los que existe una línea directa de visión entre ellos (evidentemente, las líneas de visión indican la ruta más corta entre esos dos vértices). El grafo que se obtiene se denomina **grafo de visibilidad** o *grafo-V*.

Los grafos de visibilidad son (¿fueron?) bastante populares en robótica móvil, principalmente por dos características importantes:

- Son relativamente sencillos de construir. En particular cuando el medio ambiente es naturalmente poligonal (oficinas y pasillos, por ejemplo).
- Es posible demostrar que la solución óptima, es decir, la trayectoria más corta entre la posición inicial y la posición final, está representada dentro del grafo de visibilidad.

Sin embargo, el inconveniente principal de este tipo de planificación es que la solución obligatoriamente pasa muy cerca de los obstáculos, lo que en la práctica reduce enormemente los márgenes de seguridad y de maniobra del robot.

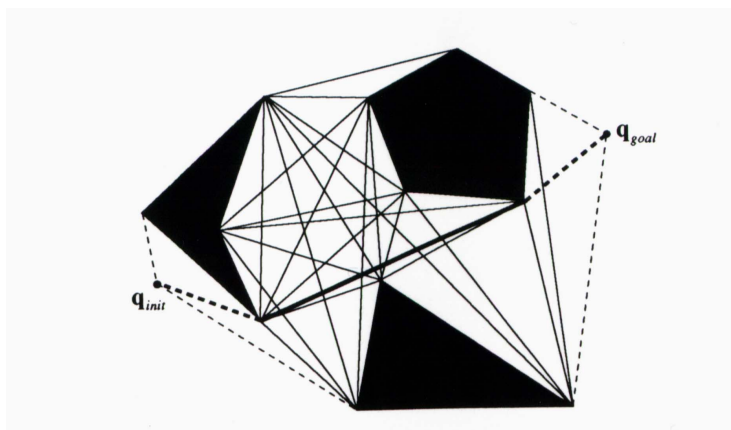


Figura 1.13: Ejemplo de construcción de un grafo de visibilidad.

Tarea: Potencialmente, el número de aristas que pueden aparecer en un grafo de visibilidad es n^2 , con n el número total de vértices en el medio ambiente considerado. Para reducir el número de aristas (y reducir el espacio en memoria requerido para el almacenamiento del grafo), sin perder la solución óptima, se propuso una construcción alternativa denominada grafo de cotangentes o grafo-T. Documentar algún desarrollo científico que emplee esta representación. ¿Cuál es el principio de construcción de esta representación? ¿Cuál es el algoritmo empleado para su construcción?

Grafo de Voronoi

En lugar de dilatar los obstáculos, es posible contraer el espacio libre para realizar la planificación de trayectorias. El grafo de Voronoi es un ejemplo de transformación morfológica por retracción del espacio libre. Cuando en una representación geométrica de un medio ambiente se buscan todos los puntos que sean equidistantes a dos obstáculos al mismo tiempo (incluido el borde del espacio de trabajo), se obtiene la representación en **grafo de Voronoi**⁵. Para completar el grafo, se agregan a continuación la posición inicial y final deseadas, que se conectan al grafo en los puntos más cercanos. El grafo resultante es el que puede explotarse para la planificación de trayectorias.

El tipo de trayectorias resultante tiende a alejarse de los obstáculos, lo que aumenta la ejecutabilidad de la trayectoria (mayor margen de seguridad y de maniobra). Naturalmente, las trayectorias obtenidas no son óptimas en el sentido

⁵Cuando el medio ambiente es poligonal, el grafo de Voronoi está compuesto por segmentos de recta y de parábola únicamente.

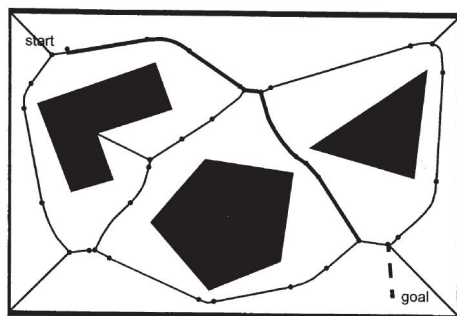


Figura 1.14: Ejemplo de construcción de un grafo de Voronoi.

de longitud recorrida por el robot.

***Tarea:** Documentar algún desarrollo científico que emplee un grafo de Voronoi para la planificación de trayectorias. La operación empleada para calcular el grafo de Voronoi se denomina **transformación de distancia** seguido de una búsqueda de máximos. En el artículo seleccionado, ¿cómo construyen el grafo de Voronoi? ¿Qué consideraciones se requieren?*

Descomposición en celdas

Ya hemos estudiado este tipo de representación y el grafo que modela la adyacencia entre las distintas regiones, tanto para el caso de celdas exactas como para el caso de celdas fijas (rejilla).

Sólo resta señalar que en la actualidad la representación por **rejilla de ocupación** es una de las más populares en robótica móvil, principalmente porque a diferencia de las otras representaciones, no se requiere de información a priori para construir el modelo del entorno.

En efecto, el modelo ya está impuesto, sólo resta llenar las casillas con la información que se vaya colectando durante la operación del robot, lo que permite utilizarla en ambientes desconocidos y no estructurados. Además, es muchísimo más sencilla de actualizar con nueva información que las otras representaciones.

Finalmente, cabe resaltar que las rejillas rectangulares son las más empleadas, pero se pueden proponer otro tipo de retículas multidimensionales donde las celdas no sean consideradas adyacentes, aunque sean físicamente contiguas, para representar la factibilidad del movimiento del robot.

***Tarea:** Estudiar el artículo “Differentially Constrained Mobile*

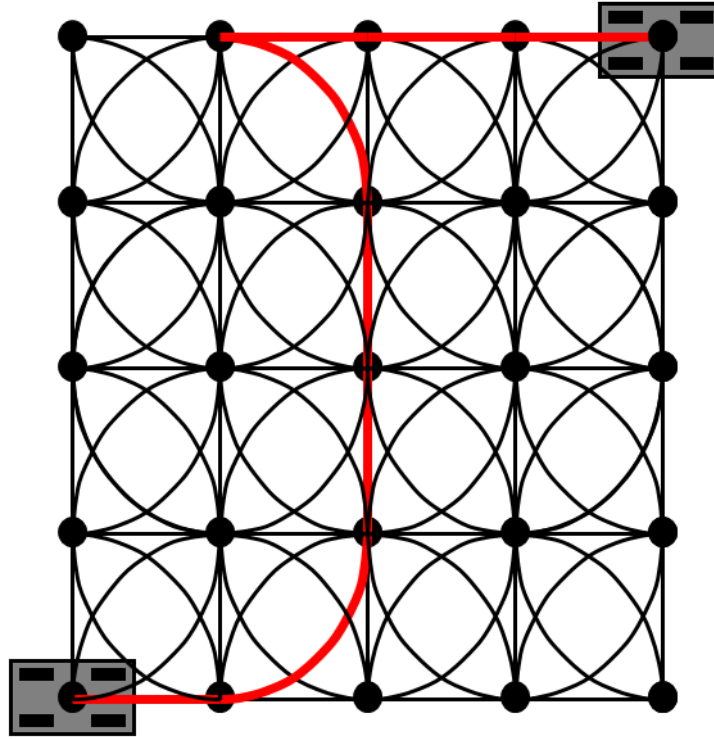


Figura 1.15: Ejemplo de un planificador de trayectorias basado en retículas de estado.

*Robot Motion Planning in State Lattices”, de M. Pivtoraiko et al.
 Describir cómo funciona su planificador de trayectorias basado en
 retículas de estado.*

1.6.3. Búsqueda de solución en un grafo

Sin importar cuál sea el método empleado para construir una representación en forma de grafo del medio ambiente, el planificador de trayectorias debe buscar en dicho grafo el camino óptimo que une el nodo inicial con el nodo final, bajo algún criterio de desempeño. De acuerdo a la manera en cómo el algoritmo de planificación explora el grafo, se pueden considerar los siguientes enfoques de resolución:

1. Algoritmos de búsqueda en anchura
2. Algoritmos de búsqueda en profundidad

3. Algoritmo de Dijkstra
4. Algoritmos heurísticos (A^* y D^*)

Búsqueda en anchura

El principio de búsqueda es sencillo: a partir de un nodo inicial, se calcula el costo de desplazamiento hacia todos los nodos que le son adyacentes (se “expande” el nodo). A continuación, cada uno de los nodos adyacentes es “expandido” a su vez, y así sucesivamente hasta alcanzar el nodo final.

Es importante aclarar que, en este tipo de algoritmos, la distancia entre nodos es constante, pues el algoritmo construye un árbol donde la distancia está dada por la profundidad a la que se encuentra el nodo. Los incrementos de distancia son, pues, discretos.

Cada nodo se visita una sola vez, en el momento de su “expansión”, por lo que el orden del algoritmo es lineal en el número de celdas que componen la representación y es independiente de la cantidad de obstáculos o la complejidad espacial del medio ambiente. Requiere, sin embargo, mantener todos los valores de distancia calculados previamente en la memoria durante todo el proceso de expansión de los nodos, pues serán requeridos para construir la trayectoria.

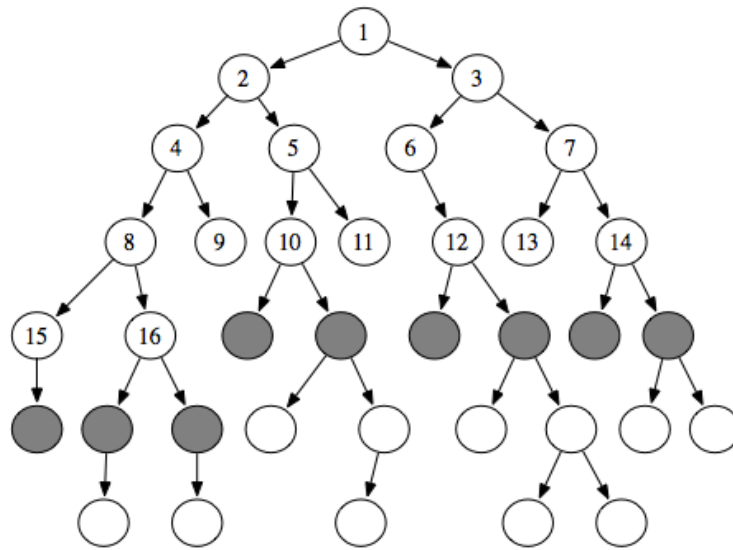


Figura 1.16: Orden de búsqueda en anchura.

Un ejemplo muy interesante de este tipo de algoritmos, aplicado a rejillas rectangulares de ocupación, es el de los algoritmos denominados de *frente de onda*, de *potencial artificial numérico* o de *incendio de la pradera*. En este caso,

la celda objetivo se marca con un '0', las celdas vecinas con un '1', las siguientes con un '2', y así sucesivamente hasta marcar todo el espacio libre. Se obtiene una **transformación de distancia** del espacio libre que emplea **distancia de Chebyshev**. La trayectoria desde cualquier punto del mapa hasta la celda objetivo es muy sencilla de determinar: basta con elegir a cada paso la celda vecina con el valor más bajo, hasta alcanzar la posición final.

Es muy simple modificar este algoritmo para reflejar diferentes tipos de distancia entre celdas adyacentes: distancia de Manhattan, distancia de Chebyshev o distancia casi-euclidiana.

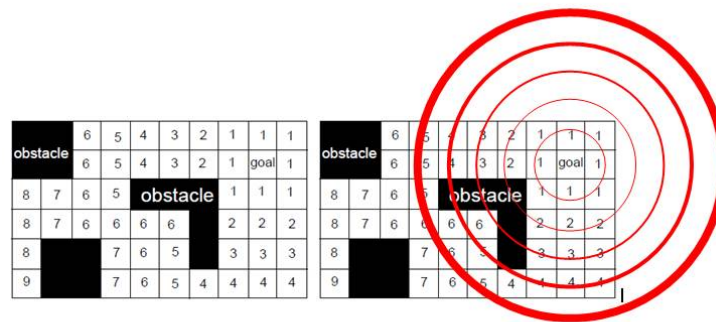


Figura 1.17: Ejemplo de planificación por frente de onda, con distancia de Chebyshev.

Tarea: La implementación “natural” de un algoritmo de frente de onda, empleando una pila para almacenar los nodos visitados, no es la más eficiente. Una forma más sencilla y eficiente es emplear una variante del algoritmo de “chamfer”, utilizado para calcular la transformada de distancia de una imagen binaria. El algoritmo de chamfer se basa en el barrido sucesivo de una(s) máscara(s) de distancias sobre la retícula de la imagen. Explicar el funcionamiento del algoritmo de chamfer. Documentar un artículo científico que emplee el algoritmo de chamfer para la planificación de trayectorias (variante propuesta inicialmente por M. A. Jarvis). ¿Cuáles son las consideraciones del algoritmo? ¿Cómo se modifica la máscara para calcular otras distancias? ¿Cuál es la diferencia entre las trayectorias obtenidas cuando se emplean distancias distintas?

Búsqueda en profundidad

En los algoritmos de búsqueda en profundidad, los nodos son visitados hasta la máxima profundidad posible (hasta alcanzar un nodo que no tiene vecinos); dicho nodo es removido del grafo y se continúa la búsqueda desde el próximo vecino del nodo anterior (*backtracking*).

El inconveniente principal de este tipo de algoritmos es que los nodos son visitados múltiples veces (tantas como nodos vecinos tengan). Sin embargo, este comportamiento puede reducirse empleando implementaciones eficientes (algoritmos de ramificación y poda, por ejemplo).

La principal ventaja es la complejidad espacial del algoritmo: en todo momento, sólo se requiere almacenar los costos de un único camino que lleva desde el nodo inicial hasta el nodo actualmente explorado. Todos los demás costos pueden ser eliminados, pues no volverán a ser requeridos por el algoritmo.

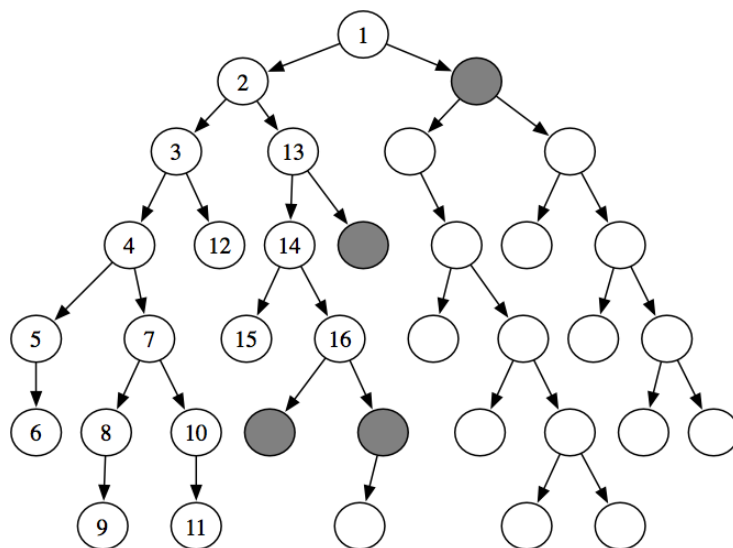


Figura 1.18: Orden de búsqueda en profundidad.

Algoritmo de Dijkstra

El algoritmo de Dijkstra es similar al algoritmo de búsqueda en anchura, sólo que los valores de distancia pueden ser continuos y los vecinos de los nodos “expandidos” se almacenan en una lista ordenada por distancias. A cada paso del algoritmo, el nodo de la lista con el costo más bajo es “expandido” y sus vecinos son introducidos a la lista ordenada o, si ya se encuentran en ella, sus valores son actualizados.

Cada nodo es visitado una sola vez. El costo principal del algoritmo está en el mantenimiento de la lista ordenada, que requiere búsqueda y reordenamiento para poder insertar los nuevos nodos vecinos considerados. Asimismo, requiere conservar todos los costos generados para poder realizar la reconstrucción de la trayectoria.

Algoritmo A*

El algoritmo de Dijkstra requiere que todos los nodos del medio ambiente sean “expandidos”, pues no incluye ningún criterio que permita distinguir cuál de los nodos más cercanos al nodo inicial tiene más “esperanzas” de pertenecer a la solución óptima.

El algoritmo A* (“A estrella”) es la versión heurística del algoritmo de Dijkstra, es decir, incluye un criterio (relativamente arbitrario) que permite guiar el proceso de búsqueda para tratar de “expandir” el menor número posible de nodos, reduciendo el costo computacional total del algoritmo.

Puesto de manera simple, el algoritmo A* evalúa para cada nuevo vecino de la lista ordenada, además de la distancia actual desde el nodo inicial, una estimación de cuánta distancia falta aún hasta el nodo final. El orden de la lista se mantiene utilizando el costo total esperado, lo que permite efectivamente procesar prioritariamente los nodos más prometedores de pertenecer a la solución óptima.

El algoritmo A* es óptimo, y su complejidad computacional depende totalmente de la estimación empleada para calcular la distancia hacia el objetivo. Además, como el espacio libre no es completamente analizado, cada nueva solitud de planificación requiere que se recalcule todo nuevamente.

Tarea: Si durante su viaje a la posición final el robot detectara cambios en el mapa (un obstáculo en el camino o un nuevo paso abierto), requeriría volver a utilizar el algoritmo A* desde la posición actual. Una variante interesante del algoritmo A* es el algoritmo D*, que permite reutilizar los cálculos realizados por el algoritmo A* en la etapa anterior, recalculado únicamente aquellas celdas que son directamente afectadas por el cambio detectado en el medio ambiente. Explicar el funcionamiento del algoritmo D*. Documentar un artículo científico que emplee el algoritmo D* para la planificación de trayectorias. ¿Cuáles son las consideraciones del algoritmo?

1.6.4. Planificación basada en funciones de potencial artificial

El campo de potencial artificial es una función que asigna a cada punto (configuración) del espacio libre del medio ambiente, un vector (o un gradiente) que dirige el robot hacia el punto destino. Este enfoque **trabaja directamente con la representación continua** del medio ambiente y no requiere de una representación discreta (grafo) del mismo.

La idea principal de este enfoque es tratar al robot como una partícula bajo el influjo de un campo de potencial, similar al experimentado por una carga eléctrica dentro de un campo electrostático. La posición final deseada genera a su alrededor un campo de potencial atractivo, tomando un valor mínimo (global) de la función de potencial. Los obstáculos, por su parte, generan un campo repulsivo (grandes valores de la función potencial) alrededor de ellos, cuya influencia disminuye con la distancia. La superposición de todos estos campos genera una superficie cuyo gradiente dirige al robot de forma continua hacia la posición final evitando los obstáculos, de la misma manera que lo haría una canica que rueda colina abajo.

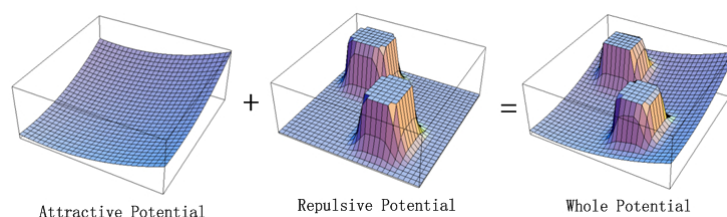


Figura 1.19: Campo de potencial artificial.

Usando este enfoque es posible también incluir obstáculos que no estaban presentes en la representación inicial del medio ambiente, lo que permite adaptar el esquema de planificación a medios ambientes desconocidos, actualizando con la información proveniente de los sensores.

Resulta importante resaltar que este enfoque es más que un planificador de trayectorias, y puede considerarse una **verdadera ley de control** para el robot móvil, pues indica, de forma continua y para cada instante de tiempo, cuáles deben ser las acciones motrices que deben aplicarse.

Desafortunadamente, este enfoque tiene un inconveniente muy importante: bajo ciertas configuraciones del medio ambiente se pueden generar mínimos locales de la función de potencial, de tal manera que el robot queda “atrapado” en alguna configuración intermedia, sin posibilidades de llegar a la posición final deseada.

Para resolver el inconveniente anterior, se han propuesto diversas soluciones:

cuando se detecta un mínimo local, introducir en esa posición un obstáculo artificial que obligue al robot a alejarse de ese lugar, o bien generar movimientos aleatorios que permitan salir del bloqueo; utilizar campos de potencial no lineales, de manera que su superposición no genere mínimos locales (por ejemplo, los campos de flujo empleados en hidrodinámica); imponer ciertas formas de campo de potencial artificial, denominados *esquemas motrices*, que dirijan al robot alrededor de los obstáculos; por sólo mencionar algunas.

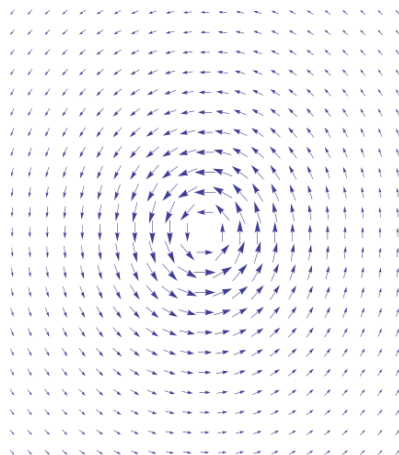


Figura 1.20: Ejemplo de un esquema motriz para rodear obstáculos.

Tarea: Documentar dos artículos científicos que empleen los campos de potencial artificial para la planificación de trayectorias. ¿Cómo son y bajo qué condiciones aparecen los mínimos locales? ¿Qué técnica emplean para resolverlos?

1.6.5. Conclusión

Dado que la planificación global requiere del manejo de una representación en memoria del medio ambiente, este proceso corresponde completamente a la capa deliberativa de la arquitectura de control del robot móvil.

La planificación puede hacerse de dos formas distintas: empleando la representación continua del medio ambiente para construir una función de potencial (o un campo vectorial) cuyo gradiente dirija al robot hasta la posición final, o construir una representación discreta (grafo) en la cual se aplican algoritmos de búsqueda para encontrar una secuencia de sub-metas que lleven al robot hasta la posición deseada.

Todas las estrategias vistas en este apartado se realizan utilizando un espa-

cio de búsqueda distinto al espacio de configuraciones reales del robot móvil, con el propósito de simplificar el proceso de planificación. Esto implica que el proceso encargado de la ejecución del plan obtenido deberá ser capaz de resolver situaciones que no han sido consideradas al momento de la elaboración del mismo.

1.7. Incluyendo la mecánica en la navegación. Planificación local.

Para poder construir la trayectoria, los planificadores globales trabajan con una versión simplificada del espacio de configuraciones, bajo los siguientes supuestos:

1. El robot es un único punto, y por lo general su orientación es irrelevante. El espacio de búsqueda es bidimensional para el trabajo en el plano y tridimensional para el trabajo en el espacio.
2. El medio ambiente es conocido y totalmente estático. Por lo tanto, no existen imprevistos en la configuración de los obstáculos, y no se requiere replanificar la trayectoria.
3. El robot es holonómico, es decir, que puede desplazarse en cualquier dirección en todo instante, sin necesitar reorientarse para hacerlo.

En la práctica, ninguno de los puntos anteriores es cierto⁶, por lo que la trayectoria obtenida no es realizable directamente por el robot móvil. La trayectoria planificada debe tomarse como una secuencia de sub-metas **factibles**, de las cuales se sabe con certeza que entre una sub-meta y la siguiente existe una trayectoria relativamente simple y sin obstáculos.

Las sub-metas son entregadas una a una a un proceso de planificación local, el cual toma en cuenta la geometría real del robot (para garantizar la no colisión), la lectura de los sensores (para incluir información actualizada del medio ambiente) y la mecánica del robot móvil (para generar trayectorias que sean realmente realizables). Algunos autores llaman al planificador local, proceso de navegación con evasión de obstáculos (*obstacle-avoidance*).

Es muy importante resaltar que el planificador local no requiere conocer el mapa global del medio ambiente, sino únicamente la posición que se desea

⁶Excepto por el punto 3, pero únicamente para robots móviles con estructuras mecánicas muy particulares, como las *swedish wheels*, por ejemplo.

alcanzar (objetivo) y la posición actual del robot, por lo que se considera que **el problema de localización está resuelto**⁷.

Por su necesidad de acceso continuo a la lectura de los sensores y a la potencia de los motores, el planificador local se considera una tarea del tipo reactivo, y en una estructura de control se coloca entre las tareas de bajo nivel.

1.7.1. Algoritmo Bug

Uno de los planificadores locales más sencillos es el algoritmo Bug, el cual requiere únicamente conocer la información proveniente de un sensor de contacto. Básicamente, el robot emplea dos comportamientos complementarios: el primero le permite desplazarse directamente hacia la posición final (*goal-reaching*), mientras que el otro le permite seguir el contorno de los obstáculos (*wall-following*). Las condiciones en las que el robot debe cambiar de uno a otro comportamiento definen las distintas variantes de este algoritmo.

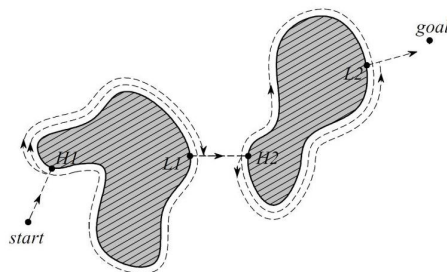


Figura 1.21: Ejemplo de navegación empleando planificación local con el algoritmo Bug1.

Tarea: Documentar el pseudocódigo de los algoritmos Bug1 y Bug2. Una tercera variante, denominada *TangentBug*, utiliza sensores de distancia en lugar de sensores de contacto. Documentar un artículo científico que emplee el algoritmo *TangentBug*. ¿Podría describir, en forma de pseudocódigo, el algoritmo necesario para seguir el perfil de un muro empleando sensores de distancia?

1.7.2. Histograma del campo vectorial (VFH)

Uno de los principales inconvenientes del uso de la última lectura de sensores para la planificación local es que la presencia de ruido e incertidumbre

⁷Empleando ya sea técnicas de posicionamiento relativo (como la odometría) o técnicas de posicionamiento global (como el GPS).

puede causar comportamientos indeseables del robot. Para superarlo, se pueden emplear técnicas basadas en un histórico de las últimas n lecturas de los sensores.

Una de estas técnicas es el histograma de campo vectorial (*Vector Field Histogram*), que consiste en utilizar una pequeña rejilla de ocupación local, centrada en el robot, que se actualiza con las lecturas de los sensores (por ejemplo, marcando en cada celda el número de veces que un objeto ha sido detectado ahí).

A continuación, se construye un histograma polar alrededor del robot, dividido en m sectores. Tras aplicar un cierto umbral, es posible detectar los espacios libres alrededor del robot en los cuales hay suficiente distancia para la navegación segura, y elegir aquella ventana que se encuentre más cercana de la dirección en la que se encuentra la posición objetivo deseada.

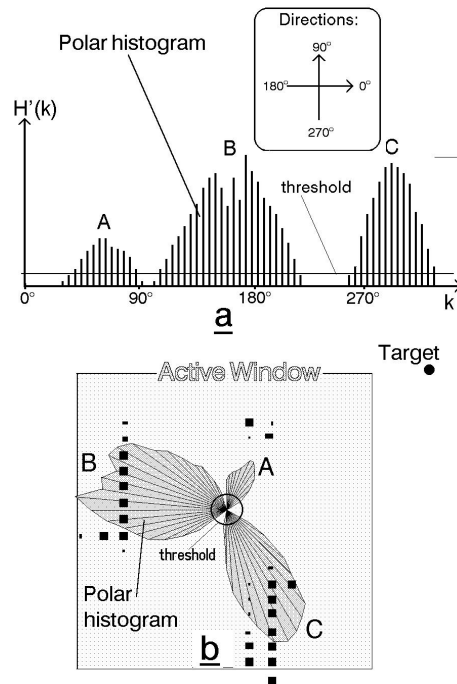


Figura 1.22: Ejemplo de navegación empleando planificación local con el algoritmo VFH.

Tarea: Una variante del VFH, es el algoritmo VFH+, que incluye en la creación del histograma polar una simplificación del modelo cinemático del robot móvil como forma de tomar en cuenta la condición no holonómica del vehículo. Explicar el funcionamiento del

algoritmo VFH+. ¿Cuál es el principio de actualización de la rejilla de ocupación local, particularmente al cambiar el robot de lugar?

1.7.3. Espacio de velocidades

En estos métodos, basados en el principio propuesto por Simmons⁸, está principalmente enfocado a robots móviles de tipo diferencial, es decir, aquellos vehículos que pueden desplazarse sobre trayectorias circulares de radio variable, controlando independientemente la velocidad de traslación v y la velocidad de rotación ω .

Los obstáculos son convertidos en restricciones en el espacio (v, ω) , ya sea por simples consideraciones de distancia sobre el arco de movimiento del robot, como en el método de ventana dinámica⁹ o empleando un modelo de amortiguamiento de velocidades¹⁰. El resultado es un espacio de velocidades admisibles o realizables por el robot, para evitar las colisiones.

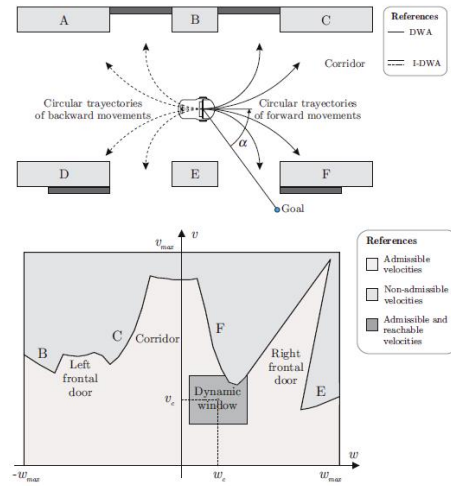


Figura 1.23: Ejemplo de representación en el espacio de velocidades.

Tarea: Tras analizar el método propuesto por Fox et al., así como método propuesto por Ramírez y Zegloul, ¿cuáles considera que sean las principales diferencias entre ambos métodos? ¿Cuáles son las ventajas y desventajas de cada uno de ellos?

⁸R. Simmons, "The curvature velocity method for local obstacle avoidance", 1991.

⁹D. Fox et al., "The dynamic window approach to collision avoidance", 1997.

¹⁰G. Ramírez, "Collision-free path planning for nonholonomic mobile robots using a new obstacle representation in the velocity space", 2001.

1.7.4. Campos de potencial artificial

Al igual que en el método de planificación global, los obstáculos son modelados como fuerzas repulsivas sobre el robot, pero a diferencia de aquél, en el caso de la planificación local son las lecturas de los sensores de distancia y la ubicación física del sensor la que determina la intensidad y dirección de la fuerza repulsiva.

La acción conjunta de todas las fuerzas generadas por los sensores, junto con la fuerza que atrae al robot a la posición final, es la que determina la dirección en que se moverá el robot. En esta etapa se incluye la mecánica del robot (restricciones no holonómicas) ya sea por proyección en el plano de movimiento del robot, ya sea por descomposición de movimientos.

Dado que se utiliza únicamente la información de los sensores, este método no requiere información del mapa en memoria. Sin embargo, es posible que un único objeto sea detectado por varios sensores a la vez, por lo que el objeto ejerce más de una fuerza repulsiva sobre el robot.

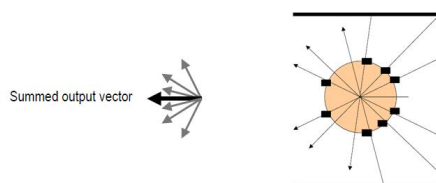


Figura 1.24: Ejemplo de campo de potencial artificial local.

1.8. Conclusiones

En la literatura clásica de robótica móvil, las estrategias de planificación local de trayectorias y las de planificación global son abordadas como dos enfoques disjuntos, con sus ventajas y desventajas opuestas.

Las principales críticas de las estrategias globales es que la planificación no se realiza en el espacio de configuraciones del robot, por lo que las trayectorias obtenidas pueden no ser realizables, ya que no toman en cuenta las restricciones mecánicas propias del vehículo. Además, es obligatorio contar con una información completa del medio ambiente para obtener soluciones que puedan ser empleadas adecuadamente.

En cuanto a las estrategias locales de planificación de trayectorias, su principal crítica es que no cumplen con criterios de optimalidad ni de factibilidad (por ello, se puede decir que incluso no realizan planificación de verdad): no siempre

encuentran una solución y cuando lo hacen, nada garantiza que sea una buena solución.

Sin embargo, en un esquema de robótica inteligente, cada uno de estos enfoques tiene su lugar y es posible explotar sus fortalezas: el planificador global emplea la información conocida del medio ambiente para generar una secuencia de trayectorias simples *libres de obstáculos*, mientras que el planificador local integra las restricciones mecánicas del vehículo para obtener trayectorias físicamente realizables por el robot, en las que sólo habrá que hacer cambios menores para evadir obstáculos no incluidos en el mapa global.

Tarea: Documentar dos artículos científicos que realicen planificación de trayectorias en dos etapas: global y local. ¿Qué tipo de arquitectura de control emplean? ¿Cuáles son los algoritmos que utilizan? ¿De qué manera incluyen las restricciones mecánicas del vehículo dentro de la planificación?

Capítulo 2

Robótica probabilista

2.1. ¿Por qué necesitamos emplear probabilidad en Robótica?

Un robot móvil es, básicamente, un agente cuyo medio ambiente es el mundo físico. Como todo agente, el robot está equipado para realizar tres importantes tareas:

- **Planificar.** Para cumplir su propósito, el robot debe poseer los mecanismos (algoritmos) que le permitan determinar adecuadamente cuál es la acción siguiente a realizar. Para ello debe contar, además de la descripción del medio ambiente (mapa), con un **modelo de estado** (o simplemente *estado*) que contiene una descripción completa de su situación interna (respecto a sí mismo y a la tarea a realizar) como de su situación externa (respecto al mundo físico). De aquí en adelante, utilizaremos la variable x para representar el estado actual del robot móvil.
- **Sentir el medio ambiente.** El medio ambiente genera información que puede ser captada por el robot, empleando elementos denominados sensores. Dicha información debe ser interpretada para obtener datos útiles para el propósito del robot, como por ejemplo, obtener a partir de la imagen de la cámara la posición espacial de un objetivo en particular. Este proceso, basado en un **modelo de percepción**, requiere de una función de observación (abstracción) que describa la relación entre el medio ambiente y la información obtenida por el sensor utilizado. Tenemos entonces:

$$z = h(x)$$

donde z es la información *observada* desde el sensor y $h()$ es la función de observación. Para la planificación de tareas, requerimos establecer el

modelo inverso del sensor:

$$x = h^{-1}(z)$$

que permite conocer el estado actual del robot a partir de las lecturas de los sensores.

- **Actuar sobre el medio ambiente.** Finalmente, el robot posee actuadores que le permiten realizar físicamente la tarea confiada. Se requiere entonces de un **modelo de acción** que describa la relación entre la acción realizada y el medio ambiente, pero sobre todo, conocer *cuál es el resultado de dicha acción sobre el propio estado del robot*. Si denominamos u a la acción realizada por el robot y x' a su estado siguiente, requerimos establecer el modelo de acción $g()$, llamado también *función de transición*, descrito de la siguiente manera:

$$x' = g(x, u)$$

En la realidad existen diversas condiciones que no permiten establecer de manera precisa el modelo inverso de los sensores y las funciones de transición de los actuadores:

- Las observaciones obtenidas desde los sensores son generalmente ruidosas, parciales e incompletas.
- Existe incertidumbre sobre el resultado exacto de las acciones realizadas.
- Los modelos deterministas empleados para describir los sensores y actuadores son parcialmente incorrectos e incompletos.
- El mundo es dinámico y, para efectos prácticos, totalmente estocástico.

Para todas estas condiciones, los modelos probabilistas permiten *aprovechar el conocimiento previo* que se tiene sobre el estado del robot y sobre la ejecución de la tarea, para disminuir el efecto del ruido y de la incertidumbre sobre el *conocimiento actual* del mismo. A este proceso de integración temporal de la información se le denominada **filtrado** o **estimación**.

2.2. Conceptos básicos de probabilidad

Usaremos la notación $P(A)$ para indicar la probabilidad de que una proposición lógica A sea verdadera. La Teoría de Probabilidad se basa en los siguientes axiomas:

1. $0 \leq P(A) \leq 1$

2. $P(\Omega) = 1$, $P(\emptyset) = 0$, Ω el conjunto Universo.
3. $P(A \cup B) = P(A) + P(B) - P(A \cap B)$

2.2.1. Variables aleatorias discretas

Una variable aleatoria X es una función real *medible* definida sobre un espacio de muestra Ω , asociada a un experimento estocástico o aleatorio.

$$X : \Omega \rightarrow \mathbb{R}$$

Si el conjunto de muestra $\Omega = \{x_1, x_2, \dots, x_n\}$ es un conjunto finito de n resultados posibles del experimento, se dice que la variable X es una variable aleatoria discreta.

La función $P(X = x_a)$, o simplemente $P(x_a)$, expresa la probabilidad de que la variable aleatoria discreta X tome el valor x_a . Dicha función es una **distribución de probabilidad discreta** que recibe el nombre de **función de cuantía**.¹

La función de cuantía cumple con la siguiente propiedad:

$$\sum_{x \in \Omega} P(X = x) = 1$$

2.2.2. Variables aleatorias continuas

En el caso de las variables aleatorias continuas, el espacio de muestra es el conjunto de los números reales, es decir $\Omega = \mathbb{R}$, que denota que el experimento estocástico tiene infinitos resultados posibles.

La función de distribución de probabilidad continua $P(X = x_a)$, o simplemente $P(x_a)$, se denomina **función de densidad de probabilidad**,² y cumple con la siguiente propiedad:

$$\int_{-\infty}^{\infty} P(x) dx = 1$$

¹Probability Mass Function, en inglés.

²Probability Density Function, en inglés. De aquí en adelante, la integral aparecerá como indefinida, pero en realidad está definida en el rango $(-\infty, \infty)$.

2.2.3. Distribución conjunta y probabilidad condicional

La probabilidad de que dos eventos distintos X y Y ocurran (sean verdad) simultáneamente se denomina **distribución conjunta de X y Y** :

$$P(X = x \cap Y = y) = P(x, y)$$

Si los eventos X y Y son independientes, entonces se tiene que:

$$P(x, y) = P(x)P(y)$$

Generalmente, el resultado de un evento Y proporciona información³ sobre el posible resultado de un evento X . La **probabilidad condicional** mide la probabilidad del evento X , cuando se sabe (se asume, se presume, se afirma o se mide) que el evento Y es verdad. Por definición, la probabilidad condicional expresa la probabilidad relativa de $X \cap Y$ sobre el espacio de muestra de Y :

$$P(x | y) = \frac{P(x, y)}{P(y)}$$

Lo anterior puede reescribirse como un axioma de probabilidad: La probabilidad de que ocurra x **dado** y está dada por:

$$P(x, y) = P(x | y)P(y)$$

Es importante resaltar que debe tenerse cuidado con la **falacia condicional**, que supone que $P(x | y) = P(y | x)$, lo cual es generalmente falso.

Si X y Y son independientes, entonces: $P(x | y) = P(x)$

Dos eventos X y Y son **condicionalmente independientes** dado un evento Z , si y sólo si:

$$P(x, y | z) = P(x | z)P(y | z)$$

De aquí, se puede deducir el **Teorema de probabilidad total** que, en el caso discreto, establece que:

$$P(x) = \sum_y P(x, y) = \sum_y P(x | y)P(y)$$

Para el caso continuo, el teorema de probabilidad total queda de la siguiente manera:

$$P(x) = \int P(x, y)dy = \int P(x | y)P(y)dy$$

³Los eventos X y Y pueden o no ser independientes.

2.2.4. Teorema de Bayes

A partir de la probabilidad condicional, tenemos:

$$P(x, y) = P(x | y)P(y) = P(y | x)P(x)$$

El teorema de Bayes se expresa entonces como:

$$P(x | y) = \frac{P(y | x)P(x)}{P(y)}$$

donde la expresión $P(y | x)$ se denomina *verosimilitud*, $P(x)$ es el *conocimiento previo* y $P(y)$ es la *evidencia*.

La versión extendida de este teorema se escribe como:

$$P(x | y, z) = \frac{P(y | x, z)P(x | z)}{P(y | z)}$$

2.2.5. Esperanza matemática de una variable aleatoria

La **esperanza matemática**, **valor esperado** o **media poblacional** de una variable aleatoria X , es el promedio ponderado $E[X]$ de todos los posibles resultados que dicha variable aleatoria X puede tomar.

Para el caso discreto:

$$E[X] = \sum_i x_i P(x_i)$$

Para el caso continuo:

$$E[X] = \int x P(x) dx$$

La esperanza es un operador lineal, es decir, para $a, b \in \mathbb{R}$:

$$E[aX + b] = aE[X] + b$$

La **varianza**⁴ $Cov[X]$ mide el grado de dispersión de una variable aleatoria X respecto a su propia media poblacional $E[X]$:

$$Cov[X] = E[(X - E[X])^2]$$

⁴En la mayoría de los documentos de robótica en inglés que emplean este valor, utilizan incorrectamente el término *covariance*, $Cov[X]$, que indica el grado de variación conjunta entre **dos** variables aleatorias distintas, mientras que la *varianza*, $Var[X]$ mide el grado de dispersión interna de **una única** variable. Mantendremos la notación $Cov[X]$ para evitar confusiones con la literatura científica, ya que más adelante la variable X es un vector y el término se aplica correctamente, aunque la fórmula es ligeramente distinta.

Aplicando las propiedades lineales de la esperanza matemática obtenemos (no olvidar que $E[X]$ es una constante):

$$\begin{aligned} Cov[X] &= E[(X - E[X])^2] \\ &= E[X^2 - 2XE[X] + E[X]^2] \\ &= E[X^2] - 2E[X]E[X] + E[X]^2 \\ &= E[X^2] - E[X]^2 \end{aligned}$$

2.2.6. Estimación a partir de una serie de datos

De manera sencilla, si se tiene una serie de muestras $x_1, x_2, x_3, \dots, x_n \in \mathbb{R}^d$ de una variable aleatoria X (vector de d componentes reales), su **media muestral** o **media estadística** \bar{x}_n se obtiene de la manera siguiente:

$$\bar{x}_n = \frac{1}{n} \sum_i x_i$$

y para obtener su **varianza muestral** o **varianza estadística** s_n^2 se emplea la siguiente ecuación:

$$s_n^2 = \frac{1}{n-1} \sum_i (x_i - \bar{x}_n)(x_i - \bar{x}_n)^\top$$

2.3. Estimación bayesiana de estado

El problema de la estimación de estado busca establecer una estimación del estado del robot y su mundo x , a partir de las observaciones z hechas a través de los sensores, así como de las acciones de control (y la odometría) u que se realizan a través de los actuadores. Dado que toda esta información posee incertidumbre, estas variables ya no son deterministas, sino aleatorias, y lo que se pretende es establecer las relaciones causales entre ellas, es decir:

$$p(x \mid z)$$

que indica cómo se ve el mundo a partir de los sensores, así como la relación:

$$p(x' \mid u, x)$$

que indica cómo afecta al mundo las acciones realizadas.

Analicemos un poco la expresión $p(z \mid x)$. Esta expresión se denomina **relación causal**, ya muestra la relación entre el mundo y la lectura del sensor, a partir del **modelo físico** del sensor.

Con esta información, es posible obtener la **relación de diagnóstico** $p(x | z)$ empleando el teorema de Bayes:

$$p(x | z) = \frac{p(z | x)p(x)}{p(z)}$$

donde el valor de $p(z)$ se obtiene empleando la fórmula de probabilidad total⁵:

$$p(z) = \sum_x p(z | x)p(x)$$

2.3.1. Ejemplo

Consideremos un VANT que busca su estación de aterrizaje, que está marcada por una luz brillante. Para ello, se emplea una variable de estado booleana $X \in \{home, \neg home\}$ que indica si el VANT se encuentra justo encima de la estación o no. Para detectar la luz brillante, el sensor provee información binaria $Z \in \{bright, \neg bright\}$.

El modelo del sensor $p(Z | X)$ es el siguiente:

$$\begin{aligned} p(bright | home) &= 0,6 \\ p(bright | \neg home) &= 0,3 \end{aligned}$$

Supongamos que, inicialmente, no tenemos conocimiento *a priori* de la ubicación del VANT, por lo que tomamos $p(X = home) = 0,5$, y que el sensor ve luz, es decir, $Z = bright$. ¿Cuál es la probabilidad de que el robot se encuentre encima de la estación de aterrizaje?

En este caso, se nos pide determinar:

$$p(home | bright) = \frac{p(bright | home)p(home)}{p(bright)}$$

Desconocemos únicamente $p(bright)$, pero podemos obtenerla empleando la fórmula de probabilidad total a partir del modelo del sensor:

$$\begin{aligned} p(bright) &= p(bright | home)p(home) + p(bright | \neg home)p(\neg home) \\ &= 0,6 * 0,5 + 0,3 * 0,5 = 0,45 \end{aligned}$$

⁵Veremos más adelante que, en ocasiones, no es posible determinar $p(z)$, por lo que se maneja un coeficiente de proporcionalidad $\eta = 1/p(z)$ en su lugar.

Podemos obtener la nueva estimación de estado:

$$\begin{aligned} p(home | bright) &= \frac{p(bright | home)p(home)}{p(bright)} \\ &= \frac{0,6 * 0,5}{0,45} = 0,67 \end{aligned}$$

De esta manera, la información $Z = bright$ proporcionada por el sensor aumenta la certidumbre de que el VANT se encuentra en la posición de aterrizaje de $p(home) = 0,5$ a $p(home) = 0,67$.

***Tarea:** En el ejemplo anterior, el modelo del sensor no está completo. ¿Por qué las probabilidades dadas $p(bright | home)$ y $p(bright | \neg home)$ no suman 1? ¿Cuál es el modelo completo del sensor? ¿Cuál es el valor de $p(\neg home | bright)$?*

2.4. Actualizaciones bayesianas recursivas

Supongamos ahora que el robot cuenta con una serie de lecturas de los sensores z_1, z_2, \dots, z_n , que provienen del mismo sensor o de diferentes sensores. ¿Cómo debemos integrar toda esta información para estimar el nuevo estado x , es decir, cómo obtener $p(x | z_1, z_2, \dots, z_n)$?

Empleando la forma extendida del teorema de Bayes tenemos que:

$$p(x | z_1, z_2, \dots, z_n) = \frac{p(z_n | x, z_1, z_2, \dots, z_{n-1})p(x | z_1, z_2, \dots, z_{n-1})}{p(z_n | z_1, z_2, \dots, z_{n-1})}$$

Si admitimos que nuestra variable de estado X es **completa**, es decir, contiene toda la información causal necesaria para determinar cualquier otra variable a partir de ella, entonces podemos establecer que:

$$p(z_n | x, z_1, z_2, \dots, z_{n-1}) = p(z_n | x)$$

es decir, todas las lecturas z_1, z_2, \dots, z_n son independientes entre sí cuando se conoce el estado x . A esta propiedad se le denomina **propiedad de Markov** e indica que el proceso aleatorio no tiene memoria, por lo que el futuro de una variable es independiente de la historia de dicha variable.

$$\begin{aligned}
\Rightarrow p(x \mid z_1, z_2, \dots, z_n) &= \frac{p(z_n \mid x, z_1, z_2, \dots, z_{n-1})p(x \mid z_1, z_2, \dots, z_{n-1})}{p(z_n \mid z_1, z_2, \dots, z_{n-1})} \\
&= \frac{p(z_n \mid x)p(x \mid z_1, z_2, \dots, z_{n-1})}{p(z_n \mid z_1, z_2, \dots, z_{n-1})} \\
&= \eta p(z_n \mid x)p(x \mid z_1, z_2, \dots, z_{n-1}) \\
&= \eta_{1:n} \prod_{i=1:n} p(z_i \mid x)p(x)
\end{aligned}$$

donde η es un factor de normalización sobre la distribución completa de probabilidad.

2.4.1. Ejemplo

Tomando el mismo VANT del ejemplo anterior, consideremos que cuenta además con una cámara para buscar un marcador visual asociado a la zona de aterrizaje, de tal manera que obtiene la información binaria

$$Z_2 \in \{marker, \neg marker\}$$

de la manera siguiente:

$$\begin{aligned}
p(marker \mid home) &= 0,8 \\
p(marker \mid \neg home) &= 0,1
\end{aligned}$$

Si el VANT percibe luz brillante ($z_1 = bright$), pero no ve el marcador ($z_2 = \neg marker$), ¿cuál es la probabilidad de que el VANT se encuentre efectivamente sobre la zona de aterrizaje?

A partir del ejemplo anterior, ya habíamos determinado que $p(home \mid bright) = 0,67$, información que debemos actualizar con el nuevo dato $z_2 = \neg marker$.

Aplicando el teorema de Bayes tenemos:

$$p(home \mid bright, \neg marker) = \frac{p(\neg marker \mid home, bright)p(home \mid bright)}{p(\neg marker \mid bright)}$$

Por la propiedad de Markov, tenemos que las lecturas de los sensores son independientes, por lo que la lectura de la cámara depende únicamente de X :

$$\begin{aligned}
p(\neg marker \mid home, bright) &= p(\neg marker \mid home) \\
&= 1 - p(marker \mid home) \\
&= 1 - 0,8 = 0,2
\end{aligned}$$

La probabilidad $p(\neg marker \mid bright)$ podemos determinarla por la fórmula de probabilidad total:

$$\begin{aligned} p(\neg marker \mid bright) &= p(\neg marker \mid home)p(home \mid bright) \\ &\quad + p(\neg marker \mid \neg home)p(\neg home \mid bright) \\ &= 0,2 * 0,67 + 0,9 * 0,33 \\ &= 0,431 \end{aligned}$$

Finalmente:

$$\begin{aligned} p(home \mid bright, \neg marker) &= \frac{p(\neg marker \mid home)p(home \mid bright)}{p(\neg marker \mid bright)} \\ &= \frac{0,2 * 0,67}{0,431} \\ &= 0,31 \end{aligned}$$

Así, la información $Z_2 = \neg marker$ modifica la certidumbre $p(X = home)$ del valor original de 0,5 a un valor final de 0,31, aún cuando el primer sensor ve una luz brillante.

Tarea: En este ejemplo, describir cómo se obtienen los valores:

$$p(\neg marker \mid \neg home) = 0,9$$

$$p(\neg home \mid bright) = 0,33$$

Si ambos sensores confirman la presencia de la zona de aterrizaje ($Z_1 = bright$ y $Z_2 = marker$), ¿cuál sería la probabilidad de que el VANT se encuentre realmente en la zona de aterrizaje? ¿Cuál cuando ($Z_1 = \neg bright$ y $Z_2 = marker$)? ¿Y si ($Z_1 = \neg bright$ y $Z_2 = \neg marker$)? ¿Cómo se relacionan entre sí todos los resultados anteriores?

2.5. Modelo de sensores

En general, un sensor no provee información binaria como en los ejemplos anteriores, sino una información continua dentro de un determinado rango. Por ello, es necesario establecer un modelo que permita determinar, de acuerdo a la lectura entregada por el sensor, la función de distribución de probabilidad que pueda usarse para actualizar el estado x del robot. A este modelo se le denomina **modelo inverso del sensor**.

La manera de obtener el modelo inverso de un sensor puede ser muy variada:

1. Método analítico. Se obtiene a través de la comprensión completa del funcionamiento físico del sensor y de los modelos matemáticos que rigen el fenómeno de transducción empleado.
2. Método subjetivo. Basado en la experiencia del diseñador del robot.
3. Método estadístico. A través de experimentación exhaustiva y repetitiva, se obtiene el mapa completo de las probabilidades de todos los resultados posibles entregados por el sensor.

2.5.1. Ejemplo: el sensor ultrasónico

El modelo básico de un sensor ultrasónico está basado en regiones. El sensor sólo es capaz de entregar información dentro del cono de detección, delimitado por un alcance máximo R y un ángulo de apertura β . Cuando el sensor detecta un objeto (a una distancia inferior a R) dentro del campo de visión, no es posible determinar la dirección en la que se encuentra el objeto, por lo que el campo de visión queda dividido en tres regiones:

- Región I. La zona en la que probablemente se encuentra el objeto realmente. Se modela como una distribución gaussiana centrada en la lectura y una desviación proporcional a la precisión del sensor y al ruido (probabilidades superiores a 0,5).
- Región II. La zona en que probablemente no hay obstáculos. Sin embargo, es posible que existan y que el sensor haya sido incapaz de percibirlos, por lo que se modela como una función decreciente (probabilidades inferiores a 0,5).
- Región III. La zona detrás del obstáculo detectado, por lo que no es posible inferir ninguna información (probabilidades iguales a 0,5).
- En todas las regiones, se agrega una probabilidad uniforme de detectar un objeto, para reflejar las lecturas aleatorias que pueden ser generadas por el sensor.

***Tarea:** Describir brevemente cómo se construye el modelo de un sensor ultrasónico y cómo se le emplea para actualizar el conocimiento del mundo.*

2.6. Modelo de acciones

Las acciones del robot sobre el medio ambiente también son inciertas. Por regla general, y en contraste con las lecturas de los sensores, las acciones del

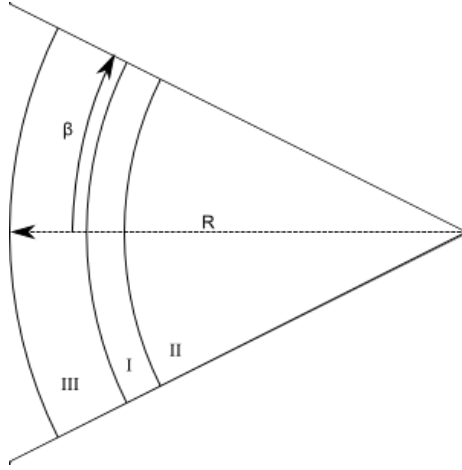


Figura 2.1: Modelo por regiones de un sensor ultrasónico.

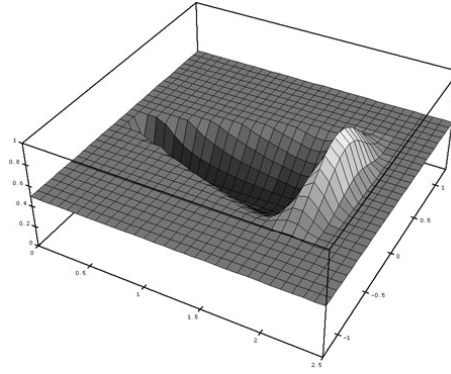


Figura 2.2: Función de distribución de probabilidad para un sensor ultrasónico.

robot tienden a aumentar la incertidumbre de la variable de estado x .

Para incorporar el efecto x' de la acción u sobre la **probabilidad de estado actual** o **creencia**⁶ x , empleamos la función de distribución de probabilidad condicional $p(x' \mid u, x)$, que especifica la probabilidad de que, estando en el estado x , aplicar la acción u lleve al robot al estado x' .

El término $p(x' \mid u, x)$ se denomina también **probabilidad de transición** y, en el caso de un conjunto finito de estados posibles x , puede modelarse por medio de una **matriz de transición**.

Para integrar el resultado de las acciones sobre el estado del robot $p(x' \mid u)$,

⁶ *Belief*, en inglés.

empleamos la fórmula de probabilidad total:

$$p(x' | u) = \sum_i p(x' | u, x_i) p(x_i)$$

2.6.1. Ejemplo

Retomemos el VANT de los ejemplos anteriores y consideremos la variable de estado $X \in \{ground, air\}$. El comportamiento del VANT ante un comando *takeoff* está regido por las siguientes probabilidades de transición:

$$\begin{aligned} p(ground | takeoff, ground) &= 0,1 \\ p(ground | takeoff, air) &= 0,01 \\ p(air | takeoff, ground) &= 0,9 \\ p(air | takeoff, air) &= 0,99 \end{aligned}$$

Si inicialmente el robot se encuentra en tierra ($x_0 = ground$), ¿cuál es la probabilidad de que siga en tierra después de recibir un comando $u = takeoff$?

De los datos anteriores, sabemos que $p(ground) = 1,0$ (obviamente, $p(air) = 0$). Aplicando la fórmula de probabilidad total tenemos:

$$\begin{aligned} p(x' = ground | takeoff) &= \sum_i p(ground | takeoff, x_i) p(x_i) \\ &= p(ground | takeoff, ground) p(ground) \\ &\quad + p(ground | takeoff, air) p(air) \\ &= 0,1 * 1,0 + 0,01 * 0,0 \\ &= 0,1 \end{aligned}$$

Tarea: Para este ejemplo, describir las probabilidades de transición como un diagrama de estados. ¿Cuál sería la probabilidad de que el VANT siga en tierra, tras recibir un segundo comando $u = takeoff$?

2.7. Filtro de Bayes y Procesos de Markov

En sentido estricto, cualquier método que emplee el teorema de Bayes de manera recursiva para obtener una estimación de una función de distribución de probabilidad desconocida, a partir de un modelo matemático y una serie de observaciones, se le denomina **filtro de Bayes**.

Particularmente en robótica, nos interesamos en los procesos estocásticos denominados **procesos de Markov**, es decir, aquellos procesos que cumplen la

propiedad de Markov, en los que la variable aleatoria de estado X se considera completa, de tal manera que el desarrollo futuro del sistema depende únicamente del estado actual y es independiente de la historia del sistema (el cómo se llegó al estado actual no aporta información sobre cómo evolucionará el sistema).

En concreto, un proceso de Markov⁷ se basa en las siguientes hipótesis:

- Las observaciones dependen únicamente del estado actual y son independientes entre sí:

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

- El estado actual depende únicamente del estado anterior y de la acción actual:

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

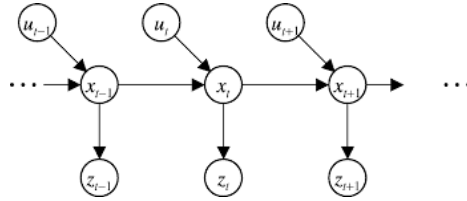


Figura 2.3: Proceso de Markov (en robótica móvil).

La consideración de que la variable aleatoria de estado X es completa implica que el comportamiento del sistema es independiente del ruido, el modelo es completo y que el mundo es estático. Evidentemente, lo anterior no es realista. Sin embargo, los algoritmos basados en procesos de Markov son sorprendentemente robustos en la práctica, lo que justifica su uso actual.

2.7.1. Algoritmo de Bayes

El algoritmo del filtro de Bayes para un proceso de Markov discreto puede plantearse de la siguiente manera. A partir de los siguientes datos iniciales:

- Una secuencia de observaciones y de acciones: $\{u_1, z_1, u_2, z_2, \dots, u_t, z_t\}$
- Un modelo de sensor: $p(z \mid x)$
- Un modelo de acción: $p(x' \mid x, u)$

⁷De hecho, esta definición corresponde a un **proceso discreto de Markov**, porque las transiciones se realizan a instantes discretos en el tiempo, pero la variable de estado X puede ser continua o discreta. Algunos autores emplean equivocadamente el término de **cadena de Markov**, que se aplica únicamente cuando el número de estados posibles del sistema es finito.

- Un conocimiento inicial de la distribución de probabilidad de estado: x_0

determinar la evolución dinámica de la distribución de probabilidad de estado x_i . Cada nuevo estado x_i de la secuencia se denomina **creencia**⁸:

$$Bel(x_t) = p(x_t \mid u_1, z_1, u_2, z_2, \dots, u_t, z_t)$$

Para cada instante de tiempo t , el algoritmo funciona en dos etapas:

1. **Predicción.** Se aplica el modelo de acción sobre el estado anterior del sistema, para obtener una predicción del estado actual:

$$\overline{Bel}(x_t) = \sum_{x_{t-1}} P(x_t \mid x_{t-1}, u_t) Bel(x_{t-1})$$

2. **Corrección.** Se aplica el modelo de sensor sobre la predicción obtenida en el paso anterior, para obtener la estimación sobre el estado actual:

$$Bel(x_t) = \eta P(z_t \mid x_t) \overline{Bel}(x_t)$$

donde η es un factor de normalización que puede calcularse *a posteriori*.

El algoritmo de Bayes funciona también en el caso de un proceso de Markov continuo, simplemente reemplazando la sumatoria por una integral. Asimismo, es importante resaltar que el algoritmo no requiere que las observaciones z_i y las acciones u_i sean sincronas.

Por su relativa sencillez, el algoritmo de Bayes es ampliamente utilizado para estimar, en tiempo real, la evolución dinámica de un proceso.

Es muy importante comprender claramente el principio de funcionamiento de este algoritmo pues, como veremos más adelante, es la base para una amplia gama de filtros altamente eficientes.

2.7.2. Ejemplo de localización usando un filtro de Bayes

Supongamos que el VANT se desplaza en un medio ambiente representado por una rejilla de 7×8 celdas, y que recibe comandos para desplazarse una celda a la vez. Dentro del medio ambiente existe una celda que contiene un marcador especial, y el VANT posee un sensor binario que detecta la presencia del marcador si se encuentra en esa casilla.

El comportamiento del sensor $Z \in \{marker, \neg marker\}$ es de tal manera que si el VANT se encuentra exactamente sobre el marcador, la probabilidad de

⁸Belief, en inglés.

		VANT			**		

Figura 2.4: Medio ambiente, posición inicial del VANT, marcador.

detectarlo correctamente es de 0.8, mientras que si se encuentra en una celda contigua la probabilidad es de 0.1.

Las órdenes de movimiento son $U \in \{N, S, E, W\}$ y aunque el VANT intenta ejecutarlas de manera precisa, sólo lo consigue el 60 % de las ocasiones, con un 10 % de terminar en una celda contigua a la buscada.

Si a partir de la situación inicial, la secuencia de datos es $\{u_1 = E, z_1 = \neg marker, u_2 = E, z_2 = marker\}$, ¿dónde se encuentra el VANT en $t = 2$?

Preparación

En este caso, la variable aleatoria de estado X es el conocimiento que tenemos de que el VANT se encuentre en cada una de las casillas individuales del medio ambiente. Para su representación, requerimos de una rejilla de 7×8 valores. Este tipo de representación se denomina también **rejilla de histograma**.

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figura 2.5: Conocimiento inicial sobre el estado $Bel(x_0)$.

Cuando se realiza una acción, cada celda propaga su probabilidad actual de

acuerdo al modelo de acción correspondiente. La probabilidad final será la suma de todas las propagaciones individuales.

0.0	0.0	0.0	0.0		0.0	0.0	0.1	0.0
0.0	1.0	0.0	0.0	$u=E$	0.0	0.1	0.6	0.1
0.0	0.0	0.0	0.0		0.0	0.0	0.1	0.0

Figura 2.6: Modelo de acción para $u_t = E$. La imagen muestra la propagación de la probabilidad dictada por $p(x_t \mid x_{t-1}, E)$.

Finalmente, el modelo del sensor muestra la probabilidad de que el marcador sea observado desde cada una de las posiciones posibles del VANT⁹. Por simplicidad, en este caso se muestran las probabilidades individuales para cada celda, pero no debe perderse de vista que para reflejar probabilidades reales, la función debe ser normalizada sobre TODAS las celdas del medio ambiente.

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0		1.0	1.0	1.0	1.0	1.0	0.9	1.0	1.0
0.0	0.0	0.0	0.0	0.1	0.8	0.1	0.0		1.0	1.0	1.0	1.0	0.9	0.2	0.9	1.0
0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0		1.0	1.0	1.0	1.0	1.0	0.9	1.0	1.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Figura 2.7: Modelo de sensor. (a) $p(marker \mid x_t)$, (b) $p(\neg marker \mid x_t)$.

Primer paso. Predicción en $t = 1$.

El comando recibido es $u_1 = E$, por lo que se aplica el modelo de acción sobre todas las celdas de la creencia $Bel(x_0)$ para obtener la predicción del siguiente estado $\overline{Bel}(x_1)$.

Segundo paso. Corrección en $t = 1$.

Puesto que el sensor arrojó la lectura $z_1 = \neg marker$, se debe emplear el modelo de sensor $p(\neg marker \mid x_t)$. Se multiplican celda a celda las distribuciones $\overline{Bel}(x_1)$ y $p(\neg marker \mid x_t)$ para obtener una corrección no normalizada de $Bel(x_1)$.

⁹Para este caso, la función de distribución de probabilidad es constante pues no depende de la posición actual del robot, pero en general es necesario recalculer la función tomando en cuenta la posición del robot.

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0
0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	$u=E$	0.0	0.0	0.1	0.6	0.1	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figura 2.8: Predicción en $t = 1$. (a) $Bel(x_0)$, (b) $\overline{Bel}(x_1)$.

Como se comentó anteriormente, el factor de normalización se calcula *a posteriori*. Para que el resultado sea una distribución de probabilidad válida, la suma de todos los elementos de X debe ser igual a 1.

El factor de normalización η_1 es el inverso de la suma de todas las celdas obtenidas. En este caso, la suma de las celdas de $\overline{Bel}(x_1) * p(\neg marker | x_t)$ es 0.99, por lo que dividimos todas las celdas por ese valor, para obtener finalmente la nueva distribución de probabilidad de estado $Bel(x_1)$.

	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0		1.0	1.0	1.0	1.0	1.0	0.9	1.0	1.0
	0.0	0.0	0.1	0.6	0.1	0.0	0.0	0.0	*	1.0	1.0	1.0	1.0	0.9	0.2	0.9	1.0
	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0		1.0	1.0	1.0	1.0	1.0	0.9	1.0	1.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00		0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00
=	0.00	0.00	0.10	0.60	0.09	0.00	0.00	0.00		0.00	0.00	0.10	0.61	0.09	0.00	0.00	0.00
	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00		0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
								0.99									

Figura 2.9: Corrección en $t = 1$. (a) $\overline{Bel}(x_1)$, (b) $p(\neg marker | x_t)$, (c) $\overline{Bel}(x_1) * p(\neg marker | x_t)$, (d) $Bel(x_1)$ normalizado.

Tercer paso. Predicción en $t = 2$.

El comando recibido es $u_2 = E$, por lo que se aplica el modelo de acción sobre todas las celdas de la creencia $Bel(x_1)$ para obtener la predicción del siguiente estado $Bel(x_2)$.

0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00
0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00		0.00	0.00	0.00	0.02	0.12	0.02	0.00	0.00
0.00	0.00	0.10	0.61	0.09	0.00	0.00	0.00	$u=E$	0.00	0.00	0.01	0.12	0.40	0.12	0.01	0.00
0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00		0.00	0.00	0.00	0.02	0.12	0.02	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Figura 2.10: Predicción en $t = 2$. (a) $Bel(x_1)$, (b) $\overline{Bel}(x_2)$.

Último paso. Corrección en $t = 2$.

Ahora el sensor arrojó la lectura $z_2 = marker$, por lo que se debe emplear el modelo de sensor $p(marker | x_t)$. Se multiplican celda a celda las distribuciones $\overline{Bel}(x_2)$ y $p(\neg marker | x_t)$ para obtener una corrección no normalizada de $Bel(x_2)$.

El factor de normalización η_2 es el inverso de la suma de todas las celdas obtenidas. En este caso, la suma de las celdas de $\overline{Bel}(x_2) * p(\neg marker | x_t)$ es 0,137, por lo que dividimos todas las celdas por ese valor, y tenemos finalmente la nueva distribución de probabilidad de estado $Bel(x_2)$.

	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00		0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.02	0.12	0.02	0.00	0.00		0.00	0.00	0.00	0.00	0.00	0.10	0.00
	0.00	0.00	0.01	0.12	0.40	0.12	0.01	0.00	*	0.00	0.00	0.00	0.00	0.10	0.00	0.00
	0.00	0.00	0.00	0.02	0.12	0.02	0.00	0.00		0.00	0.00	0.00	0.00	0.00	0.10	0.00
	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00		0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.00	0.00	0.00	0.00	0.00	0.00	0.00
=	0.00	0.00	0.00	0.00	0.04	0.09	0.00	0.00		0.00	0.00	0.00	0.00	0.29	0.66	0.01
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.00	0.00	0.00	0.00	0.00	0.00	0.00
								0.14								

Figura 2.11: Corrección en $t = 2$. (a) $\overline{Bel}(x_2)$, (b) $p(\neg marker | x_t)$, (c) $\overline{Bel}(x_2) * p(\neg marker | x_t)$, (d) $Bel(x_2)$ normalizado.

Solución.

Por las incertidumbres en las observaciones del sensor y de las acciones del robot, es imposible dar una respuesta determinista a la pregunta original de dónde se encuentra el VANT. Sin embargo, es posible responder que el VANT tiene un 66 % de probabilidades de encontrarse justo encima del marcador, y un 29 % de encontrarse una casilla al oeste del mismo.

Puede observarse que, aun cuando el robot sólo recibió dos comandos $u = E$, muy probablemente se desplazó 3 casillas a la derecha.

Tarea: En esta ocasión, no se sabe en cuál de las cinco posiciones iniciales se encuentra el VANT. Si los datos generados son $u_1 = E, z_1 = \neg \text{marker}, u_2 = E, z_2 = \text{marker}, u_3 = N, z_3 = \neg \text{marker}$, ¿cuál es la posición del VANT en $t = 3$?

		VANT			**		
	VANT	VANT	VANT				
		VANT					

Figura 2.12: Conocimiento inicial sobre el estado $Bel(x_0)$.

2.8. Distribuciones normales

Como se mencionó anteriormente, es posible aplicar el filtro de Bayes sobre una variable aleatoria de estado X definida en un espacio continuo. Una importante familia de filtros continuos son los llamados **filtros gaussianos**, que comparten el hecho de representar la creencia de estado (la distribución de probabilidad de X) empleando una distribución normal (o distribución gaussiana).

La importancia que reviste la distribución normal es que la gran mayoría de los fenómenos naturales se pueden aproximar empleando una distribución de este tipo. Este resultado puede explicarse por la interacción de una multitud de efectos aleatorios, no observables (medibles) individualmente, sobre la variable que se está estudiando. Por el **teorema del límite central**, sabemos que si S_n

es la suma de n variables aleatorias independientes de varianza finita, entonces la función de distribución de S_n se asemeja a una función normal cuando n es *suficientemente* grande, independientemente de la forma de las distribuciones individuales.

Otra manera de enunciar el teorema del límite central, es que si tomamos muestras de una población con cualquier tipo de distribución (pero con media y varianza finitas), entonces la distribución de las medias de las muestras tiende a una distribución normal al aumentar el número de muestras¹⁰.

Una variable aleatoria X tiene distribución normal con media μ y varianza σ^2 , si su función de distribución de probabilidad $p(x)$ está definida de la siguiente manera:

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)$$

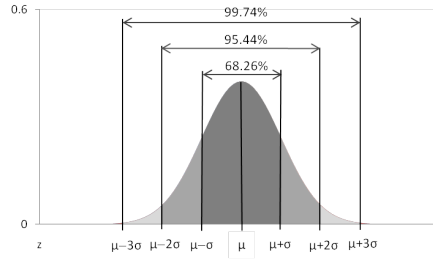


Figura 2.13: Distribución normal univariable.

De esta manera, la variable X queda totalmente descrita por su media y su varianza, lo cual se representa de la siguiente manera:

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

Una distribución normal multivariada sobre un vector aleatorio X de dimensión d , con vector de esperanza μ y matriz de covarianza¹¹ Σ , queda definida de la siguiente manera:

$$X \sim \mathcal{N}(\mu, \Sigma)$$

$$p(x) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^\top \Sigma^{-1}(x - \mu)\right)$$

¹⁰La máquina de Galton puede considerarse el primer generador de números aleatorios con una distribución normal.

¹¹ Σ es una matriz simétrica semipositiva, de dimensión $d \times d$, que puede reescribirse como $\Sigma = AA^\top$.

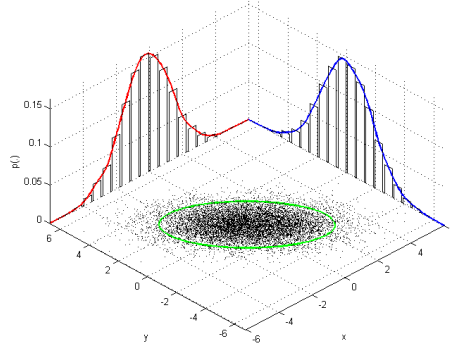


Figura 2.14: Distribución normal biviada.

La transformación lineal de una variable con distribución normal es otra variable aleatoria con distribución normal:

$$\begin{aligned} X &\sim \mathcal{N}(\mu, \Sigma), \quad Y = AX + B \\ \Rightarrow Y &\sim \mathcal{N}(A\mu + B, A\Sigma A^\top) \end{aligned}$$

Asimismo, dadas dos variables aleatorias independientes $X_1 \sim \mathcal{N}(\mu_1, \Sigma_1)$ y $X_2 \sim \mathcal{N}(\mu_2, \Sigma_2)$, su probabilidad conjunta es también una distribución normal:

$$\begin{aligned} \Rightarrow p(X_1, X_2) &\propto p(X_1)p(X_2) \\ p(X_1, X_2) &\sim \mathcal{N}\left(\frac{1}{\Sigma_1}\mu_1 + \frac{1}{\Sigma_2}\mu_2, \frac{1}{\Sigma_1^{-1} + \Sigma_2^{-1}}\right) \end{aligned}$$

Es posible demostrar (aunque no es trivial), que la distribución $p(X_1, X_2)$ definida anteriormente es la mezcla ponderada $f(x) = w_1p(X_1) + w_2p(X_2)$ con la mínima covarianza posible¹².

Si consideramos las distribuciones X_1 y X_2 como evidencias o muestras independientes de un estado X , podemos observar que la distribución conjunta $p(X_1, X_2)$ tiene media:

$$\mu = \frac{1}{\Sigma_1}\mu_1 + \frac{1}{\Sigma_2}\mu_2$$

que es una media ponderada basada en la confianza (dispersión) de las evidencias individuales. Asimismo, la dispersión de la distribución conjunta:

$$\Sigma = \frac{1}{\Sigma_1^{-1} + \Sigma_2^{-1}}$$

es menor que la menor de las dispersiones individuales, lo que muestra que la distribución resultante tiene una mayor confianza o precisión que las observaciones individuales, sin importar qué tan malas sean éstas.

¹²Dada la definición de covarianza, se trata de una minimización por mínimos cuadrados.

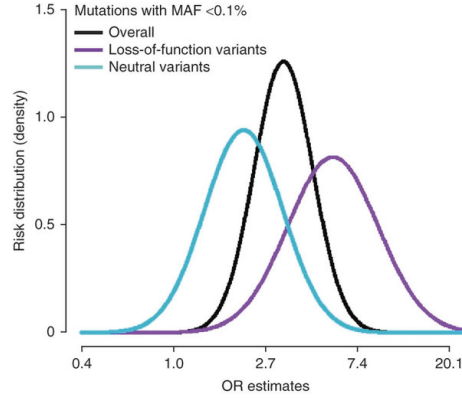


Figura 2.15: Ejemplo de fusión de distribuciones normales, para obtener una mejor estimación.

La distribución conjunta $p(X_1, X_2)$ puede reescribirse como:

$$p(X_1, X_2) \sim \mathcal{N}\left(\mu_1 + \frac{\Sigma_1}{\Sigma_1 + \Sigma_2}(\mu_2 - \mu_1), \frac{\Sigma_1 \Sigma_2}{\Sigma_1 + \Sigma_2}\right)$$

donde se tiene que la media μ puede interpretarse como una “mejora” de la estimación μ_1 hacia μ_2 con una corrección proporcional a la diferencia entre las dos observaciones. Dicha diferencia $\mu_2 - \mu_1$ se denomina **innovación**.

Por el teorema de Bayes, si tenemos un conjunto de creencias independientes X_1, X_2, \dots, X_n sobre un estado X , cada una con distribución normal, podemos mejorar la estimación μ_1 aplicando recursivamente la fórmula anterior, para fusionar las innovaciones de información de cada una de las creencias individuales.

2.9. Sistemas lineales y Filtro de Kalman

El filtro de Kalman es un filtro gaussiano empleado para estimar, bajo un esquema de optimización por mínimos cuadrados, el estado de un proceso estocástico lineal en instantes discretos que cumple la propiedad de Markov¹³.

El estado del sistema en el instante t está representado por una creencia x_t con distribución normal, que evoluciona a través del tiempo, caracterizada por su media μ_t y su dispersión Σ_t :

$$x_t \sim \mathcal{N}(\mu_t, \Sigma_t)$$

¹³Dicho de otra manera, el filtro de Kalman es un filtro gaussiano óptimo para un proceso de Markov lineal.

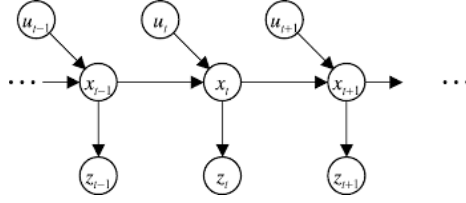


Figura 2.16: Proceso de Markov (en robótica móvil).

Bajo la propiedad de Markov, el estado x_t depende **linealmente** del estado anterior x_{t-1} y de la acción actual u_t , con una incertidumbre (ruido) gaussiana ϵ_t de media cero y varianza R_t :

$$x_t = Ax_{t-1} + Bu_t + \epsilon_t$$

Asimismo, las observaciones z_t son linealmente dependientes del estado x_t , con una incertidumbre gaussiana δ_t de media cero y varianza Q_t :

$$z_t = Cx_t + \delta_t$$

Resumiendo, el proceso de Markov estudiado es el siguiente:

- Vector de variable de estado $x_t \in \mathbb{R}^n$
- Vector de acciones $u_t \sim \mathbb{R}^l$
- Vector de observaciones $z_t \sim \mathbb{R}^m$
- Ruido gaussiano sobre el estado $\epsilon_t \in \mathbb{R}^n$ y $\epsilon_t \sim \mathcal{N}(0, R)$
- Ruido gaussiano sobre las observaciones $\delta_t \in \mathbb{R}^m$ y $\delta_t \sim \mathcal{N}(0, Q)$
- Ecuación de proceso (modelo de acción): $x_t = Ax_{t-1} + Bu_t + \epsilon_t$; con $A_{n \times n}$ y $B_{n \times l}$.
- Ecuación de lectura (modelo de sensor): $z_t = Cx_t + \delta_t$; con $C_{m \times n}$.
- Una creencia inicial sobre el estado $Bel(x_0)$, representada por su estimación (media) μ_0 y su confianza (varianza) Σ_0 .

$$Bel(x_0) \sim \mathcal{N}(\mu_0, \Sigma_0)$$

El filtro de Kalman, al igual que el filtro de Bayes que estudiamos anteriormente, emplea el modelo de acción sobre la creencia anterior $Bel(x_{t-1})$ para obtener la hipótesis *a priori* (una distribución normal) del estado actual del sistema $\overline{Bel}(x_t)$:

$$\overline{Bel}(x_t) \sim \mathcal{N}(\bar{\mu}_o, \bar{\Sigma}_0)$$

La hipótesis *a priori* $\overline{Bel}(x_t)$ se emplea para obtener una **predicción** $C\overline{Bel}(x_t)$ sobre las observaciones sobre el sistema, empleando el modelo directo del sensor. Con las lecturas reales z_t de los sensores y la predicción anterior se construye el **vector de innovación**:

$$z_t - C\overline{Bel}(x_t)$$

el cual se emplea para **actualizar** la hipótesis *a priori* y obtener la hipótesis *a posteriori*:

$$Bel(x_t) = \overline{Bel}(x_t) + \mathbf{K}_t(z_t - \overline{Bel}(x_t))$$

donde \mathbf{K}_t es una matriz de dimensión $n \times m$, denominada **ganancia de Kalman**, calculada para minimizar la covarianza entre la hipótesis *a priori* y la hipótesis *a posteriori*, conocida como **covarianza residual**:

$$\min_K Cov\{Bel(x_t), \overline{Bel}(x_t)\}$$

El problema de optimización anterior es en realidad un problema de optimización por mínimos cuadrados, que conduce al planteamiento de una ecuación matricial no lineal de segundo orden conocida como **Ecuación matricial de Riccati**, cuya solución óptima es precisamente la matriz de Kalman \mathbf{K}_t .

El filtro de Kalman tiene el siguiente algoritmo:

Algoritmo del filtro de Kalman ($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):

1. $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
2. $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^\top + R_t$
3. $K_t = \bar{\Sigma}_t C_t^\top (C_t \bar{\Sigma}_t C_t^\top + Q_t)^{-1}$
4. $\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$
5. $\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$
6. *return* (μ_t, Σ_t)

Las líneas 1 y 2 calculan la hipótesis *a priori*, empleando la estimación anterior x_{t-1} y las acciones de control u_t , para predecir el nuevo promedio y varianza de $\overline{Bel}(x_t)$. La línea 3 calcula la matriz de ganancia de Kalman. Las líneas 4 y 5 corrigen la estimación, empleando la ganancia de Kalman y el vector de innovación de información.

Puede apreciarse que el filtro de Kalman es una herramienta computacionalmente eficiente, con pocas operaciones matriciales, lo que permite su uso en aplicaciones de tiempo real.

También, la derivación matemática del filtro de Kalman no impone que los ruidos ϵ_t y δ_t sean gaussianos; sin embargo, la solución de la ecuación de Riccati es exacta cuando los ruidos son gaussianos.

Para determinar las matrices de covarianza Q_t y R_t procedemos de la siguiente manera: si el vector de observación y_t tiene desviaciones estándar σ_j^y en cada una de las componentes, entonces la matriz de covarianza de la observación está dada por:

$$Q_t = \text{diag}_j(\sigma_j^y)^2$$

que es la matriz diagonal de las varianzas individuales.

Por regla general, conocemos el ruido sobre cada una de las componentes individuales de la acción u_t , pero para aplicar el filtro de Kalman necesitamos conocer la covarianza R_t sobre el **estado** del sistema x_t . Si el vector de acción u_t tiene desviaciones estándar individuales σ_i^u , entonces la matriz de covarianza R_t se obtiene como sigue:

$$R_t = B_t \text{diag}_i(\sigma_i^u)^2 B_t^\top$$

En ambos casos, es posible reemplazar la matriz diagonal de varianzas por una estimación experimental de la matriz de covarianza de la acción y de la observación.

Finalmente, hay que subrayar que el proceso puede ser dinámico, es decir, las matrices A_t , B_t , C_t , R_t y Q_t pueden variar a través del tiempo t . La única condición es que no dependan del estado x_t ni del control u_t .

2.9.1. Ejemplo

Supongamos que deseamos conocer la altura a la que vuela un VANT. Los motores del VANT pueden generar una fuerza ascensional f_u con una desviación estándar σ_u . Asimismo, el VANT cuenta con un sensor de altitud con una desviación estándar σ_y .

Sobre el VANT actúan tanto su propio peso $W = mg$ como la fuerza de los motores f_u . Aplicando la segunda ley de Newton tenemos:

$$\begin{aligned} m\ddot{y} &= \sum f \\ \ddot{y} &= -g + \frac{1}{m}f_u \end{aligned}$$

Este último término depende únicamente de la acción de motores, por lo que haremos:

$$u_t = -g + \frac{1}{m}f_u$$

Integrando \ddot{y} , obtenemos la siguiente expresión:

$$\begin{aligned} \dot{y} &= \dot{y}_0 + u_t t \\ y &= y_0 + \dot{y}_0 t + \frac{1}{2}u_t t^2 \end{aligned}$$

Aplicando en el intervalo Δt , obtenemos las ecuaciones de diferencias siguientes:

$$\begin{aligned} y_t &= y_{t-1} + \dot{y}_{t-1}\Delta t + \frac{1}{2}u_t\Delta t^2 \\ \dot{y}_t &= \dot{y}_{t-1} + u_t\Delta t \end{aligned}$$

Reescribiendo de manera matricial:

$$\begin{bmatrix} y_t \\ \dot{y}_t \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y_{t-1} \\ \dot{y}_{t-1} \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix} u_t$$

Evidentemente, la ecuación de observación del sistema es la siguiente:

$$z_t = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} y_t \\ \dot{y}_t \end{bmatrix}$$

Con lo anterior tenemos la descripción completa del movimiento del VANT:

$$\begin{aligned} x_t &= Ax_{t-1} + Bu_t \\ z_t &= Cx_t \end{aligned}$$

De las mismas ecuaciones anteriores podemos obtener la matrices de covarianza del ruido sobre el estado x_t y sobre la lectura z_t :

$$\begin{aligned} R &= B\sigma_u^2 B^\top \\ &= \begin{bmatrix} 0,25\Delta t^4 & 0,5\Delta t^3 \\ 0,5\Delta t^3 & \Delta t^2 \end{bmatrix} \sigma_u^2 \\ Q &= \sigma_y^2 \end{aligned}$$

Con toda esta información, es posible implementar un filtro de Kalman.

Tarea: Inicialmente, el VANT se encuentra en tierra ($y_0 = 0$ y $\dot{y}_0 = 0$). La aceleración total sobre el VANT proporcionada por los motores es $u = 0,1\text{m/s}^2$, con una desviación estándar de $\sigma_u = 0,1\text{m/s}^2$ debida principalmente a la interacción entre el VANT y el suelo. El sensor de altitud (extremadamente barato) entrega lecturas cada décima de segundo, con una desviación estándar $\sigma_y = 0,5\text{m}$. Implementar en Matlab la simulación correspondiente a los primeros 10 segundos del vuelo del VANT, así como la estimación hecha por un filtro de Kalman. Analizar y reportar el efecto de la estimación y la covarianza iniciales sobre la evolución de la estimación proporcionada por el filtro. Analizar y reportar la sensibilidad del filtro a los cambios de desviación estándar σ_u y σ_y .

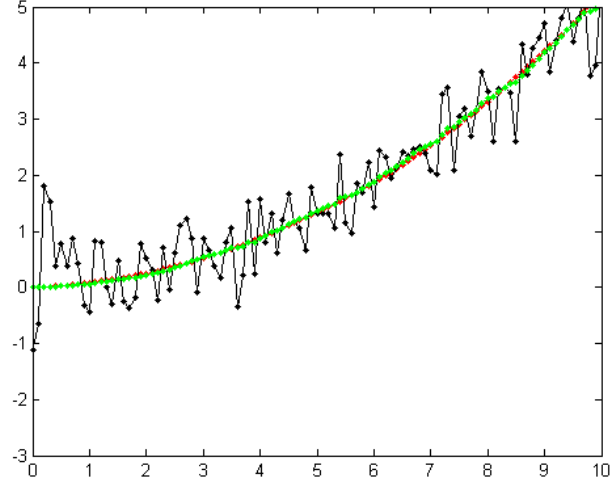


Figura 2.17: Aplicación del filtro de Kalman al despegue de un VANT.

2.10. Filtro de Kalman Extendido

En el caso del Filtro de Kalman, el principal interés de considerar únicamente sistemas lineales es que permite actualizar la varianza (gaussiana) de las distintas variables aleatorias empleando operaciones lineales, para obtener una nueva varianza (gaussiana).

Desafortunadamente en el mundo real, y particularmente en robótica móvil, son pocos los sistemas que pueden modelarse empleando ecuaciones lineales. Bajo el supuesto de que la distribución de probabilidad de las variables aleatorias del **sistema es gaussiana**, podemos emplear una linearización del modelo alrededor de la media estimada, para extender el uso del filtro de Kalman a sistemas no lineales.

En este caso, consideramos que el sistema está gobernado por los siguientes modelos de acción y de observación:

$$\begin{aligned}x_t &= g(x_{t-1}, u_t) + \epsilon_t \\z_t &= h(x_t) + \delta_t\end{aligned}$$

donde las funciones g y h son funciones no lineales y los ruidos ϵ_t y δ_t son los ruidos de estado y de observación, respectivamente. Dado que el sistema no es lineal, aún cuando consideremos que la distribución de la variable x_{t-1} es gaussiana, no es posible actualizar la varianza de la variable x_t , pues en general no existe una expresión en forma cerrada para la ganancia de Kalman necesaria para incluir la innovación de información entre la acción y la observación.

La solución es aproximar las funciones g y h por sus primeras aproximaciones en series de Taylor:

$$\begin{aligned}
g(x_{t-1}, u_t) &\approx g(\mu_{t-1}, u_t) + (x_{t-1} - \mu_{t-1}) \frac{\partial g(\mu_{t-1}, u_t)}{\partial x_{t-1}} \\
&= g(\mu_{t-1}, u_t) + (x_{t-1} - \mu_{t-1}) G_t \\
h(x_t) &\approx h(\mu_t) + (x_t - \mu_t) \frac{\partial h(\mu_t)}{\partial x_t} \\
&= h(\mu_t) + (x_t - \mu_t) H_t
\end{aligned}$$

El filtro de Kalman Extendido tiene el siguiente algoritmo:

Algoritmo del filtro de Kalman ($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):

1. Compute G_t, H_t, R_t, Q_t
2. $\bar{\mu}_t = g(\mu_{t-1}, u_t)$
3. $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^\top + R_t$
4. $K_t = \bar{\Sigma}_t H_t^\top (H_t \bar{\Sigma}_t H_t^\top + Q_t)^{-1}$
5. $\mu_t = \bar{\mu}_t + K_t (z_t - h(\bar{\mu}_t))$
6. $\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t$
7. *return* (μ_t, Σ_t)

La línea 1 calcula la linearización del sistema alrededor del punto μ_{t-1} . La línea 2 predice la nueva media del estado empleando el modelo no lineal del sistema. En la línea 3 se predice la nueva varianza utilizando la aproximación lineal del sistema. En la línea 4 se calcula la ganancia de Kalman para la versión lineal del sistema. La línea 5 actualiza la media utilizando la ganancia lineal sobre la innovación no lineal. Finalmente, la línea 6 actualiza la varianza del estado por medio de una aproximación gaussiana lineal.

Únicamente para resaltar la linearización que se ha hecho del sistema:

- La matriz A_t se ha reemplazado por la matriz de derivadas parciales G_t de la función de transición g respecto al estado x_{t-1} (la *jacobiana* de g respecto a x).

$$G_t = \frac{\partial g(\mu_{t-1}, u_t)}{\partial x_{t-1}}$$

- La matriz C_t se ha aproximado empleando la matriz jacobiana H_t de la función de observación h respecto a x .

$$H_t = \frac{\partial h(\mu_t)}{\partial x_t}$$

- En caso de ser necesaria, la matriz B_t puede aproximarse con la jacobiana W_t de la función de transición g respecto a la acción u .

$$W_t = \frac{\partial g(\mu_{t-1}, u_t)}{\partial u_t}$$

- La linearización se emplea solamente para calcular la ganancia de Kalman y actualizar la varianza de las variables. La estimación de la variable (la media μ_t) se calcula **empleando el modelo no lineal**.

2.10.1. Ejemplo

Consideremos un robot móvil de tipo diferencial, que se desplaza utilizando comandos de velocidad lineal v y de velocidad angular ω . Tras un lapso de tiempo Δt , el robot se ha desplazado una distancia Δd y girado un ángulo $\Delta\beta$, con desviaciones estándar de σ_d y σ_β respectivamente¹⁴.

Además, el robot posee un sensor GPS que le permite conocer su posición actual $(\hat{x}_t, \hat{y}_t, \hat{\theta}_t)$, con desviaciones estándar $(\sigma_x, \sigma_y, \sigma_\theta)$.

Diseño del Filtro de Kalman

Las ecuaciones odométricas que rigen el movimiento del robot son las siguientes:

$$\begin{aligned} x_t &= x_{t-1} + \Delta d \cos\left(\theta_{t-1} + \frac{\Delta\beta}{2}\right) \\ y_t &= y_{t-1} + \Delta d \sin\left(\theta_{t-1} + \frac{\Delta\beta}{2}\right) \\ \theta_t &= \theta_{t-1} + \Delta\beta \end{aligned}$$

Las ecuaciones de observación son directas:

$$\begin{aligned} z_{x,t} &= \hat{x}_t \\ z_{y,t} &= \hat{y}_t \\ z_{\theta,t} &= \hat{\theta}_t \end{aligned}$$

A partir de estas ecuaciones podemos obtener la linearización del modelo de desplazamiento del robot móvil:

¹⁴En la práctica, los valores de estos incrementos y sus errores asociados pueden obtenerse directamente a partir de las lecturas de los *encoders* de los motores, como se efectuó durante el curso de Robots Móviles Inteligentes.

- Linearización del modelo de acción:

$$\begin{aligned}
G_t &= \frac{\partial}{\partial x_{t-1}} g(\mu_{t-1}, u_t) \\
&= \begin{bmatrix} 1 & 0 & -\Delta d \sin\left(\theta_{t-1} + \frac{\Delta\beta}{2}\right) \\ 0 & 1 & \Delta d \cos\left(\theta_{t-1} + \frac{\Delta\beta}{2}\right) \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned}$$

- Linearización del modelo de observación:

$$\begin{aligned}
H_t &= \frac{\partial}{\partial x_{t-1}} h(\mu_t) \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned}$$

- Linearización del aporte de ruido proveniente de la acción u_t :

$$\begin{aligned}
W_t &= \frac{\partial}{\partial u_t} g(\mu_{t-1}, u_t) \\
&= \begin{bmatrix} \cos\left(\theta_{t-1} + \frac{\Delta\beta}{2}\right) & \frac{-\Delta d}{2} \sin\left(\theta_{t-1} + \frac{\Delta\beta}{2}\right) \\ \sin\left(\theta_{t-1} + \frac{\Delta\beta}{2}\right) & \frac{\Delta d}{2} \cos\left(\theta_{t-1} + \frac{\Delta\beta}{2}\right) \\ 0 & 1 \end{bmatrix}
\end{aligned}$$

de donde se puede obtener la matriz de covarianza R_t como:

$$\begin{aligned}
R_t &= W_t \Sigma_u W_t^\top \\
&= W_t \begin{bmatrix} \sigma_d^2 & 0 \\ 0 & \sigma_\beta^2 \end{bmatrix} W_t^\top
\end{aligned}$$

- La matriz de covarianza Q_t del aporte de ruido proveniente de la observación:

$$Q_t = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{bmatrix}$$

Con todos los elementos anteriores, sólo se requiere conocer la estimación inicial de la posición del robot, así como la confianza inicial (covarianza) sobre esa estimación, para poder implementar un filtro de Kalman Extendido sobre la evolución del sistema.

Tarea: Un VANT navega en un medio ambiente a altura fija, recibiendo comandos de desplazamiento frontal, lateral y de giro sobre

sí mismo, respecto a su propio sistema coordenado. Por perturbaciones del medio ambiente y otros fenómenos aleatorios, los desplazamientos del robot están contaminados con un ruido gaussiano (no medible) de media cero y varianzas σ_x , σ_y y σ_θ , para cada uno de los desplazamientos. Para ayudar a su localización, se colocaron 3 balizas dentro del ambiente en posiciones conocidas $r_i = (x_i, y_i)$ con respecto a un marco de referencia fijo. El VANT cuenta con un sensor láser de espejo giratorio, que detecta el ángulo relativo al eje principal del robot en el que se encuentran las balizas reflejantes. El sistema de percepción es capaz también de identificar las balizas individualmente, por lo que la observación del sensor consiste en la tripleta de ángulos $(\beta_1, \beta_2, \beta_3)$ sin importar el orden en que hayan sido detectadas las balizas. Cada ángulo tiene una imprecisión aleatoria de lectura, de distribución gaussiana de media cero y varianza σ_i . Implementar en Matlab la simulación correspondiente a los primeros 10 segundos del vuelo del VANT en algunas trayectorias predeterminadas, así como la estimación hecha por un filtro extendido de Kalman. Analizar y reportar el efecto de la estimación y la covarianza iniciales sobre la evolución de la estimación proporcionada por el filtro. Analizar y reportar la sensibilidad del filtro a los cambios de desviación estándar.

2.11. Filtro de partículas

El filtro de Kalman es un filtro que maneja una única hipótesis de trabajo. Cuando el sistema que se desea modelar no es lineal o linealizabile (o simplemente no contamos con el modelo) o bien desconocemos la forma de la distribución de probabilidad (es decir, no podemos decir si es gaussiana o no), se puede emplear entonces un filtro multi-hipótesis que permita muestrear la función de probabilidad que deseamos describir. La función es descrita por una secuencia finita de muestras que nos dan idea de su forma general, en lugar de contar con una expresión matemática de ella. Cuando el muestreo es gobernado por un proceso aleatorio, se denominan filtros de MonteCarlo.

2.11.1. Métodos de MonteCarlo

Los métodos de MonteCarlo son métodos no deterministas empleados para aproximar funciones complejas o costosas de evaluar con exactitud, utilizando repetidamente un muestreo aleatorio hasta obtener una descripción numérica de la función.

En general, en un método de MonteCarlo se procede a someter a la función (o sistema) que se desea modelar a una serie de valores generados aleatoriamente.

te, analizando los resultados obtenidos y confrontándolos con el conocimiento previo sobre la función, con el propósito de estimar la función y/o predecir el comportamiento de la misma.

La implementación de un método de MonteCarlo puede variar de una situación a otra, pero en general se siguen los siguientes pasos:

1. Se define el dominio de valores de entrada de la función a determinar
2. Se generan valores de entrada de manera aleatoria, dentro del dominio definido y siguiendo algún patrón de distribución establecido
3. Se desarrolla algún análisis determinístico sobre los resultados obtenidos
4. Se actualiza el conocimiento sobre la función a determinar

Un detalle importante al implementar un método de MonteCarlo es realizar una buena elección del generador de números aleatorios. Un generador sesgado producirá resultados sesgados.

Ejemplo: Cálculo de π

Consideremos un blanco de dardos con la forma de un cuadrado de lado 1, en el cual se ha inscrito un cuarto de círculo con centro en una de las esquinas del cuadrado. Sabemos que el área del cuadrado es 1 y que el área del sector de círculo es $\pi/4$. Si lanzáramos un dardo hacia el blanco, tendríamos una posibilidad de $\pi/4$ de caer dentro del sector circular. El método de MonteCarlo para calcular una aproximación del valor de π podría quedar como sigue:

1. El dominio de valores de entrada es 0 a 1, tanto para x como para y (posición del dardo)
2. Se genera aleatoriamente la posición (x, y) del dardo, dentro del dominio definido, con una distribución de probabilidad uniforme
3. Si $\sqrt{x^2 + y^2} \leq 1$ entonces el dardo cayó en el sector circular
4. La aproximación de π se obtiene con la expresión $4N_c/T_t$, donde N_c es el número de dardos que han caído en el sector circular y T_t es el número total de dardos lanzados
5. Se repite lo anterior un número suficiente de veces, hasta alcanzar la precisión deseada.

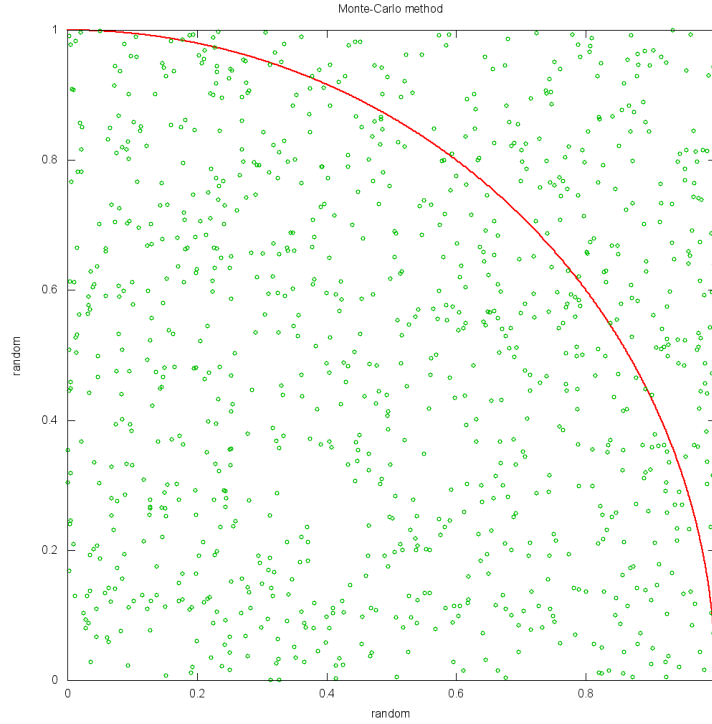


Figura 2.18: Método de MonteCarlo para cómputo de π .

Tarea: Implementar en Matlab un método de MonteCarlo para el cálculo de π , mostrando la evolución de la aproximación conforme se "lanzan dardos". ¿Cuántos "dardos" se necesitan para obtener una aproximación correcta de 4 cifras decimales?

2.11.2. Algoritmo del filtro de partículas

El filtro de Kalman mantiene (actualiza) una única hipótesis (gaussiana) sobre el estado del sistema que tiene una dinámica lineal. El filtro de partículas, a diferencia del filtro de Kalman, mantiene (actualiza) una población de hipótesis para tratar de explicar el estado (cualquiera, ya no únicamente gaussiano) del sistema que tiene una dinámica cualquiera (incluso no lineal).

$$\begin{aligned} x_t &= g(x_{t-1}, u_t) + \epsilon_t \\ z_t &= h(x_t) + \delta_t \end{aligned}$$

El filtro de partículas trabaja de manera poblacional, generando una nueva colección de hipótesis de acuerdo a las observaciones hechas a través de los

sensores. Cada una de las hipótesis (partículas) tiene un peso asociado, que indica qué tan *verdadera* o coherente es dicha hipótesis con la evolución del sistema y las observaciones que se han estado realizando. La probabilidad de que una hipótesis participe en la población siguiente es directamente proporcional a su peso.

El desarrollo teórico del filtro de partículas no es trivial, y requiere un análisis sobre el tipo de sistema que deseamos observar. Sin embargo, en robótica móvil, la implementación del algoritmo más común es la denominada SIR (Sequential Importance Resampling), que se desarrolla en las siguientes etapas:

1. **Inicialización:** Se inicia la población de N partículas (hipótesis) con un muestreo uniforme del espacio de estado del sistema. El peso de cada partícula x_0^i es de $w_i = 1/N$.
2. **Predicción:** A cada una de las partículas x_{t-1}^i se le aplica el modelo dinámico de del sistema, es decir, se predice cuál sería el nuevo estado a partir del estado descrito por la partícula.

$$x_t^i = g(x_{t-1}^i, u_t) + \epsilon_t$$

3. **Corrección:** Se ajustan los pesos w_i de las partículas, comparando la lectura predicha:

$$z_t^i = h(x_t^i) + \delta_t$$

con el modelo del sensor y la lectura real z_t del mismo:

$$w_i = p(z_t | z_t^i)$$

Los pesos actualizados se normalizan de tal manera que:

$$\sum_i w_i = 1$$

Este conjunto de partículas y pesos describe la creencia actual sobre el estado del sistema.

4. **Remuestreo:** Los pesos normalizados se emplean para generar aleatoriamente (proceso de MonteCarlo) una nueva población de N partículas (hipótesis) que describe la nueva creencia discreta sobre el estado del sistema. Esta nueva población se emplea como entrada del algoritmo, en la etapa de predicción, en el instante siguiente $t + 1$.

2.11.3. Algoritmos de remuestreo

Existen diferentes estrategias para construir una nueva población de N partículas a partir de los pesos w_i de la generación anterior. En esta sección se explicarán dos de ellos, el remuestreo multinomial y remuestreo universal.

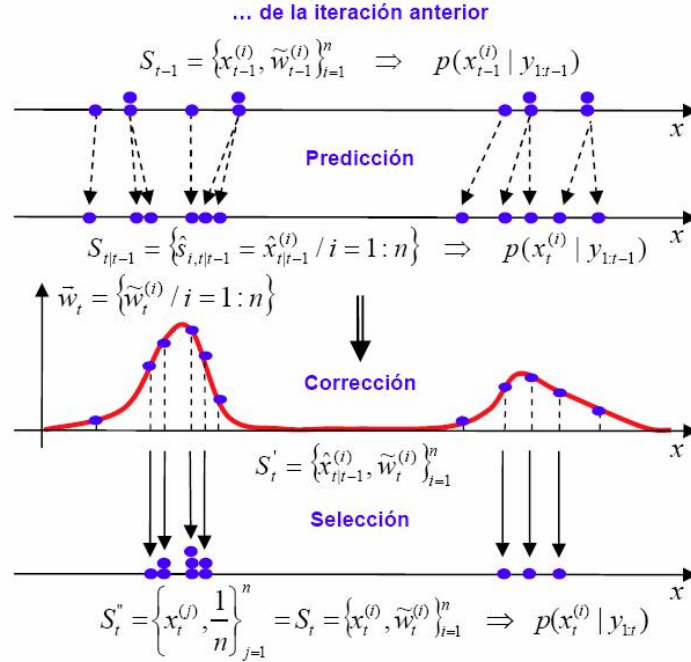


Figura 2.19: Principio del filtro de partículas.

Para comprender ambos esquemas, podemos considerar los lanzamientos de una ruleta donde los sectores son proporcionales a los pesos de las partículas. Cada lanzamiento de la ruleta indicará cuál partícula debe pasar a la siguiente generación.

- **Remuestro multinomial.** El más simple de los métodos de remuestreo. Para elegir las N partículas de la nueva generación, se realizan N lanzamientos aleatorios independientes de la ruleta. Recibe su nombre porque la distribución de la nueva función de probabilidad resultante es la distribución multinomial de los pesos individuales de las partículas. El proceso de selección de la partícula se puede implementar con búsqueda binaria, por lo que el algoritmo de remuestreo es de complejidad $O(N \log N)$.
- **Remuestro universal.** También llamado *remuestreo sistemático*. Se hace un único lanzamiento aleatorio de la ruleta, y se seleccionan las partículas a intervalos fijos de $1/N$ a partir del número obtenido. Tiene la ventaja de ser un método rápido ($O(N)$) y de minimizar la varianza de la generación siguiente.

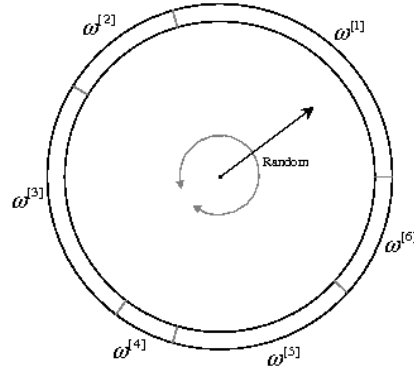


Figura 2.20: Principio del remuestro multinomial.

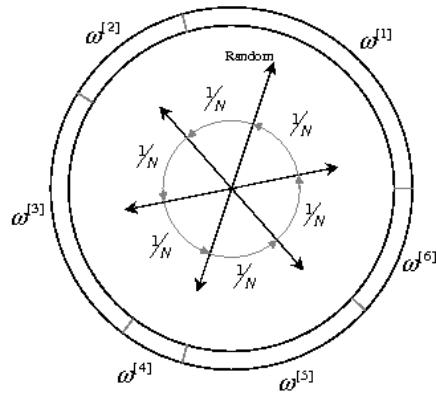


Figura 2.21: Principio del remuestro universal.

2.12. Conclusiones

A lo largo de este capítulo, hemos estudiado las técnicas matemáticas necesarias para integrar las incertidumbres que existen sobre las distintas variables de un robot, tanto en la acción, como en la observación y en la descripción del estado del robot.

En la actualidad, el filtro de Kalman (y su versión extendida) y el filtro de partículas son las principales herramientas para modelar dichas incertidumbres.

Capítulo 3

Visión por computadora

**** Completar con las presentaciones de Ramiro y de Ángel.