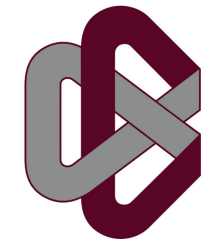


CIMAT

CENTRO DE INVESTIGACIÓN
EN MATEMÁTICAS, A. C.

Estrategias de Movimiento para la Exploración y Construcción de Mapas bajo Incertidumbre con Múltiples Robots Heterogéneos

T E S I S

QUE PARA OBTENER EL GRADO DE:

Doctor en Ciencias
con orientación en
Ciencias de la Computación

PRESENTA:

M. I. Luis Manuel Valentín Coronado

DIRECTOR DE TESIS:

Dr. Rafael Murrieta Cid

GUANAJUATO, GTO. Diciembre 2014

Agradecimientos.

A **Dios** por permitirme concluir esta etapa de mi vida. Gracias por poner en mi camino a las personas indicadas en los momentos indicados.

A mi esposa **Dany** por siempre estar conmigo e impulsarme aún en los momentos difíciles. Gracias por recorrer esta aventura a mi lado, por brindarme tu paciencia y amor incondicional, no lo hubiera logrado sin ti.

A mis hijos **Lydany** y **Ramsés** por siempre darme esas pequeñas lecciones de vida que me ayudan a ser una mejor persona. Gracias por llenar mi vida de alegría y amor. Son el motor de mi vida.

A mis padres **Manuel y Nora** por su apoyo incondicional en todo momento. Y sobre todo por ser un excelente ejemplo de vida a seguir.

A mi hermano **Ricardo** por brindarme su apoyo y estar siempre conmigo, aun cuando la distancia no se lo permitiera de forma física. Eres un ejemplo de desarrollo profesional.

Al **Dr. Rafael Murrieta** por darme la oportunidad de ser parte de su equipo de trabajo. Gracias por todas aquellas reuniones llenas de ideas y sobre todo gracias por ser parte de mi desarrollo profesional.

Agradecimientos Institucionales.

- Al Consejo Nacional de Ciencia y Tecnología por el apoyo otorgado mediante la asignación de una beca de Doctorado con registro 208617.

- Al Centro de Investigación en Matemáticas A.C., por permitirme formar parte de uno de sus posgrados.

Índice general

Índice de figuras

Capítulo 1

Introducción y Trabajos Relacionados

La robótica es la ciencia de la percepción y manipulación del mundo físico a través de dispositivos controlados por medio de una computadora. Actualmente es cada vez más común ver robots que se integran en las actividades ligadas al ser humano, por ejemplo, los brazos robóticos usados en las lineas de ensamblaje, vehículos que son capaces de conducirse automáticamente, manipuladores que son aptos para realizar intervenciones quirúrgicas, etc. El hecho de que los robots sean cada vez más usados se debe a que pueden realizar tareas de forma más exacta o más barata que los humanos. Esto ha producido un aumento significativo en la investigación relacionada con el desarrollo de nuevos robots capaces de resolver problemas cada vez más complicados. Uno de los principales objetivos en estos desarrollos es que los robots cuenten con una gran autonomía. Los robots autónomos deben poseer la habilidad de explorar su ambiente, construir una representación de dicho ambiente, y luego usarla para navegar eficientemente en él. Para poder llevar a cabo estas tareas, el robot debe ser capaz de percibir, almacenar y procesar la información útil del ambiente en el que se encuentra.

1.1. Objetivo General

El objetivo de este trabajo es generar estrategias de movimiento para explorar un ambiente desconocido y construir un mapa del mismo. Deseamos que el mapa construido

sea fiel al ambiente donde el robot se mueve, y también deseamos que la construcción del mapa sea rápida. Suponemos que para explorar el ambiente tenemos un equipo de robots heterogéneos (diferentes capacidades de movimiento y percepción). Por lo tanto deseamos que los robots exploren el ambiente aprovechando al máximo sus capacidades. Es decir, tenemos un problema de asignación de recursos con el fin de explorar el ambiente.

1.2. Contenido del Documento

El contenido de este documento se organiza como sigue. En el Capítulo 1 presentamos el estado del arte de problemas de exploración para la construcción de mapas. En el Capítulo 2 se presentan estrategias de movimiento para la exploración de ambientes. En el Capítulo 3 se presenta la descripción del algoritmo de programación dinámica tanto para el caso determinístico como para el caso probabilístico. Esta herramienta es un componente esencial en la solución del problema abordado. Las estrategias de movimiento para la exploración y construcción de mapas bajo incertidumbre con múltiples robots heterogéneos son presentadas en el Capítulo 4. En el Catítulo 5 se presenta la implementación en el robot real de este proyecto. Las conclusiones y el trabajo futuro se presentan en el Capítulo 6. Finalmente se incluyen 6 apéndices donde se presentan a detalle conceptos utilizados a lo largo de todo el trabajo.

La investigación realizada en esta Tesis se ha reportado en la siguiente publicación:

- Motion Strategies for Exploration and Map-Building under Uncertainty with Multiple Heterogeneous Robots. L. Valentin, R. Murrieta-Cid, L. Muñoz-Gomez, R. Lopez-Padilla and M. Alencastre *Journal Advanced Robotics, Vol. 28, No. 17, pages 1133-1149, July 2014.*

Considero importante mencionar que además del proyecto de investigación doctoral que reporté en esta Tesis, durante mi estancia en el CIMAT participé en el proyecto *Estrategias de Movimiento Basadas en Apariencias para la Detección de Objetos*, investigación que se ha reportado en la siguiente publicación:

- Appearance-based Motion Strategies for Object Detection. I. Becerra, L. Valentin, R. Murrieta-Cid and J.-C. Latombe. *IEEE International Conference on Robotics and Automation, pages 6455-6461, Hong Kong, China, ICRA 2014.*

1.3. Trabajos Relacionados

En el campo de la robótica móvil, los problemas de la exploración del ambiente y la construcción de mapas son muy importantes. Los robots autónomos deben poseer la habilidad de explorar sus ambientes, construir una representación de dichos ambientes (mapas), y luego usar estas representaciones para navegar eficientemente. La estrategia para explorar un ambiente desconocido y construir una representación de dicho ambiente con un robot móvil se puede realizar de la siguiente manera: (i) el robot construye un mapa local a través de la lectura de sus sensores, (ii) un mapa global es actualizado fusionando la información entre el mapa global actual y el nuevo mapa local, y (iii) el robot se mueve hacia una meta intermedia, la cual es definida con base en ciertas propiedades.

Esto implica esencialmente que el robot debe ser capaz de navegar ante un escenario desconocido, con el fin de explorarlo y descubrir a través de la ejecución de controles, el mayor espacio posible del escenario en cuestión. La construcción automática de un modelo del ambiente ha sido un tópico ampliamente explorado en la investigación relacionada a robots móviles autónomos [2], [3], [4], [5]. La importancia de la construcción de un modelo del entorno se debe a que éstos son utilizados para realizar tareas tales como, navegación (ir de un punto A a un punto B sin colisionar con los obstáculos), búsqueda de objetos, seguimiento de blancos, etc.

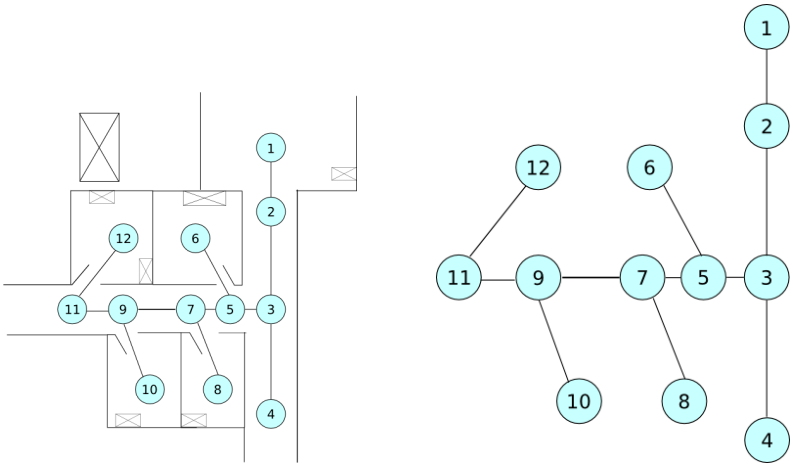
1.3.1. Modelos del Ambiente

El primer paso para generar una representación del modelo del ambiente es determinar qué modelo vamos a utilizar. En [2] se propone que los modelos del ambiente pueden ser clasificados en términos de tres categorías de representación: topológicos, geométricos y semánticos. En [6] se mencionan que los mapas topológicos [7], las rejillas de ocupación y los mapas basados en primitivas geométricas ([8] [9]) son los modelos que han sido principalmente utilizados para la construcción automática de representaciones del ambiente.

Modelos Topológicos

Una representación topológica del ambiente trata de capturar la información de conectividad entre los lugares de un ambiente, sin importar las medidas exactas. La representación está basada en una abstracción del ambiente en términos de lugares

discretos conectados. Por lo tanto, los modelos topológicos pueden ser expresados como un grafo, donde los nodos representan lugares y las aristas representan adyacencia o conectividad ente ellos [2]. En un mapa topológico el concepto de *lugar* o nodo es un área seleccionada dependiendo del ambiente. La principal ventaja de la representación topológica, es que es un modelo compacto. En la Figura 1.1 se muestra un posible mapa topológico para el ambiente.



(a) Mapa topológico (b) Grafo del mapa

Figura 1.1: Mapa topológico del ambiente

En [10] se presenta una estructura topológica de navegación (GNT) que no se basa en ninguna métrica. Esta estructura se basa en el seguimiento de brechas (discontinuidades espaciales) para la construcción de la estructura topológica.

Modelos Geométricos

Los modelos geométricos utilizan alguna primitiva geométrica para la representación del ambiente, como pueden ser puntos, líneas, polígonos, funciones polinomiales, poliedros, superficies planas, etc.; dependiendo de si el modelo del ambiente es en 2D [2] o en 3D [9].

La primitiva geométrica más popular es el segmento de recta. Este modelo supone un proceso de ajuste de curva durante su construcción. Para la obtención de las primitivas geométricas es necesario partir de los datos obtenidos de los sensores, por ejemplo de sensores de ultrasonido [11], información de telémetros láser [12] o información visual [13], [14].

Un enfoque comúnmente utilizado es la integración de la información de un telémetro láser. En la Fig. 1.2 (b) se muestra la información obtenida por un telémetro láser. En [1] el robot construye un modelo poligonal local moviéndose entre posiciones sucesivas del entorno, las cuales son seleccionadas por un planificador. Estas posiciones también pueden ser escogidas por un usuario humano. La Figura 1.3 muestra cuatro modelos parciales de un ambiente. El robot se encuentra en una localidad donde rota a cuatro orientaciones sucesivas espaciadas 90 grados. El modelo en (a) fue construido en la primera iteración. El modelo en (b) fue obtenido uniendo el modelo de (a) con el modelo local generado en la segunda orientación y así sucesivamente.

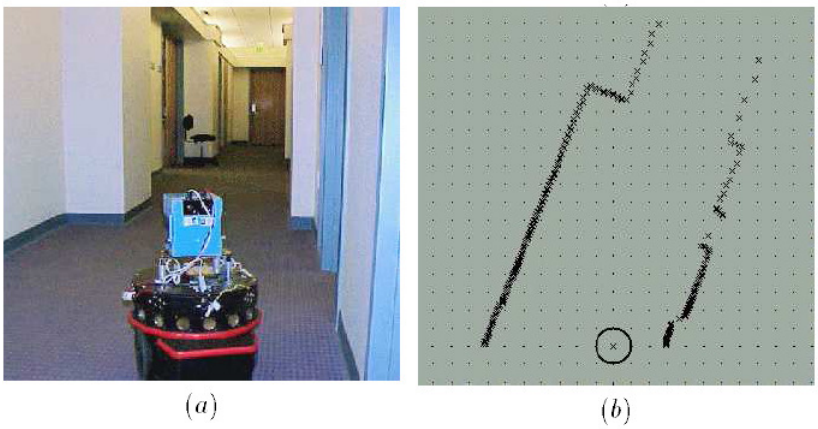


Figura 1.2: Lecturas del telémetro láser. (a) Escena. (b) Puntos capturados [1].

Además, es necesario para la construcción de un mapa geométrico global un proceso de alineamiento o registro entre los modelos locales y el modelo global, que como ya se mencionó es construido de forma incremental (ver Figura 1.3).

Adicionalmente, los datos crudos obtenidos por el sensor pueden ser agrupados en otras entidades, por ejemplo los puntos obtenidos por un láser pueden ser agrupados en segmentos de línea por medio de agrupamiento "clustering" y de un ajuste de curvas. Los segmentos de línea resultantes a su vez pueden ser agrupados en listas de segmentos que dan origen a polígonos.

Modelos Semánticos

Los modelos semánticos son una representación simbólica de alto nivel que da un significado a partes del mapa y puede estar formada por información que está relacio-

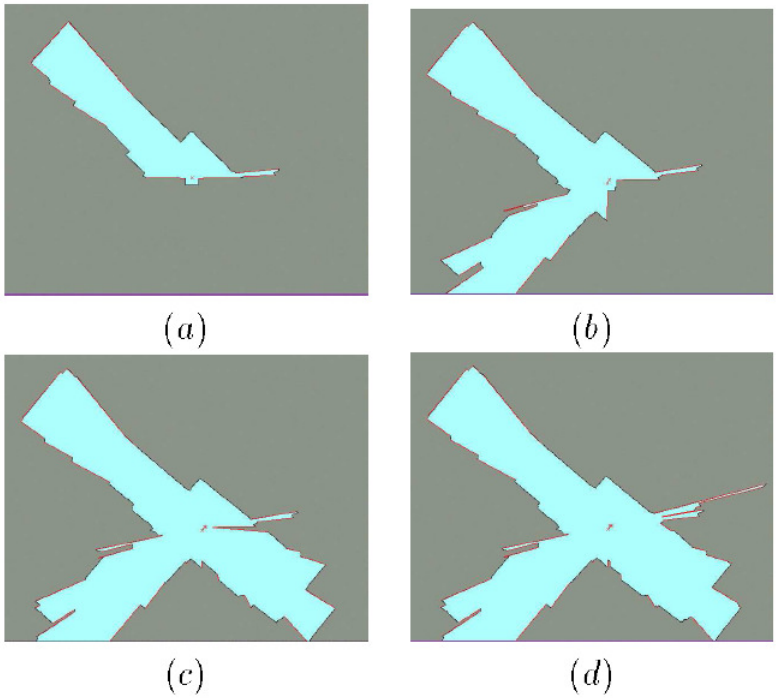


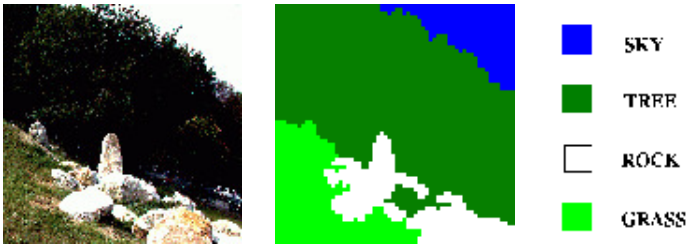
Figura 1.3: Unión de cuatro modelos parciales para una posición dada [1].

nada con objetos, propiedades del espacio y relaciones entre los objetos o el espacio [2], [15]. Una representación semántica sería por ejemplo definir un mapa etiquetando los cuartos de un ambiente por colores o funciones. La principal ventaja que presenta este tipo de mapas es que su representación puede llegar a ser más cercana a cómo los seres humanos guardamos la información sobre determinados lugares. Sin embargo, es necesario que el robot cuente con algoritmos lo suficientemente sofisticados para identificar el significado de los lugares, o la identificación de los objetos.

En la Figura 1.4 se observa un ejemplo de un modelo semántico para un ambiente de exterior [15].

1.3.2. Estrategias de Movimiento

Mucha de la investigación relacionada con la generación de un modelo del ambiente se ha enfocado en desarrollar técnicas para extraer la información relevante a partir de los datos adquiridos por sensores e integrarla en la generación de un modelo único



(a) Imagen original (b) Imagen segmentada y clasificada (c) Clase

Figura 1.4: Ejemplo de modelo semántico.

del ambiente. Sin embargo, estrategias de movimiento para explorar el ambiente han sido menos abordadas. Las estrategias de exploración para la construcción de mapas mediante un robot se pueden clasificar principalmente en dos tipos: (*i*) exploración sistemática y (*ii*) estrategias en donde la información de los sensores es tomada en cuenta para definir la siguiente localidad de sensado.

En la exploración sistemática (exploración tipo (*i*)), el robot sigue un patrón de movimiento predefinido, por ejemplo, seguir una pared [16], moverse en círculos concéntricos [17] o moviéndose en trayectorias de espirales [18].

En una exploración no sistemática (tipo (*ii*)), la información adquirida por los sensores es usada para seleccionar una siguiente posible localidad de sensado. Algunas estrategias del tipo (*ii*) utilizan la exploración basada en fronteras, originalmente propuesta por Yamauchi [19]. Esta estrategia trata de determinar una localidad de sensado para explorar el ambiente y enviar al robot a la frontera entre lo que el robot ha percibido y el mundo aún no explorado.

En [20, 21, 22], la estrategia de exploración propuesta lleva al robot hacia una configuración en la que se espera se obtenga la máxima información posible. En estos enfoques se define una función de utilidad tal que se maximice la nueva información que será adquirida en la siguiente localidad de sensado. Por otro lado, también se han propuesto diferentes trabajos en donde la estrategia de exploración se basa en la generación de localidades de sensado de manera aleatoria, las cuales son empleadas para realizar el proceso de exploracion (por ejemplo [23, 24]). El trabajo reportado en [24] presenta una técnica de exploración basada en la información adquirida por los sensores. Las localidades de sensado generadas son evaluadas a través de una función de utilidad seleccionando la localidad que maximiza dicha función [23]. La estrategia presentada en [23] está basada en el cálculo de la siguiente mejor vista (NBV[1]) y el

[1]Por sus siglas en ingles Next Best View

uso de planificación de movimientos aleatorizada. Esa investigación ha mostrado que es posible encontrar una función que refleja intuitivamente cómo debe el robot explorar el espacio. En un esquema simple, la función de evaluación debe asignar un valor grande a la posición que mejor se ajuste al compromiso entre la posible eliminación de espacio no explorado y la distancia recorrida. Así, en [23] se toma en cuenta tanto el costo asociado a la siguiente localidad de sensado, como la ganancia de la misma en términos de nueva área percibida.

En [25], una estrategia de exploración ha sido propuesta usando el concepto de entropía relativa y que permite a un robot móvil construir de forma eficiente el mapa de un ambiente desconocido. En [26], se propone un enfoque basado en toma de decisiones de múltiples criterios (MCDM por sus siglas en inglés). Un resultado importante que se reporta en ese trabajo es que tomar decisiones de forma local considerando la mayor información posible resulta en un mejor desempeño de manera global. Los autores utilizan este método en tareas de búsqueda y rescate y han evaluado el desempeño de su enfoque en ambientes simulados.

En [27], se presenta una comparación entre diferentes técnicas de exploración y construcción de mapas. Estas técnicas han sido comparadas en ambientes simulados bajo diferentes criterios como tiempo de exploración o calidad del mapa.

1.3.3. Estrategias de Movimiento para Múltiples Robots

El uso de múltiples robots tiene muchas ventajas sobre utilizar un solo robot. La cooperación de robots permite realizar una tarea más rápido que con un solo robot. Sin embargo, generar un plan para varios robots es en general más complejo que generar un plan para un solo robot. En particular si los robots son heterogéneos como se propone en esta tesis.

Algunos trabajos han sido propuestos para la exploración y construcción de mapas utilizando múltiples robots [28, 29]. En [28, 29] la ganancia de la información y el costo de la exploración son considerados simultáneamente, para definir las siguientes localidades de sensado para cada robot del equipo [28]. En [30] los autores proponen estrategias para múltiples robots usando una segmentación del entorno, para determinar el objetivo de cada robot. Esta segmentación mejora la distribución de los robots sobre el entorno.

En [6] se presenta una técnica que permite que uno o múltiples robots exploren eficientemente y construyan un modelo del entorno. Se utiliza una función de utilidad para medir la calidad de las configuraciones de sensado propuestas y un algoritmo genera

y selecciona configuraciones para la siguiente configuración de sensado. Ese trabajo considera un equipo de robots con las mismas capacidades de sensado y movimiento.

En [31], los autores proponen un método para la exploración utilizando múltiples robots basado en un proceso de decisión de Markov descentralizado (Dec-MDP por sus siglas en inglés). En ese trabajo se consideran robots homogéneos a diferencia de nuestro enfoque en donde consideramos robots heterogéneos. Además en [31], el estado de los robots es observable, mientras que en nuestro trabajo el estado de los robots es parcialmente observable. Sin embargo, una ventaja importante en [31] es que presentan un sistema descentralizado lo cual permite considerar un número grande de robots para realizar la tarea de exploración.

Algunos trabajos han propuesto exploración con múltiples robots que tienen diferentes capacidades de sensado y movimiento. El trabajo presentado en [32] considera un equipo de robots con diferentes tamaños. Durante la exploración, si un robot es demasiado grande para navegar entre obstáculos y alcanzar una localidad de sensado, entonces pide a los robots más pequeños realizar la tarea.

En [33], se presenta un estudio formal de los enfoques de asignación de tareas para múltiples robots (MRTA por sus siglas en inglés). En ese trabajo se presenta la taxonomía de problemas abordados con un enfoque de asignación de tareas. No obstante, en [33] se indica que el problema de asignación de localidades de sensado para un equipo de robots con el fin de explorar un ambiente desconocido, no está considerado en la taxonomía propuesta. Una dificultad en el problema de asignación de robots a localidades de sensado es que el costo para alcanzar la localidad "C" depende de si el robot visita primero la localidad "A" o la localidad "B".

En [34], se presenta un estudio acerca de la coordinación de múltiples robots mediante técnicas de mercadeo. En este trabajo se muestra que el uso de técnicas de mercadeo provee de un esquema versátil y poderoso para la coordinación de sistemas con múltiples robots. No obstante, en [34] también se indica que en casos donde es factible tener sistemas centralizados, el enfoque basado en mercadeo puede ser más complejo de implementar y produce resultados deficientes. En nuestro trabajo proponemos diferentes estrategias para hacer viable un enfoque centralizado. Sin embargo, el enfoque basado en mercadeo podría ser implementado bajo un esquema descentralizado, lo cual escala mejor para un mayor número de robots en comparación de los esquemas centralizados.

En [35] los robots tienen uno de dos roles: navegador o cartógrafo. Los navegadores se mueven aleatoriamente en el entorno hasta que encuentran una localidad objetivo para el cartógrafo. Después el cartógrafo se mueve a la localidad objetivo. En el trabajo mencionado previamente cada robot sigue un rol específico y predefinido de acuerdo a su tipo. En nuestro trabajo, se propone una estrategia de movimiento para la exploración

y construcción de mapas de ambientes desconocidos considerando las capacidades de los robots, pero no se asignan roles predefinidos a estos robots. En lugar de eso, modelamos las capacidades de los robots probabilísticamente.

Capítulo 2

Trabajo previo: Planificación de Movimientos para Estrategias de Exploración y Construcción de Mapas

El proceso de exploración de un entorno desconocido usando un robot móvil con un sensor, durante un tiempo t, es esbozado de la siguiente manera:

(a) Construir un mapa local, e_t, que representa la porción del entorno que rodea al robot en una configuración q_t.

(b) Actualizar un mapa global, M_t, de acuerdo a la nueva información adquirida e_t, obteniendo M_{t+1}.

(c) Usar una estrategia de exploración para determinar la siguiente configuración de observación q_{t+1}, alcanzarla, y empezar de nuevo desde (a).

Este proceso de exploración es presentado en [6] y se realiza usando el Algoritmo presentado en la Sección 2.2. En ese trabajo se abordan problemas relacionados a la incertidumbre en el sensado y el control, sin embargo, su principal interés es en la planificación de estrategias de exploración. A continuación presentaremos un resumen del enfoque realizado en [6] para posteriormente presentar nuestra propuesta de métodos de exploración para construcción de mapas.

2.1. Sensado e integración de la percepción en un mapa global

En [6] se supone que el robot puede distinguir entre referencias perceptuales (landmarks) y primitivas geométricas del ambiente. Las referencias tienen una identidad (están ligadas a una clase, e.g., la pelota roja, el póster de Angelina Jolie, etc) mientras que las primitivas son entidades geométricas sin una identificación específica (e.g., segmentos de líneas, esquinas, etc.). La detección e identificación de referencias es útil para localizar al robot en un mismo marco global y para unir diferentes vistas del entorno. La fusión se vuelve más fácil porque estas referencias se vuelven características "pivote" para alinear las diferentes vistas. Las referencias son útiles para determinar la posición absoluta del robot.

Se usa la información de un telémetro láser para modelar los obstáculos a partir de los puntos que el telémetro láser proporciona. En particular, se generan poli-líneas a partir de la información obtenida del telémetro láser. El proceso para la generación de estas poli-líneas es descrito en la sección 2.1.2.

A fin de alinear mapas locales (resultado de nuevas percepciones) con el mapa global (resultado de la construcción incremental), se utiliza la distancia parcial de Hausdorff. La distancia de Hausdorff es calculada sobre el conjunto original de datos (puntos láser), pero el mapa almacenado corresponde a listas de segmentos de línea (ver sección 2.1.3).

2.1.1. Polígono de Visibilidad

La exploración del ambiente se realiza de una forma no sistemática, de tal manera que para generar un modelo del ambiente, se utiliza la información proveniente de un sensor (e.g. un telémetro laser). El sensor genera como información un conjunto de puntos 2D los cuales describen a los obstáculos del ambiente. Cada punto es denotado como $s_j \in \mathbb{R}^2$ y el conjunto es denotado como:

$$S = \bigcup_j s_j.$$

Basados en este conjunto, los obstáculos pueden ser modelados con poli-líneas, mediante técnicas de ajuste de curvas [6] sobre los puntos sensados. Este paso es necesario para poder calcular la visibilidad.

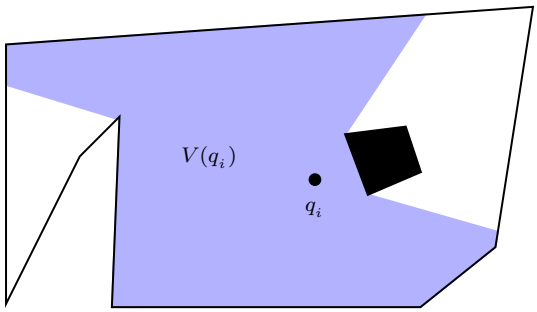


Figura 2.1: Polígono de Visibilidad

El cálculo de la visibilidad es usado para definir las fronteras entre lo conocido y lo no conocido del ambiente.

La ubicación del robot es representada por un vector con tres componentes (x, y, θ). (x, y) representan las coordenadas que ubican al robot con respecto a un marco global y θ representa la orientación del robot, medida con respecto al eje x del marco global de referencia. Así la configuración del robot i se denota $q_i \in \mathbb{R}^2 \times SO(2)$. Además denotamos con $V(q_i)$ al polígono de visibilidad del robot i en la configuración q_i (ver Figuar 2.1).

Se considera al robot no como un punto sino como un disco. Denotamos a $F(q_i)$ como el polígono de visibilidad reducido por el radio del robot. Para un conjunto de robots se defina a V_{tot} como la unión de todos los polígonos de visibilidad individuales, es decir, región de visibilidad total, y a F_{tot} como la unión de todas las regiones en donde el robot puede mover

$$V_{tot} = \bigcup_i V(q_i) \qquad \text{y} \qquad F_{tot} = \bigcup_i F(q_i).$$

Es posible utilizar el polígono de visibilidad modificado para garantizar que el robot se puede mover sin chocar con obstáculos.

Una frontera –*eje libre*– se define como el eje entre la región de visibilidad $V(q_i)$ y el área no vista del ambiente. Explorar el espacio desconocido es equivalente a enviar un robot a un eje libre.

Una vez que se ha determinado la visibilidad es posible calcular el grafo de visibilidad reducido (ver el Apéndice B) sobre el espacio de configuraciones (ver el Apéndice A); y este grafo es utilizado para que el robot se mueva.

2.1.2. Extracción de Poli-líneas

El procedimiento utilizado para la extracción de poli-líneas presentado en [6] es detallado a continuación.

Se generan poli-líneas a partir de los datos adquiridos por el sensor, los cuales están almacenados en una lista de datos ordenada en coordenadas polares (r, θ). r es la distancia entre el obstáculo sensado y el sensor montado en el robot. θ es el ángulo con respecto al eje del robot perpendicular a sus ruedas.

El ajuste de rectas es realizado en dos pasos. El primer paso es generar grupos de puntos de tal forma que la distancia entre dos puntos consecutivos sea pequeña. Enseguida, al igual que en los trabajos [1, 23], se realiza la siguiente transformación $u = \cos(\theta)/\sin(\theta)$, $v = 1/r \sin(\theta)$. Después, se utiliza un método de mínimos cuadradas combinado con una técnica de divide y vencerás para realiza el ajuste de las rectas.

La transformación (u, v), se utiliza por dos razones:

1) (u, v) preserva el hecho de que una de las coordenadas es prácticamente libre de error. En efecto, note que el láser es mucho más preciso en cuanto a los valores de θ que de r. La precisión sobre θ se debe a la electrónica del dispositivo (el haz láser se dirige por el movimiento de un espejo, con un motor muy preciso), mientras que r, dado que corresponde a la distancia a los obstáculos, en general es más ruidosa. En el nuevo espacio (u, v), se preserva esta propiedad, ya que u puede considerarse libre de error.

2) Se utiliza el espacio (u, v) para poder realizar un ajuste lineal. El problema es transformado en un ajuste lineal $v = a + bu$ (el cual mapea a $bx + ay = 1$ en el espacio Cartesiano (x, y)) [23]. Varios algoritmos bien conocidos existen para determinar poli-líneas en el espacio (u, v) [36]. Los puntos láser en cada cluster son ajustados a una poli-línea tal que cada punto se encuentre dentro de un ϵ de cada segmento de linea de la poli-linea. El método de ajuste de líneas aprovecha que los datos entregados por el láser en coordenadas polares, satisfacen una restricción de orden a lo largo de la coordenada libre de error θ.

En el espacio (u, v) es posible determinar una cota del error que depende de la posición. Esta cota está dada por $e = \epsilon v \sqrt{a^2 + b^2}$, así, es posible garantizar que cada punto (x, y) está a ϵ de la poli-línea generada.

2.1.3. Fusión de datos

Para llevar a cabo la fusión de los datos entre una región previamente explorada y una nueva se utiliza la distancia parcial de Hausdorff.

La distancia de Hausdorff se calcula sobre los datos originales del sensor de las polilíneas previamente calculadas. La distancia de Hausdorff permite medir la semejanza entre dos conjuntos de puntos.

Dados dos conjuntos de puntos P y Q, la distancia de Hausdorff se define como (ver [37]),

$$H(P,Q) = \max(h(P,Q), h(Q,P)) \tag{2.1}$$

donde,

$$h(P,Q) = \max_{p\in P} \min_{q\in Q} \| p - q \| \tag{2.2}$$

y $\| \cdot \|$ es una norma para medir la distancia entre los puntos p y q. La función $h(P,Q)$ es la distancia del conjunto P al conjunto Q. Valores pequeños de $h(P,Q)$ implican que cada punto en P se encuentra cerca a un punto en Q. La distancia de Hausdorff mide el grado en que cada punto de P se encuentra cerca a un punto en Q y viceversa. Al calcular la distancia de Hausdorff, se obtiene el punto más desigual entre los dos conjuntos comparados. Este proceso es muy sensible a la presencia de ruido. Por esta razón es mejor utilizar una medida más robusta, la cual reemplaza el cálculo de la maximización por el cálculo de la media de los datos. Esta medida (distancia parcial de Hausdorff) se define como,

$$h = \underset{p\in P}{M} \min_{q\in Q} \| p - q \| \tag{2.3}$$

donde $\underset{p\in P}{M} f(p)$ denota la media estadística de $f(p)$ sobre el conjunto P.

En la construcción incremental del mapa, a fin de facilitar el alineamiento (registro) del mapa global con la nueva información adquirida (nuevo mapa local), es común utilizar el desplazamiento y rotación que se le ordenó al robot (ese movimiento se transforma en una rotación y traslación usando matrices homogéneas), como una condición inicial conveniente, al método de registro, para encontrar la mejor alineación (rotación y traslación) que fusiona los mapas.

2.2. Algoritmo para la construcción del mapa

En esta sección se describe el procedimiento propuesto en [6] para la construcción de mapas.

Los Algoritmos A y B muestran el procedimiento de la construcción del mapa. Además, en el procedimiento utilizado en [6] se plantea una función de utilidad denotada como $\mathcal{T}$, la cual evalúa posiciones de sensado candidatas que han sido generadas a través de muestreo aleatorio. Sólo son consideradas aquellas localidades de sensado (muestras) que se encuentren dentro de la región de visibilidad. Las configuraciones previamente seleccionadas son guardadas de manera que puedan ser utilizadas para alcanzar otras muestras en el futuro.

Algoritmo A EXPLORACION($\mathcal{R}$,$\mathcal{W}$)

Entrada:
$\mathcal{R}$: Configuración del robot
$\mathcal{W}$: Entorno
Salida:
$\mathcal{M}$: Mapa del entorno

P_n: Percepción actual del robot
U_f: Lista de etiquetas de ejes libres
$U_f \leftarrow \{1\}$
while $U_f \neq \emptyset$ **do**
  $P_n \leftarrow$TOMA_PERCEPCION($\mathcal{R}$,$\mathcal{W}$)
  $\mathcal{M}$.integrar(P_n)
  $U_f \leftarrow$ ID_AREAS_NO_EXPLORADAS($\mathcal{M}$)
  if $U_f \neq 0$ **then**
    $[x, y, \theta] \leftarrow$SELECC_NUEVA_CONF($U_f, \mathcal{R}, \mathcal{M}$)
    $\mathcal{R}$.mover(x, y, θ)
  else
    Regresar $\mathcal{M}$, OK
    $U_f \leftarrow \emptyset$
  end if
end while

2.2.1. Varios pasos adelante en la exploración

Supongamos que en un instante de tiempo dado, el robot está en una cierta configuración, $q_{ini} = (x_{ini}, y_{ini}, \theta_{ini})$, y que aún hay n fronteras restantes (ejes libres) a visitar. Para cada una de estas fronteras, se genera una seria de localidades de sensado. Existen al menos dos posibilidades para mover al robot: un plan corto, un paso adelante, o uno largo, varios pasos adelante.

Algoritmo B SELECC_NUEVA_CONF($U_f, \mathcal{R}, \mathcal{M}$)

Entrada:

$\mathcal{R}$: Configuración del robot

$\mathcal{M}$: Mapa del entorno

U_f: Lista de etiquetas de ejes libres (o bordes libres)

Salida:

q_{nbv}: Configuración del robot para la siguiente mejor vista

n: número de muestras por cada eje libre

l_n: Lista de configuraciones del robot por cada muestra

L_q: Lista de todas las posiciones del robot candidatas

1: **for** $u_f \in U_f$ **do**
2: $l_n \leftarrow$MUESTRAS$(n, u_f, \mathcal{R}, \mathcal{M})$
3: $L_q = L_q \cup P_n$
4: **end for**
5: $q_{nbv} \leftarrow$MAXIMIZA$(\mathcal{T}, L_q)$
6: Regresar q_{nbv}

Para un paso adelante se planifica solo una configuración a ser visitada y se ejecuta el movimiento, para varios pasos adelante, se determina un orden de visita de varias configuraciones antes de ejecutar el plan.

Todas las posibles trayectorias que el robot pude seguir para explorar los ejes libres se pueden modelar usando un árbol. Cada nodo del árbol representa una configuración del robot, y cada arista representa un paso en el tiempo. Además, en este caso se ha impuesto una restricción para la construcción del árbol: si un eje libre ha sido explorado en un paso previo, el robot no puede regresar a él. Es decir, si por ejemplo se tiene un nodo en el nivel 1, todas las ramas que cuelgan de él no lo contendrán. Por consiguiente, las ramas que cuelgan del nodo no tendrán nodos que representen configuraciones que estén en el mismo eje libre. El nodo raíz tendrá n niveles colgando de él de acuerdo a los n ejes libres que serán visitados.

Para la construcción del árbol necesitamos conocer todas las trayectorias que pueden ser posiblemente formadas permutando cada configuración del robot sin repetir configuraciones visitadas. El costo de cada trayectoria entre dos configuraciones en dos ejes libres diferentes es calculado usando una función de utilidad. Una vez construido el árbol, se puede usar un algoritmo de búsqueda para encontrar las configuraciones que el robot debe alcanzar y así visitar todos los ejes libres con utilidad máxima.

2.2.2. Heurística para la reducción de ramas

Supongamos que en un tiempo dado, el robot tiene que visitar n ejes libres, y que para cada uno de estos ejes libres hay m configuraciones. Entonces, la cantidad de nodos que cuelgan de la raíz será de $n \cdot m$, de manera que si m y n son grandes, la búsqueda de una trayectoria óptima es intratable. Para tratar con este problema, se puede reducir el espacio de búsqueda podando el árbol mediante un algoritmo de branch-and-bound [38]. La idea del algoritmo es construir un árbol hasta un nivel w, en donde $w < n$. En ese punto, seleccionar el nodo con la máxima utilidad y continuar expandiendo el árbol desde ese nodo. Esta idea es representada en la Fig. 2.2 (imagen tomada de [6]).

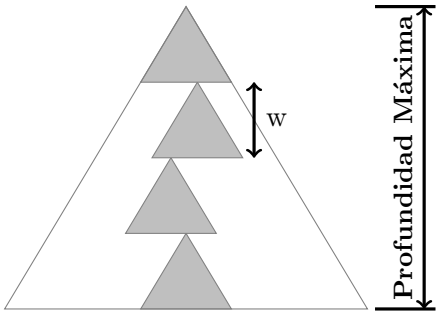


Figura 2.2: Heurística para la reducción de ramas. El triángulo grande representa el árbol completo hasta la profundidad máxima, la parte gris representa el árbol reducido.

En el trabajo presentado en [6] se suponía que los robots eran homogéneos, es decir, mismas capacidades de sensado y movimiento. Sin embargo, en nuestro trabajo se supone que los robots son heterogéneos. Además, consideramos un modelo probabilístico a través de un POMDP que es optimizado usando programación dinámica estocástica. Estos dos aspectos no eran considerados en los trabajos previos. Los detalles de estas aportaciones se darán en el Capítulo 4.

Capítulo 3

Programación Dinámica

La mayoría del material presentado en este capítulo, constituye un resumen del capítulo 5 de [39].

La creación de algoritmos de planificación en tiempo discreto se define a partir del uso del modelo de espacio de estados. La idea básica es que a cada situación distinta del mundo se le llama *estado*, denotado por x, y al conjunto de todos los posibles estados se le conoce como *espacio de estados*, X. Para el caso de planificación en tiempo discreto, será importante que este conjunto sea contable; en la mayoría de los casos será finito. Es importante que X sea lo suficientemente descriptivo para incluir toda la información relevante para resolver la tarea.

El mundo puede ser transformado a través de la aplicación de *acciones* las cuales son elegidas por un planificador. Cada acción, u, cuando es aplicada al estado actual, x, produce un nuevo estado, x', según lo especifique una *función de transición de estados*, f. Es conveniente usar f para expresar una *ecuación de transición de estados*,

$$x' = f(x, u). \tag{3.1}$$

Denotemos con $U(x)$ el espacio de acciones para cada estado, el cual representa al conjunto de todas las acciones que pueden aplicarse desde x. Para un x distinto, $x' \in X$, $U(x)$ y $U(x')$ no son necesariamente disjuntas; la misma acción pude ser aplicada en múltiples estados. Por lo tanto, resulta conveniente definir al conjunto U de todas las posibles acciones sobre todos los posibles estados,

$$U = \bigcup_{x \in X} U(x). \tag{3.2}$$

Como parte del problema de planificación, el conjunto $X_G \subset X$ de estados objetivo es definido. La tarea del algoritmo de planificación es encontrar una secuencia finita de acciones que al ser aplicadas, transformen al estado inicial x_I en algún estado en X_G. Es posible resumir el modelo como,

- Un *espacio de estados* no vacío X, el cual es un conjunto finito o contable de *estados*.
- Para cada $x \in X$, un espacio finito de acciones $U(x)$.
- Una *función de transición de estados*, f que produce un estado $f(x, u) \in X$ para cada $x \in X$ y $u \in U(x)$. La *ecuación de transición de estados* se deriva de f como $x' = f(x, u)$.
- Un estado inicial $x_I \in X$.
- Un conjunto objetivo $X_G \subset X$.

3.1. Planificación óptima discreta

La programación dinámica (D.P. por sus siglas en inglés) es una herramienta para resolver problemas complejos dividiéndolos en sub-problemas más simples. La idea principal detrás de la D.P. es muy simple. En general, para resolver un problema dado, es necesario resolver diferentes partes del problema, y luego combinar las soluciones y llegar a la solución final.

La D.P. es tanto un método de optimización, como un método de programación. En ambos contextos, la D.P. trata de dividir un problema muy complicado en sub-problemas simples de una forma recursiva. La D.P. se basa en el *principio de optimalidad de Bellman que dicta que, dada una secuencia óptima de decisiones, toda subsecuencia de ella es, a su vez, óptima.*

Es necesario definir algunos conceptos para poder explicar el funcionamiento de la D.P. Sea π_k un plan de $k-$pasos, el cual es una secuencia de $(u_0, u_1, \ldots, u_k)$ $k+1$ acciones. Si π_k y el estado inicial del sistema (x_0) son dados, entonces una secuencia de estados, $(x_0, x_1, \ldots, x_{k+1})$, puede ser derivados utilizando la función de transición de

estados, f. Inicialmente $x_0 = x_I$, y cada estado subsecuente puede obtenerse mediante $x_{k+1} = f(x_k, u_k)$. Otro elemento sumamente importante en D.P. es la función de costo. Sea L una función de costo (o pérdida) aditiva, la cual es aplicada en cada instante del plan, π_k. Esto significa que la secuencia $(u_0, u_1, \ldots, u_k)$ de acciones y la secuencia $(x_0, x_1, \ldots, x_{k+1})$ de estados, debe aparecer en la expresión de L. Denotemos a x_F como el estado final. La función de costo es,

$$L(\pi_k) = l_F(x_F) + \sum_{k=1}^{K} l(x_k, u_k), \tag{3.3}$$

en donde K es el horizonte de planificación.

El término de costo $l(x_k, u_k)$ arroja un valor real para cada $x_k \in X$ y $u_k \in U(x_k)$.

El término $l_F(x_F)$ se encuentra fuera de la sumatoria y se define como $l_F(x_F) = 0$ si $x_F \in X_G$ y $l_F(x_F) = \infty$ de otra forma.

Ahora la tarea es encontrar un plan que minimice (o maximice) L. La observación principal que podemos hacer es que porciones del plan óptimo, son también óptimas.

El principio de optimalidad lidia directamente con un algoritmo iterativo, llamado iteración por valor. La idea es calcular de forma iterativa el costo óptimo de pasar de un estado a otro. Existen dos enfoques para el algoritmo de iteración por valor, iteración por valor hacia atrás e iteración por valor hacia adelante.

3.1.1. Iteración por valor hacia atrás

La clave para generar planes óptimos extensos a partir de planes cortos reside en la construcción de la función de costo sobre los estados X. Sea G_k^* el costo acumulado de los estados x_k a x_F bajo la ejecución del plan óptimo,

$$G_k^*(x_k) = \max_{u_k, \ldots, u_K} \left\{ \sum_{i=k}^{K} l(x_i, u_i) + l_F(x_F) \right\} \tag{3.4}$$

mientras que el costo del último estado es,

$$G_F^*(x_F) = l_F(x_F). \tag{3.5}$$

Como podemos observar en la primera iteración, G_F^* se obtiene directamente de l_F. En la segunda iteración, G_K^* se calcula para cada $x_K \in X$ como,

$$G_K^*(x_K) = \max_{u_K} \Big\{ l(x_K, u_K) + l_F(x_F, u_F) \Big\}. \tag{3.6}$$

Ya que $l_F = G_F^*$ y $x_F = f(x_K, u_K)$, si sustituimos en la Ecuación 3.6 obtendremos,

$$G_K^*(x_K) = \max_{u_K}\Big\{l(x_K, u_K) + G_F^*(f(x_K, u_K))\Big\}, \tag{3.7}$$

lo cual es fácil de calcular para cada $x_K \in X$. Con esto calculamos el costo para un plan óptimo a un paso adelante.

De igual forma, podemos calcular G_k^* una vez que se tiene G_{k+1}^*. De la Ecuación 3.4,

$$G_k^*(x_k) = \max_{u_k,\ldots,u_K}\left\{\sum_{i=k}^{K} l(x_i, u_i) + l_F(x_F)\right\}$$

podemos obtener,

$$G_k^*(x_k) = \max_{u_k,\ldots,u_K}\left\{l(x_k, u_k) + \sum_{i=k+1}^{K} l(x_i, u_i) + l_F(x_F)\right\} \tag{3.8}$$

de donde resulta,

$$G_k^*(x_k) = \max_{u_k}\left\{\max_{u_{k+1},\ldots,u_K}\left\{l(x_k, u_k) + \sum_{i=k+1}^{K} l(x_i, u_i) + l_F(x_F)\right\}\right\}. \tag{3.9}$$

Podemos observar que el término $l(x_k, u_k)$ pude salir de la primer maximización, de tal forma que,

$$G_k^*(x_k) = \max_{u_k}\left\{l(x_k, u_k) + \max_{u_{k+1},\ldots,u_K}\left\{\sum_{i=k+1}^{K} l(x_i, u_i) + l_F(x_F)\right\}\right\}. \tag{3.10}$$

A partir de la Ecuación 3.10 es claro que la maximización interior es exactamente la definición de la función G_{k+1}^*, por lo tanto, al sustituir obtenemos,

$$G_k^*(x_k) = \max_{u_k}\Big\{l(x_k, u_k) + G_{k+1}^*(x_{k+1})\Big\} \tag{3.11}$$

donde $x_{k+1} = f(x_k, u_k)$. Nótese que el lado derecho de 3.11 depende unicamente de x_k, u_k y G_{k+1}^*.

Resumiendo, la iteración por valor procede de la siguiente manera,

$$G_F^* \rightarrow G_K^* \rightarrow G_{K-1}^* \cdots G_k^* \rightarrow G_{k-1}^* \cdots G_1^* \rightarrow G_0^*. \tag{3.12}$$

Ejemplo

En la Figura 3.1 se muestra una representación por medio de un grafo de un problema de planificación, en donde $X = \{a, b, c, d, e\}$ son los estados del sistema. Supongamos que $K = 4$ y que $x_I = a$ y $X_G = \{d\}$. El objetivo es calcular el camino (secuencia de nodos) o plan de menor costo para ir de "d" a cualquier nodo inicial en retro-tiempo. Note que el camino de menor costo para ir de "d"a "a" está incluido en este conjunto. Note también que la salida del algoritmo que nos interesa *es la secuencia de controles " u_k "para realizar la tarea.*

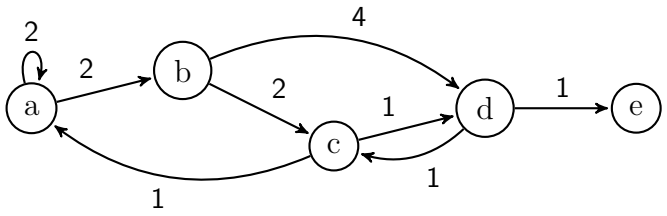


Figura 3.1: Grafo de cinco estados. Cada nodo es un estado, y los ejes representan la transición entre estados. Los pesos sobre los ejes representan $l(x_k, u_k)$.

El costo en el estado final, G_5^* está dado como,

$$\begin{aligned} G_5^*(a) &= \infty \\ G_5^*(b) &= \infty \\ G_5^*(c) &= \infty \\ G_5^*(d) &= 0 \\ G_5^*(e) &= \infty. \end{aligned}$$

Para calcular el costo mínimo para los demás instantes de tiempo partiremos de la Ecuación 3.11,

$$G_k^*(x_k) = \min_{u_k}\Big\{ l(x_k, u_k) + G_{k+1}^*(x_{k+1}) \Big\}.$$

Por lo tanto, para $k = 4$ tenemos,

$$G_4^*(x_4) = \min_{u_4}\Big\{ l(x_4, u_4) + G_5^*(x_5) \Big\}$$

entonces, para $x_4 = a$,

$$\begin{aligned} G_4^*(a) &= \min_{u_4}\Big\{ l(a,u_4) + G_5^*(x_5)\Big\} \\ &= \min\Big\{ l(a,u_4^{(1)}) + G_5^*(a)\ ,\ l(a,u_4^{(2)}) + G_5^*(b)\Big\} \\ &= \min\Big\{ 2+\infty\ ,\ 2+\infty\Big\} \\ &= \min\Big\{ \infty\ ,\ \infty\Big\} \end{aligned}$$

por lo tanto,

$$G_4^*(a) = \infty.$$

Para $x_4 = b$ tenemos,

$$\begin{aligned} G_4^*(b) &= \min_{u_4}\Big\{ l(b,u_4) + G_5^*(x_5)\Big\} \\ &= \min\Big\{ l(b,u_4^{(1)}) + G_5^*(c)\ ,\ l(b,u_4^{(2)}) + G_5^*(d)\Big\} \\ &= \min\Big\{ 1+\infty\ ,\ 4+0\Big\} \\ &= \min\Big\{ \infty\ ,\ 4\Big\} \end{aligned}$$

por lo tanto,

$$G_4^*(b) = 4 \quad (\text{control } b \to d).$$

Para $x_4 = c$ tenemos,

$$\begin{aligned} G_4^*(c) &= \min_{u_4}\Big\{ l(c,u_4) + G_5^*(x_5)\Big\} \\ &= \min\Big\{ l(c,u_4^{(1)}) + G_5^*(a)\ ,\ l(c,u_4^{(2)}) + G_5^*(d)\Big\} \\ &= \min\Big\{ 1+\infty\ ,\ 1+0\Big\} \\ &= \min\Big\{ \infty\ ,\ 1\Big\} \end{aligned}$$

por lo tanto,

$$G_4^*(c) = 1 \quad (\text{control } c \to d).$$

Para $x_4 = d$ tenemos,

$$\begin{aligned} G_4^*(d) &= \min_{u_4}\Big\{ l(d,u_4) + G_5^*(x_5) \Big\} \\ &= \min\Big\{ l(d,u_4^{(1)}) + G_5^*(c)\ ,\ l(d,u_4^{(2)}) + G_5^*(e) \Big\} \\ &= \min\Big\{ 1+\infty\ ,\ 1+\infty \Big\} \\ &= \min\Big\{ \infty\ ,\ \infty \Big\} \end{aligned}$$

por lo tanto,

$$G_4^*(d) = \infty.$$

Para $x_4 = e$ tenemos,

$$\begin{aligned} G_4^*(e) &= \min_{u_4}\Big\{ l(e,u_4) + G_5^*(x_5) \Big\} \\ &= \min\Big\{ \infty \Big\} \end{aligned}$$

por lo tanto,

$$G_4^*(e) = \infty.$$

A partir del cálculo de estas ganancias ya es posible tomar una decisión sobre el control que se va a aplicar. En este caso como el estado final está dado por el estado d entonces el control que nos lleva con costó mínimo a " d " en un paso es el control $c \to d$.

Con $k = 3$ tenemos,

$$G_3^*(x_3) = \min_{u_3}\Big\{ l(x_3,u_3) + \underbrace{\min_{u_4}\big\{ l(x_4,u_4) + G_5^*(x_5) \big\}}_{G_4^*(x_4)} \Big\}$$

por lo tanto,

$$G_3^*(x_3) = \min_{u_3}\Big\{ l(x_3,u_3) + G_4^*(x_4) \Big\}.$$

Entonces, para $x_3 = a$

$$\begin{aligned} G_3^*(a) &= \min_{u_3}\Big\{ l(a,u_3) + G_4^*(x_4) \Big\} \\ &= \min\Big\{ l(a,u_3^{(1)}) + G_4^*(a)\ ,\ l(a,u_3^{(2)}) + G_4^*(b) \Big\} \\ &= \min\Big\{ 2+\infty\ ,\ 2+4 \Big\} \\ &= \min\Big\{ \infty\ ,\ 6 \Big\} \end{aligned}$$

por lo tanto,

$$G_3^*(a) = 6 \quad (\text{control } a \to b).$$

Para $x_3 = b$

$$\begin{aligned} G_3^*(b) &= \min_{u_3}\Big\{ l(b,u_3) + G_4^*(x_4) \Big\} \\ &= \min\Big\{ l(b,u_3^{(1)}) + G_4^*(c)\ ,\ l(b,u_3^{(2)}) + G_4^*(d) \Big\} \\ &= \min\Big\{ 1+1\ ,\ 4+\infty \Big\} \\ &= \min\Big\{ 2\ ,\ \infty \Big\} \end{aligned}$$

por lo tanto,

$$G_3^*(b) = 2 \quad (\text{control } b \to c).$$

Para $x_3 = c$

$$\begin{aligned} G_3^*(c) &= \min_{u_3}\Big\{ l(c,u_3) + G_4^*(x_4) \Big\} \\ &= \min\Big\{ l(c,u_3^{(1)}) + G_4^*(a)\ ,\ l(c,u_3^{(2)}) + G_4^*(d) \Big\} \\ &= \min\Big\{ 1+\infty\ ,\ 1+\infty \Big\} \\ &= \min\Big\{ \infty\ ,\ \infty \Big\} \end{aligned}$$

por lo tanto,

$$G_3^*(c) = \infty.$$

Para $x_3 = d$

$$\begin{aligned} G_3^*(d) &= \min_{u_3}\Big\{ l(d,u_3) + G_4^*(x_4) \Big\} \\ &= \min\Big\{ l(d,u_3^{(1)}) + G_4^*(c)\ ,\ l(d,u_3^{(2)}) + G_4^*(e) \Big\} \\ &= \min\Big\{ 1+1\ ,\ 1+\infty \Big\} \\ &= \min\Big\{ 2\ ,\ \infty \Big\} \end{aligned}$$

por lo tanto,

$$G_3^*(d) = 2 \quad (\text{control } d \to c).$$

Para $x_3 = e$

$$\begin{aligned} G_3^*(e) &= \min_{u_3}\Big\{ l(e,u_3) + G_4^*(x_4) \Big\} \\ &= \min\Big\{ \infty \Big\} \end{aligned}$$

por lo tanto,

$$G_3^*(e) = \infty.$$

En este instante el control que necesitamos es aquel que nos lleve a "c", por lo que al revisar qué control con costo mínimo nos lleva al estado "c" podemos observar que existen dos posibilidades, el control $b \to c$ y el control $d \to c$ ambos con costo mínimo de 2.

Con $k = 2$ tenemos,

$$G_2^*(x_2) = \min_{u_2}\Bigg\{ l(x_2,u_2) + \min_{u_3}\Big\{ l(x_3+u_3) + \underbrace{\min_{u_4}\big\{ l(x_4,u_4) + G_5^*(x_5) \big\}}_{G_4^*(x_4)} \Big\} \Bigg\}$$

$$G_2^*(x_2) = \min_{u_2}\Bigg\{ l(x_2,u_2) + \underbrace{\min_{u_3}\Big\{ l(x_3+u_3) + G_4^* \Big\}}_{G_3^*(x_3)} \Bigg\}$$

por lo tanto,

$$G_2^*(x_2) = \min_{u_2}\Big\{ l(x_2,u_2) + G_3^*(x_3) \Big\}.$$

Entonces, para $x_2 = a$

$$\begin{aligned} G_2^*(a) &= \min_{u_2}\Big\{ l(a,u_2) + G_3^*(x_3) \Big\} \\ &= \min\Big\{ l(a,u_2^{(1)}) + G_3^*(a)\ ,\ l(a,u_2^{(2)}) + G_3^*(b) \Big\} \\ &= \min\Big\{ 2+6\ ,\ 2+2 \Big\} \\ &= \min\Big\{ 8\ ,\ 4 \Big\} \end{aligned}$$

por lo tanto,

$$G_2^*(a) = 4 \quad (\text{control } a \to b).$$

Para $x_2 = b$

$$\begin{aligned} G_2^*(b) &= \min_{u_2}\Big\{ l(b,u_2) + G_3^*(x_3) \Big\} \\ &= \min\Big\{ l(b,u_2^{(1)}) + G_3^*(c)\ ,\ l(b,u_2^{(2)}) + G_3^*(d) \Big\} \\ &= \min\Big\{ 2+\infty\ ,\ 4+2 \Big\} \\ &= \min\Big\{ \infty\ ,\ 6 \Big\} \end{aligned}$$

por lo tanto,

$$G_2^*(b) = 6 \quad (\text{control } b \to d).$$

Para $x_2 = c$

$$\begin{aligned} G_2^*(c) &= \min_{u_2}\Big\{ l(c,u_2) + G_3^*(x_3) \Big\} \\ &= \min\Big\{ l(c,u_2^{(1)}) + G_3^*(a)\ ,\ l(c,u_2^{(2)}) + G_3^*(d) \Big\} \\ &= \min\Big\{ 1+6\ ,\ 1+2 \Big\} \\ &= \min\Big\{ 7\ ,\ 3 \Big\} \end{aligned}$$

por lo tanto,

$$G_2^*(c) = 3 \quad (\text{control } c \to d).$$

Para $x_2 = d$

$$\begin{aligned} G_2^*(d) &= \min_{u_2}\Big\{ l(d,u_2) + G_3^*(x_3) \Big\} \\ &= \min\Big\{ l(d,u_2^{(1)}) + G_3^*(c)\ ,\ l(d,u_2^{(2)}) + G_3^*(e) \Big\} \\ &= \min\Big\{ 1+\infty\ ,\ 1+\infty \Big\} \\ &= \min\Big\{ \infty\ ,\ \infty \Big\} \end{aligned}$$

por lo tanto,

$$G_2^*(d) = \infty.$$

Para $x_2 = e$

$$\begin{aligned} G_2^*(e) &= \min_{u_2}\Big\{ l(e,u_2) + G_3^*(x_3) \Big\} \\ &= \min\Big\{ \infty \Big\} \end{aligned}$$

por lo tanto,

$$G_2^*(e) = \infty.$$

Ya que en el tiempo $k = 3$ se conservaron dos controles como posibilidades entonces para este instante de tiempo debemos seleccionar el o los controles que nos llevan a " b" o " d" . Estos controles son $a \to b$, $b \to d$ y $c \to d$.

Con $k = 1$ tenemos,

$$G_1^*(x_1) = \min_{u_1}\Big\{ l(x_1,u_1) + \min_{u_2}\Big\{ l(x_2,u_2) + \min_{u_3}\Big\{ l(x_3+u_3) + \underbrace{\min_{u_4}\big\{ l(x_4,u_4) + G_5^*(x_5)\big\}}_{G_4^*(x_4)} \Big\}\Big\}\Big\}$$

$$G_1^*(x_1) = \min_{u_1}\Big\{ l(x_1,u_1) + \min_{u_2}\Big\{ l(x_2,u_2) + \underbrace{\min_{u_3}\Big\{ l(x_3+u_3) + G_4^*(x_4)\Big\}}_{G_3^*(x_3)} \Big\}\Big\}$$

$$G_1^*(x_1) = \min_{u_1}\Big\{ l(x_1,u_1) + \underbrace{\min_{u_2}\Big\{ l(x_2,u_2) + G_3^*(x_3)\Big\}}_{G_2^*(x_2)} \Big\}$$

por lo tanto,

$$G_1^*(x_1) = \min_{u_1}\Big\{ l(x_1,u_1) + G_2^*(x_2)\Big\}.$$

Entonces, para $x_1 = a$

$$\begin{aligned} G_1^*(a) &= \min_{u_1}\Big\{ l(a,u_1) + G_2^*(x_2)\Big\} \\ &= \min\Big\{ l(a,u_1) + G_2^*(a)\ ,\ l(a,u_1) + G_2^*(b)\Big\} \\ &= \min\Big\{ 2+4\ ,\ 2+6\Big\} \\ &= \min\Big\{ 6\ ,\ 8\Big\} \end{aligned}$$

por lo tanto,

$$G_1^*(a) = 6 \quad (\text{control } a \to a).$$

Para $x_1 = b$

$$\begin{aligned} G_1^*(b) &= \min_{u_1}\Big\{ l(b,u_1) + G_2^*(x_2) \Big\} \\ &= \min\Big\{ l(b,u_1) + G_2^*(c)\ ,\ l(b,u_1) + G_2^*(d) \Big\} \\ &= \min\Big\{ 1+3\ ,\ 4+\infty \Big\} \\ &= \min\Big\{ 4\ ,\ \infty \Big\} \end{aligned}$$

por lo tanto,

$$G_1^*(b) = 4 \quad (\text{control } b \to c).$$

Para $x_1 = c$

$$\begin{aligned} G_1^*(c) &= \min_{u_1}\Big\{ l(c,u_1) + G_2^*(x_2) \Big\} \\ &= \min\Big\{ l(c,u_1) + G_2^*(a)\ ,\ l(c,u_1) + G_2^*(d) \Big\} \\ &= \min\Big\{ 1+4\ ,\ 1+\infty \Big\} \\ &= \min\Big\{ 5\ ,\ \infty \Big\} \end{aligned}$$

por lo tanto,

$$G_1^*(c) = 4 \quad (\text{control } c \to a).$$

Para $x_1 = d$

$$\begin{aligned} G_1^*(d) &= \min_{u_1}\Big\{ l(d,u_1) + G_2^*(x_2) \Big\} \\ &= \min\Big\{ l(d,u_1) + G_2^*(c)\ ,\ l(d,u_1) + G_2^*(e) \Big\} \\ &= \min\Big\{ 1+3\ ,\ 1+\infty \Big\} \\ &= \min\Big\{ 4\ ,\ \infty \Big\} \end{aligned}$$

por lo tanto,

$$G_1^*(d) = 4 \quad (\text{control } d \to c).$$

Para $x_1 = e$

$$\begin{aligned} G_1^*(e) &= \min_{u_1}\Big\{ l(e,u_1) + G_2^*(x_2) \Big\} \\ &= \min\Big\{ \infty \Big\} \end{aligned}$$

por lo tanto,

$$G_1^*(e) = \infty.$$

Finalmente debemos escoger el control que nos lleve desde "a", que es el estado inicial deseado, hacia, "a", "b" o "c", y ese control es $a \to a$, por lo tanto la política para ir desde el estado "a" hasta el estado "d" está dada por: $a \to a \to b \to c \to d$.

En la Figura 3.2 se muestra todas las formas de llegar desde un estado inicial deseado hasta un estado final dado. Con rojo podemos apreciar el camino de costo mínimo para ir desde el estado inicial "a" hasta el estado final "d".

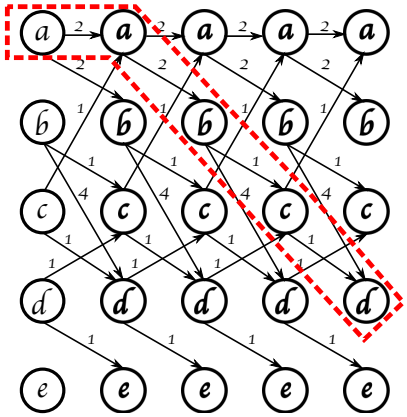


Figura 3.2: Grafo de caminos para llegar a un estado final a partir de un estado inicial.

$k=5$			$k=4$			$k=3$			$k=2$			$k=1$		
$G_5^*(x_5)$	u_5^*	x_5	$G_4^*(x_4)$	u_4^*	x_4	$G_3^*(x_3)$	u_3^*	x_3	$G_2^*(x_2)$	u_2^*	x_2	$G_1^*(x_1)$	u_1^*	x_1
∞	-	a												
∞	-	b												
∞	-	c												
			4	$d \leftarrow b$	b	6	$b \leftarrow a$	a						
						2	$c \leftarrow b$	b	4	$b \leftarrow a$	a	6	$a \leftarrow a$	a
0	-	d	1	$d \leftarrow c$	c							5	$a \leftarrow c$	c
									6	$d \leftarrow b$	b			
						2	$c \leftarrow d$	d						
									3	$d \leftarrow c$	c	4	$c \leftarrow d$	d
												4	$c \leftarrow b$	b
∞	-	e												

Tabla 3.1: Valores de costo mínimo, controles y estados

En la tabla 3.1 se pude apreciar como a partir del estado final d se puede llegar a los estados $\{a, b, c, d\}$ a partir de los controles que minimizan a la función G_k^*.

La programación dinámica supone que los estados del sistema son conocidos a través de todo el proceso de planificación. Existen métodos alternativos basados en el gradiente de la función de valor (G_k^*) llamados de lazo abierto que no requieren conocer los estados [40]. Sin embargo, utilizando estos métodos, si los controles óptimos locales no son únicos no es posible garantiza optimalidad global. Es necesario recuperar el estado por medio de una estructura combinatoria (frecuentemente un grafo) para garantizar la optimalidad global. Es por esta razón que es crucial la representación del estado por ejemplo por medio de una representación simbólica y/o combinatoria como un grafo.

3.1.2. Iteración por valor hacia adelante

Las ideas de la sub-sección anterior serán reusadas para obtener un método sistemático equivalente para este enfoque. Mientras que la iteración por valor hacia atrás puede obtener un plan óptimo para todos los estados iniciales simultaneamente, la iteración por valor hacia adelante puede ser usada para encontrar un plan óptimo desde un estado inicial dado hacia cualquier estado en X. En la iteración por valor hacia atrás, x_G debe permanecer fijo, y en el caso de iteración por valor hacia adelante, x_I, debe permanecer fijo.

En la iteración por valor hacia adelante, el rol de l_F no es importante. Sin embargo, es necesario forzar a todos los planes considerados por la iteración por valor hacia adelante a originarse desde x_I.

Denotemos con C_k^* el costo óptimo de ir del estado 0 al estado k. Para evitar planes que no comiencen en x_I, la definición de C_O^* está dada por,

$$C_0^*(x_0) = l_I(x_0) \tag{3.13}$$

en donde l_I es una función que produce $l_I(x_I) = 0$, y $l_I(x) = \infty$ para toda $x \neq x_I$.

Para un estado intermedio, $k \in \{1, \ldots, K\}$, el costo óptimo de ir de un estado a otro está dado por,

$$C_k^*(x_k) = \max_{u_0,\ldots,u_{k-1}} \left\{ l_I(x_0) + \sum_{i=0}^{k-1} l(x_i, u_i) \right\}. \tag{3.14}$$

Nótese que la suma se refiere a la secuencia de estados, $x_0, \ldots, x_{k-1}$, que es el resultado de aplicar la secuencia de acciones $(u_0, \ldots, u_{k-2})$. El último estado, x_k, no se

incluye debido a que el costo, $l(x_k, u_k)$, requiere de la aplicación de la acción u_k, la cual no se ha seleccionado.

La meta de la iteración por valor hacia adelante es llegar al estado final, F. El costo en este caso es,

$$C_K^*(x_F) = \max_{u_0,\ldots,u_K} \left\{ l_I(x_0) + \sum_{i=0}^{K} l(x_i, u_i) \right\} \tag{3.15}$$

Esta ecuación parece igual que la Ecuación 3.4, pero, en este caso l_I es usada en lugar de l_F. La iteración por valor hacia adelante encuentra planes óptimos para alcanzar cualquier estado final a partir de x_I. Este comportamiento es complementario a la iteración por valor hacia atrás. En este caso, X_G es fijo, y se encuentran planes óptimos para cualquier estado inicial. Para la iteración por valor hacia adelante, esto es al contrario.

3.2. Programación dinámica estocástica

Los problemas de optimización en donde la incertidumbre se encuentra presente poseen importantes diferencias de aquellos en donde la incertidumbre no existe. Las dos más importantes diferencias son la necesidad de tomar en cuenta el *riesgo* en la formulación del problema y la posibilidad de la recopilación de información (retroalimentación) durante el proceso de decisión.

3.2.1. Problema básico

Sea el sistema dinámico discreto en tiempo definido por la Ecuación 3.16.

$$x_{k+1} = f_k(x_k, u_k, w_k), \qquad k = 0, 1, \ldots, N-1 \tag{3.16}$$

donde $x_k \in X$ representa el estado actual del sistema, $u_k \in U(x_k)$ es el control o acción para hacer que el sistema pase de x_k a x_{k+1}, y w_k es una perturbación aleatoria, la cual depende solamente del estado x_k, y el control u_k, pero no depende de los valores de incertidumbre pasadas $w_0, w_1, \ldots, w_{k-1}$.

Al igual que en el caso determinístico se tiene una función de costo aditiva, es decir el costo que se tiene en el instante de tiempo k, denotado por $g_k(x_k, u_k, w_k)$, se acumula en el tiempo. El costo total es,

$$g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k) \tag{3.17}$$

donde g_N es el costo terminal al final del proceso. Sin embargo, debido a la presencia de w_k, el costo es en general una variable aleatoria. Por lo tanto, es necesario reformular este costo como un costo esperado,

$$\underset{\substack{w_k \\ k=0,\ldots,N-1}}{E} \left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k) \right\}. \tag{3.18}$$

Partiendo de las definiciones anteriores, dado un estado inicial x_0, el problema es encontrar controles admisible $\pi = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$, en donde μ_k mapea el estado x_k en el control $u_k = \mu_k(x_k)$ tal que $\mu_k \in U(x_k)$, de tal manera que se minimice (o maximice) el valor esperado de la función de costo,

$$J_\pi(x_0) = \underset{\substack{w_k \\ k=0,\ldots,N-1}}{E} \left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k[x_k, \mu_k(x_k), w_k] \right\}. \tag{3.19}$$

Como ya se mencionó antes, la técnica de D.P. descompone el problema en una secuencia de problemas de minimización (o maximización) más simples que se llevan a cabo en el espacio de control en lugar de en un espacio de funciones para el estado actual.

Sea $J^*(x_0)$ el valor óptimo de la función de costo denotada por la Ecuación 3.19. Entonces,

$$J^*(x_0) = J_0(x_0).$$

El valor $J_0(x_0)$ es el costo óptimo esperado para el proceso cuando el valor inicial en el tiempo 0 es x_0. La función J_0 está dada por el último paso del algoritmo de D.P., el cual procede hacia atrás en el tiempo desde el periodo $N-1$ hasta el periodo 0:

$$J_N(x_N) = g_N(x_N) \tag{3.20}$$

$$J_k(x_k) = \max_{u_k \in U(x_k)} \underset{w_k}{E}\Big\{ g_k(x_k, u_k, w_k) + J_{k+1}\big[f_k(x_k, u_k, w_k)\big] \Big\}, \quad k = 0, 1, \ldots, N-1 \tag{3.21}$$

donde el valor esperado es tomado con respecto a la distribución de probabilidad de w_k, la cual depende de x_k y u_k. Además, si $u_k^* = \mu_k^*(x_k)$ maximiza (o minimiza) el lado derecho de la Ecuación 3.21 para cada x_k y k, la política $\pi^* = \mu_0^*, \ldots, \mu_{N-1}^*$ es óptima (ver demostración en el Apéndice D).

Las Ecuaciones 3.20 y 3.21 constituyen el algoritmo básico de programación dinámica con estado de información perfecta.

3.2.2. Estado de información imperfecta

Consideremos un sistema dinámico discreto en tiempo en donde el estado del sistema es no observable para todo instante de tiempo (estado de información imperfecta). Supongamos que deseamos controlar el sistema. Sin embargo, debido a que el estado no es conocido el controlador sólo recibe alguna información acerca del estado actual del sistema. Dicha información puede ser descrita como una medida ruidosa de una función del estado del sistema. La incapacidad del controlador para observar el estado exacto del sistema puede deberse a alguna inaccesibilidad física de alguna de las variables del sistema, o a imprecisiones en los sensores, o al procedimiento utilizado para realizar las mediciones.

Desde el punto de vista analítico como computacional, los problemas con estado de información imperfecta son por lo general más complicados que los problemas en donde el estado del sistema es conocido (estado con información perfecta). Sin embargo, conceptualmente los problemas con estado de información imperfecta no son diferentes de aquellos con estado de información perfecta. De hecho, los problemas con estado de información imperfecta pueden reducirse a problemas con información perfecta como se mostrará en lo subsecuente.

Primeramente establezcamos el problema con estados de información imperfecta.

Sea el sistema dinámico discreto en tiempo definido por la Ecuación 3.16, es decir,

$$x_{k+1} = f_k(x_k, u_k, w_k), \qquad k = 0, 1, \ldots, N-1.$$

Sin embargo, el controlador no tiene conocimiento perfecto del estado, pero sí tiene acceso a la observación z_k de la forma:

$$z_0 = h_0(x_0, v_0), \quad z_k = h_k(x_k, u_{k-1}, v_k), \quad k = 1, 2, \ldots, N-1 \tag{3.22}$$

donde v_k es una perturbación aleatoria en la observación, la cual depende explícitamente del estado x_k y del control u_{k-1}.

Denotemos ahora mediante I_k la información disponible para el controlador en el tiempo k. I_k es conocido como el *vector de información*. Se define como:

$$\begin{aligned} &I_k = (z_0, z_1, \ldots, z_k, u_0, u_1, \ldots, u_{k-1}), \qquad k = 1, 2, \ldots, N-1 \\ &I_0 = z_0. \end{aligned} \tag{3.23}$$

Tomemos en cuenta ahora al conjunto de controles, el cual consiste en una secuencia finita de funciones $\pi = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$, en donde cada función μ_k mapea al vector de información I_k en el espacio de control U.

$$\mu_k(I_k) \in U \quad \forall I_k, \qquad k = 0, \ldots, N-1. \tag{3.24}$$

Tales controles se consideran admisibles. El problema es encontrar el conjunto de controles admisibles $\pi = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$ tales que maximicen (o minimicen) la función de costo descrita por la Ecuación 3.25. Esta ecuación es equivalente a la función de costo con estado de información perfecta (Ecuación 3.19), pero evidentemente para el caso de información imperfecta..

$$J_\pi = \underset{\substack{x_0, w_k, v_k \\ k=0,\ldots,N-1}}{E} \left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k[x_k, \mu_k(I_k), w_k] \right\}. \tag{3.25}$$

Podemos observar que a diferencia del caso con estado de información perfecta, en donde se busca un control u_k que aplicar para cada estado x_k, ahora buscamos un control que aplicar para cada posible vector de información I_k.

Si tomamos en cuenta que I_k contiene todas las variables conocidas que pueden servir para que el controlador pueda tomar una decisión, y considerando que el estado del sistema no es observable, entonces, resulta intuitivamente claro que ahora el sistema podría estar definido en función del vector de información.

Por definición se tiene que:

$$I_{k+1} = (I_k, z_{k+1}, u_k), \quad k = 1, 2, \ldots, N-1, \quad I_0 = z_0. \tag{3.26}$$

La Ecuación 3.26 puede ser vista como una descripción recursiva de la evolución del sistema, es decir la Ecuación 3.26 tiene la forma $x_{k+1} = f(x_k, u_k, w_k)$. El estado del

sistema es I_k, el control u_k y z_{k+1} puede ser vista como una perturbación aleatoria, la cual al igual que en el caso con estado de información perfecta, sólo depende del estado del sistema y del control, y no depende de los valores de $z_0, z_1, \ldots, z_k$.

Si consideramos que ahora la evolución del sistema se encuentra descrita por el vector de información, podemos entonces reformular la función de costo descrita por la Ecuación 3.25 en términos del vector de información. A continuación se muestra que es posible llegar al algoritmo de programación dinámica planteado para el caso con estado de información perfecta y descrito por las Ecuaciones 3.20 y 3.21.

Para mostrar esta reformulación procederemos como sigue: Partiendo de la Ecuación 3.25 y considerando el estado como I_{k+1} obtendremos el algoritmo de programación dinámica con estado de información imperfecta dado por las Ecuaciones 3.32 y 3.33, las cuales podemos constatar que son equivalentes a las presentadas en 3.20 y 3.21. Note que el punto clave es considerar a la observación como la perturbación del estado determinado por I_{k+1}.

De la Ecuación 3.25

$$J_\pi = \underset{\substack{x_0, w_k, v_k \\ k=0,\ldots,N-1}}{E} \left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k[x_k, \mu_k(I_k), w_k] \right\}$$

$$J_\pi = \underset{\substack{z_k \\ k=0,\ldots,N-1}}{E} \left\{ \sum_{k=0}^{N-1} \tilde{g}_k[I_k, \mu_k(I_k)] \right\} \tag{3.27}$$

donde la función $\tilde{g}_k$, $k = 0, 1, 2, \ldots, N-1$, está definida como:

$$\tilde{g}_{N-1}[I_{N-1}, \mu_{N-1}(I_{N-1})] = \underset{x_{N-1}, w_{N-1}}{E} \Big\{ g_N \Big[f_{N-1}[x_{N-1}, \mu_{N-1}(I_{N-1}), w_{N-1}] \Big] + g_{N-1} \Big[x_{N-1}, \mu_{N-1}(I_{N-1}), w_{N-1}) \Big] \mid I_{N-1}, \mu_{N-1}(I_{N-1}) \Big\} \tag{3.28}$$

$$\tilde{g}_k[I_k, \mu_k(I_k)] = \underset{x_k, w_k}{E} \Big\{ g_k[x_k, \mu_k(I_k), w_k] \mid I_k, \mu_k(I_k) \Big\}. \tag{3.29}$$

La probabilidad condicional $P(w_k|I_k, \mu(I_k))$, $k = 0, 1, \ldots, N-1$ está determinada por $P(w_k|x_k, \mu(I_k))$ y $P(x_k|I_k)$. La ecuación que define a $P(x_k|I_k)$ se presenta en la sub-sección 4.8.2.

El problema básico con estado de información imperfecta se ha reformulado a un problema con estado de información perfecta que implica al sistema descrito por la

Ecuación 3.26 y a la función de costo denotada por la Ecuación 3.27. A partir de esto podemos escribir el algoritmo de *Programación Dinámica* como,

$$J_{N-1}(I_{N-1}) = \max_{u_{N-1}\in U} \tilde{g}_{N-1}(I_{N-1}, u_{N-1}) \tag{3.30}$$

$$J_k(I_k) = \max_{u_k\in U}\left[\tilde{g}_k(I_k, u_k) + \underset{z_{k+1}}{E}\left\{J_{k+1}(\{I_k, z_{k+1}, u_k\}) \mid I_k, u_k\right\}\right]. \tag{3.31}$$

Si sustituimos la Ecuación 3.28 en 3.30 y la Ecuación 3.29 en 3.31 podemos escribir,

$$\begin{aligned} J_{N-1}(I_{N-1}) = \max_{u_{N-1}\in U}\Bigg[\underset{x_{N-1},w_{N-1}}{E}\Big\{&g_N\big(f_{N-1}(x_{N-1}, u_{N-1}, w_{N-1})\big) \\ &+ g_{N-1}(x_{N-1}, u_{N-1}, w_{N-1})\Big| I_{N-1}, u_{N-1}\Big\}\Bigg] \end{aligned} \tag{3.32}$$

$$J_k(I_k) = \max_{u_k\in U}\left[\underset{x_k,w_k,z_{k+1}}{E}\left\{g_k(x_k, u_k, w_k) + J_{k+1}(\{I_k, z_{k+1}, u_k\})\Big| I_k, u_k\right\}\right] \tag{3.33}$$

Las Ecuaciones 3.30 y 3.31, constituyen el algoritmo básico de programación dinámica con estado de información imperfecta. Las Ecuaciones 3.32 y 3.33 son equivalente al algoritmo con estado de información perfecta denotado por las Ecuaciones 3.20 y 3.21.

Un ejemplo concreto de la aplicación del método de programación dinámica estocástica con estado de información imperfecta se incluye en el Apéndice E. Creemos que dicho ejemplo clarifica el uso de la programación dinámica estocástica con estado de información imperfecta.

Capítulo 4

Estrategias de Movimiento para la Exploración y Construcción de Mapas bajo Incertidumbre con Múltiples Robots Heterogéneos

La exploración automática de ambientes y la construcción de mapas son problemas importantes en el campo de la robótica. Los robots autónomos deben poseer la habilidad de explorar su ambiente, construir una representación de dicho ambiente (mapas), y luego usar esas representaciones para navegar de forma efectiva.

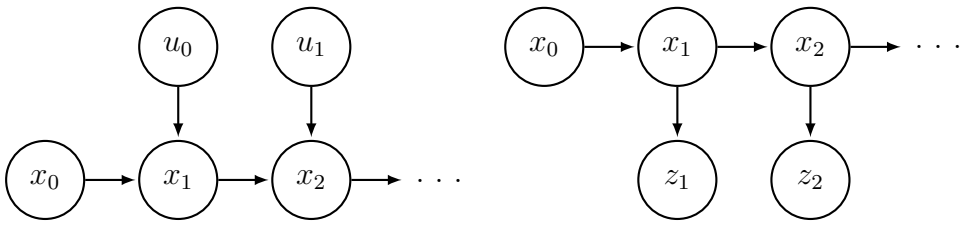
Una estrategia para explorar un entorno desconocido y construir su representación con un robot móvil puede ser realizada como sigue: (i) el robot construye un mapa local con las lecturas de los sensores, (ii) un mapa global se actualiza fusionando la información entre el mapa global actual y el nuevo mapa local, (iii) el robot se mueve a una meta intermedia, la cual se define con base en ciertas propiedades.

En las últimas dos décadas, se han propuesto varios métodos para la construcción de mapas, por ejemplo [2, 4, 23, 41], por nombrar algunos. La mayoría de las investigaciones anteriores se han centrado en el desarrollo de técnicas para extraer información relevante de los datos sin procesar, e integrar los datos recopilados en un solo modelo. Sin embargo, una estrategia de movimiento del robot para explorar el ambiente ha sido menos estudiada. En este trabajo, tratamos principalmente con este último problema. En este trabajo, se propone una estrategia de exploración. Nuestra estrategia de ex-

ploración considera las capacidades de sensado y movimiento de cada robot. Nuestro algoritmo entrega como salida las configuraciones que se visitarán y las trayectorias para alcanzarlas. Estas configuraciones están asociadas a las fronteras entre el espacio conocido y el desconocido. Nuestro método, asigna un robot del equipo a una configuración, de acuerdo con sus capacidades de sensado y movimiento, sin considerar roles predefinidos.

4.1. Principales contribuciones y originalidad

Nuestro método puede ser considerado como un proceso de decisión de Markov parcialmente observable, mejor conocido como POMDP. Esta consideración se basa en el hecho de que un POMDP modela el proceso de decisión de un agente en el cual se supone que la dinámica del sistema está determinada mediante un proceso de decisión de Markov (MDP por sus siglas en inglés), pero el agente no tiene acceso directo al estado subyacente. En su lugar, tiene que mantener una distribución de probabilidad sobre el conjunto de estados posibles con base en un conjunto de observaciones y las probabilidades de dichas observaciones, es decir, el agente razona sobre el estado del proceso de manera indirecta a través de las observaciones, tal como se hace en los modelos ocultos de Markov (HMM por sus siglas en inglés). En otras palabras, un POMDP puede ser visto como la combinación entre un Proceso de Decisión de Markov (MDP) y un Modelo Oculto de Markov (HMM). Esta combinación es más clara si observamos los modelos gráficos de un MDP y un HMM mostrados en las Fig. 4.1(a) y 4.1(b) y el modelo gráfico de un POMDP mostrado en la Fig. 4.2.



(a) MDP (b) HMM

Figura 4.1: Modelos gráficos

El equipo de robots no tiene acceso directo al estado, pero es posible obtener información de éste a través de un modelo de observación $z_k = h(x_k)$. Además, el estado siguiente del sistema ($x_{k+1} \in X$) (y consecuentemente la observación $z_{k+1} = h(x_{k+1})$) sólo dependerán del estado actual (suposición de Markov).

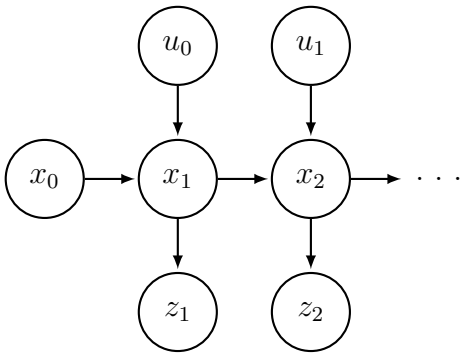


Figura 4.2: Modelos gráficos de un POMDP

Es bien sabido, que encontrar una solución óptima para los POMDPs (por ejemplo utilizando programación dinámica como en nuestro enfoque) tiene un alto costo computacional. De hecho, se ha mostrado que el problema de encontrar una solución exacta óptima es intratable [42].

Consideremos 4 robots en un ambiente $2D$ dividido en celdas con un enrejado. Suponiendo que cada robot puede alcanzar sólo a sus 8 vecinos de celda, para optimizar un paso adelante, 8^4 posibles controles deben ser evaluados. Además, mientras que el problema ya es sumamente costoso para una evaluación un paso adelante, con planes a largo plazo, una cantidad exponencial de posibilidades deben ser consideradas debido al crecimiento exponencial de posibles secuencias de observaciones [43].

Para lidiar con estos problemas y tener un enfoque tratable, se realiza lo siguiente: (1) Para el proceso de exploración, probamos los planes generados en el proceso de optimización de un paso adelante y los comparamos con planes a largo plazo. En otras palabras, para un tiempo discreto dado o paso k, comparamos el resultado de planear un orden para visitar todas las fronteras entre la parte conocida y desconocida del ambiente en ese instante de tiempo (plan a largo plazo), contra el resultado de la generación de un plan, en el que se permite a cada robot moverse una sola vez (plan a corto plazo). Comparamos los resultados de estas dos estrategias de planificación, en términos de la distancia recorrida por los robots y el número de localidades de sensado necesarios para explorar todo el ambiente. (2) Las sub-metas que los robots pueden alcanzar son generadas a través de muestreo. Algunos trabajos previos han propuesto enfoques basados en muestreo para los POMDPs [44, 45, 46] para obtener una solución aproximada en cualquier instante del plan. Sin embargo, note que en nuestro enfoque las sub-metas son generadas usando cálculos de visibilidad. Este enfoque es mejor que los métodos existentes de muestreo para los POMDPs, debido a que las muestras son generadas cerca de los bordes entre lo conocido y lo desconocido, lo que permite que

los robots tengan una buena oportunidad de sensar la parte desconocida del ambiente. (3) Nuestro modelo probabilístico de observación $p(z|x)$ es un clasificador; utilizamos la regla de Bayes para estimar esta probabilidad. En nuestro modelado, el sensado de cada robot parte probabilísticamente tanto el espacio de estados como el de observaciones (ver Fig. 4.3(a)). Una observación corresponde a una clase, y hay una relación entre un tipo de mapa local (clase) y una parte del espacio de estado, el cual, es también dividido. Note que, una observación puede corresponder a más de un estado, es decir, dos estados, x y x' pueden tener observaciones equivalentes, es decir, $h(x) = h(x')$. Por lo tanto, dos estados geométricos diferentes pueden ser indistinguibles para el sensor. Para ver esto, un ejemplo muy sencillo se muestra en la Fig. 4.3. En la figura se muestra una esquina de un mapa poligonal, en donde podemos observar tres regiones denotadas como $R1$, $R2$ y $R3$. Desde las configuraciones q_1 en la región $R1$ y q_3 en la región $R3$, el robot sensan dos conjuntos de puntos, los cuales son clasificados como una pared, mientras que en la configuración q_2 en la región $R2$, el conjunto de puntos sensado es clasificado como una esquina. La idea principal de este trabajo es transformar el estado geométrico x en una etiqueta o tipo tp, y asociar una probabilidad a cada etiqueta. De forma análoga, una observación z se obtiene al transformar las lecturas de un sensor, por ejemplo, los puntos obtenidos por un láser, en un símbolo (una clase o tipo) asociando además una probabilidad a cada clase. Esto nos permite elegir el robot más apropiado de acuerdo a sus capacidades de sensado y de movimiento para explorar esa parte del espacio. Note además que un tipo de mapa local no cambiará significativamente en un vecindario alrededor de una cierta configuración. Este tercer punto representa la principal contribución y originalidad del enfoque propuesto.

Una versión parcial de este trabajo es presentada en [47]. Las principales diferencias entre esta versión y nuestro trabajo presentado en esta tesis son: (1) En este trabajo incluimos un modelo probabilístico de movimiento. (2) Probamos planes a corto plazo y los comparamos con planes a largo plazo. (3) Se incluyen resultados en simulacion para ambientes más grandes con un mayor número de robots. (4) Se presenta una versión más clara y más detallada del enfoque propuesto.

4.2. Definición del problema y elementos del sistema

El problema consiste en explorar un ambiente desconocido e incrementalmente construir un modelo de dicho ambiente con un equipo de robots móviles, los cuales presentan diferentes capacidades tanto sensoriales como motrices. Se supone que tanto los motores

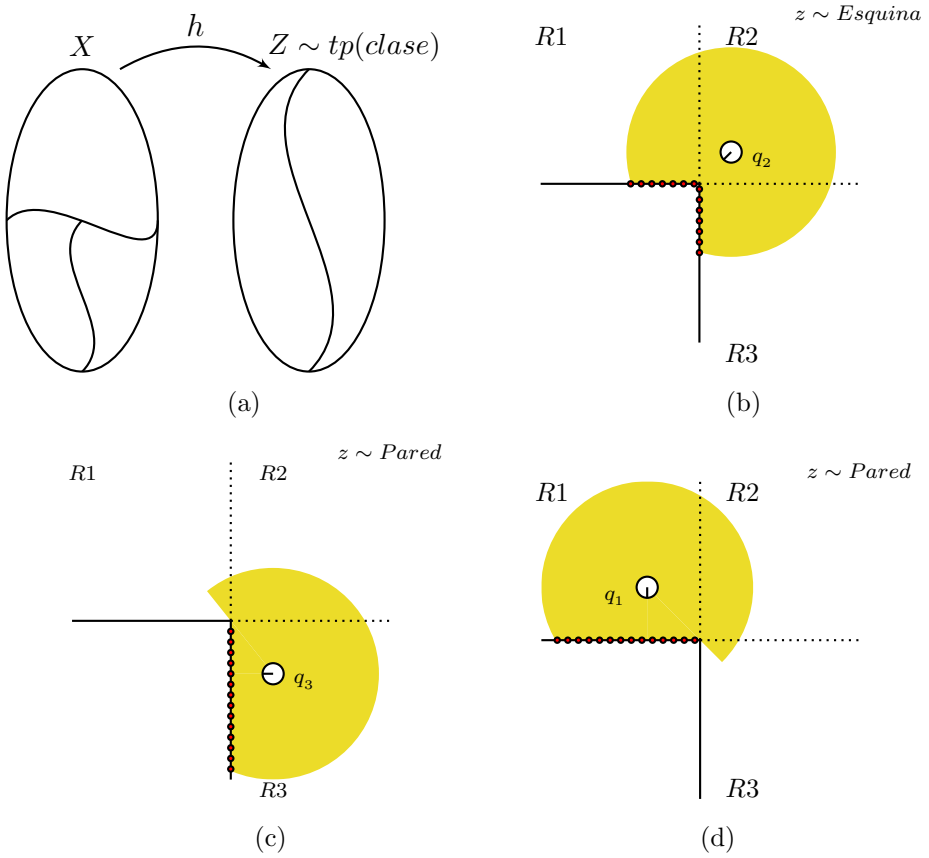


Figura 4.3: (a) Relación de equivalencia entre el espacio de observaciones y de estados, (b), (c) y (d) esquina de un ambiente poligonal. La región amarilla (gris claro) representa el área sensada, y los puntos sobre las paredes representan las lecturas del sensor láser.

como los sensores de cada robot son imprecisos. El objetivo es proponer una estrategia de movimiento, la cual genere de forma rápida y confiable la construcción del mapa. Se quiere también aprovechar al máximo las diferentes capacidades de cada robot de acuerdo con la tarea de exploración y construcción del mapa. Para explorar el ambiente lo más rápido posible, las configuraciones de sensado que proveen la máxima vista del área no explorada son preferidas.

Se supone que el ambiente es desconocido, en el sentido en que los robots no lo han sensado previamente. Sin embargo, se sabe que el entorno corresponde a un ambiente de interior estructurado. También se supone que los robots están equipados con sensores de diferente rango, resolución y precisión para determinar la distancia desde el robot hacia los obstáculos. Los robots también tienen diferentes capacidades de movimiento,

en el sentido en que algunos de ellos tienen motores más precisos comparados con los otros.

4.2.1. Elementos del sistema

Dado que suponemos ruido tanto en el sensado como en el movimiento, modelamos nuestro problema como un sistema dinámico con estado de información imperfecta; el sistema dinámico cambia en tiempo discreto. Sea $\mathcal{C}$, $\mathcal{U}$ y $\mathcal{Z}$ el espacio de configuraciones (ver Apéndice A), control y observaciones, respectivamente, los cuales corresponden a todas las posibles configuraciones del robot, controles y observaciones. El espacio geométrico de estados es $X \subset \mathcal{C} \times \mathrm{E}$, donde E es el conjunto de todos los posibles ambientes en donde pueden estar los robots. La configuración de un robot es denotada por $q_i \in \mathcal{C}$, el estado por $x_i \in X$, la observación $z_i \in \mathcal{Z}$, el control por $u_i \in \mathcal{U}$, e i denota al i-ésimo robot. Nuestro sistema dinámico se define de la siguiente manera:

Robots: Tenemos un sistema de n robots. Todos los robots son modelados geométricamente como discos del mismo radio, pero con diferentes capacidades de sensado y movimiento. Consideramos robots que pueden rotar en sitio (por ejemplo, robots de manejo diferencial).

Observaciones: Cada robot está equipado con un sensor láser, y el láser montado sobre cada robot puede ser diferente (i. e., puede tener diferente resolución, rango y precisión sobre las medidas). Los puntos pertenecientes a los obstáculos (lecturas discretas del sensor) son obtenidos del sensor láser. Cada punto es denotado por $s_j \in \mathbb{R}^2$. El conjunto de puntos sensados desde una configuración q_i es denotado por $S(q_i) = \bigcup_j s_j$. De este conjunto de puntos, una clase o tipo denotado por tp es asociada, siendo éstas las observaciones del robot.

Estados: El estado de cada robot es denotado por x_i, y está definido por $q_i \times e_i$. q_i denota la configuración del i-ésimo robot y $e_i \in E$ denota el mapa local en el cual se encuentra el robot i. Con base en el conjunto de puntos $S(q_i)$, un mapa local e_i es modelado con poli-líneas mediante una técnica de ajuste de rectas [6]. Por lo tanto, un mapa local no es más que una lista de segmentos de recta. Para un conjunto de n robots, el estado está definido como $x = q_1 \times e_1 \times \cdots \times q_i \times e_i \times \ldots q_n \times e_n$, en donde q_1, q_i y q_n denotan la configuración del robot 1, i y n respectivamente. Suponemos robots con tres grados de libertad, de tal forma que la configuración del i-ésimo robot está dada por $q_i = (p_x, p_y, \theta)^T$, p_x y p_y denotan la posición del robot en un marco de referencia global en 2D, θ denota la orientación del robot con respecto al eje de las abscisas en el marco de referencia global. Por medio de cálculos de visibilidad se obtiene el polígono de visibilidad $V(x_i)$ asociado al i-ésimo robot en el estado x_i; $V(x_i)$ es usado para definir

las fronteras entre lo conocido y lo desconocido del ambiente, y así, enviar a los robots a explorar estas partes desconocidas del ambiente.

Controles: El conjunto de todos los posibles controles es denotado por $\mathcal{U}$, y cada elemento de este conjunto $u_k \in \mathcal{U}$ es un vector que contiene los controles para cada robot. Los robots únicamente se moverán siguiendo dos primitivas de movimiento: traslaciones en línea recta y rotaciones en sitio. Este modelado nos permite tomar en cuenta errores (ruido) sobre las velocidades aplicadas a cada robot. Este modelado también nos permite simular movimientos imperfectos de los robots. Para la exploración a nivel de planificación, un control para un robot i tiene la forma $u_{(i,k)} = x_{(i,k)} \rightarrow \text{free-edge}_j$, donde $x_{(i,k)}$ representa el estado actual del robot i y free-edge_j una frontera o eje libre j entre el espacio conocido y desconocido, bajo la restricción de que dos robots no pueden visitar el mismo eje libre, pero uno o más robots pueden no moverse en determinado instante de tiempo k, por lo que el control $x_{(i,k)} \rightarrow x_{(i,k+1)} = x_{(i,k)}$ está permitido. Para más detalles ver la Sección 4.6.

Modelo de Observación: Utilizamos un modelo de observación probabilístico. En este modelo de observación, $p(z_{k+1}|u_k, x_{k+1})$ representa la probabilidad de obtener un tipo de observación z_{k+1} dado que el sistema está en x_{k+1}, habiendo aplicado un control u_k. Se supone que la observación z_{k+1} es independiente del control u_k, dado el estado x_{k+1}. Esto significa que los robots sensan el ambiente, una vez que dejan de moverse. Por lo tanto, $p(z_{k+1}|x_{k+1}, u_k) = p(z_{k+1}|x_{k+1})$. Además, se supone también que esta probabilidad es independiente para cada robot, entonces,

$$p(z_{k+1}|x_{k+1}) = p(z_{(1,k+1)}|x_{(1,k+1)})p(z_{(2,k+1)}|x_{(2,k+1)}) \dots p(z_{(n,k+1)}|x_{(n,k+1)})$$

.

En la Sección 4.3, se propone un enfoque Bayesiano para estimar $p(z_{k+1}|x_{k+1})$.

Modelo de Movimiento: El sistema cambia del estado x_k al estado x_{k+1} después de haber aplicado un control u_k con probabilidad $p(x_{k+1}|x_k, u_k)$. Debido a que tenemos un equipo de n robots, la probabilidad del estado $k+1$ dado el estado k y el control para el robot i está dada por $p(x_{(i,k+1)}|x_{(i,k)}, u_{(i,k)})$. Además, se supone que esta probabilidad es independiente para cada robot, lo que nos lleva a:

$$\begin{aligned} p(x_{k+1}|x_k, u_k) =& p(x_{(1,k+1)}|x_{(1,k)}, u_{(1,k)}) \cdots p(x_{(i,k+1)}|x_{(i,k)}, u_{(i,k)}) \\ & \cdots p(x_{(n,k+1)}|x_{(n,k)}, u_{(n,k)}). \end{aligned}$$

Estimamos estas probabilidades usando el modelo probabilístico de movimiento presentado en la Sección 4.5.

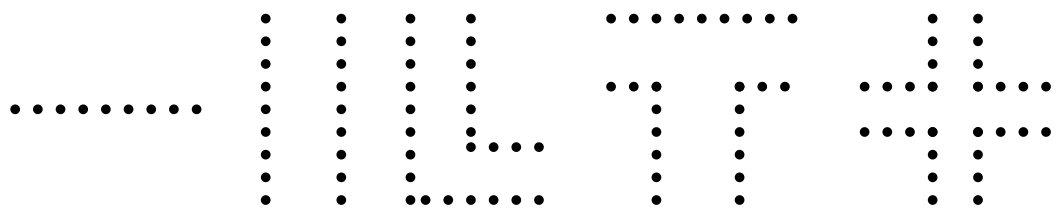
Programación Dinámica para Exploración: La programación dinámica es la herramienta utilizada para asignar robots a las siguientes configuraciones de sensado para la exploración de fronteras. Esta asignación utiliza al vector de información I_k [39]. El vector de información está definido como $I_k = (u_1, \ldots, u_{k-1}, z_1, \ldots, z_k)$; I_k es la historia de todas las observaciones hasta el tiempo k y de todos los controles que han sido aplicados al sistema hasta el tiempo $k-1$, para más información ver la Sección 4.8.

4.3. Modelo probabilístico de observación $p(z_{k+1}|x_{k+1})$

Inspirados por un enfoque Bayesiano, proponemos usar información *a priori* del ambiente, la cual puede ser incorporada a nuestros modelos con el fin de tomar mejores decisiones. Nuestro modelo de observación $z_{k+1} = h(x_{k+1})$ es un clasificador. De hecho, definimos el mapeo $z_{k+1} = h(x_{k+1})$ entre las observaciones z_{k+1} y el estado del sistema x_{k+1} como una relación de equivalencia, $z_{k+1} \sim tp$; $tp \in \mathbb{N}$. En nuestro modelo de observación, hacemos uso de las lecturas del sensor para estimar el tipo de mapa local, e, a través del método del vecino más cercano utilizando como métrica la distancia parcial de Hausdorff, esto nos permite definir el estado, el cual está dado por $x = q \times e$. Después, estimamos la clase de observación, la cual está data por $h(x) = tp$, usando la regla de Bayes. La distancia de Hausdorff, también es usada para encontrar el mejor alineamiento entre las lecturas actuales del sensor láser (resultado de nuevas percepciones) con el mapa global (resultado de la construcción incremental). Utilizamos los datos originales para alinear los mapas locales con el mapa global. Además, para facilitar el alineamiento (registro) del mapa global con la nueva información adquirida, utilizamos el desplazamiento y rotación que se le ordenó al robot como entrada al método de registro, y así encontrar la mejor alineación (rotación y traslación) que fusiona los mapas.

Queremos estimar la probabilidad de la siguiente observación z_{k+1}, dado que el robot está en x_{k+1}. Suponemos que esta probabilidad es independiente del control u_k. Hacemos uso de la regla de Bayes para estimar $p(z_{k+1}|x_{k+1})$. Los robots cuentan con un conjunto de entrenamiento de las observaciones. En este trabajo, se usan cinco tipos o clases $\Omega = \{$pared, corredor, esquina, tipo T, intersección$\}$ (ver Fig. 4.4).

En la práctica esta información *a priori* puede ser obtenida de planos de edificios o bases de datos. Cada tipo de mapa tiene asociada una probabilidad de ser sensado $p(z_{k+1})$. En este trabajo, esta probabilidad es igual para todos los tipos. La probabilidad



(a) Pared Simple (b) Corredor (c) Esquina (d) tipo T (e) intersección

Figura 4.4: Tipos de mapas locales

máxima a posteriori (MAP) es la probabilidad asignada a la observación z_{k+1} dado el estado x_{k+1}.

$$p^{\max}(z_{k+1}|x_{k+1}) = \max_{tp\in\Omega}\left\{\frac{p(x_{k+1}|z_{k+1}=tp)p(z_{k+1}=tp)}{\sum\limits_{z_{k+1}} p(x_{k+1}|z_{k+1}=tp)p(z_{k+1}=tp)}\right\} \tag{4.1}$$

$p(x_{k+1}|z_{k+1})$ es estimada utilizando el método del vecino más cercano, en donde, $p(x_{k+1}|z_{k+1}) = \frac{\zeta}{n_{tp}\phi_{tp}}$ [48]. Los parámetros ζ, n_{tp} y ϕ_{tp} se definen en la siguiente sección y se describe el método del vecino más cercano.

4.4. Método del vecino más cercano

El método del k-ésimo más próximo vecino es un método de clasificación supervisada, que sirve para estimar una función de densidad de probabilidad.

El k-ésimo más próximo vecino es un clasificador [48]. ζ es el número de elementos dentro del volumen ϕ_{tp}, n_{tp} es el número de elementos usados en la fase de entrenamiento, y ϕ_{tp} es el tamaño del volumen utilizado para asignar una clase. En general para un espacio de características de n dimensiones, y asumiendo que esas características están normalizadas, la superficie que asigna la clase es una hyperesfera. Para un espacio 2D de características, la superficie es un círculo. Para un conjunto de entrenamiento en un espacio de características con una sola característica, ϕ_{tp} está definida por $\phi_{tp} = 2d_{tp}$, en donde d_{tp} es la distancia al k-ésimo más próximo vecino (ver Apéndice F).

Dado que necesitamos medir que tan similares son dos conjuntos de puntos, utilizamos la distancia parcial de Hausdorff como métrica para d_{tp} en el método del vecino

más cercano para estimar la semejanza entre una medición del conjunto de aprendizaje de tipo tp y el conjunto hipotético $S(q_{k+1})$. De hecho, estimamos la probabilidad $p(z_{k+1}|x_{k+1})$ sin tener que llevar al sistema al estado x_{k+1}. Calculamos una región de visibilidad hipotética $V(x_{k+1})$ con base en el mapa global construido hasta el instante de tiempo k (ver Fig. 4.15). De hecho, hacemos una predicción de $S(q_{k+1})$ desde el estado x_k.

Evaluamos la distancia parcial de Hausdorff considerando una transformación tanto en traslación como en rotación. Dados dos conjuntos de puntos la distancia parcial de Hausdorff está definida como (ver [6]):

$$H(P,Q) = \max(h(P,Q), h(Q,P)) \tag{4.2}$$

donde

$$h(P,Q) = \underset{p\in P}{M} \min_{q\in Q} \|p-q\| \qquad \text{y} \qquad h(Q,P) = \underset{q\in Q}{M} \min_{p\in P} \|p-q\|. \tag{4.3}$$

En donde $\|\cdot\|$ es la norma Euclidiana para medir la distancia entre dos puntos p y q. $M_{p\in P} f(q)$ denota la media estadística de $f(q)$ sobre el conjunto P. $f(q) = \min_{q\in Q}\|p-q\|$ denota la distancia Euclidiana desde el punto p a uno de los puntos en $q \in Q$, tal que ese es el más cercano a p.

Por lo tanto, la distancia de Hausdorff es calculada entre el conjunto hipotético $S(q_{k+1})$ y el conjunto de entrenamiento, el cual ha sido generado y almacenado de acuerdo a las clases tp.

La Fig. 4.4 muestra los cinco tipos de modelos que utilizamos. Se considera que estos cinco tipos de modelos se encuentran típicamente en la estructura de un ambiente de interior. Los puntos representan los datos láser. Usamos modelos con diferentes números de puntos para simular robots equipados con sensores de diferente resolución. Además, el conjunto de datos de entrenamiento considera variantes de cada uno de estos cinco patrones mostrados en la Fig. 4.4.

Obviamente, nuestro enfoque tiene la desventaja de que otros tipos de observaciones pueden aparecer. Sin embargo, la clase sólo es usada para seleccionar al robot más apropiado de acuerdo a sus capacidades de sensado y movimiento. Al final, el mapa local integrado al mapa global corresponde directamente a las lecturas obtenidas con el sensor una vez que el sistema alcanza la configuración x_{k+1} [6].

Un ejemplo del método del vecino más cercano es presentado a continuación. Los datos de entrada a ser clasificados se muestran en la Fig. 4.5.

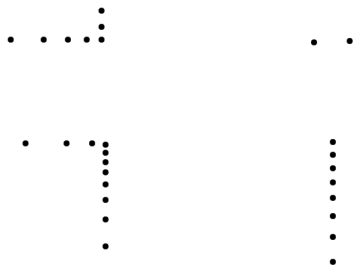


Figura 4.5: Datos de entrada.

Los elementos del conjunto de entrenamiento, con la distancia de Hausdorff más pequeña por cada clase son mostrados en la Fig. 4.6.

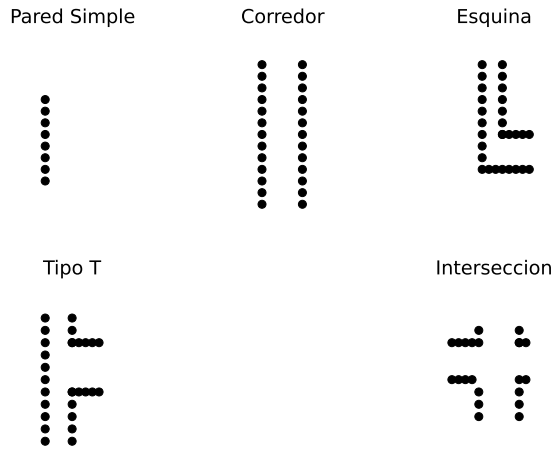


Figura 4.6: Mejor alineamiento (menor distancia de Hausdorff) por clase.

El mejor alineamiento, el cual es el que asigna la clase, entre los datos de entrada y todos los elementos del conjunto de entrenamiento se muestra en la Fig. 4.7

Los datos de entrada son mostrados en gris oscuro (rojo) en la figura y los elementos del conjunto de entrenamiento son mostrados en gris (cyan).

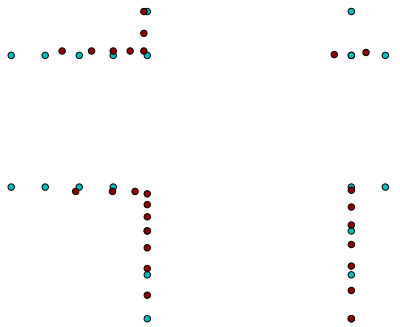


Figura 4.7: Mejor alineamiento.

La probabilidad para cada clase (siguiente observación z_{k+1}), dado que el sistema está en el estado x_{k+1} se muestra en la Tabla 4.1.

z_k	$P(z_k\|x_k)$
Pared Simple	0.0942
Corredor	0.2342
Esquina	0.1249
Tipo T	0.1373
Intersección	0.4091

Tabla 4.1: $P(z_k|x_k)$.

Utilizamos la regla de Bayes para calcular estas probabilidades.

4.5. Modelo probabilístico de movimiento $p(x_{k+1}|x_k, u_k)$

Suponemos robots que pueden rotar en sitio (por ejemplo, robots de manejo diferencial). Los robots se mueven ejecutando solamente dos primitivas de movimiento: rotación en sitio y traslación en línea recta. Los robot son controlados a través de la velocidad lineal v y la velocidad angular w. Estas velocidades no son perfectas, y existe ruido que las modifica. Para modelar la imperfección sobre las velocidades, utilizamos

una variable aleatoria con media 0 y varianza σ^2. Por lo que, las velocidad lineal y angular están dadas por las Ecuaciones 4.4 y 4.5, en donde ϵ_{σ^2} representa al error en las velocidades.

$$\hat{v} = v + \epsilon_{\sigma_v^2} \tag{4.4}$$
$$\hat{w} = w + \epsilon_{\sigma_w^2} \tag{4.5}$$

Para obtener instancias de este error, utilizamos una función denotada como sample($\cdot$). sample($\cdot$) genera una muestra aleatoria del error. Esta muestra es tomada de una distribución normal con media 0 y varianza σ^2. Por lo tanto, el movimiento imperfecto del robot dependerá de los valores de σ_v^2 y σ_w^2.

Para simular la incertidumbre sobre el movimiento de los robots, utilizamos el método de integración numérica de Euler sobre las variables de configuración de los robots (x, y, θ). Por lo tanto, tenemos:

$$\begin{aligned} x_{t+1} &= x_t + \hat{v}\cos(\theta_t)\Delta t \\ y_{t+1} &= y_t + \hat{v}\sin(\theta_t)\Delta t \\ \theta_{t+1} &= \theta_t + \hat{w}\Delta t \end{aligned}$$

Estas ecuaciones son iteradas para mover al robot cerca de la configuración final deseada.

Note que estos movimientos no son perfectos y la configuración deseada nunca es alcanzada de forma precisa. En consecuencia, la probabilidad de alcanzar el siguiente estado está dada por la Ecuación 4.6.

$$p(x_{k+1}|x_k, u_k) = p(\epsilon_{\sigma_w^2})p(\epsilon_{\sigma_v^2})p(\epsilon_{\sigma_w^2}) \tag{4.6}$$

Recordemos que un control para el robot i tiene la forma $u_{(i,k)} = x_{(i,k)} \rightarrow \text{free-edge}_j$, $x_{(i,k)}$ representa el estado actual del robot i y free-edge_j un eje libre a ser visitado. Para determinar una muestra para un eje libre, se selecciona la muestra de máxima probabilidad, la cual está dada por la Ecuación 4.6. Para calcular un plan para la exploración, utilizamos programación dinámica con estado de información imperfecta.

4.6. Generación de controles asociados a ejes libres

A nivel de planificación, un control se refiere a la asignación de un robot i a uno de los ejes libres j presentes en el instante de tiempo k, es decir, $u_{(i,k)} = x_{(i,k)} \rightarrow \text{free-edge}_j$, en donde $x_{(i,k)}$ representa el estado actual del robot. Por lo tanto, para el equipo de robots el control u_k, es un vector que contiene los controles para cada robot. En la asignación de robots a ejes libres, consideramos las siguientes restricciones. Dos robots no pueden visitar el mismo eje libre. Sin embargo, uno o más robots pueden permanecer estáticos en el instante de tiempo k, esto significa que el control $x_{(i,k)} \rightarrow x_{(i,k+1)} = x_{(i,k)}$ está permitido. Tomando en cuenta estas consideraciones el control u_k está definido como,

$$u_k = \begin{cases} x_{(i,k)} \rightarrow \text{free-edge}_j \\ x_{(i,k)} \rightarrow x_{(i,k+1)} = x_{(i,k)} \end{cases}$$

En el proceso de exploración la cantidad de controles a considerar varía en función del número de robots, y la cantidad de ejes libres presentes en el instante de tiempo k. La cantidad de controles que asocian robots a ejes libres está dada por la siguiente expresión:

Si el número de ejes libres (ne) es mayor o igual al número de robots (nr).

$$N\_ctrls = \begin{cases} P_{(ne+1,nr)} & \text{si } nr = 2 \\ P_{(ne+1,nr)} + \big(ne \cdot nr\big) & \text{si } nr = 3 \\ P_{(ne+1,nr)} + \big(ne \cdot nr\big) + \left[\sum_{(i=2)}^{(nr-2)} C_{(nr,i)} P_{(ne,i)}\right] & \text{si } nr \geq 4 \end{cases} \tag{4.7}$$

Si el número de ejes libres (ne) es menor al número de robots (nr).

$$N\_ctrls = \begin{cases} nr & \text{si } ne = 1 \\ P_{(nr+1,ne)} & \text{si } ne = 2 \\ P_{(nr+1,ne)} + \big(nr \cdot ne\big) & \text{si } ne = 3 \\ P_{(nr+1,ne)} + \big(nr \cdot ne\big) + \left[\sum_{(i=2)}^{(ne-2)} C_{(ne,i)} P_{(nr,i)}\right] & \text{si } ne \geq 4 \end{cases} \tag{4.8}$$

en donde:

nr : número de robots

ne : número de ejes libres

$$P_{(n,r)} = \frac{n!}{(n-r)!}$$

$$C_{(n,r)} = \frac{n!}{r!(n-r)!}$$

A continuación veamos un ejemplo de la asignación de robots a ejes libres. Supongamos un ambiente como el mostrado en la Fig. 4.8.

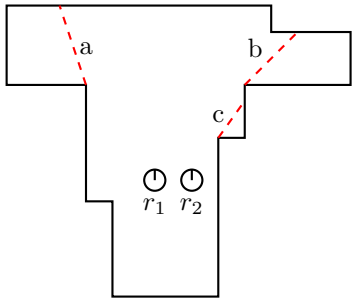


Figura 4.8: Ambiente

Para este ambiente se tiene tres ejes libres y un equipo de dos robots. Los ejes libres se denotan como a, b, c y además denotamos con x_{r_1} y x_{r_2} los estados de los robots r_1 y r_2 respectivamente. Por lo tanto, para este ejemplo $ne = 3$ y $nr = 2$, lo cual implica que para determinar la cantidad de posibles controles a nivel del planificador utilizamos la Ecuación 4.7.

$$N\_ctrls = P_{(3+1,2)}$$

$$N\_ctrls = P_{(4,2)}$$

$$N\_ctrls = \frac{4!}{(4-2)!}$$

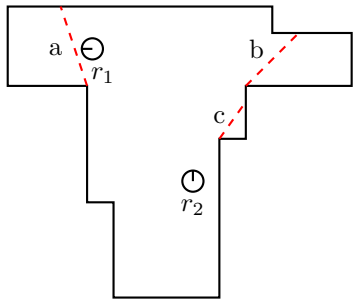
$$= 12$$

Estos doce posibles controles se muestran en la Tabla 4.9. Con guiones se especifica la ausencia de alguno de los dos robots en el eje libre. Como se puede observar, los

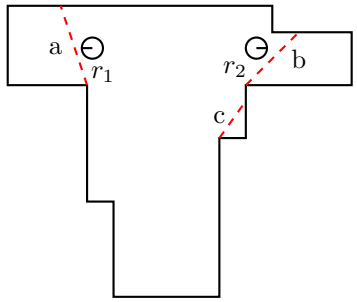
siguientes posibles estados del sistema se generan a partir de todas las permutaciones sin repetición entre los ejes libres y los robots. También podemos notar que es posible que los robots se muevan simultáneamente, tal como ya se había especificado.

Eje Libre		
a	b	c
r_1	-	-
r_2	-	-
-	r_1	-
r_2	r_1	-
-	r_2	-
r_1	r_2	-
-	-	r_1
r_2	-	r_1
-	r_2	r_1
-	-	r_2
r_1	-	r_2
-	r_1	r_2

Figura 4.9: Tabla de controles



(a) $x_{r_1} \to a$, $x_{r_2} = x_{r_2}$



(b) $x_{r_1} \to a$, $x_{r_2} \to b$

Figura 4.10: Ejemplo de controles.

En la Fig. 4.10 se muestran dos ejemplos de los doce controles posibles. En la Fig. 4.10(a) sólo el robot r_1 se mueve hacia el eje libre a, mientras que el robot r_2 permanece en su configuración inicial. En la Fig. 4.10(b) tanto el robot r_1 como el robot r_2 se mueven.

Para los casos en donde el número de ejes libres es igual a tres y cuatro podemos observar la Fig 4.11.

Cuando el número de robots es de tres, la cantidad de posibles controles está dada por,

$$\begin{aligned} N\_ctrls &= P_{(3+1,3)} + (3)(3) \\ N\_ctrls &= P_{(4,3)} + 9 \\ N\_ctrls &= \frac{4!}{(4-3)!} + 9 \\ &= 33 \end{aligned}$$

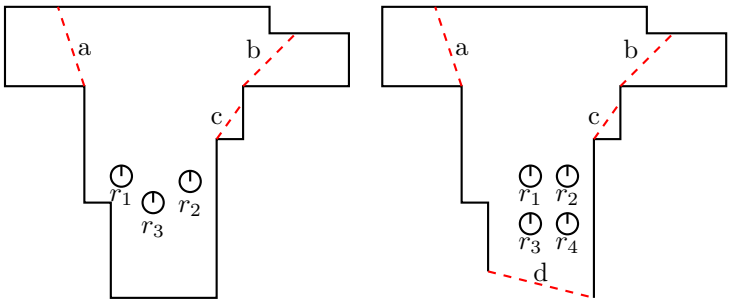


Figura 4.11: Ambientes en donde el número de robots y ejes libres es de tres y cuatro

Estos 33 posibles controles son mostrados en la Tabla 4.2.

Eje Libre			Eje Libre			Eje Libre		
a	b	c	a	b	c	a	b	c
r_1	-	-	r_2	r_3	-	-	r_1	r_2
r_2	-	-	-	-	r_1	r_3	r_1	r_2
r_3	-	-	r_2	-	r_1	-	r_3	r_2
-	r_1	-	r_3	-	r_1	r_1	r_3	r_2
r_2	r_1	-	-	r_2	r_1	-	-	r_3
r_3	r_1	-	r_3	r_2	r_1	r_1	-	r_3
-	r_2	-	-	r_3	r_1	r_2	-	r_3
r_1	r_2	-	r_2	r_3	r_1	-	r_1	r_3
r_3	r_2	-	-	-	r_2	r_2	r_1	r_3
-	r_3	-	r_1	-	r_2	-	r_2	r_3
r_1	r_3	-	r_3	-	r_2	r_1	r_2	r_3

Tabla 4.2: Tabla de controles

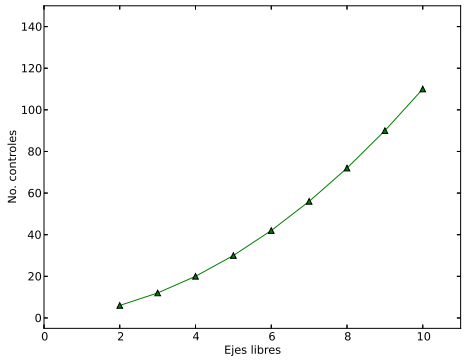
Si ahora consideramos el caso en el que se tienen cuatro robots y cuatro ejes libres, el número de posibles controles será,

$$N\_ctrls = P_{(4+1,4)} + (4)(4) + \left[\sum_{i=2}^{4-2} C_{(4,i)} \cdot P_{(4,i)}\right]$$

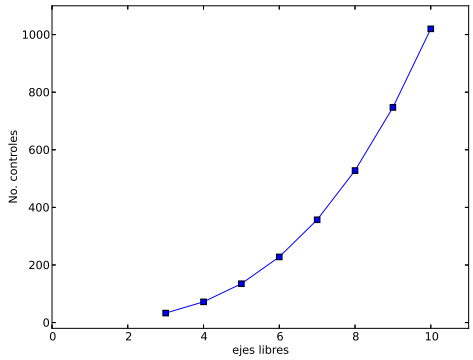
$$
\begin{aligned}
N\_ctrls &= P_{(5,4)} + 16 + \left[C_{(4,2)} \cdot P_{(4,2)}\right] \\
N\_ctrls &= \frac{5!}{(5-4)!} + 16 + \left[\frac{4!}{2!(4-2)!} \cdot \frac{4!}{(4-2)!}\right] \\
&= 208
\end{aligned}
$$

Podemos observar que la cantidad de posibles controles crece rápidamente conforme aumenta tanto el número de robots como la cantidad de ejes libres.

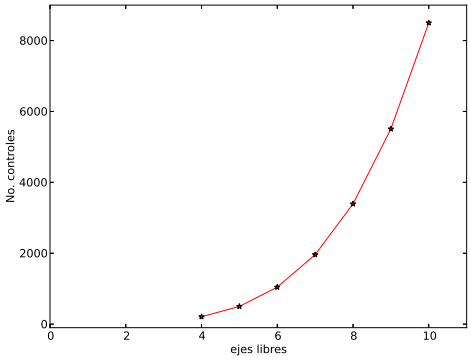
En resumen, podemos concluir que al aumentar el número de robots y el número de ejes libres, la cantidad de controles crece de forma exponencial (ver Fig 4.12), lo cual hace más tardada la generación de un plan en el proceso de exploración.



(a) Número de controles para 2 robots



(b) Número de controles para 3 robots



(c) Número de controles para 4 robots

Figura 4.12: Número de controles en función del número de robots y de ejes libres

En la Fig 4.12, el número de robots se mantiene fijo en 2, 3 y 4, mientras se varía el número de ejes libres desde un número igual al número de robots hasta 10, esto nos permite observar como es que el número de controles aumenta de forma factorial.

4.7. Visibilidad y exploración basada en fronteras

Utilizamos la exploración basada en fronteras. Para explorar el ambiente, los robots visitan los ejes libres. Es decir, explorar el espacio desconocido es equivalente a enviar un robot al borde entre la región de visibilidad $V(x_i)$ y el área no vista. Cada eje libre es visitado sólo una vez para explorar el espacio desconocido detrás de este, y además, dos robots no pueden visitar el mismo eje libre.

El robot es libre de moverse dentro del polígono de visibilidad, esto garantiza que el robot puede moverse sin chocar con los obstáculos. Denotamos con $F(x_i)$ al polígono de visibilidad reducida por el radio del robot, i.e. $F(x_i)$ es una región segura para moverse, la cual es visible desde el estado x_i. Definimos V_{tot} como la región de visibilidad del equipo de robots, y a F_{tot} como la región total en donde los robots puede mover sin chocar con obstáculos, es decir,

$$V_{tot} = \bigcup_i V(x_i) \qquad \text{y} \qquad F_{tot} = \bigcup_i F(x_i)$$

A fin de tener una o más potenciales configuraciones asignadas a un eje libre, se utiliza un muestreo aleatorio. Un segmento centrado en el punto medio del eje libre se genera usando dos parámetros: Un ángulo θ y un radio ρ, donde θ toma valores entre 0 y π con respecto a la orientación del eje libre. $\theta = 0$ corresponde a que el segmento de recta con origen en el punto medio del eje libre, sea paralelo al eje libre con la orientación del eje libre y $\theta = \pi$ corresponde a que el segmento con origen en el punto medio del eje libre tenga una orientación del eje libre más π. El radio ρ varia entre el radio del robot y el rango máximo del sensor con menor rango. En la Fig. 4.13 podemos observar la generación de una muestra (en verde) sobre un eje libre (en rojo).

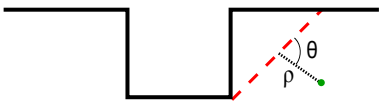


Figura 4.13: Generación de muestra

Además sólo las muestras dentro de F_{tot} son consideradas. También, las muestras generadas con ese procedimiento deben ser verificadas para determinar si la muestra es capaz de ver al eje libre. Esto es equivalente a que el polígono de visibilidad generado desde la muestra contenga totalmente o intersecte al eje libre.

La exploración comienza suponiendo que F_{tot} tiene un solo componente conectado. También suponemos que los robots siempre inician dentro de F_{tot}. Adicionalmente, para mover a los robots, combinamos el modelo probabilístico de movimiento descrito anteriormente, con una estrategia de movimiento con retroalimentación sensorial basada en la localización de las esquinas.

Hacemos que los robots se muevan sobre el grafo de visibilidad reducido (RVG), calculado sobre el espacio de configuraciones $\mathcal{C}$ y así seguir el camino más corto entre dos configuraciones. El mapa global construido hasta el tiempo k es usado para el cálculo de RVG [6] (ver Apéndice B).

En la estrategia de movimiento con retroalimentación sensorial, el robot corrige su orientación con base en la posición relativa de las esquinas con respecto a la configuración del robot. Dado que el robot está usando la ubicación de una esquina mientras se mueve, el error generado por el ruido en las velocidades no se incrementa de forma monotónica. Para más detalles ver [49].

Para ilustrar la combinación de estas dos estrategias de movimiento, podemos observar la Fig. 4.14. Mientras el robot se mueve desde su configuración inicial q_i hacia la primer esquina A, y de la esquina A a la esquina B, el robot corrige su orientación con base en la ubicación de estas esquinas, una vez que llega a la esquina B el robot comenzará a dirigirse directamente hacia la configuración q_g. Esto lo hace siguiendo el modelo probabilístico de movimiento descrito con anterioridad. Mientras el robot corrige su orientación con base en las esquinas, el movimiento se considera determinístico, dado que la retroalimentación con las esquinas ayuda a corregir el error en el movimiento. Para alcanzar la configuración que permite explorar el eje libre (último movimiento de la secuencia), se utiliza el modelo probabilístico de movimiento, dado que el robot se mueve en lazo abierto, es decir, no puede corregir su error de movimiento basándose en retroalimentación con una primitiva geométrica. En esta caso, el robot se mueve de la siguiente manera: (1) El robot rota en sitio hasta apuntar a la configuración que desea alcanzar, enseguida (2) se mueve en linea recta hasta alcanzar esta configuración y finalmente (3) el robot rota para alcanzar la orientación final deseada. Estos movimientos son estimados utilizando el modelo probabilístico de movimiento. El caso en el que el robot no se mueve entre esquinas también se presenta cuando es posible trazar una linea de vista directamente desde la configuración inicial a la configuración final sin intersectar ningún obstáculo en $\mathcal{C}$.

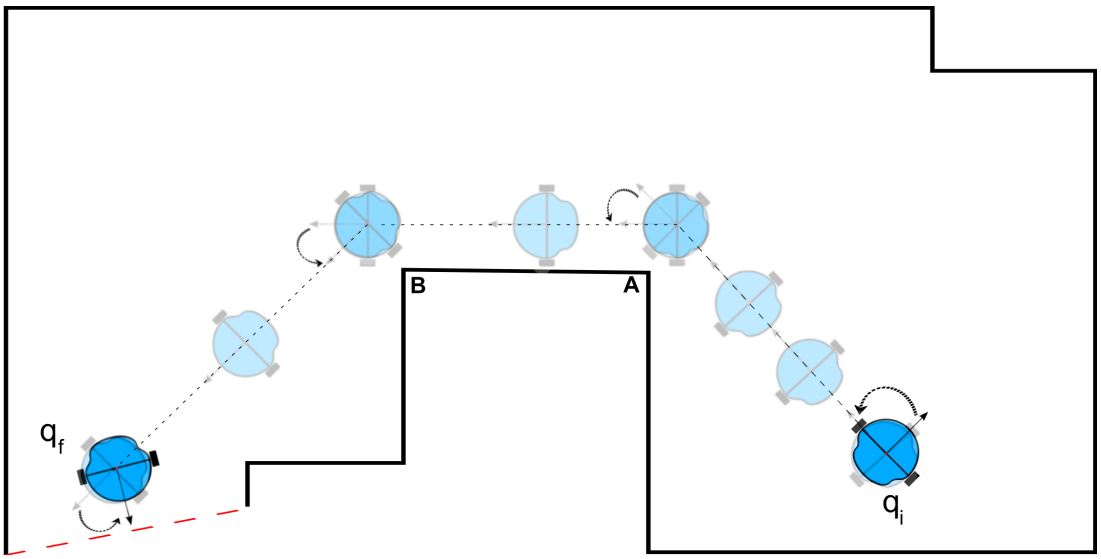


Figura 4.14: Modelo de movimiento.

Cuando un mapa global no contiene ejes libres, la exploración ha finalizado. Dado que el mapa global es actualizado constantemente, somos capaces de determinar la presencia o no de ejes libres.

4.8. Programación dinámica estocástica para determinar una acción

Seleccionamos una acción para ser ejecutada en el instante k con base en $I_k = \{u_0, u_1, \cdots, u_{k-1}, z_0, z_1, \cdots, z_k\}$. I_k es la historia de todos los controles aplicados hasta el tiempo $k-1$ y todas las observaciones tomadas hasta el tiempo k. El algorítmo de programación dinámica estocástica (SDP), está dado por la Ecuación 4.9, [39].

$$\begin{aligned} J_{N-1}(I_{N-1}) &= \max_{u_{N-1}\in U} \tilde{g}_{N-1}(I_{N-1}, u_{N-1}) \ \text{ y} \\ J_k(I_k) &= \max_{u_k \in U}\Big[\tilde{g}(I_k, u_k) + J_{k+1}(I_k, z_{k+1}, u_k) \\ &\quad \times \sum_{x_{k+1}} p^{\text{máx}}(z_{k+1}|x_{k+1}) \sum_{x_k} p(x_{k+1}|x_k, u_k)\, p(x_k|I_k)\Big] \end{aligned} \tag{4.9}$$

En la Ecuación 4.9, J_{k+1} es la función de utilidad, la cual depende de I_k, z_{k+1} y u_k. N es el horizonte de planificación. La probabilidad del estado siguiente dado el estado acutual y el control $p(x_{k+1}|x_k, u_k)$, se obtiene con la Ecuación 4.6 (ver sección 4.5). La probabilidad del estado dado el vector de información $p(x_k|I_k)$ es calculada utilizando la Ecuación 4.13 (ver subscección 4.8.2). En general, en el algoritmo de programación dinámica estocástica, se calcula el valor esperado sobre todas la posibles observaciones $E_{z_{k+1}}$, dado el vector de información I_k y el control u_k. Sin embargo, en nuestro enfoque, sólo utilizamos la observación con máxima probabilidad $p^{\max}(z_{k+1}|x_{k+1})$ dada por la Ecuación 4.1, seleccionamos sólo esta observación para evitar el crecimiento exponencial de posibilidades de secuencias de observaciones.

En la Ecuación 4.9, $\tilde{g}_k(I_k, u_k)$ representa la utilidad (ganancia sobre costo) de aplicar una acción. Es decir $\tilde{g}_k(I_k, u_k)$, indica que tan útil es aplicar un control, u_k, para un vector de información, I_k, dado; $\tilde{g}_k(I_k, u_k)$ puede ser calculado en términos de $g(x_k, u_k)$ y $p(x_k|I_k)$, de la siguiente manera [39].

$$\begin{aligned} \tilde{g}_k(I_k, u_k) &= \underset{x_k}{E}\Big\{ g(x_k, u_k) \mid I_k, u_k \Big\} \\ &= \sum_{x_k} g(x_k, u_k) p(x_k \mid I_k) \end{aligned} \tag{4.10}$$

Por lo tanto, $\tilde{g}_k(I_k, u_k)$ depende tanto de $p(x_k|I_k)$ como de $g(x_k, u_k)$ tomando en cuenta que la incertidumbre en el control y la observación son consideradas; $p(x_k|I_k)$ representa la probabilidad de estar en el estado x_k en el tiempo k, considerando los controles aplicados y las observaciones obtenidas.

En SDP, el horizonte de planificación N puede fijarse con diferentes valores. En el problema de exploración abordado en este trabajo, pueden realizarse planes a corto y largo plazo. En un plan a corto plazo $N = 1$, permitiendo que cada robot puede moverse sólo una vez. En este trabajo, a diferencia del trabajo presentado en [47], en nuestra implementación, cuando un plan es ejecutado, más de un robot puede moverse en el mismo instante de tiempo k. En contraste, en un plan a largo plazo, calculamos N de tal forma que todos los ejes libres presentes en el instante de tiempo k deben ser visitados, además, los robots son libres de moverse más de una vez.

Cuando un plan es ejecutado, con el fin de evitar que dos robots moviéndose puedan colisionar, se ejecuta el siguiente esquema de coordinación de movimiento. Se evalúa si existe intersección entre las trayectoria que siguen los robots, si no existe tal intersección entre las trayectorias, todos los robots pueden moverse de forma simultánea. En caso contrario, el robot con la trayectoria más corta se mueve primero. Considere que nuestro trabajo tiene como objetivo principal la generación de un plan, asignando el robot más

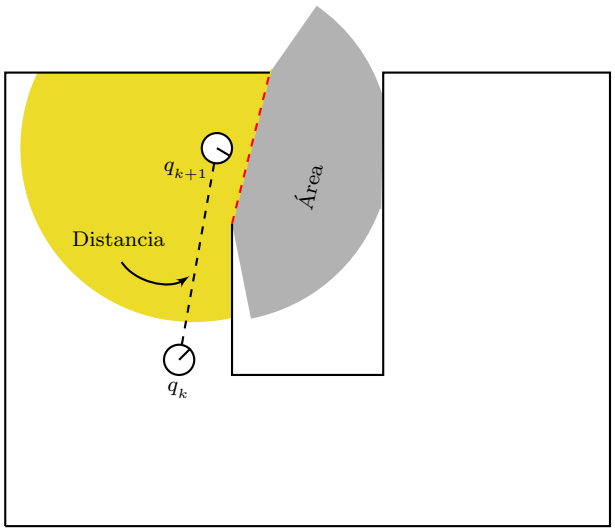


Figura 4.15: Área esperada por descubrir

apropiado para explorar una parte del ambiente, enfoques recientes (e.g. [50]) pueden usarse para coordinar la ejecución del movimiento de los robots.

4.8.1. Función objetivo $g(x_k, u_k)$

La función $g(x_k, u_k)$ indica que tan útil es aplicar un control, u_k, estando en el estado x_k. Para cada robot se define $g(x_{(i,k)}, u_{(i,k)})$ como una función que depende solamente de dos factores. Estos dos factores son: (1) El área nueva $A(q_{(i,k+1)})$, que cada robot puede percibir en la siguiente configuración. (2) La distancia que el robot debe viajar para alcanzar esa configuración $d(q_{(i,k)}, q_{(i,k+1)})$.

La Fig. 4.15 muestra al robot en una configuración cercana a la frontera entre lo conocido y lo desconocido del ambiente. La parte sombreada en gris claro (amarillo) mas la parte sombreada en gris alrededor del robot muestra el área sensada suponiendo un mapa global construido hasta ese tiempo. La linea gris oscura (roja) punteada representa un eje libre. La parte sombreada en gris que se encuentra a la derecha del eje libre y que está etiquetada como *Área* ($A(q_{(i,k+1)})$) representa el área nueva que el robot puede descubrir al estar en esa configuración, suponiendo el mapa global hasta ese instante de tiempo.

La función objetivo $g(x_k, u_k)$ quede definida para el caso de un robot como,

$$g(x_{(i,k)}, u_{(i,k)}) = \frac{A(q_{(i,k+1)})}{d(q_{(i,k)}, q_{(i,k+1)}) + 1} \tag{4.11}$$

La Ecuación 4.12 define la distancia entre la configuración $q_{(i,k)}$ y la configuración $q_{(i,k+1)}$, en donde $\alpha = \min\Big(|\theta_{(i,k)} - \theta_{(i,k+1)}|\ ,\ 2\pi - |\theta_{(i,k)} - \theta_{(i,k+1)}|\Big)$

$$d(q_{(i,k)}, q_{(i,k+1)}) = \sqrt{(p_{x_{(i,k)}} - p_{x_{(i,k+1)}})^2 + (p_{y_{(i,k)}} - p_{y_{(i,k+1)}})^2 + \alpha^2} \tag{4.12}$$

Para n robots se tiene,

$$g(x_k, u_k) = \sum_i \frac{A(q_{(i,k+1)})}{d(q_{(i,k)}, q_{(i,k+1)}) + 1}$$

La razón por la cual se suman distancias con ángulos, es que queremos que los robots no sólo se trasladen pequeñas distancias, sino que también roten ángulos pequeños. Esto es debido a que un error en la rotación del robot induce un error importante en la configuración final del robot. Reducir la rotación de los robots tiene como consecuencia que el tiempo para explorar el ambiente también sea reducido. De hecho la Ecuación 4.12 define una métrica comúnmente usada en planificación de movimientos (para más detalles ver [51]).

4.8.2. Cálculo de $p(x_k|I_k)$

$p(x_k|I_k)$ está dada por la Ecuación 4.13

$$p(x_k|I_k) = \frac{\sum\limits_{x_{k-1}} p(x_{k-1}|I_{k-1})p(x_k|x_{k-1}, u_{k-1})p(z_k|x_k)}{\sum\limits_{x_k}\sum\limits_{x_{k-1}} p(x_{k-1}|I_{k-1})p(x_k|x_{k-1}, u_{k-1})p(z_k|x_k)} \tag{4.13}$$

La Ecuación 4.13 es usada en [52] para calcular el *belief* en un filtro de Bayes y en [51] para estimar la probabilidad de un estado dado el vector de información I. La Ecuación 4.13 puede ser derivada utilizando la regla de Bayes, y utilizando marginalizaciones sobre probabilidades conjuntas (ver Apéndice C); $p(x_k|I_k)$ es expresada en términos de $p(z_k|x_k)$, $p(x_k|x_{k-1}, u_{k-1})$ y $p(x_{k-1}|I_{k-1})$. Para el cálculo de $p(x_k|I_k)$, se necesita en un principio de $p(x_0|I_0)$, en donde $I_0 = z_0$, entonces tenemos que $p(x_0|I_0) = p(x_0|z_0)$. Esta probabilidad la podemos calcular con: $p(x_0|z_0) = \frac{p(z_0|x_0)p(x_0)}{\sum_{x_0} p(z_0|x_0)p(x_0)}$.

4.9. Asignación de robots a estados asociados a un eje libre

El Algoritmo 1 representa el método general para asignar robots a estados en el proceso de exploración. En la línea 7 de este algoritmo, se generan muestras de configuraciones candidatas cercanas a un eje libre. Enseguida, se conserva sólo una de estas muestras por eje libre. Cada muestra que se conserva es la que presenta la probabilidad $p(x_{k+1}|x_k, u_k)$ más alta. Recordemos que la probabilidad $p(x_{k+1}|x_k, u_k)$ está dada por la Ecuación 4.6. En la línea 8 y 9 se utiliza programación dinámica para encontrar la estrategia o política de exploración $\Pi_k(I_k)$, la cual provee el control que al ser aplicado mueve a los robots a las localidades seleccionadas. Por lo tanto, $\Pi_k(I_k)$ también asigna robots a estados de acuerdo a sus capacidades de movimiento y sensedo.

Algorithm 1: EXPLORATION($\mathcal{Q}_k$,$\mathcal{W}$)

Input : $\mathcal{Q}_k$: Robots Configurations at time k
$\mathcal{W}$: Environment
Output: $\mathcal{M}$: Global Map

1 **begin**
2 **repeat**
3 $S \longleftarrow$ SENSORS_READING($\mathcal{Q}_k$, $\mathcal{W}$);
4 $\mathcal{M}$.add(S);
5 $\mathcal{F} \longleftarrow$ GET_FREE_EDGES($\mathcal{M}$);
6 **if** $\mathcal{F} \neq \emptyset$ **then**
7 $\mathcal{Q}_{k+1}$= SAMPLES($\mathcal{F}$, $\mathcal{W}$);
8 $$\Pi_k(I_k) = \arg\max_{u_k \in U_k} \Big[\tilde{g}(I_k, u_k) + J_{k+1}(I_k, z_{k+1}, u_k) \sum_{x_{k+1}} p^{\max}(z_{k+1}|x_{k+1}) \sum_{x_k} p(x_{k+1}|x_k, u_k)\, p(x_k|I_k) \Big];$$
9 $X_{k+1} \longleftarrow$ MOVE_NEW_STATES($\mathcal{F}$, $\mathcal{Q}_k$, $\mathcal{Q}_{k+1}$, $\mathcal{M}$, $\Pi(k, I_k)$);
10 **end**
11 **until** $\mathcal{F} = \emptyset$;
12 Return $\mathcal{M}$;
13 **end**

4.10. Resultados en simulación

En esta sección se presentan los resultados en simulación. Todas nuestras simulaciones fueron ejecutadas en una PC equipada con un procesador de cuatro núcleos, 3 GB de memoria RAM y un sistema operativo GNU/Linux. Nuestro software fue implementado usando C++. Con el fin de distinguir a los robots, en todas las figuras presentadas en esta sección, los robots con mejores capacidades de sensado son representados con un cuadrado, mientras que los robots con mejores capacidades de movimiento son representados mediante un circulo. Sin embargo, para encontrar un camino libre de colisiones, ambos tipos de robots son modelados como un disco del mismo radio. La región de visibilidad de los robots con mejores capacidades de sensado se muestra en gris (verde) y la región de visibilidad de los robots con mejores capacidades de movimiento se muestra en gris claro (amarillo).

Se llevaron a cabo siete experimentos de simulación diferentes en cuatro ambientes diferentes. Las capacidades de movimiento se definen mediante dos parámetros σ_v^2 y σ_w^2 los cuales corresponden, a la varianza del error sobre las velocidades de traslación y rotación respectivamente. Valores grandes de estos parámetros corresponden a movimientos más imprecisos. Los parámetros que definen las capacidades de movimiento de los robots son mostrados por experimento en la Tabla 4.3. Las capacidades de sensado son definidas con tres parámetros: (1) rango de sensado, (2) resolución del sensor y (3) error máximo en la distancia sensada hacia un obstáculo; esto parámetros se muestran por experimento en la Tabla 4.4. Se realizaron 20 ejecuciones con las mismas configuraciones iniciales para cada uno de los experimentos.

(BM) σ_v^2	(BM) σ_w^2	(WM) σ_v^2	(WM) σ_w^2
Experimentos 1, 2, 3, 4, 5 y 6			
0.06	0.04	0.7	0.5
Experimento 7			
0.006	0.004	0.7	0.5

Tabla 4.3: Capacidades de movimiento.

En todos los experimentos, reportamos la media y desviación estándar de las siguientes métricas de desempeño: (i) numero total de localidades de sensado que necesitó un equipo de robots para explorar el ambiente, (ii) ángulo total de rotación de todos los robots, (iii) distancia total viajada por los robots, (iv) porcentaje de área percibida. La

	(BS)			(WS)	
rango	error máximo en la distancia sensada	resolución del sensor	rango	error máximo en la distancia sensada	resolución del sensor
Experimento 1 y 2, Fig. 4.16 1200 unidades	2 unidades	2 grados	800 unidades	4 unidades	4 grados
Experimento 3 y 4, Fig. 4.17 1250 unidades	2 unidades	2 grados	800 unidades	2 unidades	3 grados
Experimento 5 y 6, Fig. 4.18 1300 unidades	2 unidades	2 grados	900 unidades	5 unidades	3 grados
Experimento 7, Fig. 4.19 1200 unidades	2 unidades	2 grados	700 unidades	4 unidades	4 grados

Tabla 4.4: Capacidades de sensado.

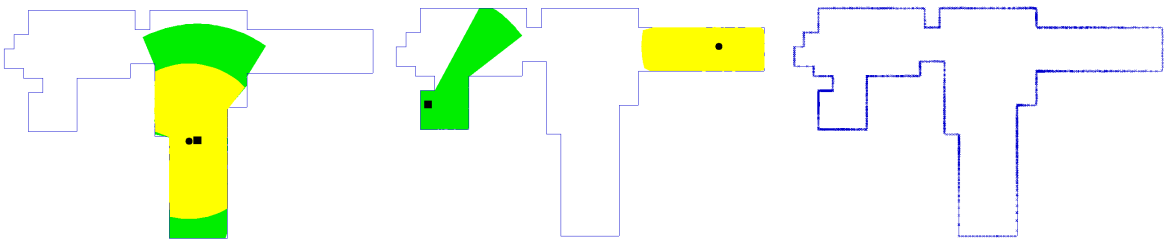
media y desviación estandar de las métricas anteriores de cada experimento han sido calculadas sobre el número total de simulaciones. También presentamos el tiempo total de planificación necesario para generar un plan para explorar todo el ambiente.

En los experimentos 1, 2, 3 y 4, solamente se utilizaron dos robots, cada uno de ellos con diferentes capacidades de sensado y de movimiento. En los experimentos 5, 6 y 7 , los robots son agrupados en equipos. Cada equipo está compuesto por elementos que tiene mejores o peores capacidades de sensado o movimiento con respecto del otro equipo. (BS) y (WS) denota mejores capacidades o peores capacidades de sensado respectivamente. De forma análoga (BM) y (WM) mejores capacidades de movimiento o peores capacidades de movimiento.

4.10.1. Experimentos 1 y 2, planes a corto plazo vs. planes a largo plazo

Experimentos 1 y 2 se realizaron en el mismo ambiente con las mismas configuraciones iniciales de los robots.

El ambiente en el que se realizaron los experimentos 1 y 2 es mostrado en la Fig. 4.16, este ambiente es de 3000 unidades de largo, por 2300 unidades de ancho. Los robots utilizados son del mismo radio el cual es de 50 unidades. En el Experimento 1, un plan



(a) Configuración Inicial de los robots (b) Instante de exploración (c) Puntos sensados

Figura 4.16: Experimento 1.

a corto plazo es generado, mientras que en el Experimento 2 se generan planes a largo plazo.

En la Fig. 4.16(a) se muestran las configuraciones iniciales de los robots, en la Fig. 4.16(b) se muestra un instante de tiempo de la exploración del ambiente. En 15 de las 20 simulaciones realizadas sucedió lo siguiente: el robot con mejores capacidades de sensado fue seleccionado para explorar la parte visualmente más rica del ambiente (parte izquierda del ambiente) y el robot de mejores capacidades de movimiento fue seleccionado para explorar la parte del ambiente compuesta solamente por líneas paralelas, el corredor en la parte derecha del ambiente (ver Fig 4.16(b)), las cuales son más difíciles de emparejar con el mapa global. En la Fig. 4.16(c) se muestran los puntos sensados por los dos robots.

Comparando las métricas de desempeño definidas con anterioridad, de un plan a corto plazo (Experimento 1) vs. un plan a largo plazo (Experimento 2), se puede observar que cuando planes a largo plazo son generados, los robots se trasladan y rotan más. En planes a largo plazo, los robots rotan cerca de tres veces más que en cuando se generan planes a corto plazo, además, en los planes a largo plazo, los robots se trasladan alrededor de un 60 % más comparado con los planes a corto plazo, mientras que las otras métricas de desempeño se mantienen muy similares. Además, el tiempo necesario para calcular un plan a largo plazo es alrededor de tres veces más grande con respecto a la generación de planes a corto plazo (Tabla 4.5 y 4.6).

Métricas de desempeño	Robot 1 (BS) y (WM)	Robot 2 (WS) y (BM)
Media: número de localidades de sensado	4.55	4.8
Desv. std.: número de localidades de sensado	0.73	0.67
Media: ángulo total (radianes)	6.46	7.48
Desv. std.: ángulo total (radianes)	2.10	1.94
Media: longitud del trayecto	3019.41	5031.54
Desv. std.: longitud del trayecto	528.81	907.63
Media: área percibida	86.96 %	74.23 %
Desv. std.: área percibida	2.28 %	5.36 %
Tiempo de Planificación		
Media 12.95 s.	Desv. std. 1.71 s.	

Tabla 4.5: Estadísticas del Experimento 1 mostrado en la Fig. 4.16, planes a corto plazo.

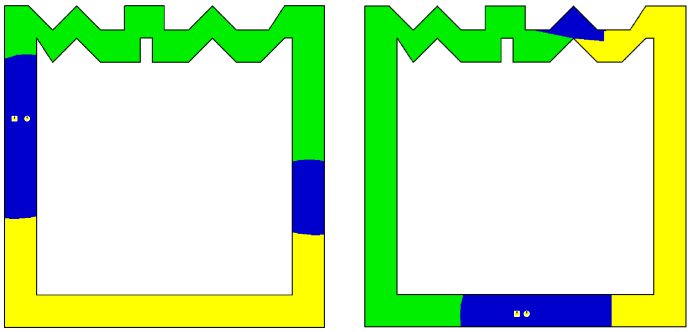
Métricas de desempeño	Robot 1 (BS) y (WM)	Robot 2 (WS) y (BM)
Media: número de localidades de sensado	4.5	4.9
Desv. std.: número de localidades de sensado	0.59	0.53
Media: ángulo total (radianes)	16.16	17.44
Desv. std.: ángulo total (radianes)	6.46	5.43
Media: longitud del trayecto	5282.71	9064.66
Desv. std.: longitud del trayecto	2202.32	2894.15
Media: área percibida	87.2825 %	75.3732 %
Desv. std.: área percibida	3.06 %	6.40 %
Tiempo de Planificación		
Media 38.05 s.	Desv. std. 6.87 s.	

Tabla 4.6: Estadísticas del Experimento 2, planes a largo plazo.

4.10.2. Experimentos 3 y 4, configuraciones iniciales diferentes

En los experimentos 3 y 4, el ambiente es de 4000 unidades de largo y 4000 unidades de ancho, para estos experimentos los robots tiene un radio de 30 unidades. En ambos experimentos se generaron planes a corto plazo.

En estos dos experimentos, las configuraciones iniciales de los robots son modificadas. En la Fig 4.17(a) se muestra el ambiente en donde el Experimento 3 es realizado, así como las configuraciones iniciales de los robots. En la misma figura, el área percibida por el robot con mejores capacidades de sensado pero malas capacidades de movimiento es mostrada en gris (verde), en gris claro (amarillo) se muestra el área percibida por el otro robot, y con gris oscuro (azul oscuro) el área explorada por los dos robots.



(a) Primera configuración inicial (b) Otra configuración inicial

Figura 4.17: Experiments 3 and 4.

La Fig 4.17(b) muestra el Experimento 4 con otras configuraciones iniciales diferentes al experimento anterior. Nuevamente, el área percibida por el robot con mejores capacidades de sensado pero malas capacidades de movimiento es mostrada en gris (verde), el área percibida por el otro robot se muestra en gris claro (amarillo), y el área explorada por los dos robots se muestra en gris oscuro (azul oscuro).

Para este ambiente, es posible observar que independientemente de las configuraciones iniciales de los robots, el robot con buenas capacidades de sensado tiene la tendencia de explorar partes del ambiente con esquinas, y el robot con buenas capacidades de movimiento explora partes del ambiente compuestas por corredores. En la Fig 4.17(a), este comportamiento es muy claro. Dado que ambos robots comienzan percibiendo al mismo tiempo partes del mapa con esquinas y con corredores compuestos por segmentos linea recta. Los robots con movimientos más precisos exploran el corredor, y el otro robot explora las partes del mapa con esquinas. En la Fig. 4.17(b) ambos robots comienzan percibiendo solamente corredores, y un robot se mueve hacia la izquierda mientras que el otro se mueve hacia la derecha. Una vez que estos llegan a la parte del mapa que contiene más esquinas, el robot con buenas capacidades de sensado explora una porción más grande de esta parte del ambiente. En las Tablas 4.7 y 4.8, se presentan las estadísticas de los Experimentos 3 y 4. En el Experimento 3 y 4, el robot con mejores capacidades de sensado explora un poco más de área del ambiente, pero ambos robots viajan básicamente la misma distancia.

Métricas de desempeño	Robot 1 (BS) y (WM)	Robot 2 (WS) y (BM)
Media: número de localidades de sensado	9.1	8.9
Desv. std.: número de localidades de sensado	0.7	0.83066
Media: ángulo total (radianes)	10.2328	6.95555
Desv. std.: ángulo total (radianes)	1.6258	1.78402
Media: longitud del trayecto	7082.54	7193.68
Desv. std.: longitud del trayecto	2607.13	1409.26
Media: área percibida	61.9506 %	59.5007 %
Desv. std.: área percibida	7.59017 %	4.02811 %
Tiempo de Planificación		
Media 15.82 s.	Desv. std. 1.787 s.	

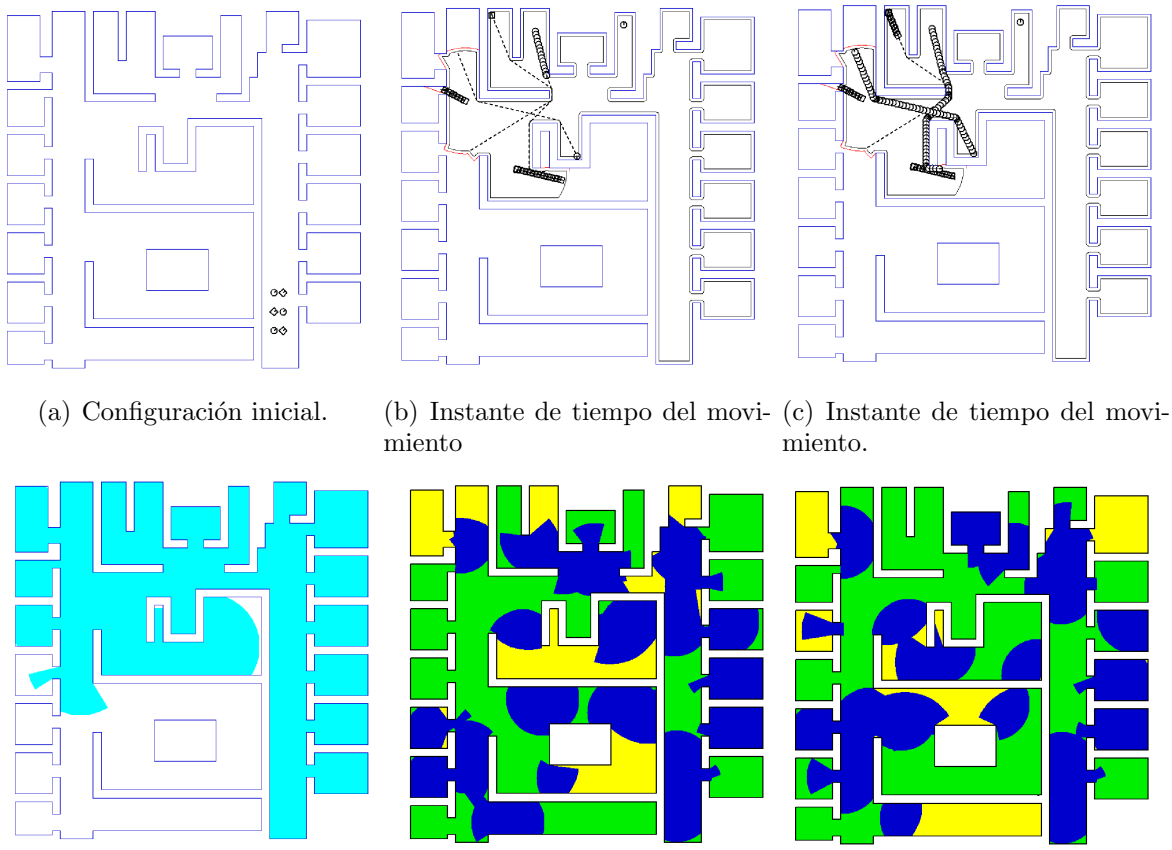
Tabla 4.7: Estadísticas del Experimento 3 mostrado en la Fig. 4.17(a).

Métricas de desempeño	Robot 1 (BS) y (WM)	Robot 2 (WS) y (BM)
Media: número de localidades de sensado	7.8	9.05
Desv. std.: número de localidades de sensado	1.36382	1.24399
Media: ángulo total (radianes)	8.55254	7.04063
Desv. std.: ángulo total (radianes)	2.49364	1.51604
Media: longitud del trayecto	9317.03	6673.82
Desv. std.: longitud del trayecto	4206.27	676.41
Media: área percibida	62.0572 %	56.4073 %
Desv. std.: área percibida	5.38649 %	5.67474 %
Tiempo de Planificación		
Media 14.84 s.	Desv. std. 1.2571 s.	

Tabla 4.8: Estadísticas del Experimento 4 mostrado en la Fig. 4.17(b).

4.10.3. Experimentos 5 y 6, planes a corto plazo vs. planes a largo plazo, con un equipo de robots

Los experimentos 5 y 6 son realizados en el mismo ambiente mostrado en la Fig 4.18. El ambiente es de 8500 unidades de largo por 8700 de ancho, y todos los robots tienen un radio de 75 unidades. Seis robots son utilizados en cada experimentos. En el Experimento 5, los robots realizan planes a corto plazo, mientras que en el Experimento 6 los robots realizan planes a largo plazo. En un plan a corto plazo, más de un robot puede moverse en el mismo instante de tiempo k, pero cada robot se mueve solo una vez. En contraste, en los planes a largo plazo, los robots son libres de moverse mas de una vez, de tal forma que, todos los ejes libres presentes en tiempo k deban ser visitados.



(a) Configuración inicial. (b) Instante de tiempo del movimiento (c) Instante de tiempo del movimiento.

(d) Exploración parcial. (e) Exploración plan a corto plazo. (f) Exploración plan a largo plazo.

Figura 4.18: Experimento 5 y 6.

En la Fig. 4.18 se muestran los resultados de los Experimentos 5 y 6. Los resultados del Experimento 5 se presentan en la Fig. 4.18(a)-4.18(e). En la Fig. 4.18(a) se observan las configuraciones iniciales de los 6 robots, en la Fig. 4.18(b) y 4.18(c) se muestran instantes de tiempo en donde se realiza el movimiento de los robots, 5 de 6 robots se mueven simultáneamente, mientras que un robot no es seleccionado para explorar un eje libre. En la Fig. 4.18(d) se presenta en gris claro (cyan) el área percibida por todos los robots hasta un cierto instante de tiempo antes de finalizar con la exploración. La Fig. 4.18(e) muestra el resultado final de la exploración, el área percibida por el equipo de robots con buenas capacidades de sensado pero malas capacidades de movimiento se muestra en gris (verde). En gris claro (amarillo) es mostrada el área percibida por el otro equipo, y en gris oscuro (azul oscuro) se muestra el área explorada por ambos

equipos. Las métricas de desempeño relacionadas a la tarea de exploración se muestran en la Tabla 4.9. Las métricas de desempeño del Experimento 6 son presentadas en la Tabla 4.10. La exploración final es mostrada en la Fig. 4.18(f). Comparando las métricas de desempeño resultantes de los Experimentos 5 y 6, podemos observar que cuando se generan planes a largo término, los robots se desplazan y rotan más (para este experimento alrededor de un 25 %), mientras que las otras métricas se mantienen muy similares. Además, el tiempo necesario para generar planes a largo plazo es alrededor de 5 veces con respecto a la generación de planes a corto plazo.

Métricas de desempeño	Robot 1 (BS) y (WM)	Robot 2 (WS) y (BM)
Media: número de localidades de sensado	43	33.85
Desv. std.: número de localidades de sensado	3.57	5.17
Media: ángulo total (radianes)	131.76	100.88
Desv. std.: ángulo total (radianes)	15.88	22.40
Media: longitud del trayecto	110257	95255.4
Desv. std.: longitud del trayecto	11187.8	15276.5
Media: área percibida	87.73 %	54.34 %
Desv. std.: área percibida	3.57 %	5.46 %
Tiempo de Planificación		
Media 7 min. 3 s.	Desv. std. 41.4 s.	

Tabla 4.9: Estadísticas del Experimento 5 mostrado en la Fig. 4.18 (a), (b), (c), (d) y (e).

Métricas de desempeño	Robot 1 (BS) y (WM)	Robot 2 (WS) y (BM)
Media: número de localidades de sensado	42.35	37.2
Desv. std.: número de localidades de sensado	2.35	4.78
Media: ángulo total (radianes)	166.96	134.12
Desv. std.: ángulo total (radianes)	33.97	30.86
Media: longitud del trayecto	134656	119774
Desv. std.: longitud del trayecto	26666.4	27262.6
Media: área percibida	86.53 %	59.11 %
Desv. std.: área percibida	3.65 %	7.59 %
Tiempo de Planificación		
Media 38 min 41 s.	Desv. std. 21 min 31 s.	

Tabla 4.10: Estadísticas del Experimento 6, planes a largo plazo, mostrado en la Fig. 4.18 (a) y (f).

4.10.4. Experimento 7, agrupando los mejores robots en el mismo equipo

Para el Experimento 7, el ambiente utilizado es de 4000 unidades de largo por 3500 unidades de ancho. Todos los robots tienen un radio de 50 unidades. En este experimento, existen dos equipos, cada equipo consiste de dos robots, y ambos equipos realizan planes a corto plazo. En este experimento los robots con buenas capacidades de sensado (BS) y buenas capacidades de movimiento (BM) (con respecto al otro equipo) están agrupados en el mismo equipo. El objetivo es evaluar el desempeño de cada equipo, cuando un equipo se compone de los mejores robots, mientras que el otro se compone por los robots con peores capacidades. En la Fig. 4.19, la configuración inicial de los robots y el área total descubierta por cada equipo son mostradas. El área descubierta por el equipo con mejores robots es mostrada en gris (verde), el área sensada por los robots con peores capacidades es mostrada en gris claro (amarillo), y el área percibida por los dos equipos es mostrada en gris oscuro (azul oscuro).

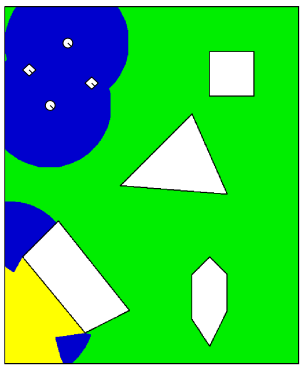


Figura 4.19: Experiment 7.

La Tabla 4.11 presenta las métricas de desempeño del Experimento 7. Los robots con mejores capacidades de sensado y movimiento exploran más del doble de área explorada por el otro equipo. El número de localidades de sensado es igualmente el doble para el equipo con los mejores robots. Sin embargo, la distancia total recorrida por ambos equipos es casi la misma. Como una interpretación de los resultados de este experimento, concluimos que aún cuando ambos equipos se mueven prácticamente lo mismo, el equipo compuesto por los mejores robots es capaz de descubrir una porción significativamente grande del entorno.

Con base en los resultados de todos los experimentos, podemos concluir que hacer planes a largo plazo con información parcial y dinámica a menudo producirá trayec-

Métricas de desempeño	Robot 1 (BS) y (WM)	Robot 2 (WS) y (BM)
Media: número de localidades de sensado	8.15	4.85
Desv. std.: número de localidades de sensado	0.57	1.27
Media: ángulo total (radianes)	8.42	8.24
Desv. std.: ángulo total (radianes)	3.20	4.936
Media: longitud del trayecto	10141.8	8763.92
Desv. std.: longitud del trayecto	2586.42	3752.88
Media: área percibida	95.6372 %	42.3139 %
Desv. std.: área percibida	2.58 %	6.28 %
Tiempo de Planificación		
Media 38 min 41 s.	Desv. std. 1.52 s.	

Tabla 4.11: Estadísticas del Experimento 7 mostrado en la Fig. 4.19.

torias innecesariamente largas. Note que tan pronto como nuevas fronteras (eje libres) aparecen (las cuales no han sido consideradas en el tiempo k), el robot podría necesitar regresar a configuraciones cercanas a otras configuraciones previamente visitadas, y en consecuencia trasladarse más de lo requerido. También podemos concluir que para algunos ambientes, nuestro enfoque genera comportamientos en los robots, esto es, robots con buenas capacidades de sensado son seleccionados para explorar porciones el ambiente visualmente ricas, y robots con buenas capacidades de movimiento, son usados para explorar porciones del ambiente que son difíciles de emparejar con el mapa global. Por lo tanto, al igual que en [26] podemos concluir que tomar decisiones locales con mejor información resulta en un mejor desempeño global.

Capítulo 5

Implementación en Robots Reales

Actualmente nos encontramos desarrollando, e integrando la arquitectura necesaria para llevar a cabo el enfoque presentado en el capítulo anterior en un robot real. Hasta ahora se ha desarrollado e integrado el manejador de los sensores, el controlador de movimiento y el módulo de procesamiento de los datos láser en el robot real. Estamos trabajando en la integración del planificador.

La implementación parcial en robots reales para el problema de construcción de mapas con múltiples robots se llevó a cabo con dos robots, uno con mejores capacidades de movimiento que el otro y utilizando dos láser, uno con mayor rango y precisión que el otro.

En las siguientes secciones de este capítulo hablaré sobre la implementación que se realizó en los robots de manejo diferencial para el problema de construcción de mapas.

5.1. Hardware utilizado

A continuación se detalla cuales fueron las plataformas sobre las cuales se llevó a cabo la implementación y cuales fueron los sensores que se utilizaron.

5.1.1. Robots

Utilizamos como plataformas para la implementación dos robots móviles, un Pioneer P3-DX (ver Fig. 5.1(a)) y un Pioneer P3-AT (ver Fig. 5.1(b)). El Pioneer P3-DX es un robot móvil compacto de manejo diferencial que cuenta con una computadora abordo, con un procesador de 400 MHz y 1 GB de memoria RAM. Además, el Pioneer P3-DX cuenta también con dos ruedas con motores independientes; cada rueda tiene un *encoder* o generador de pulsos que permite medir cuanto es que ésta ha girado. El Pioneer P3- AT es un robot móvil de cuatro ruedas, sin embargo, puede comportarse como lo hace un robot de manejo diferencial. Este robot cuenta con una computadora abordo con un procesador de 400 MHz y 1 GB de memoria RAM. Al igual que el Pioneer P3-DX, este robot cuenta con motores independientes en cada rueda y *encoders*.

La plataforma robótica Pioneer P3-DX puede alcanzar velocidades de 1.6 m/s y llevar una carga de hasta 26 Kg, mientras que la plataforma Pioneer P3-AT puede alcanzar velocidades de 0.8 m/s y llevar una carga de hasta 12 Kg.

El Pioneer P3-DX y el Pioneer P3-AT son robots móviles que proporcionan una plataforma para diversos fines de investigación. Estos robots están fabricados por la empresa MobileRobots Inc. (ActivMedia Robotics).

(a) Pioneer P3-DX (b) Pioneer P3-AT

Figura 5.1: Plataformas robóticas.

Ambas plataformas robóticas fueron equipadas con sensores, los cuales son necesarios para poder realizar la tarea de exploración. A continuación se describen los sensores que fueron montados sobre estos robots.

5.1.2. Lásers

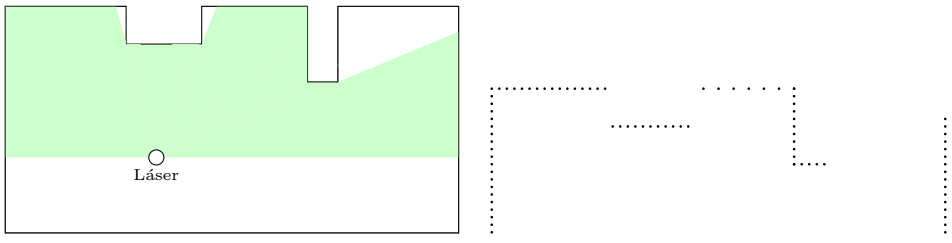
Los robots están equipados con sensores láser. Estos sensores permiten determinar la presencia de obstáculos en el ambiente. A partir de las lecturas de los sensores es posible realizar la construcción del mapa del ambiente que se está explorando.

Para esta implementación se utilizaron dos tipos de sensores láser. El primero es un sensor láser del tipo Sick LMS 200 como el que se muestra en la Fig. 5.2(a). Este sensor mide la distancia a los objetos sobre un plano en un rango de 180 grados, con una precisión angular de 0.25 grados y un alcance de hasta 10 metros. El segundo es un sensor láser del tipo Hokuyo como el que se muestra en la Fig. 5.2(b). Este sensor mide la distancia a los objetos sobre un plano en un rango de 240 grados, con una precisión angular de 0.35 grados y un alcance de hasta 5 metros.

(a) Sick LMS 200 (b) Hokuyo

Figura 5.2: Sensores láser

En la Fig. 5.3 podemos observar un ejemplo de las lecturas de un sensor láser.



(a) Ambiente (b) Lecturas del Láser

Figura 5.3: Ejemplo de simulación de datos obtenidos por el láser.

5.2. Software

El software está constituido por diferentes módulos los cuales realizan funciones específicas. Debido a la poca capacidad de procesamiento con la cual cuentan los robots, sólo algunos de los módulos son ejecutados directamente en ellos, mientras que el resto son ejecutados en una computadora portátil. Los módulos principales que constituyen la arquitectura de nuestro software son: Planificador de movimiento, interfaz gráfica del usuario (GUI), controlador del movimiento del Pioneer P3-DX, controlador del movimiento del Pioneer P3-AT, manejador del láser Sick LMS 200 y manejador del láser Hokuyo.

El esquema de la arquitectura del software se muestra en la Fig. 5.4. En este esquema podemos observar que solamente los módulos referentes a los sensores y al movimiento del robot se encuentran implementados sobre las plataformas robóticas, mientras que el módulo de planificación (Planificador) y la interfaz gráfica para el usuario (GUI), están implementados en una computadora portátil, la cual tiene un capacidad de procesamiento muy superior a la computadora que se encuentra abordo de ambos robots.

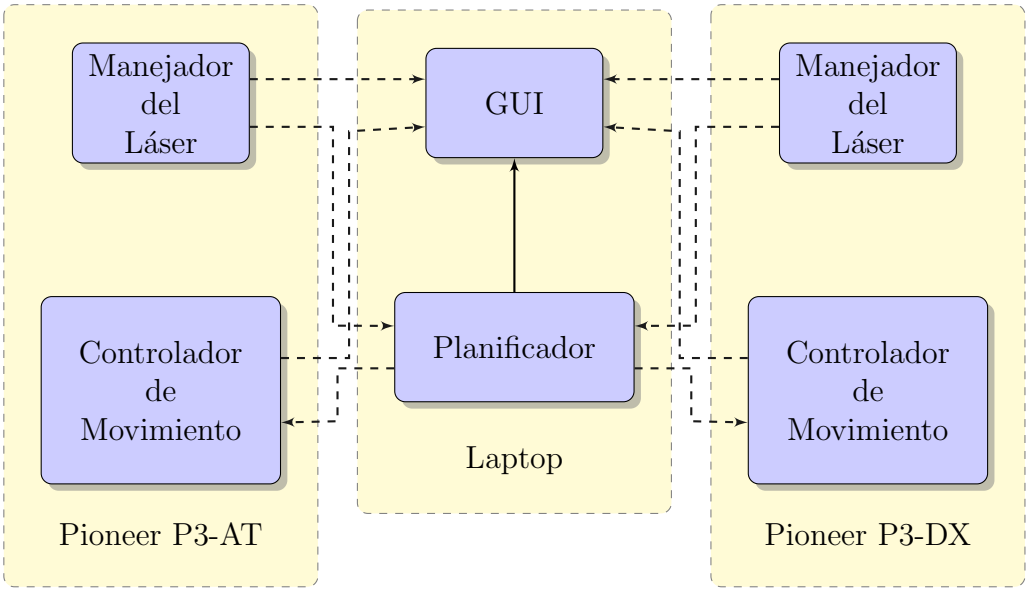


Figura 5.4: Arquitectura del software.

El software fue implementado usando C++ y un sistema operativo para robots, ROS [53], por sus siglas en inglés. ROS constituye un marco de trabajo de código abierto para el desarrollo de software para robots, que provee de librerías y herramientas para ayudar en la creación de aplicaciones robóticas. Además, también provee los servicios estándar de un sistema operativo tales como abstracción del hardware, control de dispositivos de bajo nivel, implementación de funcionalidad de uso común, paso de mensajes entre procesos y mantenimiento de paquetes. En general, un sistema construido utilizando ROS consiste en un número de procesos conectados en una red punto a punto, en donde es posible compartir e intercambiar información de forma directa entre dos o más procesos, mediante un protocolo de comunicación HTTP (Hypertext Transfer Protocol).

Cada uno de los módulos de nuestro software se describen con mayor detalle en las siguientes subsecciones.

5.2.1. GUI

La interfaz gráfica de usuario (GUI) provee de información relacionada directamente con los sensores montados sobre las plataformas robóticas. Esta interfaz está compuesta de diferentes ventanas en donde es posible visualizar los datos adquiridos por los sensores láser, así como una aproximación de la posición del robot sobre el mapa en el que se está moviendo. Este módulo fue implementado sobre una computadora portátil en donde los datos de entrada son recibidos de forma inalámbrica para luego ser procesados y finalmente desplegados.

Esta interfaz es importante ya que da al usuario una idea de lo que está sucediendo durante el proceso de exploración.

5.2.2. Planificador

El módulo de planificación es la parte central del sistema. Este módulo fue implementado en C++ y ya ha sido probado en simulación. Actualmente se está llevando a cabo la integración de este módulo para ser probado con los robots reales. Para esta integración estamos utilizando ROS para gestionar el envío de información desde los robots hacia la computadora portátil y viceversa.

En el módulo de planificación es en donde se encuentra implementado el algoritmo que genera la política de movimiento. Dicho módulo indica a cada robot cuándo es que éstos deben ejecutar esta política y así continuar con el proceso de exploración.

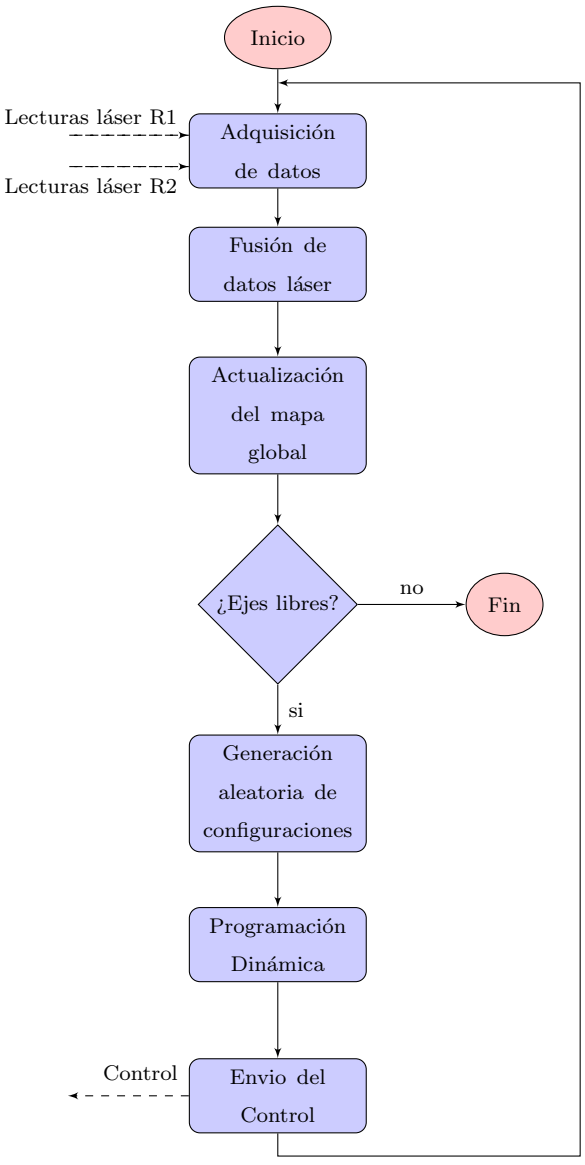


Figura 5.5: Diagrama de flujo del Planificador

En la Fig 5.5 se muestra el diagrama de flujo del algoritmo de planificación. Este algoritmo recibe como entrada las lecturas de los sensores láser que se encuentran montados sobre los robots. Estas lecturas son fusionadas con el mapa global conocido hasta ese instante de tiempo. De esta forma el mapa global es actualizado y a partir de esta actualización se determinan los ejes libres. Si ya no existen ejes libres, el proceso de exploración ha concluido. En caso contrario, se generan configuraciones (de forma alea-

toria) cercanas a los ejes libres. Después, utilizamos programación dinámica estocástica para generar una política de movimiento, es decir, la secuencia de acciones que los robots deben ejecutar; estos controles son enviados a través de la red inalámbrica hacia los robots para su ejecución. El proceso se repite hasta que no existen más ejes libres en el mapa global.

5.2.3. Manejador del Láser

El manejador del láser es el módulo encargado de configurar, inicializar y tomar los datos del sensor láser. Estos datos nos proporcionan información sobre la distancia que existe entre el sensor y los elementos del ambiente. Las distancias están dadas por un conjunto de puntos en coordenadas polares referenciadas al centro del láser. Una vez que una lectura del sensor láser está lista, puede ser consultada por cualquier módulo que la requiera.

5.2.4. Controlador de Movimiento

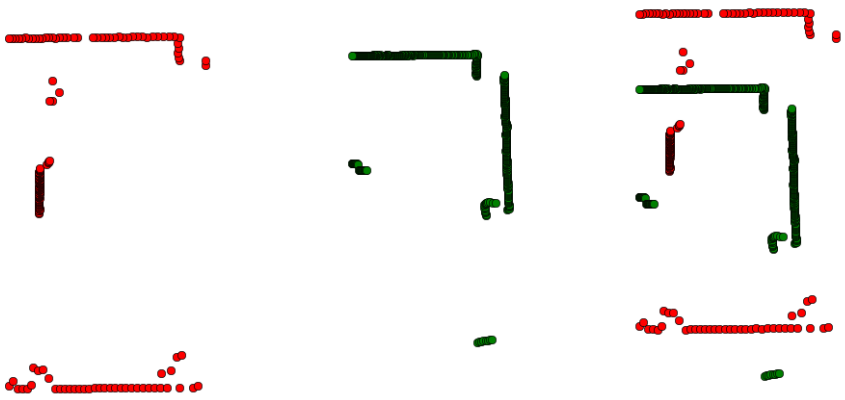
El controlador de movimiento es el módulo encargado de ejecutar los controles dictados por el módulo de planificación, es decir, mover al robot de una configuración a otra. Para alcanzar una configuración el robot ejecuta dos primitivas de movimiento, rotaciones en sitio y traslaciones en línea recta.

5.2.5. Experimento

El objetivo de este experimento es mostrar el adecuado funcionamiento de los módulos de adquisición de datos y su alineamiento, de tal forma que sea posible construir un mapa global del ambiente durante el proceso de exploración. En la Fig. 5.6 se pueden observar las configuraciones iniciales de los robots y el ambiente en el cual se realizó el experimento.

En la Fig. 5.7(a) y 5.7(b) se muestran las lecturas tomadas por los sensores láser montados sobres los robots P3-DX y P3-AT en el tiempo t. En la Fig. 5.7(c) observamos que ambas lecturas tienen como marco de referencia el origen. Lo anterior representa un problema ya que todos los datos adquiridos en cualquier instante de tiempo, siempre tendrán como marco de referencia el origen. Para solucionar este problema, seleccionamos como marco de referencia global a alguno de los marcos de referencia establecidos

Figura 5.6: Experimento.



(a) Lecturas del P3-DX. (b) Lecturas del P3-AT. (c) Superposición de las lecturas.

Figura 5.7: Lecturas láser.

inicialmente por uno de los dos robots. Para este experimento, seleccionamos como marco de referencia global, el marco establecido por la posición inicial del robot P3-DX.

Una vez establecido el marco de referencia global, realizamos alineamiento de los datos. En la Fig. 5.8 se muestra el resultado del alineamiento. Los puntos verdes representan los datos provenientes del sensor montado sobre el robot P3-AT y en rojo los datos provenientes del sensor montado sobre el robot P3-DX.

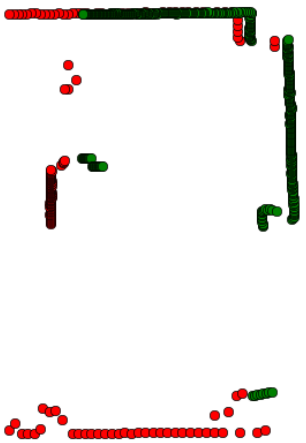


Figura 5.8: Alineamiento de los datos.

Después de haber alineado los datos, los robots fueron llevados hacia otras configuraciones. En el robot P3-DX se ejecutó una rotación en sitio y enseguida una traslación en linea recta, mientras que en el robot P3-AT sólo se llevó a cabo una rotación en sitio. Una vez finalizadas estas acciones se procedió con la adquisición de datos. En la Fig 5.9 podemos observar en azul el conjunto de puntos que representa el mapa global hasta el instante de tiempo t, en verde los datos provenientes del sensor montado sobre el robot P3-AT y en rojo los datos provenientes del sensor montado sobre el robot P3-DX, en el instante de tiempo $t + 1$.

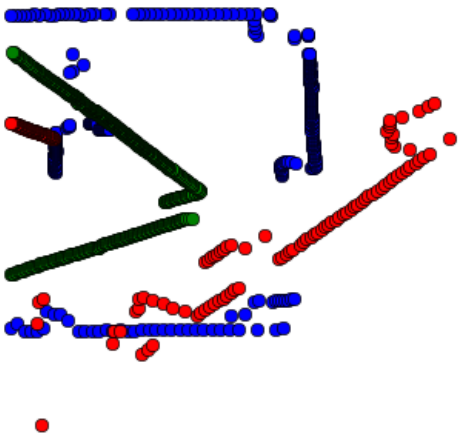


Figura 5.9: Mapa global y lecturas láser.

Una vez que hemos adquirido los nuevos datos láser, procedemos al alineamiento de los mismos con el mapa global generado en el instante de tiempo t. En la Fig. 5.10 se muestra los datos láser alineados con el mapa global que se tenía.

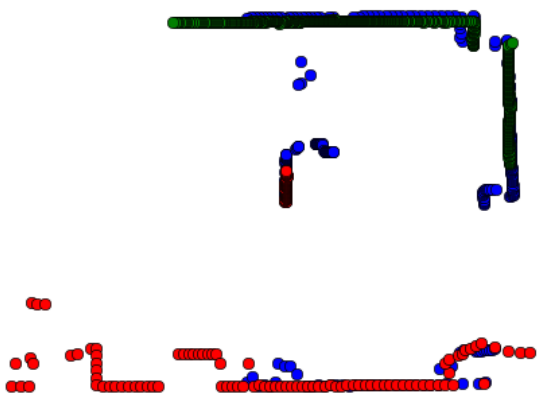


Figura 5.10: Mapa global y lecturas láser alineadas.

En la Fig. 5.11 tenemos finalmente el mapa global generado después de haber adquirido los datos desde las configuraciones iniciales, alinear estos datos, mover a los robots, adquirir nuevos datos y volver a realizar el alineamiento.

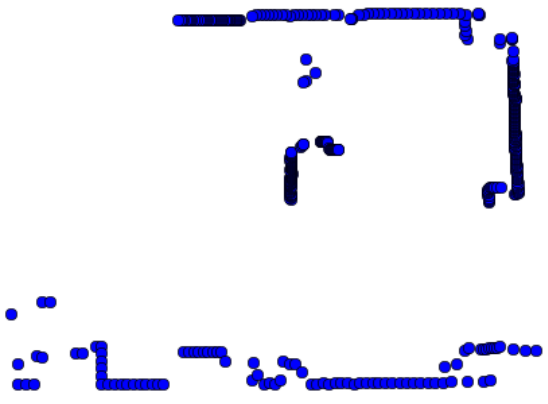


Figura 5.11: Mapa global.

Con este experimento se mostró que tanto la adquisición de datos láser como el alineamiento se realizan de forma adecuada. El comportamiento adecuado de estos dos

procesos es importante, ya que son la base para poder construir gradualmente el mapa global del ambiente que se está explorando. Como trabajo futuro inmediato, deseamos implementar el módulo de planificación de movimiento en los robots reales.

Capítulo 6

Conclusiones y Trabajo Futuro

En este trabajo, hemos propuesto una estrategia de exploración para la construcción de mapas utilizando múltiples robots. Hemos modelado el problema de exploración, como una tarea de asignación de recursos. Además, consideramos robots heterogéneos con diferentes capacidades de sensado y de movimiento. Utilizamos programación dinámica con estado de información imperfecta (también llamada programación dinámica estocástica) como herramienta para asignar un robot (el cual es el recurso) para explorar una parte del ambiente (la cual es la tarea), de acuerdo a sus capacidades. Hemos propuesto un enfoque Bayesiano para estimar la probabilidad de la siguiente observación dado el estado del equipo de robots. Así, una observación corresponde a una clase, y existe una relación entre el tipo de mapa local (clase) y una parte del espacio de estados. Esto nos permite elegir el robot más apropiado de acuerdo a sus capacidades de sensado y movimiento para explorar esa parte del espacio de estados. También hemos propuesto una estrategia de optimización de un paso adelante con base en muestreo, lo que mejora tanto la calidad del plan como el tiempo para generarlo. Nuestro enfoque de un paso adelante evita movimientos innecesarios de los robots que podrían derivarse de la falta de información sobre el área no explorada del ambiente.

Una desventaja de nuestro método es que nuestro planificador es centralizado y además, supone que toda la información de cada uno de los robots en un cierto instante de tiempo k está disponible para la toma de decisiones. Como trabajo futuro, vamos a investigar métodos para distribuir el proceso de planificación. Finalmente, deseamos probar el método propuesto en robots reales.

Apéndices

Apéndice A

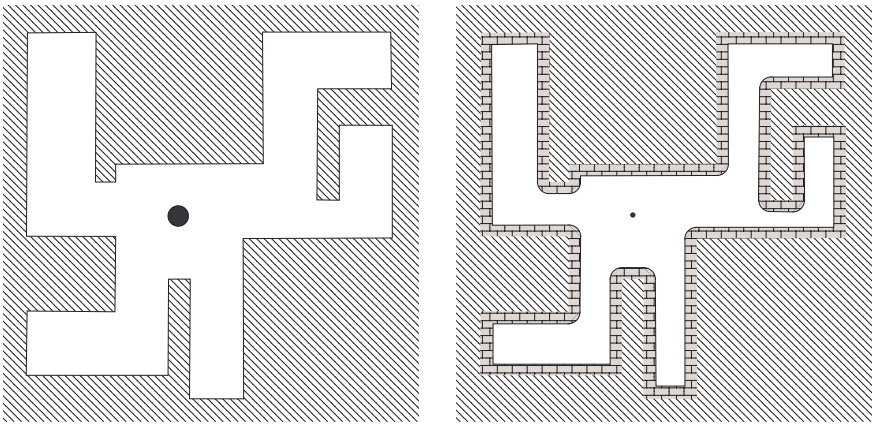
Espacio de configuraciones $\mathcal{C}$

Una configuración de un robot es la especificación de las posiciones y orientaciones de todos los puntos del robot relativos a un sistema de coordenadas fijo.

El espacio de configuraciones es el espacio de todas las posibles configuraciones del robot. En el espacio de configuraciones cualquier robot sin importar su forma geométrica puede ser representado por un punto. $\mathcal{C}$ puede tener una topología no euclidiana. El espacio de configuraciones se divide en el espacio de configuraciones libre $\mathcal{C}_{free}$ y el espacio de configuraciones obstáculo $\mathcal{C}_{obs}$.

Para un robot tipo disco el espacio de configuraciones corresponde simplemente al polígono original (también llamado el espacio de trabajo) expandido por el radio del robot disco. En la Fig. A.1(b) se muestra el espacio de configuraciones para un robot tipo disco. El espacio de trabajo es mostrado en las Fig. A.1(a). La expansión de un entorno poligonal se calcula mediante una suma de Minkowski entre el disco que representa al robot y el polígono que representa el espacio de trabajo.

La suma de Minkowski corresponde a una convolución del robot disco con el polígono. Es decir, el robot disco se mantiene en contacto con la frontera del polígono y la curva que dibuja el centro del disco es la frontera entre el espacio de configuraciones obstáculo $\mathcal{C}_{obs}$ y el espacio de configuraciones libre $\mathcal{C}_{free}$.



(a) Espacio de trabajo (b) Espacio de configuraciones

Figura A.1: Espacio de Configuraciones para un robot tipo disco

Apéndice B

Grafo de visibilidad reducido

El grafo de visibilidad reducido de un polígono es aquel en donde los nodos del grafo son los vértices reflejos del polígono y las aristas son segmentos tangentes de pares de vértices reflejos que son visibles entre si o un segmento perteneciente al polígono original cuyos puntos terminales son vértices reflejo.

Un vértice reflejo es aquel vértice de un polígono cuyo ángulo interior es mayor a π.

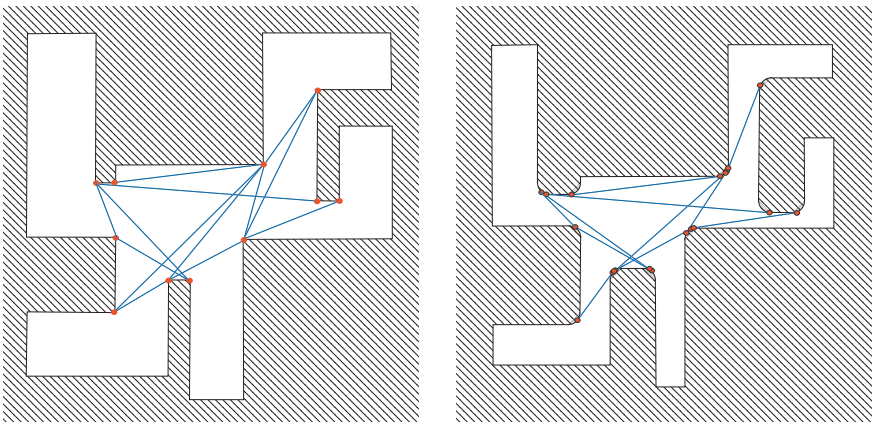
Si L es la línea que pasa por el par (p, q) de vértices reflejos. El segmento $\overline{pq}$ es un segmento tangente si L es una línea bi-tangente al polígono en los vértices p y q.

Los vértices p y q son visibles si el segmento $\overline{pq}$ no intersecta el polígono.

En la Fig. B.1 se muestran ejemplos de los grafos de visibilidad reducido. En la Fig. B.1(a) se muestra el grafo de visibilidad reducido sobre el polígono que representa el mapa donde el robot se mueve (también llamado espacio de trabajo). En la Fig. B.1(b) se muestra el grafo de visibilidad reducido sobre el espacio de configuraciones de un robot con forma de disco.

Note que el espacio de configuraciones de un robot puntal es igual que su espacio de trabajo, es decir, el polígono no se altera.

Una propiedad importante del grafo de visibilidad reducido es que permite mover al robot en la más corta trayectoria en términos de distancia Euclidiana, entre cualquier dos configuraciones no visibles entre ellas. Un punto en el polígono es visible a otro si el segmento de línea que los une no intersecta los obstáculos que representan el polígono.



(a) Espacio de trabajo (b) Espacio de configuraciones

Figura B.1: Grafo de visibilidad reducido en: (a) el espacio de trabajo y (b) el espacio de configuraciones.

Apéndice C

Deducción de la Ecuación $P(x_k|I_k)$

En este apéndice se muestra la deducción de la ecuación 4.13. Esta ecuación, tal y como se indicó en la sub-sección 3.2.2 representa la probabilidad de estar en el estado x_k considerando los controles aplicados y las observaciones obtenidas.

Por definición se sabe que:

$$P(x_k|I_k) = \frac{P(x_k, I_k)}{P(I_k)} \tag{C.1}$$

donde I_k es el vector de información y está dado por:

$$I_k = (I_{k-1}, u_{k-1}, z_k)$$

Sustituyendo I_k al lado derecho de C.1 tenemos,

$$P(x_k|I_k) = \frac{P(x_k, I_{k-1}, u_{k-1}, z_k)}{P(I_{k-1}, u_{k-1}, z_k)} \tag{C.2}$$

Por otro lado, se sabe que se puede expresar a la probabilidad conjunta como un producto de probabilidades condicionales, de tal forma que:

$$P(a_1, a_2, a_3, \ldots, a_n) = P(a_1)P(a_2|a_1)P(a_3|a_1, a_2)\cdots P(a_n|a_1, a_2, \ldots, a_{n-1})$$

Si aplicamos esta definición al lado derecho de (C.2) se llega a la expresión,

$$P(x_k|I_k) = \frac{P(I_{k-1})P(u_{k-1}|I_{k-1})P(x_k|I_{k-1}, u_{k-1})P(z_k|I_{k-1}, u_{k-1}, x_k)}{P(I_{k-1})P(u_{k-1}|I_{k-1})P(z_k|I_{k-1}, u_{k-1})} \tag{C.3}$$

Al eliminar los factores repetidos tanto en el numerador como en el denominador se obtiene,

$$P(x_k|I_k) = \frac{P(x_k|I_{k-1}, u_{k-1})P(z_k|I_{k-1}, u_{k-1}, x_k)}{P(z_k|I_{k-1}, u_{k-1})} \tag{C.4}$$

Si consideramos que una observación depende del lugar donde es tomada, es decir, la observación depende solo del estado del sistema, entonces podemos decir que,

$$z_k \perp I_{k-1}, u_{k-1}|\ x_k$$

Entonces:

$$P(z_k|I_{k-1}, u_{k-1}, x_k) = P(z_k|x_k) \tag{C.5}$$

Sustituyendo C.5 en C.4 tenemos,

$$P(x_k|I_k) = \frac{P(x_k|I_{k-1}, u_{k-1})P(z_k|x_k)}{P(z_k|I_{k-1}, u_{k-1})} \tag{C.6}$$

Ahora tomemos el término del denominador de la ecuación C.6

$$P(z_k|I_{k-1}, u_{k-1})$$

Aplicando la regal de Bayes,

$$P(z_k|I_{k-1}, u_{k-1}) = \frac{P(I_{k-1}, u_{k-1}|z_k)P(z_k)}{P(I_{k-1}, u_{k-1})} \tag{C.7}$$

$$\therefore$$

$$P(z_k|I_{k-1}, u_{k-1}) = \frac{P(I_{k-1}, u_{k-1}, z_k)}{P(I_{k-1}, u_{k-1})} \tag{C.8}$$

Considerando que la marginalización indica $P(a) = \sum_b P(b, a)$, entonces, si marginalizamos (C.8) respecto de x_k,

$$P(z_k|I_{k-1}, u_{k-1}) = \sum_{x_k} \frac{P(I_{k-1}, u_{k-1}, z_k, x_k)}{P(I_{k-1}, u_{k-1})} \tag{C.9}$$

Si aplicamos la regla del producto para la probabilidad conjunta a la ecuación C.9 llegamos a,

$$P(z_k|I_{k-1},u_{k-1}) = \sum_{x_k} \frac{P(u_{k-1})P(I_{k-1}|u_{k-1})P(x_k|I_{k-1},u_{k-1})P(z_k|x_k,I_{k-1},u_{k-1})}{P(u_{k-1})P(I_{k-1}|u_{k-1})} \tag{C.10}$$

Al simplificar la expresión y considerando la independencia condicional de la ecuación C.5 tenemos,

$$P(z_k|I_{k-1},u_{k-1}) = \sum_{x_k} P(x_k|I_{k-1},u_{k-1})P(z_k|x_k) \tag{C.11}$$

Sustituyendo la ecuación C.11 en la ecuación C.6 tenemos,

$$P(x_k|I_k) = \frac{P(x_k|I_{k-1},u_{k-1})P(z_k|x_k)}{\sum\limits_{x_k} P(x_k|I_{k-1},u_{k-1})P(z_k|x_k)} \tag{C.12}$$

Consideremos ahora el término $P(x_k|I_{k-1},u_{k-1})$ que aparece en la ecuación C.12 y apliquemos la regla de Bayes

$$\begin{aligned} P(x_k|I_{k-1},u_{k-1}) &= \frac{P(I_{k-1},u_{k-1}|x_k)P(x_k)}{P(I_{k-1},u_{k-1})} \\ &= \frac{P(I_{k-1},u_{k-1},x_k)}{P(I_{k-1},u_{k-1})} \end{aligned} \tag{C.13}$$

Si marginalizamos (C.13) respecto a x_{k-1}

$$P(x_k|I_{k-1},u_{k-1}) = \sum_{x_{k-1}} \frac{P(x_k,I_{k-1},u_{k-1},x_{k-1})}{P(I_{k-1},u_{k-1})} \tag{C.14}$$

Si a (C.14) le aplicamos la regla del producto para la probabilidad conjunta,

$$P(x_k|I_{k-1},u_{k-1}) =$$

$$\sum_{x_{k-1}} \frac{P(u_{k-1})P(I_{k-1}|u_{k-1})P(x_{k-1}|I_{k-1},u_{k-1})P(x_k|x_{k-1},I_{k-1},u_{k-1})}{P(u_{k-1})P(I_{k-1}|u_{k-1})} \tag{C.15}$$

Al simplificar esta expresión resulta:

$$P(x_k|I_{k-1},u_{k-1}) = \sum_{x_{k-1}} P(x_{k-1}|I_{k-1},u_{k-1})P(x_k|x_{k-1},I_{k-1},u_{k-1}) \tag{C.16}$$

Ahora se consideran las siguientes suposiciones de independencia:

- Se puede suponer que $x_{k-1} \perp u_{k-1} \mid I_{k-1}$. Esto debido a que el estado del sistema en $k-1$ no depende del control u_{k-1} que se aplicará para llegar al estado k.

$$P(x_{k-1}|I_{k-1},u_{k-1}) = P(x_{k-1}|I_{k-1})$$

- Por otro lado se puede suponer que $x_k \perp I_{k-1} \mid x_{k-1}, u_{k-1}$. Esto debido a que si se conoce el estado en $k-1$ y el control que se aplicará, entonces se puede no tomar en cuenta todo lo que se encuentra en el vector de información.

$$P(x_k|x_{k-1},I_{k-1},u_{k-1}) = P(x_k|x_{k-1},u_{k-1})$$

A partir de estas suposiciones podemos reescribir la ecuación C.16 como:

$$P(x_k|I_{k-1},u_{k-1}) = \sum_{x_{k-1}} P(x_{k-1}|I_{k-1})P(x_k|x_{k-1},u_{k-1}) \tag{C.17}$$

Si sustituimos (C.17) en (C.12)

$$P(x_k|I_k) = \frac{\sum\limits_{x_{k-1}} P(x_{k-1}|I_{k-1})P(x_k|x_{k-1},u_{k-1})P(z_k|x_k)}{\sum\limits_{x_k}\sum\limits_{x_{k-1}} P(x_{k-1}|I_{k-1})P(x_k|x_{k-1},u_{k-1})P(z_k|x_k)} \tag{C.18}$$

De esta manera se ha logrado demostrar que la probabilidad del estado del sistema dado el vector de información está en términos del modelo de observación y del modelo de movimiento.

Apéndice D

Optimalidad del Algoritmo de D.P. con Estado de Información perfecta

La siguiente demostración fue tomada de [39].

Demostración: Para cualquier política admisible $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$ y cada $k = 0, 1, \ldots, N-1$, denotemos $\pi^k = \{\mu_k, \mu_{k+1}, \ldots, \mu_{N-1}\}$. Para $k = 0, 1, \ldots, N-1$. sea $J_k^*(x_k)$ el costo óptimo para los $(N-k)$ estados que comienzan en x_k al tiempo k, y terminan al tiempo N, esto es,

$$J_k^*(x_k) = \max_{\pi^k} \underset{w_k \ldots, w_{N-1}}{E} \left\{ g_N(x_N) + \sum_{i=k}^{N-1} g_i(x_i, \mu_i(x_i), w_i) \right\}$$

Para $k = N$, definimos $J_N^*(x_N) = g_N(x_N)$. Se mostrará por inducción que la función J_k^* es igual a la función J_k generada por el algoritmo del D. P., por tanto para $k = 0$, se obtendrá el resultado deseado. En efecto, por definición tenemos $J_N^* = J_N = g_N$. Asumamos que para algún k y todos los x_{k+1}, tenemos $J_{k+1}^*(x_{k+1}) = J_{k+1}(x_{k+1})$.

Entonces, ya que $\pi^k = (\mu_k, \pi^{k+1})$, tenemos para toda x_k,

$$J_k^*(x_k) = \max_{(\mu_k,\pi^{k+1})} \underset{w_k,\ldots,w_{N-1}}{E} \left\{ g_N(x_N) + g_k(x_k, \mu_k(x_k), w_k) + \sum_{i=k+1}^{N-1} g_i(x_i, \mu_i(x_i), w_i) \right\}$$

$$= \max_{\mu_k} \underset{w_k}{E} \left\{ g_k(x_k, \mu_k(x_k), w_k) + \max_{\pi^{k+1}} \left[\underset{w_{k+1},\ldots,w_{N-1}}{E} \left\{ g_N(x_N) + \sum_{i=k+1}^{N-1} g_i(x_i, \mu_i(x_i), w_i) \right\} \right] \right\}$$

$$= \max_{\mu_k} \underset{w_k}{E} \left\{ g_k(x_k, \mu_k(x_k), w_k) + J_{k+1}^*\Big(f_k(x_k, \mu_k(x_k), w_k) \Big) \right\}$$

Ya que se asume que $J_{k+1}^*(x_k + 1) = J_{k+1}(x_{k+1})$, tenemos,

$$J_k^*(x_k) = \max_{\mu_k} \underset{w_k}{E} \left\{ g_k(x_k, \mu_k(x_k), w_k) + J_{k+1}\Big(f_k(x_k, \mu_k(x_k), w_k) \Big) \right\}$$

$$= \max_{u_k \in U_k(x_k)} \underset{w_k}{E} \left\{ g_k(x_k, \mu_k(x_k), w_k) + J_{k+1}\Big(f_k(x_k, \mu_k(x_k), w_k) \Big) \right\}$$

$$\therefore$$

$$J_k^*(x_k) = J_k(x_k)$$

Los argumentos de la demostración anterior proveen una interpretación de $J_k(x_k)$ como el costo óptimo para un problema de $(N - k)$ estados partiendo del estado x_k al tiempo k, y terminando en el tiempo N. Consecuentemente llamamos a $J_k(x_k)$ el costo del estado x_k, y nos referimos a J_k como la función de costo al tiempo k.

Apéndice E

Ejemplo de programación dinámica con estado de información imperfecta

En este apéndice se muestra un ejemplo en donde se quiere calcular la política para un sistema en donde el estado no es observable. El sistema está representado por el grafo mostrado en la Fig. E.1.

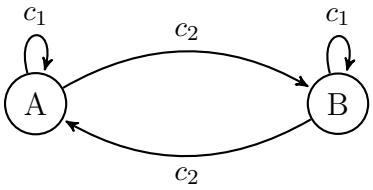


Figura E.1: Grafo de dos estados.

En la Fig. E.1 A y B representan los posibles estados, c_1 es el control que indica permanecer en el mismo estado, mientras que c_2 es el control que produce un cambio de estado. Además, como se estableció en un principio, el estado no es observable, pero puede ser estimado a través de las observaciones, en particular para este ejemplo se consideran dos posibles observaciones o_1 y o_2.

La ganancia final de estar en los estados A y B es:

$$g_N(x_N = A) = 1 \qquad g_N(x_N = B) = 2$$

El modelo de movimiento ($P(x_{k+1}|x_k, u_k)$) está dado por:

$$\begin{array}{ll} P(x_{k+1} = A|x_k = A, u_k = c1) & = 0.7 \\ P(x_{k+1} = B|x_k = A, u_k = c1) & = 0.3 \end{array} \qquad \begin{array}{ll} P(x_{k+1} = A|x_k = B, u_k = c1) & = 0.3 \\ P(x_{k+1} = B|x_k = B, u_k = c1) & = 0.7 \end{array}$$

$$\begin{array}{ll} P(x_{k+1} = A|x_k = A, u_k = c2) & = 0.1 \\ P(x_{k+1} = B|x_k = A, u_k = c2) & = 0.9 \end{array} \qquad \begin{array}{ll} P(x_{k+1} = A|x_k = B, u_k = c2) & = 0.9 \\ P(x_{k+1} = B|x_k = B, u_k = c2) & = 0.1 \end{array}$$

El modelo de observación es ($P(z_k|x_k)$) el siguiente:

$$\begin{array}{ll} P(z_k = o_1|x_k = A) & = 0.8 \\ P(z_k = o_2|x_k = A) & = 0.2 \\ P(z_k = o_1|x_k = B) & = 0.6 \\ P(z_k = o_2|x_k = B) & = 0.4 \end{array}$$

Además, la función objetivo ($g(x_k, u_k, w_k)$) está determinada por:

$$\begin{array}{l} g(x_k = A, u_k = c_1, x_{k+1} = A) = 0.5 \\ g(x_k = A, u_k = c_1, x_{k+1} = B) = 0.75 \\ g(x_k = B, u_k = c_1, x_{k+1} = A) = 0.6 \\ g(x_k = B, u_k = c_1, x_{k+1} = B) = 0.4 \\ g(x_k = A, u_k = c_2, x_{k+1} = A) = 0.4 \\ g(x_k = A, u_k = c_2, x_{k+1} = B) = 1.2 \\ g(x_k = B, u_k = c_2, x_{k+1} = A) = 1.5 \\ g(x_k = B, u_k = c_2, x_{k+1} = B) = 0.7 \end{array}$$

Considerando toda la información anterior, utilicemos el algoritmo de programación dinámica con estado de información imperfecta para un horizonte de planificación $N = 2$.

Para hacer más claro el cálculo de la política podemos generar el árbol de posibilidades (Fig. E.2) utilizando el vector de información I_k.

Utilizamos el algoritmo de programación dinámica para generar la política para este sistema. Este algoritmo está dado por:

$$J_{N-1}(I_{N-1}) = \min_{u_{N-1} \in U} \tilde{g}_{N-1}(I_{N-1}, u_{N-1})$$

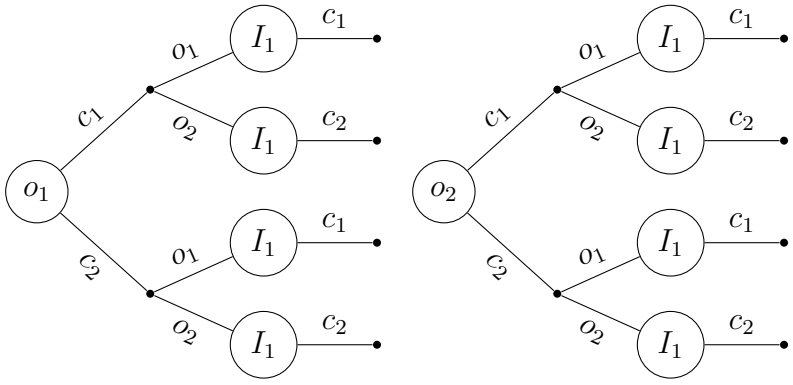


Figura E.2: Árbol de posibilidades.

$$J_k(I_k) = \min_{u_k \in U} \left[\tilde{g}_k(I_k, u_k) + \underset{z_{k+1}}{E} \left\{ J_{k+1}(I_k, z_{k+1}, u_k) \mid I_k, u_k \right\} \right]$$

Mediante el desarrollo de la expresión en términos de las probabilidades conocidas, tenemos:

$$\begin{aligned}
J_{N-1}(I_{N-1}) &= \min_{u_{N-1} \in U} \tilde{g}_{N-1}(I_{N-1}, u_{N-1}) \\
&= \min_{u_{N-1} \in U} \left\{ \underset{x_{N-1}}{E} \hat{g}_{N-1}(x_{N-1}, u_{N-1}) \mid I_k, u_k \right. \\
&\quad + \underset{x_{N-1}}{E} \left\{ \underset{x_N}{E} g_N | x_{N-1}, u_{N-1} \right\} | I_{N-1}, u_{N-1} \Big\} \\
&= \min_{u_{N-1} \in U} \left\{ \sum_{x_{N-1}} \hat{g}_{N-1}(x_{N-1}, u_{N-1}) \; P(x_{N-1} | I_{N-1}) \right. \\
&\quad \left. + \sum_{x_{N-1}} \sum_{x_N} g_N(x_N) P(x_N | x_{N-1}, u_{N-1}) P(x_{N-1} | I_{N-1}) \right\}
\end{aligned}$$

$$J_k(I_k) = \min_{u_k \in U} \left[\tilde{g}_k(I_k, u_k) + \underset{z_{k+1}}{E} \left\{ J_{k+1}(I_k, z_{k+1}, u_k) \mid I_k, u_k \right\} \right]$$

$$= \min_{u_k \in U} \left[\underset{x_k}{E}\, \hat{g}(x_x, u_x) \mid I_k, u_k + \underset{z_{k+1}}{E} \left\{ J_{k+1}(I_k, z_{k+1}, u_k) \mid I_k, u_k \right\} \right]$$

$$= \min_{u_k \in U} \left[\sum_{x_k} \hat{g}(x_k, u_k)\ P(x_k|I_k) \right.$$

$$\left. + \sum_{z_{k+1}} J_{k+1}(I_k, z_{k+1}, u_k)\ P(z_{k+1}|I_k, u_k) \right]$$

$$J_k(I_k) = \min_{u_k \in U} \left[\sum_{x_k} \hat{g}(x_k, u_k)\ P(x_k|I_k) \right.$$

$$\left. + \sum_{z_{k+1}} J_{k+1}(I_k, z_{k+1}, u_k) \sum_{x_{k+1}} P(z_{k+1}|I_k, u_k, x_{k+1})\ P(x_{k+1}|I_k, u_k) \right]$$

$$= \min_{u_k \in U} \left[\sum_{x_k} \hat{g}(x_k, u_k)\ P(x_k|I_k) \right.$$

$$\left. + \sum_{z_{k+1}} J_{k+1}(I_k, z_{k+1}, u_k) \sum_{x_{k+1}} P(z_{k+1}|x_{k+1})\ P(x_{k+1}|I_k, u_k) \right]$$

$$= \min_{u_k \in U} \left[\sum_{x_k} \hat{g}(x_k, u_k)\ P(x_k|I_k) \right.$$

$$\left. + \sum_{z_{k+1}} J_{k+1}(I_k, z_{k+1}, u_k) \sum_{x_{k+1}} P(z_{k+1}|x_{k+1}) \sum_{x_k} P(x_{k+1}|I_k, u_k, x_k)\ P(x_k|I_k, u_k) \right]$$

$$= \min_{u_k \in U} \left[\sum_{x_k} \hat{g}(x_k, u_k)\ P(x_k|I_k) \right.$$

$$\left. + \sum_{z_{k+1}} J_{k+1}(I_k, z_{k+1}, u_k) \sum_{x_{k+1}} P(z_{k+1}|x_{k+1}) \sum_{x_k} P(x_{k+1}|x_k, u_k)\ P(x_k|I_k) \right]$$

Por lo tanto, el algoritmo de programación dinámica puede ser reescrito como:

$$J_{N-1}(I_{N-1}) = \min_{u_{N-1}\in U} \left\{ \sum_{x_{N-1}} \hat{g}_{N-1}(x_{N-1}, u_{N-1})\ P(x_{N-1}|I_{N-1}) \right.$$
$$\left. + \sum_{x_{N-1}} \sum_{x_N} g_N(x_N) P(x_N|x_{N-1}, u_{N-1}) P(x_{N-1}|I_{N-1}) \right\}$$

$$J_k(I_k) = \min_{u_k\in U} \left[\sum_{x_k} \hat{g}(x_k, u_k)\ P(x_k|I_k) \right.$$
$$\left. + \sum_{z_{k+1}} J_{k+1}(I_k, z_{k+1}, u_k) \sum_{x_{k+1}} P(z_{k+1}|x_{k+1}) \sum_{x_k} P(x_{k+1}|x_k, u_k)\ P(x_k|I_k) \right]$$

En donde:

$$\hat{g}_k(x_k, u_k) = \sum_{x_{k+1}} g_k(x_k, u_k, x_{k+1}) P(x_{k+1}|x_k, u_k)$$

Para poder calcular las política utilizando el algoritmo anterior, necesitamos una expresión para $P(x_{k+1}|I_{k+1})$, la cual representa la probabilidad de estar en el estado x_{k+1} considerando el vector de información I_{k+1} (controles aplicados y observaciones realizadas). Esta expresión está dada por:

$$P(x_{k+1}|I_{k+1}) = \frac{\sum_{x_k} P(x_k|I_k) P(x_{k+1}|x_k, u_k) P(z_{k+1}|x_{k+1})}{\sum_{x_{k+1}} \sum_{x_k} P(x_k|I_k) P(x_{k+1}|x_k, u_k) P(z_{k+1}|x_{k+1})} \tag{E.1}$$

donde:

$$I_k = \{z_0, z_1, \ldots, z_k, u_0, u_1, \ldots, u_{k-1}\} = \{I_{k-1}, z_k, u_{k-1}\}$$
$$I_0 = z_0$$

Tomando en cuenta que el horizonte de planificación será $N = 2$, podemos calcular todos los $P(x_k|I_k)$.

Para $P(x_0|I_0)$ tenemos:

$$P(x_0|I_0) = P(x_0|z_0)$$

$$P(x_0|z_0) = \frac{P(z_0|x_0)P(x_0)}{\sum_{x_0} P(z_0|x_0)P(x_0)}$$

Al sustituir por el valor de las variables, y considerando que los dos posibles estados son igual probables, es decir $P(x_k) = 0.5$, tenemos:

$$P(x_0 = A|z_0 = o_1) = \frac{P(z_0 = o_1|x_0 = A)P(x_0 = A)}{\sum_{x_0} P(z_0 = o_1|x_0)P(x_0)}$$

$$P(x_0 = A|z_0 = o_1) = \frac{P(z_0 = o_1|x_0 = A)P(x_0 = A)}{P(z_0 = o_1|x_0 = A)P(x_0 = A) + P(z_0 = o_1|x_0 = B)P(x_0 = B)}$$

$$P(x_0 = A|z_0 = o_1) = \frac{(0.8)(0.5)}{(0.8)(0.5) + (0.6)(0.5)} = 0.57$$

Repitiendo este cálculo para los demás valores tenemos:

$$P(x_0 = B|z_0 = o_1) = 0.43$$
$$P(x_0 = A|z_0 = o_2) = 0.33$$
$$P(x_0 = B|z_0 = o_2) = 0.67$$

Para $P(x_1|I_1)$, tenemos que el vector de información es $I_1 = \{z_0, z_1, u_0\}$.

$$P(x_1|I_1) = \frac{\sum_{x_0} P(x_0|I_0)P(x_1|x_0, u_0)P(z_1|x_1)}{\sum_{x_1}\sum_{x_0} P(x_0|I_0)P(x_1|x_0, u_0)P(z_1|x_1)}$$

Para mostrar como se efectúa el cálculo, desarrollaremos para el primer valor del vector de información y para uno de los estados.

Para $I_1 = \Big\{ z_0 = o_1, z_1 = o_1, u_0 = c_1 \Big\}$

con $x_1 = A$

$$P(x_1 = A|I_1) = \frac{\sum\limits_{x_0} P(x_0|o_1)P(x_1 = A|x_0, u_0 = c_1)P(z_1 = o_1|x_1 = A)}{\sum\limits_{x_1}\sum\limits_{x_0} P(x_0|o_1)P(x_1|x_0, u_0 = c_1)P(z_1 = o_1|x_1)}$$

$$P(x_1 = A|I_1) = \frac{(0.57)(0.7)(0.8) + (0.43)(0.3)(0.8)}{\big[(0.57)(0.7)(0.8) + (0.43)(0.3)(0.8)\big] + \big[(0.57)(0.3)(0.6) + (0.43)(0.7)(0.6)\big]}$$

$$= \frac{0.4224}{0.4224 + 0.2832}$$

$$= \frac{0.4224}{0.7056} = 0.5986$$

con $x_1 = B$

$$P(x_1 = B|I_1) = 0.4013$$

Para $I_1 = \Big\{ z_0 = o_1, z_1 = o_2, u_0 = c_1 \Big\}$

con $x_1 = A$

$$P(x_1 = A|I_1) = 0.3586$$

con $x_1 = B$

$$P(x_1 = B|I_1) = 0.6402$$

Para $I_1 = \Big\{ z_0 = o_1, z_1 = o_1, u_0 = c_2 \Big\}$

con $x_1 = A$

$$P(x_1 = A|I_1) = 0.5156$$

con $x_1 = B$

$$P(x_1 = B|I_1) = 0.4843$$

Para $I_1 = \Big\{ z_0 = o_1, z_1 = o_2, u_0 = c_2 \Big\}$

con $x_1 = A$

$$P(x_1 = A|I_1) = 0.2853$$

con $x_1 = B$

$$P(x_1 = B|I_1) = 0.7146$$

Repetimos pero ahora considerando que la primer observación es o_2, es decir, ahora vamos a recorrer el árbol de la Fig. E.2(b).

Para $I_1 = \left\{ z_0 = o_2, z_1 = o_1, u_0 = c_1 \right\}$

con $x_1 = A$

$$P(x_1 = A|I_1) = 0.5034$$

con $x_1 = B$

$$P(x_1 = B|I_1) = 0.4965$$

Para $I_1 = \left\{ z_0 = o_2, z_1 = o_2, u_0 = c_1 \right\}$

con $x_1 = A$

$$P(x_1 = A|I_1) = 0.2755$$

con $x_1 = B$

$$P(x_1 = B|I_1) = 0.7244$$

Para $I_1 = \left\{ z_0 = o_2, z_1 = o_1, u_0 = c_2 \right\}$

con $x_1 = A$

$$P(x_1 = A|I_1) = 0.6996$$

con $x_1 = B$

$$P(x_1 = B|I_1) = 0.3003$$

Para $I_1 = \left\{ z_0 = o_2, z_1 = o_2, u_0 = c_2 \right\}$

con $x_1 = A$

$$P(x_1 = A|I_1) = 0.5625$$

con $x_1 = B$

$$P(x_1 = B|I_1) = 0.4375$$

Una vez calculados todos las probabilidades de los estados dados los posibles vectores de información, podemos proceder con el algoritmo de programación dinámica.

Como el horizonte de planificación es $N = 2$, entonces tenemos que:

$$J_1(I_1) = \min_{u_1 \in U} \left\{ \sum_{x_1} \hat{g}_1(x_1, u_1)\ P(x_1|I_1) + \sum_{x_1} \sum_{x_2} g_2(x_2) P(x_2|x_1, u_1) P(x_1|I_1) \right\}$$

Para realizar el cálculo de $J_1(I_1)$, primeramente vamos a calcular todos los valores de $\hat{g}_1(x_1, u_1)$,

$$\begin{aligned}
\hat{g}_1(x_1 = A, u_1 = c1) &= \sum_{x_2} g_1(x_1 = A, u_1 = c_1, x_2) P(x_2|x_1 = A, u_1 = c_1) \\
\hat{g}_1(x_1 = A, u_1 = c1) &= g_1(x_1 = A, u_1 = c_1, x_2 = A) P(x_2 = A|x_1 = A, u_1 = c_1) \\
&\quad + g_1(x_1 = A, u_1 = c_1, x_2 = B) P(x_2 = B|x_1 = A, u_1 = c_1) \\
&= (0.5)(0.7) + (0.75)(0.3) \\
&= 0.575 \\
\hat{g}_1(x_1 = B, u_1 = c1) &= 0.46 \\
\hat{g}_1(x_1 = A, u_1 = c2) &= 1.12 \\
\hat{g}_1(x_1 = B, u_1 = c2) &= 1.47
\end{aligned}$$

Al igual que en el cálculo de la probabilidad del estado dado el vector de información, para mostrar como se efectúa el cálculo de $J_1(I_1)$ desarrollaremos para el primer valor del vector de información.

Para $I_1 = \Big\{ z_0 = o_1, z_1 = o_1, u_0 = c_1 \Big\}$

$$\begin{aligned}
J_1(I_1) = \min_{u_1 \in U} \Bigg\{ & \hat{g}_1(x_1 = A, u_1) P(x_1 = A|I_1) + \hat{g}_1(x_1 = B, u_1) P(x_1 = B|I_1) \\
& + \Bigg(\Big(g_2(x_2 = A) P(x_2 = A|x_1 = A, u_1) + g_2(x_2 = B) P(x_2 = B|x_1 = A, u_1) \Big) P(x_1 = A|I_1) \\
& + \Big(g_2(x_2 = A) P(x_2 = A|x_1 = B, u_1) + g_2(x_2 = B) P(x_2 = B|x_1 = B, u_1) \Big) P(x_1 = B|I_1) \Bigg) \Bigg\}
\end{aligned}$$

$$
\begin{aligned}
J_1(I_1) = \min\Big\{ & \big((0.575)(0.5986) + (0.46)(0.4013)\big) \\
& + \Big(\big((1)(0.7) + (2)(0.3)\big)(0.5986) + \big((1)(0.3) + (2)(0.7)\big)(0.4013)\Big), \\
& \qquad \big((1.12)(0.5986) + (1.47)(0.4013)\big) \\
& + \Big(\big((1)(0.1) + (2)(0.9)\big)(0.5986) + \big((1)(0.9) + (2)(0.1)\big)(0.4013)\Big)\Big\}
\end{aligned}
$$

$$
\begin{aligned}
J_1(I_1) &= \min\Big\{\big(0.5287\big) + \big(1.4603\big), \big(1.2603\big) + \big(1.5787\big)\Big\} \\
&= \min\Big\{1.989, 2.839\Big\} \\
&= 1.989 \rightarrow u_1^* = c_1
\end{aligned}
$$

Repitiendo este proceso para los demás valores del vector de información:

Para $I_1 = \Big\{z_0 = o_1, z_1 = o_2, u_0 = c_1\Big\}$

$$
\begin{aligned}
J_1(I_1) &= \min\Big\{2.05, 2.73\Big\} \\
&= 2.05 \rightarrow u_1^* = c_1
\end{aligned}
$$

Para $I_1 = \Big\{z_0 = o_1, z_1 = o_1, u_0 = c_2\Big\}$

$$
\begin{aligned}
J_1(I_1) &= \min\Big\{2.01, 2.8\Big\} \\
&= 2.01 \rightarrow u_1^* = c_1
\end{aligned}
$$

Para $I_1 = \Big\{z_0 = o_1, z_1 = o_2, u_0 = c_2\Big\}$

$$
\begin{aligned}
J_1(I_1) &= \min\Big\{2.07, 2.69\Big\} \\
&= 2.07 \rightarrow u_1^* = c_1
\end{aligned}
$$

Para $I_1 = \Big\{z_0 = o_2, z_1 = o_1, u_0 = c_1\Big\}$

$$
\begin{aligned}
J_1(I_1) &= \min\Big\{2.01, 2.8\Big\} \\
&= 2.02 \rightarrow u_1^* = c_1
\end{aligned}
$$

Para $I_1 = \Big\{ z_0 = o_2, z_1 = o_2, u_0 = c_1 \Big\}$

$$\begin{aligned} J_1(I_1) &= \min\Big\{ 2.08, 2.7 \Big\} \\ &= 2.08 \rightarrow u_1^* = c_1 \end{aligned}$$

Para $I_1 = \Big\{ z_0 = o_2, z_1 = o_1, u_0 = c_2 \Big\}$

$$\begin{aligned} J_1(I_1) &= \min\Big\{ 1.96, 2.89 \Big\} \\ &= 1.96 \rightarrow u_1^* = c_1 \end{aligned}$$

Para $I_1 = \Big\{ z_0 = o_2, z_1 = o_2, u_0 = c_2 \Big\}$

$$\begin{aligned} J_1(I_1) &= \min\Big\{ 2.02, 2.77 \Big\} \\ &= 2.02 \rightarrow u_1^* = c_1 \end{aligned}$$

Con los valores calculados para $J_1(I_1)$, podemos obtener los valores para $J_0(I_0)$, a través de:

$$\begin{aligned} J_0(I_0) = \min_{u_0 \in U} \Bigg[& \sum_{x_0} \hat{g}(x_0, u_0)\ P(x_0|I_0) \\ & + \sum_{z_1} J_1(I_0, z_1, u_0)\ \sum_{x_1} P(z_1|x_1)\ \sum_{x_0} P(x_1|x_0, u_0)\ P(x_0|I_0) \Bigg] \end{aligned}$$

Con $I_0 = o_1$

$$
\begin{aligned}
J_0(I_0) = \min_{u_0 \in U} \Bigg\{ & \hat{g}(x_0 = A, u_0)\ P(x_0 = A|I_0) + \hat{g}(x_0 = B, u_0)\ P(x_0 = B|I_0) \\
& + \Bigg[J_1(I_0, z_1 = o_1, u_0) \Big(P(z_1 = o_1|x_1 = A) \Big(P(x_1 = A|x_0 = A, u_0)\ P(x_0 = A|I_0) \\
& \qquad + P(x_1 = A|x_0 = B, u_0)\ P(x_0 = B|I_0) \Big) + \\
& \qquad P(z_1 = o_1|x_1 = B) \Big(P(x_1 = B|x_0 = A, u_0)\ P(x_0 = A|I_0) \\
& \qquad + P(x_1 = B|x_0 = B, u_0)\ P(x_0 = B|I_0) \Big) \Big) \\
& + J_1(I_0, z_1 = o_2, u_0) \Big(P(z_1 = o_2|x_1 = A) \Big(P(x_1 = A|x_0 = A, u_0)\ P(x_0 = A|I_0) \\
& \qquad + P(x_1 = A|x_0 = B, u_0)\ P(x_0 = B|I_0) \Big) + \\
& \qquad P(z_1 = o_2|x_1 = B) \Big(P(x_1 = B|x_0 = A, u_0)\ P(x_0 = A|I_0) \\
& \qquad + P(x_1 = B|x_0 = B, u_0)\ P(x_0 = B|I_0) \Big) \Big) \Bigg] \Bigg\}
\end{aligned}
$$

$$
\begin{aligned}
J_0(I_0) = \min\Bigg\{ & (0.575)(0.57) + (0.46)(0.43) \\
& + \Bigg[(1.989)\Big((0.8)\Big((0.7)(0.57) + (0.3)(0.43)\Big) + \\
& \qquad (0.6)\Big((0.3)(0.57) + (0.7)(0.43)\Big)\Big) \\
& \quad + (2.05)\Big((0.2)\Big((0.7)(0.57) + (0.3)\ (0.43)\Big) + \\
& \qquad (0.4)\ \Big((0.3)(0.57) + (0.7)(0.43)\Big)\Big)\Bigg], \\
& (1.12)(0.57) + (1.47)(0.43) \\
& + \Bigg[(2.01)\Big((0.8)\Big((0.1)(0.57) + (0.9)(0.43)\Big) + \\
& \qquad (0.6)\Big((0.9)(0.57) + (0.1)(0.43)\Big)\Big) \\
& \quad + (2.06)\Big((0.2)\Big((0.1)(0.57) + (0.9)\ (0.43)\Big) + \\
& \qquad (0.4)\ \Big((0.9)(0.57) + (0.1)(0.43)\Big)\Big)\Bigg]\Bigg\}
\end{aligned}
$$

$$
\begin{aligned}
J_0(I_0) &= \min\Big\{0.5255 + 2.0069, 1.2705 + 2.025\Big\} \\
J_0(I_0) &= \min\Big\{2.5324, 3.2955\Big\} \\
J_0(I_0) &= 2.5324 \rightarrow u_0^* = c_1
\end{aligned}
$$

Con $I_0 = o_2$

$$
\begin{aligned}
J_0(I_0) &= \min\Big\{2.53, 4.20\Big\} \\
J_0(I_0) &= 2.53 \rightarrow u_0^* = c_1
\end{aligned}
$$

Apéndice F

Estimador no Paramétrico del k-ésimo más Próximo Vecino

El cálculo de $P(x|z)$ se realiza utilizando el estimador no paramétrico del k-ésimo más próximo vecino, dado por,

$$P(x|z) = \frac{\zeta}{n_{tp}\phi_{tp}}$$

ϕ_{tp} es la superficie de decisión definida por la siguiente ecuación:

$$\phi_{tp} = \frac{2\, d_{tp}^{n_c}\, \pi^{n_c/2}}{n_c\, \Gamma(n_c/2)}$$

donde n_c es el número de características a utilizar en la clasificación, en nuestro caso $n_c = 1$, dado que solo se utiliza la distancia de Hausdorff como característica de asignación; d_{tp} es la métrica usada, en nuestro caso la distancia de Hausdorff al ζ-ésimo más próximo vecino, ζ es el número de elementos adentro de ϕ_{tp} y n_{tp} el número de elementos por clase usados en la fase de entrenamiento. Γ denota a la función gamma. En general, cuando hay más de una característica, ϕ_{tp} define una hiper-esfera en el espacio de atributos cuya dimensión es el número de atributos. En el caso de un aprendizaje con diferente número de muestras por clase, $\zeta = t_0\sqrt{n_{tp}}$ garantiza la convergencia del estimador. Donde t_0 es el k-ésimo más próximo vecino.

Con estas definiciones podemos reescribir la ecuación para la probabilidad $P(x|z)$ como,

$$P(x|z) = \frac{\zeta}{n_{tp} \cdot \dfrac{2\ d_{tp}^{n_c}\ \pi^{n_c/2}}{n_c\ \Gamma(n_c/2)}}$$

Al sustituir los valores de la función ϕ_{tp} resulta,

$$P(x|z) = \frac{t_0\sqrt{n_{tp}}}{n_{tp} \cdot \dfrac{2\ d_{tp}\ \pi^{1/2}}{\Gamma(1/2)}}$$

$$P(x|z) = \frac{t_0\sqrt{n_{tp}}}{n_{tp} \cdot \dfrac{2\ d_{tp}\ \pi^{1/2}}{\pi^{1/2}}}$$

$$P(x|z) = \frac{t_0}{2\ \sqrt{n_{tp}}\ d_{tp}}$$

La expresión anterior es utilizada en el modelo probabilístico de observación, el cual está dado por la regla de Bayes.

$$p(z_{k+1}|x_{k+1}) = \frac{p(x_{k+1}|z_{k+1})p(z_{k+1})}{\sum\limits_{z_{k+1}} p(x_{k+1}|z_{k+1})p(z_{k+1})}$$

Si consideramos la probabilidad apriori $P(z_{k+1})$ igual para cada clase, y además, sustituimos la expresión de $P(x|z)$, tenemos,

$$p(z_{k+1}|x_{k+1}) = \frac{\dfrac{t_0}{\sqrt{n_{tp}}\ d_{tp}}}{\sum\limits_{tp} \dfrac{t_0}{\sqrt{n_{tp}}\ d_{tp}}}$$

Al simplificar resulta,

$$p(z_{k+1}|x_{k+1}) = \frac{\dfrac{1}{\sqrt{n_{tp}}\ d_{tp}}}{\sum\limits_{tp} \dfrac{1}{\sqrt{n_{tp}}\ d_{tp}}}$$

Bibliografía

[1] Gonzalez-Baños, H., E. Mao, J. Latombe, T. M. Murali, A. Efrat, Planning robot motion strategies for efficient model construction, in: Robotics Research - The 9th International Symposium, 1999, pp. 345–352.

[2] R. Chatila, J. P. Laumond, Position referencing and consistent world modeling for mobile robots, in: IEEE Int. Conf. on Robotics and Automation, ICRA 1985, St. Louis, Missouri, USA, 1985, pp. 135–145.

[3] S. Thrun, Robotic mapping: A survey, in: Exploring Artificial Intelligence in the New Millenium, Morgan Kaufmann, 2002.

[4] S. Thrun, D. Fox, W. Burgard, Probabilistic mapping of an environment by a mobile robot, in: Proc. of IEEE Int. Conf. on Robotics and Automation, ICRA 1998, Leuven, Belgium, 1998, pp. 1546–1551.

[5] F. Amigoni, S. Gasparini, M. Gini, Building segment-based maps without pose information, In Proc. of the IEEE International Conference on Robotics and Automation (ICRA) 94 (7) (2006) 1340–1359.

[6] B. Tovar, L. Munoz-Gomez, R. Murrieta-Cid, M. Alencastre-Miranda, R. Monroy, S. Hutchinson, Planning exploration strategies for simultaneous localization and mapping, Robotics and Autonomous Systems 54 (4) (2006) 314 – 331.

[7] H. Choset, J. Burdick, Sensor based motion planning: The hierarchical generalized voronoi graph, in: Workshop on Algorithmic Foundations of Robotics, 1996.

[8] B. Kuipers, R. Froom, W.-Y. Lee, D. Pierce, The semantic hierarchy in robot learning, in: Robot Learning, Kluwer Academic Publishers, 1993, pp. 141–170.

[9] S. Teller, Automated urban model acquisition: Project rationale and status, in: DARPA98, 1998, pp. 455–462.

[10] B. Tovar, R. Murrieta-Cid, S. M. LaValle, Distance-optimal navigation in an unknown environment without sensing distances, IEEE Transactions on Robotics 23 (3) (2007) 506–518.

[11] J. L. Crowley, World modeling and position estimation for a mobile robot using ultrasonic ranging, in: Proc. Conf. IEEE Int Robotics and Automation, 1989, pp. 674–680.

[12] J. Gonzalez, A. Ollero, A. Reina, Map building for a mobile robot equipped with a 2d laser rangefinder, in: Proc. Conf. IEEE Int Robotics and Automation, 1994, pp. 1904–1909.

[13] N. Ayache, O. Faugeras, Building, registrating, and fusing noisy visual maps, Int. J. Rob. Res. 7 (6) (1988) 45–65.

[14] S. Hutchinson, Exploiting visual constraints in robot motion planning, in: Proc. Conf. IEEE Int Robotics and Automation, 1991, pp. 1722–1727.

[15] R. Murrieta-Cid, C. Parra, M. Devy, Visual navigation in natural environments: From range and color data to a landmark-based model, Journal on Autonomous Robots 13 (2002) 143–168.

[16] A. Bemporad, M. D. Marco, A. Tesi, Wall-following controllers for sonar-based mobile robots, in: Proc. IEEE Conference on Decision and Control, San Diego, USA, 1997, pp. 3063–3068.

[17] R. Sim, G. Dudek, Effective exploration strategies for the construction of visual maps, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS 2003, Las Vegas, Nevada, USA, 2003, pp. 3224–3231.

[18] R. Sim, G. Dudek, Online control policy optimization for minimizing map uncertainty during exploration, in: In Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), 2004, pp. 1758–1763.

[19] B. Yamauchi, Frontier-based exploration using multiple robots, in: Proc. of the Second International Conference on Autonomous Agents, AGENTS '98, ACM, Minneapolis, USA, 1998, pp. 47–53.

[20] R. Sim, N. Roy, Global a-optimal robot exploration in slam, in: Proc. of IEEE Int. Conf. on Robotics and Automation, ICRA 2005, Barcelona, Spain, 2005, pp. 661–666.

[21] H. Feder, J. Leonard, C. Smith, Adaptive mobile robot navigation and mapping., International Journal of Robotics Research 18 (7) (1999) 650–668.

[22] A. Makarenko, B. Williams, F. Bourgault, H. Durrant-Whyte, An experiment in integrated exploration, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS 2002, Lausanne, Switzerland, 2002, pp. 534–539.

[23] H. H. González-Baños, J.-C. Latombe, Navigation strategies for exploring indoor environments, International Journal of Robotics Research 21 (10-11) (2002) 829–848.

[24] G. Oriolo, M. Vendittelli, L. Freda, G. Troso, The srt method: randomized strategies for exploration, in: Proc. of IEEE Int. Conf. on Robotics and Automation, ICRA 2004, New Orleans, LA, USA, 2004, pp. 4688–4694 Vol.5.

[25] F. Amigoni, V. Caglioti, An information-based exploration strategy for environment mapping with mobile robots, Robotics and Autonomous Systems 58 (5) (2010) 684–699.

[26] N. Basilico, F. Amigoni, Exploration strategies based on multi-criteria decision making for searching environments in rescue operations, Autonomous Robots 31 (4) (2011) 401–417.

[27] M. Juliá, A. Gil, O. Reinoso, A comparison of path planning strategies for autonomous exploration and mapping of unknown environments, Autonomous Robots 33 (4) (2012) 427–444.

[28] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, H. Younes, Coordination for multi-robot exploration and mapping, in: Proc. of the AAAI National Conference on Artificial Intelligence, AAAI, Austin, TX, 2000, pp. 852–858.

[29] W. Burgard, M. Moors, C. Stachniss, F. Schneider, Coordinated multi-robot exploration, IEEE Transactions on Robotics 21 (3) (2005) 376–386.

[30] K. Wurm, C. Stachniss, W. Burgard, Coordinated multi-robot exploration using a segmentation of the environment, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS 2008, Nice, France, 2008, pp. 1160–1165.

[31] L. Matignon, L. Jeanpierre, A.-I. Mouaddib, Distributed value functions for multi-robot exploration, in: Proc. of IEEE Int. Conf. on Robotics and Automation, ICRA 2012, Saint Paul, Minnesota, USA, 2012, pp. 1544–1550.

[32] K. Singh, K. Fujimura, Map making by cooperating mobile robots, in: Proc. of IEEE Int. Conf. on Robotics and Automation, ICRA 1993, Atlanta, Georgia, USA, 1993, pp. 254–259.

[33] B. P. Gerkey, M. J. Mataric, A formal analysis and taxonomy of task allocation in multi-robot systems, International Journal of Robotics Research 23 (9) (2004) 939–954.

[34] M. Dias, R. Zlot, N. Kalra, A. Stentz, Market-based multirobot coordination: A survey and analysis, Proc. of the IEEE 94 (7) (2006) 1257–1270.

[35] W. Cohen, Adaptive mapping and navigation by teams of simple robots, Robotics and Autonomous Systems 18 (4) (1996) 411–434.

[36] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Numerical Recipes in C: The Art of Scientific Computing, 2nd Edition, Cambridge University Press, New York, NY, USA, 1992.

[37] D. P. Huttenlocher, G. A. Klanderman, G. A. Kl, W. J. Rucklidge, Comparing images using the hausdorff distance, IEEE Transactions on Pattern Analysis and Machine Intelligence 15 (1993) 850–863.

[38] A. Aho, J. Hopcropft, D. Ullman, Data Structures and Algorithms, Addison-Wesley, 1983.

[39] D. Bertsekas, Dynamic programming and optimal control, Athena Scientific, Belmont, MA, 2000.

[40] T. Başar, G. J. Olsder, Dynamic Noncooperative Game Theory, 2nd Edition, SIAM Series in Classics in Applied Mathematics, 1999.

[41] A. Elfes, Sonar-based real world mapping and navigation, IEEE Transactions on Robotics 3 (3) (1987) 249–264.

[42] L. P. Kaelbling, M. L. Littman, A. R. Cassandra, Planning and acting in partially observable stochastic domains, Artificial Intelligence 101 (1–2) (1998) 99–134.

[43] S. Candido, J. C. Davidson, S. Hutchinson, Exploiting domain knowledge in planning uncertain robot systems modeled as POMDPs., in: Proc. of IEEE Int. Conf. on Robotics and Automation, ICRA 2010, Anchorage, Alaska, USA, 2010, pp. 3596–3603.

[44] S. Thrun, Monte Carlo POMDPs, in: S. Solla, T. Leen, K.-R. Muller (Eds.), Advances in Neural Information Processing Systems 12, MIT Press, 2000, pp. 1064–1070.

[45] T. Smith, R. Simmons, Point-based POMDP algorithms: Improved analysis and implementation, in: Proc. of Uncertainty in Artificial Intelligence, Edinburgh, Scotland, 2005, pp. 542–555.

[46] H. Kurniawati, D. Hsu, W. Lee, Sarsop: Efficient point-based POMDP planning by approximating optimally reachable belief spaces, in: Robotics Science and Systems, 2008.

[47] L. Muñoz Gómez, M. Alencastre-Miranda, R. Lopez-Padilla, R. Murrieta-Cid, Exploration and map-building under uncertainty with multiple heterogeneous robots, in: Proc. of IEEE Int. Conf. on Robotics and Automation, ICRA 2011, Shanghai, China, 2011, pp. 2295–2301.

[48] R. Duda, P. Hart, Pattern classification and scene analysis, John Wiley And Sons Inc, New York, 1973.

[49] R. Lopez-Padilla, R. Murrieta-Cid, S. M. LaValle, Optimal gap navigation for a disc robot, in: E. Frazzoli, et. al. (Eds.), Algorithmic Foundations of Robotics X, Vol. 86 of Springer Tracts in Advanced Robotics, Springer Berlin Heidelberg, 2013, pp. 123–138.

[50] J. Snape, J. van den Berg, S. Guy, D. Manocha, Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles, in: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS 2009, St. Louis, MO, USA, 2009, pp. 5917–5922.

[51] S. M. LaValle, Planning algorithms, Cambridge University Press, Cambridge, U.K., 2006.

[52] S. Thrun, W. Burgard, D. Fox, Probabilistic robotics, MIT Press, Cambridge, MA, 2005.

[53] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, Ros: an open-source robot operating system, in: ICRA Workshop on Open Source Software, 2009.