Universidad de Buenos Aires, Facultad de Ingeniería

Práctica 5 - ICP y Planeamiento

Robótica Móvil - Un enfoque probabilístico

Prof. Dr. Ignacio Mas

10 de junio de 2022

Fecha límite de entrega: 23/06/22, 23.59hs.
Modo de entrega: Enviar por el Aula Virtual del Campus **en un solo archivo comprimido** el código (.m) comentado y los gráficos (.jpg ó .pdf) y/o animaciones.

1. Implementación de algoritmo de asociación de datos

En este ejercicio se implementará una técnica de asociación de datos. Los archivos incluidos en esta práctica implementan los pasos básicos del algoritmo ICP (Iterative Closest Point). En este algoritmo, editando la línea *42* del script `test_icp.m` se pueden probar 4 conjuntos de datos (P1, P2, P3 y P4). El algoritmo funciona correctamente con los datos con correspondencias conocidas (i.e. P1 y P2), pero no funciona para conjuntos de datos con correspondencias desconocidas (i.e. P3 y P4). Si las correspondencias no son conocidas, deben ser estimadas antes de aplicar el algoritmo ICP.
Implementar el método de asociación de datos que considera correspondencias por puntos más cercanos en el archivo `closest_point.m` y verificarlo usando los conjuntos de datos P3 y P4.

2. Planeamiento de caminos

Los algoritmos de búsqueda en grafos como Dijkstra o A* pueden ser usados para planear caminos en grafos desde un lugar de inicio hasta un objetivo. Si las celdas de un mapa de grilla se representan como nodos conectados con sus celdas vecinas, estos algoritmos pueden aplicarse directamente para realizar planeamiento para robots. Para este ejercicio, consideramos las 8 celdas vecinas de una celda $\langle x, y \rangle$, que se definen como las celdas adyacentes a $\langle x, y \rangle$ horizontalmente, verticalmente y en diagonal.

El archivo incluido contiene una implementación de planeamiento en 2-D basado en grafos. El script `planning_framework.m` contiene la parte principal del algoritmo y es el que debe ejecutarse. Este archivo no necesita ser modificado, pero es aconsejable entender lo que hace. Los ejercicios de esta sección se realizan implementando las funciones vacías que acompañan al script principal.

2.1. Algoritmo de Dijkstra

El algoritmo de Dijkstra se usa para calcular caminos de costo mínimo dentro de un grafo. Durante la búsqueda, siempre se busca el nodo del grafo con el menor costo desde el punto de inicio y se agregan los nodos vecinos al grafo de búsqueda.

1. Sea $M(x, y)$ un mapa de grilla de ocupación. Durante la búsqueda, las celdas se conectan con sus celdas vecinas para construir el grafo de búsqueda. Completar la función `neighbors` provista que define los vecinos de una celda. La función toma como entrada las coordenadas de una celda y el tamaño del mapa, y devuelve un vector de $n \times 2$ con las coordenadas de sus celdas vecinas, teniendo en cuenta los límites del mapa.

2. Implementar una función para los costos de un arco entre nodos que permita planear caminos de mínima longitud y libre de colisiones. Considerar la celda como un obstáculo si su probabilidad de ocupación supera cierto umbral. ¿Qué umbral se debería elegir? Implementar la función `edge_cost`.

3. Incluir información de ocupación en la función de costo que permita que el algoritmo elija celdas con baja probabilidad de ocupación sobre celdas con mayor probabilidad de ocupación.

2.2. Algoritmo $A^\star$

El algoritmo $A^\star$ utiliza una heurística para realizar una búsqueda informada que resulta ser más eficiente que el algoritmo de Dijkstra.

1. ¿Qué propiedades debe tener dicha heurística para asegurar que $A^\star$ es óptimo?

2. Definir una heurística para planeamiento de robots móviles en 2-D. Completar la función `heuristic` provista. La función toma como entrada las coordenadas de una celda y del objetivo, y devuelve el costo estimado hasta el objetivo.

3. ¿ Qué pasa si se aumenta la heurística usando h_2, siendo h_2 un múltiplo de la heurística h definida en el punto anterior. Analizar el comportamiento con diferentes factores: $h_2 = \{1; 2; 5; 10\} * h$