Examen Parcial

Construcción de Robots Móviles, FI, UNAM, 2020-2

Nombre: Patricio Jaime Porras

Instrucciones: Subir la solución en un solo archivo PDF con el nombre Examen.pdf, al repositorio en GitHub en la carpeta Entregables.

Fecha límite de entrega: 04 de junio de 2020.

1. Explique qué es la configuración, espacio de configuraciones y grados de libertad de un robot móvil.

    a La configuración de un robot está dada por las partes y capacidades que tiene el robot móvil, estas incluyen, por ejemplo:

        i Cuerpo del robot: Parte donde se alojarán componentes principales para interconectar distintos dispositivos

        ii Extremidades: Cualquier anexo al cuerpo del robot como patas, ruedas, brazos entre otros.

        iii Sensores: Dispositivos que dotan al robot de información como cámaras RGB(D), sensores de presión, giroscopios, sensores de distancia, micrófonos entre otros.

        iv Motores: Dispositivos que convierten distintos tipos de energía en energía mecánica. Estos permiten la movilidad del mismo robot a través de las extremidades del mismo.

        v Restricciones: Estas son las cosas que el robot móvil NO puede hacer. Una restricción nos permite conocer los límites de nuestro robot para evitar problemas futuros.

    b El espacio de configuración hace referencia al entorno en el que está o estará sumergido el robot móvil. De esta manera podemos estudiar este espacio para encontrar soluciones a distintos problemas derivados de la interacción del robot con este ambiente, desde su planeación de movimiento hasta las posibles restricciones que le impidan completar una tarea.

    c Se le conoce grado de libertad a cada una de las coordenadas independientes para describir el estado mecánico del robot. Entre mas posibilidades de rotación y/o traslación, mayor grado de libertad tendrá el supuesto robot. Este concepto se ejemplifica en la figura 1 y 2[1].

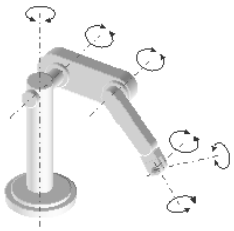

Figura 1. Robot con 6 grados de libertad

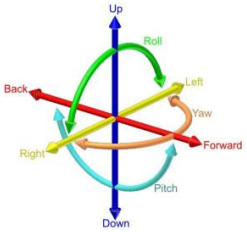

Figura 2. Los 6 grados de libertad sobre un punto.

2. Investigue dos métodos basados en grafos para planeación de rutas.

   a Algoritmo de Dijkstra: Este algoritmo basado en grafos encuentra el camino más corto de nodo *A* a nodo *B* en un grafo con un numero de estados discreto. El pseudocódigo del algoritmo es el siguiente:

```
 1  function Dijkstra(Graph, source):
 2
 3      create vertex set Q
 4
 5      for each vertex v in Graph:
 6          dist[v] ← INFINITY
 7          prev[v] ← UNDEFINED
 8          add v to Q
10      dist[source] ← 0
11
12      while Q is not empty:
13          u ← vertex in Q with min dist[u]
14
15          remove u from Q
```

Obtenido de: https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#Pseudocode

   b. Algoritmo A*: Este algoritmo basado en grafos es una extensión del algoritmo de Dijkstra. Este algoritmo hace lo mismo que Dijkstra pero añade una función de costo a la transición de estados. Este algoritmo tiene un costo extra en términos de memoria y velocidad en muchas áreas de implementación. Existen algoritmos mas eficientes para distintas aplicaciones como D*, Campo D*, Theta*, entre otros. El pseudocódigo es el siguiente (sin anotaciones):

```
1  function reconstruct_path(cameFrom, current)
2      total_path := {current}
3      while current in cameFrom.Keys:
4          current := cameFrom[current]
5          total_path.prepend(current)
6      return total_path
7
8  function A_Star(start, goal, h)
9      cameFrom := an empty map
10     gScore := map with default value of Infinity
11     gScore[start] := 0
12     fScore := map with default value of Infinity
13     fScore[start] := h(start)
14
15     while openSet is not empty
16         current := the node in openSet having the lowest fScore[] value
17         if current = goal
18             return reconstruct_path(cameFrom, current)
19
20         openSet.Remove(current)
21         for each neighbor of current
22 neighbor through current
```

```
            tentative_gScore := gScore[current] + d(current, neighbor)
            if tentative_gScore < gScore[neighbor]
                cameFrom[neighbor] := current
                gScore[neighbor] := tentative_gScore
                fScore[neighbor] := gScore[neighbor] + h(neighbor)
                if neighbor not in openSet
                    openSet.add(neighbor)
    return failure
```
Obtenido de: https://en.wikipedia.org/wiki/A* search algorithm#Pseudocode

3. Investigue dos métodos basados en muestreo para planeación de rutas.
    a RRT: Es un algoritmo basado en muestreo que tiene una naturaleza "subóptima" al sólo encontrar un camino POSIBLE y no el mejor, ya que el algoritmo se detiene al encontrar una solución. La probabilidad de encontrar una solución óptima tiende a 1 a medida que el árbol crece. Este algoritmo razona sobre la geometría de su espacio devolviendo una solución (no necesariamente óptima) para llegar de punto A al B. Existen variaciones de este algoritmo para manejar mínimos locales de forma que se encuentren mejores soluciones. Estos son RRT-Connect y RRT multi-query. [2]
    b RRT*: Se basan en la lógica de RRT, pero su naturaleza es asintóticamente óptima, esto quiere decir que devuelven una solución óptima cuando el tiempo de búsqueda tiende al infinito. La diferencia entre éste y RRT es que la solución optima tiende a infinito cuando el árbol se expande. Para poder disminuir el tiempo de convergencia, existen alteraciones al algoritmo como RT-RRT* y RRT*FN. [2]
4. Explique en qué consiste el proceso de SLAM (Simultaneous Localization and Mapping).
    a El problema SLAM pregunta sobre si es posible que un vehículo autónomo cree un mapa para encontrarse en un ambiente sin tener información previa del lugar ni de su localización inicial. El proceso para solucionar este problema sería que el robot obtenga información del ambiente para disminuir la incertidumbre sobre la localización del robot. Según Gamini Dissanayake, Paul Newman, Steven Clark, Hugh F. Durrant-Whyte y M. Scorba, este problema se soluciona utilizando filtros de Kalman, minimizando la incertidumbre y obteniendo correlaciones entre puntos del ambiente en cuestión al obtener observaciones de estos y entre los trayectos de punta A a punto B.
5. Explique en qué consiste la localización mediante filtros de partículas, sus ventajas sobre el Filtro de Kalman y los paquetes de ROS que lo implementan.
    a La localización mediante filtros de partículas es un método para aproximar la ubicación de un objeto en movimiento, en nuestro caso, un robot móvil. Este filtro tiene las siguientes fases:[3]
        i Generar partículas distribuidas normalmente sobre el mapa del ambiente.
        ii Calcular parámetros obtenidos a través de sensores del robot y hacer lo mismo para cada una de nuestras partículas. (por ejemplo, la distancia entre la partícula y un obstáculo enfrente de la partícula)

iii Asignar pesos a las partículas según su similitud con las lecturas del robot en cuestión. Por ejemplo, si el robot obtiene una distancia a un objeto igual a 5 metros y una partícula obtiene una distancia de 5.2 metros, esta partícula tiene una ALTA probabilidad de estar en la misma posición del robot. Si la lectura de la distancia se aleja a la leída por el robot (5m), la probabilidad que su posición sea la del robot es BAJA. Entre mas alta la probabilidad, mayor será su peso.

iv Se eliminan partículas según la probabilidad asignada, por lo tanto aquellas con una probabilidad baja serán eliminadas, conservando la mayoría de partículas con probabilidad alta. Las partículas que quedaron "vivas" se les asigna un peso 0.

v El robot en cuestión se desplaza hacia un nuevo punto y las nuevas partículas obtenidas del punto anterior igual hacen un desplazamiento suponiendo que son el robot. De esta manera obtenemos nuevos puntos y se vuelve a realizar el algoritmo desde el punto iii. Si las partículas están acumuladas en puntos MUY cercanos, se puede decir que nos hemos localizado en el mapa.

b Los filtros de Kalman necesitan información a priori sobre la localización del robot (aunque sea una mala aproximación). Encontré una aplicación para los filtros de Kalman que explica como funciona. Esta aplicación es sobre robots jugando fútbol, donde el campo esta limitado por cuatro puntos (las esquinas del campo) en donde hay cámaras que pueden determinar la posición del robot de manera no exacta, pero aproximada. Esta información sumada a la lectura de los sensores del robot, utilizando el filtro de Kalman, mejoran la información y reducen el error de localización del robot. Ahora, un filtro de partículas no requiere esta información de las cámaras, ya que la localización inicial es completamente desconocida.

c Paquetes de ROS que implementan los filtros de partículas:

i bfl (the Bayesian Filtering Library)

ii mcl_pi

iii amcl

6. Investigue qué son los campos potenciales y explique los pasos generales para implementarlos.

a Los campos potenciales son modelos que permiten lograr una trayectoria de punto A a punto B a través de la implementación de campos atractivos y repulsivos según sea un objetivo o un obstáculo respectivamente. Utilizando un gradiente descendiente el robot puede empezar a moverse en sentido contrario del gradiente. De esta manera, el robot se alejará de obstáculos, ya que estos se encuentran en un nivel más alto y el robot busca un nivel siempre más bajo. Al contrario de los obstáculos (repulsivos), la meta o metas están representados como los puntos mas bajos a donde, por naturaleza, el contrario del gradiente siempre va a estar apuntando. Esto se puede representar gráficamente como se muestra en la figura 3.

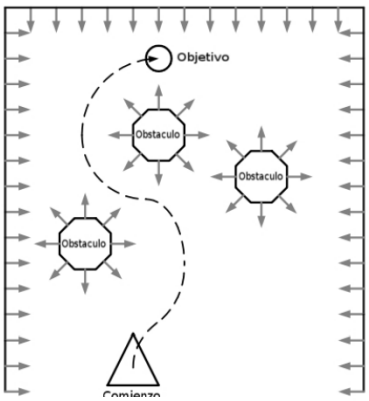

Figura 3. Camino del opuesto al gradiente.

b Para implementar un campo potencial en un plano, se deben sumar distintas funciones para cada objetivo y/u obstáculo. Todos los objetivos deben de tener una pendiente negativa y los obstáculos una positiva o asignar valores por celdas. Siempre se debe tener en cuenta que entre más obstáculos la probabilidad de caer en mínimos locales aumenta y se debe buscar funciones que disminuyan este problema. La figura 4 representa las celdas que rodean a un objetivo mientras que la figura 5 representa las celdas que rodean a un obstáculo. El robot siempre debe buscar la celda próxima más baja o una posición donde su potencial sea más bajo.[4]

3	3	3	3	3	3	3
3	2	2	2	2	2	3
3	2	1	1	1	2	3
3	2	1	0	1	2	3
3	2	1	1	1	2	3
3	2	2	2	2	2	3
3	3	3	3	3	3	3

Figura 4. Campo alrededor de un objetivo.

49	49	49	49	49
49	50	50	50	49
49	50	200	50	49
49	50	50	50	49
49	49	49	49	49

Figura 5. Campo alrededor de un obstáculo.

7. Explique qué es una transformación homogénea y para qué se utiliza en robots móviles.

   a Una matriz de transformación homogénea T es una matriz de dimensión 4x4. Representan la transformación de un vector de coordenadas homogéneas de un sistema a otro. Esta compuesta por 4 términos: Escalamiento, traslación, rotación y perspectiva.[5]

   b Estas transformaciones nos son útiles para:

      i Representar la posición y orientación de un sistema girado y trasladado con respecto a un sistema fijo. (la posición y dirección del robot en un ambiente) [5]

      ii Transformar un vector expresado en coordenadas movibles y su representación en un sistema fijo. (alterar la posición y dirección del robot en un ambiente) [5]

      iii Rotar y trasladar un vector con respecto a un sistema fijo. (alterar la posición y dirección del robot en un ambiente) [5]

8. Investigue qué es un robot con restricciones no holonómicas de movimiento.

   a Los robots holonómicos se definen como aquellos cuyo número de grados de libertad controlables es igual o superior al número total de grados de libertad, siempre que no se encuentren sometidos a las restricciones holonómicas. Una restricción holonómica es aquella que impide modificar la dirección del robot instantáneamente, como se puede suponer de un automóvil común, donde se mueve solo en un eje y no en ambos. Por lo tanto, un robot con restricciones no holonómicas de movimiento es aquel que sólo puede moverse instantáneamente en menos del total de grados de libertad (no podrá moverse instantáneamente en X, Y ni girar en Z sin la necesidad de utilizar otro eje para realizar la transformación). [6]

Se recomienda consultar las siguientes referencias:

- González, D., Pérez, J., Milanés, V., & Nashashibi, F. (2015). A review of motion planning techniques for automated vehicles. IEEE Transactions on Intelligent Transportation Systems, 17(4), 1135-1145.
- Paden, B., Cáp, M., Yong, S. Z., Yershov, D., & Frazzoli, E. (2016). A survey of motion planningˇ and control techniques for self-driving urban vehicles. IEEE Transactions on intelligent vehicles, 1(1), 33-55.
- Dissanayake, M. G., Newman, P., Clark, S., Durrant-Whyte, H. F., & Csorba, M. (2001). A solution to the simultaneous localization and map building (SLAM) problem. IEEE Transactions on robotics and automation, 17(3), 229-241.
- Choset, H. M., Hutchinson, S., Lynch, K. M., Kantor, G., Burgard, W., Kavraki, L. E., & Thrun, S. (2005). Principles of robot motion: theory, algorithms, and implementation. MIT press.

Referencias adicionales:

[1] Inteligencia Artificial (2020). Grados de Libertad, *Grados de Libertad*, recurso web obtenido de <https://freedoomforlife.wordpress.com/grados-de-libertad/> (recuperado 2 de junio de 2020)

[2] Montes Romero, Ángel Manuel (2017). Planificación de caminos basada en modelo combinando algoritmos de búsqueda en grafo, derivados de RRT y RRT*, *Métodos probabilísticos o basados en muestreo. (1.2.2.3 pág 5).* Documento web obtenido de <http://bibing.us.es/proyectos/abreproy/71064/fichero/1064-MONTES.pdf> (recuperado 2 de junio de 2020)

[3] Zhang, Jeremy (2019). Particle Filter on Localisation *A brief summarise of applying particle filter on self-driving car.* Recurso web obtenido de <https://towardsdatascience.com/particle-filter-on-localisation-9e0802282aaf>(recuperado 3 de junio de 2020)

[4] Barrero Pérez Jaime G., Martínez Ángel Roberto, Tibaduiza Burgos Diego A. (2011). Planeamiento de caminos y trayectorias mediante algoritmos genéticos y campos de potencial para un robot móvil*. B. Campos de potencial.* Recurso web obtenido de https://www.researchgate.net/publication/258630262_Planeamiento_de_caminos_y_trayectorias_mediante_algoritmos_geneticos_y_campos_de_potencial_para_un_robot_movil (recuperado 3 de junio de 2020)

[5] F. Hugo Ramírez Leyva (2012). Modelado Cinemática de Robots. Curso de Robótica, Instituto de Electrónica y Mecatrónica. recuperado 3 de junio de 2020)

[6] Cordero Conde Iván (2018). Sistema de Navegación para Robot Holonómico, capítulo 1, Introducción, página 2. Documento web obtenido de <http://oa.upm.es/50487/1/TFG_IVAN_CORDERO_CONDE.pdf> (recuperado 3 de junio de 2020)