Universidad del Valle de Guatemala
Departamento de Ingeniería Mecatrónica
MT2002 - Robótica 1
MSc. Miguel Zea

Laboratorio 9

Planificación de Movimiento y Robótica Móvil

Objetivos

- Implementar un algoritmo de control de punto a punto y de seguimiento de trayectorias para un robot móvil Pioneer P3-DX.
- Implementar algoritmos de planificación de movimiento basados en búsqueda D* y Probabilistic Roadmaps empleando la *Robotics Toolbox*.
- Efectuar una aplicación sencilla de solucionar un laberinto dentro del entorno de simulación V-REP controlado desde MATLAB.

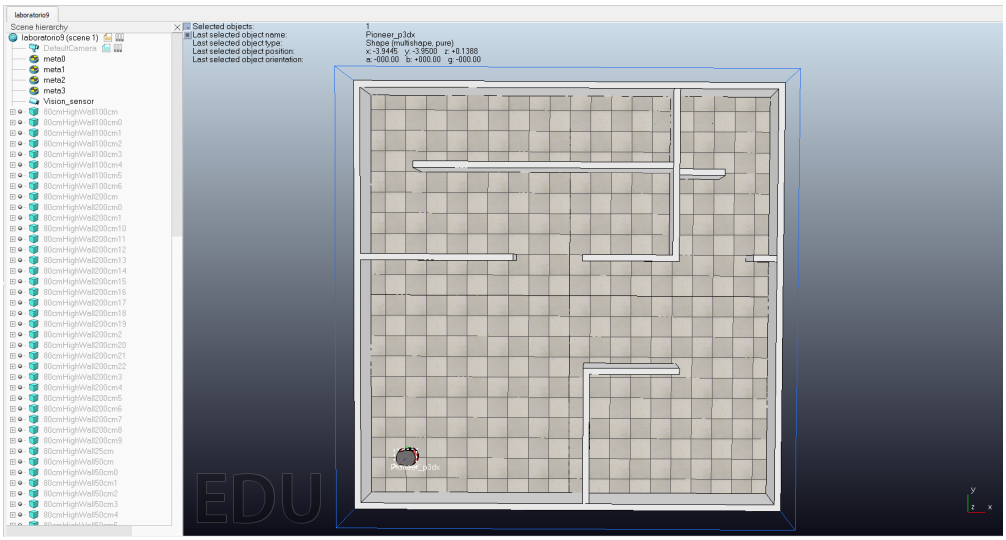
Procedimiento

En la práctica de esta semana usted deberá efectuar tres tareas específicas: lograr mover a un robot móvil de punto a punto, generar una ruta válida para el mismo entre un punto inicial y una meta dentro de un mapa y finalmente visualizar la combinación de ambas tareas dentro de V-REP. Primero, descargue de Canvas el archivo `mt402lab9.zip` y extraiga sus contenidos dentro de una carpeta en una ubicación de su preferencia. Cambie el folder actual de MATLAB para que coincida con esta carpeta. Luego, descargue el documento auxiliar `Navigation-PC.pdf` y siga las instrucciones de cada una de las secciones que se le presentan a continuación.

Control del robot Pioneer P3-DX

Abra la escena `laboratorio9` en V-REP y observe que se le presenta un robot móvil en la esquina superior derecha de un piso de 10x10m en donde cada cuarto se encuentra delimitado por paredes de 80cm de alto, como se muestra en la siguiente figura. En el techo de este cuarto se encuentra instalada una cámara que actuará como sensor de visión para poder obtener el mapa de obstáculos al momento de la planificación de movimiento. Finalmente, se encuentran *dummies* dispersos en todo el piso que servirán como indicadores de puntos de interés a los cuales debe llegar el robot móvil. Abra el script `simulacion` en MATLAB y siga los siguientes pasos:

1. Identifique el nombre de los handles para el chasis del robot, los motores del robot y los dummies (metas).
2. Emplee los handles y la función `vrep.simxGetObjectPosition` para obtener la posición xy de las metas y la posición inicial del robot. Recuerde que puede encontrar la sintaxis de la función en la documentación del API externa de MATLAB para V-REP. Para saber en dónde modificar el código, guíese por la documentación incluida dentro del mismo (las secciones que indican *MODIFICAR AQUI*).



3. Dentro del ciclo de actualización de la simulación, encuentre la posición (x, y) y la orientación θ actual del robot empleando el handle de su chasis y las funciones `vrep.simxGetObjectPosition` y `vrep.simxGetObjectOrientation` (recuerde que debe tomar sólo la orientación alrededor del eje z del marco inercial en este caso) respectivamente. Adicionalmente, haga que el punto (x_g, y_g) deseado para el algoritmo de control sea igual a las coordenadas del dummy `meta0` dentro de la escena en V-REP (se escoge esta meta ya que no existe ningún obstáculo entre el robot y ella por lo cual puede probarse el algoritmo de punto a punto sin problemas).

4. Implemente, también dentro del ciclo de actualización, el algoritmo de control punto a punto para un robot diferencial

$$\dot{\phi}_r = \frac{v + \omega\ell}{r}, \qquad \dot{\phi}_\ell = \frac{v - \omega\ell}{r},$$

en donde

$$v = K_p e_p, \qquad \omega = K_o e_o,$$

$$e_p = \left\| \begin{bmatrix} x_g \\ y_g \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix} \right\|, \qquad e_o = \arctan 2\left(\frac{\sin(\theta_g - \theta)}{\cos(\theta_g - \theta)}\right), \qquad \theta_g = \arctan 2\left(\frac{y_g - y}{x_g - x}\right).$$

Recuerde que r y ℓ corresponden al radio de las ruedas del robot y a la distancia del centro del robot a cada rueda respectivamente (encuentre estos parámetros en la hoja de especificaciones del robot Pioneer P3-DX adjunta a la guía del laboratorio en Canvas). Asegúrese de ajustar las constantes $K_p > 0$ y $K_o > 0$ hasta obtener un comportamiento adecuado. Si observa que el robot, luego de llegar a la posición deseada, comienza a oscilar alrededor de la misma, coloque una condición en el error de posición e_p que haga que las velocidades sean cero cuando el robot se encuentre lo suficientemente cerca de la meta.

5. Utilice el Video recorder de V-REP para producir un video de su simulación en donde el robot sea exitoso en la tarea e inclúyalo en el espacio asignado en Canvas. Tome en consideración que puede modificar el tiempo final de simulación mediante el parámetro `tf` en caso que su simulación esté tardando mucho tiempo en finalizar luego de que el robot haya llegado a la meta.

Planificación de movimiento empleando la Robotics Toolbox

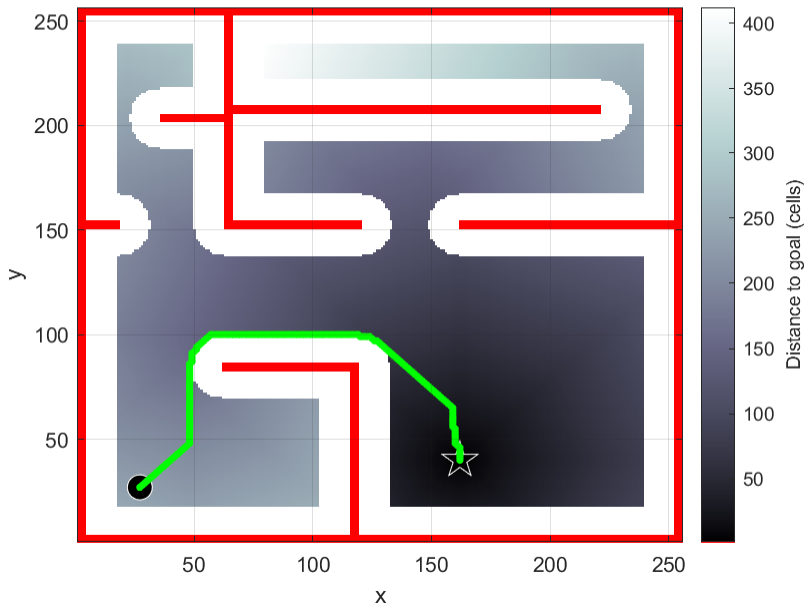
Ya con el control de punto a punto funcionando adecuadamente para el robot, prosiga a determinar rutas válidas entre el robot y las metas restantes (las cuales no pueden alcanzarse sólo con el control punto a punto ya que presentan obstáculos) mediante planificación de movimiento. Para ello, ejecute los siguientes pasos:

1. Dentro del script `simulacion`, después de encontrar la posición de las metas pero antes del ciclo de actualización, emplee la función `vrep.simxGetVisionSensorImage2` para obtener una imagen I del mapa del cuarto en la escena de V-REP mediante el sensor de visión. Asegúrese de establecer la opción correspondiente para que la imagen obtenida esté en escala de grises.

2. Emplee *thresholding* para obtener una *occupancy grid* a partir de la imagen del mapa. Una occupancy grid es un array en donde cada casilla puede tomar sólo 2 valores, 1 si la casilla contiene un obstáculo y 0 en caso contrario, la cual se empleará como base para los algoritmos de planificación de movimiento. Para efectuar el thresholding emplee la línea de código `map = img > thresh;` y modifique el valor de `thresh` (entre 0 y 255) hasta que su occupancy grid contenga sólo a las paredes como obstáculo (es decir, que el mapa presente color blanco sólo en las paredes y negro en caso contrario). Puede emplear el comando de MATLAB `imshow(map)` para visualizar su occupancy grid en forma de imagen. Nótese que, a pesar que la imagen sí representa al mapa de la escena, la orientación del mismo no es la correcta (dada la pose de la cámara con respecto al marco inercial) y esto puede ocasionar confusión al momento de calcular los recorridos para el robot dentro del mapa. Para corregir la orientación puede emplear el comando `map = flip(flip(map,1),2)`. Finalmente, cambie el tipo de dato a `double` para su occupancy grid ya que la Robotics Toolbox lo requiere en los algoritmos que se emplearán en pasos posteriores.

3. Emplee el documento de referencia `Navigation-PC.pdf` para implementar un algoritmo de búsqueda D* que determine una ruta válida entre la posición inicial del robot y la `meta1`. Para hacer esto tome en consideración lo siguiente:

   - Al momento de emplear los métodos constructores (`Dstar` y `PRM`) puede colocarles como opción (para el caso D* por ejemplo) `Dstar(map, 'inflate', radius_in_cells);` lo cual efectúa una inflación/aumentación del mapa asumiendo que el robot cabe dentro de un círculo con radio dado en número de celdas. Puede determinar este radio empleando las dimensiones del robot descritas en su hoja de especificaciones y sabiendo que las dimensiones del piso en V-REP (10x10m) corresponden a las dimensiones del mapa (256x256 celdas).
   - Es necesario que efectúe una conversión de metros a celdas para las posiciones de las metas y la posición inicial del robot. Tome en consideración que el origen del mapa dentro de V-REP se encuentra exactamente en el centro de la escena, lo cual quiere decir que toda posición se encuentra en el intervalo $[-5, 5]$. Es necesario entonces hacer una conversión de distancia en este intervalo al intervalo $[1, 256]$.
   - Al momento de hacer una consulta al algoritmo de planificación, puede emplear (para el caso D* por ejemplo) `path = ds.query(goal, 'animate');` para visualizar la animación del recorrido encontrado al igual que guardar los puntos del mismo dentro del array `path`. Guarde la figura generada, que debería verse similar a la figura de la página siguiente, ya que deberá presentarla en el documento a subir en el espacio correspondiente en Canvas.



4. Repita el paso anterior para las `meta2` y `meta3` y luego repita para todas las metas pero empleando Probabilistic Roadmaps (PRMs). Guarde las figuras generadas para cada uno de los recorridos.

Resolviendo el laberinto en V-REP

Ya con los recorridos encontrados, prosiga a hacer que el robot móvil en V-REP recorra estas rutas mediante control punto a punto, en donde el punto a donde se desea llegar debe desplazarse según los puntos del recorrido. Para lograr esto siga los siguientes pasos:

1. En el script de `simulacion`, modifique dentro del ciclo de actualización la asignación del punto deseado (x_g, y_g) para que tome puntos del array del recorrido encontrado con los algoritmos de planificación de movimiento. Modifique su código tal que no se tome un nuevo punto del recorrido mientras que no se haya llegado lo suficientemente cerca del punto actual. Esto puede codificarse como una condición para el error de posición e_p. Tome en consideración que el array del recorrido encontrado está en coordenadas en celdas, por lo cual es necesario regresarlo a metros antes que el robot trate de ejecutar su control de punto a punto.

2. Para cada uno de los 6 recorridos (3 con D* y 3 con PRMs) determinados en la sección anterior, haga que el robot siga la ruta dentro de V-REP y emplee el Video recorder para producir un video de cada caso. En caso de ser necesario, ajuste los valores de las constantes K_p y K_o al igual que el tiempo final de simulación `tf` hasta obtener los resultados deseados.

Cuando esté seguro de haber terminado con todo lo solicitado, redacte un pequeño documento pdf que incluya las figuras obtenidas en la planificación de movimiento y una breve discusión de cuáles fueron las dificultades que encontró durante toda la práctica. Suba este documento junto a los videos generados en el espacio correspondiente en Canvas.