Assignment 4: Path Planning

Profesor: Martin Adams
Auxiliar: Ignacio Romero
Ayudantes: Sebastián Herrera - Mattias Prieto

En la última experiencia del componente de robótica del curso, se implementará un algoritmo de path planning que controle el robot con el objetivo de moverlo a un destino, sin incurrir en colisiones con su entorno.

El algoritmo que se debe utilizar es el de Campos Potenciales, el cual reduce el entorno del robot a una sumatoria de fuerzas, donde la atracción al objetivo es de naturaleza elástica, mientras que la respuesta a obstáculos es un potencial repulsivo análogo al de cargas de igual signo. Su explicación puede ser vista en más detalle en el manual del curso.

1. Tareas

En este laboratorio, se debe implementar el algoritmo de Campos Potenciales, de forma análoga al caso del Assignment 2. Es decir, dado un punto de partida A y un punto de llegada B publicado via terminal en el tópico /`target_pose`, la implementación a realizar debe llevar a cabo los cálculos necesarios para luego generar una orden de movimiento del robot.

Para ello, realizar lo siguiente:

1. Tomar de base la implementación del assignment 2. Dada la ubicación actual del robot, calcular fuerza de atracción al objetivo X_{att} y Y_{att}. Adicionalmente, con el mensaje de LaserScan generado por el robot, calcular las fuerzas de repulsión X_{rep} y Y_{rep}.

2. Utilizar las fuerzas calculadas en el paso anterior para obtener la dirección y magnitud de la fuerza resultante sobre el robot. Comande al robot un movimiento con magnitud proporcional a P en la dirección de la fuerza resultante por un periodo corto de tiempo.

3. Repetir esta lógica de movimiento hasta que el robot pueda converger a la posición deseada y P tienda a 0 (bajo algún umbral pequeño).

4. Ajustar los valores de los parámetros $K_{att}, K_{rep}, R_{max}$ e identificar y explicar sus efectos en el comportamiento del algoritmo.

2. Formato de entrega

El tercer reporte del componente de robótica incluye el assignment 4. Si bien los laboratorios pueden realizarse en grupos de 2-3 personas, los reportes son entregas **individuales**. Este debe incluir:

- **Portada**: Incluir nombre, fecha, cuerpo docente y código de curso.
- **Marco Teórico:** Descripción de los principales sistemas y algoritmos utilizados. Como mínimo se debe describir: Campos potenciales, tipos de sensores y mensajes utilizados y teoría revisada en clases.
- **Tarea(s) a Resolver:** Describir brevemente las tareas a resolver.
- **Metodología:** Explicar cómo resolvieron el problema. No basta con pegar códigos.
- **Resultados:** Presentar imágenes, tablas, gráficos u otro tipo de resultado solicitado.
- **Análisis de Resultados:** Analizar los resultados presentados, responder preguntas del enunciado en base a estos.
- **Conclusiones:** Referirse a los principales aprendizajes y alcances de la experiencia.

El reporte debe ser en formato pdf y se deben adjuntar códigos implementados. **Muy importante** responder las preguntas del enunciado, ya que sus respuestas son parte de la nota. Los reportes deben ser breves pero concisos.

3. Cheat Sheet de ROS

Comandos de terminal

Situarse en el directorio del workspace:

```
$ cd el5206_ws
```

Lanzar la simulación en el mundo *house*:

```
$ roslaunch el5206_gazebo turtlebot3_house.launch
```

Lanzar la simulación en el mundo vacío:

```
$ roslaunch el5206_gazebo el5206_empty_world.launch
```

Nota: La primera vez que se ejecutan un comando `roslaunch` de Gazebo, la simulación puede tardar un poco en comenzar.

Lanzar nodo de teleoperación (Modo Mario Kart):

```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

Para operar al robot se debe estar sobre la terminal que corre el nodo de teleoperación, usando las teclas w, a, s, d y x.

Correr el nodo del robot, abriendo una nueva terminal, con la simulación corriendo:

```
$ rosrun el5206_example el5206_main.py
```

Programación del Nodo EL5206_robot

En los assignment, se deberá modificar el código **el5206_main.py** (en *src/el5260_example/scripts/*), que inicializa un nodo de ROS que escucha la odometría del robot, su pose real, las órdenes de movimiento, el láser, etc.

Se proporciona método de la clase llamado dance, que muestra un demo de movimiento del robot con el publicador de velocidades. También conviene inspirarse de los callback implementados para cada subscriber.

Las funciones de callback no deben llamarse explicitamente en el código. Están hechas para ejecutarse cada vez que el tópico del subscriber reciba un mensaje. Esto se explicita en la creación del tópico, por ejemplo:

```
rospy.Subscriber("/odom", Odometry, self.odometryCallback)
```

Aquí, se crea un subscriber, el cual escucha el tópico `/odom`, donde recibe mensajes del tipo `Odometry` y cada vez que se recibe un mensaje por este tópico, automáticamente se ejecutará la función `self.odometryCallback` con el mensaje recibido como único argumento.

Rviz

RViz es una interfaz gráfica de ROS que permite observar información sobre sensores, el robot y el ambiente. Con la simulación corriendo,se lanza rviz utilizando el siguiente comando:

```
$ rosrun rviz rviz
```

Se debe configurar rviz para mostrar los mensajes que se deseen. Para hacer esto haga click en el botón “add” y seleccione si buscar por tipo de mensaje o por tópico. Al elegir un mensaje debería aparecer en la pestaña de la izquierda y se puede ver la configuración del display. No olvidar seleccionar un frame válido para el parámetro `global_frame`, por ejemplo world, base_link, odom u otros.

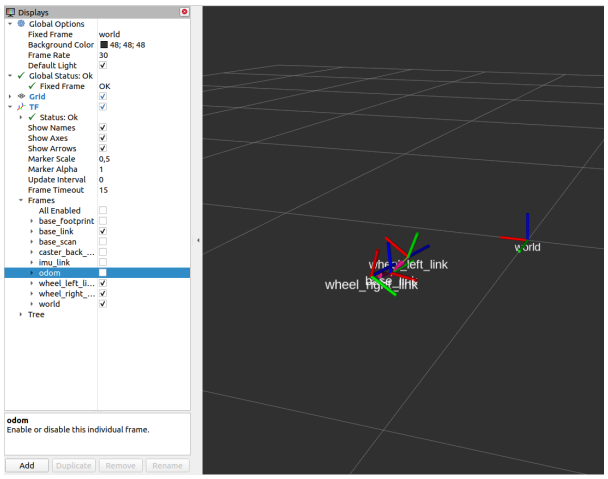

Figura 1: Ejemplo RViz