PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN
IIC-2685 ROBÓTICA MÓVIL

LAB 3: NAVEGACIÓN REACTIVA CON TURTLEBOT (REAL) Y LOCALIZACIÓN

Fecha de Entrega: viernes 24 de junio de 2022

Introducción

¡ Ha llegado el momento !. Después de todos estos meses aprendiendo ROS y teoría sobre mecanismos de navegación robótica, tendremos la oportunidad de aplicar nuestros conocimientos sobre un TurtleBot real. Para ello, aplicaremos parte del software desarrollado en el laboratorio 2 y veremos como se comporta en la vida real.

Junto a lo anterior, seguiremos utilizando nuestro simulador para poder programar nuestro primer algoritmo de robótica cognitiva desde la comodidad de nuestros hogares. En particular, desarrollaremos un algoritmo de localización que permitirá al robot responder una de las preguntas más importantes que un robot móvil puede hacerse: *¿ Dónde estoy ?*.

1 Navegación Reactiva con TurtleBot (real)

Para que nuestra primera experiencia con el TurtleBot real nos deje gratos recuerdos, aplicaremos y adaptaremos un algoritmo ya desarrollado en el Laboratorio 2. Este algortimo corresponde a los movimientos controlados para seguimiento de pasillo, descrito en la sección 2 del Laboratorio 2.

Para ello deberán instalar sus códigos en los netbooks que controlan los robots, lo cual implica la creación de workspace, creación de paquete, estableceimiento de variables de entorno medieante configuración de archivo *.bashrc*, etc.

Como resultado de esta experiencia, solo deberá presentar la demostración práctica del TurtleBot navegando a través de un pasillo (con algunas curvas) sin chocar con las paredes.

Considerando que estamos entrando a fines de semestre y el uso del laboratorio se incrementa, usted podrá presentar esta parte del laboratorio antes de la fecha oficial establecida en el encabezado. Una vez que tengan esta parte funcionando, solo deberán ponerse de acuerdo con el profesor y mostrar sus resultados (sin slides).

2 Localización

En el laboratorio anterior, aprendimos a controlar los movimientos del robot para aproximarse a las poses deseadas, y a dotarlo de comportamientos reactivos sin información del ambiente donde se moverá. En nuestro aprendizaje sobre el diseño de robots autónomos, uno de los elementos más importantes es disponer de un mapa del entorno, y que el robot sea capaz de razonar sobre su posición relativa al mapa usando sus sensores. Este par de elementos en su conjunto es lo que se conoce como Localización con mapa conocido.

Para la realización del laboratorio, dispondrán de un mapa métrico simple, el cual se compone de dos archivos:

- **`mapa.pgm`**: archivo de imagen en formato PGM (Portable Gray Map), donde el valor 0 corresponde a casilla ocupada, 255 a casilla desocupada, y valores intermedios a celdas con ocupación desconocida.

Como corresponde a una imagen, puede ser leída directamente con OpenCV. Notar que el eje x en el mapa son las columnas de izquierda a derecha en la imagen, y el eje y, las filas de abajo hacia arriba en la imagen. La figura 1 muestra el mapa que utilizarán en la presente experiencia.

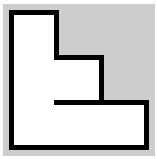

Figure 1: Imagen de laberinto que se utilizará como mapa.

- **`mapa.yaml`**: archivo de parámetros del mapa. Importantes son el origen y la resolución, que permite hacer la conversión de una coordenada (u, v) en pixeles a una coordenada (x, y) en metros.

```
image: map.pgm
resolution: 0.01
origin: [0.0, 2.7, 0.0]
negate: 0
occupied_thresh: 0.65
free_thresh: 0.196
```

El formato de los archivos descritos anteriormente es definido por ROS. Esto permite facilitar su lectura y publicación a través de tópicos, utilizando mensajes estándar que pueden ser recibidos e interpretados por cualquier nodo. Para mayor información, pueden revisar la especificación en:
`http://wiki.ros.org/map_server#Map_format`.

Para el caso particular de nuestro simulador, usted podrá abrir el mapa a través del menú "File", y escogiendo la opción "Open map". Una vez abierto el archivo yaml, además de mostrar la imagen del mapa en la interfaz gráfica, se publicará su descripción completa (mapa y metadata) en el tópico `/map`, la cual quedará disponible para cualquier nodo que desee obtenerla.

En esta experiencia, trabajaremos con un sensor del tipo rangefinder que simula un lidar 2D y entrega informaciones de distancias en un plano. Las lecturas de este sensor son publicadas en el tópico `/scan` utilizando un mensaje del tipo `sensor_msgs::LaserScan`.

Para su utilización, deberá levantar el nodo que lo implementa a través del archivo *lidar_simulator.launch*, el cual viene incluido dentro de los archivos *lanzadores* del simulador. Se recomienda fuertemente no levantar el nodo *kinect_simulator*, para así hacer un uso eficiente de los recursos de procesamiento de su equipo.

Para interpretar correctamente los datos recibidos a través del tópico */scan*, deberá considerar lo siguiente:

- Los valores numéricos recibidos estarán en unidades de metros
- El valor de z_{max} es igual a 4.0 [m]
- El rango total de lectura abarca un ángulo de 180°, los cuales varían entre -90°y 90°. Sin embargo, dado que el ángulo de visión del kinect es inferior a 180°, las lecturas válidas estarán en un rango de visión de solo 57°y el resto entregará el valor z_{max}. El ángulo de 0°es considerado como el ángulo formado por la línea que se proyecta desde el frontis del robot hacia adelante, tal como se muestra en la figura 2

Localización usando Filtro de Partículas

El trabajo del laboratorio consiste en localizar al robot dentro de un mapa conocido. Para esto, su labor será programar un localizador basado en Filtro de Partículas. Usando el mapa conocido y la lectura del *laser*, la tarea es encontrar la pose más probable del robot mediante una secuencia de movimientos y lecturas de laser.

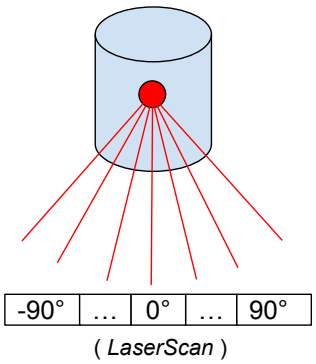

Figure 2: Interpretación geométrica de datos transportados dentro de mensaje *LaserScan*.

Se recomienda utilizar los modelos *Likelihood Fields* o *Map-matching* como modelo de sensor, según el que más les acomode.

Usando funciones de *OpenCV*, deberá mostrar una imagen en vivo donde se aprecie el mapa y las partículas en cada iteración de la secuencia de movimientos, demostrando que al cabo de unas pocas iteraciones las partículas tienen una forma de nube compacta. Una vez esto ocurra, deberá imprimir en pantalla la pose más probable del robot. Se recomienda fuertemente que la visualización de las partículas sobre el mapa sea implementada en un nodo independiente, el cual deberá recibir los datos desde el nodo encargado de los cálculos.

Para determinar cuándo el robot está localizado, deberán definir un criterio que les permita evaluar dicha situación. Este criterio puede ser, por ejemplo, cuando la distribución de las partículas converge alderedor de un punto único. Una vez establecida la localización, debe usar la capacidad de expresión del robot e indicar por los parlantes que se encuentra localizado. Pueden usar la expresión que deseen para este efecto.

Para poder buscar más información del ambiente que le ayude a mejorar su localización, usted deberá explorarlo utilizando una rutina de navegación reactiva como la desarrollada para el seguimiento de pasillos. En esta oportunidad, se aconseja navegar reactivamente a una distancia fija de alguna de las paredes (izquierda o derecha), y en caso de encontrar un obstáculo de frente, rotar consecutivamente 90°hasta encontrar un camino abierto que pueda seguir. Deben programar sus movimientos de manera tal que no choque con ninguna pared.

Como modelo de movimiento, pueden usar una distribución Normal (Gaussiana) tanto en distancia recorrida como en ángulo.

Como éste es un laboratorio con fines académicos, no es necesario que su localizador sea en tiempo real. Puede demorar unos segundos en obtener el *likelihood* de la medición en cada etapa de la secuencia de movimientos. Pueden suponer que no existirán obstáculos dentro del mapa.

En su presentación deberán indicar el modelo de sensor utilizado, y explicar brevemente que estrategia siguió para programarlo. Por ejemplo, explicar qué medidas tomó para hacer su código más eficiente y robusto. Junto con ello, deberá incluir una secuencia de imágenes que muestre la evolución de las partículas al ejecutar cada movimiento.

Demostración

Para la demostración deberá levantar sus códigos mediante un archivo launch de nombre *localization.launch*. Este archivo deberá ser independiente del archivo launch que levantará los nodos del simulador, ya que como primer paso, usted deberá ubicar al robot en una posición arbitraria del laberinto utilizando el modo *p*, y posteriormente, comenzar el proceso de exploración y localización ejecutando:

$$roslaunch < package\_name > localization.launch$$