Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Escuela de Ciencias Exactas y Naturales
Departamento de Ciencias de la Computación
R-521 — Robótica Móvil

Trabajo Práctico

ROS2 y Simulador Gazebo

1. Introducción

El objetivo de este trabajo práctico es comprender el modelo cinemático y la estimación de pose (posición y orientación) mediante el cálculo de odometría de un robot con ruedas de tracción diferencial que se mueve sobre una superficie plana. Para esta tarea se utilizará ROS2 y el simulador Gazebo.[1]

2. Entrega

- Se debe proveer un repositorio git que contenga el código desarrollado y un archivo `README.md` con las instrucciones de compilación y ejecución. Se recomienda hacer una imagen Docker para facilitar la reproducción de los resuldatos.
- Se debe entregar un informe en Lyx o LaTeX explicando el trabajo realizado y analizando los resultados obtenidos.
- En caso de utilizar el framework `https://www.theconstructsim.com`, se deberá proporcionar además el link al proyecto.

3. Evaluación

- Haciendo únicamente los ejercicios obligarorios (no opcionales), el trabajo tiene una nota máxima de 8. Los ejercicios opcionales permiten llegar a la nota máxima de 10.
- Entregas Fuera de Término: Si la entrega se realiza durante la primer semana luego del plazo, se decuentan 2 puntos. Si la entrega es más tarde que esto último la nota máxima es de 6.

4. Instrucciones de compilación

Pasos para trabajar en el simulador Gazebo con el robot TurtleBot3. Mas información en: `https://navigation.ros.org/getting_started/`

a) (Opcional) Si no puede trabajar en su propia computadora (con la última versión de Ubuntu LTS disponible) entonces realice los siguientes pasos:

i. Iniciar sesión en `https://www.theconstructsim.com`.

ii. Crea un proyecto ROSject y seleccionar ROS2 `<distro>` como distribución.

b) Instalar los paquetes de ROS2 para la simulación del robot TurtleBot3:

Actualizar la lista de paquetes de linux

```
sudo apt update
```

Instalar los paquetes de Nav2

[1] Este trabajo práctico está basado en el trabajo práctico del curso Fundamentos de Robótica Móvil del Departamento de Ingeniería Electrónica de la Facultad Regional Córdoba de la Universidad Tecnológica Nacional `https://www.profesores.frc.utn.edu.ar/electronica/fundamentosroboticamovil`.

```
sudo apt install ros-<distro>-navigation2
sudo apt install ros-<distro>-nav2-bringup
```

Instalar los paquetes de Turtlebot 3

```
sudo apt install ros-<distro>-turtlebot3*
```

c) Descargar el script `dump_odom.py` provisto por la cátedra

5. Ejecución de la simulación

a) Setear las variables de entorno:

```
export TURTLEBOT3_MODEL=waffle
export GAZEBO_MODEL_PATH=$GAZEBO_MODEL_PATH:/opt/ros/<distro>/share/
   turtlebot3_gazebo/models
```

b) Ejecutar simulación

```
ros2 launch nav2_bringup tb3_simulation_launch.py headless:=False
```

c) Una vez corriendo, setear la pose inicial del robot desde RViz. Para esto primero presionar el botón "2D Pose Estimate" y luego hacer click es la zona donde se encuentra el robot en el mapa. Esto es para que el sistema de localización del robot tenga una semilla inicial de dónde esta. Luego presionar "Navigaton2 Goal" y luego hacer click en algún lugar del mapa para indicarle al robot que debe navegar hasta ahí.

d) Utilizar el launch y el modelo world provisto por la cátedra para lanzar la simulación pero sin un mapa y sin el sistema de localización. El launch provisto es una modificación del archivo `tb3_simulation_launch.py` donde se comentó el código relacionado a la localización (`slam`) y al mapa (`map`). El modelo world `world_only.model` resulta de comentar el código relacionado al modelo del mundo a utilizar.

e) Una vez teniendo el robot corriendo en un mundo sin obstáculos y sin un sistema de localización, enviar comandos de velocidad para mover el robot:

i. En una nueva terminal o pestaña, ejecutar el comando:

```
ros2 topic pub --rate 1 /cmd_vel geometry_msgs/msg/Twist "{linear: {x: 0.2, y:
    0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.2}}"
```

ii. Detener el robot fijando la velocidad lineal y angular a cero.

f) Comandar el robot con el teclado (teleoperación), para ello ejecutar:

Puede ejecutarse el mismo comando de teleoperación por teclado

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

(ver la información en pantalla y decrementar las velocidades si el moviemiento del robot resulta demasiado agresivo)

6. Ejercicios

Ejercicio 1. Determinar de forma analítica el radio del camino circular que realiza el robot al ajustar la velocidad lineal y angular a valores constantes. Realizar el cálculo para dos velocidades cualquieras

teniendo en cuenta las velocidades máximas del robot.

Nota: Los límites de velocidad y los parámetros cinemáticos (el radio de la rueda R y la distancia entre ruedas b) de los diferentes modelos del robot TurtleBot3 se obtienen de las especificaciones[2].

Ejercicio 2. Calcular la velocidad lineal y angular para que el robot realice un camino circular con radio de 1 m.

Ejercicio 3. Calcular las velocidades lineales y angulares de las ruedas (izquierda y derecha) del robot para el camino circular del punto anterior.

Ejercicio 4. Generar un registro (log) de odometría y velocidad del robot, para lo cual hay que ejecutar nuevamente la simulación y utilizar el script `dump odom.py`. Este script muestra en pantalla 6 columnas con los siguientes datos: tiempo (timestamp), coordenadas x, y, orientación, velocidad lineal y angular. El registro de datos debe ser realizado con el robot en movimiento utilizando teleoperación por teclado. Para guardar los datos generados por el script hay que redireccionar la salida a un archivo como:

```
./dump_odom.py > log.txt
```

Ejercicio 5. Escribir un script en Python que cargue los datos del archivo log y genere gráficos de:

- **a)** el camino seguido por el robot,
- **b)** la trayectoria (pose respecto al tiempo), y
- **c)** la velocidad del robot respecto al tiempo.

Nota: Utilizar una relación de aspecto 1:1 para el gráfico del camino y evitar en lo posible el registro de datos iguales a cero.

Ejercicio 6. (Opcional) Obtener otro registro de datos para un camino circular del robot y graficar el camino y la trayectoria.

Ejercicio 7. (Opcional) Marcar tres puntos cualquiera en el gráfico del camino del robot y sus correspondientes puntos en la trayectoria (Ver Figura 1). No elegir los puntos de inicio y final del camino.

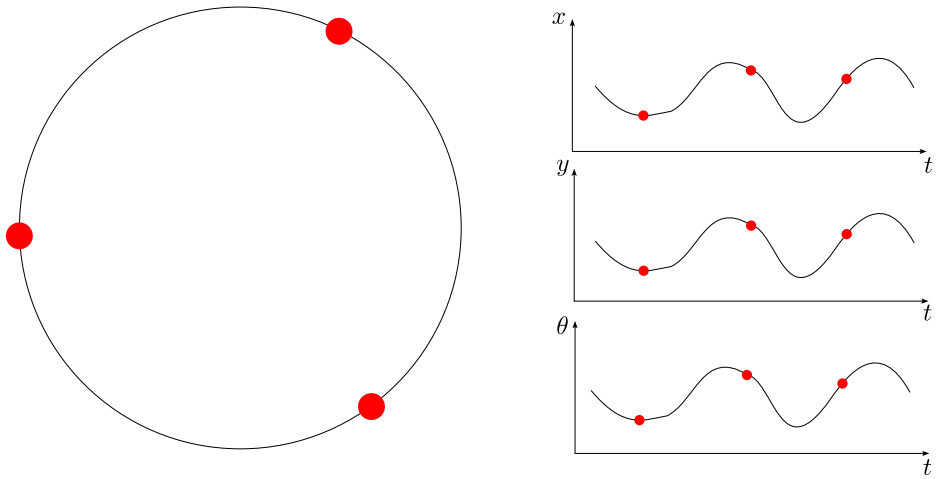

Figura 1: Ejemplo de camino y trayectoria.

En base a los gráficos anteriores:

[2] `https://emanual.robotis.com/docs/en/platform/turtlebot3/features/`

a) ¿Cuáles son los rangos de valores de las coordenadas x e y y por qué?

b) ¿Cuál es el rango de valores de la orientación del robot y por qué?

c) Obtener diferentes registros y gráficos para caminos circulares con diferentes valores (positivos y negativos) de velocidades lineales y angulares (utilizar todas las combinaciones de signos posibles). Indicar en los gráficos el sentido de avance del robot.

d) Describir cuál sería la secuencia de comandos de velocidad a aplicar al robot para seguir uno de los caminos mostrados en la Figura 2 (elegir solo uno).

Nota: Para setear los comandos de velocidad por un tiempo determinado se puede utilizar el comando `ros2 topic pub` con los argumentos `-1` y `--keep-alive`.

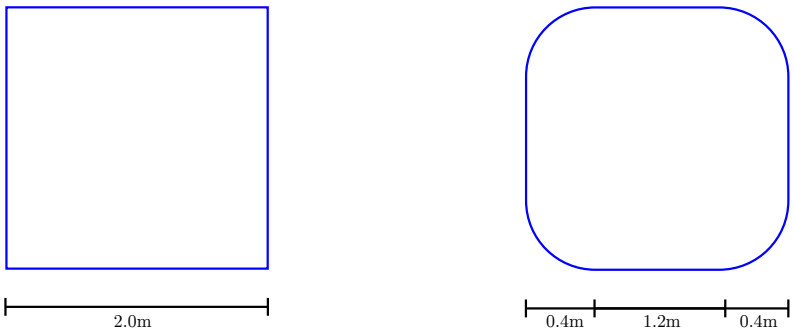

Figura 2: Caminos cuadrados.

Ejercicio 8. Realizar una simulación donde:

- El mundo debe contar con varios cilindros de un mismo radio r. Los cilindros deben estar distribuidos en el entorno y ser todos observados por el láser del robot al momento de inicio de la simulación.
- Utilizar una medición láser para generar un mapa de la escena observada por el robot en simulación. Para esto debe detectar los cilindros en la medición laser obtenida. Cada cilindro es utilizado como un landmark en el mapa virtual del robot. Debe estimar el centro del cilindro, dicha posición será la posición de un landmark en el mapa virtual del robot.
- Debe crear un mapa de landmarks, publicarlo y visualizarlo en RVIZ utilizando el marker de cilindro. Obtenga una captura de pantalla de RViz donde se visualice las mediciones del láser y los landmarks reconstruidos.