TP1 Robótica Móvil

Sacha Varela
Gerónimo González Marino

8 de octubre de 2025

Introducción

A continuación se presenta la resolución de cada ejercicio con los resultados finales obtenidos en forma de matrices e imágenes. Para el detalle se redirige al código ubicado en el repositorio Git del siguiente link Repositorio Robótica Móvil Gerónimo González.

Ejercicio 1 y 2

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.

La ecuación que relaciona la velocidad angular con la velocidad lineal y el radio de giro de la trayectoria descrita es

$$R_{\text{giro}} = \frac{v}{\omega}, \tag{1}$$

siendo R_{giro} el radio de giro del robot, v la velocidad lineal y ω la velocidad angular.

Las velocidades máximas del robot son $v_{\max} = 0{,}26$ m/s y $\omega_{\max} = 1{,}82$ rad/s, obtenidas de [1].

Teniendo en cuenta dichas restricciones, podemos ver el radio que realiza el robot cuando ambas velocidades son máximas. Luego, para obtener un camino circular con radio 1 metro, ambas velocidades deben ser iguales.

v	ω	R_{giro}
0.26 m/s	1.82 rad/s	0.14 m
0.2 m /s	0.2 rad/s	1.0 m

Tabla 1: Relación entre velocidad lineal, angular y radio de giro

Ejercicio 3

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

siguiendo la cinemática de un robot diferencial y sabiendo que la distancia entre ruedas es D, si el robot se mueve a lo largo de un círculo con un radio R, la rueda interior (su centro) se mueve a lo largo de un círculo con radio $R - D/2$ y la rueda exterior (su centro) con radio $R + D/2$. De esta manera

$$R_{int} = R - \frac{D}{2} \qquad R_{ext} = R + \frac{D}{2} \tag{2}$$

Para calcular las velocidades lineales del centro de cada rueda hacemos:

$$v_{ri} = w * R_{int} \qquad v_{re} = w * R_{ext} \tag{3}$$

Ya que la velocidad angular del centro de la rueda respecto del centro de la trayectoria realizada es la misma que para el centro del robot y para todos los otros puntos. Luego para calcular las velocidades angulares de cada rueda respecto a su eje utilizo la misma relación $v = w * r$ ahora con la velocidad de la rueda y el radio de la rueda. de esta manera:

$$w_{int} = \frac{v_{int}}{r} \qquad w_{ext} = \frac{v_{ext}}{r} \tag{4}$$

Los cálculos anteriores se resumen en la siguiente tabla:

Tabla 2: Parámetros de la base y ruedas

Parámetro	Base_link	Rueda int	Rueda ext
Radio trayectoria (m)	1	0.8565	1.14350
Radio característico (m)	0.1435	0.033	0.033
Velocidad lineal (m/s)	0.26	0.22	0.30
Velocidad angular (rad/s)	0.26	6.75	9.01

Ejercicio 4 y 5

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.

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.

Se realizo un código el cual toma el archivo de salida del script “dump_odom.py” y se realizaron los graficos solicitados obteniendo como salida las figuras 1, 2 y 3. Los datos utilizados se encuentran en [2].

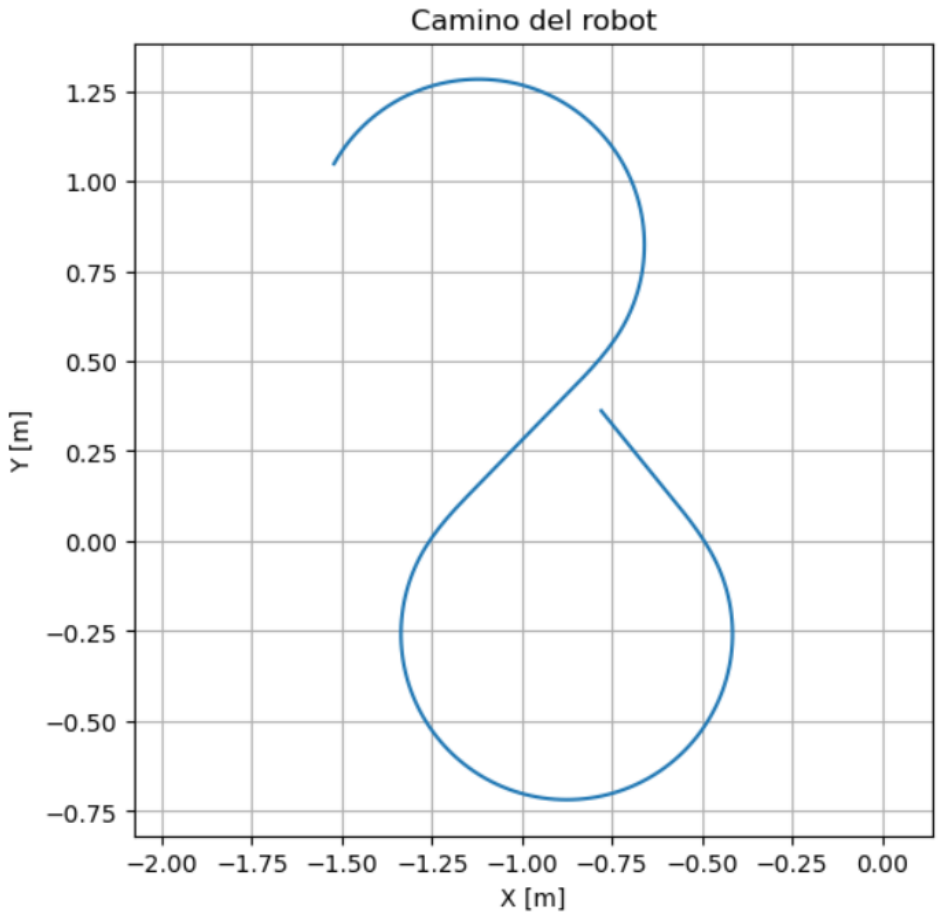


Figura 1: Trayectoria realizada por el robot

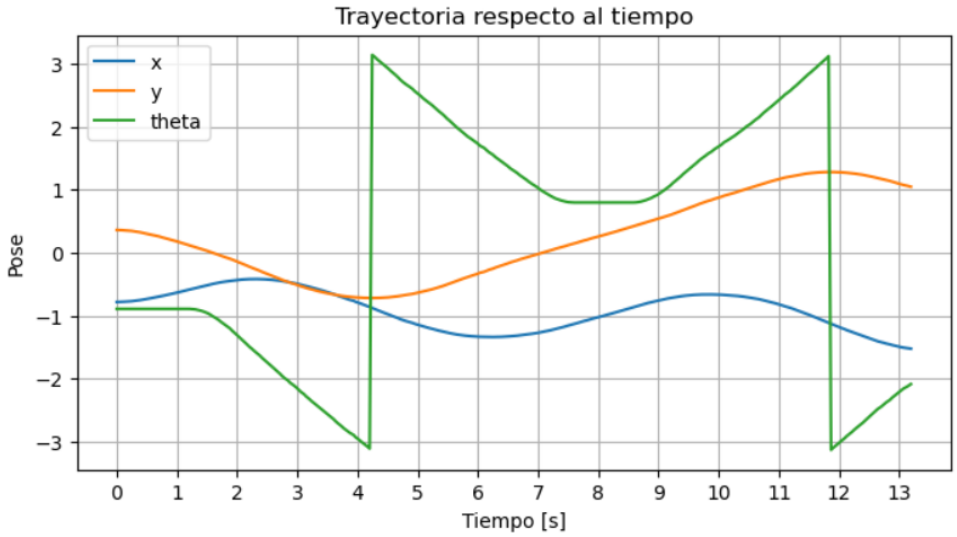


Figura 2: Pose del robot durante la trayectoria

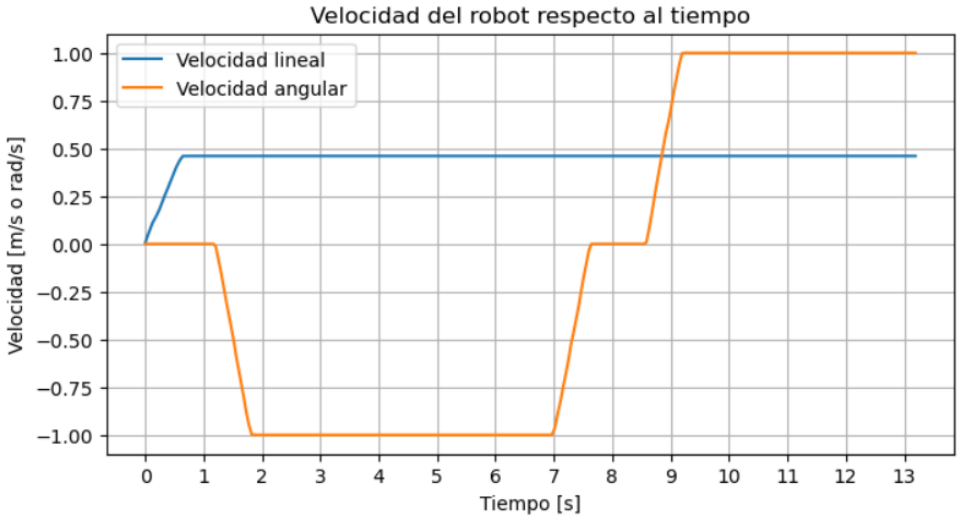


Figura 3: Velocidades durante la trayectoria del robot

Ejercicio 6 y 7 (opcionales)

Obtener otro registro de datos para un camino circular del robot y graficar el camino y la trayectoria. Marcar tres puntos cualquiera en el gráfico del camino del robot y sus correspondientes puntos en la trayectoria. No elegir los puntos de inicio y final del camino.

Se publicó en el tópico `/cmd_vel` una velocidad angular de 0,8 rad/s y una velocidad lineal de 0,2 m/s. Se exportaron los registros con el código "dump_odom.py", el cual se encuentra en [2].

Las gráficas obtenidas se observan a continuación.

Tabla 3: Valores elegidos en trayectoria

Indices	tiempo	x	y
50	2.29	-0.20	0.13
150	7	0.26	0.31
250	11.78	-0.23	0.26

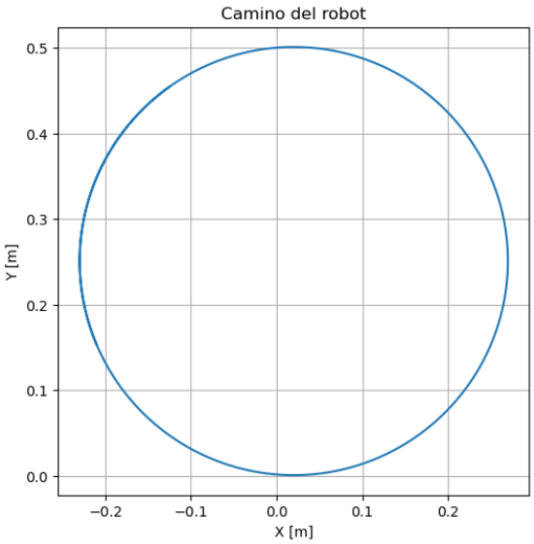
En base a los gráficos anteriores:

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

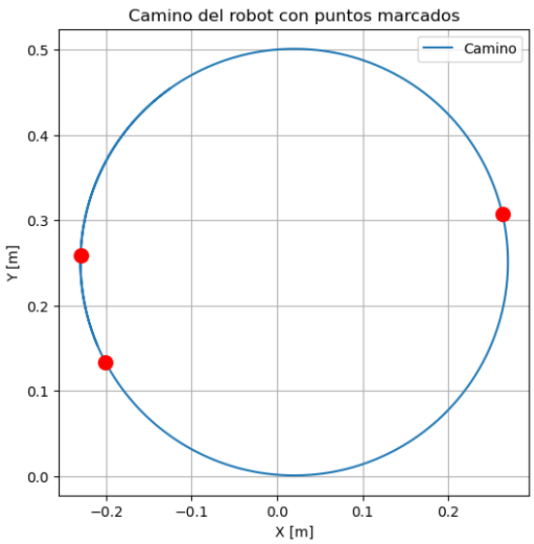
b) ¿Cuál es el rango de valores de la orientación del robot y por qué?

Tabla 4: Rango de valores

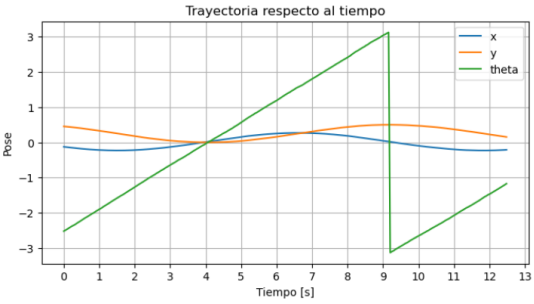
parámetro	mínimo	máximo	variación
x	-0.229	0.263	0.493
y	0.133	0.307	0.174
ϑ	-1.08	1.80	2.88



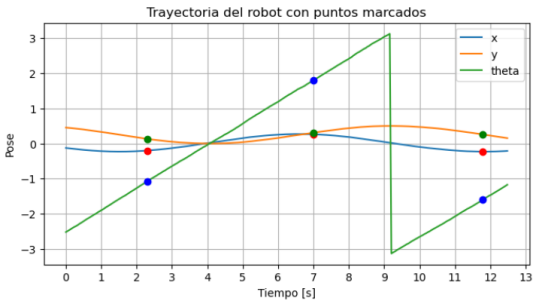
(a) Camino del robot



(b) Puntos en camino del robot



(c) Trayectoria del robot



(d) Puntos en trayectoria del robot

Figura 4: Trayectoria circular y puntos marcados

El valor numérico de estos puntos dependerá de **la posición inicial**. Sin embargo, debido a su trayectoria **circular**, la variación entre los extremos debería estar acotada por el diámetro del círculo, que para las velocidades dadas, corresponde a 0.5m. Por último, la variación de θ tomará cualquier valor entre 0 y 2π dependiendo de los puntos seleccionados. La trayectoria se realiza para un Radio de 1m.

$$0{,}5m > 0{,}49m \qquad 0{,}5m > 0{,}174m \tag{5}$$

La variación entre las coordenadas de los puntos es acorde a la acotación de la trayectoria. Se puede observar, en el gráfico, que 2 de los puntos seleccionados representan casi lados opuestos del circulo, por lo que su variación es próxima al diámetro de la trayectoria. $0{,}49m \approx 0{,}5m$.

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.

Se realizan caminos circulares con todas las opciones posibles de velocidad lineal y angular, las mismas se listan como **circXX** donde XX puede ser **pp**, **mm**, **pm**, **mp** (p:plus, m:minus). Con el launch Dump_odom se levantan los datos [3] y luego se grafican.

Si se graficaran los círculos en una misma figura solo se verían 2 trayectorias circulares, donde los pares de combinación **pp** y **mm** corresponderían al circulo de la izquierda y los pares **pm** y **mp** al circulo de la derecha.

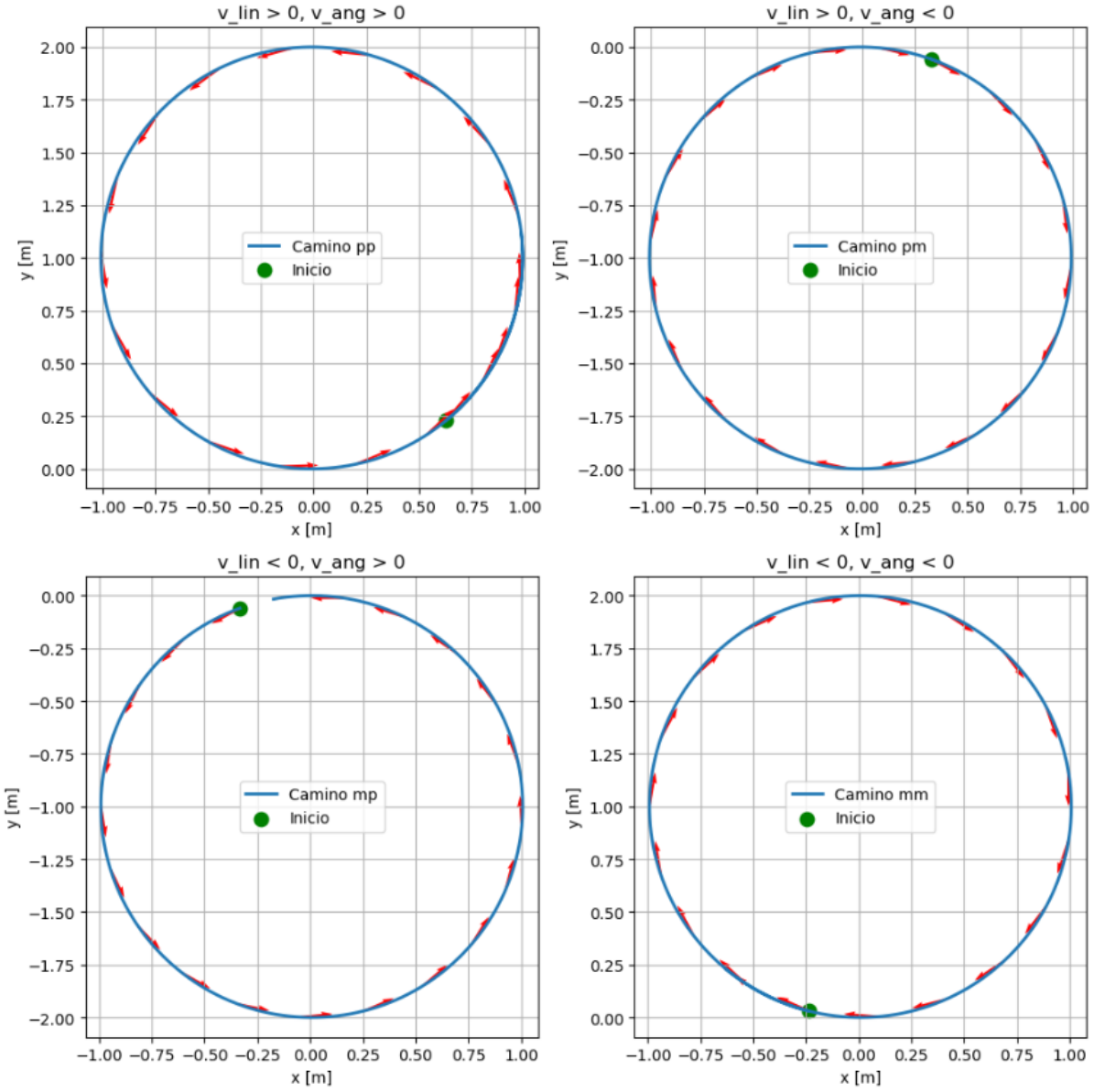


Figura 5: Trayectorias circulares en todas las combinaciones

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).

Para dibujar la figura 2 de los caminos posibles (cuadrado de lado 2m con esquinas redondeadas de radio 0.4m) se separa el conjunto de comandos en 2 tipos: Avance recto, esquina curva. Además para cada uno hará falta definir las velocidades lineales y angulares.

Para el caso de avance recto se selecciona:

$$v_{lineal} = 0{,}2 \;\; m/s \tag{6}$$

Para el caso de esquina curva:

$$v_{lineal} = 0{,}104 \ \ m/s \qquad y \qquad v_{ang} = 0{,}26 \ \ rad/s \tag{7}$$

De esta manera no se superan las velocidades límites del robot, se tiene un poco más de control sobre el tiempo en el que se envían los mensajes y se garantiza un radio curvo de 0.4m para las esquinas (r=v/w). Luego se calculan los tiempos teóricos durante los que se debe publicar cada mensaje.

$$para \ 1{,}2m \ rectos \qquad 1{,}2/0{,}2 = 6 \ segundos \tag{8}$$

$$para \ 90^o \ de \ giro \qquad \frac{\pi}{2}/0{,}26 \approx 6{,}041 \ segundos \tag{9}$$

Luego se publican los mensajes en formato .YAML en el tópico /cmd_vel durante el tiempo calculado de la siguiente manera.

1. Enviarle velocidad lineal
   ros2 topic pub –once /cmd_vel geometry_msgs/msg/Twist '{linear: {x: 0.2, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}'
   durante 6 segundos

2. Rotar 90° **ros2 topic pub –once 1 /cmd_vel geometry_msgs/msg/Twist '{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.785}}'**
   durante 6.041 segundos

Si bien en la teoría funciona, en la práctica el robot simulado está a lazo abierto, por lo que no tiene acciones de control y aparece mucho ruido.entre los tiempos de publicación de los nodos y la ejecución de comando, por lo que la figura no se realiza como debe ser.

Para solucionar esto se realiza un ajuste empírico con el simulador abierto. Además, en vez de publicar el mensaje una sola vez con "`--once`", se elige una frecuencia de publicación que aporta otra variable más para ajustar el simulador empíricamente.

Debajo se presentan 4 casos del experimento con distintos tiempos y frecuencias de publicación que van de peor a mejor. Los valores finales ajustados empíricamente son:

$$tiempo\_lineal = 5{,}7 \ seg \qquad timepo\_curva = 6{,}5 \ seg \qquad frec = 20 \ Hz \tag{10}$$

Los log.txt de cada experimento se encuentran en el repositorio [3], bajo el nombre de camino1, camino2, camino3 y camino4. En el caso de correrlos localmente en el código presentado es IMPERATIVO cambiar el path para leer cada archivo correctamente.

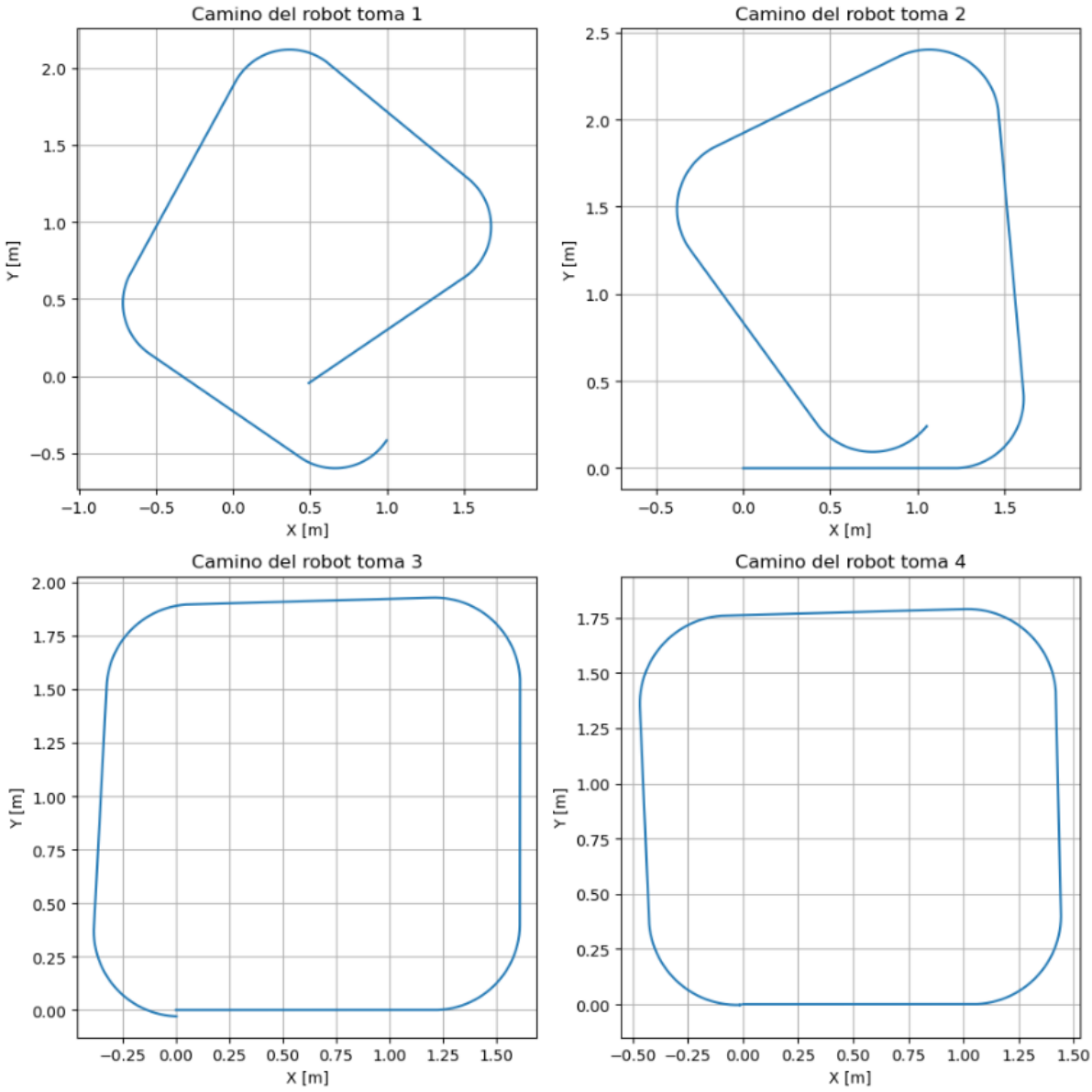


Figura 6: Resultados del tunning de comandos sobre Waffle_robot

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.

A la simulación se le agregaron cilindros seleccionando dicha forma dentro de las opciones propias del simulador Gazebo y colocándola en el mundo, teniendo en cuenta que estas figuras deben ser visibles para el LIDAR del robot.

Una vez que la simulación estuvo funcional, se desarrolló un nodo que se suscribe al tópico /scan (donde se publican las mediciones del LIDAR) y publica en el tópico /landmark la estimación del centro de cada cilindro. En RViz se agregó el Display de Marker y se configuró para que escuche dicho tópico.

El código del nodo mencionado es "detect_landmarks.py", disponible en [4]. Este código toma las mediciones del LIDAR, descarta aquellas que están demasiado alejadas y agrupa los puntos contiguos restantes, formando conjuntos de mediciones asociados a un mismo cilindro. Con estos grupos se estima el centro del cilindro calculando el promedio de las distancias y de los ángulos de los puntos que los componen. Finalmente, se ajusta la posición de los centros de los cilindros considerando el radio del cilindro y aplicando un desplazamiento. Dicho radio se calcula dentro del nodo suponiendo que el diámetro del cilindro es la diferencia entre la posición del primer punto del grupo, con la última del mismo.

Con el centro del cilindro estimado, se publica un marcador de tipo cilíndrico y color rojo para ser visualizado en RViz. El funcionamiento del código puede observarse en las figuras 7 y 8, donde no solo se aprecian los landmarks, sino también las mediciones del LIDAR.

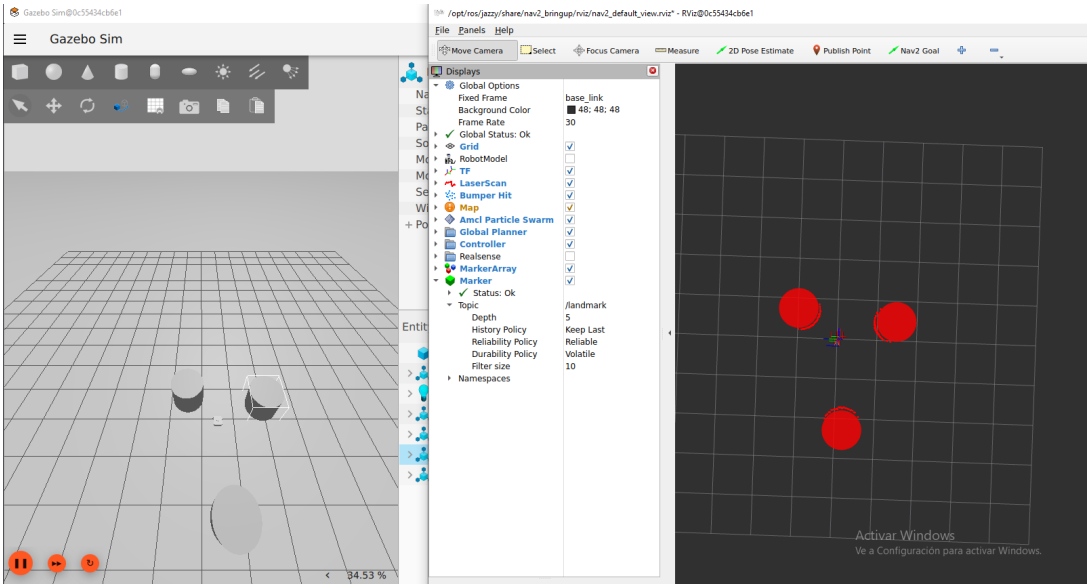


Figura 7: Captura de simulador y visualizador con la detección de cilindros en ejecución

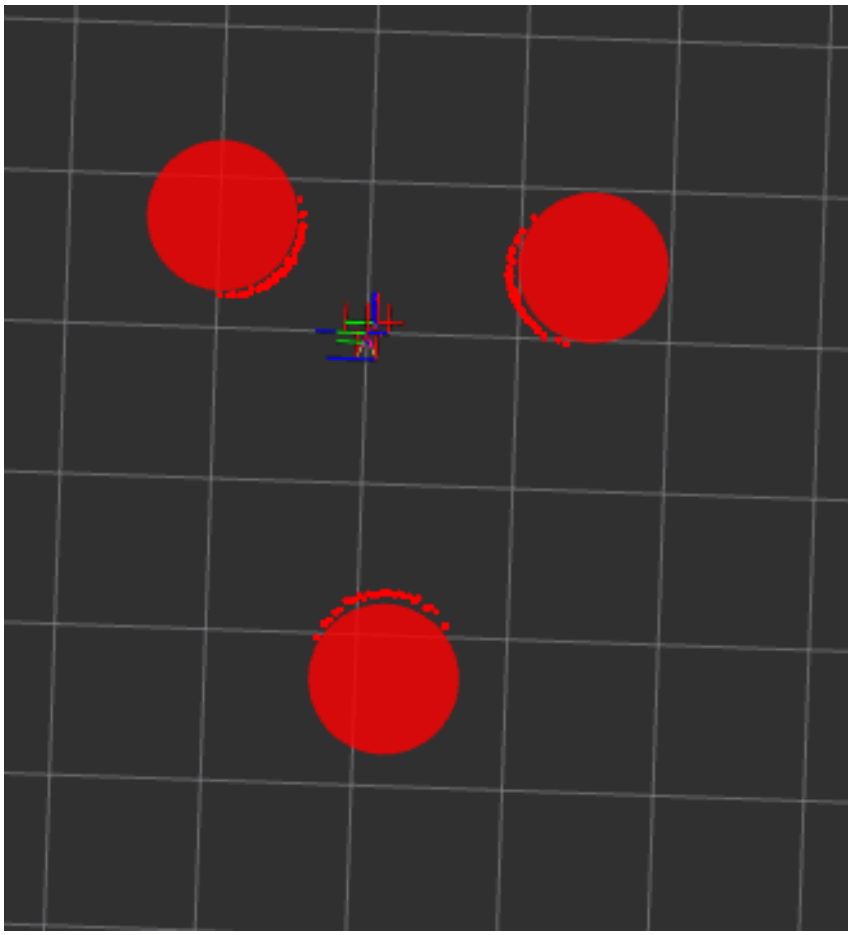


Figura 8: Zoom sobre el visualizador con detección de cilindros en ejecución

Referencias

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

[2] https://github.com/geronimogonzalez/Robotica-movil/blob/main/TP-2/Ejercicios_1_a_7.ipynb.

[3] https://github.com/geronimogonzalez/Robotica-movil/tree/main/TP-2/Logs-graficas

[4] https://github.com/geronimogonzalez/Robotica-movil/blob/main/TP-2/Codigo-Ej8/detect_landmarks.py.