TP Final: Seguimiento de trayectoria a lazo cerrado basado en EKF para un robot omnidireccional

Robótica Móvil

2^{do} cuatrimestre 2025

Introducción

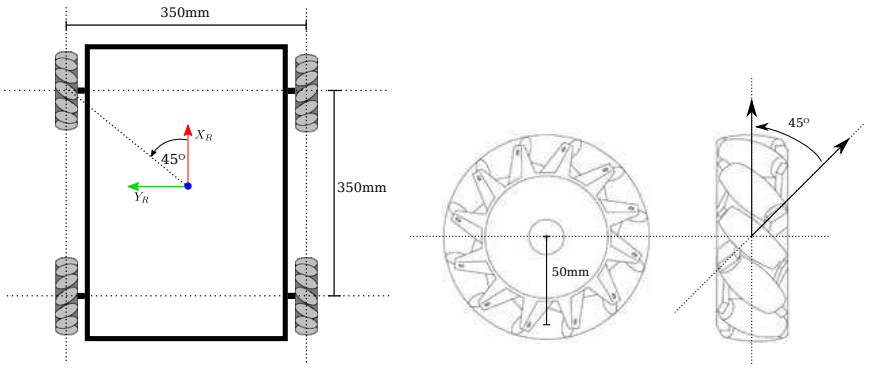
Un robot móvil omnidireccional (u holonómico) es capaz de desplazarse en cualquier dirección sin la necesidad de alcanzar previamente una orientación específica. Esta capacidad le permite al robot realizar trayectorias muy diversas y resultan de interés para el transporte de carga.

En este trabajo se busca unificar los contenidos vistos en la materia de manera de obtener un sistema funcional que permita la realización precisa de trayectorias utilizando un robot omnidireccional. Se trabajará con un entorno de simulación de CoppeliaSim y el sistema será desarrollado utilizando el entorno de desarrollo ROS2.

Plataforma omnidireccional

Existen numerosos tipos de robots omnidireccionales, nosotros trabajaremos con uno provisto de 4 ruedas fijas con un diseño particular el cual se lo conoce como Mecanum. Este tipo de rueda especial posee pequeños rodillos colocados alrededor del diámetro externo y montados en ángulo. Estos rodillos no son sensados ni actuados pero transmiten una porción de la fuerza ejercida, por el eje principal de la rueda, en dirección al ángulo definido por los rodillos. Esta configuración genera cuatro fuerzas debido a la dirección y velocidad de rotación de cada rueda, estas fuerzas pueden integrarse en una única fuerza total la cual permite el desplazamiento en cualquier dirección.

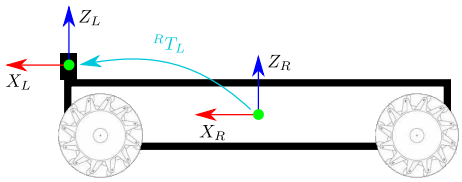
A continuación se presentan las dimensiones de la plataforma que utilizaremos:



Los rodillos interactúan de manera pasiva con el suelo mientras que los ejes de cada rueda son actuados y poseen encoders que permiten sensar la velocidad de rotación.

- La resolución del encoder corresponde a **500 ticks por revolución** de la rueda.
- El encoder acumula ticks positivos cuando la rueda rota "hacia adelante" mientras que acumula ticks negativos en caso contrario. Es decir, se trata de un encoder absoluto.

La plataforma cuenta además con un sensores láser el cual permite sensar objetos existentes delante del robot:



Entorno simulado

Para la realización del trabajo se provee un entorno simulado en CoppeliaSim: `omni_efk.ttt`. En la escena se presenta la plataforma robot antes descripta junto a una serie de postes situados alrededor.

Actuadores y sensores disponibles

El robot es capaz de acatar comandos de velocidad de rotación para cada una de sus ruedas, específicamente se suscribe a los siguientes tópicos:

1. `/robot/front_left_wheel/cmd_vel:` Velocidad objetivo de la rueda frontal izquierda.
2. `/robot/front_right_wheel/cmd_vel:` Velocidad objetivo de la rueda frontal derecha.
3. `/robot/rear_left_wheel/cmd_vel:` Velocidad objetivo de la rueda trasera izquierda.
4. `/robot/rear_right_wheel/cmd_vel:` Velocidad objetivo de la rueda trasera derecha.

El tipo de los mensajes recibidos es:

std_msgs/msg/Float64

float64 data

Este valor del mensaje representa entonces la velocidad objetivo de cada rueda en rad/s.
De manera acorde, se publican los ticks de encoders para cada una de sus ruedas a través del tópico:

5. `/robot/encoders`

El tipo de los mensajes enviados es:

robmovil_msgs/msg/MultiEncoderTicks

std_msgs/Header header
std_msgs/Int32[ ] ticks

El arreglo de enteros **ticks posee siempre cuatro posiciones**, una por cada rueda:

a. `ticks[0]:` Ticks de encoders de la rueda frontal izquierda.
b. `ticks[1]:` Ticks de encoders de la rueda frontal derecha.
c. `ticks[2]:` Ticks de encoders de la rueda trasera izquierda.
d. `ticks[3]:` Ticks de encoders de la rueda trasera derecha.

La información del sensor láser se publica por medio del tópico:

6. `/robot/front_laser/scan`

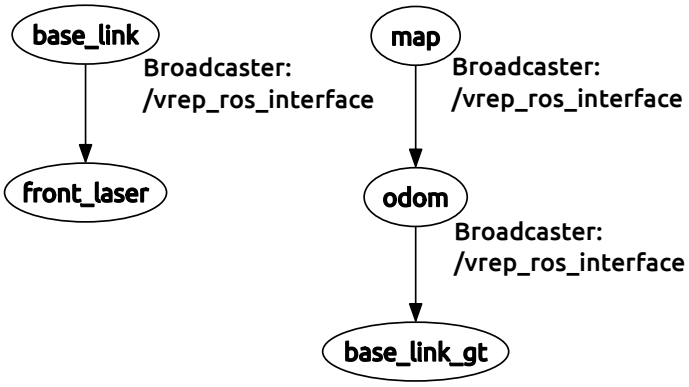
Y el tipo de mensajes enviados corresponde al visto durante la cursada:

sensor_msgs/msg/LaserScan

std_msgs/Header header
float32 angle_min
float32 angle_max
float32 angle_increment
float32 range_min
float32 range_max
float32[ ] ranges

Transformaciones e información del mapa

La simulación provee información de la relación espacial entre marcos de coordenadas involucrados. A continuación se presentan los diferentes marcos y la relación entre estos:



- **base_link → front_laser** (${}^{R}T_{L}$) **:** Pose del láser en referencia al marco de coordenadas del robot.
- **map → odom** (${}^{map}T_{odom}$) **:** Pose inicial donde comienza el robot en referencia al marco de coordenadas del mapa.
- **odom → base_link_gt** (${}^{odom}T_{R_{gt}}$) **:** Pose real del robot en referencia al marco inercial odom (donde comienza). Esta información es provista solo para tareas de depuración y como ground truth para los experimentos.

Así mismo, se provee la información del mapa. La locación de los postes se publica en el tópico:

7. **/posts**

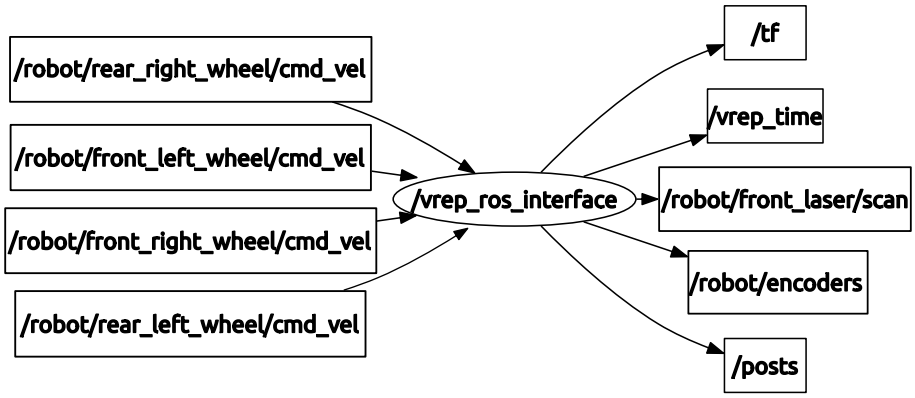
Utilizando el mensaje:

geometry_msgs/PoseArray

std_msgs/Header header
geometry_msgs/Pose[ ] poses

El arreglo `poses` es de 16 posiciones, una por cada poste presente en el mapa. Las poses se encuentran en relación al marco de coordenadas del mapa. Todos los postes poseen 0.1 metros de diámetro.

A continuación se presenta el modelo general de tópicos a los cuales publica y se suscribe la simulación:


/robot/rear_right_wheel/cmd_vel
/robot/front_left_wheel/cmd_vel
/robot/front_right_wheel/cmd_vel
/robot/rear_left_wheel/cmd_vel
/vrep_ros_interface
/tf
/vrep_time
/robot/front_laser/scan
/robot/encoders
/posts

Requerimientos a resolver

1 Modelo Cinemático

Dadas las particularidades de la plataforma omnidireccional con la que se trabaja, se debe resolver un modelo cinemático adecuado que permita caracterizar el movimiento del robot. En nuestro caso, los actuadores corresponden a las cuatro ruedas Mecanum antes descriptas.

Se requiere implementar el modelo cinemático propuesto por los autores del trabajo **"Kinematic Model Of Four Mecanum Wheeled Mobile Robot"**, de forma de poder controlar la plataforma que disponemos por medio de comandos de velocidad (cinemática inversa). Así mismo, se requiere además, estimar la velocidad lineal y angular del robot a partir de la información sensada por los encoders (cinemática directa).

Prestar especial atención a la definición de postura y configuración de ruedas (página 6), la tabla de parámetros (página 8) y las ecuaciones 19 a 26 (modelo cinemático).

1. Implementar un nodo de ROS2, el cual debe:

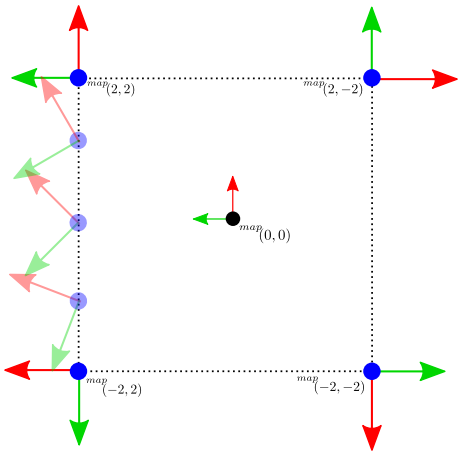
   a. Recibir comandos de velocidad lineal y angular `geometry_msgs/msg/Twist` por el tópico `/robot/cmd_vel` y publicar velocidades de rotación por cada rueda a través de los tópicos anteriormente descriptos.
   b. Utilizar las mediciones de los encoders para la estimación de la pose actual del robot (odometría) en relación al momento inercial (marco odom).
   c. Publicar la información odométrica como una transformación **odom** $\rightarrow$ **base_link** ($^{odom}T_R$) por el tópico `/tf` y a su vez como un mensaje de tipo `nav_msgs/msg/Odometry` por el tópico `/robot/odometry`.

2 Control a lazo cerrado y seguimiento de trayectoria

Al trabajar con un robot holonómico es posible controlar las velocidades del mismo de manera independiente. Cualquier posición es alcanzable independientemente de la orientación que tenga el robot en un momento dado. Esto permite simplificar los métodos de control y convergencia a una pose objetivo.

1. Plantear un controlador Proporcional (P) para cada uno de los grados de libertad del robot que permitan la convergencia de la pose del robot hacia una pose objetivo. Deberán utilizar la estimación de pose provista por la odometría como *feedback*, y en cada actualización determinar velocidades lineales y angulares de control que permitan una correcta convergencia. Es de importancia llevar a cabo pruebas experimentales que permitan establecer las constantes involucradas en el método.

A partir del método planteado, se requiere el seguimiento de una trayectoria cuadrada de 2 metros de lado. Al seguir esta trayectoria el robot debe mantener una orientación "opuesta al centro" como se observa en la figura siguiente:



2. Establecer poses objetivo intermedias (waypoints) que describan la trayectoria requerida en pequeños intervalos. El seguimiento de la trayectoria deberá realizarse seleccionando, de manera sucesiva, una pose objetivo cercana (*Pursuit Based*).

3. Implementar un nodo de ROS el cual debe:

   a. Publicar comandos de velocidad de control (lineal y angular) de manera periódica a través del tópico `/robot/cmd_vel` que permitan la convergencia hacía una pose objetivo.

   b. Utilizar la estimación de la pose actual provista como la transformación **map** → **base_link** (${}^{map}T_R$) como *feedback* del método.

   c. Redefinir la pose objetivo actual si se considera que el robot se encuentra lo suficientemente cerca (*Pursuit-Based goal selection*).

3 Seguimiento de trayectoria utilizando EKF

Integrar la información obtenida del modelo cinemático para la estimación de la pose puede llevar a la acumulación de error. Esto se debe a un número de asunciones y simplificaciones que el modelo realiza, factores externos como el rozamiento con el piso pueden llegar a invalidar el modelo por completo. Para lidiar con estos problema es posible utilizar información conocida del entorno, es decir, un mapa de referencias. Estas referencias proveen nueva información valiosa que es posible integrar y así mejorar la estimación de la pose.

A partir del Filtro de Kalman Extendido, se requiere un método que permita obtener una estimación robusta de la pose del robot. Para esto se deberá trabajar con la información provista por el sensor LiDAR y definir modelos adecuados para la plataforma omnidireccional presentada.

1. Implementar un nodo de ROS2 que interprete la información provista por el sensor LiDAR, debe:

   a. Recibir los escaneos por el tópico `/robot/front_laser/scan` y detectar postes que se encuentren frente al robot por medio del método de clusterización.

   b. Publicar mensajes de tipo `robmovil_msgs/msg/LandmarkArray` por el tópico `/landmarks`. La información publicada de las referencias (*landmarks*) debe estar en relación al marco de coordenadas del robot.

2. Modelar el estado $\vec{x}$, las entradas de control $\vec{u}$, las mediciones $\vec{z}$, el ruido del actuador $\vec{w}$, el ruido del sensor $\vec{v}$, el modelo de movimiento $f(\vec{x}, \vec{u}, \vec{w})$, el modelo de sensado $h(\vec{x}, \vec{v})$ y los respectivos Jacobianos.

3. Proponer matrices de covarianza iniciales para el modelo de movimiento y de sensado de manera que reflejen una mayor incertidumbre al momento de predecir la pose. Se espera que en la etapa de corrección la información proveniente de los sensores sea considerada “más confiable”.

4. Implementar un nodo de ROS2 que aplique el modelo del filtro:

a. Considerar que el mapa de postes se encuentra en referencia al marco de coordenadas del mapa. Esto se debe a que, en esta oportunidad, el mapa no es construido a partir del primer sensado del láser (a diferencia a lo visto durante la cursada).

b. Utilizar la estimación odométrica publicada por el tópico `/robot/odometry` como entrada de control del método.

c. Se debe publicar una estimación refinada de la pose del robot a través de una transformación **map** $\rightarrow$ **base_link_ekf** ($^{map}T_{R_{ekf}}$).

5. Realizar el seguimiento de la trayectoria planteada anteriormente utilizando la estimación refinada de la pose y el método de lazo cerrado.

Presentación del trabajo

La presentación del trabajo práctico consiste de una estructura de paquetes que conformen la implementación de un sistema desarrollado en ROS2 que resuelva el problema propuesto y un informe que documente el trabajo.

El informe es evaluado y este deberá desarrollar los diferentes métodos involucrados en profundidad. Se deben presentar experimentos que verifiquen el desempeño, resultados y conclusiones.

A continuación se sugiere una organización del mismo:

- **Introducción**
  Se debe introducir el problema que se esta abarcando, dando motivaciones y aplicaciones del mismo. Enunciar el enfoque propuesto para su resolución.
- **Modelo Cinemático**
  Presentar las particularidades de la plataforma y motivaciones para la resolución de un modelo cinemático. Exponer los parámetros utilizados para la plataforma particular sobre la que se trabajo. Documentar la metodología utilizada para la conformación de la estimación odométrica. Plantear experimentos que permitan validar y evaluar la efectividad del modelo. Entre los experimentos a realizar se deben incluir gráficos de consignas de velocidades lineales y angular del robot junto a las correspondientes velocidades reales ejercidas por el robot, para su comparación. Asimismo, se debe incluir un gráfico que permita comparar la pose del robot estimada según odometría con la real informada por el simulador.
- **Control a lazo cerrado y seguimiento de trayectorias**
  Presentar claramente el controlador que se implementó, utilizando el típico diagrama para representar un sistema de control a lazo cerrado. Se piden realizar una serie de experimentos en los cuales se prueben distintos valores de las constantes del controlador proporcional. Para ello, presentar gráficos en los cuales se comparen los valores de la pose real y la deseada (el *setpoint*) del robot, variando alguna de estas dos en distintas repeticiones del experimento. Incluir los valores de las velocidades lineales y angular asignadas en cada momento durante el experimento. Analizar los comportamientos observados y justificar la elección de las constantes de control.
- **Localización basada en EKF**
  Detallar en forma clara los modelos asociados el filtro de EKF que se propusieron para abordar el problema planteado. Una vez realizada la implementación del sistema de localización, se deberán realizar una serie de experimentos en los cuales se ponga a prueba la precisión del método, en comparación con la localización basada únicamente en odometría. En esta etapa, el robot será movido en forma manual (publicando velocidades constantes) y se reportará la pose del robot a lo largo del tiempo contrastándola con la pose real (*ground-truth*).
- **Seguimiento de trayectorias a lazo cerrado utilizando localización basada en EKF**
  Explicar en forma detallada la implementación del método de seguimiento de trayectorias implementado. Presentar el comportamiento del sistema completo (seguimiento y localización con EKF) al seguir la trayectoria propuesta. En caso de necesitar volver a realizar un ajuste de las constantes de control para lograr un mejor seguimiento, presentar los experimentos realizados para realizar dichos cambios. Realizar un análisis de la trayectoria realizada efectivamente por el robot vs. la trayectoria requerida *ground-truth*. Presente imágenes del RViz2 mostrando el elipsoide de covarianza en distintas porciones del recorrido.
- **Sistema desarrollado**
  Detallar los aspectos de implementación generales, tales como los nodos de ROS2 utilizados/implementados, documentar interacciones entre tópicos y mensajes. Instrucciones para ejecutarlo.
- **Conclusiones**
  Presentar las conclusiones que pueden extraerse a partir del trabajo y de los experimentos realizados y los resultados que se obtuvieron. Proponer posibles soluciones a problemas que haya encontrado así como potenciales mejoras que pudieran incluirse.

Anexo

Comentarios sobre el paquete robmovil_ekf

La librería `EKFilter` requiere que se declaren las dimensiones de las matrices y vectores en la inicialización. Dadas las libertades de movimiento de la plataforma omnidireccional es necesario modificar el archivo `localizer_ekf.cpp` específicamente donde dice:

```
robmovil_ekf::LocalizerEKF::LocalizerEKF(void) : EKFilter(3, 2, 3, 2, 2)
```

Por:

```
robmovil_ekf::LocalizerEKF::LocalizerEKF(void) : EKFilter(3, 3, 3, 2, 2)
```

De esta manera se define que u es de dimensión 3 en el caso del modelo cinemático omnidireccional a diferencia del diferencial.

Comentarios sobre el paquete `sim_ros2_interface`

El desarrollo de este Trabajo Práctico requiere que el el paquete `sim_ros2_interface` tenga declaradas las estructuras de mensajes utilizadas durante el desarrollo de los talleres de la materia. A éstos se le agregan:

- `robmovil_msgs/msg/MultiEncoderTicks`
- `geometry_msgs/msg/PoseArray`