Taller 6: Modelo diferencial y odometría

Introducción a la Robótica Movil

April 17, 2018

1 Introdución

El objetivo de este taller es lograr una implementación del modelo cinemático diferencial que nos permita traducir requerimientos de velocidad lineal y angular en velocidad consigna para los motores y estimar la pose del robot en base a la información de los encoders provistos.
De la página de la materia pueden descargarse los **paquetes Catkin** que serán utilizados durante este taller.

1.1 Añadir paquetes al workspace Catkin

Los **paquetes Catkin** poseen una determinada estructura de directorios. Para añadir nuevos paquetes a un **workspace Catkin** ya creado, se requiere únicamente **copiar el paquete** en el directorio **/catkin_ws/src**.

Una vez copiados los paquetes, es posible compilarlos utilizando el comando:

```
$ catkin build
```

No olviden configurar las variables de entorno del **workspace** utilizando en la raíz del directorio:

```
$ source devel/setup.bash
```

NOTA: Recordar volver a compilar luego de cambiar el código de algún nodo contenido en un paquete.

1.2 roslaunch y universal teleop

roslaunch es una herramienta interna de **ROS** que permite ejecutar una configuración establecida de nodos descripta en un archivo de texto plano. Para cada taller, la materia incluye un archivo **.launch** que permite lanzar los nodos requeridos de manera ordenada.

Para este taller utilizaremos:

```
$ roslaunch modelo_diferencial modelo_diferencial.launch
```

Además de esto, la materia provee de un nodo de tele-operación universal. Este nodo de tele-operación es el encargado de convertir pulsaciones de teclas a comandos de velocidad para el robot. Tiene configurado valores de velocidad

seguros para la plataforma Pioneer 3-DX y responde a las teclas **w,a,s,d**. Para que los comandos de velocidad sean efectivamente publicados el nodo tiene un sistema de **override** de manera que hay que **mantener apretada la barra espaciadora**.

Ejercicio 1: Cinemática inversa

Se requiere calcular la velocidad de rotación de cada rueda a partir de un comando de velocidad. El nodo encargado de realizar esto se encuentra dentro del paquete **modelo_diferencial**, implementado en el archivo:
modelo_diferencial/src/pioneer_odometry.cpp.

Deberán completar el callback **on_velocity_cmd** el cual recibe un **Twist** y deberá publicar un mensaje de velocidad de rotación por cada rueda (std_msgs/Float64).

NOTA: Al comienzo del código existen constantes definidas referidas a la plataforma. Leer los comentarios en el código para la construcción de los mensajes.

a) Inicializar el entorno de simulación con **V-Rep** utilizando la escena provista en **modelo_diferencial/vrep/modelo_diferencial.ttt** (**revisar taller anterior**).
   Lanzar los nodos requeridos utilizando el comando **roslaunch** y comprobar que se condice el movimiento del Pioneer en la simulación con respecto a los comandos de velocidad publicados para las ruedas.

Ejercicio 2: Odometría

A partir de la información provista por los encoders, se requiere calcular la pose del robot en referencia a la **pose inicial** (momento cuando se inició el experimento) y la velocidad actual del robot. Para esto deben modificar el mismo nodo del paquete **modelo_diferencial** implementado en el archivo:
modelo_diferencial/src/pioneer_odometry.cpp.

Deberán completar el callback **on_encoder_ticks** el cual recibe un **EncoderTicks**. Para obtener los ticks referidos a la rueda izquierda utilizar:
encoder_msg.ticks_left.data y **encoder_msg.ticks_right.data**.

Las cuentas de encoder son acumulativas **y poseen signo**. Al avanzar, los ticks de encoder de una determinada rueda se incrementan mientras que al retroceder los ticks de encoder disminuyen (**notar que los ticks de encoder podrían ser negativos**).

NOTA: Para operar con π pueden utilizar **M_PI** en el código.

Deberán construir un mensaje de odometría y publicarlo por el tópico **/robot/odometry**. Tener en cuenta que el mensaje de odometría contiene tanto la pose del

robot como la velocidad del mismo. Leer con atención los comentarios en el código para la construcción de los mensajes.

a) Inicializar el entorno de simulación con **V-Rep** utilizando la escena provista en **modelo_diferencial/vrep/modelo_diferencial.ttt** (**revisar taller anterior**).
Utilizar el nodo de tele-operación para enviar comandos de velocidad al robot y utilizar el **RViz** para visualizar la odometría publicada.

b) La simulación provee información de **ground-truth** de manera de poder comparar la calidad de la odometría computada. ¿Que sucede cuando requerimos una velocidad únicamente angular al robot? ¿Cómo se comporta la odometría calculada en comparación a la información de **ground-truth**?

NOTA: El marco correspondiente al centro del robot calculado por su odometría se lo llama "base_link" mientras que el marco de referencia **ground-truth** se lo llama "base_link_gt".

c) Modificar las propiedades dinámicas del suelo de la simulación reduciendo la fricción que aplica sobre objetos:
Expandir objeto ResizableFloor → Doble click a "ResizableFloor element" → click en "show dynamic properties" → Edit → Cambiar los valores de fricción "antes y después de Bullet".
¿Que sucede con la odometría calculada en esta situación?