Taller 9: Percepción con láser e IMU

Introducción a la Robótica Movil

May 8, 2018

1 Introdución

En este taller se desarrollarán dos funcionalidades concretas que son requeridas para afrontar la localización de un robot móvil utilizando el esquema de *Extended Kalman Filter* que se resolverá en los siguientes talleres. Concretamente, se trabajará con dos sensores: un telémetro láser y una unidad inercial (IMU). De la IMU, utilizaremos únicamente la información provista por el **giróscopo**. El taller consiste en implementar diversos algoritmos utilizando los esqueletos de código indicados en cada ejercicio. Lea **atentamente** el enunciado y los comentarios del código esqueleto.

1.1 Material requerido

Para resolver el taller, se deberá descargar de la página de la materia un código esqueleto del paquete **imu_laser**. Descargar y descomprimir el mismo en el directorio **src** de su **workspace** de catkin. En este paquete se incluye el código, archivos roslaunch, configuraciones de RViz y escenas de V-Rep.

1.2 Configuraciones de RViz

Para facilitar la configuración del entorno de visualización, se proveen **archivos .rviz** los cuales pueden ser cargados desde la interfaz gráfica del **RViz**:

File → Open Config → Seleccionan el archivo .rviz correspondiente

Estos archivos de configuración añaden la visualización de tópicos pre definidos a la interfaz.

Ejercicio 1: Compensación de *bias* del giróscopo

En este ejercicio se busca realizar un pre-procesamiento de la información obtenida por un sensor giroscópico (velocidad angular) montado sobre un robot móvil, con el objetivo de eliminar su *bias*. Para ello, se debe primero adquirir una serie de muestras de los valores arrojados por el sensor y, a partir de las mismas, estimar el *bias*. Una vez completada la calibración (que ejecutará un tiempo determinado por el usuario), los posteriores sensados recibidos deberán ser corregidos para remover el *bias* correspondiente.

Deberán trabajar con los archivos **/imu_laser/imu_calibrator.cpp** y **/imu_laser/imu_calibrator.h** (donde encontrarán algunas variables globales definidas). Se utilizará el archivo **/imu_laser/launch/imu.launch** para la ejecución de los nodos requeridos:

```
roslaunch imu_laser imu.launch
```

Tienen la posibilidad de configurar el tiempo de calibración modificando el parámetro:

```
<param name="calibrate" type="int" value="0"/>
```

El cual controla la **cantidad de segundos** que se deberán acumular mediciones. Pasado el tiempo de calibración configurado el nodo llama al método:

```
void calculate_bias(const ros::TimerEvent& event)
```

El cual deberán **completar** para calcular efectivamente el *bias*.

Las **mediciones de la IMU** son recibidas de manera continua por medio del método:

```
void on_imu_measurement(const sensor_msgs::Imu& msg)
```

El cual deberán **completar** para acumular mediciones durante el tiempo de calibración y realizar la **integración de las mediciones** corregidas.

Se pide implementar:

- La estimación del bias y la posterior compensación de los valores recibidos.
- El cálculo de la orientación del robot a partir de integrar en el tiempo, las velocidades ya compensadas. Verificar que se elimina el *bias* y no se observa *drift* (deriva) significativo.

Nota: debido a que estamos trabajando con un robot que se mueve en el plano, solo **necesitaremos la velocidad angular sobre el eje Z**, es decir el ángulo *yaw*.

Para verificar el funcionamiento correcto, utilice la configuración de **RViz** ubicada en **/imu_laser/launch/imu.rviz**. La escena a utilizar del V-Rep se encuentra en **/imu_laser/vrep/imu.ttt**.

En RViz podrán observar un marco de referencia que representa la **orientación estimada** del robot.

- ¿Que comportamiento es posible visualizar si se integran las mediciones **sin calibrar** la IMU? ¿Al calibrar, el comportamiento "no deseado" desaparece por completo?
- Pueden enviar comandos de velocidad al robot para corroborar que las estimaciones se condicen con la simulación:

```
rostopic pub /robot/cmd_vel geometry_msgs/Twist '[0,0,0]' '[0,0,-0.1]'
```

Ejercicio 2: Detección de landmarks

El futuro sistema de localización a realizar utilizará el sensor láser para detectar objetos especialmente colocados en la escena, que cumplirán el rol de *landmarks* o referencias. En este ejercicio se pide procesar los datos crudos de dicho sensor y realizar la detección de estos *landmarks*.
En el escenario a resolver, se sabe que existen "postes" (objetos cilíndricos que pueden ser detectados por el láser) de 10 cm de diámetro y que están fijos. Debido a que el láser devuelve muchas mediciones de distancia sobre el plano por cada sensado, un poste puede ser detectado por uno o más rayos del sensor. Por otro lado, el sistema de localización solo debe conocer de landmarks, identificados como una posición relativa al robot.

Deberán trabajar con el archivo **/imu_laser/src/landmark_detector.cpp**, las mediciones de sensado del láser se reciben a través del método:

```
void on_laser_scan(const sensor_msgs::LaserScanConstPtr& msg)
```

El cual deberán **completar** para establecer **centroides** que representen la posición de los postes. Se pide, entonces:

- Filtrar mediciones invalidas utilizando la información provista en el mensaje (ángulos y rangos validos). Pasar la información de los mensajes a **coordenadas cartesianas**.
- Agrupar las mediciones por cercanía teniendo en cuenta las dimensiones conocidas de los postes a detectar (utilizar coordenadas cartesianas y distancia Euclídea).
- Establecer centroides que representen los postes en **relación al robot**.
- Se debe publicar un mensaje de tipo `robmovil_msgs::LandmkarkArray`, que consiste de un arreglo de mensajes de tipo `robmovil_msgs::Landmark`. Estos elementos describen la posición de un landmark en forma relativa al robot en **coordenadas polares**.
- Implementar un algoritmo de detección de postes a partir de procesar un sensado laser. Tenga en cuenta las dimensiones conocidas de los postes a detectar. Ante cada sensado, se debe publicar un mensaje de tipo `robmovil_msgs::LandmkarkArray`, que consiste de un arreglo de mensajes de tipo `robmovil_msgs::Landmark`. Estos elementos describen la posición de un landmark en forma relativa al robot en **coordenadas polares** (deberán convertir los centroides calculados a coordenadas polares)

Para resolver el ejercicio, utilizar la escena de V-Rep **/imu_laser/vrep/laser_landmarks.ttt** y la configuración de RViz de **/imu_laser/launch/landmark_detector.rviz**.
Para lanzar el nodo de detección, ejecutar:

```
roslaunch imu_laser landmark_detector.launch
```