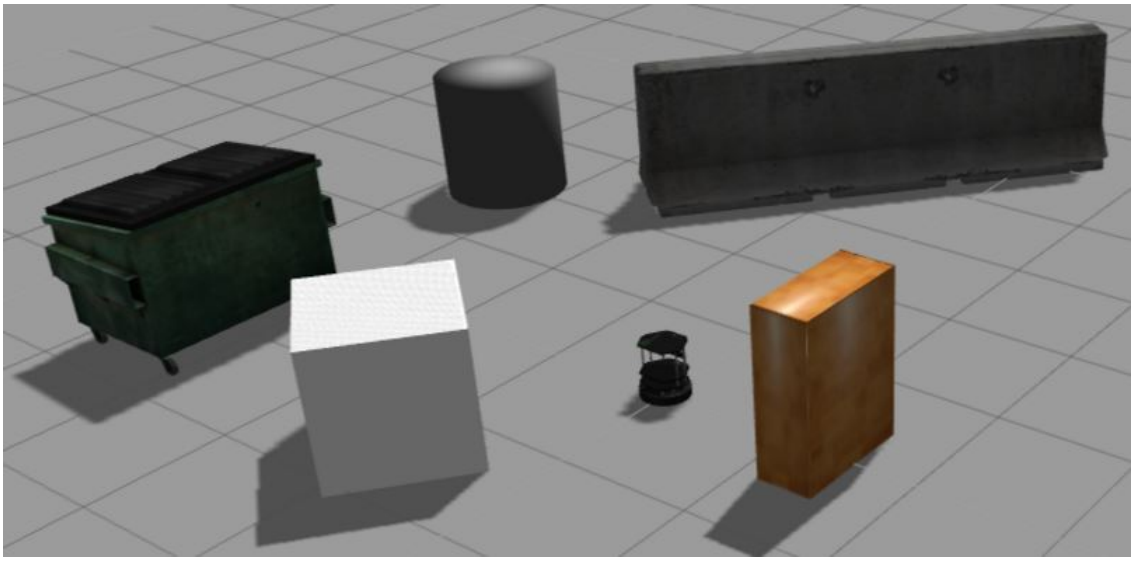
Curso 2016-2017
Grado en Ingeniería Informática
Universidad de Granada

Documentación práctica 1 TSI: Robot deambulador con ROS y Gazebo

Carlos Manuel Sequí Sánchez

17 de marzo de 2017



Índice

1. Primeros pasos

Lo primero que hice de todo para la realización de la práctica fue crear un nuevo paquete ROS denominado random_walk tal y como se indica en el guión. Tras haber solucionado un par de problemas que me surgieron con este tema, comencé analizando el código stopper.cpp y stopper.h, dado que basándonos en él prácticamente está montado el esqueleto del código a realizar.

Una vez analizado y comprendido el código proporcionado, la idea de resolución era sencilla: el robot anda hacia alante y, cuando encuentra un obstáculo (cosa que ya estaba implementada) simplemente calcula un número aleatorio analiza hacia donde hay una menor cantidad de obstáculos, gira hacia allí y reanuda la marcha (en caso de no haber más obstáculos).

2. Cambios realizados sobre el código stopper.cpp

Lo primero que he modificado del código ha sido el método startMoving, el cual solo tenia en cuenta la acción de ir hacia alante con el booleano "keepMoving". Simplemente he añadido un condicional if-else para que, en caso de que no pueda avanzar, gire hacia algún lado.

Tras ello me fuí al método scanCallBack y realicé los cambios para el mismo propósito, es decir, he incorporado un if-else para que una vez que (dentro de ese método) se ha comprobado que hay un obstáculo enfrente del turtlebot con el escaneo del laserscan, entonces llame a un método creado por mí el cual analiza cual de los dos lados es más conveniente para realizar el giro (método ChooseTurningSide). En caso de no hbaer obstáculo, evidentemente continua su marcha en linea recta.

Por último, como modificación también puede decirse que he añadido unos cuantos atributos de estado (booleanos) a la clase, por lo que los he inicializado en el constructor.

3. Métodos añadidos a la clase

A continuación paso a explicar los únicos dos métodos que he visto conveniente agregar a la clase:

- **ChooseTurningSide(escaneo_turtlebot, minIndex, maxIndex)**:
  - Recibimos como parámetros el escáner del turtlebot y los índices mínimo y máximo del sensor de proximidad.
  - En un solo bucle "for" condensamos dos análisis:
    - Cantidad de obstáculos que hay a la derecha, utilizando el vector "ranges" el cual nos dice si hay objeto o no a la distancia asignada al sensor turtlebot. Para analizar la parte derecha recorremos el vector desde minIndex hasta maxIndex/2.
    - Cantidad de obstáculos que hay a la izquierda, de la misma manera que para los de la derecha, pero esta vez recorriendo el vector desde maxIndex/2 hasta el final.
  - Una vez tenemos en 2 variables cuantos objetos hay a la izquierda y cuantos a la derecha, simplemente activamos una variable de estado "leftTurning" en caso de que haya más obstáculos a la derecha, o activamos "rightTurning" en el caso contrario.
- **Turning()**
  - Lo primero que hago es calcular un valor aleatorio entre 5 y 25, el cual hará que el robot gire durante una cantidad de tiempo aleatoria ("randTurningValue").
  - En función de la variable de estado activada en el método anterior, la velocidad angular será negativa (para un giro hacia la derecha) o positiva (para un giro hacia la izquierda).
  - Como último, tras conocer hacia donde conviene girar, solo nos queda realizar el giro en sí, cosa de la que se encarga el publisher con el mensaje:
    msg.angular.z = «sentido_giro» * «cantidad_aleatoria»

Una vez realizado dicho giro hasta donde no se hayan obstáculos y, teniendo en cuenta todas las variables de estado pertinentes, el turtlebot continua su deambulación por el mapa.
Por último he añadido una línea al fichero launch con el fin de indicar desde ahí la distancia a la que turtlebot ha de detectar si hay o no obstáculos delante suya. Esta es la línea añadida:
<node name="walker" pkg="random_walk" type="walker" args="0.8" output="screen"/>
Con el parámetro "args" le indico que pasamos al programa como argumento el valor 0.8, para así recogerlo en el main con arg[1] y usarlo como distancia mínima de proximidad a un objeto.