Práctica 1: Técnicas de los Sistemas Inteligentes

Néstor Rodríguez Vico. Grupo B2.

Para resolver este problema me he basado en el código que se nos proporcionó para probar Gazebo, *Stopper.cpp*, *Stopper.h* y *run_stopper.cpp*. En dicho código el robot avanzaba hasta que detectaba un objeto en su camino. Basándonos en esta idea he realizado la práctica.

Lo primero que hace el programa es crear un nodo *Publisher* y un nodo *Subscriber*. Con el primero de ellos le enviamos la información acerca del movimiento a nuestro robot *turtle_bot* y con el segundo leemos la información que genera nuestro robot. En nuestro caso nos interesa obtener información acerca del sensor de proximidad que tiene incorporado nuestro robot para poder procesarla y tomar una decisión en función de la misma. El procesamiento de la información la hacemos en la función *callBack*. La información viene dada en forma de vector (*scan->ranges*) en el que cada componente representa la distancia al objeto que detecta dicho rayo. Debemos tener en cuenta que pueden aparecer valores "basura" (*nan*) cuando el objeto es muy lejano o muy cercano. La idea es doble, primero calcular si hay algún objeto a menor distancia que la menor distancia permitida para nuestro robot, (variable *MIN_DIST_FROM_OBSTACLE* en nuestro código), y calcular que rayo ha detectado el objeto más lejano para saber hacia donde debemos girar. Una vez se haya calculado ambas cosas en la función *callBack*, si hay algún objeto más cercano de lo que debería, detenemos el robot y realizamos el giro. Una vez terminado, seguimos avanzando por el mapa. Un pequeño psudocódigo sería el siguiente:

```
Para cada rayo en scan->ranges:
   menor_distancia = infinito, mayor_distancia = 0, posicion_mas_lejano = 0
   Si scan->ranges[i] < menor_distancia
      menor_distancia = scan->ranges[i]
   Fin Si
   Si  scan->ranges[i] < 10 //Para evitar valores nan
      Si  scan->ranges[i] > mayor_distancia
         mayor_distancia =  scan->ranges[i]
         posicion_mas_lejano = i
      Fin Si
   Fin Si
Fin Para
```

Casi de forma paralela a la ejecución de la función *callBack* se está ejecutando la función que mueve nuestro robot, *startMoving*. En esta función hay un bucle con una toma de decisión, ya sea girar o avanzar. En el caso de que haya un objeto a menor distancia de la establecida se llama a la función *girar* y en caso contrario a la función *moveForward* (la cual se encarga de avanzar el robot). La función girar no recibe ningún parámetro pero el proceso de rotación lo realiza en función de varios datos:

- Posición del rayo que ha detectado el objeto más lejano: Se usa para poder calcular hasta cuando debe girar el robot.
- Cantidad de ángulos (rayos láseres): Con este dato podemos calcular si debemos girar en un sentido u otro, según la posición del rayo esté en la primera o en la segunda mitad del escaner, es decir, sea menor o mayor que la mitad de la cantidad de ángulos.
- Incremento de ángulo: Este dato representa el ángulo que ha escaneado cada rayo láser. Sabiendo la posición del rayo láser que ha detectado el objeto más lejano y este ángulo es fácil calcular el ángulo que debe girar nuestro robot. Dicho ángulo se calcula como: Incremento de ángulo * posición del rayo.

Una vez tenemos todo lo necesario para girar, giramos. Esto se hace publicando un mensaje de tipo *Twist* indicando la velocidad de giro de nuestro robot en el campo *z* de la velocidad angular. Suponiendo que msg es el mensaje que se va a publicar sería lo siguiente: *msg.angular.z = velocidad.* Para controlar que se realiza el giro necesario sólo tenemos que controlar el tiempo durante el cual se está girando, ya que de la fórmula *velocidad = espacio / tiempo* sabemos la *velocidad* y podemos restringir el *tiempo* para que gire el *espacio* que deseemos.

Finalmente, creamos el fichero *random_walk.launch* para lanzar el simulador de Gazebo y nuestro paquete *random_walk.* También se ha añadido un parámetro que permite modificar la distancia mínima (variable *MIN_DIST_FROM_OBSTACLE* en nuestro código) a la que se puede acercar el robot a un objeto del mapa y luego ese parámetro se lee desde el programa para ser usado por nuestro robot.