Universidad de Alcalá

Práctica 1

Mapeado y Localización

1. PREPARACIÓN

Para la realización de esta práctica es necesario instalar previamente los siguientes paquetes, que en el caso de utilizar la máquina virtual de la asignatura, ya se encuentran preinstalados:

`sudo apt-get install ros-kinetic-bfl` → Instala los paquetes necesarios para los filtros bayesianos

`sudo apt-get install ros-kinetic-teleop-twist-keyboard` → Instala un programa auxiliar para controlar el robot con un teclado (ya utilizado en la guía rápida)

Además será necesario configurar los aspectos siguientes:

1.1. Incorporación de ruido en la odometría del simulador STDR

El simulador STDR no es realista en cuanto a la simulación de los datos de la odometría del robot, por dos motivos:

1. El valor de la posición está referido al marco de referencia del mapa, por lo que en todo momento toma su valor absoluto respecto al mismo. Esto no es realista respecto al funcionamiento real de la odometría, que siempre comienza en el valor (0,0,0).

2. No incorpora ruido, cuando la presencia de un ruido acumulativo es una de las características más destacables del funcionamiento real de un sistema odométrico.

El resultado de estas dos simplificaciones es que en el topic */robot0/odom* leeremos la posición absoluta del robot respecto al mapa sin ningún tipo de error (como si se tratara de un sensor GPS ideal).

Los algoritmos que se van a implementar en esta práctica requieren una simulación más realista de la odometría, que proporcione un valor de posición relativo a la posición inicial del robot, y con un error acumulativo.

Para conseguir esto, se van a hacer algunas modificaciones en el simulador, siguiendo los siguientes pasos:

1. En primer lugar se va a "redirigir" el árbol de transformadas creado por el simulador en el topic */tf* a otro topic que no utilizaremos y llamaremos */tf_sim*. De esta manera crearemos después (en el paso 2) de nuevo el árbol de transformadas en */tf* de manera más realista, que coincida con la información del robot real.

   Para ello, editaremos el fichero *.launch* del simulador y en todos los nodos haremos un *<remap from="tf" to="tf_sim"/>* del siguiente modo:

```
<launch>

        <include file="$(find stdr_robot)/launch/robot_manager.launch" />

        <node type="stdr_server_node" pkg="stdr_server" name="stdr_server" output="screen" args="$(find
stdr_resources)/maps/simple_rooms.yaml">
                <remap from="tf" to="tf_sim"/>
        </node>

        <node pkg="tf" type="static_transform_publisher" name="world2map" args="0 0 0 0 0 0  world map 100" >
                <remap from="tf" to="tf_sim"/>
        </node>

        <include file="$(find stdr_gui)/launch/stdr_gui.launch"/>

        <node pkg="stdr_robot" type="robot_handler" name="$(anon robot_spawn)" args="add $(find stdr_resources)/
resources/robots/amigobot.xml 2 2 0" >
                <remap from="tf" to="tf_sim"/>
        </node>

</launch>
```

Es importante editar también los ficheros que están incluidos en este *launch* (mediante la instrucción *include*) y realizar la misma operación.

2. Instalar el paquete *aux_files* (creado por los profesores de la asignatura) que se encarga de modificar el valor de la odometría para que sea local, añadirle un ruido acumulativo, y publicar el resultado en dos topics:

   - */robot0/local_odom* -> éste será el topic al que deberemos suscribirnos a partir de ahora para leer la odometría del robot, ya que está modificada para ser más realista.
   - */tf* -> en este topic se generan de forma apropiada los sistemas de referencia de manera que coindidan con los del robot real.

   Para instalar el paquete, es necesario bajarlo de la página de la asignatura, descomprimirlo dentro de la carpeta */src*, y recompilar el espacio de trabajo ejecutando *catkin_make*.

   Una vez realizado esto, y con el simulador ya abierto, hay que ejecutar:

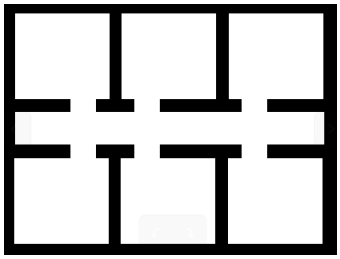
   roslaunch aux_files aux_files.launch

   para que se generen los topics antes mencionados (NOTA: si se desea, se puede incluir este fichero launch dentro del que lanza el simulador).

   Por último, para modificar el ruido de la odometría puede editarse el fichero *aux_files.launch* y cambiar el valor de los parámetros *error_x*, *error_y* y *error_a*. Por defecto dichos parámetros toman un error similar al de la odometría del Amigobot.

1.2. Configuración del mapa del simulador para la realización de las prácticas

Para las prácticas se puede cargar cualquier mapa de entorno en el simulador (editar para ello el fichero *.launch*), pero para las pruebas iniciales se recomienda utilizar el mapa *simple_rooms.yaml*:



MAPEADO

Se van a utilizar dos técnicas de mapeado para obtener una mapa de ocupación del entorno del robot utilizando la información del láser y de la odometría.

2.1. Mapeado con posiciones conocidas

En primer lugar se va a probar una técnica de mapeado puro, en la que se supone conocida la posición del robot sin error. Esta técnica consiste básicamente en convertir las medidas de distancia de los diferentes haces del láser en coordenadas cartesianas relativas al robot, e insertar dichos obstáculos en una rejilla de ocupación respecto a la posición actual del robot.

Para desarrollar este método se tomará como punto de partida el script de Matlab *MappingWithKnownPoses_2019_20.mlx* disponible en el apartado de Software de la asignatura en Blackboard. Este script proporciona un esqueleto y parte del código ya implementado. Deben completarse las partes indicadas con puntos suspensivos (…).

Se realizarán las adaptaciones necesarias sobre este script para poder ejecutarlo tanto con el simulador STDR como con el robot real Amigobot. En cualquiera de los dos casos, primero deberán ejecutarse el script de conexión y el de inicialización ya desarrollados en la práctica 0 (guía rápida de ROS). Los scripts finales deberán nombrarse del siguiente modo:

Script para ejecutar con el simulador: *MappingWithKnownPoses_Simulator.m*
Script para ejecutar con el robot real: *MappingWithKnownPoses_RealRobot.m*

NOTA: es importante respetar los nombres de los ficheros para la entrega de las prácticas.

PASOS Y PRUEBAS A REALIZAR:

1. Investigue sobre la clase *robotics.OccupancyGrid* definida en la Robotics System Toolbox para almacenar rejillas de ocupación. Describa sus propiedades y métodos más importantes.

2. Adapte el script inicial *MappingWithKnownPoses_2019_20.mlx* para poder utilizarlo con el simulador STDR ejecutándose en la máquina virtual (renombrar como *MappingWithKnownPoses_Simulator.m*). La teleoperación del robot por el entorno para ir obteniendo el mapa se realizará mediante el nodo de ROS *teleop_twist_keyboard.py* dentro de la máquina virtual. Realice a continuación las siguientes pruebas:

    - Obtenga del mapa del entorno configurando un error nulo para la odometría en el simulador STDR.

    - Obtenga del mapa de entorno configurando un error no nulo (el que viene por defecto en el nodo *aux_files*) para la odometría en el simulador.

    - Obtenga conclusiones sobre los resultados anteriores.

3. Adapte el script de nuevo para poder utilizarlo con el robot real (renombrar como *MappingWithKnownPoses_RealRobot.m*). La teleoperación del robot por el entorno para ir

obteniendo el mapa se realizará mediante el nodo de ROS *teleop_twist_keyboard.py* dentro de la máquina virtual. Pruebas a realizar:

- Obtenga un mapa de entorno correspondiente al pasillo exterior al laboratorio.
- Obtenga conclusiones sobre los resultados obtenidos.

2.2. Localización y mapeado simultáneos (SLAM)

A continuación se va a implementar una técnica de SLAM basada en las medidas del láser utilizando una técnica de optimización del grafo de posiciones (*pose graph optimization*). Esta técnica permite corregir tanto el mapa como la trayectoria reconstruida sobre él cada vez que se detecta un cierre de lazo (el robot "revisita" una zona por la que ya ha pasado previamente). De esta manera se compensan los errores de la odometría en la creación del mapa y la reconstrucción de la trayectoria.

Para desarrollar este método se tomará como punto de partida el script de Matlab *OnlineSLAM_2019_20.mlx* disponible en el apartado de Software de la asignatura en Blackboard. Este script proporciona un esqueleto y parte del código ya implementado. Deben completarse las partes indicadas con puntos suspensivos (…).

Se realizarán las adaptaciones necesarias sobre este script para poder ejecutarlo tanto con el simulador STDR como con el robot real Amigobot. En cualquiera de los dos casos, primero deberán ejecutarse el script de conexión y el de inicialización ya desarrollados en la práctica 0 (guía rápida de ROS). Los scripts finales deberán nombrarse del siguiente modo:

Script para ejecutar con el simulador: *OnlineSLAM_Simulator.m*
Script para ejecutar con el robot real: *OnlineSLAM_RealRobot.m*

NOTA: es importante respetar los nombres de los ficheros para la entrega de las prácticas.

PASOS Y PRUEBAS A REALIZAR:

1. Investigue sobre la clase *robotics.LidarSLAM* definida en la Robotics System Toolbox para realizar localización y mapeado simultáneos mediante medidas de láser. Describa sus propiedades y métodos más importantes.

2. Adapte el script de ejemplo para poder utilizarlo con el simulador STDR ejecutándose en la máquina virtual (renombrar como *OnlineSLAM_Simulator.m*). La teleoperación del robot por el entorno para ir obteniendo el mapa se realizará mediante el nodo de ROS *teleop_twist_keyboard.py* dentro de la máquina virtual. Incluya error de odometría en el simulador. Cree el mapa probando diferentes estrategias y obteniendo conclusiones sobre la calidad del mapa obtenido, por ejemplo en los siguientes casos:

   - Recorriendo varias veces por zonas o recorriendo el mapa de principio a fin antes de volver a visitar el mismo sitio.

3. Repita todas las configuraciones y pruebas utilizando el robot real. En este caso, el nombre del fichero .m debe ser "*OnlineSLAM_RealRobot.m*". Guarde el mapa generado para poder utilizarlo en el apartado de localización.

3. LOCALIZACIÓN CON "AMCL"

La Robotics System Toolbox incluye la clase *robotics.MonteCarloLocalization* que permite localizar un robot utilizando datos del láser y la odometría sobre un mapa conocido a priori. Este mapa puede haber sido obtenido experimentalmente.

Para desarrollar este método se tomará como punto de partida el script de Matlab *AMCL_Localization_2019_20.mlx* disponible en el apartado de Software de la asignatura en Blackboard. Este script proporciona un esqueleto y parte del código ya implementado. Deben completarse las partes indicadas con puntos suspensivos (…).

Se realizarán las adaptaciones necesarias sobre este script para poder ejecutarlo tanto con el simulador STDR como con el robot real Amigobot. En cualquiera de los dos casos, primero deberán ejecutarse el script de conexión y el de inicialización ya desarrollados en la práctica 0 (guía rápida de ROS). Los scripts finales deberán nombrarse del siguiente modo:

Script para ejecutar con el simulador: *AMCL_Localization_Simulator.m*
Script para ejecutar con el robot real: *AMCL_Localization_RealRobot.m*

NOTA: es importante respetar los nombres de los ficheros para la entrega de las prácticas.

PASOS Y PRUEBAS A REALIZAR:

1. Investigue sobre la clase *robotics.MonteCarloLocalization*. Describa sus propiedades y métodos más importantes. Identifique la forma utilizada para definir los modelos de actuación y de observación.

2. Adapte el script de ejemplo para poder utilizarlo con el simulador STDR ejecutándose en la máquina virtual (renombrar como *MonteCarloLocalization_Simulator.m*). La teleoperación del robot por el entorno se realizará mediante el nodo de ROS *teleop_twist_keyboard.py* dentro de la máquina virtual. Realice a continuación las siguientes pruebas:

    - Configure el localizador de manera que la posición inicial sea conocida con cierto error, para realizar un seguimiento de la trayectoria realizada por el robot (localización local). Realice pruebas de localización en este caso.
    - Configure el localizador de manera que la posición inicial sea desconocida (localización global). Realice pruebas de localización en este caso. ¿Consigue el robot localizarse en todos los casos? Comente los resultados.

    - Obtenga conclusiones sobre los resultados anteriores.

3. Adapte el script de ejemplo para poder utilizarlo con el robot real (renombrar como *MonteCarloLocalization_RealRobot.m*). Realice las mimas pruebas que en el caso anterior y obtenga conclusiones a partir de los resultados.