SIMULTANEOUS LOCALIZATION AND MAPPING (SLAM) UTILIZANDO PAQUETES DEL AMBIENTE ROBOT OPERATING SYSTEM (ROS)

Gabriel Omar Flores Aquino1
gfloresa0500@alumno.ipn.mx Rogelio Ernesto
García Chávez22
Edson Marín Raudales22
Ramón Silva Ortigoza22

1 Instituto Politécnico Nacional UPIITA.
2 Instituto Politécnico Nacional CIDETEC Área de
Mecatrónica.

Boletín No. 88
1o. de enero de 2022

Resumen

En la actualidad la navegación autónoma sigue siendo uno de los grandes retos de la robótica, es claro que son casi inexistentes las tareas que un robot puede o debe realizar sin necesidad de moverse o cambiar su configuración. En el ámbito de la robótica móvil, las técnicas de navegación basada en mapas se mantienen como uno de los enfoques más utilizados, debido a su factibilidad, pero hace necesario el conocimiento de un mapa de navegación esto puede resolverse utilizando técnicas de localización y mapeo, conocidas por sus siglas en inglés como SLAM. Estas técnicas permiten conocer y ubicar al robot en su entorno al mismo tiempo que se desplaza. En el presente reporte se describe los pasos necesarios para generar un mapa a través de técnicas de SLAM, utilizando un sensor laser y las herramientas provistas por el meta-sistema operativo ROS el cual en años recientes se ha convertido en un estándar en el desarrollo de software de robótica.

1. Introducción

En la actualidad ROS es una de las herramientas más extendidas para desarrollar software de robots debido a su capacidad de modularidad permite reutilizar el software desarrollado previamente, con este enfoque se puede lograr que los robots resuelvan tareas complejas dividiendo el trabajo en tareas más sencillas y tratables con bloque de código denominados nodos. Es además ampliamente utilizado tanto en el ámbito académico y de investigación como en la industria, sin embargo, el uso de este meta sistema operativo requiere una curva de aprendizaje extendida debido al extenso contenido de su paquetería básica además de requerir un manejo fluido de Linux.

Entre los paquetes que forman parte del núcleo de ROS se encuentra los paquetes referentes a la pila de navegación, la cual implementa algoritmos clásicos para la localización y mapeo del ambiente también conocidos por sus siglas en inglés como SLAM. El manejo e implementación de técnicas de SLAM son el primer paso para desarrollar algoritmos más complejos de navegación basada en mapas. En este tutorial se explora el uso y aplicación práctica de uno de los paquetes que implementa un algoritmo clásico de SLAM para generar mapas de ocupación. Para las pruebas realizadas se utiliza un robot de configuración tipo carro denominado, robot modelcar, Figura 1. El cual cuenta con un sensor LIDAR de un solo haz de luz, el cual es un sensor de distancia que se puede utilizar para reconstruir ambientes bidimensionales. Es importante remarcar que, aunque lo planteado en este documento se

prueba en el robot modelcar debido a la capacidad de modularidad de ROS todos los pasos pueden ser reproducidos en el mismo modo con cualquier robot que utilice ROS y cuente con un sensor LIDAR.

Figura 1. Robot modelcar v1.

2. Hardware necesario

Para generar un mapa bidimensional necesitamos los datos generados por un sensor lidar como es el sensor de bajo costo RPLIDAR modelo A1 de RoboPeack que se muestra en la Figura 2.

Figura 2. Sensor lidar.

3. Instalar paquetes de ROS necesarios

El paquete *gmapping* no es parte de la distribución básica de ROS por esta razón hace falta instalarlo, esto se puede hacer desde consola con el siguiente comando

sudo apt-get install.

Para guardar los mapas generados es necesario instalar el paquete *map server* , esto puede realizarse ejecutando el siguiente comando.

sudo apt-get install ros-indigo-map-server

4. Generar datos

4.1. Publicar marcos de transformación

Para que el paquete funcione deben de existir las transformaciones adecuadas entre los marcos de los distintos componentes, nuestro objetivo es generar un mapa a partir de la información tomada por el sensor lidar, los datos del sensor están referenciados al marco del lidar denominado laser, este marco de referencia permanece fijo respecto al robot cuyo marco de referencia es base link, base link que es un marco fijo a la base del robot pero móvil respecto al marco global denominado **odom** , dado que el mapa debe ser generado respecto al marco global **odom** entonces necesitamos que cada dato tomado por el lidar este referenciado a**odom**, esta cadena de transformación **laser** $\rightarrow$ **base link** $\rightarrow$ **odom** es calculada por ROS, solo hace falta publicar **frame id y child frame id** de cada componente.

Para este robot el nodo **odometry publisher** del paquete **Odom** , publica la cadena **base link** $\rightarrow$ **odom** por lo que solo hace falta publicar la cadena **laser** $\rightarrow$ **base link** para esto utilizamos el comando.

static transform publisher $<$ **x y z yaw pitch roll frame id child frame id period in ms**$>$

Esta instrucción pública la transformación del marco laser sobre base link. Usualmente los mapas son generados a partir de una bolsa (rosbag) para después ser utilizados, en este caso el comando **static transform publisher** debe ser ejecutada desde la consola del robot de otro modo la estampa de tiempo no coincidirá con la grabada en la bolsa, utilizar un periodo de 100 ms es una buena idea la posición del marco laser respecto a base link es tomada directamente de las medidas físicas del robot.

rosrun tf static transform publisher 0.035 0.0 0.048 0 0 0 /base link /laser 100

Una imagen de la visualización de estos marcos de transformación usando el software integrado en ROS denominado rviz se puede observar el la Figura 3.

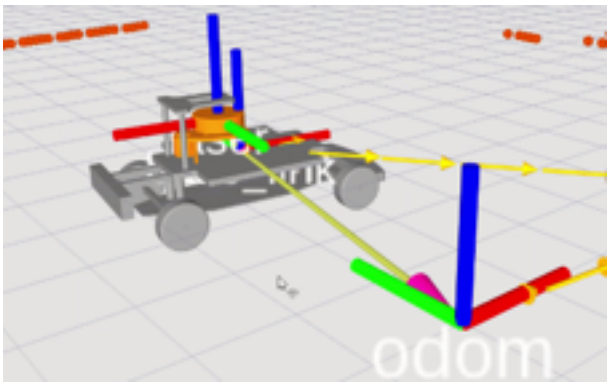

Figura 3. Relación entre los marcos coordenados.

A continuación, grabamos los datos con el comando de rosbag, los unicos topicos necesarios son **/laser y /tf.**

rosbag record –O parking_tf_tree.bag /scan /tf /odom /model_car/yaw

4.2. Herramientas graficas

Existen una serie de herramientas que nos permite verificar que la publicación de los marcos dentro de la bolsa sea la correcta, una de ellas es verificar el árbol de transformación ejecutando el comando **rosrun rqt tf tree rqt tf tree** , cuando la bolsa se esté ejecutando el resultado será como el de la Figura 4.

Figura 4. Árbol de transformaciones.

Otra herramienta útil para verificar el flujo de las transformaciones es ejecutar el comando tf monitor con el cual podemos ver si las transformaciones se conectan correctamente.

rosrun tf tf_monitor /laser /odom

La cadena entre padres e hijos es la correcta **laser** → **base link** → **odom.**

5. Crear un mapa

Ahora que hemos comprobado que todos los datos están correctamente grabados podemos generar un mapa utilizando el paquete *gmapping*

Ejecutar los comandos en el siguiente orden:

1. **roscore**
2. Habilitar el tiempo de simulación.
   rosparam set use_sim_time true
   Es importante recordar regresar este valor a false al finalizar de trabajar con bolsas.
3. Ejecutar *gmapping* con los siguientes parámetros.
   rosrun gmapping slam_gmapping scan:=scan _xmin:=-10 _ymin:=-10 _xmax:=10 _ymax:=10 _maxUrange:=5.5 _maxRange:=5.5 _minimumScore:=50 _linearUpdate:=0.2 _angularUpdate:=0.25 _temporalUpdate:=5.0 _delta:=0.025
   rosrun gmapping slam_gmapping scan:=scan _xmin:=-10 _ymin:=-10 _xmax:=10 _ymax:=10 _maxUrange:=5.5 _maxRange:=5.5 _minimumScore:=50 _linearUpdate:=0.2 _angularUpdate:=0.25 _temporalUpdate:=5.0 _delta:=0.025

Figura 5. Mapa resultante.

4. 4.-Ejecutar bolsa con el tiempo de simulación, Figura 6.

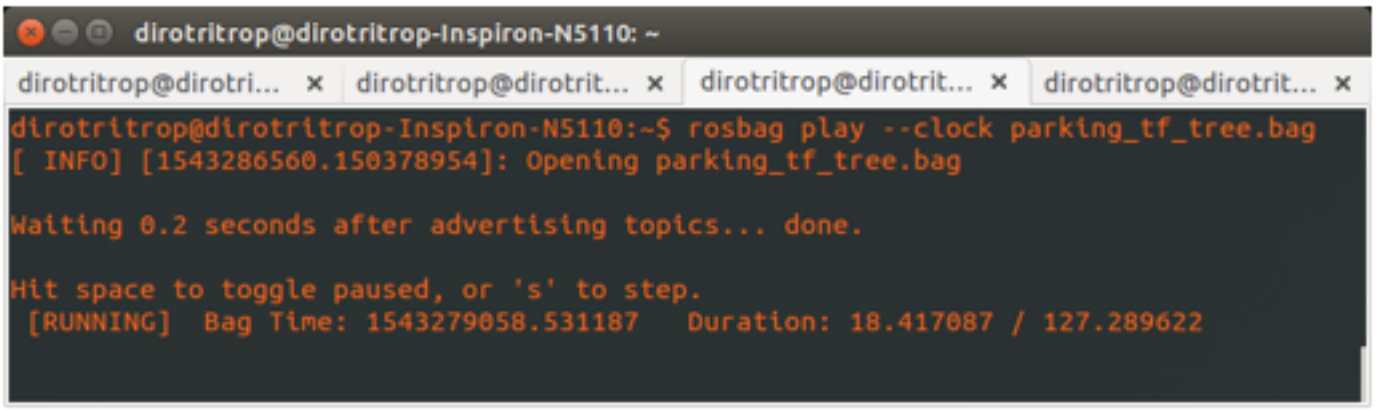

Figura 6. Ejecutar bolsa.

5. 5. Guardar el mapa

Para guardar el mapa generado se utiliza el paquete **map server** el cual nos guardara dos archivos **(.yaml .png)** en el directorio desde donde se ejecutó el comando, Figura 7.

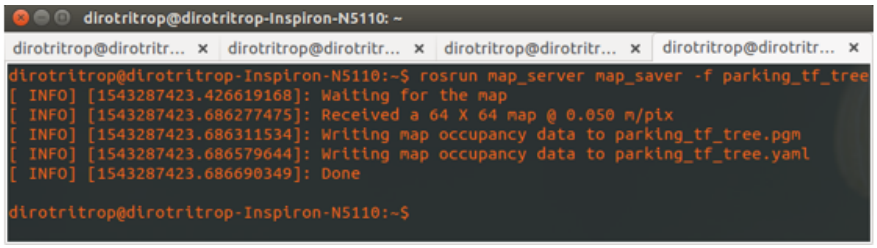

Figura 7. Guardar mapa.

3. Conclusión

En el presente trabajo se implementó un ejemplo detallado de la utilización del paquete *gmapping* para generar mapas de ocupación utilizando algoritmos de SLAM. Además, se detalló como se pueden utilizar conjuntamente con las bolsas de ROS y como se pueden monitorear a través de las aplicaciones incluidas en el meta sistema operativo.

Referencias

1. (**s.a.**). *ROS wiki.* Recuperado: Oct 12, 2021, de. http://wiki.ros.org/gmapping
2. (**s.a.**). *ROS wiki.* Recuperado: Oct 12, 2021, de. http://wiki.ros.org/slamgmapping/Tutorials/MappingFromLogged
3. (**s.a.**). *ROS wiki.* Recuperado: Oct 12, 2021, de. http://wiki.ros.org/tf/Tutorials
4. (**s.a.**). *ROS wiki.* Recuperado: Oct 12, 2021, de. http://wiki.ros.org/navigation/Tutorials/RobotSetup/TF
5. Autor (**s.a.**). *Geduino Foundation.* Recuperado: Oct 12, 2021, de http://geduino.blogspot.com/2015/04/gmapping-and-rplidar.html