

Map My World Robot ?

Miguel A. Colmenares Barboza

Abstract — El proyecto presentado en este documento se centran en implementar la localización y mapeo simultáneo (SLAM) utilizando dos sensores (Sensor Lidar RGB-D y cámara Kinect) instalados en un robot de dos ruedas. Los entornos que serán mapeados son dos mundos diferentes que se simulan en Gazebo. El robot transita cada entorno por separado y mapea el entorno utilizando el enfoque GraphSLAM de asignación basada en apariencia en tiempo real (RTAB-Map).

Index Terms —Robot, SLAM, RTAB-Map, Udacity, IEEEtran.

I. INTRODUCCIÓN

Para un robot móvil que transita una zona predefinida es imprescindible saber dónde está? y a donde debe ir?. Para resolver este problema de ubicación se implementa SLAM. Esta técnica resuelve el problema de localización para mapear el entorno y al mismo tiempo mapea el entorno para resolver el problema de localización. En este proyecto dos entornos 3D son simulados en Gazebo y mapeados en Rviz implementando SLAM con la ayuda de la herramienta RTAB-Map. El primer entorno se nombró “Comedor de cocina” y el segundo entorno se nombró “Bot_world”.

II. BACKGROUND.

SLAM a menudo se llama el problema del huevo o la gallina porque el mapa es necesario para la localización, y la posición del robot es necesaria para el mapeo, por lo tanto, es un problema desafiante. Hay muchos enfoques para realizar SLAM, pero hasta ahora, los dos enfoques más útiles para SLAM son FastSLAM basado en cuadrícula y GraphSLAM.

FastSLAM basado en cuadrícula.

El algoritmo FastSLAM utiliza un enfoque de filtro de partículas tradicional para resolver el problema de SLAM completo con correspondencias conocidas. Usando partículas, FastSLAM estima una posición anterior sobre el camino del robot junto con el mapa. Cada una de estas partículas contiene la trayectoria del robot que dará la ventaja a SLAM para resolver el problema del mapeo con poses conocidas. Además de la trayectoria del robot, cada partícula contiene un mapa y cada característica del mapa está representada por un gaussiano local. FastSLAM tiene una gran desventaja ya que siempre debe suponer que hay posiciones históricas conocidas, y por lo tanto con FastSLAM no podemos modelar un entorno arbitrario.

FastSLAM basado en cuadrícula amplía el algoritmo FastSLAM y soluciona el problema SLAM en términos de mapas de cuadrícula, por lo que ahora se puede resolver el problema SLAM en un entorno arbitrario. Específicamente, el algoritmo FastSLAM basado en la cuadrícula estima la trayectoria del robot utilizando el algoritmo de localización Monte Carlo (MCL). Entonces el algoritmo FastSLAM basado en cuadrícula estima el mapa asumiendo poses conocidas y utilizando el algoritmo de mapeo de la cuadrícula de ocupación.

GraphSLAM.

GraphSLAM es un algoritmo SLAM que resuelve el problema completo de SLAM. Este algoritmo recupera toda la ruta y el mapa, lo que le permite considerar las dependencias entre las poses actuales y anteriores. GraphSLAM tiene algunas ventajas sobre FastSLAM. GraphSLAM mejora la necesidad de capacidad de procesamiento a bordo, al tiempo que mejora la precisión sobre FastSLAM. Como GraphSLAM puede retener información de ubicaciones pasadas, es una ventaja sobre FastSLAM que utiliza menos información y tiene un número finito de partículas.

El algoritmo de mapeo basado en la apariencia en tiempo real es un enfoque GraphSLAM y se usará en este proyecto para realizar SLAM. Este algoritmo utiliza datos recopilados de sensores para localizar el robot y mapear el entorno. En RTAB-Map, un proceso llamado cierre de bucle se usa para permitir que el robot determine si la ubicación se ha observado anteriormente. Mientras el robot continúa atravesando su entorno, el mapa continúa creciendo. Para otros métodos basados en Apariencia, el robot continúa comparando imágenes nuevas con imágenes pasadas para identificar si ya ha estado en esa ubicación anteriormente. Esto, con el tiempo, aumenta el número de imágenes para comparación, lo que hace que el proceso de cierre del circuito tarde más, aumentando la complejidad proporcionalmente a lo largo del camino. Sin embargo, RTAB-Map está optimizado para SLAM a gran escala y de largo plazo, lo que permite que el cierre de bucle se procese lo suficientemente rápido como para ser conocido en tiempo real sin afectar de forma significativa el rendimiento.

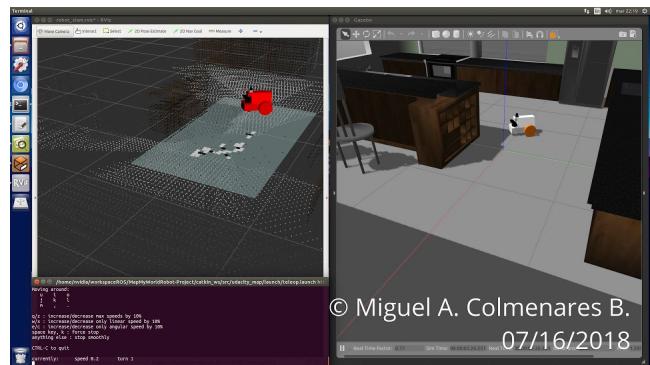


Fig 1. Robot simulado en un entorno gazebo implementando SLAM

III. CONFIGURACIÓN.

En este proyecto se desarrolló el modelo del robot y el modelo del entorno desde cero. Ambos elementos se construyeron, simularon y probaron en Gazebo.

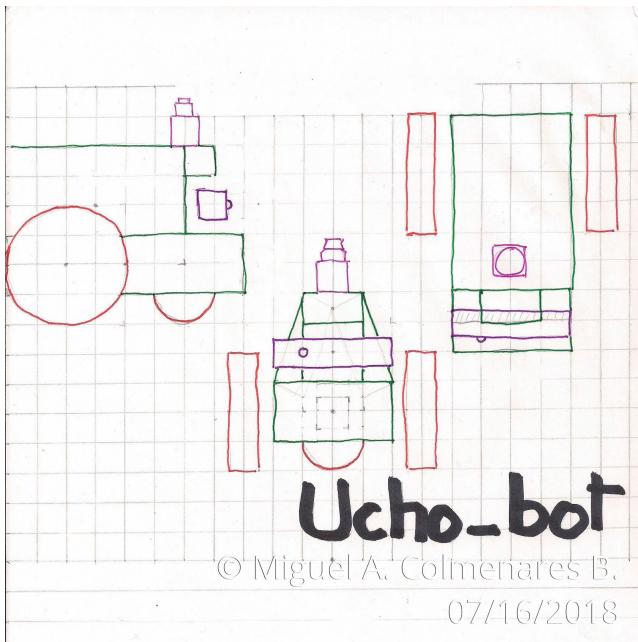


Fig 2. Boceto del Robot.

Configuración del Robot.

El modelo del robot está conformado por una base, dos ruedas, un sensor láser Hokuyo ubicado en la parte superior del robot y una cámara Kinect ubicada en la parte frontal del robot.

Los archivos que definen el modelo del robot son dos archivos ubicados en la carpeta URDF. El archivo ucho_bot.xacro proporciona la descripción de la forma del robot en formato macro, mientras el ucho_bot.gazebo proporciona definiciones del controlador diferencial de la unidad, la cámara RGB-D, el escáner láser y el controlador de estos sensores para Gazebo.

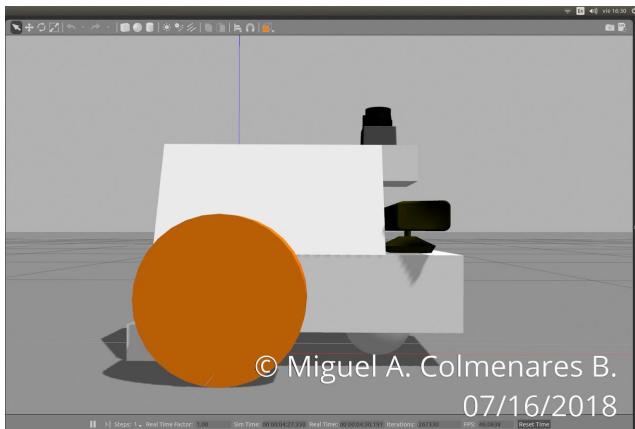


Fig 3. Vista Lateral del Robot.



Fig 4. Vista Frontal del Robot.



Fig 5. Vista Superior del Robot.

El robot fue desarrollado desde Gazebo en un mundo vacío. Luego de tener el modelo del robot listo, el robot se ubica en el entorno de prueba y se ejecuta el siguiente comando:

```
$ rosrun tf view_frames
```

El comando anterior genere un PDF donde se puede observar el marco de coordenadas de las uniones del robot, junto con el marco de coordenadas adicionales para el mundo y la odometría.

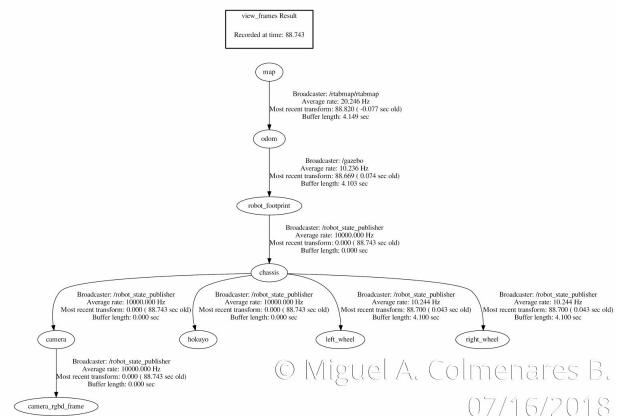


Fig 6. Marcos de Coordenadas Union, Mapa, Odometría, etc.

Configuración del Entorno.

El robot realiza la prueba de implementación del SLAM en dos entornos diferentes. El primer entorno en ser evaluado (`worlds/kitchen_dining.world`) es proporcionado por un tercero, es una cocina y un comedor. El segundo entorno es más simple. El segundo entorno (`worlds/udacity.world`) y con elementos muy variados.



Fig 7. Entorno kitchen_dining.world



Fig 8. Entorno udacity.worlds

IV RESULTADOS.

Para iniciar la simulación de alguno de los entornos (`kitchen_dining/bot_world`) se ejecutan cuatro nodos:

```
| $ rosrun udacity_map world.launch
| $ rosrun udacity_map teleop.launch
| $ rosrun udacity_map mapping.launch
| $ rosrun udacity_map rviz.launch
```

Respectivamente en ese orden y esperando un tiempo prudencial entre ejecución para que cada nodo se cargue correctamente.

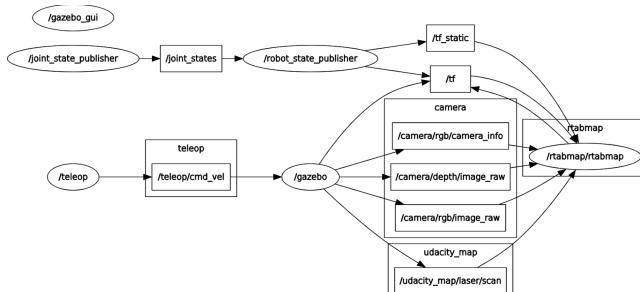


Fig 9. Gráfico ROS del paquete udacity_map.

Escena Kitchen and Dining.

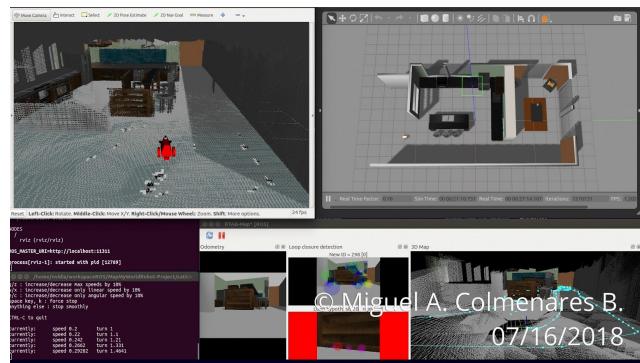


Fig 10. Mapeo del entorno Kitchen_dining.

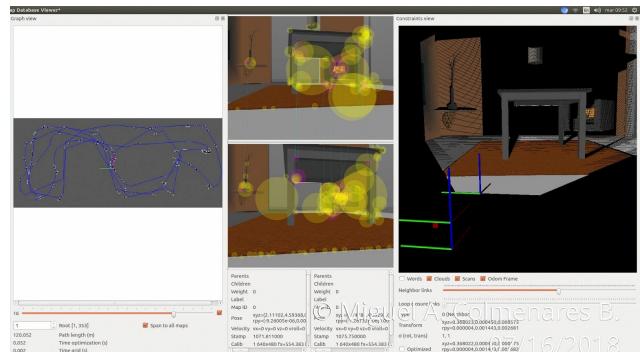


Fig 11. Análisis de la base de datos rtabmapKitchen.db.

Escena Bot World.

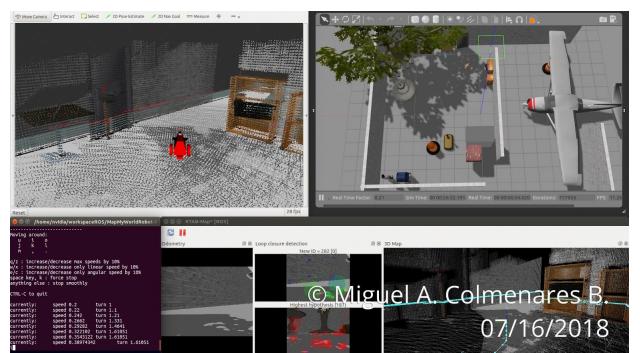


Fig 12. Mapeo del entorno Bot world.

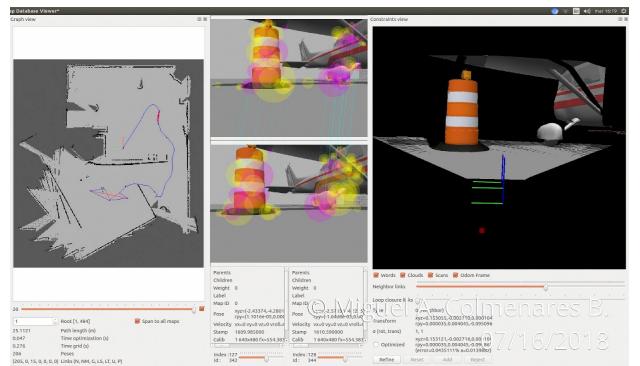


Fig 13. Análisis de la base de datos rtabmapBotWorld.db.

Luego de cargar cada nodo, en la pantalla se abrio una ventana Gazebo con la escena que se está probando, una ventana Rviz donde se muestra el mapeo realizado por el robot, una ventana de la herramienta RTAB-Map y una terminal con el nodo teleop cargado para controlar al robot desde el teclado.

Después de atravesar y navegar por todo el mundo Gazebo, el robot fue capaz de generar un mapa preciso de la escena (Kitchen dining/bot_world). Seguidamente se ejecuto el comando:

```
| $ rtabmap-databaseViewer ~/ros/rtabmap###.db
```

Para abrir la base de datos rtabmap###.db (rtabmapKitchen.db para el mundo Kitchen dining y rtabmapBotWorld.db para el mundo Bot world). Con esta herramienta se pueden verificará el cierre de bucles, generar mapas en 3D para ver, extraer imágenes, verificar zonas de gran riqueza de mapas de funciones y ¡mucho más!.

V DISCUSIÓN.

El modelo de robot construido fue adecuado para las pruebas de mapeo. El robot tránsito bien por ambos entornos. La escena Kitchen dining fue ideal para que el robot implementará SLAM, de esta escena se obtuvieron buenos resultados y el robot pudo mapear el entorno. El único problema fue conseguir un lazo cerrado del recorrido del robot por toda la escena. La escena Bot world fue más complicada y tuvo sus dificultades. La primera dificultad fue la falta de experiencia manejando el programa Gazebo por lo cual se llevo tiempo familiarizarse con el programa. La segunda dificultad fue manejar el robot por la escena que resultó ser muy grande para el robot, por lo cual el robot tardó más tiempo en recorrer la escena. Y la tercera dificultad fue que durante la prueba con la segunda escena, en varias ocasiones el robot deslizó y no se pudo recuperar. De por lo demás, el robot no tuvo mayores problemas en mapear ambas escenas con éxito.

VI TRABAJOS FUTUROS.

Como trabajos futuros, quedan tres puntos específicos por atender:

- Construir un robot de con dos ruedas oruga, equipado con el Jetson TX2 y un sensor de cámara Kinect RGBD para mapear un apartamento completo de la vida real.
- Agregar un algoritmo de planificación de movimiento.
- Agregar un algoritmo de detección de colisión.