Departamento de Ingeniería Eléctrica
Facultad de Ciencias Físicas y Matemáticas
Universidad de Chile
UChile Robotics Team

Pepper Navigation:

ORB SLAM 3

ROS Wrapper

Estudiantes:
Sebastián Herrera
Ignacio Romero
Tutora:
Michelle Valenzuela
Profesor:
Javier Ruiz del Solar
Fecha:
1 de septiembre de 2023

Índice

1. Introducción

Pepper es el robot oficial utilizado en la Liga de la Plataforma Estándar RoboCup@Home. Presenta varias ventajas para la interacción humano-robot, como su apariencia amigable, pero tiene limitaciones importantes, como sus capacidades reducidas de detección y cómputo. En contraste con los robots personalizados que generalmente dependen de LIDAR costosos para la localización y navegación métrica, que funcionan tanto en entornos interiores como exteriores, Pepper tiene LIDARs de corto alcance y una cámara RGBD que proporcionan una localización confiable solo en habitaciones interiores pequeñas, siendo incapaz de proporcionar información útil para localizar al robot en entornos grandes. Esto es un problema importante para Pepper, que se espera que se utilice no solo en hogares, sino también en lugares públicos como hospitales, centros comerciales y escuelas [1]. Para abordar esta limitación, se propone una solución conceptual de auto-localización basada en Visual-SLAM que presenta se presentan los siguientes objetivos:

- Objetivo General:
  - Mejorar las capacidades de localización y navegación de Pepper mediante el desarrollo de un sistema de mapeo y localización basado en visión.
- Objetivos Específicos:
  - Lograr mapeo en 3D de un entorno utilizando ORB_SLAM3.
  - Implementar Navegación en un mapa creado con ORB_SLAM3.

En este sentido, tras investigar el estado del arte, se quiso implementar un sistema de mapeo, localización y navegación basado en *ORB_SLAM3*, el primer sistema capaz de realizar SLAM visual, visual-inercial y multi-mapa con cámaras monoculares, estéreo y RGB-D, utilizando modelos de lente *pinhole* y *fisheye*. *(...)* El resultado es un sistema que opera de manera robusta en tiempo real, en entornos pequeños y grandes, interiores y exteriores, y es de 2 a 5 veces más preciso que enfoques anteriores [2].

El desarrollo del equipo permite crear; a partir de información 3D del entorno, recopilada con ORBSLAM3, obtenida mediante cámaras monoculares o RGB-D; una reconstrucción 2D para la directa implementación a, prácticamente, cualquier algoritmo de navegación en ROS.

2. Desarrollo

2.1. ¿Qué se desarrolló?

A lo largo del semestre, se realizó una investigación del estado de arte en sistemas de SLAM visual y mapeo 3D, se evaluaron diferentes opciones de bibliotecas open source. Dentro de las opciones que se barajaron, se encontraba ORBSLAM2, que ya había tenido una implementación en Pepper por parte de un antiguo equipo del laboratorio [1].

En un comienzo, debido a que ya existía una implementación directa para Pepper de ORBSLAM2, se había decidido realizar continuar con este trabajo, sin embargo, tras conversar con el equipo que había realizado ese desarrollo, se recomendó *actualizar* la implementación a ORBSLAM3, debido a que ofrecía el mejor balance entre precisión, rendimiento y flexibilidad para su intregración con ROS.

La forma en que funciona ORBSLAM3 es mediante la creación de una nube de puntos 3D representando el entorno mapeado, utilizando la información capturada por la cámara. Una vez creado este modelo 3D, el sistema es capaz de localizarse dentro de dicho modelo, comparando las nuevas imágenes capturadas con la nube de puntos previamente generada. De esta manera, ORBSLAM3 puede estimar en tiempo real la posición de la cámara con respecto al modelo 3D del entorno.

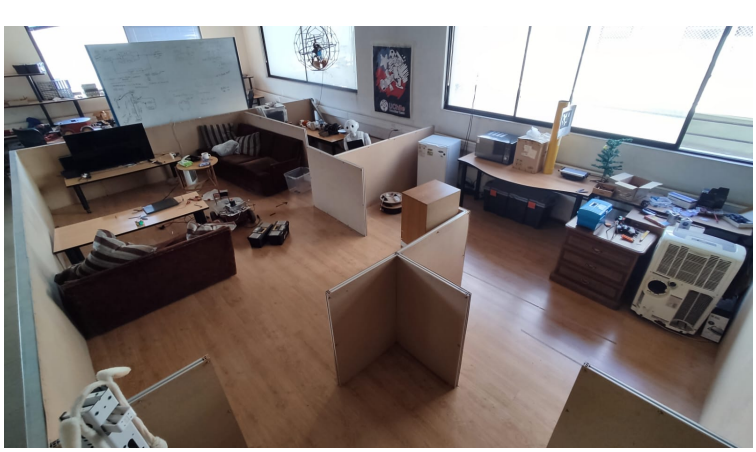
Luego, la información 3D provista por ORBSLAM3 es transformada a una representación 2D para generar un mapa utilizando el algoritmo gmapping en ROS. Esta representación 2D del entorno mapeado puede ser luego utilizada por la mayoría de algoritmos de navegación en ROS, permitiendo al robot moverse de forma autónoma en el entorno modelado.

Al comienzo del desarrollo, se invirtió tiempo en entender en detalle el funcionamiento interno de ORBSLAM3, sus diferentes modos de operación y parámetros de configuración, sin embargo, se tuvieron problemas con la limitada capacidad de cómputo y sensores del robot Pepper, lo que dificultó obtener mapeados 3D precisos y en tiempo real utilizando su cámara monocular.

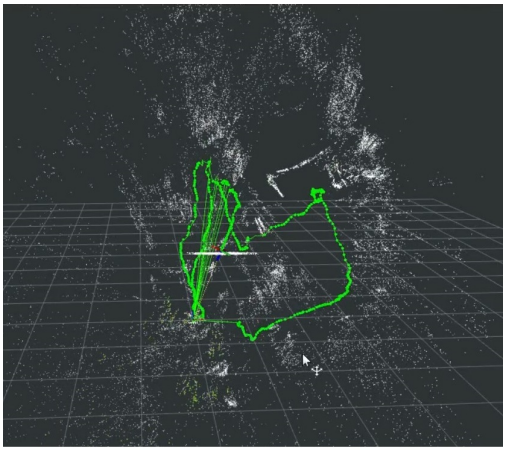
2.1.1. Funcionamiento del sistema

Una noción de cómo funciona el entorno completo es la siguiente:

- ORBSLAM crea una nube de puntos del entorno con la informa que recibe de la cámara.



((a)) Espacio real



((b)) Mapa generado. La línea verde representa la trayectoria del robot en la instancia actual.

Figura 1: Vista cenital de ejemplo de una nube de puntos creada con ORBSLAM3.

- ORBSLAM entra en modo localización y, con las imágenes que recibe, es capaz de determinar en qué parte del espacio se encuentra dentro del mapa que ya creó.

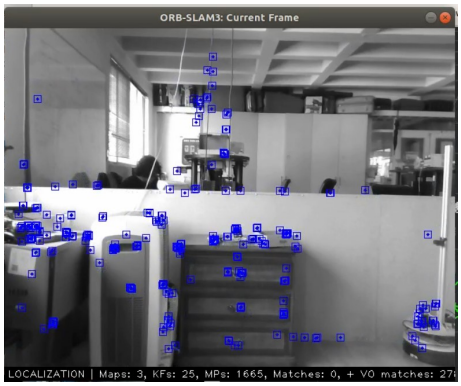


Figura 2: ORBSLAM localizandose dentro de un entorno con un mapa ya creado con una Realsense D435i.

Los puntos azules que se aprecian en la figura anterior representan las coincidencias con algún mapa cargado previamente.

- Una vez que ORBSLAM es capaz de localizarse en el entorno, se transforma su información 3D en información 2D, para crear un mapa utilizando el algoritmo *gmapping*[1]. El propósito trás generear un mapa 2D con gmapping es para facilitar su uso en algoritmos de navegación. También es posible guardar el mapa generador por ORBSLAM a través de su propio servicio de ROS, estos mapas son guardados en formato ".osa". Se pueden cargar los mapas guardados para siguientes instancias de ORBSLAM.

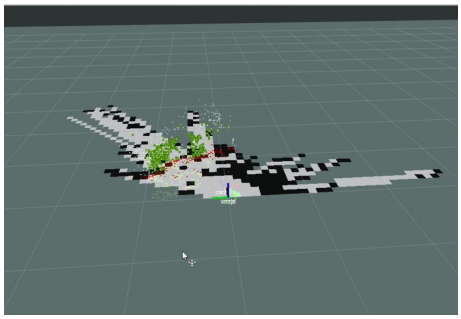


Figura 3: Creación de un mapa 2D a partir de la información de un entorno 3D proporcionada por ORBSLAM

Por otro lado, el desarrollo del proyecto, cómo utilizarlo y la guía de instalación para el caso del robot Pepper y para una cámara RealsenseD435i, se pueden encontrar, de manera específica y completa en el repositorio de Github[2] del proyecto. A continuación se menciona de manera general el instructivo de funcionamiento del proyecto, asumiendo que el repositorio y todas sus dependencias ya se encuentran instaladas:

- Pepper's case (monocular):
  - Lanzar el core de Maqui: *roslaunch maqui_bringup maqui.launch*
  - Conectarse a Pepper exportando el ROS MASTER en cada nueva terminal del computador: *export ROS_MASTER_URI=http://pepper.local:11311*
  - Lanzar ORBSLAM: ***roslaunch orb_slam3_ros ntuviral_mono_maqui.launch***
  - Mapear: Teleoperar el robot y mapear el entorno, procurando que la cabeza de Maqui esté estática en todo momento para evitar errores indeseados en el proceso.
- Realsense's case (RGB-D):
  - Lanzar el nodo de la cámara: ***roslaunch orb_slam3_ros realsense_d435i.launch***
  - Lanzar ORBSLAM: ***roslaunch orb_slam3_ros tum_rgbd_rs.launch***
  - Mapear el entorno.
  - **Realsense montada en un robot**: Si este es el caso, asegurarse de que los *frames* camera y world, dentro del launch de orbslam correspondiente, son los correctos respecto al robot. Además, para mapear, bastará teleoperar al robot.

[1] `http://wiki.ros.org/gmapping`
[2] `https://github.com/tnacho-23/orb_slam3_ros`

- Guardar y Cargar un Mapa

  El archivo de mapa tendrá la extensión .osa y se ubicará en la carpeta ROS_HOME (~/.ros/ de manera predeterminada).

  - Cargar mapa:
    - Establecer el nombre del archivo de mapa a cargar con el parámetro System.LoadAtlasFromFile en el archivo de configuración (.yaml).
    - Si el archivo de mapa no está disponible, el parámetro System.LoadAtlasFromFile debe estar comentado, de lo contrario, se producirá un error.
  - Guardar Mapa:
    - Guardar mapa:
      - Opción 1: Si se establece System.SaveAtlasToFile en el archivo de configuración, el archivo de mapa se guardará automáticamente cuando finalices el nodo ROS.
      - Opción 2: También puedes llamar al siguiente servicio de ROS al final de la sesión: ***rosservice call /orb_slam3/save_map [nombre_del_archivo]***

  Una vez que el mapeo 3D esté listo, se debe guardar el mapa con alguno de los dos métodos descritos anteriormente para posteriormente cargar el mapa en el modo de localización. Luego, se debe iniciar el lanzamiento que permite pasar de una nube de puntos a escaneo láser (***roslaunch orb_slam3_ros pointcloud_to_laser_scan.launch***) y por ultimo se puede intentar guardar el mapa con gmapping ejecutando el siguiente comando: "***roslaunch orb_slam3_ros orb_mapping.launch***".

  Ahora, cada vez que se desee cargar un mapa para su uso en navegación, se debe cargar el mapa 3D de orbslam y el archivo PGM generado con gmapping utilizando algún algoritmo de navegación.

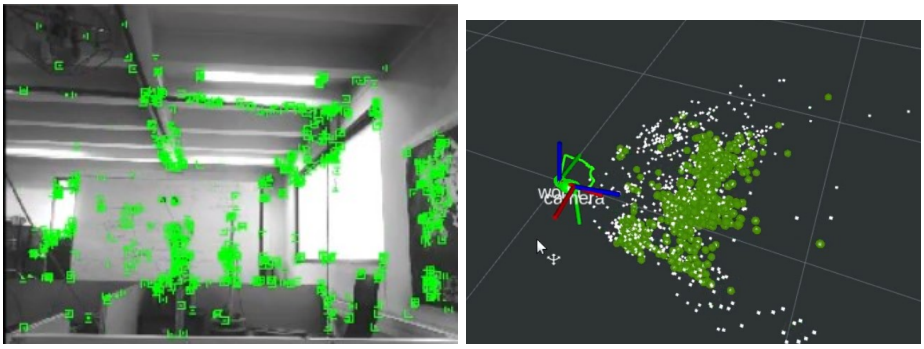
  Es importante aclarar que el proceso de SLAM y localización propios de ORBSLAM3 se realizan en tiempo real.

3. Resultados

3.1. Caso monocular: Pepper Robot

Al realizar la implementación del sistema en el robot Pepper, con una cámara monocular, se obtuvieron los siguientes resultados:

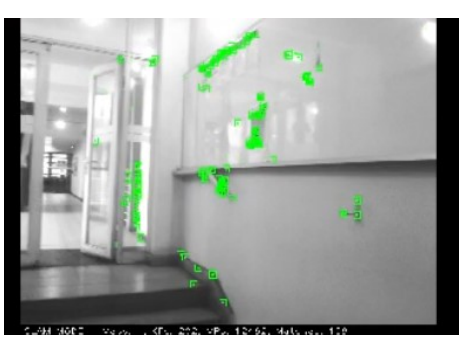
- Los mapas 3D creados por Pepper en entornos con mucha información visual, no son precisos y no tienen coherencia ni cohesión con el entorno real. Tal y como se muestra en la figura 4, si bien, como se ve en *a*, ORBSLAM es capaz de detectar correctamente, sin embargo, la recreación del entorno, mostrada en *b*, no parece representar correctamente el entorno en el que se está moviendo el robot, esto es, la proyección de los puntos detectados parecen estar puestos de manera *arbitraria* frente al robot.



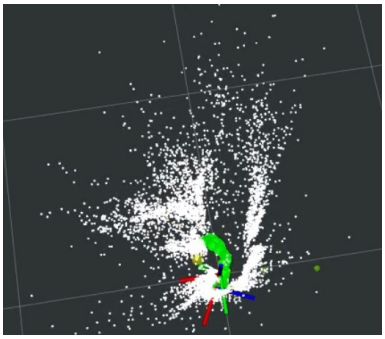
(a) (b)

Figura 4: Robot Pepper Mapeando el stage del laboratorio de robótica.

- En base al punto anterior, al probar mapear un entorno con menor cantidad de información visual, esto es, un pasillo como el ubicado afuera del laboratorio de robótica, se obtuvo un mapa como en de la figura 5. Como se ve en *a*, la cantidad de puntos que detecta como información relevante para el mapa, es mucho menor comparando con la figura 4, sin embargo,la recreación 3D del entorno mostrada en *b* es mucho más coherente a la realidad, es decir, se pueden identificar a simple vista paredes y objetos relevantes, no obstante, es posible ver cómo la escala del espacio no corresponde a la realidad, esto es, las dimensiones de las paredes, espacios de pasillos, etc. No coinciden con la escala del mundo real. Se cree que esto pude venir dado por el hecho de estar utilizando una cámara monocular en lugar de una cámara que entregue información de profundidad.



(a) Puntos creados por ORBSLAM.



(b) Nube de puntos vista en RViz.

(c) Espacio físico a mapear.

Figura 5: Mapa 3D creado con una cámara monocular con el robot Pepper.

- Una vez creado el mapa 3D del entorno, se procedió a probar el modo *localización* de ORBSLAM. Sin embargo, como se ve en la figura 6, pese atener cargado el mapa, el robot no fue capaz de detectar el lugar del entorno en el que se encontraba.

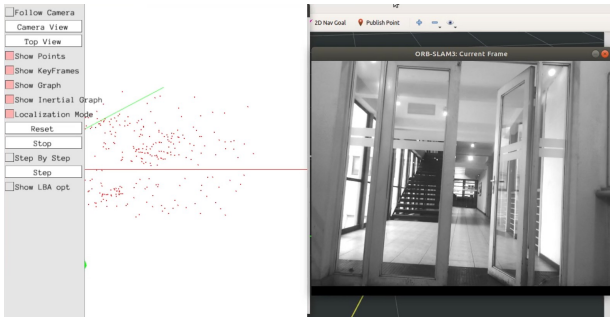


Figura 6: Pepper intenta localizarse.

En la imagen se observa la representación 3D nativa del mapa creado por ORBSLAM, también en la misma ventana se encuentra una interfaz que permite habilitar el modo de localización (el cual está activo en el momento que se capturó la imagen de la figura), a su derecha está la ventana correspondiente a como ORBSLAM está procesando las imagenes que recibe, en este caso en

partícular se observa que sólo hay una imagen en escalas de grises. Normalmente debería haber texto en el pequeño espacio negro debajo de la imagen con información (tal como se muestra en la figura 6), la carencia de este texto se debe a que ORBSLAM necesita localizarse para comenzar a mostrar el texto con información, pero como se mencionó anteriormente, Pepper no logra localizarse por lo que nunca se mostrará un texto en esa ventana.

3.2. Caso RGB-D: Pepper v/s Realsense Camera

Debido a que no todo resultó como lo esperado con la cámara monocular del robot Pepper, se decidió "imitar" un sensor RGB-D a partir de otros sensores propios del robot. No obstante, tampoco se llegó a buen puerto, por lo que finalmente se optó por realizar pruebas con una cámara RGB-D. Para ello, se utilizó la cámara Realsense D435i con el modo RGB-D.

Las últimas pruebas realizadas con la Realsense D435i resultaron mucho más fructíferas que las realizadas previamente con Pepper. Un ejemplo claro para ejemplificar esto es al momento de localizar, si se observa la figura 6, no se aprecian detalles más allá de una imagen en escala de grises, esto implica que Pepper no logra localizarse con el mapa que creó previamente, que es totalmente lo opuesto al resultado de la Realsense, ya que ésta sí logra localizarse con los mapas cargados previamente marcando con color azul las coincidencias que encuentra con respecto al mapa cargado (Figura 2).

Además de localizarse correctamente con la Realsense, los mapas obtenidos poseen coherencia y cohesión, con escalas acorde al mundo real sin importar la cantidad de obstáculos disponibles en el entorno, tal como se puede observar en la figura 1(a).

4. Conclusiones

Para concluir, se logró implementar un algoritmo capaz de crear mapas *navegables* utilizando ROS, en particular, el equipo fue capaz de hacer la implementación de un algoritmo de mapeo en 3D que es capaz de conectarse de manera directa con el stack de navegación existente. Aunque este no funciona como era esperado en Pepper debido a las limitaciones técnicas impuestas por su hardware, tales como el hecho de que el robot no cuenta con una cámara capaz de entregar información de color y profundidad del entorno, como su baja capacidad de cómputo.

Sin embargo, y a pesar de que no se logró el objetivo inicial que era mejorar las capacidades de navegación en Pepper, sí se logró que el sistema funcionara en otro hardware. En este sentido, el utilizar una Realsense (u otra cámara con similares características) abre la posibilidad de una implementación directa en otros robots, como Jaime o Bender. Esto les permitiría detectar de mejor manera su entorno, lo que abre las posibilidades a fusionar el sistema de navegación que ellos ya poseen con esta implementación de ORBSLAM3, lo que posibilitaría robustecer la navegación de estos y otros robots e invita a profundizar en la aplicabilidad futura del proyecto.

Para una eventual integración en Pepper, aunque su hardware actual presente desafíos, sería factible explorar una solución externa, como la instalación de una Realsense en una ubicación estratégica, conectada a una plataforma de procesamiento como una *NVIDIA Jetson*. Esta configuración permitiría llevar a cabo el procesamiento necesario de manera efectiva y proporcionar las instrucciones requeridas a Pepper para lograr la funcionalidad deseada, además de abrir la posibilidad de realizar todo tipo de proyectos mucho más profundos, sin limitarse a la navegación, tal y como se ha hecho en otros casos, donde, pese a que el robot cuenta con sensores de profundidad, micrófono y capacidad de procesamiento, se hace necesario la utilización de software externo como se ve en la figura 7, en un proyecto llevado a cabo en la *Università degli Studi di Salerno* donde se requiere utilizar estas características en el robot[3].

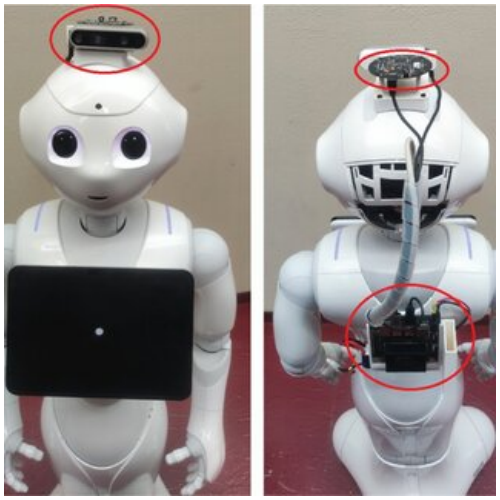


Figura 7: El sistema robótico propuesto. Pepper ha sido equipado con una cámara RGB-D (marcada en rojo en la imagen de la izquierda) y una matriz de micrófonos (marcada en rojo en la imagen de la derecha, ubicada en la parte superior de la cabeza). El NVIDIA Jetson Xavier NX (marcado en rojo en la imagen de la derecha, en la parte inferior) está montado en la parte trasera del robot a través de un soporte impreso en 3D y alimentado por una batería de litio.

Referencias

[1] Cristopher Gómez et al. "Visual SLAM-Based Localization and Navigation for Service Robots: The Pepper Case". En: *RoboCup 2018: Robot World Cup XXII*. Springer International Publishing, 2019, págs. 32-44. DOI: 10.1007/978-3-030-27544-0_3. URL: https://doi.org/10.1007%2F978-3-030-27544-0_3.

[2] Carlos Campos et al. "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM". En: *IEEE Transactions on Robotics* 37.6 (dic. de 2021), págs. 1874-1890. DOI: 10.1109/tro.2021.3075644. URL: https://doi.org/10.1109%2Ftro.2021.3075644.

[3] Pasquale Foggia et al. "Few-shot re-identification of the speaker by social robots". En: *Autonomous Robots* 47 (nov. de 2022). DOI: 10.1007/s10514-022-10073-6.