

AMPLIACIÓN DE ROBÓTICA

Comparativa de técnicas de SLAM empleando un robot movil

Grado en Ingeniería Electrónica, Mecatrónica y Robótica

Índice

1. Introducción al proyecto	2
2. Hardware empleado en el proyecto	3
2.1. Robot movil diseñado	3
2.2. Diseño PCB	3
2.3. Controladores de abordo	3
2.3.1. Raspberry Pi	3
2.3.2. Arduino	3
2.4. Cámaras empleadas	3
3. Software empleado en el proyecto	4
3.1. Lo de ROS y tal	4
3.2. Uso de librerías externas	4
4. Implementación filtro estadístico	4
4.1. Filtro de Kalman Extendido	4
5. Introducción al SLAM(<i>Simultaneous Localization and Mapping</i>)	5
5.1. Clasificación técnicas de SLAM	5
5.2. <i>RTAB-Map SLAM</i>	6
5.2.1. Fundamento teórico de la técnica	6
5.2.2. Análisis de resultados obtenidos	7
5.3. <i>ORB-SLAM 2</i>	8
5.3.1. Fundamento teórico de la técnica	8
5.3.2. Análisis de resultados obtenidos	9

Autores:

López Gil, Miguel
Montes Grova, Marco Antonio
Osuna Cañas, Alfonso Carlos

1. Introducción al proyecto

En este proyecto se implementan diferentes técnicas de SLAM visuales, así como diversas fuentes de odometría, sobre un robot móvil de bajo presupuesto.

El robot ha sido diseñado con una topología inspirada en los vehículos de exploración espacial conocidos como *rovers*, utilizando materiales y métodos de construcción asequibles y manteniendo como objetivo una sencilla sustitución e iterabilidad de cada componente. La estructura está diseñada en Free-Cad y compuesta por piezas de madera mecanizadas en una cortadora láser cnc. Algunos elementos sin funcionalidad estructural, como los encoders o el soporte de la cámara, han sido fabricados mediante impresión 3D.

La disposición geométrica es la de un vehículo diferencial, con tres pares de ruedas, cada una con su respectiva pareja de motor y encoder. Como drivers de los motores se han usado tres puentes H *L293D* integrados en una placa de circuito impreso diseñada expresamente para ello.

La odometría del robot se obtiene a partir de una estimación de la posición a partir de encoders ópticos, y de la velocidad y aceleración medidas por una IMU *mpu-6050*. La fusión sensorial se realiza mediante un filtro de Kalman extendido, implementado en el paquete *robot localization* de ROS kinetic. Las imágenes y medidas de profundidad utilizadas para las técnicas SLAM provienen de una cámara *Kinect for Xbox 360* de Microsoft, que cuenta con una cámara RGB y un sensor de profundidad matricial.

2. Hardware empleado en el proyecto

2.1. Robot movil diseñado

2.2. Diseño PCB

2.3. Controladores de abordó

2.3.1. Raspberry Pi

2.3.2. Arduino

2.4. Cámaras empleadas

3. Software empleado en el proyecto

3.1. Lo de ROS y tal

3.2. Uso de librerías externas

4. Implementación filtro estadístico

4.1. Filtro de Kalman Extendido

5. Introducción al SLAM (*Simultaneous Localization and Mapping*)

¿Qué cojones es el SLAM?

5.1. Clasificación técnicas de SLAM

La forma en que se toma información de las imágenes que recibe y de su entorno y cómo la procesa permite distinguir las siguientes tipos de sistemas SLAM visual:

- *Denso vs Disperso*

En función de la cantidad de datos tomada de cada imagen, se realizará ésta clasificación. Las técnicas de SLAM disperso sólo emplean una pequeña región de píxeles, cómo pueden ser puntos característicos. Sin embargo, las técnicas densas emplean la mayoría o todos los píxeles de cada frame que reciben.

Debido a que la cantidad y tipo de información que toman son diferentes, el tipo de mapa que se obtendrá también lo es. Con las técnicas de SLAM disperso se obtendrá una nube de puntos que será una representación de los puntos característicos de las escena y se usará sobre todo para hacer un tracking de la pose de la cámara. Por otro lado, un mapa dentro tendrá muchos más detalles y, requerirá de una mucho más elevada carga computacional.

- *Directo vs Indirecto*

En función de cómo las técnicas de SLAM empleen y traten la información que reciben, se podrá clasificar en SLAM directo o indirecto.

El SLAM infirecto, se basa en extraer primero las features de la imagen, puntos característicos, para posteriormente emplearlos para localizarse y contruir el mapa. Para extraer estos puntos característicos existen muchos descriptores: ORB, SIFT, FAST, etc.

En contraste, el SLAM Directo, emplean directamente la intensidad de los píxeles, de tal modo que se obtengan features intermedios. Estos metodos tratan de obtener la profundidad y estructura del entorno a partir de una optimización del mapa. Los procesos de extracción de puntos característicos son mucho más pesados computacionalmente si se trabaja al mismo rate que un slam indirecto.

Por último, destacar, que los metodos indirectos de SLAM son más tolerantes a los cambios de iluminación del entorno.

A continuación, se monstarán una comparativa de una serie de técnicas de SLAM:

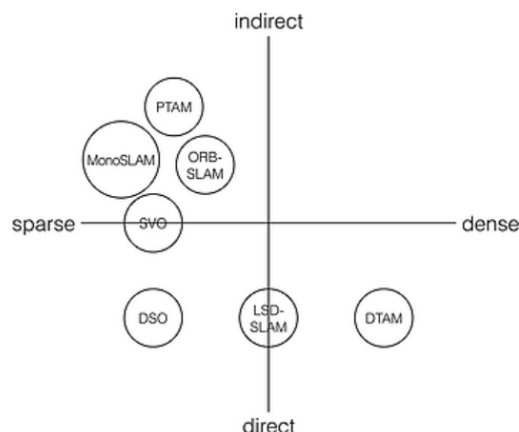


Figura 1: Comparativa de técnicas SLAM

En éste proyecto, se probará ORB-SLAM2 y RTAB-Map, los cuales abordan el problema del SLAM desde dos puntos de vista completamente distinto, cómo se verá a continuación.

5.2. RTAB-Map SLAM

RTAB-Map (*Real-Time Appearance-Based Mapping*) es una técnica de Graph-SLAM¹ basada en la detección de bucles cerrados incrementales. Es totalmente funcional con sensores RGB-D, Stereo y LIDAR.

El detector de bucles cerrados se basará en la comparativa de cuán semejantes son la imágenes en una localización y la previa. Cuando una hipótesis de bucle cerrado es aceptada, se añade una nueva restricción al *graph* del mapa y, tras ello, el optimizador minimiza el error del mapa.

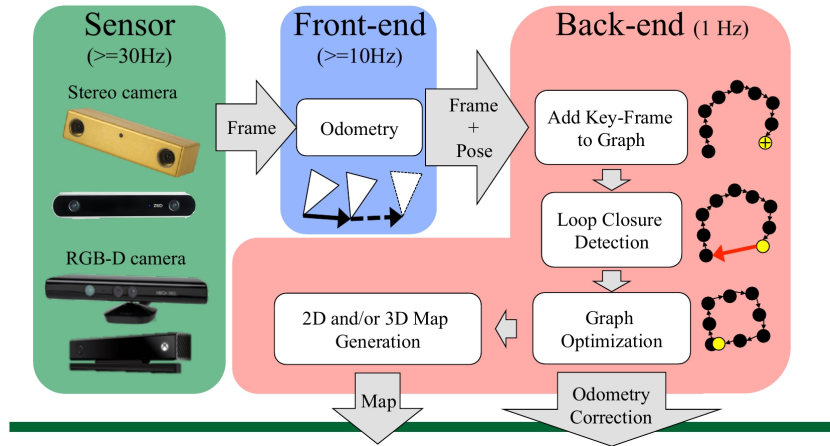


Figura 2: Esquema del Back-End y Front-End de RTAB-Map

5.2.1. Fundamento teórico de la técnica

Este algoritmo plantea una estrategia de particionado de memoria que pretende asemejarse al funcionamiento de la memoria humana, donde ésta se estructura en:

Memoria de trabajo del robot (*Working Memory*), la memoria a largo plazo (*Long Term Memory*), la memoria a corto plazo (*Short-term memory*) y la memoria sensorial (*Sensory Memory*).

De ese modo, se mantendrán en la memoria de trabajo del robot aquellas localizaciones que se han visitado recientemente y con más frecuencia, mientras que el resto pasarán a la memoria de largo plazo.

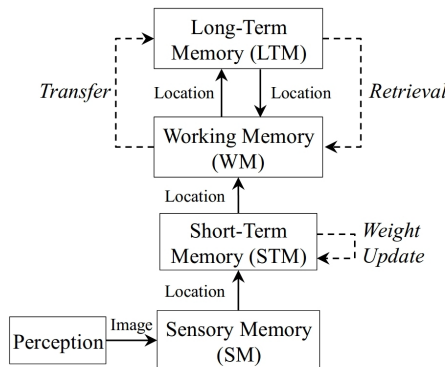


Figura 3: Estructura de memoria de la técnica RTAB-Map

Se partirá de la premisa de que aquellas localizaciones que son visitadas de forma más frecuente son más propensas a crear bucles cerrados. Por ello, el número de veces que una localización sea visitada será

¹<http://robots.stanford.edu/papers/thrun.graphslam.pdf>

empleado como peso, de esta forma serán transferidas desde la memoria de trabajo a la memoria de largo plazo aquellas observaciones que tengan mayor peso.

La memoria a corto plazo, *STM*, tiene como misión buscar las similitudes que existan entre dos imágenes consecutivas, mientras que la memoria de trabajo, *WM*, es la encargada de detectar los bucles cerrados entre las localizaciones en el espacio. El número de localizaciones almacenadas en la memoria del trabajo del robot es limitado. El tamaño de la memoria a corto plazo, *STM*, está basado en la velocidad del robot y en la frecuencia de adquisición de las localizaciones.

5.2.2. Análisis de resultados obtenidos

5.3. ORB-SLAM 2

ORB-SLAM2 es una técnica de SLAM en tiempo real para cámaras Monocular, Stereo y RGB-D que se engloba dentro de las técnicas de *Sparse-Slam*. Se computa la trayectoria y hace una reconstrucción dispersa 3D del entorno. Al igual que todas las técnicas de SLAM, se basa en la detección de bucles cerrados y se relocaliza en tiempo real.

Se basa en la detección de *keyframes* que emplea para hacer un tracking de los mismos y a partir de ello crear el mapa local que, posteriormente, se optimizará junto al mapa global.

Unos de los puntos destacables de ésta técnica de SLAM son los siguientes:

- Emplea *grafo de covisibilidad*. Tanto el seguimiento como el mapping se focalizan en el área covisible, independientemente del tamaño del mapa completo, consiguiendo así explorar entornos amplios sin aumentar el tiempo y la carga de computación.
- La estrategia para detectar los bucles cerrados de visión en tiempo real se basa en la optimización de un de grafo denominado *Essencial Graph*, lo cuál se desarrollará mas adelante.

5.3.1. Fundamento teórico de la técnica

A continuación, se tratará un poco el funcionamiento interno del algoritmo, el cuál presenta el siguiente esquema:

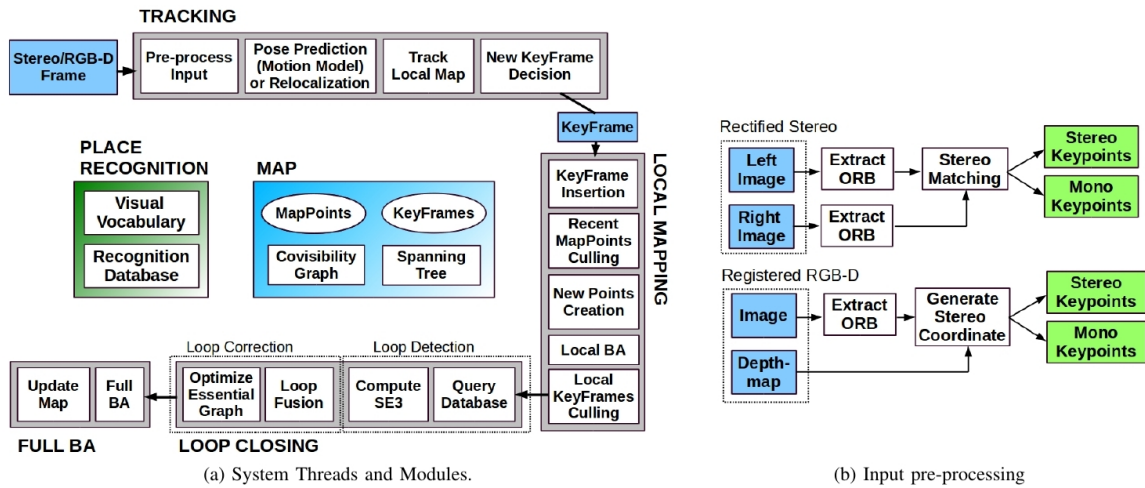


Figura 4: Estructura interna de ORB-SLAM2

Se observa cómo existen 6 grandes módulos, los cuales se lanzarán en 3 hilos paralelos.

En primer lugar, se tendrá el preprocesamiento de la imagen, de la cuál se obtendrán ORB features (*Oriented Fast and Rotated BRIEF*).

Los hilos desempeñarán las siguientes funciones:

- El *tracking* se encargará de localizar la cámara en cada frame buscando matches entre las features del mapa local y minimizando la reproyección del error aplicando un *Bundle Adjustment*² sólo de movimiento.
- El *Local Mapping* se encargará de gestionar y optimizar el mapa local aplicando un *BA* local.
- El hilo de *Loop Closing* se encargará de detectar bucles cerrados grandes y corregir la deriva acumulada realizando una optimización del grafo del *pose* obtenido. Este hilo, lanza 4 hilos internos que se encargarán de realizar un *Bundle Adjustment* completo tras la optimización del grafo del *pose*, para hacer la solución más optima.

²<https://homes.cs.washington.edu/~sagarwal/bal.pdf>

En caso de que el sistema pierda el tracking, lleva integrado un modulo de *Plane recognition* basado en un vocabulario de palabras que básicamente es una especie de preentrenamiento de la técnica, para poder obtener y asociar keyframes más fácilmente.

5.3.2. Análisis de resultados obtenidos

En la imagen a continuación, puede observarse la técnica corriendo en un ordenador. Destacar que ésta técnica de SLAM presenta una carga computacional bastante elevada.

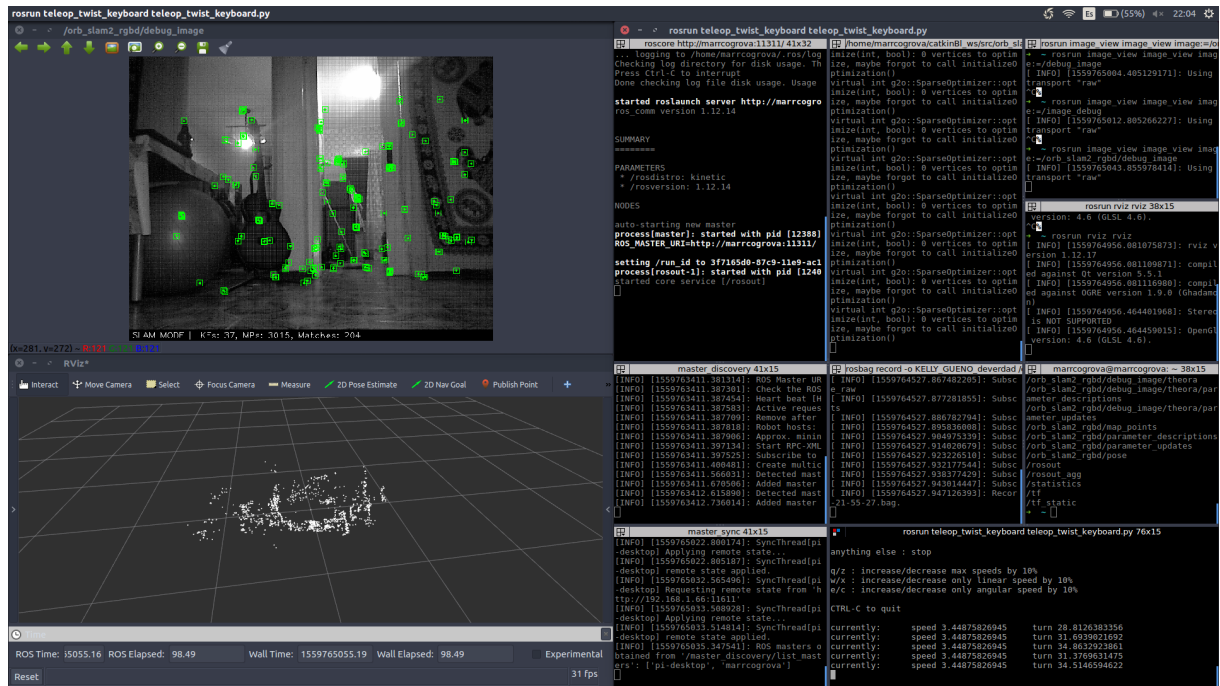


Figura 5: ORB-SLAM2 en funcionamiento

Se observa en la imagen superior izquierda las features que está obteniendo en ese frame, a partir de las cuales se ha creado en mapa local y, por extensión el global. En la imagen inferior izquierda, se puede observar el mapa creado, el cuál se analizará más tarde. Por último, a la derecha se tienen las terminales a partir de las cuales se ha lanzado el SLAM y se realiza la comunicación con el robot.

Referencias

- [1] R. Mur-Artal and J. D. Tardós, “ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras,” *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.
- [2] IntRoLab, “RTAB-Map(real-time appearance-based mapping): RGB-D, stereo and lidar graph-based SLAM.” <https://github.com/introlab/rtabmap>, 2014.
- [3] T. Moore and D. Stouch, “A generalized extended kalman filter implementation for the robot operating system,” in *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*, Springer, July 2014.
- [4] M. Harms, “Python interface for cmdmessenger arduino serial communication library.” <https://github.com/harmsm/PyCmdMessenger>, February 2017.
- [5] T. Elenbaas, “Command messenger communication library for arduino & .NET.” <https://github.com/thijse/Arduino-CmdMessenger>, September 2017.
- [6] “Gammon software solutions.” <http://www.gammon.com.au>.
- [7] Atmel, *Atmel ATmega640 Datasheet*, February 2014.