Trabajo Fin de Máster

Máster en Ingeniería Industrial

Análisis de algoritmos para localización y mapeado simultáneo de objetos

Autor: Patricia López Torres
Tutores: Begoña C. Arrue Ullés

Dep. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016

Trabajo de Fin de Máster
Máster en Ingeniería Industrial

Análisis de algoritmos para localización y mapeado simultáneo de objetos

Autor:
Patricia López Torres

Tutor:

Begoña C. Arrue Ullés
Titular

Ingeniería de Sistemas y Automática
Universidad de Sevilla

Sevilla, 2016

Trabajo de Fin de Máster:

Autor: Patricia López Torres

Tutor: Begoña Arrúe Ullés

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:
Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2016

El Secretario del Tribunal

AGRADECIMIENTOS

Mi más sincero agradecimiento a Begoña Arrúe, por todo el apoyo y ayuda que he recibido durante la elaboración del proyecto. También me gustaría dar las gracias a Pablo Ramón Soria por responder siempre a todas mis dudas por muy tontas que fueran.

Resumen

El SLAM es uno de los algoritmos que han surgido con la intención de ser capaz de posicionar al robot en un entorno desconocido a la misma vez que navega por él. El objetivo de este proyecto es analizar el estado del arte de esta técnica, mediante el estudio de diferentes proyectos de los últimos años que pretenden realizar un posicionamiento en interiores con diferentes variantes del algoritmo SLAM. Todas las técnicas estudiadas en el presente trabajo usan sensores de visión para adquirir datos del entorno. Tras el análisis de los algoritmos, se presenta un estudio sobre la fiabilidad de los mapas generados por cada uno de ellos.

Índice

ÍNDICE DE ILUSTRACIONES

1 INTRODUCCIÓN

En los últimos años, ha aumentado considerablemente el uso de la robótica móvil con fines comerciales e industriales con el objetivo de realizar tareas de forma más exacta o más barata que los humanos. A medida que aumenta el número de tareas que son capaces de realizar lo hace también la complejidad de las mismas, ya que se pretende que sean lo más autónomos posible. Para que los robots sean capaces de realizar sus funciones con éxito es necesario que sean capaces de desenvolverse en el entorno que navegan con facilidad, para ello es necesario que conozcan su posición, aunque se trate de un entorno desconocido.

El posicionamiento de los robots móviles en cualquier tarea que realice es un aspecto fundamental. Para misiones en exteriores este aspecto suele estar cubierto por un sistema de navegación GPS. No obstante, para misiones en interiores los sistemas de navegación por satélite no son válidos, por lo que es necesario desarrollar otro tipo de sistema de posicionamiento que permita a los robots conocer con relativa precisión la posición que ocupa en todo momento.

1.1. Objetivo del trabajo

Actualmente, el posicionamiento en interiores de robots móviles, ya sean aéreos o terrestres, es uno de los problemas más estudiados de la robótica a día de hoy. Se han desarrollado numerosas técnicas con diversos tipos de sensores que pretenden solventar este problema.

Entre los diferentes tipos de soluciones propuestas podemos encontrar algoritmos que previamente conocen el mapa por el que se va a navegar, o la colocación de balizas por el entorno que servirán de referencia para posicionarse a medida que el robot se desplaza. Estas soluciones son válidas si de antemano se conoce el entorno por el que navegará el robot. No obstante, existen numerosas aplicaciones en las que no se tiene acceso al entorno en que se realiza la tarea. Ante estas situaciones surgen técnicas capaces de posicionar al robot en un entorno desconocido a la misma vez que navega por él, como por ejemplo el SLAM.

El SLAM (Simultaneous Localization And Mapping) es un algoritmo muy estudiado durante los últimos años. Se planteó por primera vez en la Conferencia sobre Robótica y Automática del IEEE, en la ciudad de San Francisco, en 1986 y desde entonces se han desarrollado numerosas técnicas que pretenden solucionar el posicionamiento en interiores con diferentes variantes del algoritmo.

El objetivo de este proyecto es realizar un análisis de diferentes soluciones para el posicionamiento en interiores basadas en SLAM que se han desarrollado en los últimos años. En primer lugar, se analizarán las estrategias desarrolladas por varias técnicas basadas en SLAM con la intención de realizar un amplio estudio sobre el estado del arte sobre el SLAM. Posteriormente, se exponen una serie de experimentos que pretenden

estudiar la fiabilidad y robustez de los métodos analizados en la generación de los mapas y el posicionamiento. Pese a que el SLAM se puede realizar con diferentes tipos de sensores, todas las técnicas empleadas en el presente trabajo usan sensores de visión para adquirir datos del entorno, no obstante, en algunos métodos será visión monocular y en otros se emplean sensores que constan de una cámara y un sensor de profundidad.
.

1.1 Metodología

Este trabajo ha sido desarrollado en lenguaje C++ usando el framewok ROS (Robotic Operating System)., las librerías de visión OpenCv y la herramienta CMake. El entorno utilizado ha sido Ubuntu, usando un portátil Hewlett Packard ENVY 15 Notebook PC (Procesador Intel® Core ™ i7-4742MQ CPU @ 2.20 GHz, RAM 8.00 GB, Sistema operativo de 64 bits, procesador x64) y se ha hecho uso de la WebCam (Logitech HD WEBCAM C525) y de undispositivo Kinect.

1.2 Estructura de la memoria.

El trabajo está dividido en ocho capítulos. En primer lugar, se explica el algoritmo de SLAM de forma general y aclara ciertos conceptos que serán necesarios para comprender el funcionamiento de las técnicas que se analizarán posteriormente. Los capítulos siguientes, recogen el análisis de las herramientas de SLAM RTAB-Map, ORB-SLAM, LSD-SLAM y RGB-D SLAM además de una serie de pruebas que pretenden comprobar el funcionamiento de los mismos. Por último, se realiza una comparativa de los métodos estudiados.

2 EL ALGORITMO SLAM

El SLAM (Simultaneous Localization And Mapping) es una técnica que tiene como objetivo construir mapas de un entorno desconocido por un robot al mismo tiempo que estima su trayectoria al desplazarse por ese entorno usando dicho mapa. Conseguir dar solución al problema de posicionamiento en interiores con técnicas como esta, es una meta importantísima para la comunidad robótica ya que haría posible que el robot fuera realmente autónomo.

Como ya se adelantó en la introducción de este proyecto, la idea del SLAM surge en 1986 durante un congreso del IEEE(1). Por aquellos años, se había comenzado a incluir los métodos probabilísticos en la inteligencia artificial y en la robótica. Durante la conferencia un grupo de investigadores mantuvieron una conversación sobre la aplicación de métodos de estimación a los problemas de mapeado y localización. De dichas conversaciones se concluyó que la generación de mapas consistentes y probabilística era un problema fundamental en que necesitaba ser estudiado. En los años siguientes se publicaron varios artículos que establecieron las bases estadísticas para describir relaciones entre los objetos de referencia (landmarks) y manipular la incertidumbre geométrica. Dichos artículos mostraron que era necesario que existiera una gran correlación entre las estimaciones de localización de los diferentes landmarks en el mapa, y que dichas correlaciones deben crecer con las sucesivas correlaciones.

Al mismo tiempo, se estaban consiguiendo los primeros avances en el campo de la navegación basada en visión y con sensores tipo sonar para robots móviles, usando algoritmos basados en filtro de Kalman. Estas dos líneas de investigación fueron clave para impulsar la investigación sobre la navegación de robots en entornos desconocidos empleando landmarks.

El resultado de estas investigaciones mostró como mientras un robot móvil se movía a través de un entorno desconocido tomando observaciones relativas de landmarks, las estimaciones de estos landmarks estaban necesariamente correlacionadas unas con otras a causa del error común en la estimación de posición de la plataforma móvil. Esto indicaba que una solución consistente al problema de localización y mapeado simultáneo requería un estado conjunto compuesto por la posición del robot y la de los landmarks, que se actualizaría con cada nueva observación de dichos landmarks. Por ello, se necesitaría que el estimador de posición almacenara un vector de estados de un orden muy elevado, ya que dicho orden sería igual al número de landmarks que se extraigan del mapa, con escalado computacional igual al cuadrado del número de landmarks. Fundamentalmente, estos trabajos no buscaban la convergencia de propiedades del mapa o su comportamiento en estado estacionario. De hecho, se asumió que los errores en la estimación del mapa no convergirían.

Debido a la complejidad computacional del problema de generación de mapas y sin conocimientos del comportamiento convergente del mapa, los investigadores se centraron en una serie de aproximaciones al problema de generación de mapas, que asumen o incluso fuerzan las correlaciones entre landmarks para ser minimizadas o eliminadas, para así reducir el filtro completo a una serie de filtros landmark-vehículo desacoplados. Como consecuencia, se paralizó el estudio del problema combinado de localización y generación de mapas, y se abordaron por separado.

Al estudiar los problemas por separado se descubrió que el una vez formulado con un problema simple de estimación, era convergente. Esto supuso un gran avance para abordar de nuevo los problemas de forma conjunta. De hecho, se observó que las correlaciones entre landmarks, que algunos investigadores intentaron minimizar o eliminar, eran una pieza fundamental del problema y que cuánto mayor fueran esas correlaciones mejor sería la solución.

En el año 1995 en el *International Symposium on Robotics Research* fue presentado por primera vez la estructura del problema SLAM, el resultado convergente y se empleó el acrónimo SLAM para el algoritmo.

Desde entonces el algoritmo de SLAM es una técnica para posicionamiento en exteriores muy estudiada y con unos resultados bastante buenos, por ello este proyecto pretende abordar un estudio de diferentes métodos basados en dicho algoritmo.

2.1 Sensores

Uno de los aspectos más importante a tener en cuenta a la hora de implementar SLAM es el sensor que se va a emplear extraer la información del entorno por el que se desplaza el robot(2). Uno de los más usados es un escáner láser debido a la precisión que presentan (algunos tienen un error inferior a 10 milímetros) y a que son muy rápidos. Otra de las ventajas que presentan, es el poco procesamiento que necesitan los datos a la salida, ya que pueden ser interpretados directamente. No obstante, son sensores muy caros y no siempre dan buenos resultados. En algunas superficies, como el vidrio, la calidad de las medidas no es adecuada. Tampoco son adecuados para realizar SLAM bajo el agua, ya que en este medio los sensores láser tienen muy poco alcance.

Otro de los sensores que más se usan como aparatado de medida en estas técnicas son los de ultrasonido (sonar). Son más mucho más baratos que lo láseres, aunque sus medidas menos precisas. Sin embargo, son adecuados para usarlos bajo agua.

Sin embargo, los sensores vistos anteriormente son incapaces de extraer información de las superficies observadas e identificar objetos. Por este motivo, una de las formas más comunes de realizar SLAM es mediante visión. El mayor inconveniente es la gran cantidad de información que se debe procesar tras extraer los datos de las imágenes. No obstante, con el gran desarrollo que ha experimentado este campo, el procesamiento de los datos no supone un gran problema.

2.1.1 Uso de cámaras en SLAM

En el presente trabajo se abordan métodos de SLAM con cámaras. Para los algoritmos monoculares se emplea una cámara tipo webcam y para los demás un dispositivo Kinect.

Para que los resultados obtenidos por los algoritmos sean adecuados es importante realizar una calibración adecuada de las cámaras. La calibración tiene el objetivo obtener los parámetros internos de la cámara, y así mejorar la precisión de las operaciones que se apliquen o efectúen con estas imágenes. En nuestro caso la calibración se ha realizado mediante la librería de ROS *"camera_calibration"*. La calibración se ha realizado usando un patrón con una cuadrícula tipo tablero de ajedrez. Como resultado, se obtienen unos archivos de extensión "*. yaml"* que contienen las matrices de la cámara, de rectificación y de proyección y los coeficientes de distorsión.

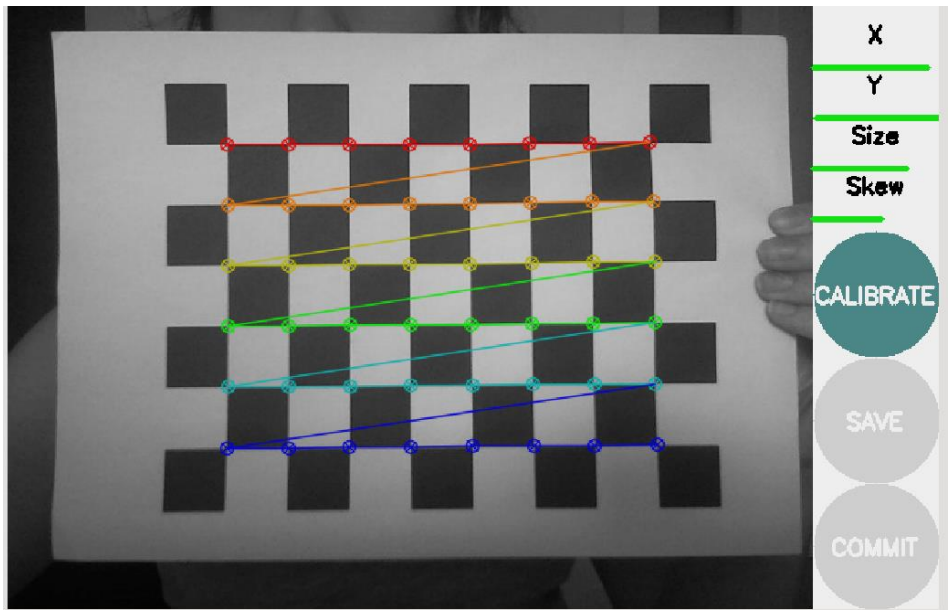

Figura 1: Interfaz empleada para la calibración de las cámaras

2.1.2 Kinect

Debido a que en dos de los métodos estudiados en este proyecto se emplea Kinect, se ha decido realizar una breve descripción del dispositivo, ya que es más compleja cámara y gracias a los sensores que lleva se puede obtener mucha más información del entorno por el que se navega. Kinect es un dispositivo que está formado por una barra horizontal de unos 23 cm conectada a una base con un pivote motorizado con un rango de movimiento de aproximadamente 30°.

El dispositivo cuenta con una cámara RGB, un sensor de profundidad, un micrófono de múltiples matrices y un procesador con el que ejecuta su propio software de Kinect. Para el caso de SLAM se harán uso de la cámara RGB y del sensor de profundidad.

Figura 2: Kinect

La captura de las imágenes a color se lleva a cabo a través de la cámara RGB. El sensor de profundidad que está formado por dos componentes: un generador de láser de infrarrojos y un sensor CMOS monocromo de luz infrarroja. Esta composición permite a la cámara capturar vídeo con información en tres dimensiones bajo cualquier condición de iluminación.

Debido a la popularidad del dispositivo hay varias librerías *open source* que nos permiten conectar la cámara al ordenador y manejar la información que maneja. Para las técnicas estudiadas en este proyecto se ha necesitado que la Kinect publicara cierta información, para ello se ha utilizado la librería Openni de ROS.

2.2 Odometría

La odometría es el uso de datos de sensores de movimiento para estimar la posición durante la navegación(2). En robótica, se emplea para conocer la posición relativa del sistema con respecto a la inicial. La precisión de la odometría permite conocer como de bien es capaz el robot de estimar su posición basado únicamente en el conocimiento de su forma de desplazamiento. Debido a que se trata de un cálculo incremental inevitablemente se acumulan errores, no obstante, para poder realizar SLAM el error debería ser inferior a 2cm/m y menor de 2° por cada 45° que el robot gire.

2.3 Landmarks

Los landmarks son aquellos puntos característicos del entorno que pueden ser fácilmente re-observables y distinguidos por el robot. Estos puntos se emplean para que el robot pueda localizarse dentro del entorno en el cual está navegando(2).

Es importante que los landmarks sean observables desde diferentes posiciones y ángulos. Además, deben ser únicos, para que puedan identificarse fácilmente y el algoritmo pueda determinar de forma rápida si se trata de una nueva observación o de un punto que ya se había observado con anterioridad. Para evitar errores, es necesario que estos puntos sean estacionarios y de un tamaño considerable.

Un aspecto importante en el algoritmo de SLAM es la asociación de datos, es decir hacer corresponder las distintas observaciones de un mismo landmark en distintas imágenes. Para realizar un posicionamiento adecuado, se debe identificar correctamente cuándo se está observando una referencia ya extraída en una iteración anterior.

Pueden surgir los siguientes problemas en la asociación:

- Es posible no re-observarse landmarks en cada paso del algoritmo.
- Podría identificarse un punto como landmark, pero fallar en su identificación al observarlo de nuevo.
- Es posible asociar erróneamente un landmark con otro visto anteriormente.

Con la intención de evitar los problemas expuestos, se suele comenzar con una base de datos vacía en la que se irán añadiendo los landmarks. Como primera regla para evitar errores, ningún punto será considerado como landmark hasta no ser observado N veces. Así, evitaremos extraer un landmark incorrecto. En cada nueva observación, se extraen todos los landmarks visibles. Una vez extraídos, los puntos característicos se asocian con los landmarks más cercanos que formen parte de la base de datos. Cada pareja asociada se analiza y se comprueba si son el mismo punto o no. En el caso de que sean iguales, se incrementa el número de veces que ese landmark ha sido observado. En caso contrario, añadimos el nuevo punto a nuestra base de datos y se pone a 1 el número de veces que ha sido visto el landmark.

2.4 Diagrama del algoritmo SLAM

Como ya se comentó anteriormente, el objetivo del SLAM es usar el entorno para conseguir estimar la posición del robot. Debido a que la odometría no es del todo fiable, no podemos confiar todo el posicionamiento en ella. Por este motivo se emplean sensores de distancias capaces de corregir los errores cometidos por la odometría.

Para ello, se extraen una serie de puntos característicos del entorno (*landmarks*) que se emplearan como referencia. Estas características deben ser fácilmente distinguibles y re-observables por robot. El encargado de actualizar estimación de la posición suele ser un EKF (Extended Kalman Filter) y es uno de los elementos más importantes del algoritmo de SLAM. En la *Figura 3* se observa el esquema general de funcionamiento del algoritmo SLAM.

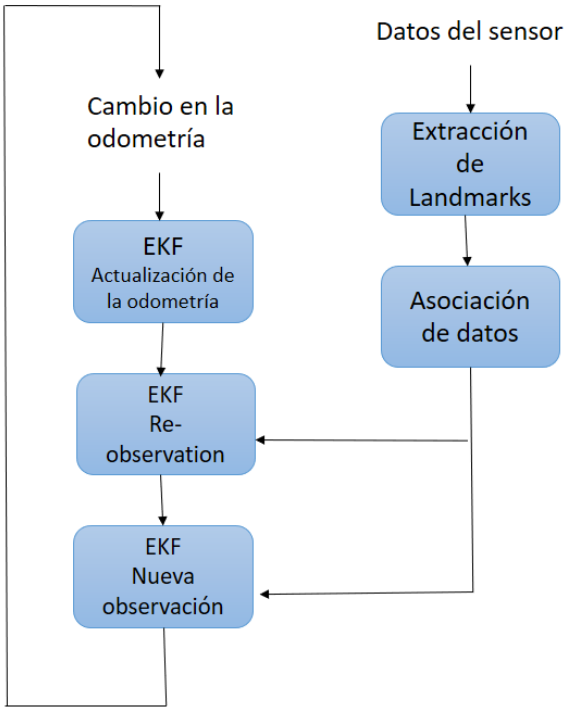

Figura 3: Esquema del algoritmo SLAM

Cuando el robot se mueve, su odometría cambia y se actualiza empleando el EKF. Por otra parte, los datos obtenidos por el sensor se procesan y se extraen una serie una serie *landmarks*, posteriormente se determina si se trata de algún punto característico del entorno observado por primera vez o uno que ya se haya captado antes. En el caso de que sea un punto ya conocido, se introduce en el filtro de Kalman para actualizar la posición. Por el contrario, si se trata de un nuevo punto se añade al EKF como nueva observación y se empleará más adelante para actualizar la estimación de la posición.

A la hora de generar los mapas de los entornos por los que se va navegando, se pueden diferenciar dos tipos de mapeado: denso y disperso. Los métodos de reconstrucción densos, son aquellos donde se evalúa la correspondencia de todos los píxeles en la imagen, y se calcula la disparidad de cada uno. En los dispersos se utilizan los puntos de interés de la imagen y se halla su correspondiente par para determinar la profundidad o distancia al sistema de cámaras.

En este proyecto se analizan técnicas de SLAM con los dos tipos de mapeado, RTAB-Map y RGB-D SLAM emplean mapas densos, a diferencia de ORB-SLAM donde el mapa generado se realiza mediante un mapeado disperso.

3 HECTOR SLAM

Hector SLAM es una implementación *open source* de la técnica de SLAM en 2D que suele ser usada como base en muchos algoritmos de SLAM 3D.

El sensor utilizado por este método es un sensor láser, aunque también se puede usar un dispositivo como Kinect, ya que a través de un tratamiento de la nube de puntos que genera dicho dispositivo se pueden obtener datos similares a los obtenidos por sensor láser. Este tratamiento se puede hacer a través de las librerías de ROS como "*pointcloud_to_LaserScan"* o *"depthimage_to_LaserScan"*. La parte izquierda de la Figura 4 muestra la imagen obtenida por la cámara RGB, con los datos de la nube de puntos y la altura a la que se realizará el barrido láser. En la parte derecha se muestra la nube de puntos en color naranja y el LaserScan (la línea gris sobre la nube de puntos naranja) obtenido con la librería "*depthimage_to_LaserScan"*.

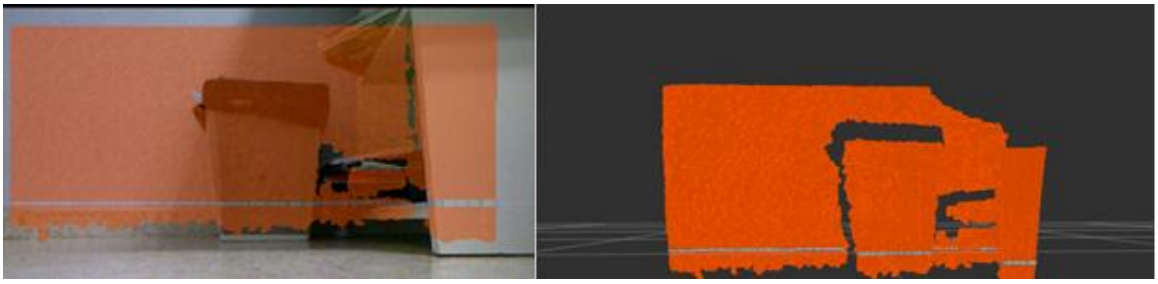

Figura 4: Conversión de nube de puntos de Kinect en barrido láser

En comparación con otras técnicas SLAM 2D cabe destacar que el método Hector SLAM no requiere de la información de la odometría de la plataforma móvil, únicamente emplea para calcular el posicionamiento la información obtenida por los barridos láser. Gracias a la alta tasa de actualización y a la precisión de los sensores tipo LIDAR se consigue un algoritmo que se ejecuta muy rápidamente y con una gran precisión a la hora de calcular el posicionamiento del robot. Este algoritmo es muy utilizado y se ha implementado de forma satisfactoria tanto en robótica móvil terrestre como en UAVs.

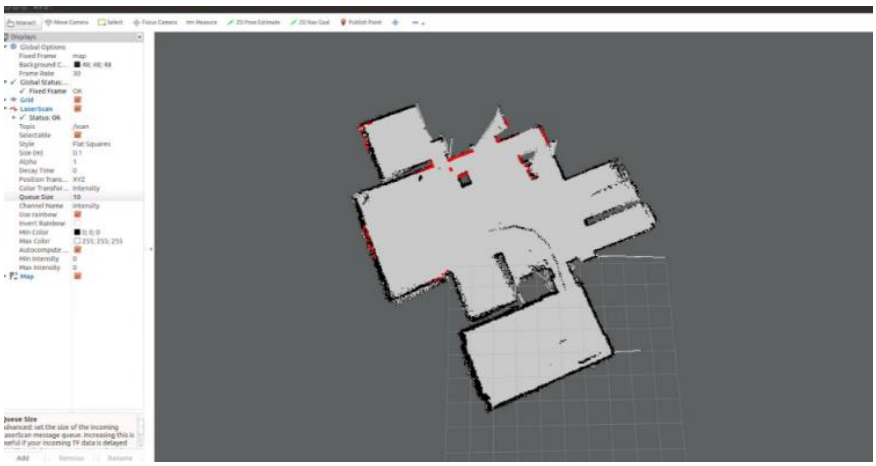

Figura 5: Mapa generado con Hector SLAM a través de Rviz

4 RTAB-MAP (REAL-TIME APPEARANCE-BASED MAPPING)

En este capítulo se describe el funcionamiento del método RTAB-Map(3), el primero de los métodos que se estudiaran en este trabajo. En primer lugar, se presentan las ideas en las que se basa esta técnica para posteriormente explicar el funcionamiento del algoritmo. Por último, se exponen los resultados obtenidos al realizar diferentes pruebas con este método y se explica la interfaz gráfica del programa.

4.1 Método

Uno de los aspectos fundamentales de las técnicas de SLAM es la capacidad de reconocer localizaciones en las que ya se estuvo anteriormente. El proceso capaz de determinar si la observación actual es nueva o pertenece a una localización que ya se visitó con anterioridad se conoce como "Detección de bucle cerrado" (Loop closure detection).

Generalmente, la detección de un bucle cerrado se realiza comparando la observación actual con todas las obtenidas anteriormente, independientemente de la posición estimada del robot. Si al buscar entre las localizaciones conocidas no encontramos ninguna que sea la misma que la que se acaba de obtener, ésta última posición se añade a la base de datos como nueva localización. Sin embargo, si el robot realiza un recorrido muy largo y añade demasiadas localizaciones nuevas,el tiempo de cálculo aumenta considerablemente. En el momento en el que el tiempo de cálculo supere al tiempo de adquisición de las imágenes comenzarán a aparecer retrasos que dificultarán la construcción del mapa en tiempo real.

La técnica RTAB-Map (Real-Time Appearance-Based Mapping) pretende ser capaz de realizar SLAM en tiempo real, sin importar que el recorrido del robot sea demasiado extenso o que el tiempo de la misión llevada a cabo por el robot sea elevado. Se trata de una técnica de SLAM capaz de detectar bucles cerrados en la navegación del robot a partir de las imágenes de la cámara.

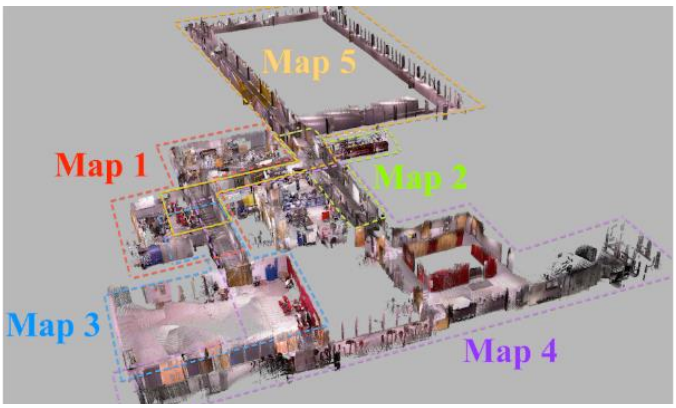

Figura 6: El algoritmo RTAB-Map es capaz de funcionar con recorridos muy largos

Estudios psicológicos afirman que los seres humanos recuerdan mejor los sitios en los que pasan más tiempo, esto crea una relación entre el espacio y el tiempo. En dichas observaciones psicológicas está basado el algoritmo presentado en este capítulo. Por ello, plantea una estrategia de particionado de memoria que pretende asemejarse al funcionamiento de la memoria humana, donde las principales particiones de memoria son la memoria de trabajo del robot (Working Memory, WM), la memoria a largo plazo (Long Term Memory, LTM), la memoria a corto plazo (Short-term memory, STM) y la memoria sensorial (Sensory Memory, SM). Así, 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.

4.1.1 Tratamiento de la memoria

Como se ha comentado anteriormente, el objetivo de este método es ser capazde realizar un mapa y estimar la localización del robot, independientemente de la duración de la misión o de la distancia recorrida. La base del algoritmo es limitar el número de localizaciones que forman un *loop closure detection* satisfaciendo unas restricciones asociadas al tiempo requerido para procesar una imagen adquirida por la cámara, todo ello sin impedir el acceso al mapa completo en el momento que sea necesario.

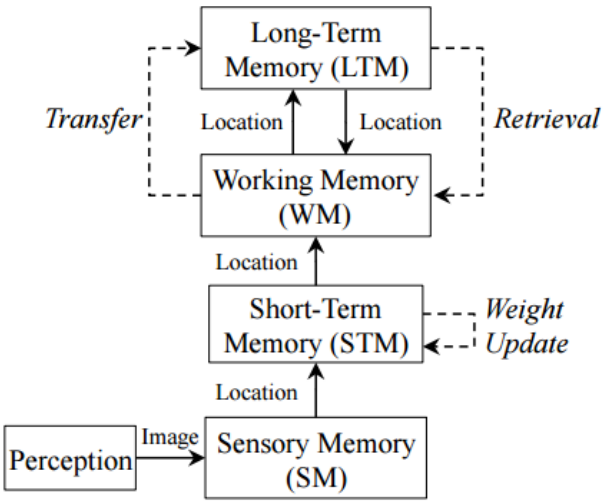

Figura 7: Modelo del tratamiento de memoria usado en RTAB-Map

La Figura 7 presenta un esquema de los diferentes tipos de memoria que se usan en esta técnica. Tal y como se observa en el esquema, partimos de una imagen obtenida del módulo de percepción. El módulo de percepción es el encargado de adquirir las imágenes y enviarlas a la memoria sensorial. Esta memoria procesa las imágenes obtenidas, reduciéndolas y extrayendo las características útiles para encontrar un bucle cerrado. La memoria sensorial crea una nueva localización y la envía a la memoria de corto plazo, donde se van actualizando las nuevas observaciones a través de un proceso llamado "actualizador de pesos". Si el actualizador de pesos considera que la nueva actualización es similar a la última almacenada en la STM, se unen para crear una nueva y se incrementa el valor de su peso.

Tal y como se explicó anteriormente, se parte 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 empleado como peso, de esta forma serán transferidas desde la memoria de trabajo a la memoria de largo plazo aquellas observaciones que tengan menor peso.

La memoria a corto plazo tiene como misión buscar las similitudes que existan entre dos imágenes consecutivas, mientras que la memoria de trabajo 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 está basado en la velocidad del robot y en la frecuencia de adquisición de las localizaciones.

Es importante destacar que todas las observaciones almacenadas en la memoria a largo plazo del robot no se usan para detectar bucles cerrados. No obstante, es importante elegir cuidadosamente las localizaciones que se

almacenan en esta memoria. Almacenar las observaciones según la técnica FIFO (First In First Out), sería un error debido a que, como el algoritmo establece un número máximo de localizaciones que se pueden almacenar mientras se está explorando un entorno, podríamos alcanzar el superar el umbral de tiempo establecido sin llegar a cerrar el bucle haciendo que las localizaciones más antiguas nunca consigan asociarse. Como alternativa, el orden de almacenamiento de las observaciones se elige de forma aleatoria, aunque es preferible mantener en la memoria de trabajo aquellas localizaciones que son más susceptibles de ser observadas.

4.1.2 Algoritmo RTAB-Map

A continuación, se explica el funcionamiento del algoritmo RTAB-Map, el cual puede observarse en forma de pseudo-código en la Figura 8.

Algorithm 1 RTAB-Map

$time \leftarrow$ TIMENOW() ▷ TIMENOW() returns current time
$I_t \leftarrow$ acquired image
$L_t \leftarrow$ LOCATIONCREATION(I_t)
if z_t (of L_t) is a bad signature (using T_{bad}) **then**
  Delete L_t
else
  Insert L_t into STM, adding a neighbor link with L_{t-1}
  Weight Update of L_t in STM (using $T_{\text{similarity}}$)
  if STM's size reached its limit (T_{STM}) **then**
    Move oldest location of STM to WM
  end if
  $\boldsymbol{p}(S_t|L^t) \leftarrow$Bayesian Filter Update in WM with L_t
  Loop Closure Hypothesis Selection ($S_t = i$)
  if $S_t = i$ is accepted (using T_{loop}) **then**
    Add loop closure link between L_t and L_i
  end if
  Join $trash$'s thread ▷ Thread started in TRANSFER()
  RETRIEVAL(L_i) ▷ LTM $\rightarrow$ WM
  $pTime \leftarrow$ TIMENOW() $- time$ ▷ Processing time
  if $pTime > T_{\text{time}}$ **then**
    TRANSFER() ▷ WM $\rightarrow$ LTM
  end if
end if

Figura 8: Pasos en el proceso de detección de un bucle cerrado

En primer lugar, se adquiere una imagen y se crea una nueva localización. Se emplea una bolsa de palabras para crear una "firma" de la imagen. La imagen firmada es representada por un conjunto de palabras visuales que están contenidas en un vocabulario visual que se irá incrementando en las sucesivas iteraciones del algoritmo. En este caso el vocabulario empleado se encuentra "pre-entrenado" para facilitar y disminuir el tiempo de ejecución, ya que entrenar de cero un vocabulario completo es una tarea muy costosa computacionalmente. Las características de estas imágenes (a partir de ahora las llamaremos *features*) son extraídas con el descriptor SURF. SURF transforma la imagen en coordenadas, utilizando una técnica llamada multi-resolución. Consiste en hacer una réplica de la imagen original de forma Piramidal Gaussiana o Piramidal Laplaciana, y obtener imágenes del mismo tamaño, pero con el ancho de banda reducido. Consiguiendo así, un efecto de borrosidad sobre la imagen original, llamado Scale-Space. Esta técnica asegura que los puntos de interés son invariantes en el escalado.

Una vez se ha creado la nueva localización comprobamos si la localización se ha creado correctamente. Si se han extraído pocas características SURF (menos del valor medio de características extraídas por imagen) se considerará que no es válida y se eliminará.Por el contrario, si la localización es válida la introduciremos en la memoria a corto plazo y aplicaremos sobre ella el optimizador de pesos.

Cuando el número de localizaciones en la memoria a corto plazo llega al máximo, la última localización almacenada es transferida a la memoria de trabajo. Esta localización se compara con las otras almacenadas en la WM y se calcula la probabilidad de que se detecte un bucle cerrado mediante un filtro de Bayes. Cuando el filtro de Bayes da una probabilidad muy alta entre la nueva localización y una almacenada en la memoria de trabajo se considera que se ha formado un nuevo bucle cerrado.

Cuando el número de localizaciones en el mapa provoca que el tiempo de asociación sea superior a un umbral de tiempo establecido, significa que las nuevas observaciones tienen poca probabilidad de formar parte del bucle que se está formando en ese momento. Sin embargo, si se detecta un cierre del bucle las localizaciones vecinas serán trasladadas a la memoria de trabajo para ser tenidas en cuenta en el cierre de bucles futuros.

Posteriormente, se produce la "recuperación", se traspasan a la memoria de trabajo aquellas localizaciones vecinas de la que tiene con mayor probabilidad de cerrar el bucle que se encontraban en la memoria a largo plazo. Siel tiempo de procesamiento para detectar un bucle cerrado supera al umbral de tiempo determinado, se produce la "transferencia" que consiste en que localizaciones más antiguas de la WM son transferidas a la LTM. El número de localizaciones transferidas varía en función del número de localizaciones añadidas a la WM en ese ciclo.

4.2 Pruebas realizadas

Se han realizado varias pruebas para comprobar el funcionamiento del algoritmo. Las pruebas han sido realizas en tiempo real empleando como sensor una Kinect.

En primer lugar, al iniciar la aplicación debemos escoger la cámara que vamos a usar para realizar SLAM. En nuestro caso, se trata de un dispositivo Kinect, ya que este algoritmo está pensado para realizar SLAM con la información obtenida a través de una cámara RGB y de un sensor de profundidad. No obstante, ofrece la posibilidad de usar una cámara monocular, pero parece que el uso con este tipo de cámaras no está aún optimizado, ya que aparecían demasiados errores y los resultados no eran para nada satisfactorios.

A continuación, se debe facilitar al programa la calibración de la cámara y tenemos la posibilidad de modificar una serie de parámetros tales como los drivers que usará la cámara, la frecuencia de captura de imágenes o si queremos o no que guarde la nube de puntos generada, entre otras.

La Figura 9 muestra la interfaz que aparece al ejecutar la aplicación. En ella se muestran cuatro ventanas que nos irán proporcionando diferente información sobre las imágenes que la cámara va adquiriendo. En la esquina superior izquierda se encuentra las ventanas de *loop closure detection* en ellas se observa las diferentes localizaciones nuevas que se van creando y añadiendo al mapa y como se asocian. Aparecen sobre estas ventanas círculos de colores que representan las características (features) extraídas de las imágenes, según el color que tengan en ese instante tiene un significado u otro :

- Azul: La feature ha sido localizada en una imagen anterior. Cuando la mayoría son de color azul significa que la imagen actual se ha unido con la anterior, aumentando el peso de la localización, tal y como se explicó en el apartado anterior.
- Rojo: Feature encontrada en el diccionario.
- Verde: Nueva feature.
- Amarillo: Nueva feature que aparece varias veces en la imagen.
- Rosa: Feature encontrada en dos imágenes.

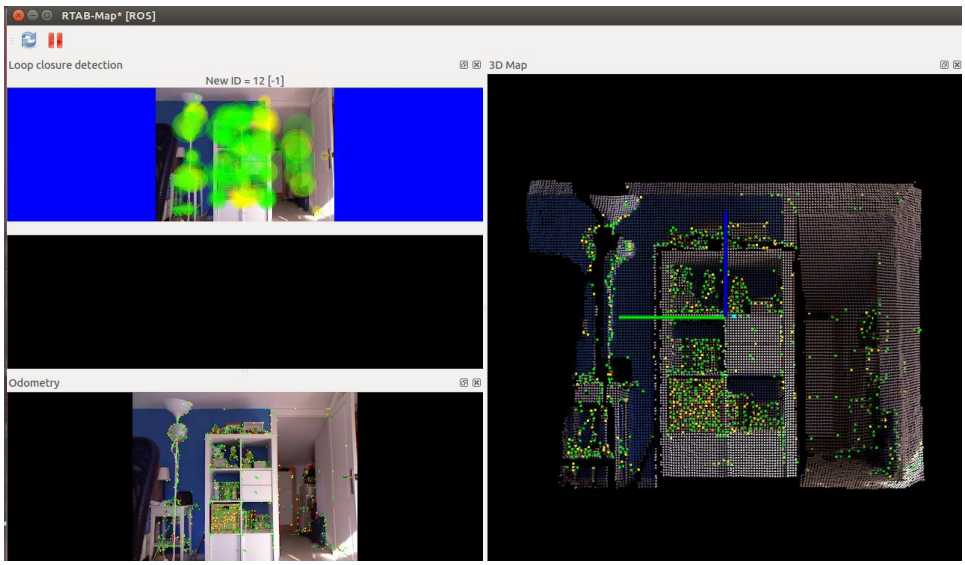

Figura 9: Interfaz de RTAB-Map

En la parte inferior izquierda se encuentra la ventana en la que se va mostrando la odometría. Si el fondo se volviera de color rojo significaría que se ha perdido la odometría y el programa sería incapaz de continuar construyendo el mapa. Para reanudar la creación de dicho mapa debemos mover la cámara hasta la última posición en la que tenía odometría y esperar a que el algoritmo vuelva a poder estimar su posición.

Por último, en la parte derecha se muestra el mapa en 3D que se va generando en forma de nube de puntos, el cual se irá actualizando conforme se vayan adquiriendo nuevas localizaciones. Cada vez que se detecta un nuevo bucle cerrado, una nueva restricción es añadida al grafo del mapa, en ese momento el grafo se optimiza y minimiza los errores del mapa.

La Figura 9 muestra el inicio de la primera prueba. Se observa que todas las features son de color verde o amarillo, es decir todas son nuevas, debido a que es lo primero que se ha enfocado con la cámara y aún no nos hemos empezado a mover. Este es el motivo también de que aparezca en negro la segunda venta del *loop closure detection*, como se ha explicado anteriormente en esta ventana aparecen las localizaciones antiguas con las que se asocian las localizaciones actuales, puesto que el sistema acaba de iniciarse no tenemos ninguna localización almacenada.

La Figura 10 es el resultado obtenido después de que se haya recorrido la habitación con la cámara para crear el mapa. Cabe destacar que ahora todas las features son rojas, rosas o azules, es decir todas son reconocidas por el algoritmo, puesto que ya se ha pasado antes por esos puntos y el programa ha guardado las localizaciones para construir el mapa. Por ello, la ventana que antes aparecía en color negro ahora muestra la localización anterior que se asocia con la actual. Aparecen también unas líneas celestes que unen las features de las dos ventanas, esas líneas muestran la correspondencia entre las diferentes features localizadas.

Por último, en la ventana del mapa 3D generado ya se ha obtenido el mapa completo de toda la habitación. Se puede observar que aparecen una serie de puntos de color turquesa unidos con unas líneas, estos muestran el recorrido seguido por la cámara.

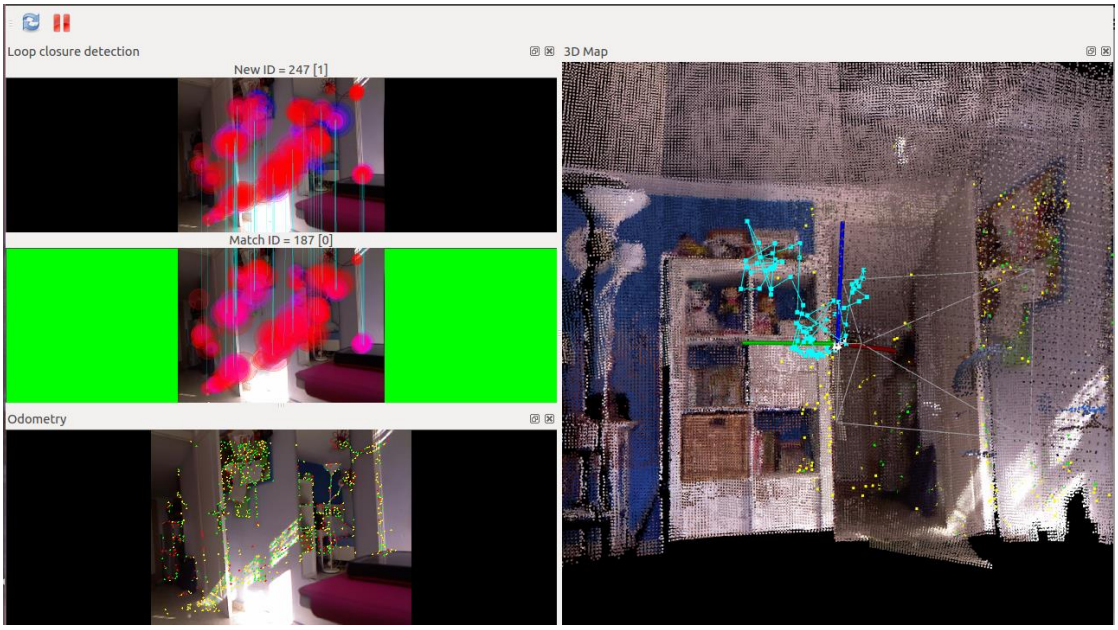

Figura 10:Mapa realizado con RTAB-Map

La habitación en los que se realizó la primera prueba era muy pequeña y se quiso probar el método en una habitación que fuera de mayor dimensión para comprobar el funcionamiento con espacios más abiertos.

La Figura 11 muestra una captura del experimento en curso. Se puede observar en la ventana donde se genera el mapa que el recorrido de la cámara es superior al de la prueba anterior.

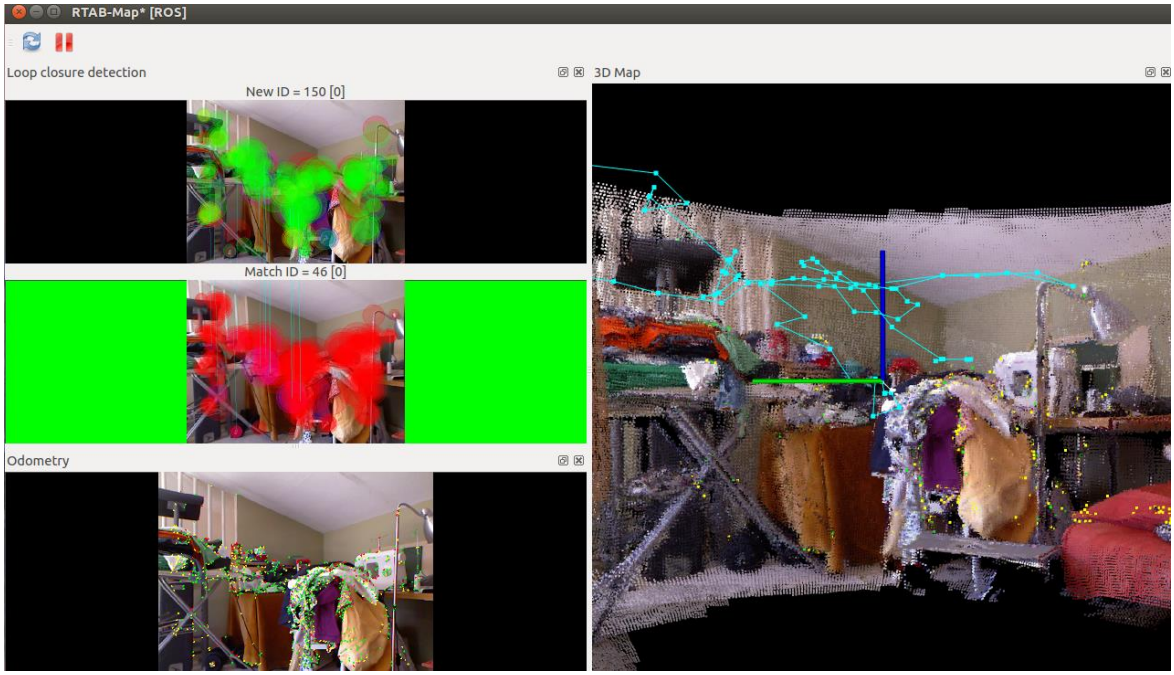

Figura 11: Mapa realizado con RTAB-Map

Finalmente, el resultado final de la prueba es el que presenta la Figura 12.

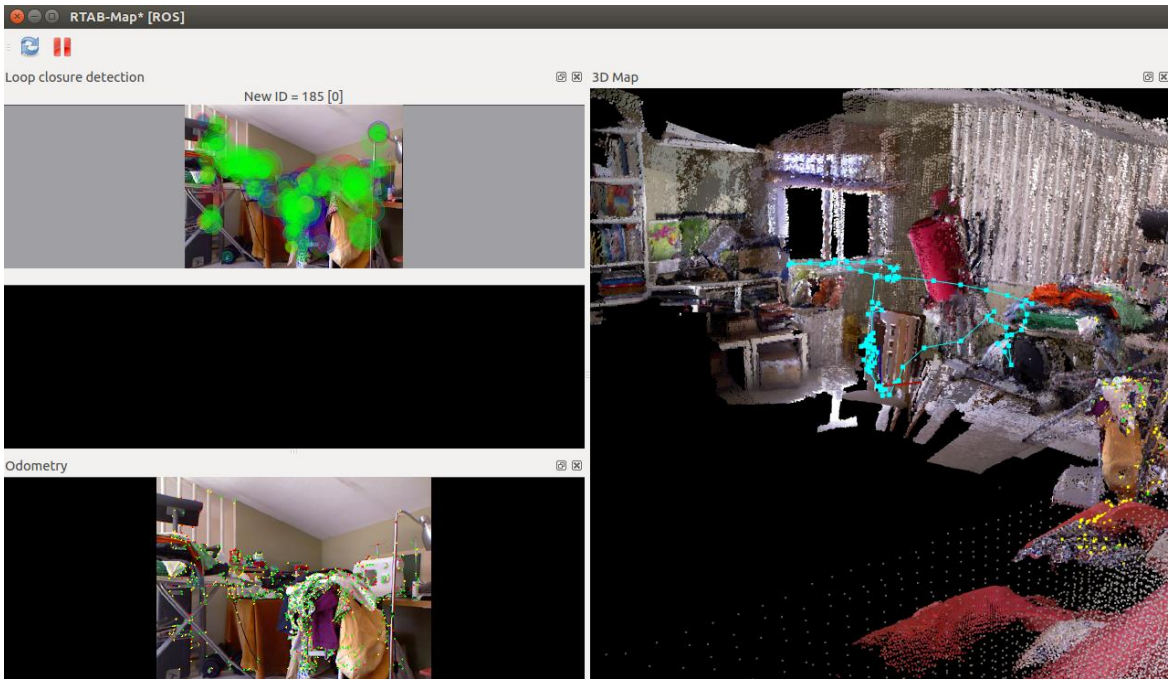

Figura 12:Mapa realizado por RTAB-Map

Tal y como se comentó anteriormente el programa te da la opción de guardar la nube de puntos generada durante la ejecución del algoritmo. En la Figura 13 se observa la nube de puntos generada para el último caso.

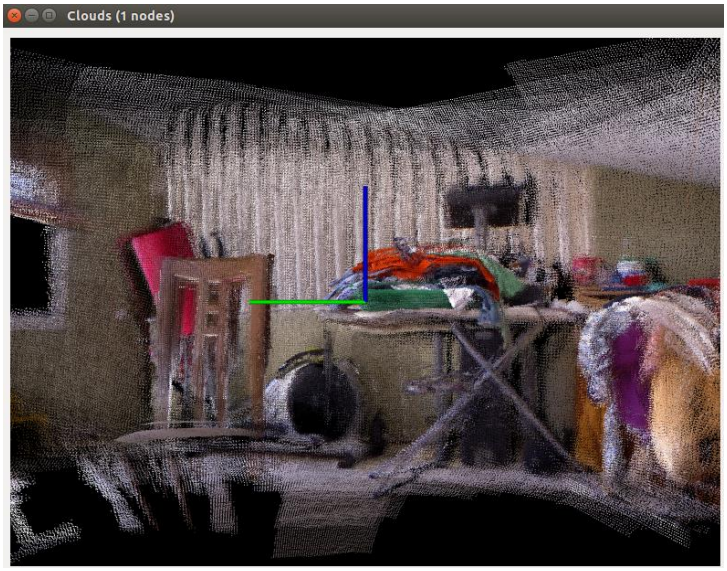

Figura 13: Nube de puntos obtenida por el método RTAB-Map

La Figura 14: Mapa generado por RTAB-Map visto desde arribaFigura 14 muestra el mapa generado desde arriba de un tercer experimento, con la intención de comprobar cómo se realiza el loop-closure en este algoritmo.

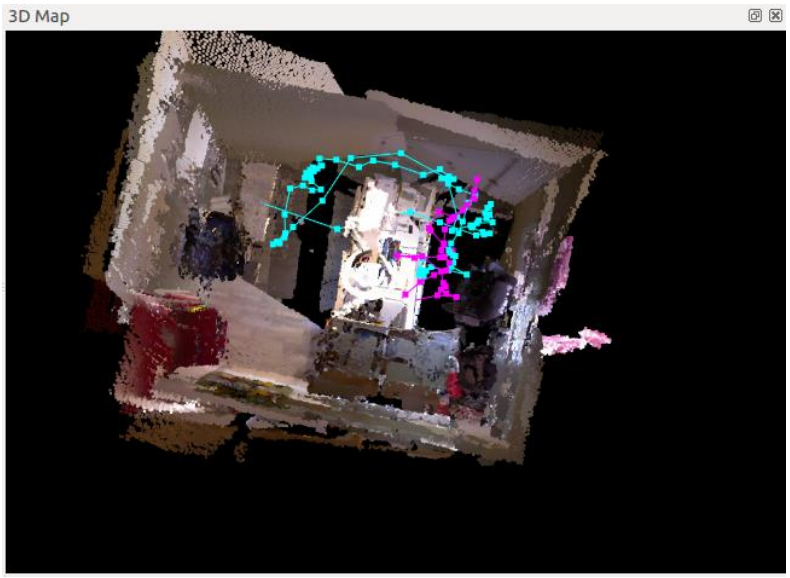

Figura 14: Mapa generado por RTAB-Map visto desde arriba

En general, los resultados obtenidos con este método son muy satisfactorios. El algoritmo se ejecuta rápido y sin apreciarse retrasos. Los mapas generados son bastante fieles a la realidad y si la cámara no se movía a mucha velocidad era capaz de ir actualizando la odometría con bastante rapidez. Además, tal y como se pretendía, el algoritmo era capaz de ejecutarse durante mucho tiempo y tanto en recorridos pequeños como el primero, como en recorridos más largos como el segundo experimento.

5 ORB-SLAM

Este capítulo describe el algoritmo ORB-SLAM(4). En primer lugar, se explica de forma general el funcionamiento del método para posteriormente entrar de forma más detallada en cada uno los procesos que forman parte del algoritmo. Por último, se muestran una serie de experimentos realizados empleando esta técnica.

5.1 Método

El método descrito en este capítulo se trata de un algoritmo de SLAM monocular basado en el reconocimiento de características (features), capaz de operar en tiempo real, en entornos tanto de pequeño como de gran tamaño, y que se puede usar tanto en exteriores como en interiores. La idea principal de la generación de mapas en esta técnica es emplear los mismos features para todas las tareas de SLAM: seguimiento, creación del mapa, localización y detección de bucles cerrados. Se seleccionarán estratégicamente puntos y fotogramas clave (a partir de ahora se referirá a ellos como *keyframes*) para generar los mapas, estos solo variarán si el contenido de la escena cambia.

El SLAM basado en visión tiene el objetivo de estimar la trayectoria de la cámara a la vez que reconstruye el entorno por el que va navegando. Uno de los conceptos más importantes en este método de SLAM es el *Bundle Adjustement, BA*. Se trata de una técnica capaz de realizar estimaciones muy precisas de posición y una reconstrucción geométrica que proporciona un marco de asociaciones y unas buenas suposiciones iniciales. En ORB se utiliza para optimizar los mapas y las trayectorias calculadas cuando se detecta un loop closure.Actualmente, se pueden conseguir resultados muy precisos en tiempo real sin que el coste computacional sea demasiado elevado.

Se puede concluir que los puntos más destacados de esta técnica de SLAM son:

- Usar los mismos features para todas las tareas de SLAM. Esto provoca que el sistema sea más eficiente, simple y fiable.
- Capaz de realizar operaciones en tiempo real en entornos de gran tamaño. Es el resultado de usar un grafo de covisibilidad.Tanto el seguimiento como el mapeo se focalizan en el área covisible, independientemente del tamaño del mapa completo, consiguiendo así explorar entornos amplios sin aumentar el tiempo de computación.
- La estrategia para detectar los bucles cerrados de visión en tiempo real estábasada en la optimización de un de grafo llamadoGrafo Esencial (Essencial Graph), se trata de una de las ideas novedosas de este método y se explica más adelante.
- Localización en tiempo real en localizaciones que se han visitado con anterioridad independientemente de que existan significantes cambio en la iluminación o en el ángulo de visión.

5.1.1 Diagrama del algoritmo ORB-SLAM

En Figura 15 se muestra el diagrama del algoritmo ORB-SLAM. Se pueden diferenciar fácilmente cinco módulos: Tracking, Local mapping, Loop closing, Map y Place recognition.

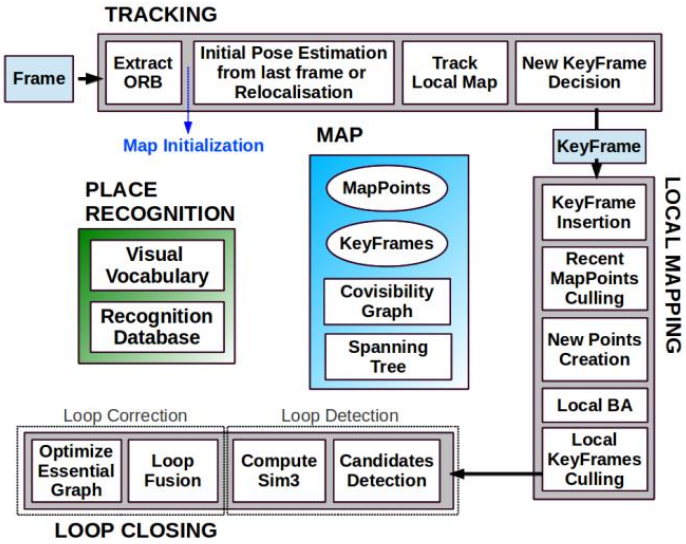

Figura 15: Módulos del algoritmo ORB-SLAM

Este método está formado por tres hilos que se ejecutan en paralelo: Tracking, Local mapping y Loop closing. El hilo de tracking es el encargado de localizar la posición de la cámara en cada imagen y decidir cuándo es necesario incluir un nuevo keyframe. Cuando el sistema pierde el seguimiento (por causa de un movimiento brusco, por ejemplo) el modulo Place recognition, es el encargado de representar una relocalización global. Una vez se ha conseguido una estimación inicial de la cámara y se han asociado los features encontrados, se recupera el mapa local usando el grafo de covisibilidad de los keyframes que estaban almacenados en la base de datos.

El segundo hilo en paralelo es el Local mapping el cual es el encargado de procesar nuevos keyframes y representar los BA locales para conseguir una reconstrucción óptima de los alrededores de la posición de la cámara. Para aquellos ORB que no están asociados aún pertenecientes al nuevo keyframe, se buscan nuevas correspondencias en los keyframes conectados del grafo de covisibilidad para triangular nuevos puntos. Posteriormente, sobre la base de la información recopilada durante el seguimiento, se aplica un filtrado muy exigente de puntos almacenados con el fin de conservar sólo los puntos de alta calidad. El módulo de Local mapping es también el encargado de eliminar aquellos keyframes almacenados que sean redundantes.

El tercer hilo es Loop closing, el cual buscará la existencia de bucles cada vez que obtengamos un nuevo keyframe. Finalmente, el bucle cerrado se incorpora al grafo del mapa global.

5.1.1.1 Mapa de puntos y keyframes

Cada punto perteneciente al mapa contiene la siguiente información:

- Su localización en 3D con respecto al sistema de referencia global.
- El vector de dirección de vista. Se trata de un vector unitario que tiene la dirección que hay al unir el punto con el keyframe al que está asociado.
- El descriptor ORB asociado.
- Las distancias máxima y mínima a las cuales el punto puede ser observado, de acuerdo a la escala límite invariante de los descriptores ORB.

Cada keyframe contiene:

- La transformación del sistema de coordenadas del mundo real al sistema de referencia de la cámara.
- Los parámetros intrínsecos de la cámara, tales como la longitud focal y el punto principal.
- Todas las features ORB extraídas de la imagen, estén asociadas o no a un punto del mapa.

En esta técnica de SLAM, la filosofía es crear puntos del mapa y keyframes de forma generosa, ya que

posteriormente se aplicará un filtro muy exhaustivo que eliminará todos aquellos keyframes redundantes y aquellos puntos del mapa que tenga una asociación incorrecta. Como resultado, se obtiene un mapa flexible que puede expandirse mientras se está explorando el entorno.

5.1.1.2 Grafo de covisibilidad y Grafo Esencial

La información de covisibilidad es muy útil en numerosas tareas de nuestro sistema, esta información es representada en forma de grafo. Cada nodo de este grafo representa un keyframe, los arcos del grafo indican que los dos keyframes que están unidos coinciden en la observación de puntos del mapa. Para que un arco una dos nodos tiene que haber al menos 15 puntos en común entre los dos keyframes. El grafo de covisibilidad se trata de un grafo ponderado, los pesos de los arcos serán el número de puntos del mapa comunes entre los dos nodos unidos.

Con el fin de corregir los bucles cerrados se realiza una optimización del grafo que provocará que el error se encuentre distribuido a lo largo de todo el grafo. Incluir todos los arcos provocaría que fuera un grafo demasiado denso y no se pudiera observar con claridad, por ello se crea el Grafo Esencial. Dicho grafo contiene todos los nodos (cada nodo es un keyframe) pero con menos arcos, de esta forma se conserva una consistente red de trabajo capaz de producir unos resultados precisos. La Figura 16 muestra el grafo de covisibilidad de un sistema y el Grafo Esencial creado a partir de éste.

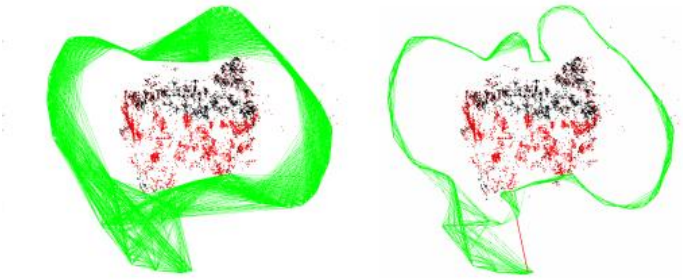

Figura 16: Grafo de covisibilidad y Grafo Esencial

El sistema va construyendo de forma incremental un árbol de expansión que abarca desde el keyframe inicial, que consiste en un subgrafo del grafo de covisibilidad con el mínimo número de arcos posibles. Cuando se inserta un nuevo keyframe, se incluye en el árbol de expansión y se une con el keyframe que más puntos comparta. En el momento que se aplique el filtrado de keyframes, y se eliminen los redundantes el sistema actualiza los arcos del árbol de expansión. En la Figura 17 se observa el árbol de expansión creado a partir de los grafos representados en la Figura 16.

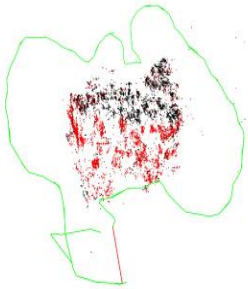

Figura 17: Árbol de expansión

El Grafo Esencial contiene al árbol de expansión, el conjunto de arcos de arcos con mayores pesos del grafo de covisibilidad y los arcos de los bucles cerrados, todo esto convierte al Grafo esencial en una red de trabajo muy potente.

5.1.1.3 Bolsa de palabras y módulo de reconocimiento de localizaciones

En este método de SLAM también se ha empleado un vocabulario de palabras visuales. A diferencia de la técnica explicada en el capítulo anterior, la bolsa de palabras empleada está creada offline con el descriptor ORB. Si se entrena el vocabulario con suficientes palabras puede servir en diferentes entornos dando unos resultados satisfactorios. El sistema construye de forma incremental una base de datos que almacena para cada palabra visual del vocabulario en que keyframe ha sido detectada. Tras realizar el proceso que se ha comentado anteriormente de filtrado de keyframes, la base de datos se actualiza.

Es posible que algunas palabras estén solapadas entre dos keyframes, esto provocaría que al consultar la base de datos no existiera un único keyframe con una puntación alta. Para evitarlo, se agrupan los keyframes que están conectados en el grado de covisibilidad.

5.1.2 Algoritmo ORB-SLAM

5.1.2.1 Inicialización del mapa

El objetivo a la hora de inicializar un mapa es obtener la posición relativa entre dos frames para poder triangularizar un conjunto de puntos que formen parte del mapa. El algoritmo debe ser capaz de conseguir esto de forma autónoma e independientemente del entorno en el que se encuentre.

Para ello, este algoritmo plantea ejecutar en paralelo dos modelos geométricos, uno con la matriz homógrafa para escenas planas y otro con la matriz fundamental para escenas no planas. El método para reconocer que modelo se ha de aplicar a la imagen es heurístico y se detalla a continuación:

1. Se encuentran las correspondencias iniciales. Se extraen todas las características ORB del frame actual.
2. Se ejecutan los dos modelos a la vez.

$$\mathbf{x}_c = \mathbf{H}_{cr}\,\mathbf{x}_r \qquad\qquad \mathbf{x}_c^T\,\mathbf{F}_{cr}\,\mathbf{x}_r = 0$$

   De antemano el número de iteraciones de cada modelo está determinado. En cada iteración obtendremos una puntuación que viene determinada por la siguiente ecuación:

$$S_M = \sum_i \left(\rho_M\left(d_{cr}^2(\mathbf{x}_c^i, \mathbf{x}_r^i, M)\right) + \rho_M(d_{rc}^2(\mathbf{x}_c^i, \mathbf{x}_r^i, M))\right)$$

$$\rho_M(d^2) = \begin{cases} \Gamma - d^2 & \text{if} \quad d^2 < T_M \\ 0 & \text{if} \quad d^2 \geq T_M \end{cases}$$

   Finalmente, se conservan aquellas matrices homógrafas y fundamentales con mayor puntuación.

3. Se elegirá la matriz homógrafa si $R_H > 0.45$, en caso contrario se eligirá el modelo de matriz

$$R_H = \frac{S_H}{S_H + S_F}$$

   fundamental.
4. Una vez seleccionado el modelo se recuperan las hipótesis del movimiento asociado. Se intentará triangular directamente con las soluciones obtenidas anteriormente y se buscará cual es la mejor de todas ellas. En caso de no encontrar ninguna adecuada se debe volver al paso 1.
5. Finalmente se representa el BA.

5.1.2.2 Tracking

Para realizar el tracking o seguimiento, lo primero que debemos hacer una estimación de la posición inicial. Podemos partir de un frame anterior si el resultado fue satisfactorio, por el contrario, si el seguimiento está perdido se debe convertir todo el frame en palabras visuales y consultar en la base de datos de keyframes para buscar candidatos que faciliten el posicionamiento.

Una vez se ha obtenido una estimación de la posición de la cámara y un conjunto de features asociadas, se puede proyectar el mapa local al que corresponde el frame y buscar más correspondencias entre los puntos del mapa. Para buscar estos puntos se utiliza el grafo de covisibilidad. Una vez se encuentran las correspondencias entre los puntos, se obtiene el mapa actualizado.

Lo último que realiza este módulo es decidir si el frame actual se convertirá o no en un keyframe. Para insertar un nuevo keyframe se han de cumplir las siguientes condiciones:

- Deben haber pasado más de 20 frames desde la última relocalización global.
- Local mapping debe estar desactivado o tienen que haber pasado más de 20 frames desde la última inserción de un nuevo keyframe.
- El frame actual debe haber localizado al menos 50 puntos
- El actual frame debe tener menos del 90% de los puntos que contiene el keyframe de referencia.

Con estas condiciones se asegura que ha habido cambios visuales, que ha habido una buena relocalización y que se ha realizado un buen seguimiento. Además, con la segunda condición nos aseguraremos de no incluir un nuevo keyframe mientas el Local mapping este ocupado.

5.1.2.3 Local Mapping

Este módulo es el encargado de insertar los nuevos keyframes. Lo primero que debe hacer es actualizar el árbol de expansión añadiendo un nuevo nodo y creando los arcos necesarios.

Como ya se ha comentado en otros apartados, uno de los aspectos característicos de este método es el filtrado exhaustivo de puntos y keyframes que realiza, es este módulo el encargado de llevarlo a cabo. Para que los puntos sigan formando parte del mapa tienen que superar ciertas condiciones durante los tres primeros keyframes que se añadan después de que el punto comience a formar parte del mapa. Las condiciones son la siguientes:

- El módulo de seguimiento debe encontrar al punto en cuestión en más del 25% de los frames en los que se prevé que será visible.
- El punto tiene que poderse observar en al menos los tres siguientes keyframes desde la incorporación del punto al mapa.

Una vez se ha añadido al mapa el punto podrá ser eliminado si en algún momento forma parte de menos de tres keyframes.

Los keyframes también deben cumplir unas condiciones para no ser eliminados. Se eliminarán aquellos keyframes que el 90% de sus puntos de mapa hayan sido observados con la misma o menor escala en al menos otros 3 keyframes. La condición de la escala asegura que siempre mantendremos aquellos keyframe que sean más precisos.

Por último, este módulo es el encargado también del local BA. El local BA optimiza el keyframe actual, todos los que estén conectados a él en el grafo de covisibilidad y todos los puntos de mapa pertenecientes a eso keyframes.

5.1.2.4 Loop closing

La función de este módulo es intentar detectar bucles (y cerrarlos) entre el keyframe actual y el último procesado por el módulo Local mapping.

En primer lugar, se detectan todos aquello bucles que puedan ser candidatos a estar cerrados. Para ello, se calcula las similitudes que hay entre la bolsa de palabras del keyframe actual y las de todos sus vecinos del grafo de covisibilidad que tengan un peso de los arcos superior a 30. Obtendremos una puntuación mínima y se comparará con la base de datos del reconocimiento de localizaciones para descartar todos aquello keyframes que tengan una puntuación menor a la mínima. Para aceptar un bucle como candidato, se debe detectar consecutivamente 3 candidatos de bucle que sean consistentes, es decir que todos sus keyframes estén conectados en el grafo de covisibilidad.

Posteriormente, se calcula una trasformación entre el keyframe actual y el que keyframe del bucle, lo cual da información sobre el error que tiene el bucle. Además, esta comprobación sirve como validación geométrica del bucle.

Para cerrar el bucle eficazmente, se optimiza el grafo a través del Grafo Esencial, que se encarga de distribuir el error a lo largo de todo el bucle. Tras la optimización cada punto del mapa es transformado de acuerdo a las correcciones de alguno de los keyframe en lo que se observe el mismo.

5.2 Pruebas realizadas

Para comprobar el funcionamiento de este algoritmo se han realizado una serie de pruebas. Al igual que el método anterior, se ha usado el framework ROS para ejecutar la interfaz de esta técnica. El sensor utilizado ha sido una Kinect, aunque al tratarse de un método monocular sólo hemos empleado la información de la cámara de color, ignorando el resto de información que podemos obtener al tratarse de una cámara RGB-D.

La interfaz del este algoritmo está formada por dos ventanas. La primera de ellas es la que aparece en la Figura 18 y en ella se muestra el frame actual de la cámara. En la parte inferior de la ventana aparece información de interés sobre el estado del algoritmo:

- Modo. En la foto podemos observar que está en modo SLAM.
- Número de keyframes.
- Número de puntos del mapa.
- Número de asociaciones encontradas.
- Posición en x e y del punto en el que se coloque el cursor sobre la ventana.
- Información RGB del punto en el que se coloque el cursor sobre la ventana.

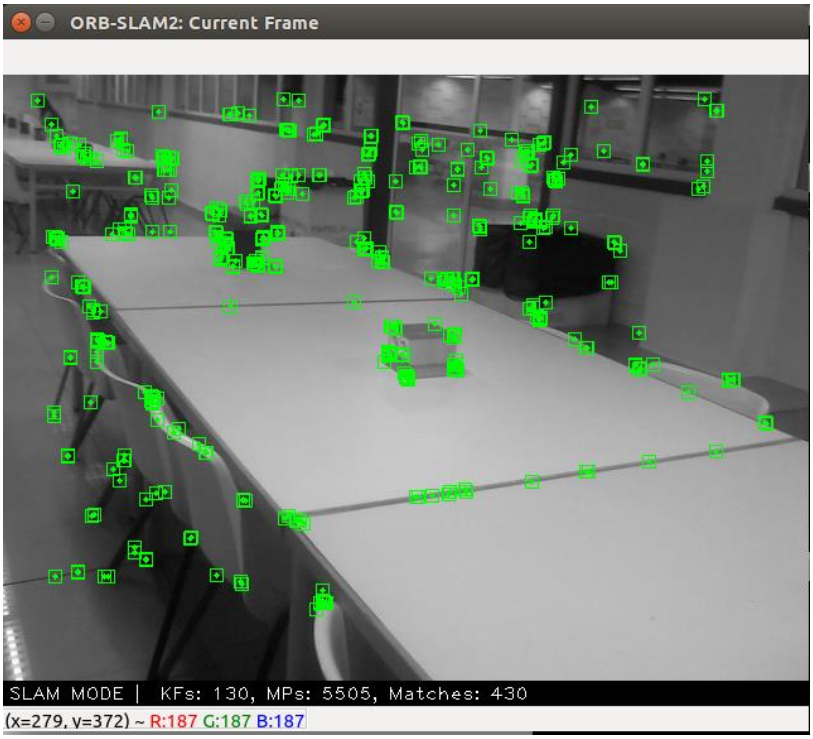

Figura 18: Ventana de visualización de frames.

La segunda ventana es la que se muestra en la Figura 19. Como se puede observar, contiene el mapa de puntos del entorno que se está explorando. En ella aparecen varias opciones que nos permiten hacer un seguimiento de la cámara, elegir qué información se desea que se muestre por pantalla y cambiar de modo de funcionamiento. Además, en la parte central se muestra el mapa que se va creando conforme el robot navega por el entorno. Los elementos que aparecen en el mapa son los siguientes:

- Rectángulo verde: Posición actual de la cámara.
- Rectángulos azules: Keyframes.
- Puntos de color rojo: Puntos del mapa local actual.
- Puntos negros: Puntos del mapa.

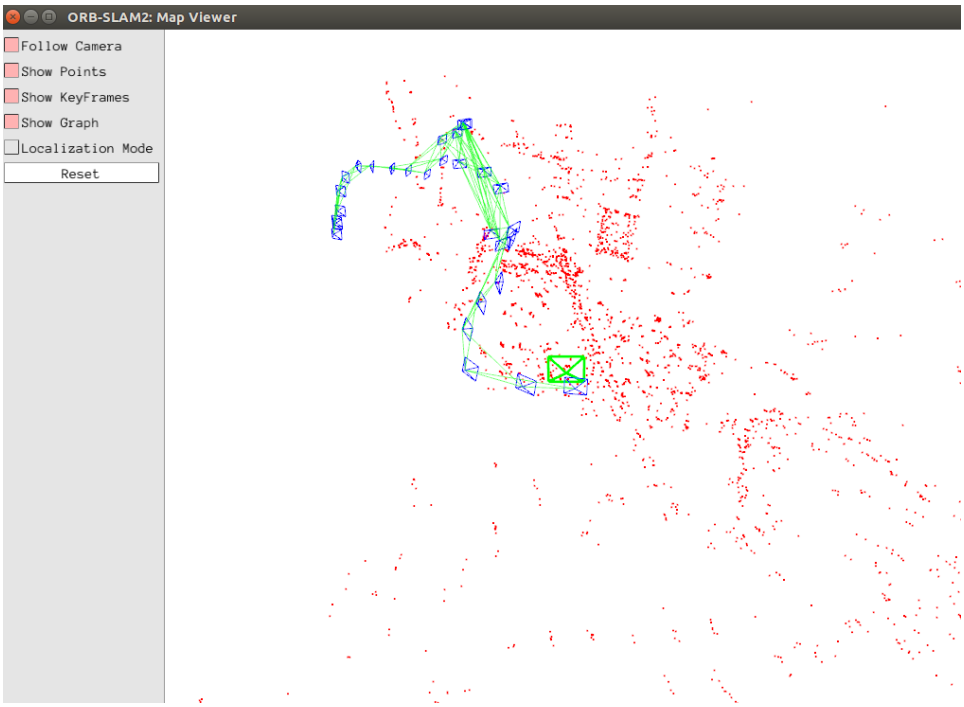

Figura 19: Visor del mapa de puntos

Al realizar las pruebas con este algoritmo se ha notado que es bastante sensible a los movimientos bruscos de la cámara y tiene dificultadas para la relocalización. Sin embargo, si la cámara se mueve de forma suave algoritmo se ejecuta rápido y sin apreciarse retrasos. Para obtener unos buenos resultados, es necesario dejar la cámara inmóvil durante varios segundos al principio para que la inicialización del mapa se ejecute correctamente.

6 LSD-SLAM (LARGE- SCALE DIRECT MONOCULAR SLAM)

En este capítulo se describe el funcionamiento del método LSD-SLAM(5). En primer lugar, se presentan las ideas en las que se basa esta técnica para posteriormente explicar el funcionamiento del algoritmo. Por último, se exponen los resultados obtenidos al realizar diferentes pruebas con este método y se explica la interfaz gráfica del programa.

6.1 Método

La técnica de SLAM presentada en este capítulo funciona de forma completamente diferente a las estudiadas en este trabajo. El método LSD-SLAM (Large- Scale Direct Monocular SLAM) permite construir mapas de gran escala. En lugar de utilizar features opera directamente sobre los contrastes de las imágenes tanto para la localización como para el mapeo. La geometría de los mapas se estima usando filtros sobre las imágenes adquiridas en escala de grises.

Al contrario que las técnicas basadas en features (la posición de la cámara y la geometría del mapa es calculado en teniendo en cuenta unicamente features observadas), los métodos directos emplean toda la información de la imagen. Esto supone que se obtiene mucha más información sobre la geometría y el entorno, lo que puede ser muy útil para los robots o para las aplicaciones de realidad aumentada.

Al igual que en la técnica de SLAM del capítulo anterior, el mundo es representado por un número de keyframes conectados por restricciones de posición, los cuales pueden se optimizados usando un grafo de optimización.

6.1.1 Algoritmo LSD- SLAM

El algoritmo se puede dividir en tres grandes bloques: Tracking (Seguimiento), Depth map estimation (Mapa de estimaciones de intensidad) y optimización del mapa, tal y como se muestra en la Figura 20.

El inicio del algoritmo se realiza con un mapa de intensidad aleatorio. Unos pocos segundos después, cuando la cámara se ha movido ligeramente, el algoritmo bloquea la configuración y tras una pareja de keyframes el algoritmo converge a una correcta configuración.

La representación del mapa es un grafo de keyframes.

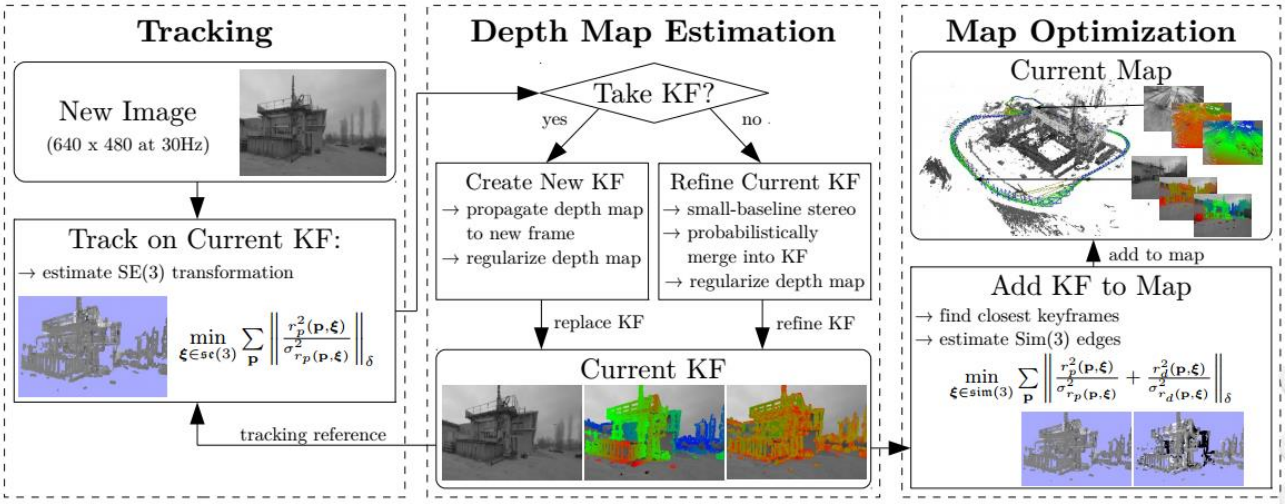

Figura 20: Algoritmo LSD-SLAM

6.1.1.1 Tracking

Este bloque es el encargado de realizar el seguimiento de las nuevas imágenes adquiridas por la cámara. Estas imágenes, son estimadas como un sólido rígido respecto al actual keyframe, usando la posición del anterior frame para la inicialización.

6.1.1.2 Estimación del mapa de intensidad

Este bloque usa los frames obtenidos para refinar o sustituir el actual keyframe. La intensidad es definida mediante un filtrado realizado pixel a pixel.

Si la cámara se mueve demasiado lejos del mapa existente es necesario crear un nuevo keyframe del frame mas reciente que se haya obtenido en el bloque de tracking.Una vez se ha elegido el frame que se convertirá en keyframe, su intensidad se inicializa proyectando los puntos desde el anterior keyframe sobre este. Por último, se reemplaza el keyframe y este se usa para el seguimiento de los nuevos frames.

Aquellos frames que se obtienen pero que no se convierten en keyframe se utilizan para refinar el keyframe actual. El resultado se incorpora al mapa actual de intensidad, lo que significa que se añaden al mapa nuevos píxeles.

6.1.1.3 Optimización del mapa

El mapa, que consiste en una serie de keyframes unido por unas restricciones, el cual es continuamente optimizado en segundo plano por el grafo de optimización.

6.2 Pruebas realizadas

El algoritmo se ha probado mediante varias pruebas en tiempo real utilizado como cámara Logitech HD WEBCAM C525. Al igual que el resto de métodos estudiados en este trabajo, la interfaz se inicia a través de ROS. Dicha interfaz está formada por dos ventanas. La primera de ellas muestra el mapa de intensidad, tal y como se aprecia en la Figura 21. En la parte inferior se encuentra información de interés con respecto al programa, muestra la tasa de actualización del mapa y del seguimiento, la densidad de puntos o el número de keyframes, entre otros datos.

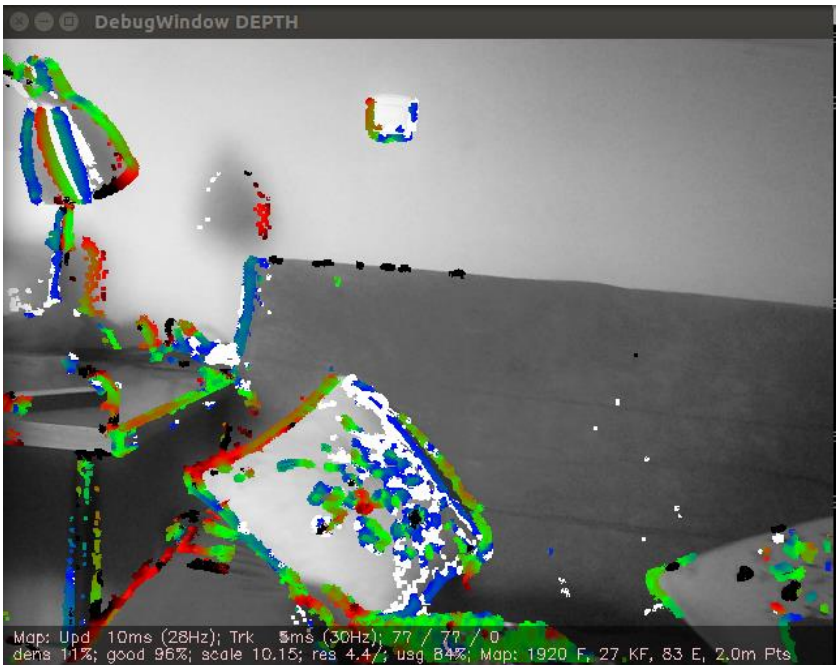

Figura 21: Visor del mapa de intensidad LSD-SLAM

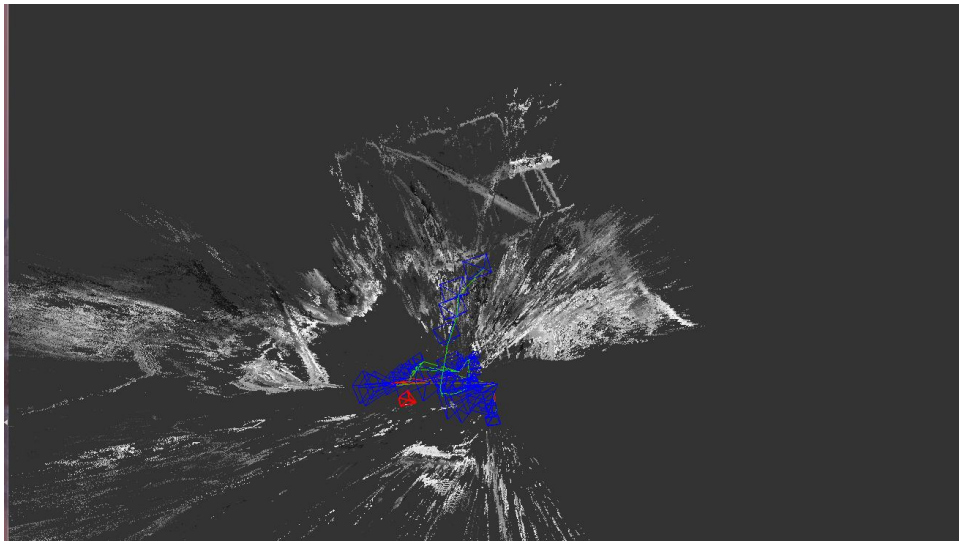

Figura 22:Visor de la nube de puntos generada LSD-SLAM

La segunda ventana que aparece al ejecutar el algoritmo es la encargada de mostrar el mapa en forma de nube de puntos que se va generando durante la navegación. La Figura 22muestra la nube de puntos correspondiente a una de las pruebas realizadas. En color azul se representan los keyframes seleccionados por el algoritmo, en rojo la posción actual de la cámara y la línea de color verde es el recorrido realizado.

A continuación, se muestra otra de las pruebas realizadas. La Figura 23 corresponde al mapa de intensidad y la

Figura 24 a la nube de puntos generada al finalizar la adquisición de imágenes de la cámara.

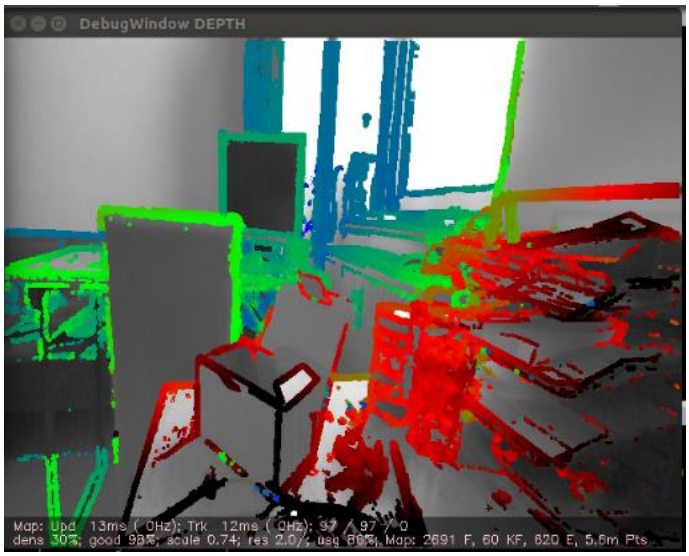

Figura 23: Mapa de intensidad obtenido

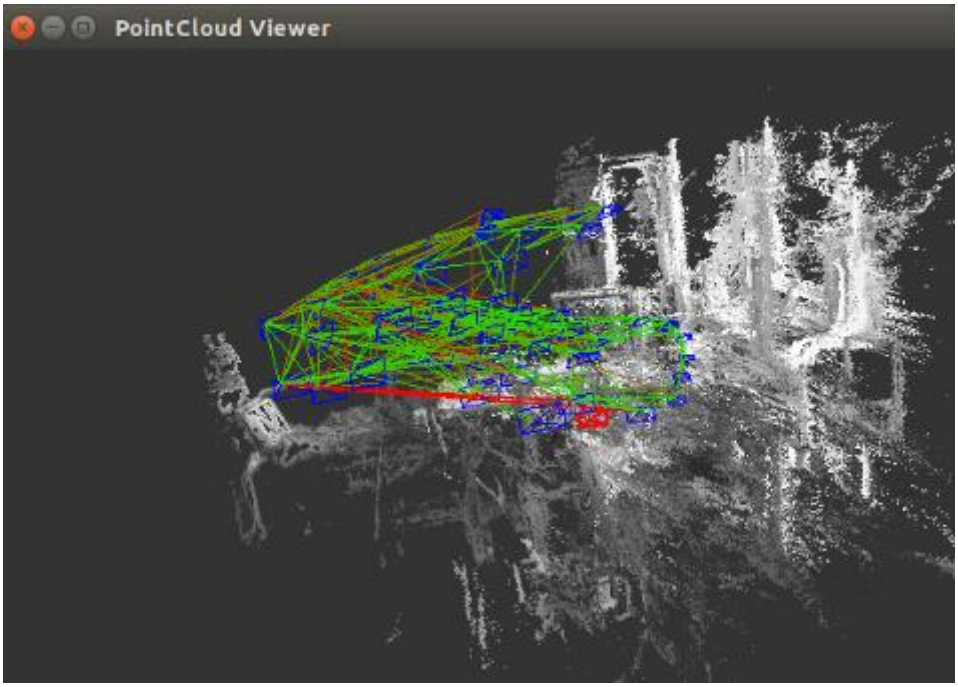

Figura 24: Nube de puntos obtenida al finalizar la navegación

Al realizar los experimentos con este algoritmo se ha detectado un elevado tiempo de ejecución, ya que era necesario mover la cámara exageradamente lento y aun así se apreciaban retrasos. Pocas veces en las que se perdió la localización el algoritmo fue capaz de recuperarlo. Obtener una densa nube de puntos es bastante complicado.

7 RGB-D SLAM

Este capítulo describe el método RGB-D SLAM (6), comenzando con una explicación del funcionamiento del algoritmo y posteriormente se muestra la interfaz del método en ROS y se comentan brevemente algunos experimentos realizados.

7.1 Método

RGB-D SLAM es uno de los primeros algoritmos que surgieron para realizar SLAM empleando las imágenes en color y de profundidad de las cámaras RGB-D; comparado con otros métodos es uno de los más fiables y robustos.

En general, el algoritmo desarrollado en este capítulo consta de cuatro pasos:

- Extracción de las características (features) y landmarks de la información obtenida por la cámara RGB-D.
- Comparación de las características extraídas con las obtenidas en iteraciones anteriores.
- Una vez obtenido el conjunto de correspondencias, se utiliza un algoritmo RANSAC para estimar la transformación relativa entre los pares de imágenes obtenidas.
- Optimización del grafo global.

El algoritmo RANSAC, se trata de un método iterativo que puede ser utilizado para extraer líneas rectas de una lectura de muestras de un sensor de distancias. Así se consigue distinguir líneas rectas en un entorno en tres dimensiones, como las formadas por las paredes, techos, suelos y muebles que pueden encontrarse en un ambiente de interior.
También se emplea el algoritmo ICP en el método RGB-D SLAM. Es utilizado para minimizar las diferencias entre dos nubes de puntos. Se trata de un proceso iterativo basado en la asociación de puntos siguiendo el criterio del vecino más cercano.

7.1.1 Algoritmo RGB-D SLAM

Este algoritmo divide la estimación de la trayectoria en "front-end" y "back-end" tal y como se aprecia en la Figura 25. El "front-end" se encarga de extraer información de las imágenes y establecer las correspondencias entre features, mientras que el "back-end" es el encargado de formar el grafo de posición y optimizarlo.

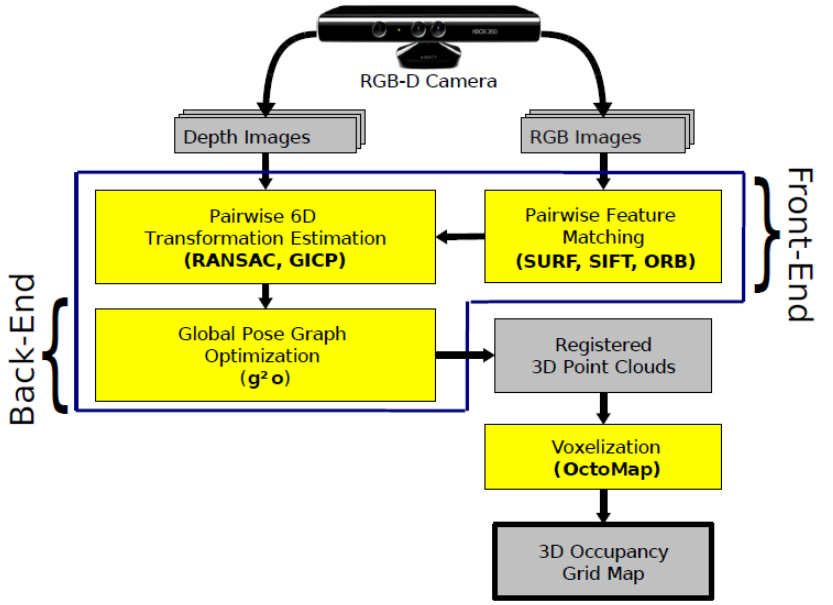

Figura 25: Esquema del algoritmo RGB-D SLAM

7.1.1.1 SLAM "Front-end"

Esta primera etapa del algoritmo es la encargada de establecer las relaciones espaciales de los datos extraídos del sensor.
Tal y como se muestra en la Figura 25, lo primero que realiza el algoritmo es el "matching" de las features extraídas de las imágenes. La detección se realiza mediante las librerías de OpenCV y los descriptores utilizados son SURFT, SIFT y ORB.
Una vez detectados los keypoints, se calcula su posición en 3D mediante las nubes de puntos extraídas de la cámara RGB-D (depth data). Para ello, se emplea el algoritmo RANSAC y el ICP debido a que la información obtenida de las features mediante visión no es lo suficiente fiable. Incluso, en algunas ocasiones, la información extraída de las imágenes no se corresponde totalmente con los datos de profundidad obtenidos, debido a una falta de sincronización entre la cámara de color y la de profundidad o a algún salto en la interpolación.

7.1.1.2 SLAM "Back-end"

Para crear una trayectoria global consistente se optimiza el grafo de posición mediante el framework g2o. Se trata de un optimizador de grafos que trata de minimizar una función de error no lineal, que generalmente es de la siguiente forma:

$$\begin{aligned} \mathbf{F}(\mathbf{x}) &= \sum_{\langle i,j\rangle \in \mathcal{C}} \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})^\top \mathbf{\Omega}_{ij} \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij}) \\ \mathbf{x}^* &= \operatorname{argmin} \mathbf{F}(\mathbf{x}). \end{aligned}$$

Figura 26: Función de error g2o

Donde x es el vector de posición, z es la restricción a cumplir entre las dos posiciones que se comparan y Ω es la matriz de restricciones.

7.1.1.3 Representación del mapa

Una vez se ha extraído toda la información de la cámara y se ha tratado, se puede usar para generar una

representación del entorno. Para ello, se utiliza la librería OctoMap, que implementa la creación de mapas 3D mediante un planteamiento en rejilla. Dicha librería realiza una estimación probabilística de la ocupación para garantizar la capacidad de actualización y hacer frente a ruidos del sensor y el entorno.

7.2 Pruebas realizadas

Este algoritmo se ha probado con una Kinect en tiempo real. Al igual que los otros métodos, se ejecuta el algoritmo en ROS. La interfaz que aparece al ejecutar el algoritmo es la que se muestra en la Figura 27.

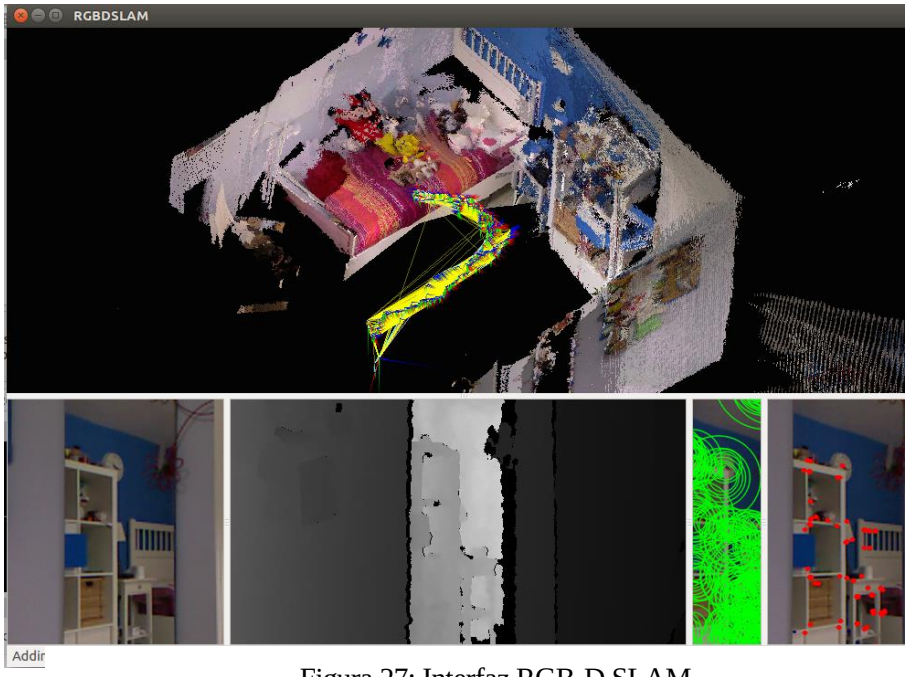

Figura 27: Interfaz RGB-D SLAM

En la parte superior se observa el mapa generado por el algoritmo, en él se observan también el recorrido realizado por la cámara y los diferentes keyframes. En la parte inferior tenemos varias ventanas; la primera de ellas comenzando por la izquierda muestra la imagen actual de la cámara rgb, justo en la ventana de al lado aparece la imagen de profundidad, los colores en escala de grises codifican la distancia detectada. Las siguientes ventanas muestran los puntos de interés extraídos, y la información del algoritmo RANSAC. El tamaño de los círculos verdes representa el grado de confianza en la estimación (menor cuanto mayor es el grado).

En general, los resultados obtenidos con este método han sido muy buenos. El algoritmo se ejecuta rápido y sin apreciarse retrasos. Ante movimientos bruscos de cámara responde de forma adecuada, ya que si pierde los puntos los recupera rápidamente.

8 COMPARATIVA ENTRE LOS DIFERENTES MÉTODOS ESTUDIADOS.

Este capítulo pretende realizar una comparativa entre los métodos estudiados a lo largo del trabajo. Uno de los aspectos más importantes a la hora de hacer SLAM es la fiabilidad que tienen los mapas generados ya que a partir de ellos se realizará el posicionamiento del robot. Por este motivo, se ha considerado de gran interés realizar una serie de pruebas con los algoritmos estudiados y comprobar la precisión que ofrece cada uno de ellos.

Para comprobar la fiabilidad y precisión del posicionamiento del robot empleando SLAM es necesario conocer con certeza la posición real del mismo. De esta forma se pueden comparar los resultados obtenidos y obtener el error que comente el algoritmo. Las medidas que consideramos reales o que más se acercan a la realidad y que se usan con el fin de comparar se conocen como variables "ground truth". En el presente trabajo, debido a no contar con un sistema de posicionamiento preciso con el que poder comparar las posiciones resultantes de realizar SLAM, se han empleado como ground truth las dimensiones de diferentes objetos que se encontraban en el entorno en el que se ejecuta el algoritmo.

Las pruebas se han realizado en dos entornos reales diferentes y todas en tiempo real. Los sensores empleados han sido una Kinect y una WebCam Logitech HD WEBCAM C525. La webcam se ha usado para las pruebas del algoritmo LSD-SLAM por ser un método monocular. No obstante, en el método ORB-SLAM pese a tratarse de un método monocular, la cámara usada ha sido la Kinect (sin utilizar las imágenes de profundidad) debido a que los resultados eran mejores. Los algoritmos RTAB-Map y RGB-D se han ejecutado utilizando la Kinect, en ambos casos empleando toda la información ofrecida por la misma.

En cada entorno se han realizado múltiples mapas con cada algoritmo estudiado en este trabajo. A partir de las diferentes medidas que se han ido consiguiendo de cada nuevo mapa generado se han calculado los errores que cometen cada uno de ellos. Todos han trabajado con los entornos en las mismas condiciones para poder realizar una comparación real entre ellos y poder comprobar cuál es el que ofrece un posicionamiento más fiable. Debido a tratarse de algoritmos diferentes, se han ajustado primero los diferentes parámetros para cada método y con aquellos con los que el método presenta mejor comportamiento en lo referente a las nubes de puntos generadas, la velocidad de ejecución del algoritmo o la respuesta ante cambios bruscos de cámara entre otras. Una vez obtenidos dichos parámetros se han realizado los experimentos que se muestran en este capítulo.

8.1 Escenario 1

El primer escenario se presenta en la Figura 28 , se trata de una habitación pequeña con pocos muebles y con iluminación artificial.

Figura 28: Escenario 1

Los ground truth empleados en este escenario han sido la altura y el ancho de la caja azul que aparece en la estantería y el largo de la cama. Durante las pruebas la estantería se ha grabado en su mayoría de frente, sin embargo, la dimensión de la cama con la que se pretende comparar las medidas obtenidas de los mapas generados no se ha enfocado directamente en ningún momento con la intención de comparar posteriormente los resultados obtenidos entre los algoritmos monoculares y los estéreos a la hora de calcular profundidades en un entorno.

A continuación, se muestran algunos de los mapas generados por los diferentes métodos.

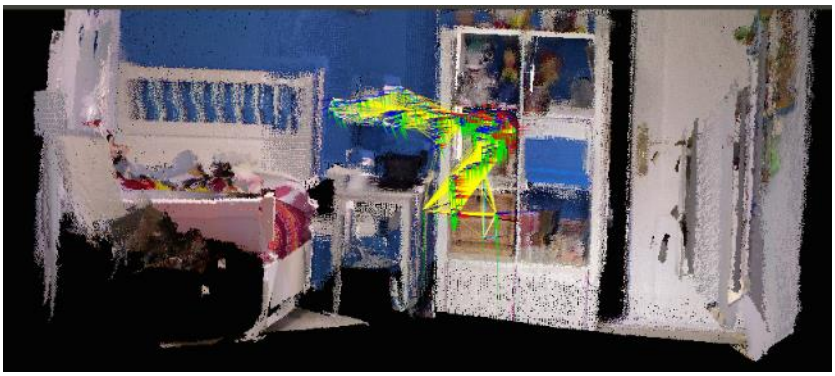

Figura 29: Mapa generado por RGB-D SLAM

Figura 30: Nube de puntos generada por RTAB-Map

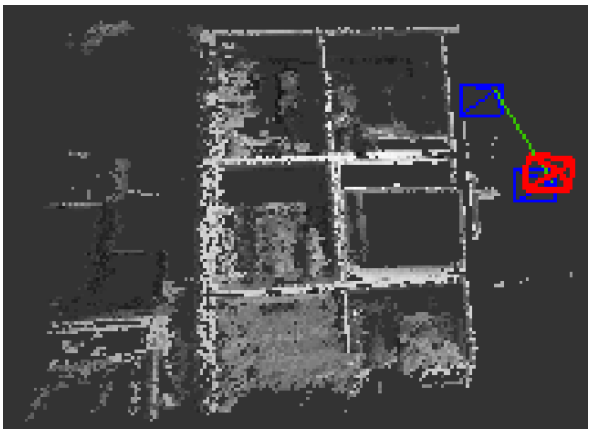

Figura 31: Parte del mapa de instensidades del

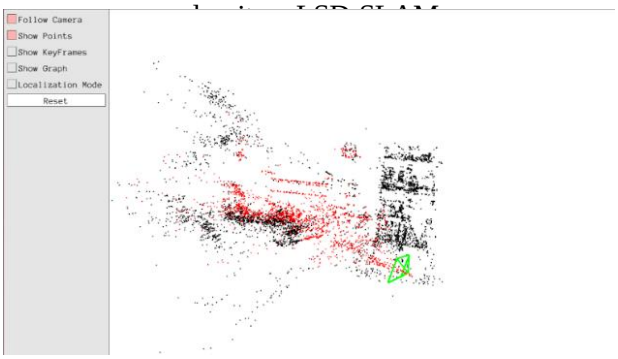

Figura 32: Nube de puntos generada con el ORB-SLAM

En la Tabla 1 se encuentran recogidos los resultados de las pruebas realizadas en este escenario. Las medidas están expresadas en metros.

	Ancho de la caja (0.32 m)		Altura de la caja (0.25 m)		Largo de la cama (2 m)	
	Media de las medidas	Máx. Error	Media de las medidas	Máx. Error	Media de las medidas	Máx. Error
RTAB-Map	0.311	0.01	0.26	0.01	2.1	0.1
ORB-SLAM	0.28	0.25	0.21	0.15	1.05	1
LSD-SLAM	0.29	0.2	0.23	0.2	0.96	1.2
RGB-D SLAM	0.3	0.02	0.25	0.03	1.77	0.2

Tabla 1:Resultados obtenidos para el escenario 1

8.2 Escenario 2

En este caso se trata de una habitación con muchos más objetos que la anterior, a diferencia del escenario anterior, la habitación tenía menos luz. La Figura 33 muestra el entorno donde se ha realizado el segundo conjunto de experimentos.

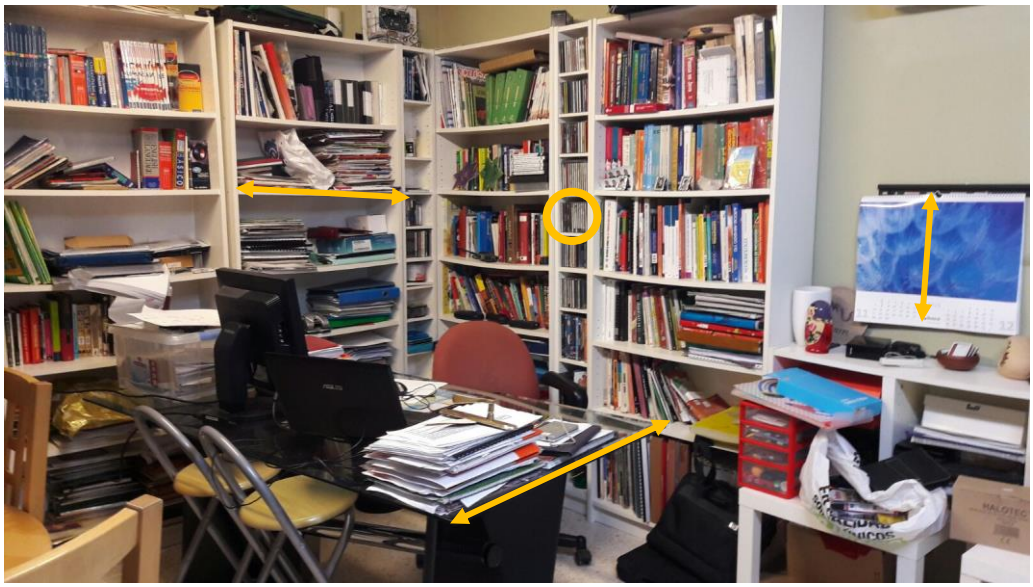

Figura 33: Escenario 2

En esta ocasión los ground truth seleccionados han sido el alto de una de las repisas, el ancho de otra de ellas, el ancho de la mesa y la altura del calendario de pared. Para facilitar la identificación de las medidas elegidas se han marcado de color amarillo en la Figura 33.

Las siguientes imágenes muestran algunos de los mapas generados por los diferentes métodos para este escenario.

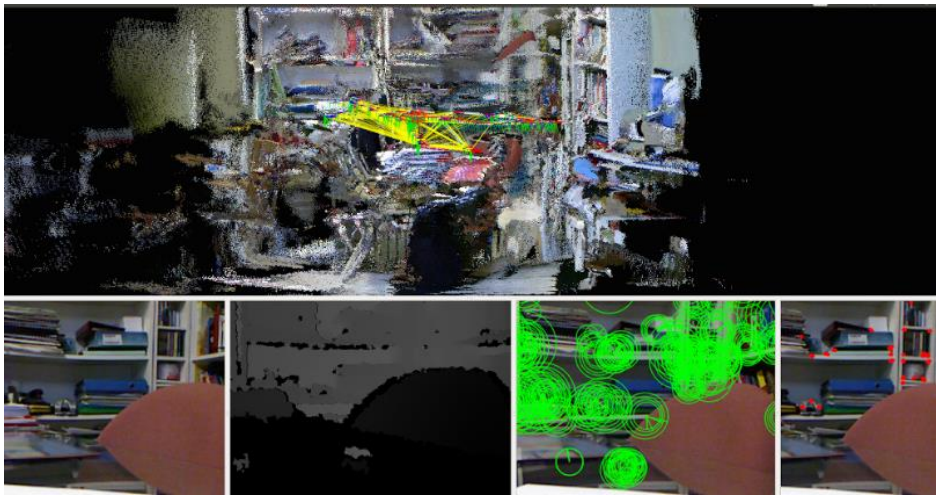

Figura 34: Método RGB-D SLAM

Figura 35: Nube de puntos RTAB-Map

Figura 36: Método LSD-SLAM

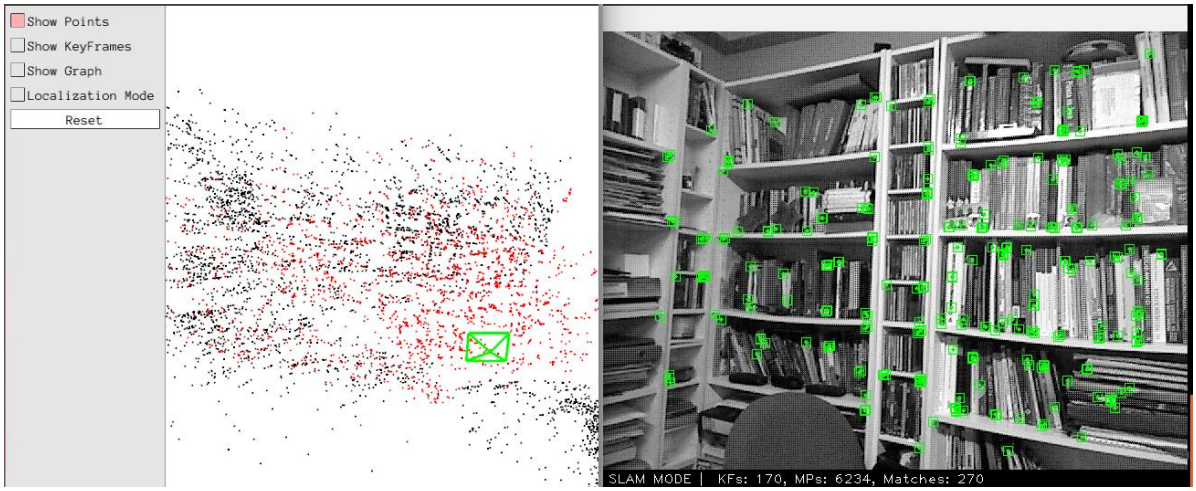

Figura 37: Método ORB-SLAM

La recoge los resultados obtenidos de las diferentes pruebas que se han realizado sobre esta habitación.

	Alto estantería (0.16 m)		Ancho estantería (0.75 m)		Altura calendario de pared (0.42 m)		Ancho mesa (0.85)	
	Media de las medidas	Máx. Error	Media de las medidas	Máx. Error	Media de las medidas	Máx. Error	Media de las medidas	Máx. Error
RTAB-Map	0.15	0.01	0.75	0.05	0.39	0.03	0.95	0.3
ORB-SLAM	0.1	0.06	0.45	0.3	0.3	0.15	0.46	0.4
LSD-SLAM	0.12	0.09	0.4	0.35	0.29	0.2	0.4	0.45
RGB-D SLAM	0.16	0.02	0.77	0.05	0.44	0.03	0.5	0.36

Tabla 2: Resultados obtenidos para el escenario 2

8.3 Conclusiones

En lo referente al comportamiento de los algoritmos durante la realización de los experimentos, cabe destacar que los métodos RTAB-Map y RGB-D SLAM se han ejecutado de forma rápida, sin apreciarse ningún retraso. En ambos métodos, cuando se ha realizado un movimiento brusco con la cámara para comprobar si se desorientaba y volvía a localizar su posición, en poco más de un segundo, los algoritmos eran capaces de recalcular su posición de forma satisfactoria permitiendo así continuar con la creación del mapa. En el método ORB-SLAM la respuesta era un poco más lenta, y generalmente consigue restablecerse tras un movimiento brusco de la cámara. No obstante, durante la realización de las pruebas con el algoritmo LSD-SLAM se ha percibido que el tiempo computacional es muy elevado, la imagen a menudo se ha quedado congelada y avanzando a saltos. La localización se ha perdido en numerosas ocasiones sin existir apenas movimientos bruscos, y en la mayoría de los casos no ha conseguido volver a estimar su posición.

Tras observar los resultados obtenidos en las diferentes pruebas realizadas, se concluye que en general los resultados para SLAM monocular son inferiores a los obtenidos con algoritmos en estéreo. En el escenario 1 se puede observar que las medidas obtenidas para la caja son bastante similares en todos los métodos, sin embargo, a la hora de medir el largo de la cama las medidas obtenidas por los algoritmos monoculares difieren demasiado de la realidad, llegando a tener 1 metro de error en algunos casos.

En el escenario 2 se aprecian resultados similares a los del escenario 1, en esta ocasión los algoritmos monoculares vuelven a presentar un mayor error que los métodos RTAB-Map y RGB-D SLAM. Sin embargo, cabe destacar que en una de las medidas realizadas los cuatro métodos presentan un error considerable, en el caso de los algoritmos monoculares no es demasiado llamativo que se produzcan estos errores, pero para los otros dos llama la atención. El motivo de estos errores es posiblemente que la mesa es de cristal, lo que puede confundir la información obtenida de las imágenes de profundidad del sensor IR de la Kinect.

9 CONCLUSIONES Y TRABAJO FUTURO.

En este proyecto se ha realizadoun análisis sobre algoritmos para localización y mapeado simultáneo de objetos. Para ello, en primer lugar, se estudiado la estructura de la técnica SLAM en general y posteriormente se ha estudiado cuatrométodos que empleaban variantes de dicho algoritmo.

Se ha realizado un estudio teórico de las técnicas RTAB-Map, ORB-SLAM, LSD-SLAM y RGB-D SLAM y posteriormente se han realizado una serie de pruebas que pretenden estudiar el funcionamiento de las mismas. Tal y como se comentó en el primer capítulo del presente trabajo, la estimación de una posición adecuada en entornos desconocidos es un aspecto fundamental, por ello al final de este trabajo se han realizadodiferentes pruebas que pretenden comprobar la fiabilidad de las técnicas estudiadas a la hora de generar los mapas y ofrecer un posicionamiento del robot. De dichos experimentos se ha concluido, que se obtienen mejores resultados de los algoritmos que emplean dispositivos tipo Kinect debido a que los cálculos de distancias son más precisos y en consecuencia el mapa creado y la estimación de la posición son de mayor calidad, esto se debe a que se tiene más información del entorno que en los métodos monoculares ya que la Kinect ofrece imágenes de profundidad que mejoran notablemente los resultados.

Por otra parte, se han comparado en tiempo real las distintas estrategias que han sido desarrollada por cada una de las técnicas estudiadas de forma que hemos podido comprobar la idea de emplear toda la información de la imagen en lugar de features para calcular el mapa como se realiza en el método LSD-SLAM, puede ser una buena idea a priori, ya que al obtener todos los píxeles y no solo algunas características podrían resultar de utilidad para ciertas aplicaciones (la realidad aumentada, por ejemplo). Sin embargo, se ha comprobado que el tiempo de computación es elevado y esto supone una limitación para muchas aplicaciones.

Cabe destacar que, tras realizar este estudio, se concluye que el algoritmo de SLAM es una técnica muy acertada para abordar el problema del posicionamiento en interiores. Esta técnica aplicada en robótica móvil permitiría ampliar las tareas capaces de realizar por este tipo de robots. En robótica móvil aérea, algoritmos de SLAM ampliarían las actividades que pueden realizar los UAVs, ya que evitaría que la aeronave deba estar siempre en el campo de visión del piloto pues es capaz de posicionarse ella misma en el entorno en que navega, dotando así de mayor autonomía al UAV.

9.1 Trabajo futuro

Los experimentos realizados se han hecho con la cámara en la mano y en tiempo real. Ya que se pretende que estas técnicas sean aplicadas a robots, sería interesante realizar los experimentos con empleando algún tipo de robot móvil. Así, se podría estudiar aún mejor las ventajas y los inconvenientes de cada uno de los métodos en aplicaciones reales. Además, se podrían realizar experimentos en algún entorno que cuente con un sistema preciso de posicionamiento que podamos usar como ground truth (como por ejemplo un sistema VICON) y comprobar directamente la localización del robot obtenida con los métodos estudiados, en lugar de comprobar medidas de objetos que se encuentren en el entorno de navegación.

Por otra parte, una vez analizadas las características de cada uno de los algoritmos, se podría realizar una nueva variante del algoritmo de SLAM teniendo en cuenta todas las conclusiones que se han extraído. Además, ya que los métodos estudiados son proyectos *open source*, podrían intentar realizarse modificaciones sobre ellos con la intención de mejorar las limitaciones que hemos encontrado.

Como última propuesta de trabajo futuro, se plantea la idea de ir un poco más allá en el planteamiento del algoritmo de SLAM. Esta propuesta está pensada para misiones en las que participe más de una una plataforma móvil. Sería interesante realizar SLAM, no solo con la información obtenida por un robot, si no con la información que van generando todas las plataformas móviles, ya sean terrestres, acuáticas o aéreas, que forman parte de la misión. De esta forma, se crearán los mapas de forma más rápida y los tiempos de computación disminuirán puesto que no es una sola plataformala que debe procesar toda la información recopilada. Los datos generados de forma individual se transmitirían al resto de plataformas para que puedan incorporarla a su mapa. Este planteamiento tendría utilidad no sólo en interiores, si no que podría emplearse en misiones en entornos exteriores. En el supuesto de que uno de los sistemas pierda la señal de GPS podría ser capaz de estimar su posición y continuar con la misión a partir de la información recopilada por el resto de aeronaves.

REFERENCIAS

1. *Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms.* **Hugh Durrant-Whyte, Fellow, IEEE, and Tim Bailey.** 2006.

2. **Søren Riisgaard, Morten Rufus Blas.***SLAM for Dummies.* 2004.

3. *Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation.* **Mathieu Labbé, François Michaud.** 2013.

4. *ORB-SLAM: a Versatile and Accurate Monocular SLAM System.* **Raul Mur-Artal, J. M. M. Montiel, Juan D. Tardos.**2015.

5. *LSD-SLAM: Large-Scale Direct Monocular SLAM.* **Jakob Engel, Thomas Sch¨ops , Daniel Cremers.** 2014.

6. *An Evaluation of the RGB-D SLAM System.* **F. Endres, J. Hess, J. Sturm, D. Cremers, W. Burgard, Nikolas Engelhard.** 2014.

7. **https://github.com/tum-vision/lsd_slam. [Online]**

8. https://github.com/raulmur/ORB_SLAM2. [Online]

9. http://wiki.ros.org/rtabmap_ros. [Online]

10. ***An iterative image registration technique with an application to stereo vision.*** **Lucas, B.D. and Kanade, T. 1983.**