Universidad Tecnológica Nacional
ABRIL 2020 / Año 18- Nº 37

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.37.123-133.2020 - ISSN 1666-6933
Creative Commons Reconocimiento-NoComercial 4.0 Internacional.

Actas de las IX Jomadas Argentinas de Robótica 15-17 de noviembre, Córdoba, Argentina

Diseño Distribuido de un Sistema de Mapeo y Localización basado en visión para un Robot Móvil

Distributed Design of a Vision-based Mapping and Localization System for a Mobile Robot

Presentación: 25/09/2017

Aprobación: 02/12/2017

Mauro De Croce
Centro Internacional Franco Argentino de Ciencias de la Información y de Sistemas (CONICET-UNR) - Argentina
maurodecroce@gmail.com

Taihú Pire
Centro Internacional Franco Argentino de Ciencias de la Información y de Sistemas (CONICET-UNR) - Argentina
pire@cifasis-conicet.gov.ar

Federico Bergero
Centro Internacional Franco Argentino de Ciencias de la Información y de Sistemas (CONICET-UNR) - Argentina
bergero@cifasis-conicet.gov.ar

Resumen

En este trabajo se presenta una arquitectura distribuida para el sistema de SLAM estéreo S-PTAM. Esta arquitectura está desarrollada sobre el framework ROS, separando los hilos de localización y mapeo en dos nodos independientes de ROS. El sistema de SLAM distribuido resulta ideal para robots móviles con bajo poder de cómputo ya que permite ejecutar el módulo de localización *onboard* y el módulo de mapeo -que tiene mayor costo computacional- en una estación base, aprovechando al máximo el procesamiento disponible *onboard*. Para la validación del sistema se realizaron experimentos en un *dataset* público. Los resultados obtenidos muestran la factibilidad de una arquitectura distribuida y su implementación eficiente.

Palabras claves: SLAM, Sistema Distribuido, Visión Estéreo, Tiempo-Real.

Abstract

In this paper we present a distributed architecture for the S-PTAM stereo SLAM system. This architecture is developed on the ROS framework, separating the localization and mapping threads in two independent ROS nodes. The distributed SLAM system is ideal for mobile robots with low computing power since it allows the onboard location module and the mapping module -which has a higher computing cost- to be executed in a base station, taking maximum advantage of the processing available onboard. For the validation of the system,

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.37.123-133.2020 - ISSN 1666-6933

experiments were carried out in a public dataset. The results obtained show the feasibility of a distributed architecture and its efficient implementation.

Keywords: SLAM, Distributed System, Stereo Vision, Real-Time.

I. Introducción

Para que un robot móvil pueda navegar por el entorno es fundamental que conozca su posición y orientación (pose) a cada momento y que cuente con una representación del entorno (mapa) que se encuentra transitando. En robótica estos problemas suelen abordarse de manera simultánea, y el problema se conoce con el nombre de SLAM (*Simultaneous Localization and Mapping*) (Durrant-Whyte and Bailey, 2006; Bailey and Durrant-Whyte, 2006; Cadena et al., 2016). En este tipo de problema se plantea la necesidad de construir un mapa del entorno (*mapping*), a la vez que se estima la trayectoria del robot al desplazarse dentro del mismo (*localization*), tomando las señales de los sensores montados en él.

Existen distintos tipos de sensores que pueden utilizarse para resolver el problema de SLAM: cámaras monoculares y estéreo, sonares, láser, GPS (*Global Positioning System*) y sensores RGB-D entre otros. Sin embargo, cada sensor tiene sus limitaciones, por ejemplo, un sonar permite detectar si hay objetos a cierta distancia, pero no es posible determinar de qué clase de objeto se trata. En el caso de un GPS sólo funciona en entornos exteriores (*outdoor*), incluso hay lugares donde la señal no es suficientemente buena. Los sensores láser suelen ser costosos, poco portables y no cuentan con un amplio rango de alcance. Por último, los sensores RGB-D tipo *Kinect* utilizan un sensor infrarrojo para estimar la profundidad de la escena observada y por ende se ven afectados por la luz solar, lo que impide un buen funcionamiento en ambientes exteriores.

En contrapartida, las cámaras superan muchas de estas limitaciones al tratarse sensores de gran portabilidad y bajo costo. Al mismo tiempo proveen abundante información de la escena que se está observando y además pueden ser utilizadas en ambientes tanto interiores como exteriores. Todo esto hace que las cámaras sean uno de los sensores más utilizados para resolver el problema de SLAM en la actualidad.

Dentro de las soluciones basadas en cámaras podemos encontrar aquellas que utilizan dispositivos monoculares o estéreo. Si bien tanto los sistemas monoculares como los estéreo permiten realizar una reconstrucción 3D de la escena observada, son estos últimos los que cuentan con la ventaja de poder determinar la escala de la misma. Actualmente existen numerosos enfoques para resolver el problema de SLAM mediante el uso de visión (SLAM Visual) como por ejemplo: PTAM (Klein and Murray, 2007), ORB-SLAM2 (Mur-Artal and Tardós, 2016), LSD-SLAM (Engel et al., 2014), ScaViSLAM (Strasdat et al., 2011), SVO (Forster et al., 2017), S-LSD-SLAM (Engel et al., 2015) y S-PTAM (Pire et al., 2015).

Los sistemas SLAM tienen un alto costo computacional debido a la necesidad de procesar gran cantidad de información en tiempo real. Actualmente, los robots móviles cuentan con una capacidad de procesamiento *onboard* limitada dada por la naturaleza de los dispositivos que los conforman. Cabe aclarar que dependiendo del entorno y la tarea que deba realizar el robot, el hardware de procesamiento deberá satisfacer restricciones de peso, tamaño y consumo eléctrico. Un exceso en cualquiera de estas magnitudes podría comprometer la capacidad de navegación o autonomía del robot. Por ello, resulta conveniente distribuir la carga de procesamiento apoyándose en distintos dispositivos externos.

En este trabajo se propone una versión distribuida del sistema de SLAM Visual S-PTAM de tal manera que se permita ejecutar el módulo de localización (*tracking*) en la unidad de procesamiento *onboard* del robot, y separadamente el módulo de mantenimiento del mapa (*mapping*) en una estación base. De esta manera, el módulo de localización dispone de manera exclusiva la capacidad de procesamiento del robot, permitiendo procesar la mayor cantidad de imágenes provistas por la cámara, y por lo tanto se obtiene un sistema más preciso y robusto a movimientos bruscos del robot. Asimismo, el módulo de mapeo que demanda un gran costo computacional, al correr en una estación remota, puede ejecutarse en una computadora de mayor capacidad de cómputo que las que usualmente se encuentran en un robot móvil.

Para tal fin, realizamos un rediseño de S-PTAM de tal manera que pueda distribuir las tareas de localización y mapeo en dos computadoras (*onboard* y estación base). Para esto, se planteó un diseño modular basado en comunicación por eventos que utiliza memoria distribuida en vez de memoria compartida.

Al transicionar a un esquema de memoria compartida surge la necesidad de disponer una copia del mapa en cada módulo (*tracker* y *mapper*), de manera que cada uno de ellos pueda trabajar de forma independiente, ante

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.37.123-133.2020 - ISSN 1666-6933

cada cambio que se produzca, ya sea en uno u otro. Por ejemplo el mapa de *tracker* es modificado a partir de la creación de nuevos elementos, mientras que el de *mapper* es modificado cuando alguno de ellos se ajusta o se refina. Para mantener la consistencia de ambos mapas se utilizaron mecanismos de comunicación basados en el paradigma que ofrece ROS en materia de mensajes. Para ello se crearon estructuras de mensajes de manera de incluir la mayor cantidad de información posible con el objetivo de minimizar la cantidad de mensajes, y por ende reducir el tráfico y el costo computacional que implica esta sincronización.

II. Trabajo Relacionado

Los problemas SLAM son un caso de estudio desde hace ya algunas décadas y muchos trabajos fueron realizados en pos de obtener soluciones basadas en sistemas distribuidos, algunos mediante la utilización de *frameworks* desarrollados para tal fin como es el caso de (Gamage and Tuceryan, 2016) y de (Riazuelo et al., 2014). El primero está orientado a disminuir la complejidad de las comunicaciones dentro de la red para facilitar las implementaciones de soluciones SLAM distribuidas utilizando ROS como plataforma subyacente. Para ello, aprovecha la abstracción de mensajes provista por ROS, como así también su sistema de estadísticas, lo que le permite generar un reporte del consumo de recursos de red utilizado por cada nodo de ejecución. El segundo trabajo propone que los datos, como el mapa y otras variables, sean almacenados en servidores externos ubicados en la nube. Esto libera de carga computacional al robot, que sólo debe encargarse de realizar el trackeo de las imágenes, aunque requiere una conexión a *Internet* mediante *WiFi* para los dispositivos.

Actualmente, la creciente utilización de vehículos aéreos en diversas aplicaciones ha desencadenado numerosos estudios sobre esta plataforma. Al igual que todos los robots móviles, los vehículos aéreos como los *drones* también requieren de soluciones SLAM para su funcionamiento autónomo, tal esel caso de (Williams et al., 2015), (Mahdoui et al., 2016) que proponen el uso de *drones* para obtener, mediante la aplicación del paradigma de PTAM (Klein and Murray, 2007), un mapa de forma colaborativa. Por otro lado, teniendo en cuenta la limitación de peso que los *drones* pueden transportar en comparación con los robots terrestres, el procesamiento distribuido toma un valor fundamental ya que permite balancear tanto carga como procesamiento en dispositivos externos y de esta manera liberar al robot de muchas responsabilidades. Encontramos un ejemplo de ello en Forster et al. (2013), donde se incorpora el uso de una estación central que sincroniza múltiples vehículos, genera un mapa por cada uno de ellos y luego realiza la unión de los fragmentos solapados obteniendo así un mapa general.

También podemos mencionar el desarrollo planteado en (Williams et al., 2014), que describe un framework en el que se incorporan técnicas de SLAM monocular basadas en PTAM, que le permiten a múltiples vehículos aéreos la posibilidad de evitar obstáculos durante el proceso de exploración del entorno. Para este desarrollo se utilizó el framework ROS, donde cada componente como los *drivers* de los vehículos, funcionalidades de trackeo y de mantenimiento del mapa, han sido implementados como nodos independientes aprovechando la arquitectura distribuida de ROS, pudiendo así ejecutarlos en distintos procesos con el fin de agregar escalabilidad al *framework*.

Por otro lado, en (Bischof, 2012) se presenta un novedoso sistema distribuido de reconstrucción volumétrica de objetos utilizando como dispositivo de entrada un vehículo aéreo que envía datos de las imágenes capturadas por sus cámaras a un servidor de gran capacidad de procesamiento que realizará las tareas más complejas. Además, permite al usuario hacer uso de una *tablet* para visualizar los resultados en tiempo real, brindando también la posibilidad de corregir resultados mientras se desarrolla el proceso.

Uno de los métodos de SLAM que ha tenido mayor impacto en la robótica móvil fue PTAM (Klein and Murray, 2007). Este método plantea un nuevo paradigma que se aplica a problemas de SLAM Visual en el que se propone separar las tareas de localización y mapeo en hilos de ejecución independientes optimizando el rendimiento en procesadores de múltiples núcleos. En particular, esta propuesta innovadora fue utilizada en S-PTAM (Pire et al., 2015) pero abordando el problema de SLAM para en entorno de grandes dimensiones. Cabe señalar que anteriormente los paradigmas utilizados para encontrar soluciones a los problemas de SLAM eran basados en filtros como por ejemplo el *Extended Kalman Filter* (Jazwinski, 1970) y *Particle Filter* (Metropolis and Ulam, 1949).

Muchos de los desarrollos que forman parte del estado del arte en el área de sistemas SLAM Visuales distribuidos abordan el problema de la colaboración entre dispositivos para la conformación de un mapa más completo del entorno, la mayoría de ellos trabaja sobre vehículos aéreos o dispositivos móviles, como *tablets* o

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.37.123-133.2020 - ISSN 1666-6933

celulares (Williams et al., 2015; Mahdouiet al., 2016; Forster et al., 2013; Williams et al., 2014; Bischof, 2012). Este trabajo, a diferencia de abordar el problema de construir un mapa más detallado del entorno, se enfoca en obtener un sistema de SLAM distribuido capaz de localizar un robot móvil en tiempo real. El sistema de SLAM propuesto se apoya tanto en el hardware de procesamiento *onboard* del robot como en otros dispositivos externos, considerando el poder de cómputo de cada uno para poder asignarle una tarea acorde a su capacidad de procesamiento.

III.S-PTAM original

III-A. S-PTAM Original

S-PTAM (*Stereo Parallel Tracking and Mapping*) (Pire et al., 2015) es un método que aborda el problema de SLAM basándose en visión estéreo. Fue desarrollado con el objetivo de ser ejecutado en tiempo real en ambientes de grandes dimensiones obteniendo una estimación precisa de la pose del robot mientras realiza una construcción del mapa del entorno que lo rodea. Siguiendo el enfoque de PTAM (Klein and Murray, 2007) (*Parallel Tracking and Mapping*) y con el fin de aprovechar el poder de los procesadores multi-núcleos, en S-PTAM se dividen las tareas de localización y mapeo en hilos de ejecución independientes, logrando de esta manera un muy buen desempeño en comparación con otros métodos de SLAM actuales.

Al comenzar su ejecución, el método asume que las cámaras están situadas en el origen de coordenadas del mundo y sin un mapa de su entorno. Luego, al recibir el primer *Stereo Frame* capturado por las cámaras, el mapa es inicializado mediante la triangulación de los puntos destacados en las imágenes de cada uno de los *frames* (*frame* izquierdo y *frame* derecho) llamados *Map Points*. A partir de entonces, los *Stereo Frames* recibidos serán localizados al estimar la pose de la cámara en base al mapa actual. Esta tarea se logra proyectando los puntos 3D sobre el plano de cada imagen para luego buscar las correspondencias con los *features* extraídos en la misma comparando cada uno de sus descriptores. Dichas correspondencias son llamadas correspondencias 2D-3D ya que representan la observación de un punto en el espacio por parte de las cámaras. Además, un *Stereo Frame* puede ser seleccionado para ser agregado al mapa y en consecuencia recibirá el nombre de *Key Frame*. Esta selección se realiza para no generar un mapa excesivamente grande, que desaproveche espacio en memoria e implique un tiempo excesivo a la hora de ser procesado. Para ello se utiliza una heurística en función de cuanta información nueva del entorno aporta el *Stereo Frame*. En particular, el *Stereo Frame* que inicia el mapa es considerado como un *Key Frame*.
Entonces, a medida que el robot avance en su recorrido obtendrá nuevas imágenes estéreo que serán procesadas con el método descripto y de esta manera se generará un mapa incremental que aumentará de tamaño en cada iteración del proceso.

De forma concurrente a *Tracker* se ejecuta el hilo que se encarga de realizar el refinamiento del mapa, llamado *Mapper*, que realiza tareas como el ajuste de posiciones dela cámara y de *Map Points*. Este hilo también busca nuevas asociaciones entre *Key Frames* y *Map Points* para mejorar las restricciones entre los elementos del mapa. Finalmente realiza la eliminación tanto de *Map Points* como de *Measurements* considerados erróneos, utilizando para ello técnicas de *Bundle Adjustment* (Triggs et al., 2000).

III-B. Arquitectura

La Figura 1 muestra un diagrama de la estructura generalde S-PTAM, la cual consta de dos funcionalidades principales: *Tracking* que recibe la información de los puntos extraídos de las imágenes estéreo y a partir de ellos calcula la pose del robot y crea los puntos que luego formarán parte del mapay *Local Mapping* (o *Mapper*) que realiza las optimizaciones sobre el mapa (*refinement*), establece nuevas conexiones entre nuevos puntos del mapa y *Key Frames* que fueron previamente agregados y además se encarga de eliminar aquellos puntos que son considerados como erróneos. Tanto *Tracker* como *Mapper* utilizan una estructura en memoria compartida que representa el mapa que se construye durante el proceso y que es accedido y modificado por estos módulos.

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.37.123-133.2020 - ISSN 1666-6933

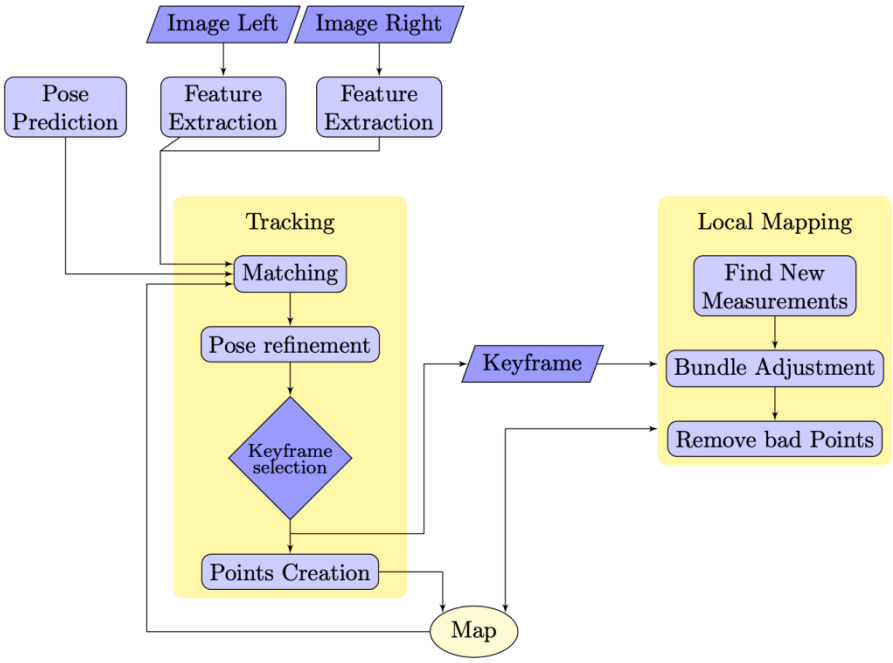

Figura 1: Diagrama de estructura general de S-PTAM (Pire et al., 2015).

IV. S-PTAM distribuido

IV-A. Arquitectura

Previo a este trabajo, el diseño de S-PTAM hacía uso de memoria compartida en el que existe un único mapa sobre el cual trabaja todo el sistema. En base a este planteo identificamos las funcionalidades predominantes y rediseñamos la arquitectura para que sean ejecutadas en módulos independientes. Cada uno de estos módulos será implementado como un nodo ROS individual, que podrá interactuar con otros mediante el sistema de mensaje que proporciona este *framework*. La comparación entre dichas arquitecturas puede apreciarse en la Figura 2.

(a) Arquitectura original. (b) Arquitectura propuesta.

Figura 2: Comparación entre la arquitectura de S-PTAM original y la arquitectura distribuida aquí propuesta. En la arquitectura distribuida tanto el *Tracker* como el *Mapper* tienen una instancia del Mapa, y la comunicación entre ambos nodos se realiza a través de mensajes de ROS.

Al proponer un funcionamiento distribuido será necesario contar con una copia del mapa (lo más actualizada posible) en cada uno de los nodos y será fundamental que ambos mapas sean consistentes entre sí. Para lograrlo, será necesario que cada nodo informe sobre los cambios que han sido realizados sobre su copia del mapa para que el otro nodo aplique esos cambios en el propio. En el diseño actual de S-PTAM los elementos del mapa tales como *Key Frames*, *Map Points* y *Measurements* pueden ser accedidos mediante el uso de punteros debido a que se encuentran en espacios de memoria compartida. En cambio, en el diseño propuesto en este trabajo fue necesario desarrollar un mecanismo de identificación de objetos que se encuentran en distintos espacios de memoria.

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.37.123-133.2020 - ISSN 1666-6933

Es así que se decide incorporar identificadores enteros en los objetos que forman parte del mapa y que son creados y modificados de forma dinámica durante el proceso. Estos identificadores se corresponderán, en cada nodo, al mismo objeto del mapa (cada uno con direcciones distinta de memoria).

Para abstraer este comportamiento de correspondencia se recurre a la creación de la clase *Translations*, que internamente consta de un vector de punteros a objetos y métodos que permiten asignar sus valores, obtener el objeto apuntado a partir de un identificador dado y otras funciones de control. Dicho vector contendrá en la posición *i-ésima* el puntero correspondiente al elemento con *id* igual a *i*.

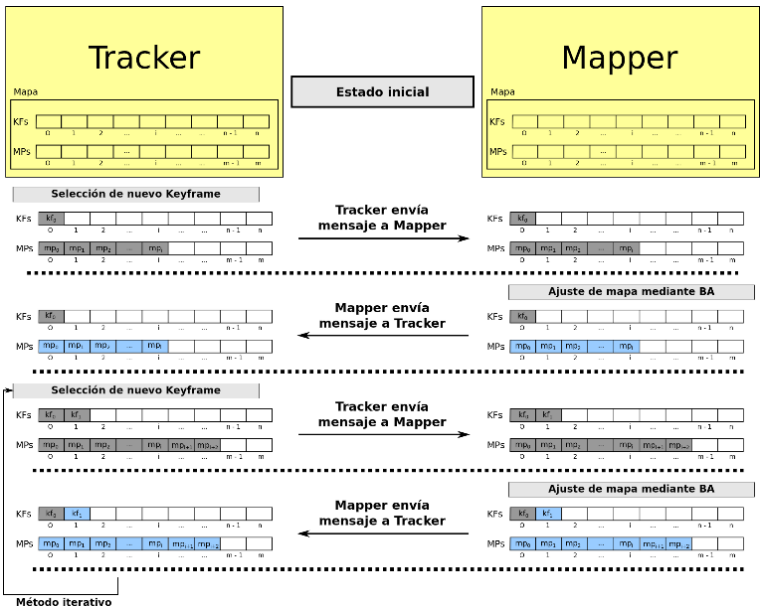

Figura 3: Modificaciones de los objetos *translations* durante el inicio de la ejecución.

Entonces, teniendo una instancia de esta clase para *Key Frames* y otra para *Map Points*, tanto *Tracker* como *Mapper* pueden hacer referencia a un objeto determinado sin importar que cada nodo lo tenga almacenado en un espacio de memoria distinta y de esta forma se provee un mecanismo para comunicarle al otro nodo sobre cambios en sus atributos de una manera clara y consistente. La Figura 3 muestra la evolución de los objetos *translations* de cada nodo durante el inicio dela ejecución.Por otro lado, se propone una estructura de mensajes para la comunicación entre los nodos definida en base a los estándares de ROS, los cuales incluyen tipos de dato como enteros, booleanos, flotante, caracteres, cadenas, etc. Además, las estructuras pueden conformarse con vectores anidados de estos tipos primitivos mencionados.

IV-B. Tracker

Con el objetivo de lograr un procesamiento distribuido, y considerando que S-PTAM dispone de un único mapa, se decide incorporar como primera medida una instancia del mapa al nodo *Tracker*. A su vez, también se incorporan las funciones y métodos que le permiten agregar, modificar y eliminar *Key Frames*, *Map Points* y *Measurements* al mapa.

Entonces, considerando que este nodo posee la funcionalidad de extracción de *features* podemos afirmar que el nodo *Tracker* está capacitado para recibir las imágenes estéreo provenientes de la cámara del robot móvil y procesarlas para luego crear y actualizar su instancia del mapa del entorno que lo rodea. De esta forma, se logra autonomía e independencia del nodo *Tracker* respecto del nodo *Mapper*, pudiendo ser ejecutado sin necesidad de que *Mapper* esté ejecutándose simultáneamente. Esta característica se constituye como uno de los principales aportes de este trabajo, ya que hasta el momento no era posible llevarla a cabo con el diseño monolítico actual de S-PTAM. Aunque como veremos luego, los resultados serán mejor al ejecutar ambos nodos concurrentemente,

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.37.123-133.2020 - ISSN 1666-6933

poder utilizar el nodo *Tracker* de forma autónoma nos permite, por ejemplo, ejecutar una versión simplificada de S-PTAM en dispositivos que no tengan gran capacidad de cálculo.

IV-C. Mapper

El otro módulo principal del sistema que fue implementado como un nodo ROS es el correspondiente a *Mapper*. Este nodo se acopla al nodo *Tracker* trabajando de forma complementaria. El nodo *Mapper* también fue dotado con una instancia del mapa el cual será ajustado continuamente mediante el método de optimización *Bundle Adjustment* y cada modificación será comunicada a *Tracker* para que actualice su mapa a fin de lograr consistencia entre ellos.

IV-D. Comunicación entre los nodos

La comunicación entre los nodos se lleva a cabo mediante la utilización del sistema de comunicación y gestión de mensajes provistos por ROS el cual facilita esta tarea a partir de canales de información entre nodos llamados *topics*. Para enviar mensajes a través de un *topic* un nodo ROS deberá publicar en él, mientras que para recibir un mensaje publicado en un *topic* deberá suscribirse al mismo. Cabe destacar que el nodo emisor no conoce quienes son los nodos receptores de su mensaje, es decir que estos son unidireccionales y asíncronos (difusión).

Entonces *Tracker* y *Mapper* se comunican utilizando la estructura de mensajes diseñada para tal fin en la que se incluye la información relacionada con los cambios en el mapa, es decir que cuando *Tracker* agrega elementos como *Key Frames*, *Map Points* y *Measurements* genera un mensaje, respetando la estructura propuesta en la que se incluye toda la información necesaria para que el nodo *Mapper* pueda agregar esos objetos en su propio mapa. Como contrapartida, cuando el nodo *Mapper* modifica su mapa al realizar el refinamiento del mismo, le comunica a *Tracker* dichos cambios utilizando el mencionado sistema de mensajes, incluyendo en ellos sólo información relevante para que *Tracker* pueda reflejar en su mapa dichas modificaciones y así evitar un mensaje que contenga información redundante e impacte negativamente en la *performance* del sistema. Por una cuestión de simplicidad en el diseño, se decidió utilizar la misma estructura de mensaje tanto para los que fueran enviados por *Tracker* como por *Mapper*.

Para facilitar el manejo de los mensajes se implementó la librería *messages handler*, la cual contiene funciones que convierten elementos de S-PTAM en mensajes ROS y viceversa. Es decir, que para crear un mensaje la función tomará como argumento objetos de C++ (que representan loselementos que S-PTAM utiliza internamente) y devolverá un mensaje correspondiente siguiendo la estructura definida para tal fin. En contrapartida, la función inversa tomará como argumento un mensaje y retornará un objeto utilizando para ello los constructores de la clase adecuada.

V. EXPERIMENTOS

En este trabajo utilizamos el dataset KITTI (Geiger et al., 2013) de dominio público, ampliamente utilizado en el campo de la robótica móvil para evaluar sistemas de SLAM basados en visión. La secuencia utilizada del dataset es la 00, la misma consiste en el conjunto de imágenes capturadas por una cámara montada sobre un vehículo que transita un ambiente urbano. La trayectoria realizada por el vehículo es de 4 km aproximadamente. Las imágenes cuentan con una resolución de 1344391 píxeles y son obtenidas a un *frame rate* de 10 Hz.

Para realizar los experimentos se utilizaron dos equipos: una PC con procesador Intel Core i3-4130 de 3.40 GHz con 4 núcleos, 8GB de RAM con Ubuntu 14.04 de 64 bits y una notebook con procesador Intel Core i3-3217U de 1.80 GHz con 4 núcleos, 8 GB de RAM con Ubuntu MATE 16.04 de 64 bits. Ambos equipos fueron conectados mediante una red cableada para simular la distribución de cómputo entre un robot móvil y una estación base. Esta configuración distribuida intenta replicar la existencia de un robot que se comunica durante todo el proceso con otra PC con mayor poder de cómputo, aunque en este caso ambos equipos cuentan con el mismo poder computacional.

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.37.123-133.2020 - ISSN 1666-6933

La Figura 4 muestra la comparación entre las trayectorias estimadas por el sistema propuesto (para distintas configuraciones) contra la trayectoria real realizada por el vehículo (*ground-truth*). El color celeste representa la trayectoria para la ejecución del nodo *Tracker* en la PC, mientras que en rojo se muestra la trayectoria para la ejecución de ambos nodos (*Tracker* y *Mapper*) en la misma PC. Finalmente, en color verde, la ejecución distribuida de los nodos en la que *Tracker* fue ejecutada en la PC mientras que *Mapper* se ejecutó en la notebook.

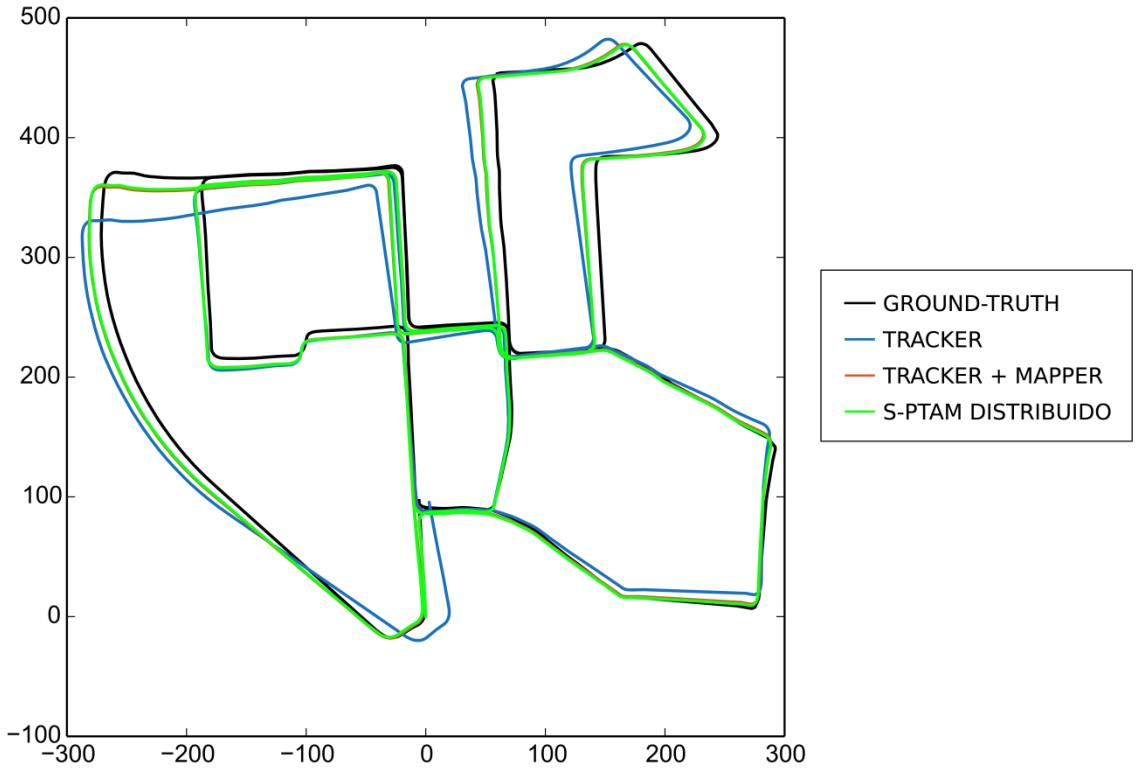

Figura 4: Comparación de las trayectorias estimadas por cada una de las configuraciones (Tracker, Tracker + Mapper y S-PTAM distribuido) para la secuencia 00 del dataset KITTI.

En la Tabla 1 se muestran los tiempos de procesamiento en promedio que demora el nodo *Tracker* en ejecutar sobre un par de imágenes estéreo para cada configuración del sistema (Tracker, Tracker + Mapper y S-PTAM distribuido). Se toma como punto de comparación dicho nodo dado quela tarea de localización debe correr en tiempo real en un sistema SLAM. La configuración con el *Tracker* individuales el que menos tiempo consume como es de esperarse. La configuración Tracker + Mapper lleva un mayor costo computacional demorando más tiempo que la configuración individual de *Tracker*. Esto se debe al alto costo computacional agregado por la ejecución del nodo de mapping en la misma PC. S-PTAM distribuido obtiene una *performance* similar a la de ejecutar el *Tracker* individualmente con el beneficio de que obtiene una mejor precisión de la localización del robot como se muestra en la Figura 4 y la Tabla 2. La Tabla 2 muestra la comparación de los errores donde t indica error de traslación, mientras que r hace mención al error de rotación, los subíndices *abs*, *rel* y *RMSE* hacen referencia a los errores absolutos, relativos y a la raíz cuadrada del error cuadrático medio (*Root Mean Square Error*), respectivamente. *Max* hace referencia a los valores máximos obtenidos para cada tipo de error. Por ejemplo, t_{abs} es el máximo valor de los errores absolutos de traslación obtenidos en toda la secuencia. Se puede observar que los errores relativos y RMSE son similares entre las tres configuraciones planteadas, los máximos errores absolutos de traslación y de rotación para la configuración del nodo *Tracker* ejecutado de manera individual son muy superiores a los valores obtenidos por las configuraciones a las que se integra el nodo de refinamiento de mapa, *Mapper*. En el caso de la configuración Tracker + Mapper se observa un marcado descenso en dichos errores, mientras que para el caso de la configuración de S-PTAM distribuido se obtienen errores aún menores, llevando a una mejora en la estimación de la trayectoria del robot al verse disminuida la tasa de error. Observar que la mejora del error entre las configuraciones Tracker+ Mapper y S-PTAM distribuido se debe a la

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.37.123-133.2020 - ISSN 1666-6933

menor carga que esta última configuración requiere sobre la computadora que se encuentra ejecutando el nodo *Tracker*.

Tiempos de ejecución	Tracker	Tracker + Mapper	S-PTAM distr.
Tracking (s)	0.043	0.051	0.044

Tabla 1: Comparación de tiempos para cada una de las configuraciones (Tracker, Tracker + Mapper y S-PTAM distribuido).

Errores	Tracker	Tracker + Mapper	S-PTAM distr.
Max t_{abs}(m)	47.95	17.32	16.63
Max t_{rel}(m)	0.30	0.30	0.30
t_{RMSE} (m)	0.030	0.029	0.029
Max r_{abs} (deg)	11.70	8.99	8.84
Max r_{rel}(deg)	2.20	2.20	2.17
r_{RMSE} (deg)	0.12	0.12	0.12

Tabla 2: Comparación de los errores absolutos y relativos máximos, y RMSE obtenidos para cada una de las configuraciones.

VI. CONCLUSIONES Y TRABAJO FUTURO

En este trabajo presentamos un sistema de SLAM distribuido. Inicialmente partimos del desarrollo de S-PTAM (*Stereo Parallel Tracking and Mapping*) (Pire et al., 2015), el cual es un método que aborda el problema de SLAM basándose en visión estéreo siguiendo el enfoque propuesto por PTAM.

Teniendo en cuenta que S-PTAM es un sistema monolítico que divide las tareas de localización y mapeo en distintos hilos de ejecución, propusimos un nuevo diseño que permite la ejecución de manera distribuida de sus componentes. De esta manera se podría lograr mayor versatilidad a la hora de contar con distintos tipos de hardware con capacidades distintas de procesamiento. Entonces, un robot móvil podría ejecutar parte del procesamiento correspondiente al módulo de localización (*tracking*) de manera *onboard* y por otro lado ejecutar el módulo de mantenimiento del mapa (*mapping*) en un dispositivo externo que de disponga mayor poder de procesamiento, ya que esta tarea es la de mayor costo computacional. Deesta manera, cada módulo dispondrá exclusivamente de toda la capacidad de procesamiento del dispositivo en el que se ejecute.

La arquitectura distribuida permite también numerosas con- figuraciones, como ejecutar sólo el *Tracker*, el *Tracker* junto con el *Mapper* o incluso configuraciones más avanzadas, como por ejemplo tener más de un *Mapper* analizando cada uno una sección distinta del mapa.

Como trabajo futuro se propone adaptar el módulo de *Loop Closing* recientemente planteado por Pire et al. (2017) a la versión de S-PTAM distribuida aquí propuesta.

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.37.123-133.2020 - ISSN 1666-6933

REFERENCIAS

Bailey, T. & Durrant-Whyte, H. (2006). Simultaneous localization and mapping (SLAM): part II. *IEEE Robotics Automation Magazine*, 13(3):108–117.

Bischof, H. (2012). Dense reconstruction on-the-fly. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 1450–1457, Washington, DC, USA. IEEE Computer Society.

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scara- muzza, D., Neira, J., Reid, I., & Leonard, J. J. (2016). Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust- Perception Age. *IEEE Transactions on Robotics*, 32(6):1309–1332.

Durrant-Whyte, H. & Bailey, T. (2006). Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110.

Engel, J., Schöps, T., & Cremers, D. (2014). LSD- SLAM: Large-Scale Direct Monocular SLAM. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 834–849, Cham. Springer International Publishing.

Engel, J., Stückler, J., & Cremers, D. (2015). Large- scale direct slam with stereo cameras. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1935–1942. IEEE.

Forster, C., Lynen, S., Kneip, L., & Scaramuzza, D. (2013). Collaborative monocular slam with multiple micro aerial vehicles. In *IROS*, pages 3962–3970. IEEE.

Forster, C., Zhang, Z., Gassner, M., Werlberger, M., & Scaramuzza, D. (2017). SVO: Semidirect visual odometry for monocular and multicamera systems. *IEEE Trans. Robotics*, 33(2):249–265.

Gamage, R. & Tuceryan, M. (2016). An experimental distributed framework for distributed simultaneous localization and mapping. In *Electro Information Technology (EIT), 2016 IEEE International Conference on*, pages 0665–0667. IEEE.

Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision Meets Robotics: The KITTI Dataset. *The International Journal of Robotics Research, IJRR*, 32(11):1231–1237.

Jazwinski, A. H. (1970). *Stochastic processes and filtering theory*. Mathematics in science and engineering. Academic press, New York. UKM.

Klein, G. & Murray, D. (2007). Parallel Tracking and Mapping for Small AR Workspaces. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 1–10, Washington, DC, USA. IEEE Computer Society.

Mahdoui, N., Natalizio, E., & Frémont, V. (2016). Multi-UAVs network communication study for distributed visual simultaneous localization and mapping. In *ICNC*, pages 1–5. IEEE Computer Society.

Metropolis, N. & Ulam, S. (1949). The Monte Carlo Method. *Journal of the American Statistical Association*, 44(247):335–341. PMID: 18139350.

Mur-Artal, R. & Tardós, J. D. (2016). ORB- SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. *CoRR*, abs/1610.06475.

Pire, T., Fischer, T., Castro, G., De Cristóforis, P.,Civera, J., & Jacobo Berlles, J. (2017). S-PTAM: Stereo Parallel Tracking and Mapping. *Robotics and Autonomous Systems (RAS)*, 93:27 – 42.

Pire, T., Fischer, T., Civera, J., De Cristóforis, P., & Berlles, J. J. (2015). Stereo parallel tracking and mapping for robot localization. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1373–1378.

Riazuelo, L., Civera, J., & Montiel, J. M. M. (2014). C2TAM: A cloud framework for cooperative tracking and mapping. 62:401– 413.

Strasdat, H., Davison, A. J., Montiel, J. M. M., & Konolige, K. (2011). Double window optimisation for constant time visual slam. In *ICCV*, pages 2352–2359. IEEE Computer Society.

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.37.123-133.2020 - ISSN 1666-6933

Triggs, B., McLauchlan, P. F., Hartley, R. I., & Fitzgibbon, A. W. (2000). *Bundle Adjustment — A Modern Synthesis*, chapter 21, pages 298–372. Springer Berlin Heidelberg, Berlin, Heidelberg.

Williams, R., Konev, B., & Coenen, F. (2014). Multi-agent environment exploration with ar.drones. In *TAROS*, volume 8717 of *Lecture Notes in Computer Science*, pages 60–71. Springer.

Williams, R., Konev, B., & Coenen, F. (2015). Scalable distributed collaborative tracking and mapping with micro aerial vehicles. In *IROS*, pages 3092–3097. IEEE.