ISBN: 978-950-42-0200-4
DOI: https://doi.org/10.33414/ajea.5.651.2020

AJEA – Actas de Jornadas y Eventos Académicos de UTN
V Jornadas de Intercambio y Difusión de los Resultados de Investigaciones de los Doctorandos en Ingeniería

Consideraciones de implementación para algoritmos de exploración robótica en plataformas de procesamiento limitado

Implementation considerations for robotic exploration algorithms on platforms with hardware constraints

Presentación: 06/10/2020

Doctorando:
Martin Nievas
Universidad Tecnológica Nacional, Facultad Regional Córdoba - Argentina
martin.nievas.ar@gmail.com

Director/a:
Gastón R. Araguás

Co-director/a:
Claudio J. Paz

Resumen

El proceso de exploración de un ambiente no estructurado presenta numerosas dificultades a la hora de implementar soluciones en plataformas con capacidades de procesamiento limitado. En el presente trabajo se mencionan las técnicas de exploración comúnmente utilizadas y consideraciones para su implementación en sistemas de hardware limitado. Se analizan problemas encontrados con mapas topológicos, en particular aquellos en forma de árbol durante la exploración. También se exponen resultados obtenidos con la plataforma Jetson Nano para procesar una nube de puntos generada mediante la cámara Intel Realsense D435. Se explican detalles a tener en cuenta con la velocidad de memoria de la misma.

Palabras clave: Exploración, Ambientes no estructurados, procesamiento limitado, Jetson Nano

Abstract

The process of exploring an unstructured environment presents several difficulties during the implementation on platforms with limited processing capabilities. Commonly used exploration techniques, and considerations for their implementation in limited hardware systems are mentioned in this paper. Problems encountered with topological maps are analyzed, particularly those in the form of a tree during exploration. The results obtained with the Jetson Nano platform to process a point cloud generated by the Intel Realsense D435 camera are also exposed. Details are explained to take into account with the memory speed are also explained

Keywords: Exploration, Unstructured environments, Hardware constraints, Jetson Nano

Introducción

La exploración de un ambiente con robots es una tarea compleja, que está sujeta a muchos factores. Entre los mismos, podemos mencionar la calidad de las mediciones en los sensores. La estrategia de exploración, entre otros factores, estará fuertemente ligada a los sensores disponibles por el robot, como también del mapa que se desee construir.

Entre las formas de representación más utilizadas, podemos encontrar por un lado los mapas métricos. Donde la información es almacenada como una grilla de resolución constante o variable, tanto para 2D (Engel, Schöps, & Cremers, 2014; Mur-Artal, Montiel, & Tardos, 2015), como para 3D (Hornung, Wurm, Bennewitz, Stachniss, & Burgard, 2013).

Por otro lado, los mapas topológicos (Blochliger, Fehr, Dymczyk, Schneider, & Siegwart, 2018; Palacios & López, 2019), mantienen una representación en forma de grafo del ambiente. Estos últimos tienen como ventaja un menor espacio ocupado en memoria, ya que solo es necesario almacenar la información de los nodos y sus conexiones mediante aristas.

Los mapas métricos tienen la ventaja de poder representar obstáculos y escenarios más complejos. En los mismos, se puede aplicar el concepto de fronteras de exploración tanto a 2D (Yamauchi, 1997), como a 3D (Zhu, Ding, Lin, & Wu, 2015), donde se obtienen buenos resultados y el problema pasa a ser el de elegir la mejor frontera a visitar. Estos mapas, por sus características, rápidamente ocupan gran cantidad de memoria, y no son fácilmente aplicables a sistemas embebidos, en particular aquellos con las restricciones de vuelo para MAVs (Ramaithitima, Whitzer, Bhattacharya, & Kumar, 2016).

En un trabajo anterior (Nievas, Paz, & Araguás, 2019) se analizó la utilización de tablas de Hash en conjunto con árboles, para mejorar la velocidad de búsqueda de los nodos en mapas topológicos. En el mismo, como era de esperarse, las tablas de hash permiten un tiempo de acceso casi constante a los diferentes nodos del árbol almacenado como mapa. También se explica que no siempre es posible generar árboles binarios en ambientes interiores, lo que puede incrementar la cantidad de nodos utilizados. Sumado a esto, si bien los árboles de búsqueda siempre aseguran un camino entre la raíz y alguna de sus hojas, el mapa obtenido, no siempre es el óptimo. Consideremos el caso en que un robot entra en una habitación, rodea una mesa y llega de nuevo a la puerta de entrada, como podemos ver en la Figura 1.

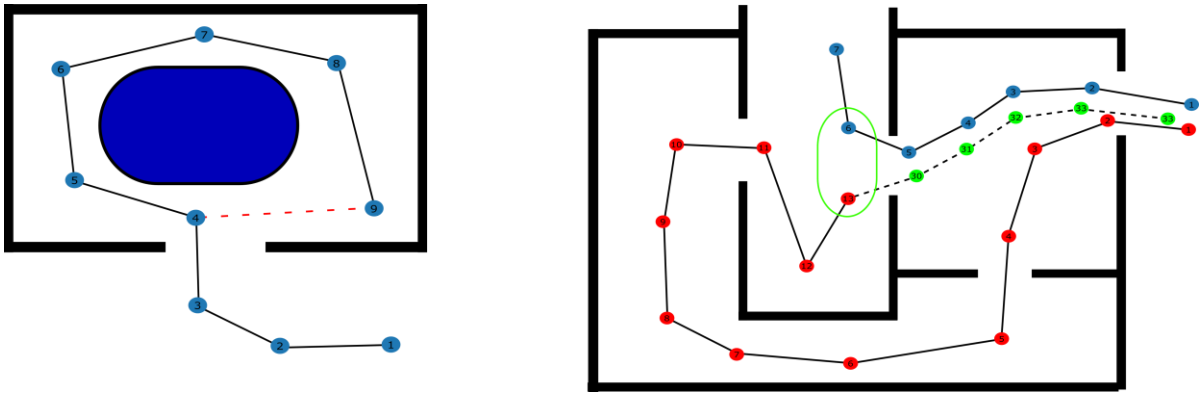

Figura 1: Mapas topológicos. Izquierda: ejemplo de exploración sub óptima por condición del árbol. Derecha: El robot azul al encontrarse con el rojo, le comunica que hay un camino más corto para volver a la base. El robot rojo regresa por el nuevo camino verde, a la par del azul.

El robot tendría que regresar rodeando la mesa, ya que no puede generar un ciclo en el mapa (porque almacenamos en forma de árbol el mapa). Si bien esto es un problema con el que tendrá que lidiar el planificador en un paso posterior, hay que tener en cuenta el incremento de distancia recorrida y, por ende, el trabajo y energía utilizada. Por otro lado en la Figura 1 derecha, se presenta el caso en que dos robots se encuentran en el medio del proceso de exploración, e intercambian los mapas generados para calcular el camino más corto al inicio.

Una de las motivaciones de la presente investigación, es generar un mapa a medida que el robot se desplaza en el ambiente, y pueda utilizarse para regresar al punto de partida, teniendo en cuenta las limitaciones de hardware de las plataformas móviles. Similar a lo ocurrido en tareas de búsqueda y rescate (Murphy, y otros, Search and Rescue Robotics, 2008), donde el robot es desplegado en la entrada de un ambiente no conocido, y posteriormente recuperado.

En el ámbito de búsqueda y rescate, podemos considerar la búsqueda como una actividad concentrada en el interior de una estructura, en cuevas o túneles, y tiene como objetivo encontrar una víctima o posibles peligros. El objetivo es la velocidad y la integridad, sin aumentar el riesgo para las víctimas o los rescatistas (Murphy, y otros, Search and rescue robotics, 2008).

Por otro lado, el reconocimiento y la construcción de un mapa, es más amplio que la búsqueda. Proporciona a los rescatistas una conciencia general de la situación y crea una referencia del entorno destruido. El objetivo es una cobertura rápida de un área grande de interés con la resolución adecuada a la tarea a desarrollar.

En trabajos recientes (Korb & Schöttl, 2018; Bhowmick, Deb, & Mukhopadhyay, 2018) indican que es posible guiar la exploración mediante un mapa topológico, mientras se mantiene al mismo tiempo un mapa métrico del ambiente. En (Korb & Schöttl, 2018) incorporan una forma de exploración basada en árboles de frontera. Los mismos sirven como una memoria del pasado, en donde los límites entre la región conocida y la desconocida son introducidos al árbol. La estructura de árbol es utilizada para decidir futuros objetivos de exploración, a diferencia de los enfoques voraces (o Greedy en inglés). Para cumplir la tarea de exploración, el algoritmo utiliza cinco mapas métricos diferentes, lo cual lo vuelve poco atractivo para plataformas de recursos limitados, debido al gran uso de memoria por parte de estos mapas.

En (Bhowmick, Deb, & Mukhopadhyay, 2018) presentan un algoritmo que toma como entrada imágenes y almacena las características en un KD-Tree (Árbol binario de k dimensiones). Representan los lugares por sus características, utilizando LBP (Local Binary Patterns) como descriptor, y se almacenan en los nodos del mapa. Sin embargo, al evaluar el algoritmo contra una base de datos, el problema se convierte en recuperar la imagen más parecida de la base de datos.

En (Oleynikova, Taylor, Siegwart, & Nieto, 2018) utilizan un ESDF (Euclidean Signed Distance Field en inglés), del cual extraen un diagrama de Voronoi generalizado para ambientes 3D. Con esto, obtienen un esqueleto del ambiente en el espacio de voxel, el cual es convertido a una representación topológica en forma de grafo disperso.

En (Palacios & López, 2019) presentan un método basado en la exploración de un grafo, donde define una forma sistemática de analizar la siguiente posición a explorar, eliminando la aleatoriedad en el proceso de decisión. Con esto se consigue minimizar los movimientos que el robot tiene que realizar para alcanzar la posición y por consiguiente el tiempo que demora en llegar a ese lugar. La representación del ambiente más próximo al robot se define como LSR, y es el área en la que el robot puede navegar sin colisiones. En cada iteración del algoritmo, se evalúan las fronteras de los nodos adyacentes al nodo actual, para determinar si las fronteras son o no parte de la LSR actual.

Desarrollo

Los mapas topológicos, por su naturaleza utilizan menos memoria, lo que los convierte en una opción atractiva para robots con limitado poder de procesamiento. Los algoritmos para la construcción de un mapa topológico podemos clasificarlos en dos grupos: Aquellos que parten de una mapa métrico ya construido (Blochliger, Fehr, Dymczyk, Schneider, & Siegwart, 2018) o progresivamente construyen el mapa (Palacios & López, 2019).

Debido a la naturaleza del problema de búsqueda y rescate, se optó por la construcción en forma progresiva del mapa. En donde cada robot almacenará un mapa topológico de la región explorada hasta el momento. Esto trae como ventaja, poder utilizar plataformas de bajo costo y consumo como la serie Jetson de NVIDIA™.

Actualmente, se está trabajando en la implementación en CUDA (Compute Unified Device Architecture en inglés), del algoritmo recientemente presentado: STVL (Macenski, Tsai, & Feinberg, 2020). Se espera correr el mismo a bordo de una Computadora Jetson Nano, la cual integra una GPU embebida. Autores han indicado (Peng, Zhang, Liu, Asari, & Loomis, 2019; Jo, Jeong, & Kang, 2020) que la utilización de la GPU dedicada en éstas plataformas reduce el consumo y el tiempo de procesamiento. Esto es debido a que las arquitecturas encontradas en las GPU tienen un mayor rendimiento en problemas masivamente paralelos, como el procesamiento de nubes de puntos.

Los mapas de profundidad se obtienen por medio de una cámara Intel Realsense D435, las cuales, son procesadas con la conocida biblioteca PCL (Point Cloud Library). Este procesamiento es realizado en un nodo de ROS (Robot Operating System), lo cual permitirá su posterior implementación en forma directa sobre un robot. En la Figura 2 puede observarse el sistema

armado para las pruebas iniciales. El mismo consta de una placa Jetson Nano en un gabinete impreso en 3D, al cual se le añadió un ventilador lateral. También, se puede ver la cámara de profundidad, montada en el soporte utilizado para las pruebas.

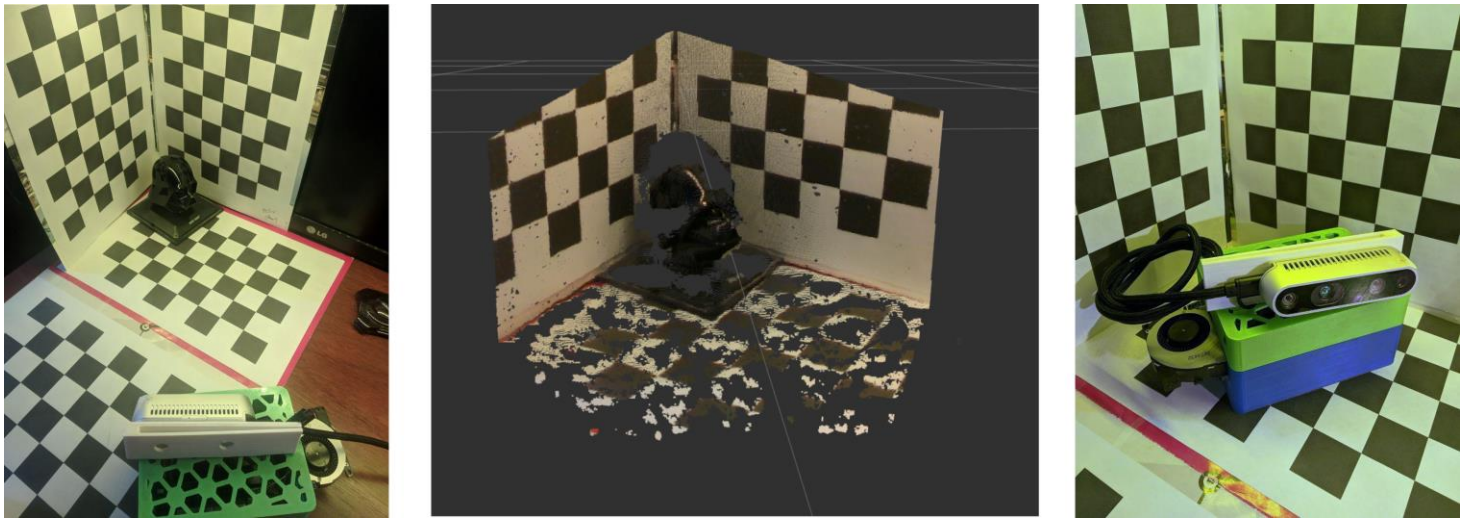

Figura 2: Sistema armado para las pruebas en el escritorio con la placa Jetson Nano + cámara Intel Realsense D435. En la imagen del centro se puede observar la nube de puntos generada.

Se esperaba realizar pruebas en los robots ROMAA-II (Paina, 2014) disponibles en el centro de investigación de la facultad, los cuales estaban preparados para realizar las pruebas en junio, pero debido al ASPO (Aislamiento Social Preventivo y Obligatorio) dictado en marzo (Nación, 2020), no se han podido efectuar. Sin embargo, gracias al trabajo previo utilizando simuladores (Nievas, Paz, & Araguás, 2019), se obtuvieron resultados prometedores en simulaciones.

Otro aspecto importante y bien conocido es que la exploración puede ser acelerada utilizando varios robots, ya que los mismos pueden cooperar y aprovechar las vistas del entorno captadas por los demás. Sin embargo, esto es así cuando existe una comunicación entre todos los robots. Si el conocimiento del mapa está distribuido en el conjunto de robots, la pérdida de un robot provocará una pérdida de información. Para tratar este problema, se está trabajando con la autora de (Pereyra, Araguás, & Kulich, 2017), para la implementación del sistema presentado. En el mismo, se emplea un esquema de planificación para múltiples robots, en los cuales se coordinan puntos intermedios de encuentro y sincronización. Se espera que los mismos sean utilizados para intercambiar información entre los robots, y regresar a la base de forma rápida.

Debido a que se planea implementar el algoritmo en múltiples robots, se utilizarán contenedores Docker. En trabajos anteriores (Nievas, Paz, & Araguás, 2019) se demostró que resultan útiles a la hora de replicar experimentos, y en nuestro caso, desplegar el mismo programa en diferentes plataformas. La única restricción es que el robot cuente con un sistema operativo que soporte el uso de contenedores, que para el caso de las plataformas Jetson de NVIDIA™ está soportado desde la versión de Jetpack 4.4, públicamente disponible (Franklin, 2020).

Conclusiones

Las primeras pruebas realizadas con la plataforma Jetson demuestran que se pueden obtener mejoras sustanciales al utilizar la GPU embebida. Una de las mayores limitantes encontrados es la velocidad de memoria, ya que es compartida entre la CPU y GPU. En la página oficial de la Jetson NANO podemos encontrar que la RAM utilizada es 4 GB 64-bit LPDDR4 con una velocidad de 25.6 GB/s. Pero las pruebas realizadas demuestran que la velocidad de lectura es menor, llegando a los 16GB/s en lecturas y escritura de memoria alineadas por parte de la GPU. Si comparamos este valor con los 196GB/s encontrados en una Nvidia GTX 970, de micro arquitectura Maxwell igual que la Jetson, hay una relación de 7.6 veces. Sumado a esto, la cantidad de SM (Streaming Multiprocessors) es inferior en la Jetson (1 frente a los 13 de la GTX 970). Pero por el lado del consumo, la GPU de escritorio llega a 145W, mientras que el sistema embebido mencionado solo a 10W. Este último valor es importante, ya que en robots de búsqueda y rescate la autonomía es un factor clave para su utilización.

Estos valores deben ser considerados, ya que varios esquemas de programación paralela aprovechan la velocidad de memoria de la GPU para realizar una gran cantidad de cálculos. Pero en el caso de sistemas embebidos, la velocidad de la memoria no llega a ser comparable con las versiones de escritorio. Para sacarle mejor rendimiento a la GPU embebida en estas plataformas, tenemos que tener presente que no hay transferencia de memoria entre dispositivo y CPU, ya que la memoria es compartida entre ambas.

Referencias

Bhowmick, S., Deb, A. K., & Mukhopadhyay, J. (2018). Monocular Vision based Topological Map Generation in Real-time. *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, (pp. 791–798).

Blochliger, F., Fehr, M., Dymczyk, M., Schneider, T., & Siegwart, R. (2018). Topomap: Topological mapping and navigation based on visual slam maps. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, (pp. 1–9).

Engel, J., Schöps, T., & Cremers, D. (2014). LSD-SLAM: Large-scale direct monocular SLAM. *European conference on computer vision*, (pp. 834–849).

Franklin, D. (2020, 2). dusty-nv/jetson-containers. *dusty-nv/jetson-containers*. NVIDIA. Retrieved from https://github.com/dusty-nv/jetson-containers

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., & Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots, 34*, 189–206.

Jo, J., Jeong, S., & Kang, P. (2020). Benchmarking GPU-Accelerated Edge Devices. *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, (pp. 117–120).

Korb, R., & Schöttl, A. (2018). Exploring Unstructured Environment with Frontier Trees. *Journal of Intelligent & Robotic Systems, 91*, 617–628.

Macenski, S., Tsai, D., & Feinberg, M. (2020). Spatio-temporal voxel layer: A view on robot perception for the dynamic world. *International Journal of Advanced Robotic Systems, 17*, 1729881420910530.

Mur-Artal, R., Montiel, J. M., & Tardos, J. D. (2015). ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE transactions on robotics, 31*, 1147–1163.

Murphy, R. R., Tadokoro, S., Nardi, D., Jacoff, A., Fiorini, P., Choset, H., & Erkmen, A. M. (2008). Search and rescue robotics. *Springer handbook of robotics*, 1151–1173.

Murphy, R. R., Tadokoro, S., Nardi, D., Jacoff, A., Fiorini, P., Choset, H., & Erkmen, A. M. (2008). Search and Rescue Robotics. In B. Siciliano, & O. Khatib (Eds.), *Springer Handbook of Robotics* (pp. 1151–1173). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-540-30301-5_51

Nación, P. d. (2020, 03 19). *AISLAMIENTO SOCIAL PREVENTIVO Y OBLIGATORIO - Decreto 297/2020.* Retrieved from boletin oficial: https://www.boletinoficial.gob.ar/detalleAviso/primera/227042/20200320

Nievas, M., Paz, C. J., & Araguás, G. R. (2019). Mejorando la exploración de mapas topológicos mediante tabla de Hash. *X JAR - Jornadas argentinas de robótica 2019*.

Nievas, M., Paz, C. J., & Araguás, R. G. (2019). Dockerización de ROS para despliegue ágil de algoritmos de exploración. *I Simposio Argentino de Informática Industrial e Investigación Operativa (SIIIO 2019)-JAIIO 48 (Salta).*

Oleynikova, H., Taylor, Z., Siegwart, R., & Nieto, J. (2018). Sparse 3d topological graphs for micro-aerial vehicle planning. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (pp. 1–9).

Paina, G. P. (2014). Romaa-ii, an open architecture mobile robot. *IEEE Latin America Transactions*, 915-921.

Palacios, A. T., & López, A. S. (2019). Extending the Limits of the Random Exploration Graph for Efficient Autonomous Exploration in Unknown Environments. In *Path Planning for Autonomous Vehicles.* IntechOpen.

Peng, T., Zhang, D., Liu, R., Asari, V. K., & Loomis, J. S. (2019). Evaluating the Power Efficiency of Visual SLAM on Embedded GPU Systems. *2019 IEEE National Aerospace and Electronics Conference (NAECON)*, (pp. 117–121).

Pereyra, E., Araguás, G., & Kulich, M. (2017). Path planning for a formation of mobile robots with split and merge. *International Workshop on Modelling and Simulation for Autonomous Systems*, (pp. 59–71).

Ramaithitima, R., Whitzer, M., Bhattacharya, S., & Kumar, V. (2016). Automated Creation of Topological Maps in Unknown Environments Using a Swarm of Resource-Constrained Robots. *IEEE Robotics and Automation Letters, 1*, 746–753. doi:10.1109/LRA.2016.2523600

Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, (pp. 146–151).

Zhu, C., Ding, R., Lin, M., & Wu, Y. (2015). A 3D frontier-based exploration tool for MAVs. *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, (pp. 348–352).