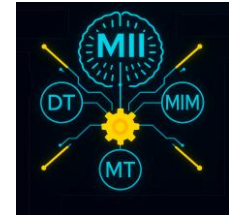


# COMPARATIVA DE ALGORITMOS DE SLAM CON ROS Y PYTHON



Conference Proceedings CIFIN – I 2025  
Noviembre 26-28 2025. Pag. XX-XX

ISSN (Online):

**Josue, Rodas**

Universidad  
Autónoma de Ciudad  
Juárez. Ciudad  
Juárez, Chihuahua,  
[josuecarlosrr@gmail.com](mailto:josuecarlosrr@gmail.com)

**Ángel, Soto**

Universidad  
Autónoma de Ciudad  
Juárez. Ciudad  
Juárez, Chihuahua,  
[angel.soto@uacj.mx](mailto:angel.soto@uacj.mx)

**Israel, Ponce**

Universidad  
Autónoma de Ciudad  
Juárez. Ciudad  
Juárez, Chihuahua,  
[israel.ulises@uacj.mx](mailto:israel.ulises@uacj.mx)

**Francesco,  
García**

Universidad  
Autónoma de Ciudad  
Juárez. Ciudad  
Juárez, Chihuahua,  
[francesco.garcia@uacj.mx](mailto:francesco.garcia@uacj.mx)

## Resumen:

*Uno de los problemas en la navegación de robots móviles es identificar las características del entorno en que se encuentran y establecer su postura, posición y orientación, en este espacio de trabajo. Una de las alternativas para resolver este problema es la técnica llamada Localización y Mapeo Simultáneo (SLAM, por sus siglas en inglés); la cual requiere la fusión de sensores que monitoreen el comportamiento del robot y entorno. Este trabajo presenta una comparación entre tres algoritmos de SLAM en tres entornos simulados mediante el uso de Robot Operating System (ROS) y Python. La comparación de dichas técnicas toma en cuenta los recursos computacionales que estos algoritmos requieren para ejecutar la simulación, tales como uso de CPU y memoria RAM. Con el fin de que los resultados presentados tengan una comparación justa, el desarrollo de las simulaciones se realiza en situaciones idénticas y los resultados obtenidos son evaluados mediante el análisis de varianza (ANOVA) y el software especializado Minitab Statistical. Estos muestran que, a pesar de que los algoritmos implementados logran crear mapas funcionales, presentan diferencias en cuanto a la cantidad de recursos que cada uno de estos utiliza siendo el algoritmo Gmapping el que requiere un mayor uso de CPU mientras que el algoritmo Hector hace un mayor uso de memoria RAM. Este estudio permite evaluar la aplicación y comparación de algoritmos SLAM con base en su funcionamiento y consumo de recursos computacionales para que sea considerado al momento de su aplicación.*

**Palabras clave:** SLAM, ROS, Gmapping, Karto, Hector.

## 1 INTRODUCCIÓN

La técnica Localización y Mapeo Simultáneos (SLAM por sus siglas en inglés) es una tecnología que permite a un robot móvil mapear un entorno desconocido, tomando en cuenta su posición de manera simultánea sin necesitar sistemas de posicionamiento externos (Alsadik & Karam, 2021). Esta técnica tiene algunas variantes, entre las que destacan los algoritmos que utiliza: Cartographer, Gmapping, Hector SLAM, Karto SLAM y RTAB-Map SLAM. Estos hacen uso de sensores tales como el sensor LiDAR e incluso cámaras de profundidad conocidas como RGB-Dept para el mapeo del entorno (Trejos et al., 2022).

Para hacer uso de esta tecnología es necesario tener acceso y capacidad de operar múltiples sensores que incorpora el robot que se usará. Para poder hacer esto es necesario tener una interfaz que permita la comunicación junto con el uso de información. Uno de los sistemas ampliamente utilizados es el sistema operativo robótico ROS, esto debido a que tiene por objeto facilitar el

control, planeación, simulación y desarrollo de los robots mediante paquetes de programación (Cuevas Castañeda, 2016; Herath & St-Onge Eds, 2022).

Es bien sabido que cada sistema presenta ventajas y desventajas, de ahí que el objetivo de este trabajo es realizar una comparación entre el uso de CPU y memoria RAM de tres algoritmos de la técnica SLAM (Gmapping, Karto y Hector-SLAM), así como de los resultados obtenidos del mapeo de ambientes simulados en el entorno de Gazebo, utilizando un robot TurtleBot3, el sistema ROS y el lenguaje de programación Python, utilizando el sistema Ubuntu en la herramienta WSL de Windows 11 home.

## **2 MARCO TEÓRICO Y CONCEPTUAL**

Las aplicaciones de la técnica SLAM son variadas y entre estas se incluyen: vehículos autónomos aéreos y terrestres, automóviles autónomos, realidad aumentada, mapeo de entornos exteriores e interiores, entre otros. Para que esta pueda ser aplicada es necesario que el robot cuente con sensores de abordaje que monitoreen la trayectoria que este tiene mediante sensores de odometría que monitorean la ubicación del robot en tiempo real, y sensores que mapeen el entorno. Algunos de estos son los sensores LiDAR, emisores y receptores GNSS, cámaras y unidades de medición de inercia (Alsadik & Karam, 2021).

A pesar de que esta técnica puede hacer uso de estos dos tipos de sensores, no todos los algoritmos necesitan los dos tipos para operar. Para el caso del algoritmo Gmapping, un algoritmo basado en el filtro de partículas Rao Blackwellized (RBPF) (Olalekan et al., 2021), se utiliza la información de los sensores de las ruedas del robot, así como la recabada por el sensor laser para mapear y localizar el robot en el entorno (Pramod Thale et al., 2020).

De manera similar al algoritmo previamente explicado, el algoritmo Karto también hace uso de la odometría del robot y del sensor laser para realizar el mapeo. Contrario a Gmapping, este es un algoritmo basado en gráficos que genera un mapa mediante nodos que almacenan la postura y la posición (Le et al., 2018). Este basa su operación en matrices Cholesky para minimizar el error obtenido y en un sistema de lazo cerrado para conectar los nodos que contengan la misma información y obtener la mejor información posible (Trejos et al., 2022).

El tercer algoritmo difiere de los dos mencionados anteriormente ya que este hace uso únicamente de la información recabada por el sensor laser sin tomar en cuenta la odometría del robot lo cual lo hace óptimo para aplicaciones que requieren un mejor aprovechamiento de recursos. No obstante, también es capaz de utilizar dichos sensores con el fin de reducir el error que se tenga del sensor laser (Patil et al., 2023; Sirigool & Kesvarakul, 2020).

## **3 METODOLOGÍA**

En este trabajo se realizó la comparación de tres variantes de la técnica SLAM y tres entornos simulados utilizando Gazebo; esto para probar el uso de recursos de cada uno de ellos cuando realizan el mapeo de los entornos. Se utilizó un robot turtlebot3 del tipo burger que integra sensores de odometría y un sensor LiDAR.

Utilizando estas variables y realizando un escaneo para cada una de las combinaciones, se tuvo un total de nueve muestras, ya que se combinaron el algoritmo usado y el entorno mapeado, manteniendo el tipo de robot como una constante.

Los tres entornos mapeados se muestran en las Figuras 1, 2 y 3, en las cuales se observan los límites de los entornos, los obstáculos presentes, así como el robot en su posición de origen en tres diferentes vistas.

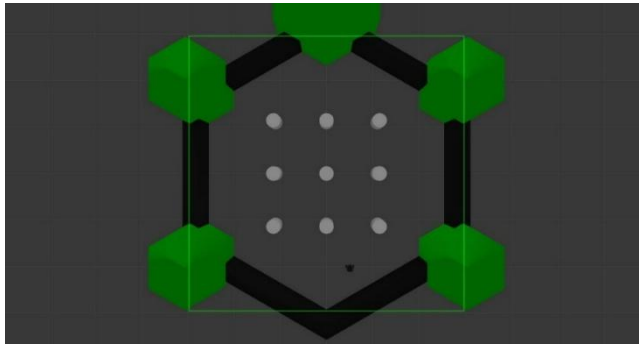


Figura 1 Mapa número 1: Hexágono

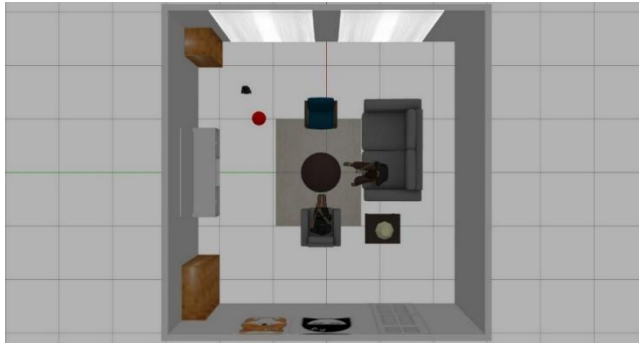


Figura 2 Mapa número 2: Sala de estar



Figura 3 Mapa número 3: Stage 4

Para controlar el movimiento del robot se usó un script en lenguaje Python corriendo directamente desde la terminal de Ubuntu para cada uno de los mundos el cual definió la ruta a seguir, así como los parámetros de: velocidad lineal, establecida en  $0.1 \frac{m}{s}$ , velocidad angular, establecida en  $0.4 \frac{rad}{s}$ , aceleración, establecida en  $0.01 \frac{m}{s^2}$  y desaceleración, establecida en  $-0.01 \frac{m}{s^2}$ .

Para tener una comparativa justa, se utilizaron tres scripts que ejecutaban una trayectoria definida para el robot y que lanzaban un script secundario que almacenaba los datos del uso de recursos del equipo que estaba siendo usado. Cada mapa tuvo su propia trayectoria que fue ejecutada para cada uno de los algoritmos sin modificaciones.

## 4 RESULTADOS

Para analizar los resultados, se aplicó un ANOVA de dos factores con una confiabilidad del 95% y una significancia del 5%, estableciendo las hipótesis para cada factor y la interacción de estos como  $H_0: \mu_1 = \mu_2$  y  $H_1: \mu_1 \neq \mu_2$ .

Para el uso del CPU se calculó un valor  $R^2 = 29.79\%$ , obteniendo los resultados mostrados en la Tabla 1 mientras que para el uso de la memoria RAM se obtuvo un valor  $R^2 = 98.28\%$  así como los resultados mostrados en la

Tabla 2.

*Tabla 1 Resultado del ANOVA de dos factores para el uso del CPU*

Fuente	DF	Adj SS	Adj MS	F-Value	P-Value
Mundo	2	233.2	116.62	7.66	0.000
Algoritmo	2	11684.5	5842.23	383.77	0.000
Mundo*Algoritmo	4	1064.6	266.16	17.48	0.000
Error	2181	33202.0	15.22		
Total	2189	47462.4			

*Tabla 2 Resultado del ANOVA de dos factores para el uso de la memoria RAM*

Fuente	DF	Adj SS	Adj MS	F-Value	P-Value
Mundo	2	708305	354153	7480.47	0.000
Algoritmo	2	4762495	2381247	50297.09	0.000
Mundo*Algoritmo	4	4860	1215	25.66	0.000
Error	2181	103256	47		
Total	2189	6030418			

Posteriormente se realizó un ANOVA de un factor a cada entorno para determinar si existen diferencias entre estos y el tiempo que toma la ejecución de cada trayectoria. En todos los análisis, el valor P fue menor al valor de significancia indicando que existe al menos un algoritmo que utiliza más o menos recursos.

Para el entorno Hexágono, se obtuvo un valor  $R^2 = 27.45\%$  para el uso del CPU y  $R^2 = 97.58\%$  para el uso de la memoria RAM. Los datos obtenidos se comparan utilizando su media, así como su agrupación por el método de Tukey en la Tabla 3.

*Tabla 3 Comparativa de los tres algoritmos en el entorno Hexágono*

Algoritmo	Entorno Hexágono				
	N	Media uso de CPU	Agrupación uso de CPU	Media uso de memoria RAM	Agrupación uso de memoria RAM
Gmapping	242	24.290	A	1097.22	B

Hector	242	18.972	B	1183.67	A
Karto	242	18.388	B	1065.80	C

Para el entorno Living room, el uso del CPU tuvo un valor  $R^2 = 36.69\%$  mientras que el uso de la memoria RAM tuvo un valor  $R^2 = 98.44\%$ . Este obtuvo los resultados mostrados en la Tabla 4.

*Tabla 4 Comparativa de los tres algoritmos en el entorno Living Room*

Algoritmo	Entorno Living Room				
	N	Media uso de CPU	Agrupación uso de CPU	Media uso de memoria RAM	Agrupación uso de memoria RAM
Gmapping	157	24.276	A	1146.90	B
Hector	157	19.321	B	1231.13	A
Karto	157	19.431	B	1116.36	C

Para el ultimo análisis realizado, el entorno Stage 4, el uso del CPU arrojó un valor  $R^2 = 29.03\%$ , aunque el uso de la memoria RAM tuvo un valor  $R^2 = 98.24\%$ . Los resultados obtenidos se presentan en la Tabla 5.

*Tabla 5 Comparativa de los tres algoritmos en el entorno Stage 4*

Algoritmo	Entorno Stage 4				
	N	Media uso de CPU	Agrupación uso de CPU	Media uso de memoria RAM	Agrupación uso de memoria RAM
Gmapping	331	23.323	A	1108.27	B
Hector	331	17.109	C	1192.22	A
Karto	331	20.067	B	1084.05	C

## 5 CONCLUSIONES

Con base en los resultados obtenidos, se puede observar que todos los ANOVA reportan un valor P menor al nivel de significancia, con lo cual se puede concluir, con una confiabilidad del 95%, que en todos los casos hay al menos un algoritmo que utiliza menos recursos que los demás.

No obstante, es necesario considerar los ANOVA posteriores para evaluar cuál algoritmo se desempeña mejor que el resto. Estos muestran al algoritmo Gmapping-SLAM haciendo un mayor uso del CPU en los tres escenarios, sin embargo, el análisis para esta variable arrojó un valor  $R^2$  menor a 35%, lo cual indica que esta no es bien explicada por los factores considerados, sino que es afectada mayormente por factores aleatorios que pueden incluir: ejecución de procesos en segundo plano, ejecución de RViz, Gazebo, entre otros. Esto indica que el uso de CPU no es una variable de respuesta confiable debido a que es mayormente explicada por eventos aleatorios.

Por otro lado, el uso de la memoria RAM es explicado en un porcentaje mayor al 95% por los factores utilizados. Dentro de este análisis, el algoritmo Hector-SLAM resultó ser el que requiere mayor uso de esta entre los tres algoritmos probados.

Estos resultados permiten concluir que el algoritmo Gmapping hace un mayor uso de CPU mientras que el algoritmo Hector hace un mayor uso de memoria RAM, siendo el algoritmo Karto el considerado más eficiente ya que es el que menos memoria RAM utiliza y es el segundo que menos CPU requiere.

## 6 REFERENCIAS

- Alsadik, B., & Karam, S. (2021). The Simultaneous Localization and Mapping (SLAM)-An Overview. *Journal of Applied Science and Technology Trends*, 2(2), 147–158. <https://doi.org/10.38094/jastt204117>
- Cuevas Castañeda, C. C. (2016). Ros-gazebo. una valiosa Herramienta de Vanguardia para el Desarrollo de la Robótica. *Publicaciones e Investigación*, 10, 145. <https://doi.org/10.22490/25394088.1593>
- Herath, D., & St-Onge Eds, D. (2022). *Foundations of Robotics A Multidisciplinary Approach with Python and ROS*. [https://doi.org/10.1007/978-981-19-1983-1\\_5](https://doi.org/10.1007/978-981-19-1983-1_5)
- Le, X. S., Fabresse, L., Bouraqadi, N., & Lozenguez, G. (2018). Evaluation of out-of-the-box ROS 2D slams for autonomous exploration of unknown indoor environments. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10985 LNAI, 283–296. [https://doi.org/10.1007/978-3-319-97589-4\\_24](https://doi.org/10.1007/978-3-319-97589-4_24)
- Olalekan, A. F., Sagor, J. A., Hasan, M. H., & Oluwatobi, A. S. (2021, May 21). Comparison of two SLAM algorithms provided by ROS (Robot Operating System). *2021 2nd International Conference for Emerging Technology, INCET 2021*. <https://doi.org/10.1109/INCET51464.2021.9456164>
- Patil, M., Thakur, A., Bagul, P., & Yogita Ajar, P. (2023). *Hector SLAM Mapping and Localization System using ROS and LIDAR*.
- Pramod Thale, S., Mangesh Prabhu, M., Vinod Thakur, P., & Kadam, P. (2020). ROS based SLAM implementation for Autonomous navigation using Turtlebot. *ITM Web of Conferences*, 32, 01011. <https://doi.org/10.1051/itmconf/20203201011>
- Sirigool, W., & Kesvarakul, R. (2020). Particle Filter for Hector SLAM to Improve the Performance of Robot Positioning by Image Processing Based. *International Journal of Machine Learning and Computing*, 10(3), 490–494. <https://doi.org/10.18178/ijmlc.2020.10.3.962>
- Trejos, K., Rincón, L., Bolaños, M., Fallas, J., & Marín, L. (2022). 2D SLAM Algorithms Characterization, Calibration, and Comparison Considering Pose Error, Map Accuracy as Well as CPU and Memory Usage †. *Sensors*, 22(18). <https://doi.org/10.3390/s22186903>