DESDE 2013
https://repository.uaeh.edu.mx/revistas/index.php/icbi/issue/archive
Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI
ISSN: 2007-6363

Publicación Semestral Pädi Vol. 11 No. Especial 2 (2023) 103–109

Navegación autónoma de un robot móvil usando ORB-SLAM2
Mobile robot autonomous navigation using ORB-SLAM2

R. Villalobos-Salazar [a,*], M. Castelán [a], P. A. Ochoa-Salinas [a], A. B. Morales-Díaz [a]

[a] *Grupo de Robótica y Manufactura Avanzada, Centro de Investigación y de Estudios Avanzados del IPN-Saltillo, Industrial, Zona Industrial, 25900 Ramos Arizpe, Coahuila, México.*

Resumen

Este trabajo presenta la implementación de un sistema de navegación autónoma para un robot móvil diferencial, que combina la técnica de Localización y Mapeo Simultáneos, *Simultaneous Localization and Mapping* (SLAM, por sus siglas en inglés) ORB-SLAM2 con una cámara RGB-D, utilizando el algoritmo BUG0 para la evasión de obstáculos, integrado en la plataforma ROS (Robot Operating System) bajo el sistema operativo Ubuntu 18.04. El sistema propuesto fue evaluado experimentalmente en un ambiente real y se logró una navegación autónoma eficiente y segura en presencia de obstáculos. Los resultados experimentales mostraron que el sistema fue capaz de generar trayectorias alrededor de obstáculos y mantener una distancia segura para evitar colisiones. Además, se obtuvieron resultados satisfactorios en términos de precisión en la regulación a la posiciona deseada y la generación de trayectorias para la evasión de obstáculos.

Palabras Clave: Robot móvil, navegación autónoma, ORB-SLAM2, ROS.

Abstract

This paper presents the implementation of an autonomous navigation system for a differential mobile robot, which combines the Simultaneous Localization and Mapping (SLAM) technique, ORB-SLAM2 with an RGB-D camera, and the BUG0 algorithm for obstacle avoidance integrated in the ROS (Robot Operating System) framework. The proposed system was experimentally evaluated in a real environment and achieved efficient and safe autonomous navigation in the presence of obstacles. Experimental results showed that the system was capable of generating trajectories around obstacles and maintained a safe distance to avoid collisions. Furthermore, satisfactory results were obtained in terms of accuracy in pose regulation and trajectory generation for obstacle avoidance.

Keywords: Mobile Robot, autonomous navigation, ORB-SLAM2, ROS.

1. Introducción

Brindarle autonomía a un robot móvil representa una gran oportunidad en aplicaciones como la manufactura, logística y distribución, agricultura, seguridad e incluso en el sector salud. En la actualidad esto pude ser logrado con la implementación de una amplia variedad de sensores como encoders, Lidar, IMU, cámaras, etc. (Zghair y Al-Araji, 2021). Las técnicas de navegación visual representan una clara ventaja pues en su mayoría se requiere de un solo sensor (una cámara), razón por la cual actualmente son ampliamente estudiadas. Podemos dividir en dos las técnicas de navegación visual (Macario Barros *et al.*, 2022): en métodos directos y en métodos basados en características. La diferencia radica en que los primeros utilizan toda la información de la imagen, mientras que los otros construyen un mapa apoyándose en la información dada por un método de extracción de características.

Entre las técnicas de navegación visual que se han implementado dentro de ROS (Robot Operating System) se ha demostrado que las que tienen un mejor funcionamiento son las basadas en características (Mingachev *et al.*, 2020) cuando se trata de tareas dentro de una amplia área de trabajo, debido a que los métodos directos trabajan mejor en obtener posiciones locales.

En este artículo, se presenta la implementación de un robot móvil que utiliza el algoritmo ORB-SLAM2 para la navegación autónoma en un ambiente real. Se describe el hardware utili-

* Autor para correspondencia: rodolfo.villalobos@cinvestav.mx
Correo electrónico: rodolfo.villalobos@cinvestav.mx (Rodolfo de Jesús Villalobos-Salazar), mario.castelan@cinvestav.edu.mx (Mario Castelán), pablo.ochoa@cinvestav.mx (Pablo Alonso Ochoa-Salinas), america.morales@cinvestav.edu.mx (América Berenice Morales-Díaz).

Historial del manuscrito: recibido el 17/03/2023, última versión-revisada recibida el 25/05/2023, aceptado el 02/06/2023 publicado el 11/09/2023. **DOI:** https://doi.org/10.29057/icbi.v11iEspecial2.10706

zado, así como los detalles de la integración de ORB-SLAM2 y una estrategia de evasión de obstáculos con todo integrado en la plataforma ROS. Además, se muestran resultados experimentales de la navegación autónoma del robot en presencia de obstáculos.

2. Planteamiento del problema

Los robots móviles pueden ser empleados en diversas tareas, a su vez, su utilidad se ve incrementada si estos son dotados de autonomía para moverse en un ambiente. Actualmente encontramos este tipo de robots en aplicaciones como la manufactura, logística y distribución, agricultura, seguridad e incluso en el sector salud. Posibles ejemplos se muestran en la figura 1.

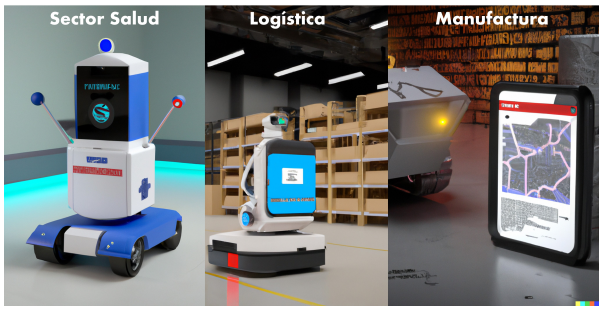

Figura 1: Ilustraciones de posibles aplicaciones de robots móviles autónomos (esta imagen fue generada con la asistencia de DALL·E (Ramesh *et al.*, 2022))

El objetivo de este trabajo es operar en interiores, lo cual supone un reto debido a que soluciones que involucran GPS no son factibles, esto nos lleva a la implementación de otro tipo de sensores exteroceptivos (cámaras, Lidar, entre otros) para brindarle percepción del ambiente al robot, además también se debe considerar que funcionar en interiores representa un espacio más reducido, por lo que la capacidad de maniobrabilidad es importante al buscar un modelo de robot. En la figura 2 observamos una caricatura que representa nuestro objetivo, darle la capacidad a un robot de moverse de un punto a otro de manera autónoma en un ambiente de interiores.

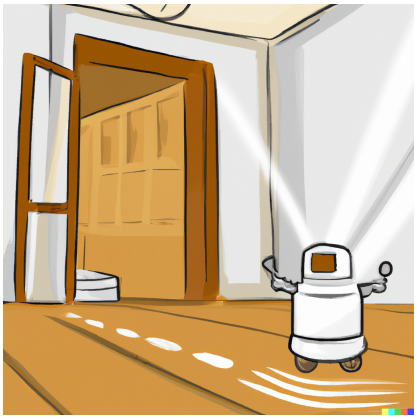

Figura 2: Dibujo de un robot navegando en interiores (esta imagen fue generada con la asistencia de DALL·E (Ramesh *et al.*, 2022))

3. Metodología

De entre los tipos de sensores usados para la navegación autónoma mencionados anteriormente, destacan las cámaras y los sensores Lidar, si bien estos últimos tiene una clara ventaja en términos de precisión, requieren de un mayor costo computacional debido al tipo de información que manejan, además de que tienen una falta de información semántica que podría venir útil en trabajos futuros, por estas razones se decidió utilizar cámaras.

Dentro de las posibles soluciones para la navegación visual hemos seleccionado ORB-SLAM2, al tratarse de un método basado en características permite trabajar en espacios de trabajo más amplios, además de la robustez ofrecida por el algoritmo y bajo costo computacional.

3.1. ORB-SLAM2

ORB-SLAM2 (Mur-Artal y Tardós, 2017) es la versión mejorada del algoritmo ORB-SLAM, utilizado para realizar tareas de mapeo y localización simultáneas (SLAM por sus siglas en inglés) en ambientes desconocidos. El funcionamiento de este algoritmo depende de ORB (Oriented FAST and Rotated BRIEF) (Rublee *et al.*, 2011), un extractor de características que se distingue por ser invariante a la rotación y al escalado, además de ser bastante robusto ante los cambios de iluminación, lo que lo vuelve ideal para entornos cambiantes. El objetivo principal de ORB-SLAM2 es construir un mapa de un ambiente desconocido y localizar la pose estimada del sensor dentro del marco de referencia del mapa en cuestión.

El algoritmo ORB-SLAM2 se compone de tres hilos paralelos. El primero es *Tracking*, que cuenta con una etapa de preprocesado de la imagen de entrada, con el objetivo de que el resto del sistema trabaje sin importar el tipo de sensor que se está utilizando. El hilo *Tracking*, además, localiza la cámara en cada frame encontrando correspondencias de características en el mapa local y minimizando el error de reproyección al aplicar *Motion-only Bundle Adjustment*. El siguiente hilo es el *Local Mapping* el cual se encarga de manejar el mapa local, optimizarlo y ejecutar el *bundle adjustment*. Por ultimo, el tercer hilo, *Loop Closing*, es el encargado de detectar los ciclos y corregir la deriva acumulada. Se podría considerar que aquí se lanza un cuarto hilo, pues se inicia el proceso de *Bundle adjustment* global para calcular la óptima solución del movimiento realizado.

Se eligió utilizar esta herramienta con el objetivo de localizar a un robot móvil dentro de un ambiente de interiores y realizar navegación autónoma dentro del mismo. Un punto clave en la elección de este algoritmo es que los puntos característicos ORB se obtienen con rapidez, lo que lo vuelve ideal para trabajar en tiempo real con CPUs convencionales.

3.2. Hardware utilizado

3.2.1. Cámara

El sensor seleccionado para la adquisición de datos visuales fue una cámara RGB-D Intel Realsense D415. La elección de este tipo de sensor se debe a que tanto RGB-D como cámaras estéreo representan mejores alternativas, debido a que con un

sistema de visión monocular se sufre de deriva de escala y puede fallar al realizar rotaciones puras.

En particular, este modelo de cámara, que se observa en la figura 3, cuenta con un modulo RGB de 2 megapixeles capaz de entregar resoluciones de hasta 1920x1080 a 30 fps y con un campo de visión de 69°x42°. El módulo de profundidad tiene un rango de funcionamiento de 0.16m-10m, resolución de 1280x720 con un campo de visión ligeramente mas reducido que el modulo RGB, de 65°x40°. Se recomienda trabajar en el rango de 0.5-3m en profundidad.

Figura 3: Cámara RGB-D Intel®RealSense™D415

Para poder utilizar ORB-SLAM2 con este sensor de adquisición de imagen es necesario conocer los parámetros intrínsecos del mismo, razón por la cual se llevó a cabo la calibración de esta cámara, utilizando el mensaje de ROS llamado camera_info. Dichos parámetros se muestran a continuación.

$$K = \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix}$$
$$K = \begin{bmatrix} 611{,}72 & 0 & 423{,}11 \\ 0 & 611{,}22 & 239{,}61 \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

En (1), la matriz K es conocida como la matriz de parámetros intrínsecos, dentro de la cual f_x y f_y son las distancias focales y c_x y c_y son las coordenadas del punto principal dentro del plano de imagen.

3.2.2. *Robot móvil*

El robot utilizado es un Pioneer 3-AT, robot móvil de tracción diferencial en una configuración de cuatro ruedas activas, que si bien este modelo es pensado para aplicaciones en exteriores, resulta también funcional para pruebas en un entorno estructurado como es el caso de este trabajo

Para el caso particular de los experimentos realizados en este artículo, se utilizó una versión restaurada del robot; de los componentes originales sólo se utilizaron los motores, encoders y el circuito de carga; el resto del control de bajo nivel (un control PID para las velocidades angulares de las ruedas) lo realizó un arduino nano. Sobre el robot se montó la cámara RGB-D y una computadora portátil que cuenta con un procesador *AMD Ryzen* 3 y 6gb de memoria *RAM*, misma que se encargó de procesar el controlador y enviar las velocidades deseadas al arduino. La construcción final del robot se aprecia en la figura 4.

Figura 4: Robot Pioneer 3-AT con cámara RGB-D y una computadora portátil para el control.

3.3. *Control para navegación autónoma*

Debido a que las dos ruedas de cada lado están actuadas por el mismo motor se considera el modelo cinemático como el mismo que el de un robot de manejo diferencial ideal (Mandow *et al.*, 2007).El diagrama del modelo se muestra en la figura 5. Donde X y Y son las coordenadas con respecto a un marco de referencia global, v es la velocidad lineal, ω es la velocidad angular y θ es el ángulo de orientación con respecto al eje X.

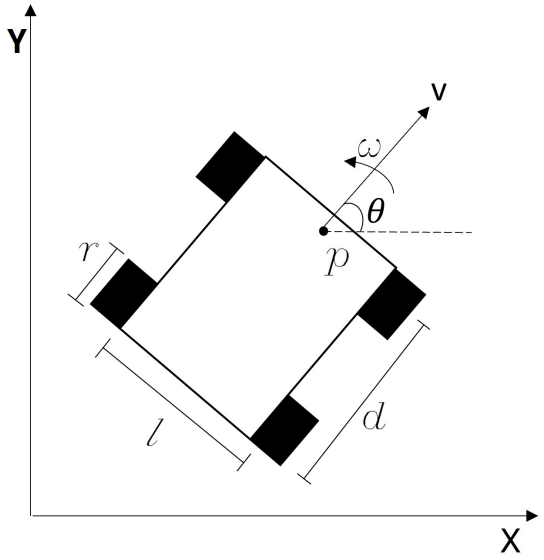

Figura 5: Diagrama de un robot móvil de manejo diferencial

Las ecuaciones que describen este modelo las observamos en (2). Siendo $\dot{x}$ la velocidad en x, $\dot{y}$ la velocidad en y y $\dot{\theta}$ la velocidad angular.

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\,\mathrm{sen}\,\theta \\ \dot{\theta} &= \omega \end{aligned} \quad (2)$$

Si adaptamos el modelo para considerar un punto **p** perpendicular a centro del eje de rotación, desplazado una distancia

d del mismo, como el punto cuya posición se controlará, pues esta será la posición de la cámara, nos queda lo que se ve en (3).

$$\begin{aligned} \dot{x} &= v\cos\theta - \omega d\,\text{sen}\,\theta \\ \dot{y} &= v\,\text{sen}\,\theta + \omega d\cos\theta \end{aligned} \tag{3}$$

El error se define como se muestra en (4).

$$\begin{aligned} e_x &= x - x^r \\ e_y &= y - y^r \end{aligned} \tag{4}$$

Donde e_x es el error en x, x^r es la coordenada x de la referencia, e_y es el error en y y y^r es la coordenada y de la referencia. Debido nuestro objetivo es hacer regulación a un punto, es decir, ir de un punto inicial a un punto deseado, solo se necesita un controlador cinemático donde solo es necesario controlar las velocidades del robot y no es requerido controlar las fuerzas, además se estarán saturando las velocidades, teniendo como valor máximo 10 rad/s, para proteger la integridad de los motores y desestimar las inercias, por lo tanto, la ley de control queda expresada de forma matricial, como se muestra en (5), k_1 es la ganancia de velocidad lineal y k_2 es la de velocidad angular, se trata de un control proporcional que involucra la cinemática del modelo.

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \cos\theta & -d\,\text{sen}\,\theta \\ \text{sen}\,\theta & d\cos\theta \end{bmatrix}^{-1} \begin{bmatrix} -k_1 & 0 \\ 0 & -k_2 \end{bmatrix} \begin{bmatrix} e_x \\ e_y \end{bmatrix} \tag{5}$$

La ley de control propuesta nos genera velocidades lineales y angulares deseadas, sin embargo no es posible asignarle esas velocidades de manera directa al robot, por el modelo cinemático sabemos que el movimiento de un robot de manejo diferencial viene dado por la diferencia de velocidades en las ruedas, por lo tanto sabemos que la velocidad lineal está dada por el promedio de la velocidad lineal de las ruedas, mientras que la velocidad angular está definida por la diferencia de velocidades lineales de las ruedas dividido entre l que es la distancia entre las ruedas, esto se expresa en (6).

$$\begin{aligned} V &= \frac{V_R + V_L}{2} \\ \omega &= \frac{V_R - V_L}{l} \end{aligned} \tag{6}$$

donde V_R y V_L son las velocidades lineales de las ruedas derecha e izquierda respectivamente, las cuales son dadas por las velocidades angulares de las ruedas (ω_R y ω_L) multiplicadas por el radio r de las mismas, como se muestra en (7).

$$\begin{aligned} V_R &= r\omega_R \\ V_L &= r\omega_L \end{aligned} \tag{7}$$

Si sustituimos (7) en (6) y despejamos las velocidades angulares de cada rueda, misma que serás las enviadas al robot para producir su movimiento, obtenemos (8).

$$\begin{aligned} \omega_L &= \frac{2v - \omega l}{2r} \\ \omega_R &= \frac{2v + \omega l}{2r} \end{aligned} \tag{8}$$

Este tipo de control es empleado en diversas

3.4. *Estrategia de evasión de obstáculos*

Los algoritmos BUG (Lumelsky y Stepanov, 1987) son un enfoque de planeación de trayectorias para robots móviles, utilizados ampliamente para la generación de trayectorias alrededor de obstáculos. Básicamente, se divide el espacio de trabajo en dos regiones, el área libre y el área de obstáculos. Si la trayectoria hacia el punto deseado no se ve interrumpida, se considera que fue óptima y se detiene el algoritmo; en caso contrario, si se encuentra dentro del área de obstáculos, se busca una forma de rodearlo, esto se describe en el algoritmo 1. En particular, se utilizó el algoritmo BUG0, el cual genera trayectorias como las de la figura 6.

Algoritmo 1: Algoritmo Bug

```
Ir al objetivo
si El camino está bloqueado entonces
    Seguir el contorno del obstáculo
    si El camino está libre entonces
        Ir al objetivo
    fin
fin
```

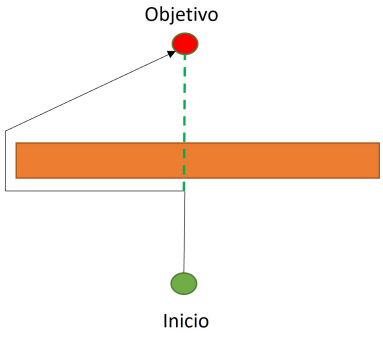

Figura 6: Funcionamiento del algoritmo Bug 0

3.5. *Integración con ROS*

ROS (Quigley *et al.*, 2009) es un conjunto de bibliotecas de código abierto y gratuito que proporciona una variedad de herramientas para ayudar en el desarrollo de software para robots. Representa una plataforma modular y distribuida que permite la comunicación entre componentes de software, lo que facilita la creación de sistemas complejos y flexibles para robots como el presentado en (Mishra y Javed, 2018).

En la figura 7 la computadora a bordo del robot recibe las imágenes obtenidas por la cámara RGB-D, mismas que son procesadas por el nodo de ORB-SLAM2 de ROS, el cual devuelve la posición estimada del sensor dentro del mapa. La imagen de profundidad, es a su vez transmitida al nodo *depthimage_to_laserscan* que permite conocer la distancia a la que se encuentran los objetos en frente transformando la imagen de profundidad en una señal de escaneo láser; con la anterior información, se ejecuta un código de Python que contiene la ley de control y el algoritmo BUG0. La salida de dicho script son velocidades recibidas por el arduino, que se encarga de que las ruedas del robot se muevan a la velocidad requerida.

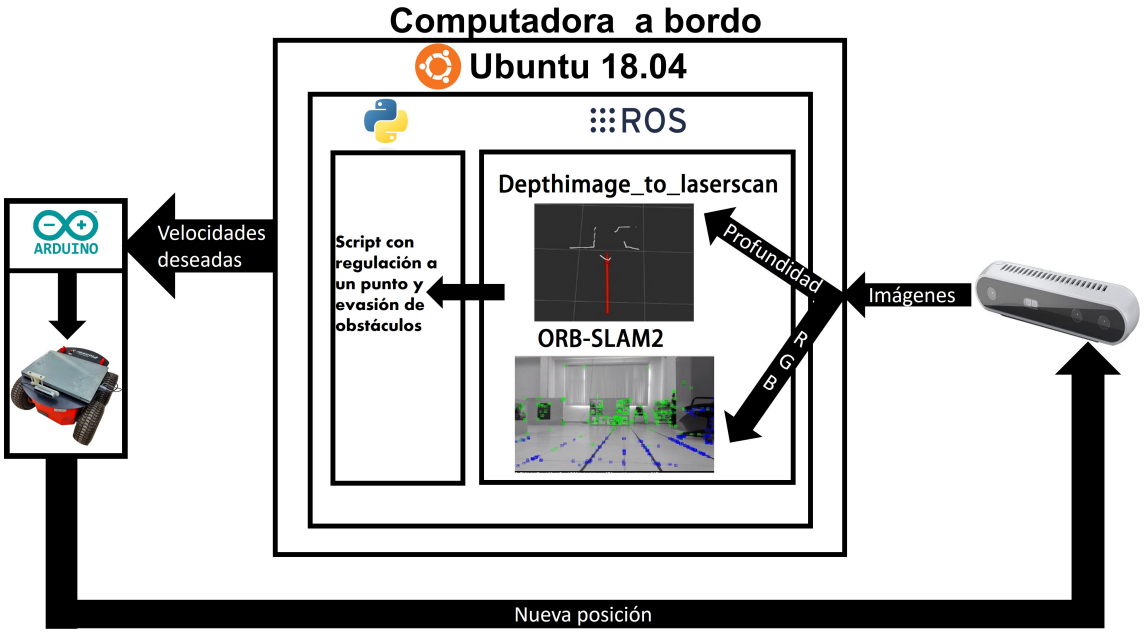

Figura 7: Diagrama de comunicación ROS-Python-Arduino

4. Resultados

4.1. Creación del mapa

Para la generación del mapa fue necesario controlar al robot de manera manual, buscando que se moviera a lo largo del entorno. El algoritmo ORB-SLAM2 nos devuelve el mapa en forma de una nube de puntos, que coloca en el plano 3D las características ORB detectadas en cada frame, luego de varias pruebas se identificaron ciertos factores que son necesarios para asegurar la correcta densidad de la nube de puntos, lo cual asegurará una correcta localización.

- Se deben realizar trayectorias que inicien y terminen en el mismo punto, para que la etapa de *Loop Closing* pueda hacer su trabajo de manera correcta.
- Es necesario no realizar rotaciones bruscas durante la generación del mapa, intentando mantener características coincidentes dentro del plano de imagen en todo momento.
- Es vital que existan características distintivas en el ambiente, de lo contrario no generará un mapa adecuado debido a la baja densidad de la nube de puntos.
- Las características visuales deben estar distribuidas a lo largo de todo el plano de imagen, es decir es recomendable evitar que las texturas o características importantes se ubiquen solo en áreas especificas del entorno.
- Después de cada cierre de ciclo se requiere que el robot se mantenga estático hasta que se termine de actualizar el mapa y se corrijan los errores de reproyección.
- Es importante saber que se deben realizar trayectorias en sentido horario y antihorario; de no ser así, la localización se perderá al trabajar en el sentido que no se consideró.

Tomando en cuenta los puntos anteriores, el mapa que se generó se muestra en la figura 8; el resultado es una nube de puntos que describe el ambiente. En la figura 9 se muestra una fotografía del ambiente a través del cual se obtuvo el mapa de la figura 8.

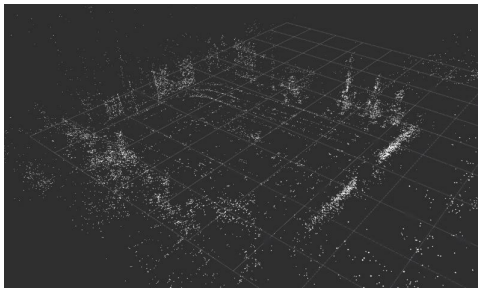

Figura 8: Mapa del entorno

Figura 9: Entorno experimental real

Es necesario comentar que el mapa tiene metros como unidades de sus coordenadas, además en base a este es que el nodo ORB-SLAM2 de ROS nos devuelve la localización de la cáma-

ra, de esta podemos obtener las variables x, y y θ.

4.2. *Regulación a un punto*

Utilizando el mapa generado previamente como marco de referencia, es posible proporcionar coordenadas objetivo al robot. Las pruebas que se realizaron contemplan una tolerancia de 0.15 metros para el error de posición expresado por la distancia euclidiana de la posición actual y deseada de cada iteración. Se trabajó con una ganancia de velocidad angular mayor a la ganancia de velocidad lineal debido a que, por su construcción, al robot no le es posible realizar trayectorias curvas. Para solucionar este problema se propuso la mencionada ganancia, buscando que se corrigiera, primero, el error de orientación (girando sobre su propio eje). Los valores de ganancia utilizados son $k_1 = 0{,}1$ y $k_2 = 2{,}5$, estos fueron sintonizados de manera experimental.Posteriormente, cuando la velocidad angular necesaria calculada por el control era mínimo, es decir se requería solo velocidad lineal, se incrementaba el valor de la ganancia lineal multiplicándola por 10 para evitar que el movimiento en linea recta fuera excesivamente lento.

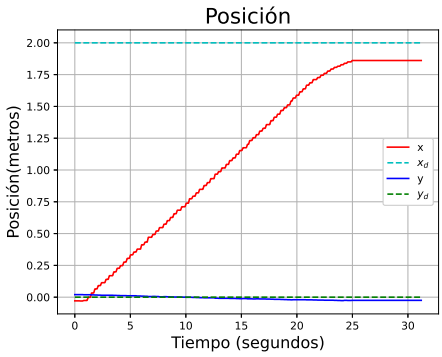

Figura 10: Posición del robot

Figura 11: Gráfica de error

Como se puede observar en la figura 10, las condiciones iniciales fueron $(0, 0)$ y la posición deseada fue $(2, 0)$ lo que representó un movimiento en línea recta sobre el eje x del marco de referencia del mapa. Se observa, además, como se detuvo una vez que se encontró dentro de la tolerancia, lo anterior es también visible en la figura 11, donde se muestra el error de posición.

4.3. *Evasión de obstáculos*

Para la realización de esta prueba se consideró que los obstáculos deberían ser al menos de la misma altura a la que se encontraba la cámara en el robot, debido a que no es posible ver objetos de menor altura, lo cual resultaría en colisión. Además, se utilizaron las mismas ganancias que en la prueba anterior y se eligió que la dirección de giro al encontrar un obstáculo fuera en sentido antihorario para, posteriormente, ejecutar el algoritmo Bug0.

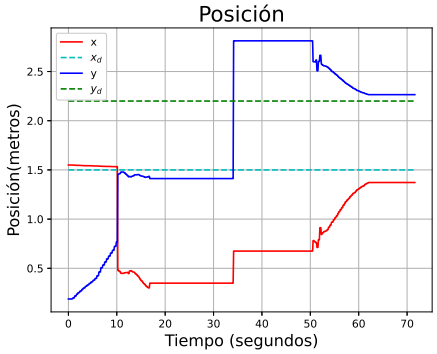

Figura 12: Posición del robot con evasión de obstáculos

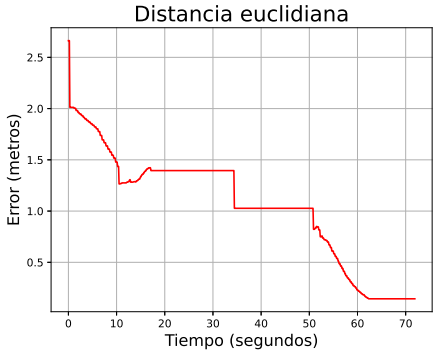

Figura 13: Error de posición con evasión de obstáculos

Si prestamos atención a la figura 12, la posición inicial fue $(1{,}5, 0{,}2)$, mientras que la posición deseada fue $(1{,}5, 2{,}2)$. Esto indica que también podría tratarse de una trayectoria en línea recta, en este caso sobre el eje y, sin embargo, la trayectoria directa resultó estar bloqueada, razón por la cual se accionó la etapa de seguimiento del contorno entre la iteración 10 y 50 del ciclo de control, como indica la figura 13. El algoritmo BUG0 funcionó adecuadamente, lo que ultimó en una trayectoria libre de colisiones.

5. Conclusiones

En lo que respecta a la navegación con ORB-SLAM2 es necesario puntualizar que representa una poderosa herramienta para la estimación de la pose de un robot dentro de un ambiente, siempre y cuando se tome en cuenta lo descrito en la sección 4.1 para la construcción del mapa, ya que de lo contrario, la estimación de la pose puede verse ampliamente afectada. Es importante también tener en cuenta que rotaciones con altas velocidades

ocasionarán pérdida de localización, por lo que se recomienda operar con velocidades dentro de un rango preestablecido.

Por su parte a la evasión de obstáculos le queda mucho por mejorar. Se propone como trabajo futuro explorar otros algoritmos de planeación de trayectorias. Además, inconvenientes como el punto ciego por debajo de la altura a la que está la cámara pueden ser solucionados con la incorporación de sensores extras.

El trabajo presentado ofrece una alternativa utilizando un solo sensor exteroceptivo, encontrando posibles aplicaciones en tareas con varios objetivos, que requieran realizar desplazamientos con autonomía, a lo largo de ambientes de interiores estructurados.

Agradecimientos

El primer autor de este artículo agradece a CONACYT por el apoyo brindado dentro de su programa nacional de becas, así mismo a CINVESTAV unidad Saltillo por la formación brindada al ser parte del grupo de Robótica y Manufactura avanzada.

Referencias

Lumelsky, V. J. y Stepanov, A. A. (1987). Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2(1-4):403–430.

Macario Barros, A., Michel, M., Moline, Y., Corre, G., y Carrel, F. (2022). A comprehensive survey of visual slam algorithms. *Robotics*, 11(1):24.

Mandow, A., Martinez, J. L., Morales, J., Blanco, J. L., Garcia-Cerezo, A., y Gonzalez, J. (2007). Experimental kinematics for wheeled skid-steer mobile robots. En *2007 IEEE/RSJ international conference on intelligent robots and systems*, pp. 1222–1227. IEEE.

Mingachev, E., Lavrenov, R., Tsoy, T., Matsuno, F., Svinin, M., Suthakorn, J., y Magid, E. (2020). Comparison of ros-based monocular visual slam methods: Dso, ldso, orb-slam2 and dynaslam. En *Interactive Collaborative Robotics: 5th International Conference, ICR 2020, St Petersburg, Russia, October 7-9, 2020, Proceedings 5*, pp. 222–233. Springer.

Mishra, R. y Javed, A. (2018). Ros based service robot platform. En *2018 4th International Conference on Control, Automation and Robotics (ICCAR)*, pp. 55–59. IEEE.

Mur-Artal, R. y Tardós, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262.

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., y Ng, A. (2009). Ros: an open-source robot operating system. En *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., y Chen, M. (2022). Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*.

Rublee, E., Rabaud, V., Konolige, K., y Bradski, G. (2011). Orb: An efficient alternative to sift or surf. En *2011 International conference on computer vision*, pp. 2564–2571. Ieee.

Zghair, N. A. K. y Al-Araji, A. S. (2021). A one decade survey of autonomous mobile robot systems. *International Journal of Electrical and Computer Engineering*, 11(6):4891.