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. 10 No. Especial 5 (2022) 140–145

Desarrollo de un sistema de navegación autónoma para un robot móvil basado en una cámara RGB-D
Development of an autonomous navigation system for a mobile robot based on an RGB-D camera

S.D. Vázquez-Muñoz [a,*], J.M. Ibarra-Zannatha [a], O. González-Miranda [a]

[a] *Departamento de Control Automático, CINVESTAV-IPN, 07360, CDMX, México.*

Resumen

En este trabajo se presenta el desarrollo de un sistema de navegación autónoma para un robot móvil basado en una cámara RGB-D, que le permita navegar en un entorno estructurado, pero desconocido, formado por pasillos y las salas que tengan sus entradas accesibles desde dichos pasillos. Entre las diferentes aplicaciones que podría tener un robot móvil con este sistema de navegación están: el mapeo, la desinfección en interiores (SARS-CoV-2) con luz ultravioleta, la inspección autónoma de estos recintos y otras muchas que requieran visitar de manera autónoma las salas a las que se tenga acceso desde un pasillo o conjunto de pasillos interconectados. Se implementó dicho sistema tanto en simulación como en tiempo real, obteniendo para ambos casos los resultados esperados al navegar por un recinto.

Palabras Clave: Robot móvil, Navegación autónoma, Control visual, Cámara RGB-D, Imágenes de profundidad.

Abstract

This paper presents the development of an autonomous navigation system for a mobile robot based on an RGB-D camera, which allows it to navigate in a structured but unknown environment, formed by corridors and the rooms that have their entrances accessible from said corridors. hallways. Among the different applications that a mobile robot with this navigation system could have are: mapping, indoor disinfection (SARS-CoV-2) with ultraviolet light, autonomous inspection of these enclosures and many others that require autonomous visits. rooms that are accessed from a corridor or set of interconnected corridors. Said system was implemented both in simulation and in real time, obtaining for both cases the expected results when navigating through an enclosure.

Keywords: Mobile robot, Autonomous navigation, Visual control, RGB-D camera, Depth images.

1. Introducción

Durante y después de estos años de la Pandemia causada por el virus SARS-CoV-2 se ha incrementado la necesidad de prevenir y limitar contagios mediante la sanitización y desinfección de recintos cerrados, tales como escuelas, comercios, oficinas y muy especialmente hospitales puesto que en estos se concentra la mayor cantidad de virus, bacterias y todo tipo de gérmenes, no sólo el causante de la COVID-19. En los procesos de limpieza y desinfección se prefiere que haya la menor cantidad posible de intervención humana, lo cual genera un gran nicho de oportunidad para la Robótica y los robots de servicio capaces de realizar estas tareas de manera rápida, eficiente y, sobre todo, de manera autónoma y sin exponer a seres humanos al contacto con los gérmenes. Así, en la literatura especializada se encuentran diversas soluciones robóticas al problema de sanitización automática.

1.1. Antecedentes

En (Nosirov *et al.*, 2020) se presenta el diseño un robot móvil capaz de desinfectar lugares en entornos hospitalarios de forma autónoma mediante una luz ultravioleta de la banda C (UV-C) con longitudes de onda desde 200 nm hasta 280 nm, evitando que los seres humanos entren en contacto con los gérmenes. Dicho robot se utiliza, además, para dar servicios a

*Autor para correspondencia: daniela.vazquez@cinvestav.mx
Correo electrónico: daniela.vazquez@cinvestav.mx (Sdeiby Daniela Vázquez-Muñoz), jibarra@cinvestav.mx (Juan Manuel Ibarra-Zannatha), ogonzalez@ctrl.cinvestav.mx (Oscar González-Miranda).

Historial del manuscrito: recibido el 28/10/2022, última versión-revisada recibida el 28/10/2022, aceptado el 08/11/2022, publicado el 11/11/2022. **DOI:** https://doi.org/10.29057/icbi.v10iEspecial5.10142

los pacientes internados. Cabe destacar que la luz ultravioleta UV-C producida artificialmente se ha utilizado con éxito como germicida y bactericida durante décadas, por lo que actualmente suele denominarse como UV-GI, abreviación internacional de *Irradiación germicida ultravioleta*. En (Vidyashankar y Srinivasa, 2021) se presenta un robot autónomo capaz de evitar obstáculos, equipado con rociadores para saneamiento, diseñado para la desinfección de pasillos utilizando peróxido de hidrógeno; su movimiento consiste en avanzar hacia el frente y, con la ayuda de tres sensores infrarrojos (uno al frente y uno en cada lado), detectar obstáculos y evadirlos; sus limitaciones de movilidad y autonomía lo hace funcional únicamente en pasillos largos.

En (Bista *et al.*, 2016) se aborda el problema de navegación de un robot de sanitización realizando un mapeo preliminar del lugar por medio de teleoperación para posteriormente implementar distintas técnicas de control que le otorgan algún grado de autonomía. Por su parte, en (Ran *et al.*, 2021) se aborda el problema de navegación en entornos cerrados desconocidos basada en visión empleando una Red Neuronal Convolucional (CNN) y con datos recopilados fuera de línea mediante movimientos teleoperados, para posteriormente poder navegar en este entorno usando técnicas de control estándar. Otra alternativa para navegar por pasillos en un entorno cerrado se presenta en (Purwanto *et al.*, 2017) donde se estudia el control de un robot móvil utilizando la información de una imagen de profundidad y, convirtiéndola en información de múltiples puntos, calcula la presencia de obstáculos y la orientación del robot empleando esta información en un algoritmo de navegación que mantiene al robot caminando por los pasillos y evitando los obstáculos.

Todas estas técnicas y metodologías no sólo se han implementado en robots móviles, también se encuentran en robots humanoides como es el caso en (Faragasso *et al.*, 2013). En dicho trabajo se desarrolla un controlador visual simple en el que no se cuenta con un mapa y que presenta una forma de navegar por un laberinto formado por una serie de pasillos, el robot debe caminar lo más cercano posible al centro del pasillo para maximizar la seguridad del movimiento.

1.2. Problema a resolver

En este trabajo se desarrolló un sistema de navegación autónoma para un robot móvil que evoluciona en entornos interiores desconocidos, pero estructurados, los cuales están formados por pasillos y cuartos cuyas puertas deben estar abiertas y dar a dichos pasillos. La tarea de este robot consiste en navegar a lo largo de la mediana de los pasillos, detectar las puertas abiertas e ingresar a las habitaciones a fin de realizar tareas específicas, como por ejemplo, irradiar dicha habitación durante unos segundos con luz UV-C con el objetivo de sanitizarla, luego dará media vuelta, saldrá de la habitación y continuará su recorrido por los pasillos del recinto. Al mismo tiempo que navega por el recinto tiene la tarea de generar un mapa del lugar.

1.3. Solución propuesta

Para realizar la tarea descrita en la sección previa es necesario que el robot sea capaz de navegar automáticamente a lo largo de la mediana de los pasillos, detectar y localizar las puertas, generar la trayectoria que permita ingresar por dichas puertas y, eventualmente seguir los pasillos. Se propone utilizar una cámara RGB-D (Kinect) como sensor para determinar las trayectorias a seguir (mediana, ingreso y salida a las habitaciones, y girar en los cambios de dirección de los pasillos). En la siguiente lista se presenta el algoritmo desarrollado para realizar esta aplicación, mismo que es mostrado en la figura 1, el cual requiere inicialmente ubicar el robot en cualquier punto del pasillo, asegurando que el eje x_R de su referencial quede paralelo a la pared de dicho pasillo.

1. Ubicar el robot al inicio del pasillo.
2. El robot debe posicionarse al centro del pasillo manteniendo su eje x_R paralelo a la pared.
3. Avanzar en línea recta hasta detectar una puerta o pasillo.
4. Realizar las mediciones y cálculos que determinen el ancho y ubicación de la puerta o pasillo.
5. Si se determina que es una puerta, entonces se realiza el paso 6, sino realiza el paso 8.
6. El robot entra por la puerta y realiza un giro sobre su eje para hacer el mapa de la habitación y sanitizarla hasta reincorporarse con su orientación hacia el pasillo.
7. Regresa al centro del pasillo, se posiciona con su eje x_R paralelo a la pared y regresar al paso 3.
8. Girar para incorporarse al pasillo detectado y regresar al paso 3.

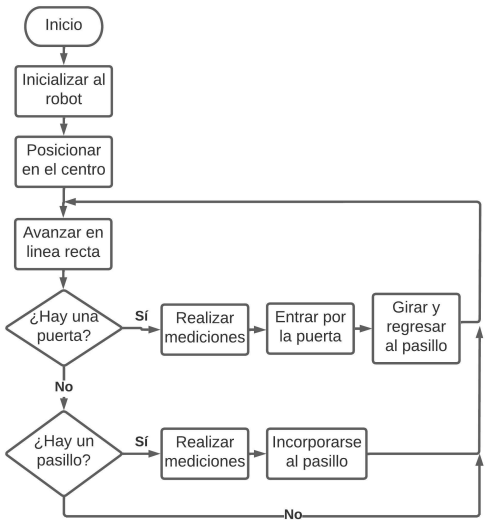

Figura 1: Las figuras a doble columna deben respetar márgenes y estilo.

1.4. Organización de este artículo

Después de esta Sección introductoria, se presenta en la Sección 2 los materiales utilizados en el desarrollo del proyecto. La Sección 3 presenta con algún nivel de detalle la implementación del sistema de navegación desarrollado, mientras que en la Sección 4 se presentan los resultados obtenidos. Finalmente, la Sección 5 presenta las conclusiones.

2. Materiales utilizados

2.1. Robot móvil

El robot móvil utilizado en este proyecto es el TurtleBot 2 ilustrado en la figura 2. Se trata de un robot tipo diferencial for-

mado por una base con un par de llantas motorizadas sobre un mismo eje y un par de llantas pequeñas, una al frente y la otra atrás para dar soporte a la base. Cuenta con un Kinect (cámara RGB y un sensor de profundidad de resolución 640x480), empleado como sensor de visión, cuyo campo de visión vertical es de 43,5° y su campo de visión horizontal es de 57°, este sensor captura la profundidad en un rango de 0.8 a 4 metros (Silliman, 2022).

Figura 2: TurtleBot 2.

2.2. *Software*

Robot Operating System (ROS) (Pyo *et al.*, 2017) es un SDK libre (*open source software development kit*) para el desarrollo de aplicaciones robóticas de todo tipo que se monta sobre Ubuntu. ROS ofrece a los desarrolladores, una plataforma estándar que los apoya desde la investigación, el prototipado y el desarrollo hasta la producción de sus aplicaciones. ROS es, además, una biblioteca de herramientas de software que van desde simples *drivers* hasta complejos algoritmos en el estado del arte que permite desarrollar aplicaciones específicamente utilizadas en robots, en otras palabras, gestiona la comunicación entre los distintos componentes de un robot. ROS permite el trabajo distribuido con distintas plataformas de hardware y con diferentes lenguajes de programación como Python y C++. Las aplicaciones de ROS se construyen como una red de programas ejecutables llamados *nodos*, que interactúan entre sí mediante *mensajes* publicados como *tópicos*. ROS se complementa con el uso de Gazebo (Gazebo, 2022), para probar el algoritmo propuesto antes de utilizar el robot real, el cual es un simulador 3D de robots, que permite la interacción dinámica entre objetos, robots y sensores en distintos entornos.

3. Desarrollo del Sistema de Navegación Autónoma

La implementación del sistema de navegación desarrollado contempla el siguiente conjunto de movimientos básicos:

i) Posicionarse al centro del pasillo, *ii*) Avanzar en línea recta, *iii*) Buscar puertas (o cambio de dirección del pasillo), *iv*) Calcular la trayectoria para ingresar por la puerta detectada y volver al pasillo, *v*) Ejecutar estas trayectorias, *vi*) Diseñar el controlador, así como el autómata de estados finitos que comanda la secuencia de movimientos a realizar. Así como generar un mapa de los lugares por los cuales navega el robot, haciendo uso del algoritmo gmapping de ROS basado en SLAM.

3.1. *Posicionarse al centro del pasillo*

El robot inicia su tarea en algún lugar desconocido al inicio de un pasillo de dimensiones también desconocidas, pero se sabe que el eje x_R del referencial asociado al robot es paralelo a las paredes del pasillo (Yaw=ψ_0). Este proceso comienza haciendo la medición de profundidad, utilizando para ello la cámara RGB-D, en los extremos horizontales de la imagen de profundidad, obteniendo las distancias l_i y l_d hacia las paredes de la izquierda y de la derecha respectivamente a un ángulo ϕ igual a la mitad del ángulo de apertura horizontal de la cámara, es decir $\phi = 28{,}5°$, tal como se muestra en la figura 3. Así, las distancias x_i y x_d se calculan como:

$$\begin{aligned} x_i &= l_i \sin\phi \\ x_d &= l_d \sin\phi \end{aligned} \tag{1}$$

Entonces, el ancho del pasillo se calcula como $A = x_i + x_d$, mientras que la mediana del pasillo está a $A/2$ metros de cualquiera de las dos paredes. Ahora, el robot debe girar 90° hacia la pared más alejada y avanzar d metros de acuerdo con los dos casos descritos a continuación:

$$Si: x_i > x_d \Longrightarrow d = \left(\frac{A}{2}\right) - x_d \tag{2}$$

$$Si: x_i < x_d \Longrightarrow d = \left(\frac{A}{2}\right) - x_i \tag{3}$$

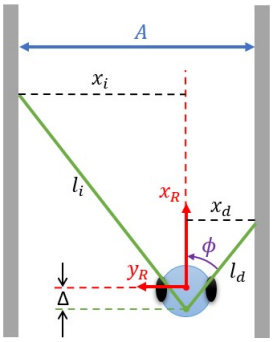

Figura 3: Pasillo.

Una vez ubicado sobre la mediana del pasillo el robot debe girar 90° para recuperar su orientación original (ψ_0). Esta acción correspondiente al centrado del robot se concentra en la figura 4.

0. *Inicio*
1. *Guardar orientación* $\psi_0 \leftarrow \psi_R$
2. *Si* $x_d < x_i$, *hacer* $\theta \leftarrow +90°$ *y calcular d (eq 1.a). Ir a 6*
3. *De lo contrario continuar*
4. *Si* $x_d > x_i$, *hacer* $\theta \leftarrow -90°$ *y calcular d (eq 1.b). Ir a 6*
5. *De lo contrario ir a 9*
6. *Girar* θ *grados*
7. *Avanzar d metros*
8. *Girar* $-\theta$ *grados*
9. *Fin*

Figura 4: Algoritmo para el centrado del robot.

3.2. *Avanzar buscando puertas*

Una vez que el robot se encuentra sobre la mediana del pasillo debe avanzar siempre por el centro, manteniendo su orientación paralela a las paredes mediante su control lateral y buscando alguna puerta abierta tanto a su derecha como a su izquierda. Ahora la medición de profundidad en el extremo izquierdo de

la imagen de profundidad es la misma que se tiene a la derecha y es:

$$P = \frac{A}{2\sin\phi} \tag{4}$$

Entonces, si esta medición aumenta en alguno de los dos lados significa que se ha encontrado el inicio de una puerta; se registra la posición del robot dentro del pasillo $x_0 = x_R$ (se tiene disponible la odometría) y, cuando el valor de profundidad recupera el valor P significa que se ha encontrado el final de la puerta, además, el avance que el robot tuvo entre estos dos eventos es igual al ancho de la puerta $A_p = x_R - x_0$. En ese momento se puede calcular la trayectoria a seguir para entrar por la puerta detectada, formada por un avance de l_x metros, un giro de 90° hacia la puerta detectada y un avance de l_y metros para ingresar a metros al interior de la habitación de acuerdo con la figura 5 y con las siguientes ecuaciones:

$$\begin{aligned} l_x &= P\cos\phi - \frac{A_p}{2} - \Delta \\ l_y &= \frac{A}{2} + a \end{aligned} \tag{5}$$

El centro del Kinect se encuentra Δ centímetros detrás del centro del robot. Una vez dentro de la habitación el robot realiza alguna tarea especifica como, por ejemplo, activar su lámpara UV-C durante un cierto tiempo t a fin de sanitizarla, a continuación debe salir girando 180°, avanzando l_y metros y girando 90° en el sentido opuesto al que utilizó para ingresar en la habitación. Una vez ubicado nuevamente sobre la mediana del pasillo continúa avanzando.

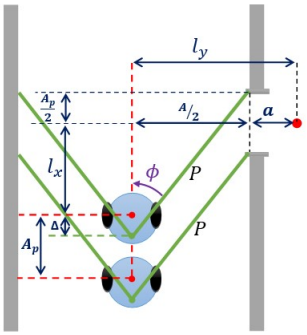

Figura 5: Mediciones para encontrar una puerta.

3.3. *Avanzar buscando cambios en la dirección del pasillo*

Esta rutina es exactamente la misma utilizada para buscar puertas, con la salvedad de que, si A_p es mayor a un cierto umbral, th, el cual permite discriminar entre puerta (típicamente menor o igual a un metro) y pasillo, entonces se tiene un pasillo. Si tenemos que $A_p > th$ entonces la trayectoria a seguir consiste tan solo en avanzar l_x metros, hacer un giro de 90° hacia el pasillo detectado y continuar avanzando. Esta situación se ilustra en la figura 6. La figura 7 muestra completo el algoritmo que permite al robot encontrar de manera autónoma tanto las puertas a la izquierda o a la derecha, como los cambios de dirección en el pasillo; cabe recordar que, antes de iniciar este algoritmo, los parámetros $P, \Delta, \phi, A, a, t, th, \psi_0$ tienen valores conocidos.

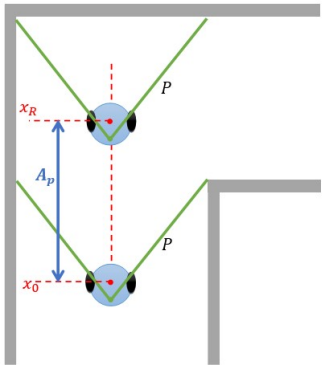

Figura 6: Mediciones para encontrar un Pasillo en L.

3.4. *Control de movimiento del robot*

Los algoritmos resumidos en las figuras 4 y 7 hacen referencia explícita a las siguientes tareas básicas de control: *control longitudinal* (Avanzar y Avanzar $l_{x,y}$ metros) y *control de giro* (Girar θ grados), además de mantener la dirección paralela a las paredes cuando avanza, misión a la que se denomina *control lateral*, las cuales se implementaron de la manera que se explica a continuación. Cabe mencionar que esto se hace mediante dos señales: *i*) velocidad de avance V_R a lo largo del eje x_R del referencial asociado al robot, y *ii*) velocidad angular ω_R alrededor del eje z_R de dicho referencial del robot.

0. *Inicio*
1. *Avanzar*
2. *Si* $l_d = l_i < P$ *ir a 25*
3. *De lo contrario continuar*
4. *Si* $l_d > P$ *hacer hacer* $\theta \leftarrow -90°$. *Ir a 8*
5. *De lo contrario continuar*
6. *Si* $l_i > P$ *hacer hacer* $\theta \leftarrow +90°$. *Ir a 8*
7. *De lo contrario ir a 1*
8. $x_0 \leftarrow x_R$
9. *Avanzar*
10. *Si* $l_d = P$ *ir a 14*
11. *De lo contrario continuar*
12. *Si* $l_i = P$ *ir a 14*
13. *De lo contrario ir a 9*
14. $A_p \leftarrow x_R - x_0$
15. *Avanzar* l_x *(eq 5)*
16. *Girar* θ *grados*
17. *Si* $A_p > th$ *hacer* $\psi_0 \leftarrow \psi_R$. *Ir a 1*
18. *De lo contrario continuar*
19. *Avanzar* l_y *(eq 5)*
20. *Esperar t segundos*
21. *Girar 180°*
22. *Avanzar* l_y *(eq 5)*
23. *Girar* $-\theta$ *grados*
24. *Ir a 1*
25. *Fin*

Figura 7: Algoritmo para buscar puertas y/o cambios en la dirección del pasillo.

3.4.1. *Control Longitudinal*

La tarea *Avanzar* consiste en enviar al robot un valor constante denominado *velocidad de crucero* denotada V_c, en el entendido de que durante su movimiento se están supervisando los sensores del robot (en este caso el Kinect y la IMU) en espera de la señal de terminación y/o de la señal para registrar los valores de la odometría. Pero, cuando la tarea de movimiento es *Avanzar* $l_{x,y}$ *metros* primeramente se registra el valor actual de la posición $x_0 \longleftarrow x_R$, se calcula la consigna de posición x_d sumándole la distancia deseada $l_{x,y}$ y se aplica un control

proporcional con ganancia k_x para calcular el comando de velocidad para el robot:

$$\begin{aligned} x_0 &\longleftarrow x_R \\ x_d &= x_0 + l_{x,y} \\ e_x &= x_d - x_R \\ V_R &= k_x e_x \end{aligned} \tag{6}$$

3.4.2. *Control Lateral*

Es importante señalar que, cada vez que el robot avanza a lo largo de un pasillo, debe mantener una dirección paralela a las paredes de dicho pasillo, es decir mantener su orientación igual a la orientación de la pared, es decir $\psi_R = \psi_0$. Para ello debe implementarse un control lateral, que trabaja simultáneamente con el *control longitudinal* para corregir errores de orientación mediante un controlador proporcional con ganancia k_θ:

$$\begin{aligned} e_\theta &= \psi_0 - \psi_R \\ \omega_R &= k_\theta e_\theta \end{aligned} \tag{7}$$

3.4.3. *Control de Giro*

Como ya se habrá notado, el esquema de control de navegación autónoma adoptado es del tipo conocido como *dead reackoning*, en donde las trayectorias se definen como avance en línea recta del punto actual al punto siguiente del camino haciendo un alto para realizar el giro que permita que la orientación del robot sea la adecuada para alcanzar al siguiente punto. El control asociado al comando *Girar θ grados* también se implementó como un control proporcional que comanda la velocidad angular ω_R con una ganancia k_ω. La consigna de este controlador se calcula como la suma de la orientación actual del robot ψ_R más los θ grados que se desea girar, de la siguiente manera:

$$\begin{aligned} \psi_0 &\longleftarrow \psi_R \\ \psi_d &= \psi_0 + 90 \\ e_\psi &= \psi_d - \psi_R \\ \omega_R &= k_\psi e_\psi \end{aligned} \tag{8}$$

4. Resultados

El sistema de navegación autónoma desarrollado se probó primero en simulación y luego directamente en el robot obteniendo los resultados que se comentan a continuación.

4.1. *Simulaciones*

Este proyecto se desarrolló bajo ROS (Kinetic) en Ubuntu (Versión 16.04 mate) y, para las pruebas en simulación se utilizó, además, Gazebo. Primeramente, se creó un mundo virtual en Gazebo formado por pasillos y habitaciones cuyas puertas dan a esos pasillos como se muestra en la figura 8. Luego, se instaló el modelo del TurtleBot 2 equipado con una cámara RGB-D tipo Kinect, el cual incluye tanto la parte visual (modelo geométrico) como la parte física (modelo matemático).

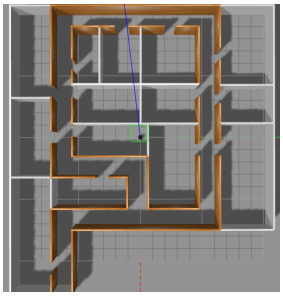

Figura 8: Mundo virtual en Gazebo.

Se comienza la simulación colocando al robot al inicio del pasillo más cerca de una pared que de la otra, pero con x_R paralelo a las paredes del pasillo. Se eligen los parámetros siguientes: $k_x = 0{,}4$, $k_\psi = 1{,}2$, $k_\theta = 4$, $a = 1$ y $th = 1{,}2$ metros. En la figura 9 se muestra una de las imágenes de profundidad que produce el Kinect y en la cual se hace la medición de la distancia entre esta cámara y las paredes del pasillo. Para ello se leen los valores extremos de la línea horizontal central de esa imagen.

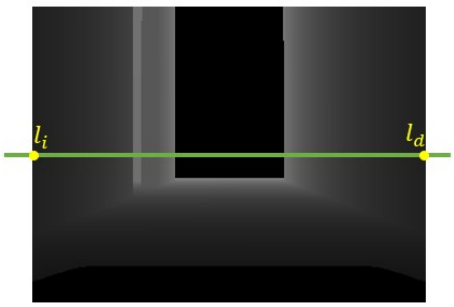

Figura 9: Imagen de profundidad en simulación.

Se activan sucesivamente los procedimientos *Centrado* y *Búsqueda de puertas y pasillos*. Para evaluar el comportamiento del *control lateral* en la figura 11 se muestra el comportamiento del error de orientación a lo largo de un desplazamiento en línea recta que se obtiene con el control lateral dado por la ecuación 7, notando que permanece bastante estable en un valor muy cercano a cero, con algunas milésimas de radian.

Para evaluar el desempeño del *control longitudinal* en la figura 12 se muestra el error de posición que se presenta cuando se desea llegar al centro de una puerta para, posteriormente, ingresar a la habitación correspondiente. En dicha figura se ve la posición deseada en rojo, la trayectoria del robot en azul y, el error de posicionado aparece en color verde y se nota que prácticamente es anulado por completo. La velocidad de avance del robot es de $0{,}4 m/s$. Con el objetivo de evaluar el comportamiento del sistema de navegación desarrollado se presenta un vídeo de la navegación realizada de manera autónoma en el entorno ya mencionado, el cual se puede ver en: `https://youtu.be/dNkqD-1tW90`.

4.2. *Control en tiempo real*

Una vez que los resultados obtenidos en simulación fueron satisfactorios, se procedió a la implementación del sistema de

navegación autónomo en el robot real. Primeramente, en la figura 10 se muestra una de las imágenes de profundidad realmente obtenidas por el Kinect, en este caso se trata de un pasillo largo. Al igual que en el sistema simulado, en este caso también se grafican los resultados del control lateral en la figura 13 y del control longitudinal en la figura 14. El primer controlador mantiene la orientación con un error entre -6 y $4\text{x}10^{-3}$ radianes o sea de menos de 0.3°. Mientras que, el control longitudinal tiene un error de algunos centímetros. Puede verse un vídeo de la navegación en: `https://youtu.be/j0frGJMixoQ`.

Figura 10: Imagen de profundidad tomada por el robot real.

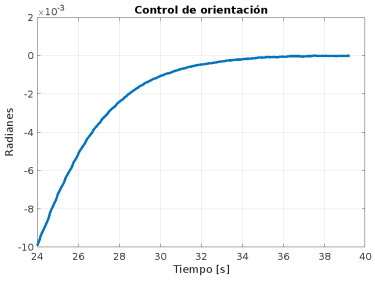

Figura 11: Gráfica del error de orientación en simulación.

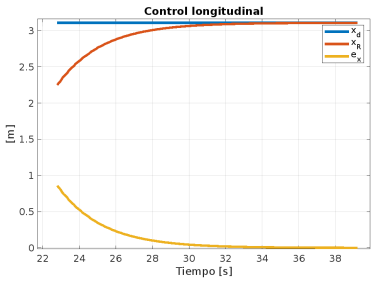

Figura 12: Control de posición en simulación.

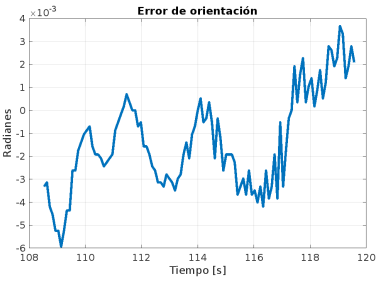

Figura 13: Error de orientación en el robot real.

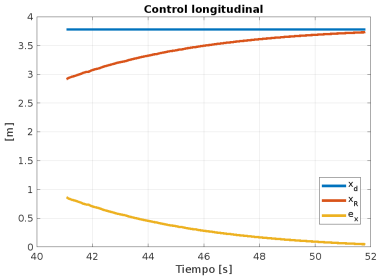

Figura 14: Control de posición en el robot real.

5. Conclusiones

El sistema de navegación autónoma desarrollado para un robot móvil resultó adecuado para recorrer los pasillos de un entorno arquitectónico clásico (escuela, oficinas, hospital, etc.) entrando a las habitaciones para, por ejemplo, hacer el mapa del entorno, sanitizar las habitaciones con luz UV-C o para llevar mensajería. Los controles diseñados para comandar el movimiento del robot fueron controladores proporcionales debido a que internamente este ya cuenta con servomecanismos en sus dos ruedas, además de un control cinemático que tan sólo requiere consignas en velocidad lineal y angular.

Cabe mencionar que, entre las tareas realizadas en simulación está la de generar el mapa del entorno, para lo cual se utilizó el sistema de vídeo SLAM basado en imágenes de profundidad utilizando para ello el algoritmo *gmapping*, librería libre de OpenSLAM. La idea de utilizar técnicas de Vídeo SLAM es poder resolver la navegación en grandes sistemas de pasillos con bifurcaciones en **T**, lazos o puertas que aparecen una frente a la otra. Un área de oportunidad complementaria sería la implementación de controles laterales basados en visión y no sólo en la odometría como en el proyecto aquí reportado.

Finalmente, se trabaja en la eliminación de la restricción inicial de que el eje x_R del robot sea paralelo a las paredes del pasillo, a fin de facilitar la realización de la tarea. Además, se está iniciando la aplicación de sistemas conexionistas para intentar *aprender* del control actualmente implementado a fin de evitar las mediciones que lo caracterizan.

Referencias

Bista, S. R., Giordano, P. R., y Chaumette, F. (2016). Appearance-based indoor navigation by ibvs using line segments. *IEEE Robotics and Automation Letters*, 1:423–430.

Faragasso, A., Oriolo, G., Paolillo, A., y Vendittelli, M. (2013). Vision-based corridor navigation for humanoid robots. En *2013 IEEE International Conference on Robotics and Automation*, pp. 3190–3195.

Gazebo (2022). https://gazebosim.org/home.

Nosirov, K., Begmatov, S., Arabboev, M., y Medetova, K. (2020). Design of a model for disinfection robot system. *2020 International Conference on Information Science and Communications Technologies (ICISCT)*, pp. 1–4.

Purwanto, D., Rivai, M., y Soebhakti, H. (2017). Vision-based multi-point sensing for corridor navigation of autonomous indoor vehicle. En *2017 International Conference on Electrical Engineering and Computer Science (ICECOS)*, pp. 67–70.

Pyo, Y., Cho, H., Jung, L., y Lim, D. (2017). *ROS Robot Programming (English)*. ROBOTIS.

Ran, T., Yuan, L., y Zhang, J. (2021). cene perception based visual navigation of mobile robot in indoor environment. *ISA Transactions*, pp. 389–400.

Silliman, M. (2022). Getting started"guide for developers interested in robotics `https://learn.turtlebot.com/`.

Vidyashankar, S. y Srinivasa, G. (2021). Low-cost robot for autonomous disinfection of corridors. *2021 Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pp. 1737–1742.