Revista Iberoamericana de Automática e Informática Industrial

Revista Iberoamericana de Automática e Informática Industrial 22 (2025) 184-195

www.revista-riai.org

Navegación autónoma en línea: Integración de PRM y SLAM para la exploración de entornos desconocidos

Karla-I. Salado-Chavez, Oscar-D. Ramírez-Cárdenas*, José-A. Arias-Aguilar

División de Estudios de Posgrado, Universidad Tecnológica de la Mixteca, Huajuapan de León, Oaxaca, México.

To cite this article: Salado-Chavez, K., Ramírez-Cárdenas, O., Arias-Aguilar, J. A. 2025A Online Autonomous Navigation: Integration of PRM and SLAM for Exploration of Unknown Environments. Revista Iberoamericana de Automática e Informática Industrial 22, 184-195. https://doi.org/10.4995/riai.2025.21961

Resumen

Este trabajo presenta un enfoque para la navegación autónoma de robots móviles en entornos interiores desconocidos, integrando los algoritmos SLAM y PRM. El método propuesto permite tanto alcanzar un objetivo específico como explorar áreas no mapeadas, gestionando situaciones en las que el robot pueda quedar atrapado y optimizando las trayectorias mediante la replanificación de rutas. Mientras navega, el robot actualiza continuamente el mapa del entorno y ajusta su trayectoria en función de la información obtenida. La efectividad del enfoque fue validada mediante simulaciones en ROS Gazebo y pruebas experimentales con un TurtleBot 2, demostrando su aplicabilidad en tareas de exploración, como en áreas afectadas por desastres naturales.

Palabras clave: Robots móviles, Percepción y sensado, Navegación y control de orientación.

Online Autonomous Navigation: Integration of PRM and SLAM for Exploration of Unknown Environments

Abstract

This work presents an approach for autonomous navigation of mobile robots in unknown indoor environments by integrating SLAM and PRM algorithms. The proposed method enables both goal-directed navigation and autonomous exploration of unmapped areas, handling situations where the robot may become trapped and optimizing trajectories through path replanning. While navigating, the robot continuously updates the environment map and adjusts its trajectory based on the acquired information. The effectiveness of the approach was validated through simulations in ROS Gazebo and experimental tests with a TurtleBot 2, demonstrating its applicability to exploration tasks, such as in disaster-affected areas.

Keywords: Mobile robots, Perception and sensing, Guidance navigation and control.

1. Introducción

En el campo de la robótica móvil, se han desarrollado sistemas de navegación autónoma para permitir a los robots moverse de manera independiente en entornos desconocidos. Uno de los desafíos fundamentales en este ámbito es la creación de algoritmos eficaces que faciliten esta capacidad, ya sea con el conocimiento previo del entorno o explorando de manera autónoma para adquirirlo (Salado-Chávez et al., 2023).

Entre los algoritmos de planificación de rutas que se basan en el conocimiento previo del mapa, destaca el algoritmo de Hoja de Ruta Probabilística (PRM, por sus siglas en inglés), reconocido por su capacidad para capturar la conectividad del espacio libre de colisiones y abordar problemas de planificación de rutas de manera efectiva. Diversos trabajos se han enfocado en la aplicación y mejora de este algoritmo. Por ejemplo, Qiao et al. (Qiao et al., 2022) mejoraron la densidad y distribución de puntos de muestreo en canales estrechos mediante la combinación de procesos de aprendizaje de PRM y el algoritmo de Campo Potencial Artificial (APF). Kumar et al. (Kumar and Sikander, 2023) desarrollaron una estrategia de toma de decisiones que mejoró la eficiencia de la trayectoria al eliminar nodos no deseados y seleccionar una trayectoria óptima basada

*Autor para correspondencia: oscar6ri@hotmail.com

en el peso de las aristas. Cuando no se dispone de conocimiento previo sobre el mapa, el algoritmo de Mapeo y Localización Simultáneos (SLAM, por sus siglas en inglés) facilita la construcción simultánea del mapa y la localización del robot. A medida que el robot se desplaza, el algoritmo actualiza tanto su localización como el mapa, utilizando sensores como cámaras, LiDAR o encoders. Durrant-Whyte and Bailey (2006)

Existen diversos enfoques de SLAM, como el SLAM visual, que utiliza cámaras para la percepción del entorno y la localización del robot; el SLAM basado en LiDAR, que emplea sensores láser para la detección precisa de objetos y estructuras en 2D y 3D; y el EKF-SLAM, que utiliza el filtro de Kalman extendido para estimar la posición y orientación del robot, mientras mapea el entorno y cierra bucles mediante correlaciones entre puntos de referencia remotos Barrau and Bonnabel (2015). Por otro lado, el GraphSLAM es un algoritmo que utiliza grafos para calcular la trayectoria recorrida y generar un mapa del entorno a medida que el robot se desplaza Thrun and Montemerlo (2006). En el contexto de SLAM 2D, existen ejemplos que demuestran su eficiencia. Por ejemplo, Hess et al. (Hess et al., 2016) presentaron y validaron experimentalmente un sistema de SLAM 2D utilizando un sensor LiDAR, logrando mapeo en tiempo real y cierre de bucle con alta resolución. Afanasyev (Afanasyev et al., 2015) describió un enfoque de simulación en Gazebo para SLAM basado en el Sistema Operativo Robótico (ROS), utilizando filtros de partículas Rao-Blackwellized (RBPF) para estimar tanto la localización como el mapa de un entorno desconocido a partir de datos láser. Un enfoque interesante fue el propuesto en Rixon (Rixon Raj and Bhuvaneswari, 2020), Li et al. (Li et al., 2022b), Xuexi et al. (Xuexi et al., 2019), y Ren et al. (Ren et al., 2022), quienes emplean el algoritmo SLAM para obtener un mapa del entorno y luego utilizan algún algoritmo para generar una trayectoria planificada basada en ese mapa. Este enfoque ha sido implementado en diferentes entornos, como se muestra en Cheein et al. (Cheein et al., 2011), donde se obtuvieron resultados experimentales utilizando una silla de ruedas robotizada real. Sin embargo, este método podría enfrentar complicaciones si el entorno experimenta cambios significativos durante la navegación, lo que requeriría replanificaciones frecuentes de las rutas para adaptarse a los nuevos obstáculos o condiciones. Además, este enfoque puede no ser aplicable en entornos donde las dimensiones del mapa no se conocen de antemano, lo que podría limitar su utilidad en escenarios desconocidos.

Para superar estas limitaciones, OpenStreetMap (OSM) ofrece una base de datos geoespacial pública que incluye información detallada sobre redes viales, tipos de caminos y puntos de interés, facilitando la planificación de rutas. Al integrar esta información con datos locales obtenidos de sensores como LiDAR y cámaras, es posible ajustar las rutas en tiempo real y mejorar la precisión en entornos cambiantes (Li et al., 2022a). Métodos como Naive-Valley-Path (NVP) permiten generar trayectorias seguras, evitando obstáculos dinámicos y estáticos (Muñoz-Bañón et al., 2022). Además, la integración de OSM con mapeo semántico y topológico facilita la planificación de rutas en entornos cerrados (Naik et al., 2019). La incorporación de datos de código abierto en un sistema autónomo de navegación robótica mejora la eficiencia y adaptabilidad del sistema (Hentschel and Wagner, 2010).

Observando esta problemática, diversos investigadores han desarrollado técnicas de navegación en línea para abordar los desafíos de la navegación autónoma en entornos cambiantes (Trivun et al., 2015; Carrillo et al., 2012; Leung et al., 2006; Carlone et al., 2014; Maurović et al., 2017). Por ejemplo, Trivun et al. presentan un algoritmo para la exploración y mapeo completamente autónomos en entornos interiores, utilizando puntos de objetivo automáticos para mapear el espacio desconocido y revisitar puntos de referencia conocidos para mejorar la precisión del mapa (Trivun et al., 2015). Asimismo, Carlone et al. proponen una aplicación de la divergencia de Kullback-Leibler para evaluar la aproximación posterior del SLAM basada en partículas, permitiendo que el robot tome decisiones autónomas entre explorar nuevas áreas y revisar lugares conocidos (Carlone et al., 2014). La mayoría de estos trabajos se centran en entornos típicos para interiores, aunque hay una clara necesidad de investigar en entornos de grandes dimensiones y desconocidos, donde pueden surgir casos diversos y desafiantes.

Este artículo presenta una integración de los algoritmos SLAM y PRM para la navegación autónoma en entornos interiores. El sistema opera en dos modos principales: *modo meta definida*, orientado a alcanzar un objetivo específico, y *modo exploración*, diseñado para cubrir áreas desconocidas. Ambos modos contemplan eventualidades como que el robot quede atrapado en una habitación. Para resolver esto, el sistema registra los puntos previamente recorridos y evita seleccionar submetas cercanas, lo que le permite encontrar rutas alternativas hacia la meta deseada o abarcar nuevas áreas durante la exploración.

Adicionalmente, se estableció una condición que detiene el algoritmo cuando no se detectan más áreas por explorar. Durante la navegación, el robot actualiza continuamente el mapa utilizando SLAM en línea mientras genera rutas planificadas en tiempo real, adaptándose a las características del entorno. Se implementó el Modo meta definida empleando un *Turtlebot 2*, en la División de Estudios de Posgrado, lo cual permitió que el robot llegara a un punto deseado, generando el mapa y trayectorias conforme descubre el entorno.

Este enfoque tiene aplicaciones en distintos campos, como la exploración planetaria, la inspección de áreas afectadas por desastres naturales y la exploración submarina, donde es crucial operar en entornos desconocidos.

2. Preliminares

2.1. Modelo y control cinemático del robot diferencial

El modelo cinemático del robot diferencial considera un punto de operación desplazado una distancia h del eje de las ruedas (ver Figura 1). Es importante tener en cuenta las siguientes suposiciones (Hernández Millán et al., 2016):

- El robot se mueve sobre una superficie plana.
- El eje guía es perpendicular al plano de movimiento.
- Las ruedas se desplazan sin restricciones.
- El robot no posee partes flexibles.

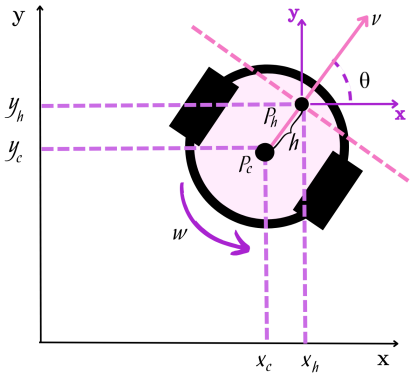

Figura 1: Modelo del robot en configuración diferencial

El diagrama del robot móvil en configuración diferencial (RMD) se muestra en la Figura 1. El modelo cinemático en un punto de operación arbitrario h (Ph) se representa en (1) (Salado-Chávez et al., 2023).

$$\begin{aligned} \dot{x_h} &= v\ cos(\theta_h)\ -\ h\ \omega\ sen(\theta_h) \\ \dot{y_h} &= v\ sen(\theta_h)\ +\ h\ \omega\ cos(\theta_h) \end{aligned} \tag{1}$$

donde $\dot{x}_h$ y $\dot{y}_h$ representan las velocidades del punto h en los ejes coordenados x y y, respectivamente. Por otro lado, v y ω denotan las velocidades lineal y angular del robot, respectivamente. Además, θ_h corresponde al ángulo de dirección formado entre la velocidad lineal v y el eje coordenado x (Sanz, 2009).

Definiendo las leyes de control como $u_x = \dot{x}_h = -k(x - xd)$ y $u_y = \dot{y}_h = -k(y - yd)$, donde $[x, y]$ son las posición del robot en el plano, y $[xd, yd]$ la posición deseada u objetivo, el modelo cinemático inverso queda expresado en (2).

$$\begin{aligned} v &= u_x\ cos(\theta_h) + u_y\ sen(\theta_h) \\ \omega &= (-u_x\ sen(\theta_h) + u_y\ cos(\theta_h))/h \end{aligned} \tag{2}$$

Algoritmo 1 Pseudocódigo para recorrer una trayectoria

```
1: función RecorrerTrayectoria(x, y, ruta_x, ruta_y)
2:     i = 1                                  ▷ Inicializar índice
3:     mientras i < tamaño(ruta_x)+1 hacer
4:         si umbral_distancia < d entonces
5:             ControlCinemático(x, y, ruta_x[i], ruta_y[i])
6:             i = i + 1
7:         fin si
8:     fin mientras
9: fin función
```

2.1.1. *Control cinemático para trayectorias*

El Algoritmo 1 permite que un robot siga una trayectoria definida por los puntos almacenados en los vectores *ruta_x* y *ruta_y*. La función RecorrerTrayectoria inicia con la posición actual (x, y) del robot y recorre los puntos de la ruta utilizando un índice i, el cual comienza en 1. Mientras existan puntos por recorrer, el algoritmo verifica si la distancia euclidiana d al siguiente punto es menor que un umbral predefinido (*umbral_distancia*). Si se cumple esta condición, se ejecuta la función ControlCinemático(x, y, *ruta_x*[*i*], *ruta_y*[*i*]) para ajustar la dirección y velocidad del robot, tras lo cual el índice i se incrementa, avanzando al siguiente punto de la trayectoria. El proceso se repite hasta que se recorren todos los puntos definidos en la trayectoria.

2.2. *Entorno*

El edificio de la "División de Estudios de Posgrado" fue modelado en 3D e incluye siete aulas, un laboratorio y oficinas. Este diseño contempla pasillos angostos, esquinas y entradas a las aulas, elementos que representan posibles puntos donde el robot podría quedar atrapado, además de permitir evaluar su capacidad para maniobrar en el entorno. La distribución se muestra en la Figura 2, junto con la vista isométrica. Las consideraciones tomadas en cuenta para la creación del modelo son las siguientes:

- Se establecieron muros para limitar áreas donde el robot no puede bajar ni subir escalones.
- Se omitieron las puertas, ya que el robot no tiene la capacidad de abrirlas.

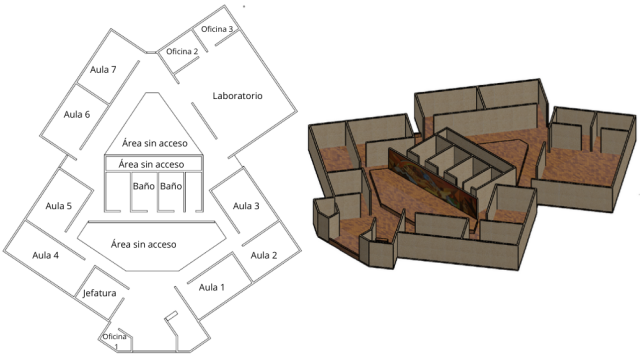

Figura 2: Vista superior del entorno

2.3. *Introducción a SLAM y PRM*

Como se menciona anteriormente, se han implementado algoritmos para la obtención del mapa del entorno y, posteriormente, se ejecuta un algoritmo de navegación. A continuación, se muestra el proceso para realizar el mapeo y, posteriormente, navegar empleando el método PRM.

Uno de los algoritmos de SLAM más utilizados es el Gmapping el cual emplea sensores láser. ROS cuenta con un paquete que utiliza Gmapping de OpenSlam, el cual proporciona un nodo llamado *slam_gmapping* (Villarreal Onofre et al., 2021). Con *slam_gmapping* se genera un mapa en 2D como se muestra en la Figura 3, a partir de datos láser y de la posición recopilados por el robot.

El procesamiento de imágenes convierte el mapa (.pgm) generado por SLAM en una imagen binarizada, identificando las áreas transitables. Posteriormente, el proceso de erosión define zonas de seguridad alrededor de los obstáculos, asegurando rutas libres de colisiones (el Algoritmo se puede consultar en el Contenido Digital). La Figura 3 ilustra el mapa obtenido, la binarización y la erosión, antes de la obtención de la trayectoria mediante PRM. Una vez obtenido el mapa del entorno mediante SLAM y haber establecido las zonas transitables, considerando una zona de seguridad mediante el procesamiento de imagen, se

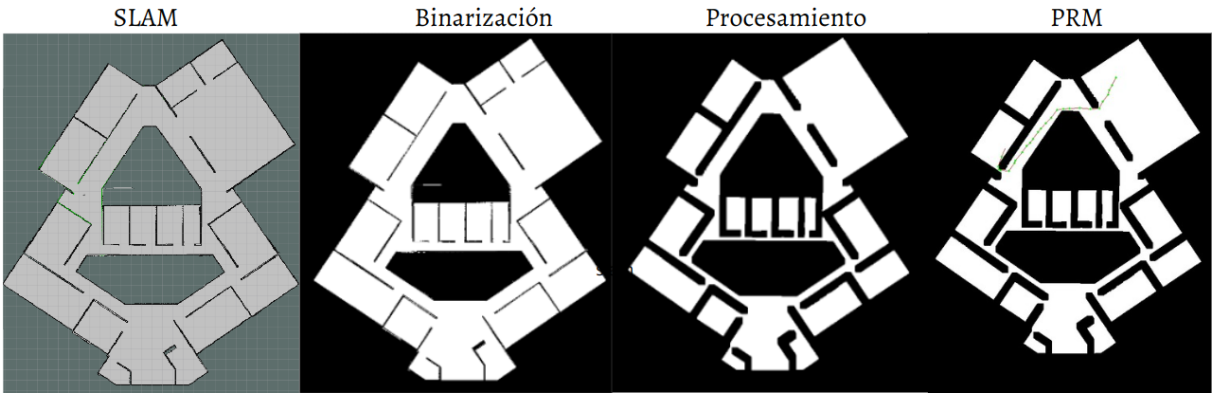

Figura 3: Mapa obtenido con SLAM en RVIZ

necesita planificar una trayectoria segura para el robot móvil. El algoritmo de muestreo de caminos probabilístico (PRM), propuesto por Kavraki en 1996 (Kavraki et al., 1996), se utiliza para generar rutas libres de obstáculos. Este algoritmo consta de dos fases:

- Se crea un grafo de red compuesto por nodos y conexiones en áreas libres de colisiones.
- Se buscan los nodos conectados que van desde el punto inicial hasta la posición objetivo (Alpkiray et al., 2018).

Los pseudocódigos se pueden consultar en el Contenido Digital[1]. Siguiendo este enfoque, se obtiene una ruta que permite al robot navegar en la zona libre de obstáculos, como se muestra en la Figura 3.

3. Algoritmo PRM-SLAM: Modo meta definida

3.1. *Navegación con ruta preestablecida (offline)*

Como se mencionó anteriormente, un enfoque utilizado es ejecutar el algoritmo SLAM para obtener un mapa del entorno y luego utilizar un algoritmo para generar una trayectoria basada en el mapa obtenido previamente (Rixon Raj and Bhuvaneswari, 2020), (Li et al., 2022b), (Xuexi et al., 2019), (Ren et al., 2022). En este caso se utilizó PRM (Figura 3), y se recorrió la trayectoria mostrada en la Figura 4. Para más información se puede consultar en el Contenido Digital[1].

Partiendo de la implementación independiente de los algoritmos, el presente trabajo propone utilizar SLAM en línea, siguiendo el enfoque utilizado por (Trivun et al., 2015) y (Carlone et al., 2014), para la navegación sin una ruta preestablecida. En el caso anterior, se obtuvo un mapa previo en el que no existían las oficinas 2 y 3. Si se producen cambios en el edificio, con un enfoque basado en una ruta preestablecida, es necesario repetir todo el proceso para actualizar el mapa. Por lo tanto, se propone el uso simultáneo de PRM-SLAM para planificar la ruta a medida que el robot genera el mapa mientras navega.

El proceso para navegar hasta la meta deseada, sin conocer el mapa, se detalla en el Algoritmo 2 (y el Diagrama en el Contenido Digital[1]).

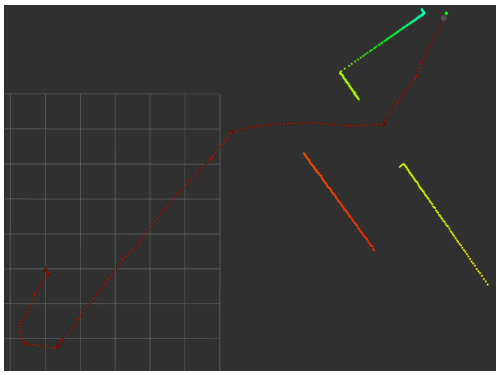

Figura 4: Trayectoria seguida por el Turtlebot 3

Algoritmo 2 Pseudocódigo PRM-SLAM Meta Definida

```
1:  función PRMSLAM
2:      mientras Posición <> Meta hacer
3:          ImagenRVIZ=EjecutarSLAM
4:          ProcesarImagen(ImagenRVIZ)
5:          si La Meta está en la imagen entonces
6:              Ruta = PRM(Imagen, pr, Meta, m, v)
7:          si no
8:              PCM = ObtenerPCM(Imagen)
9:              mientras Ruta==None hacer
10:                 Ruta = PRM(Imagen, pr, PCM, m, v)
11:                 PCM = ObtenerNuevoPCM(Imagen)
12:             fin mientras
13:             Limpiar PCM no Accesible
14:             Guardar PCM y puntos de la trayectoria.
15:             RecorrerTrayectoria((x, y, PCM_x, PCM_y))
16:         fin si
17:     fin mientras
18: fin función
```

Una vez que se establece la meta deseada se ejecuta el SLAM, el cual genera una representación del entorno mediante RVIZ. Partiendo de esta representación se obtiene una imagen la cual se somete a un proceso de binarización y erosión, ajustándose al radio del robot para establecer una distancia de seguridad. En la zona blanca del mapa, correspondiente a las

[1]El Contenido Digital se puede consultar en `https://github.com/itzchav/PRM-y-SLAM-Exploracion-de-Entornos`

áreas libres de obstáculos, se generan puntos aleatorios. Luego, se calcula la distancia de cada punto a la meta y se selecciona el *Punto más cercano a la meta (PCM)*, que se convierte en el destino final del algoritmo PRM para el mapa descubierto hasta ese momento (ver Algoritmo 3).

Algoritmo 3 Obtener PCM

```
1: función ObtenerPCM(imagen, PCM anteriores, ruta)
2:     Definir radios
3:     mientras puntos generados < nodos hacer
4:         Punto=GenerarPuntoAleatorioEnEspacioBlanco
5:         Evaluar que el Punto no este dentro del radio de los
   PCM anteriores o de los puntos de las rutas anteriores
6:         Calcular la Distancia a la Meta (DM)
7:         si DM < distanciaPCM entonces
8:             PCM = (px, py)
9:             distanciaPCM = DM
10:        fin si  ▷ Búsqueda del punto más cercano a la meta
11:    fin mientras
12:    return PCM
13: fin función
```

La posición del robot se establece como punto inicial, y se ejecuta el algoritmo PRM para determinar la ruta óptima hacia la meta. A medida que el robot avanza y se desplaza, se actualiza el mapa parcial del entorno utilizando SLAM. Se generan nuevos puntos aleatorios que funcionan como nodos en la planificación de la ruta mediante PRM, se consideran los *PCM* anteriores y los puntos de trayectorias como *landmarks*, alrededor de los cuales se define una zona radial, excluyendo esta área en la que se pueden generar los puntos. Este proceso se repite iterativamente hasta que el robot alcanza la meta deseada.

3.2. *Simulación PRM-SLAM: Modo meta definida*

A continuación, se presenta la ejecución del método PRM-SLAM en su *modo meta definida*. El funcionamiento general del algoritmo se ilustra en el siguiente código, el proceso se describe en el diagrama de funcionamiento. Para una mejor comprensión, se expone también el proceso de navegación en el diagrama de navegación, adicionalmente, se puede consultar un video del procedimiento, en el Contenido Digital[1].

En esta prueba, el robot comienza en el Aula 6 y tiene como objetivo llegar a la Oficina 2. Para ello, el robot debe primero encontrar una salida desde el Aula 6. De acuerdo con el Algoritmo 2, se genera un mapa utilizando la técnica SLAM, cuyos resultados se visualizan en la herramienta RVIZ, tal como se muestra en la Figura 5. El sensor utilizado tiene un alcance de 12 metros, lo que significa que las paredes y otros obstáculos en el entorno bloquean la visión directa del punto deseado, complicando la navegación hacia la meta. Se ejecuta el algoritmo SLAM y se procesa la imagen para determinar las áreas libres y los obstáculos. Para determinar la primera submeta, se utilizaron inicialmente 30 puntos aleatorios, de los cuales se seleccionó el *PCM* como el punto deseado para el PRM.

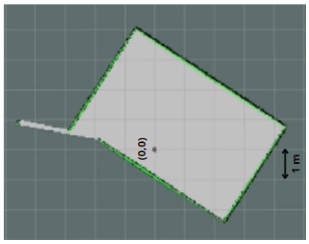

Figura 5: Primer lectura realizada con el robot

El número inicial de nodos para ejecutar el PRM fue de 2000, los cuales se fueron aumentando en 50 por cada *PCM* que se necesitara generar. Esto se debió a que, al aumentar el área que se descubrió, incrementaba el número de píxeles en la imagen; por lo tanto, para generar un grafo en un área más grande, era necesario establecer más nodos para asegurar que se generara la trayectoria. La distancia máxima entre los nodos se estableció en 10 píxeles.

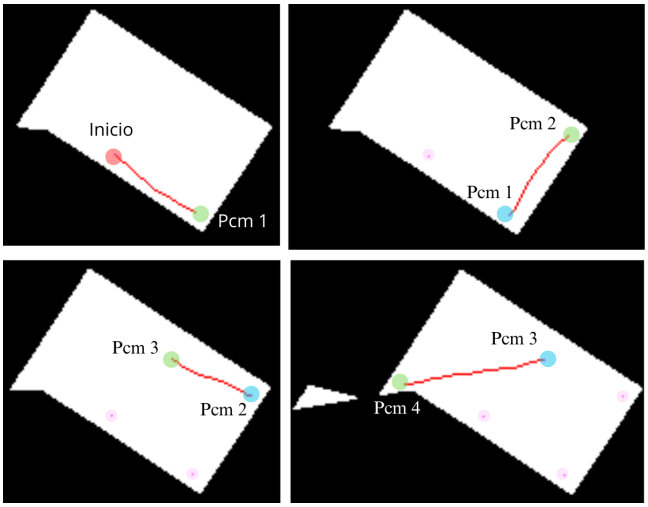

Figura 6: Rutas generadas

En la Figura 6 se muestran las primeras cuatro rutas generadas con el algoritmo PRM, se muestra el inicio en color rojo, y inicio de cada trayectoria en azul, la submeta en verde, y se muestran los *PCM* que se generan con cada iteración en color rosa. Una vez que se genera la trayectoria, el algoritmo SLAM sigue en ejecución mientras el robot se desplaza hacia el *PCM*. Una vez que el robot alcanza este punto, el mapa se convierte nuevamente en una imagen y se selecciona un nuevo *PCM*.

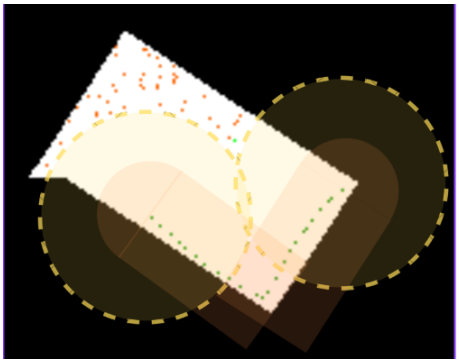

Figura 7: Área restringida

Para generar el *PCM* (ver Algoritmo 3), se crean nuevos puntos aleatorios en cada actualización del mapa, añadiendo 5 puntos adicionales. Además, tanto los *PCM* como los puntos de la trayectoria se utilizan como centros de *Áreas Restringidas*, en las que no se pueden generar nuevos puntos, con el objetivo de indicar que el robot ya ha recorrido esa zona, como se muestra en la Figura 7. De esta manera, se van almacenando los *PCM* y los puntos de la trayectoria que el robot ha recorrido. Se estableció un área con un radio de 40 píxeles, equivalente a aproximadamente 3 metros.

Uno de los problemas que puede surgir es la selección de un *PCM* (punto cercano a la meta) en un área inaccesible. Esto sucede cuando el robot detecta, a través de un espacio reducido, que es posible generar un área disponible. Sin embargo, al aplicar el proceso de erosión en el mapa, se elimina el acceso a dicha área, como se ilustra en la Figura 8 (punto verde). Esto imposibilita generar una trayectoria viable y obliga a seleccionar un nuevo *PCM*.

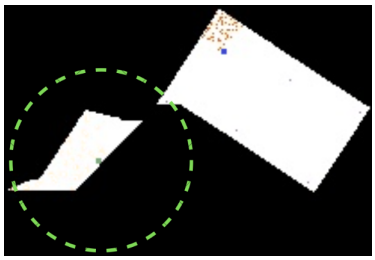

Figura 8: Nuevo *PCM* generado

Para resolver este inconveniente, se utiliza el principio de *Áreas Restringidas*, que se aplica durante la obtención de los *PCM*, tal como se detalla en el Algoritmo 3. En este caso, se modifica la línea 11 del algoritmo para considerar el vector que almacena los puntos sin acceso, como se observa en el Algoritmo 4. Este ajuste implica modificar las condiciones del área de restricción basándose en el vector de *PCM* inaccesibles, restringiendo así las áreas cercanas a estos puntos en ese momento.

Algoritmo 4 Obtener *PCM* al no encontrar ruta

```
1:  función ObtenerNuevoPCM(imagenBinaria, imagenColor, centros, ruta, nodos, PCMNoAccesible)
2:      Obtener los PCM no accesibles
3:      mientras puntos generados < nodos hacer
4:          Punto=GenerarPuntoAleatorioEnEspacioBlanco
5:          Evaluar que el Punto no este dentro del radio d de los PCM anteriores, de los PCM no accesibles o dentro del radio dt de los puntos de las rutas anteriores
6:          Calcular la Distancia a la Meta (DM)
7:          si DM < distanciaPCM entonces
8:              PCM = (px, py)
9:              distanciaPCM = DM
10:         fin si  ▷ Búsqueda del punto más cercano a la meta
11:     fin mientras
12:     Borrar los PCM no accesibles
13:     return PCM
14: fin función
15:
```

De este modo, se selecciona un nuevo *PCM*, como se ejemplifica en la Figura 8 (punto azul). Posteriormente, el vector que contiene los puntos inaccesibles se actualiza, eliminando aquellos que no representan áreas permanentemente bloqueadas.

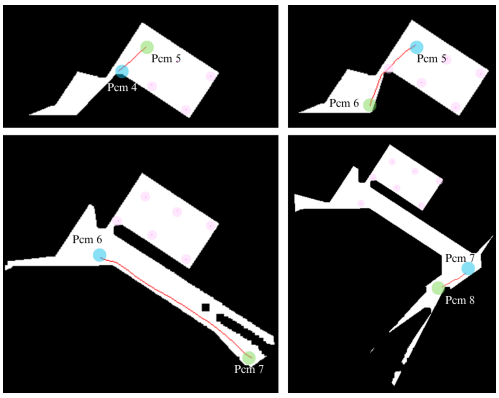

Figura 9: Rutas generadas al salir del Aula 6

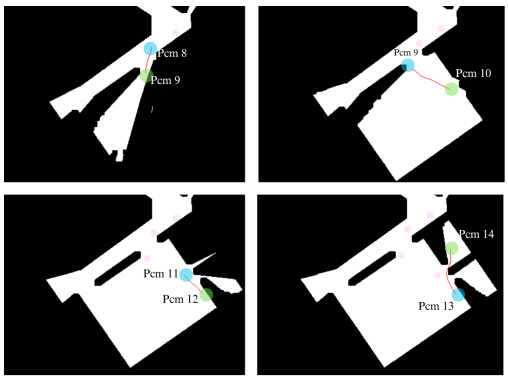

Figura 10: Rutas generadas hasta llegar a la meta

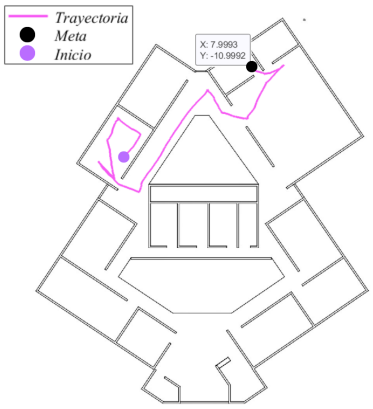

Figura 11: Trayectoria final de la prueba

Esto es relevante porque, si en el futuro se encuentra un camino alternativo, las áreas restringidas cuyo centro corresponde a los *PCM* previamente inaccesibles podrían volverse accesibles nuevamente, dado que el robot aún no ha navegado por esas zonas. Al ir restringiendo las áreas por las que el robot ya ha pasado, el robot logra salir del Aula 6. Esto se muestra en la Figura 9, donde se observan los *PCM* anteriores y se puede

ver que el algoritmo se adapta al entorno, buscando el punto deseado. Aunque en la Figura 4 se observa que el robot tiene una trayectoria directa, esto se debe a que ya se tiene conocimiento previo del mapa. En este caso, el robot logra encontrar una salida sin tener conocimiento exacto de su ubicación. Se generan cuatro trayectorias hasta que el robot encuentra la meta, las cuales se muestran en la Figura 10. También se observa que el robot se adapta a los cambios del entorno, ya que tiene en cuenta las dos oficinas agregadas después de haber obtenido el mapa de la Figura 3. En la Figura 11 se muestra la trayectoria total recorrida por el robot.

3.3. Pruebas PRM-SLAM: Modo meta definida

Las pruebas se realizaron en el interior de la División de Estudios de Posgrado, considerando diferentes sistemas de coordenadas globales, el video de las pruebas realizadas con un punto final de [8,-11] con respecto al origen del robot se puede consultar en el Contenido Digital[1]. En la Figura 12 se muestran los puntos de inicio y final de cada prueba, los cuales se describen a continuación:

1. Aula 6 al Laboratorio
2. Entrada a la Aula 2
3. Aula 4 al Baño
4. Frente al mural a un punto fuera del entorno
5. El robot con una distancia menor a la de seguridad a un punto fuera del entorno

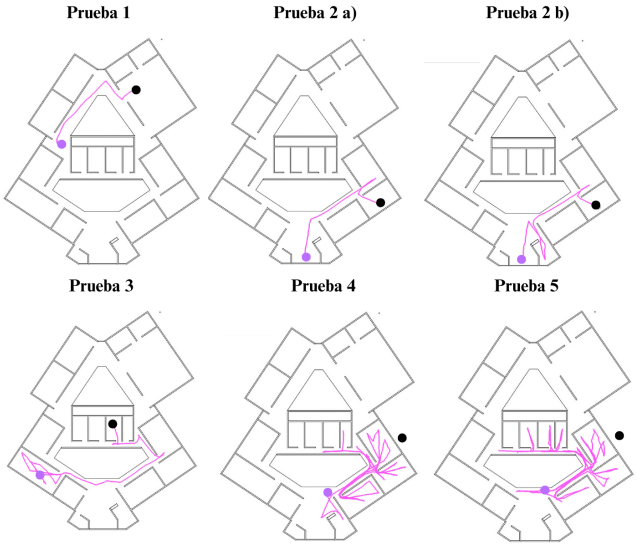

Figura 12: Trayectorias de las pruebas realizadas

En la Figura 12 se presentan las trayectorias de las pruebas realizadas. Cabe destacar que en la prueba 2 se obtuvieron dos trayectorias distintas, ya que el algoritmo depende de puntos aleatorios y seleccionó uno adyacente a un área en la que no hay acceso. Sin embargo, se logró llegar a la meta. Por otro lado, en la prueba 4 y la prueba 5, la meta no es un punto accesible, por lo que el robot navega intentando encontrar una salida. En la prueba 5 se estableció el inicio muy cerca de un obstáculo, dentro de lo que se considera el área de seguridad, para observar que el algoritmo es robusto y guía el robot hacia una zona en la que es posible navegar de forma segura.

Se realizaron pruebas a dos puntos finales. La prueba 7 inicia en la entrada y termina en [31,-6.5] la cual llega a la Oficina 3, y la prueba 8 inicia en el Aula 2 y termina en [16.5,17.5] la cual termina en el Aula 7.

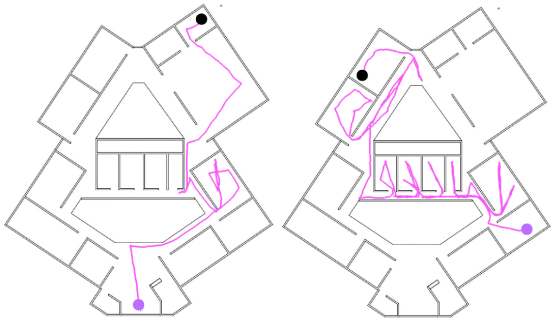

Figura 13: Pruebas con diferente punto de inicio y final

Por otro lado se propone que este algoritmo podría ser asistido, por lo que se ejecutó una prueba para obtener un mapa, hasta una meta deseada. Este mapa se conserva, y luego el algoritmo se reinicia desde el punto final de la primera ejecución, utilizando dicho punto como nuevo inicio. En la segunda ejecución, se establece una nueva meta y se asigna un número de veces que se ha ejecutado el algoritmo, lo que incrementa el número de nodos en el grafo y permite generar una trayectoria válida. De esta forma,el robot puede descubrir una mayor área del entorno mientras se asignan puntos de manera asistida.

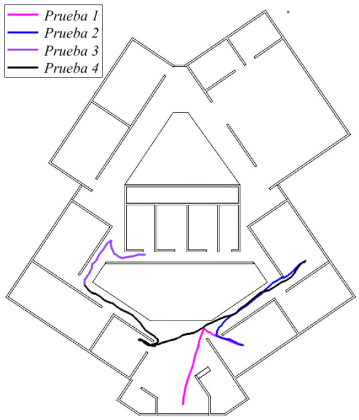

Figura 14: Trayectoria de la prueba asistida

3.4. Implementación de PRM-SLAM: Modo meta definida en Turtlebot 2

Para ejecutar el algoritmo en el *Turtlebot 2*, se utilizó una Jetson Nano. En (Salado-Chávez et al., 2023) se describen las adaptaciones realizadas para enviar datos de forma remota. La ejecución del *PRM-SLAM: modo meta definida* se llevó a cabo en el edificio de la División de Estudios de Posgrado, desde el Aula 7 hasta el laboratorio. La meta establecida fue [11,-2.5], lo que implica un recorrido de más de 10 metros, dividido en 5 trayectorias.

El código se ejecutó en una computadora, desde la cual se enviaban, vía Wi-Fi, las velocidades lineal y angular al robot. Simultáneamente, se recibía la información del LiDAR a través de Wi-Fi para generar el mapa mediante SLAM. La prueba se

llevó a cabo en un entorno controlado ubicado en la División de Estudios de Posgrado, el video y el código se pueden consultar en el Contenido Digital[1].

Los *PCM* y los puntos de las trayectorias obtenidas durante el recorrido, en relación con el entorno, se muestran en la Figura 15. Estos puntos correspondieron a las ubicaciones donde se realizaron las actualizaciones del mapa, ya que el envío y recepción de datos de forma simultánea no era viable. Cuando se intentaba mapear mientras el robot avanzaba, este se detenía cada vez que se ejecutaba una operación de mapeo. Por esta razón, los *PCM* se definieron como los puntos en los que el robot debía detenerse para realizar la lectura y actualización del mapa. Para mejorar la calidad de la lectura, se implementó un giro completo en el mismo lugar antes de determinar la siguiente ruta posible. En esta prueba, se utilizó un área restringida de 40 píxeles para las actualizaciones y un área de 10 píxeles en torno a los puntos de la trayectoria. Los mapas generados en cada *PCM* se muestran en la Figura 16.

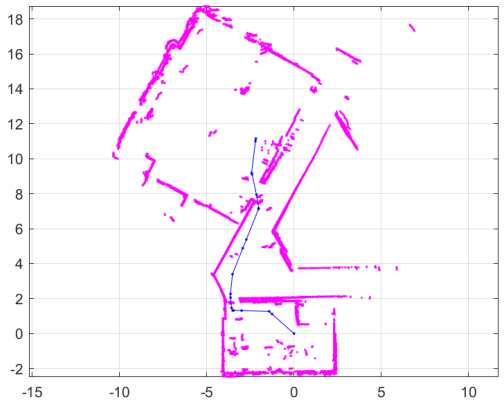

Figura 15: Puntos en los que se obtuvo el mapa con SLAM

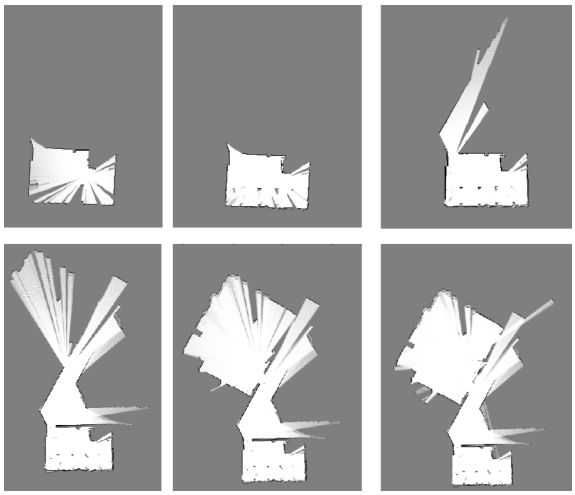

Figura 16: Actualización del mapa del entorno real

4. Algoritmo PRM-SLAM: Modo exploración

A partir del algoritmo PRM-SLAM en su modo de meta definida, se introdujo una modificación para permitir la exploración en un entorno completamente desconocido. Esta modificación incluyó dos cambios fundamentales:

1. La Selección del punto más lejano al origen (PLO). Al no contar con una meta definida, el robot selecciona como objetivo el punto más lejano al origen. Para implementar esto, se realizaron modificaciones en las líneas 14, 15 y 16 del Algoritmo 3, tal como se muestra en el Algoritmo 5. Este enfoque permite dirigir al robot hacia áreas aún no exploradas, maximizando la cobertura del entorno.
2. Cálculo de submetas mediante bisectrices. Una vez seleccionadas tres submetas, se eligen dos puntos en los bordes de la imagen y se genera un tercer punto aleatorio. Con estos tres puntos, se calcula el punto de intersección de las bisectrices de los ángulos formados entre ellos, lo que permite establecer una submeta central que guíe la exploración de manera equilibrada y estratégica.

Además, se incorporaron condiciones de detención del algoritmo, tales como:

- Un límite máximo en la cantidad de submetas generadas.
- Un límite basado en la cantidad de veces que se procesa el mismo mapa, en caso de que no existan más áreas por explorar.

Algoritmo 5 Obtener PLO

```
1: función ObtenerPlo(imagenBinaria, imagenColor, centros, ruta, nodos)
2:     si distancia > distanciaPLO entonces
3:         plo = (px, py)
4:         distanciaPLO = distancia
5:     fin si
6: fin función
```

Estas modificaciones se describen en el Algoritmo 6 (se puede consultar el diagrama de funcionamiento y el código en el Contenido Digital[1]), las cuales permiten la exploración en entornos desconocidos y evitan iterar innecesariamente sobre zonas ya conocidas. El propósito de esta modificación es evitar que la exploración se concentre en una región específica del entorno. Sin la intervención de las bisectrices, el algoritmo, al seleccionar el PLO, tiende a orientar el movimiento del robot hacia áreas particulares (hacia la derecha o hacia la izquierda del espacio), lo que podría generar un descubrimiento no uniforme del entorno. La introducción del punto de intersección de las bisectrices tiene como objetivo redistribuir la exploración, permitiendo que el robot cubra áreas más amplias del espacio de manera equilibrada, minimizando la concentración en una zona específica.

De acuerdo con el contador, se determina cuáles serán los puntos a seleccionar, y si estos se elegirán de forma aleatoria o en función del valor del contador. Así mismo, si se obtiene una misma imagen después de ejecutar las 9 condiciones, el algoritmo se detiene, pues eso indica que ya no existe área que descubrir.

1. Primer y último punto blanco en el eje X.
2. Primer punto blanco en los bordes izquierdo y derecho.
3. Primer punto blanco en los bordes inferior e izquierdo.
4. Primer y último punto blanco en el eje Y.
5. Primer punto blanco en los bordes superior e inferior.

6. Primer punto blanco en los bordes inferior e izquierdo.
7. Primer punto blanco en los bordes inferior y derecho.
8. Primer punto blanco en el borde inferior y último en el borde izquierdo.
9. Último punto blanco en el borde superior y primer en el borde izquierdo.

Algoritmo 6 Pseudocódigo PRM-SLAM: modo exploración

```
Definir limite,
mientras Contador <> Limite hacer
    ImagenRVIZ=EjecutarSLAM
    ProcesarImagen(ImagenRVIZ)
    Calcular el incremento de tamaño entre la imagen actual y la anterior
    si No Incrementa entonces
        Se cuentan cuantas veces no ha crecido el mapa
        Se obtiene el PLO a partir de las bisectrices de acuerdo a las 9 condiciones
        si Incrementa entonces
            Se reinicia el contador
        fin si
    si no
        si Incrementa n-veces entonces
            Se obtiene el PLO a partir de las bisectrices de forma aleatoria.
        si no
            Plo=ObtenerPlo( )
        fin si
    fin si
    mientras Ruta==None hacer
        Ruta = PRM(Imagen, pr, Plo, m, v)
        Plo = ObtenerNuevoPlo(Imagen)
    fin mientras
    Limpiar PloNoAccesible
    Guardar PLO y puntos de la trayectoria.
    RecorrerTrayectoria((x, y, plo_x, plo_y))
fin mientras
```

Para ejemplificar el proceso en el que el algoritmo se detiene cuando ya no existe un incremento en la imagen, se realizó una prueba en el entorno *world* de ROS ya que sus dimensiones son pequeñas, por lo que se validaron las condiciones en este espacio (El el video de la prueba se puede consultar en el Contenido Digital[1]). En la Figura 17 se muestran los puntos obtenidos a partir de la bisectriz, para terminar el algoritmo. Al cumplir con las condiciones, el algoritmo termina su ejecución.

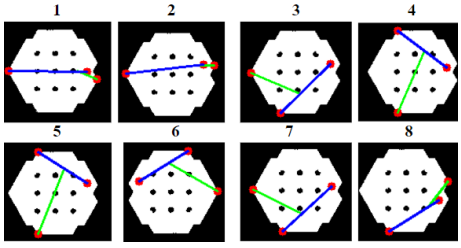

Figura 17: Bisectrices generadas

4.1. *Simulación PRM-SLAM: Modo exploración*

Se realizó una prueba en la *División de Estudios de Posgrado* con un límite de 10 submetas, antes de detener la ejecución. El el video de la prueba se puede consultar en el Contenido Digital[1]. En la Figura 18 se muestran las primeras cuatro rutas generadas.

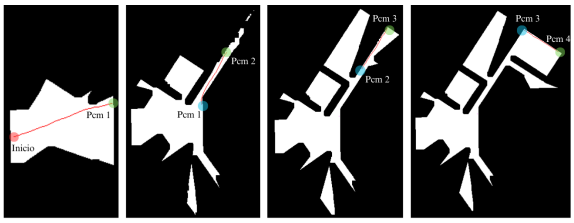

Figura 18: Primeras rutas realizadas

En las tres trayectorias mostradas (Figura 20), se ilustra el uso de los contadores para determinar el final del proceso de exploración. Estos contadores miden cuántas veces el mapa tuvo un incremento menor al 10 %. En la Figura 18, se observa que entre la tercera y la cuarta ruta no se produce un incremento superior al 10 %, lo que provoca que el contador alcance un valor de 2 y genere la primera bisectriz (Figura 19).

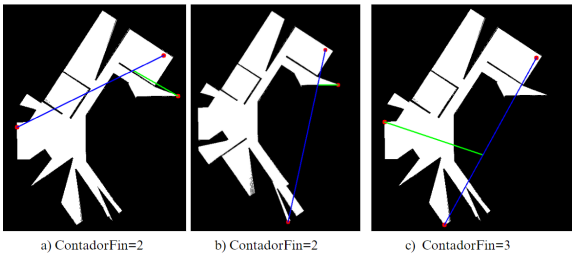

Figura 19: Intersección de las bisectrices

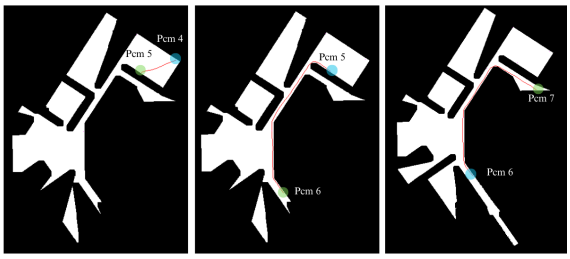

Figura 20: Rutas obtenidas a partir de las bisectrices

En esta prueba, cuando el contador alcanza el valor de 3 y no se detecta un incremento superior al 3 %, se seleccionan entre las 9 opciones disponibles los puntos necesarios para calcular la intersección de las bisectrices. Con el incremento del contador, se genera una nueva bisectriz. Sin embargo, uno de los puntos seleccionados se encuentra en una zona erosionada, lo que obliga a buscar un punto alternativo cercano. Como resultado, se generan dos rutas, tal como se muestra en la Figura 20. La primera ruta se dirige hacia la derecha, hacia un entorno aún no explorado, y posteriormente regresa hacia la izquierda, donde se descubre un área más extensa para explorar. Este descubrimiento provoca un nuevo incremento en el mapa, lo que

reinicia los contadores, indicando que aún hay áreas por explorar. El contador de finalización solo se activará nuevamente cuando no se detecten más incrementos en el mapa.

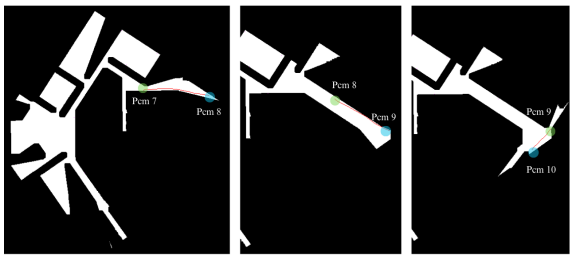

Figura 21: Últimas rutas realizadas

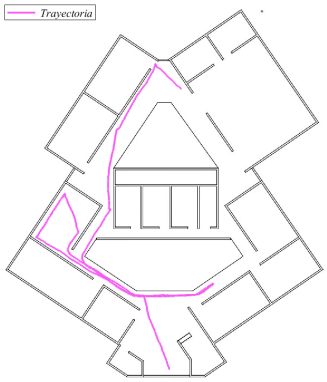

Figura 22: Trayectoria recorrida durante toda la prueba

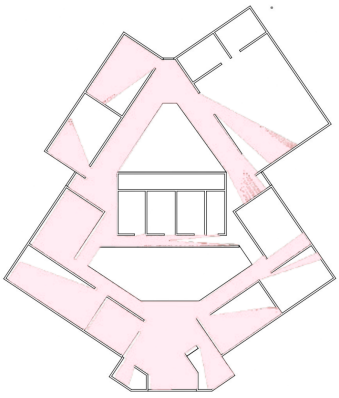

Figura 23: Mapa generado en comparación con el entorno completo

Finalmente, se obtienen las tres rutas restantes, como se muestra en la Figura 21. Dado que el contador de submetas alcanza el límite establecido, el algoritmo concluye su ejecución. La trayectoria completa realizada durante el proceso se ilustra en la Figura 22. Adicionalmente, el resultado obtenido se compara con el plano completo del entorno, como se observa en la Figura 23, donde el mapa generado aparece sombreado.

4.2. Pruebas PRM-SLAM: Modo exploración

Se realizaron tres pruebas diferentes para evaluar el rendimiento del algoritmo en diversos escenarios y configuraciones (Figura 26). El video de las pruebas está disponible en el Contenido Digital[1]. A continuación, se detallan las condiciones bajo las cuales se llevaron a cabo las pruebas; los resultados se presentan en la Figura 24:

- Prueba a): El robot comienza en la entrada, con un límite de 15 submetas.
- Prueba b): En esta prueba, se aumenta el límite de submetas a 20.
- Prueba c): El robot inicia en el aula 4, con un límite de 10 submetas.

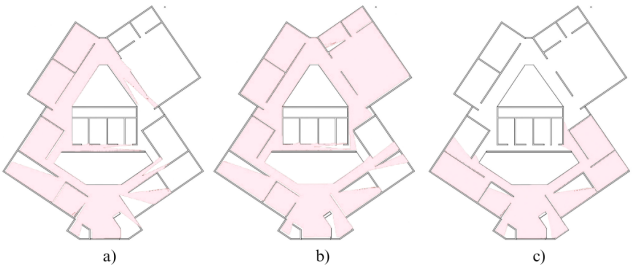

Figura 24: Pruebas de exploración realizadas

De los resultados obtenidos, se observa que al ejecutar el algoritmo repetidamente, el robot logra explorar una mayor área. Sin embargo, esto depende en gran medida del punto de inicio y de las condiciones aleatorias de los puntos generados durante la ejecución. Por ejemplo, al comparar las pruebas a) y c), que tienen el mismo número de iteraciones, se nota una menor área explorada en la prueba c). Este comportamiento indica que la localización inicial del robot tiene un impacto significativo en los resultados. El número de submetas se limita en este estudio debido a las condiciones ideales de la simulación. Sin embargo, en un entorno real con un robot físico, estas condiciones pueden variar considerablemente. Factores como errores acumulados en la odometría o la capacidad de la batería pueden afectar el rendimiento del robot, aumentando la probabilidad de colisiones o impidiendo la exploración completa del área. Se realizó una prueba con dos cajas bloqueando el acceso a zonas explorables (Figura 25). Utilizando el algoritmo SLAM Gmapping de ROS, el robot actualizó el mapa al acercarse lo suficiente para detectar la ausencia de los obstáculos. Sin embargo, si el robot no se aproxima lo suficiente, los obstáculos permanecen en el mapa debido a la dependencia de Gmapping en observaciones cercanas para actualizar las celdas ocupadas. Esto demuestra que la precisión del mapeo dinámico está condicionada a la proximidad del robot a los objetos.

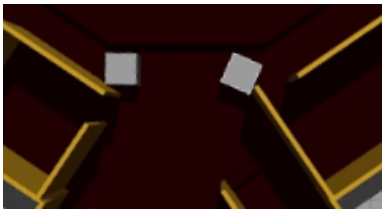

Figura 25: Obstáculos en el entorno

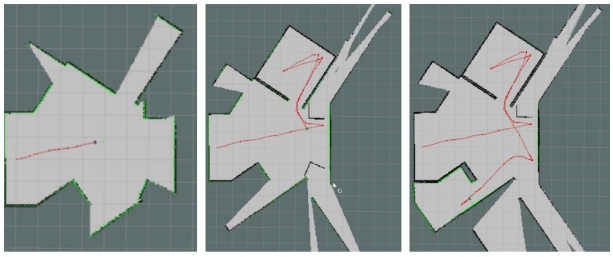

Figura 26: Obstáculos eliminados

Se realizó una comparación utilizando el código *drive* de ROS, diseñado para la navegación evitando colisiones, en tres entornos distintos: dos entornos predeterminados de ROS (*world* y *house*) y el entorno de la División de Estudios de Posgrado.

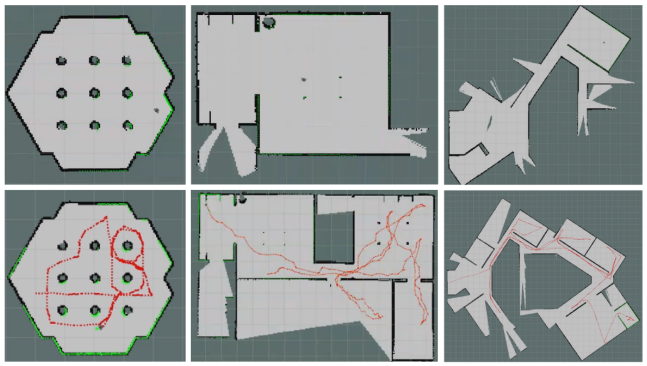

Figura 27: Comparación de métodos para explorar un entorno

En la Figura 27, se muestra, en la parte superior, la ejecución del código predeterminado de ROS. En esta ejecución, el robot logra una exploración completa en entornos pequeños; sin embargo, en entornos más grandes, solo explora zonas reducidas, quedándose atrapado en áreas específicas o llegando a colisionar. Además, en la Figura 27 inferior, se muestra la ejecución del algoritmo *PRM-SLAM: modo exploración*, donde se aprecia que este logra cubrir una mayor área al navegar en el entorno. Los resultados completos se pueden consultar en el video disponible en el Contenido Digital[1].

La Tabla 1 resume los resultados de las pruebas, incluyendo puntos de inicio y destino, distancia euclidiana recorrida y número de rutas generadas. Se realizaron seis pruebas con diferentes puntos iniciales y finales, manteniendo la misma distancia para evaluar el desempeño del robot. Luego, se probó con una distancia mayor y dos casos donde la meta estaba fuera del entorno, observando que el robot continuó navegando en busca de puntos cercanos. Finalmente, se evaluó la capacidad del algoritmo para alcanzar la meta. Los resultados muestran que el robot llega a los puntos deseados sin importar la distancia, la posición inicial o final. En la prueba asistida, se asignaron manualmente puntos de destino y, tras alcanzarlos, se reasignaron nuevas metas, permitiendo mapear el entorno con asistencia de un operador.

Tabla 1: Pruebas realizadas para rutas del robot modo meta definida

	Inicio	**Fin**	**Dist. [m]**	**N. Rutas**	**Completó**
1	Pasillo	Laboratorio	13.60	4	Si
2	Entrada	Aula 2	13.60	4	Si
3	Entrada	Aula 2	13.60	6	Si
4	Aula 4	Baño	13.60	18	Si
5	Vestíbulo	Punto Fuera	13.60	22	No
6	Vestíbulo	Punto Fuera	13.60	22	No
7	Entrada	Oficina 3	31.67	14	Si
8	Aula 2	Aula 7	24.05	26	Si
9	*Asistido*		12.85	8	Si
	Entrada	Aula 1			Si
	Aula 1	Aula 3			Si
	Aula 3	Pasillo			Si
	Pasillo	Baños			Si
10	Aula 7	Laboratorio	11.28	4	Si

Por otro lado, la Tabla 2 concentra los resultados de las pruebas del algoritmo de exploración en distintos entornos. Se iniciaron pruebas desde diversos puntos y se evaluó el porcentaje de área descubierta en función del número de submetas asignadas. En la evaluación de distintos métodos utilizando el algoritmo drive de ROS, no se estableció un número fijo de metas, sino que la navegación se detuvo cuando se observó un comportamiento cíclico.

Tabla 2: Pruebas realizadas para rutas del robot en modo exploración

	Entorno	**Inicio**	**Submetas**	**Método**	%
1	Div. Pos.	Entrada	10	Exp.	< 80
2	Div. Pos.	Entrada	15	Exp.	< 70
3	Div. Pos.	Entrada	20	Exp.	< 90
4	Div. Pos.	Aula 4	10	Exp.	< 60
5	Exp.	Aula 4	10	Exp.	< 40
6	wold	3,-0.2	-	*drive*	< 100
7	house	0,0	-	*drive*	< 50
8	Div. Pos.	Entrada	-	*drive*	< 40
9	wold	3,-0.2	11	Exp.	< 100
10	house	0,0	17	Exp.	< 90
11	Div. Pos.	Entrada	26	Exp.	< 80

5. Conclusión

La integración de SLAM y PRM ha demostrado ser una estrategia efectiva para la navegación en entornos desconocidos, permitiendo la generación de mapas actualizados en tiempo real y la planificación de rutas adaptativas. Este enfoque no solo facilita la exploración del entorno, sino que también garantiza una navegación eficiente al buscar rutas alternativas cuando la meta no es directamente accesible, evitando colisiones y asegurando que el robot alcance su objetivo.

Durante las pruebas, el algoritmo *PRM-SLAM: modo meta definida* permitió al robot desplazarse hacia distintos objetivos desde múltiples puntos de inicio. En escenarios donde la salida de habitaciones presentaba dificultades, el robot registró los puntos por los que había navegado para evitar regresar a áreas

previamente exploradas, lo que previno la formación de ciclos dentro de zonas cerradas. Además, al almacenar estos puntos junto con las trayectorias utilizadas para el cierre de bucles, se evitó que dichas zonas fueran seleccionadas nuevamente como metas. En el futuro, estos puntos podrían emplearse como landmarks, enriqueciendo la detección de cierres de bucles y mejorando la precisión del mapeo.

Por otro lado, el algoritmo *PRM-SLAM: modo exploración* demostró su capacidad para navegar en entornos desconocidos mientras recopilaba información del entorno. La implementación de bisectrices facilitó la exploración en múltiples direcciones, evitando la limitación de avanzar únicamente en una sola dirección y maximizando la eficiencia en la cobertura del espacio. Este enfoque no solo permite la generación de mapas en entornos no estructurados, sino que también fomenta una exploración continua y adaptativa, ajustándose dinámicamente a los cambios detectados en el entorno. Su capacidad de adaptación en tiempo real lo hace adecuado para diversas aplicaciones de robótica móvil, como la exploración de áreas afectadas por desastres naturales o la navegación en entornos altamente dinámicos.

Los resultados obtenidos en simulaciones con ROS Gazebo respaldan la eficacia del método PRM-SLAM en escenarios dinámicos. Asimismo, su implementación en un entorno real permitió identificar desafíos asociados a este enfoque, como el rendimiento de la batería, los errores acumulados en la odometría y las irregularidades del terreno, factores que influyen en la precisión del sistema.

En términos generales, el algoritmo PRM-SLAM demuestra una gran capacidad de adaptación a distintos requerimientos de navegación y exploración. Si la meta se encuentra en un área accesible, el modo ‘meta definida‘ permite al robot planificar rutas eficientes considerando los obstáculos presentes. Además, este modo brinda flexibilidad al usuario, permitiendo la reasignación de nuevas metas con base en el mapa generado hasta el momento. En contraste, el modo ‘exploración‘ posibilita una navegación completamente autónoma, ideal para escenarios en los que no se han definido objetivos específicos o donde el principal propósito es la obtención de información del entorno.

Referencias

Afanasyev, I., Sagitov, A., Magid, E., 2015. Ros-based slam for a gazebo-simulated mobile robot in image-based 3d model of indoor environment. In: Advanced Concepts for Intelligent Vision Systems: 16th International Conference, ACIVS 2015, Catania, Italy, October 26-29, 2015. Proceedings 16. Springer, pp. 273–283.

Alpkiray, N., Torun, Y., KAYNAR, O., 2018. Probabilistic roadmap and artificial bee colony algorithm cooperation for path planning. In: 2018 International Conference on Artificial Intelligence and Data Processing (IDAP). IEEE, pp. 1–6.

Barrau, A., Bonnabel, S., 2015. An ekf-slam algorithm with consistency properties. arXiv preprint arXiv:1510.06263.

Carlone, L., Du, J., Kaouk Ng, M., Bona, B., Indri, M., 2014. Active slam and exploration with particle filters using kullback-leibler divergence. Journal of Intelligent & Robotic Systems 75, 291–311.

Carrillo, H., Reid, I., Castellanos, J. A., 2012. On the comparison of uncertainty criteria for active slam. In: 2012 IEEE International Conference on Robotics and Automation. IEEE, pp. 2080–2087.

Cheein, F. A. A., De la Cruz, C., Carelli, R., Bastos Filho, T. F., 2011. Navegación autónoma asistida basada en slam para una silla de ruedas robotizada en entornos restringidos. Revista Iberoamericana de Automática e Informática Industrial RIAI 8 (2), 81–92.

Durrant-Whyte, H., Bailey, T., 2006. Simultaneous localization and mapping: part i. IEEE Robotics Automation Magazine 13 (2), 99–110. DOI: 10.1109/MRA.2006.1638022

Hentschel, M., Wagner, B., 2010. Autonomous robot navigation based on openstreetmap geodata. In: 13th International IEEE Conference on Intelligent Transportation Systems. pp. 1645–1650. DOI: 10.1109/ITSC.2010.5625092

Hernández Millán, G., Ríos Gonzales, L. H., Bueno López, M., 2016. Implementación de un controlador de posición y movimiento de un robot móvil diferencial. Revista Tecnura 20 (48), 123–136. DOI: 10.14483/udistrital.jour.tecnura.2016.2.a09

Hess, W., Kohler, D., Rapp, H., Andor, D., 2016. Real-time loop closure in 2d lidar slam. In: 2016 IEEE international conference on robotics and automation (ICRA). IEEE, pp. 1271–1278.

Kavraki, L. E., Svestka, P., Latombe, J.-C., Overmars, M. H., 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE transactions on Robotics and Automation 12 (4), 566–580.

Kumar, S., Sikander, A., 2023. A modified probabilistic roadmap algorithm for efficient mobile robot path planning. Journal on Engineering Optimization 55 (9), 1616–1634. DOI: https://doi.org/10.1080/0305215X.2022.2104840

Leung, C., Huang, S., Dissanayake, G., 2006. Active slam using model predictive control and attractor based exploration. In: 2006 ieee/rsj international conference on intelligent robots and systems. IEEE, pp. 5026–5031.

Li, J., Qin, H., Wang, J., Li, J., 2022a. Openstreetmap-based autonomous navigation for the four wheel-legged robot via 3d-lidar and ccd camera. IEEE Transactions on Industrial Electronics 69 (3), 2708–2717. DOI: 10.1109/TIE.2021.3070508

Li, Q., Xu, Y., Bu, S., Yang, J., 2022b. Smart vehicle path planning based on modified prm algorithm. Sensors. MDPI. 22 (17), 6581. DOI: https://doi.org/10.3390/s22176581

Maurović, I., Seder, M., Lenac, K., Petrović, I., 2017. Path planning for active slam based on the d* algorithm with negative edge weights. IEEE Transactions on Systems, Man, and Cybernetics: Systems 48 (8), 1321–1331.

Muñoz-Bañón, M. , Velasco-Sánchez, E., Candelas, F. A., Torres, F., 2022. Openstreetmap-based autonomous navigation with lidar naive-valley-path obstacle avoidance. IEEE Transactions on Intelligent Transportation Systems 23 (12), 24428–24438. DOI: 10.1109/TITS.2022.3208829

Naik, L., Blumenthal, S., Huebel, N., Bruyninckx, H., Prassler, E., 2019. Semantic mapping extension for openstreetmap applied to indoor robot navigation. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 3839–3845. DOI: 10.1109/ICRA.2019.8793641

Qiao, L., Luo, X., Luo, Q., 2022. An optimized probabilistic roadmap algorithm for path planning of mobile robots in complex environments with narrow channels. Sensors. MDPI 22 (22), 8983.

Ren, J., Wu, T., Zhou, X., Yang, C., Sun, J., Li, M., Jiang, H., Zhang, A., 2022. Slam, path planning algorithm and application research of an indoor substation wheeled robot navigation system. Journal Electronics. MDPI 11 (12), 1838.

Rixon Raj, R., Bhuvaneswari, P., 2020. Slam based autonomous navigating object identifier for an indoor environment using 3d-lidar and mobinet. International Journal of Engineering and Techniques.

Salado-Chávez, K. I., Martínez-Hernández, F.-E., Ramírez-Cárdenas, O. D., Arias-Aguilar, J. A., 2023. Implementation of potential fields in a differential mobile robot (turtlebot2). In: 2023 XXV Robotics Mexican Congress (COMRob). IEEE, pp. 62–67.

Sanz, P., 2009. Robotics: Modeling, planning, and control. IEEE Robotics Automation Magazine 16 (4), 101–101. DOI: 10.1109/MRA.2009.934833

Thrun, S., Montemerlo, M., 2006. The graph slam algorithm with applications to large-scale mapping of urban structures. The International Journal of Robotics Research 25 (5-6), 403–429.

Trivun, D., Šalaka, E., Osmanković, D., Velagić, J., Osmić, N., 2015. Active slam-based algorithm for autonomous exploration with mobile robot. In: 2015 IEEE International Conference on Industrial Technology (ICIT). IEEE, pp. 74–79.

Villarreal Onofre, R. L., et al., 2021. Un framework basado en ros para la navegación de robots móviles autónomos. B.S. thesis, Benemérita Universidad Autónoma de Puebla.

Xuexi, Z., Guokun, L., Genping, F., Dongliang, X., Shiliu, L., 2019. Slam algorithm analysis of mobile robot based on lidar. In: 2019 Chinese Control Conference (CCC). IEEE, pp. 4739–4745.