

Date of publication 2024 05, 0001, date of current version 0001 05, 2024.

Digital Object Identifier 10.1109/ACCESS.2017.127128023

Implementación de la Navegación Reactiva con Filtro de Kalman en PuzzleBot

Esteban Padilla Cerdio¹, Karen Cebreros López¹, Naomi Estefanía Nieto Vega¹, Aranza Leal Aguirre¹

¹Escuela de Ingeniería y Ciencias del Tecnológico de Monterrey Campus Querétaro, México. Departamento de Ingeniería en Robótica y Sistemas Digitales

Este trabajo fue apoyado en parte por Manchester Robotics Ltd en colaboración con el Tecnológico de Monterrey Campus Querétaro para la UF TE3003B Integración de Robótica y Sistemas Inteligentes.

RESUMEN Este proyecto se centra en el desarrollo, implementación y evaluación de algoritmos avanzados de navegación y localización para robots móviles utilizando el simulador y el entorno físico implementado. Se exploraron y ajustaron los algoritmos clásicos de navegación reactiva, Bug 0 y Bug 2, además de la integración del sofisticado filtro de Kalman para mejorar la precisión en la estimación de la posición del robot. Se llevaron a cabo extensas pruebas en diversos escenarios, donde se enfrentaron y superaron los retos mediante la implementación de algoritmos especiales para casos excepcionales, como el algoritmo de escape y el algoritmo jump. Los resultados obtenidos revelaron mejoras sustanciales en la eficiencia y precisión de la navegación del robot, destacando la eficacia de la combinación de algoritmos tradicionales y técnicas avanzadas de estimación de estado en entornos simulados de robótica móvil.

PALABRAS CLAVE algoritmos de navegación reactiva, filtro de kalman, gazebo, bug 0, bug 2, localización de robots móviles, evitación de obstáculos, marcadores aruco, simulación de robots, rviz, covarianza de pose, corrección de posición, algoritmos de excepción, algoritmo de escape, algoritmo jump, seguimiento de paredes, navegación autónoma, estimación de estados, robótica móvil, entorno de simulación, navegación precisa.

I. INTRODUCCIÓN

El presente reporte documenta el desarrollo de un sistema de navegación autónomo para el robot móvil Puzzlebot, desarrollado por Manchester Robotics Ltd. Utilizando algoritmos avanzados de control de posición, el objetivo principal de este proyecto es permitir que el robot navegue de manera autónoma hacia una meta, sorteando obstáculos y adaptándose a diferentes entornos.

Este informe está estructurado en varias secciones para proporcionar una comprensión detallada del proyecto. La sección de Objetivos presenta tanto el objetivo general como los objetivos específicos del proyecto, los cuales se han formulado utilizando la metodología SMART para asegurar metas claras y alcanzables. Estos objetivos incluyen la implementación de algoritmos de navegación como Bug 0 y Bug 2, así como la integración del filtro de Kalman y el uso de marcadores ArUco para mejorar la precisión en la localización del robot.

II. OBJETIVOS

El objetivo general del proyecto es lograr que el Puzzlebot se mueva en un entorno físico de pista, alcanzando puntos definidos (Goals) y utilizando marcadores ArUco para corregir su posición y llegar a la meta deseada. Los objetivos específicos incluyen:

1. Implementar y ajustar los algoritmos Bug 0 y Bug 2 en la simulación de Gazebo y en el entorno físico real.
2. Integrar el Filtro de Kalman para la navegación y localización del robot.
3. Crear y definir escenarios de prueba en Gazebo con base en la pista realizada en físico.
4. Analizar el comportamiento de los algoritmos en diferentes configuraciones.
5. Considerar la creciente covarianza alrededor de la pose del robot.
6. Desarrollar algoritmos de excepción para manejar situaciones problemáticas, como el algoritmo jump y el algoritmo de escape.

III. DESARROLLO

La navegación de robots móviles en entornos con obstáculos es un área ampliamente estudiada en la robótica. Los algoritmos Bug 0 y Bug 2 son dos de las estrategias más básicas y conocidas para la navegación reactiva. Estos algoritmos se basan en reglas simples para evitar obstáculos y llegar a un objetivo implementados en numerosos estudios y aplicaciones.

Los algoritmos implementados son preferidos en muchas aplicaciones debido a su simplicidad y efectividad en escenarios estáticos. Sin embargo, tienen limitaciones, como la eficiencia en términos de tiempo y distancia recorrida, y su incapacidad para manejar dinámicamente entornos cambiantes sin modificaciones adicionales.

ESTADO DEL ARTE

A continuación, en el estado del arte presentaremos algunos de los ejemplos más relevantes de la navegación reactiva.

A. Roomba (Aspiradora Robot de iRobot)

Las aspiradoras robot, como las Roomba, se han convertido en un electrodoméstico popular para la limpieza del hogar y son un perfecto ejemplo cotidiano de navegación reactiva. Utilizan sensores para detectar obstáculos y suciedad, ajustando su trayectoria en tiempo real para limpiar eficientemente. Las Roombas ofrecen comodidad y facilidad de uso en entornos domésticos, con un diseño robusto y eficiente. No obstante, están limitadas a tareas específicas como la limpieza y pueden no ser óptimas en entornos muy desordenados. (iRobot Mexico, n.d.)

Algunas de sus ventajas son:

- Comodidad y Facilidad de Uso:** Las Roombas ofrecen una forma conveniente de mantener limpios los espacios sin la necesidad de una intervención humana constante.
- Eficiencia:** Su sistema de navegación reactiva les permite limpiar de manera eficiente, cubriendo toda el área disponible.
- Adaptabilidad:** Pueden maniobrar alrededor de obstáculos y muebles con relativa facilidad, adaptándose a diferentes diseños de interiores.

Por otro lado, algunas de sus desventajas son:

- Limitaciones en la profundidad de la limpieza:** Aunque son eficaces para la limpieza superficial, pueden no ser tan efectivas como las aspiradoras tradicionales para la limpieza profunda.
- Dependencia de la carga de batería:** Su autonomía está limitada por la duración de la

batería, lo que puede requerir múltiples sesiones de limpieza para cubrir áreas grandes.

- Incapacidad para lidiar con obstáculos altos:** Pueden tener dificultades para limpiar debajo de muebles altos o alrededor de objetos muy grandes.

Comparado con nuestra propuesta, las Roombas utilizan un enfoque de navegación reactiva que les permite moverse de manera autónoma y evitar obstáculos en tiempo real mientras limpian un espacio. Nuestra propuesta podría aprovechar la simplicidad y eficiencia de este enfoque para aplicaciones específicas que requieran respuesta rápida a estímulos del entorno. Si bien las Roombas están diseñadas para un propósito específico (limpieza doméstica), nuestra propuesta podría adaptar principios similares de navegación reactiva para situaciones en otros dominios, manteniendo la capacidad de evitar obstáculos y adaptarse a entornos variables.



FIGURA 1. Roomba, la aspiradora robot de la empresa iRobot. Obtenido de: <https://www.irobotshop.mx/categoría-producto/robots>

B. Robots de Amazon (Amazon Robotics)

Los robots de Amazon, se utilizan en los almacenes de Amazon para mover estanterías llenas de productos a estaciones donde los empleados pueden recoger y empaquetar los artículos. Estos robots utilizan una navegación reactiva para moverse por el almacén, evitando obstáculos y otros robots en tiempo real.

Algunas de sus ventajas son:

- Eficiencia operativa:** Mejoran significativamente la eficiencia y velocidad de las operaciones de almacenamiento y recuperación de productos.
- Reducción de errores humanos:** Minimizan los errores en la manipulación y transporte de mercancías.
- Escalabilidad:** Pueden integrarse fácilmente en grandes almacenes y ajustarse a cambios en la disposición del almacén.

Por otro lado, algunas de sus desventajas son:

- Costo inicial:** La implementación de estos sistemas puede ser costosa debido a la inversión en robots y la infraestructura necesaria.
- Dependencia de infraestructura:** Requieren un entorno altamente controlado y marcado con códigos QR o similares para la navegación.
- Limitaciones en flexibilidad:** Pueden tener dificultades para adaptarse a cambios rápidos en el entorno que no se ajusten a su programación o sensores.

Comparado con nuestra propuesta, los robots de Amazon utilizan tecnologías avanzadas de navegación reactiva para operar de manera eficiente en entornos de almacén controlados. Nuestra propuesta puede beneficiarse de las técnicas de detección y evitación de obstáculos utilizadas por estos robots, adaptándolas para aplicaciones en entornos menos estructurados. Mientras los robots de Amazon operan en un entorno optimizado para su uso, nuestro proyecto podría enfocarse en la flexibilidad y adaptabilidad, permitiendo la navegación en una variedad más amplia de contextos, manteniendo la eficiencia y seguridad observadas en los robots de Amazon. (Robotics: Science and Systems, n.d.)



FIGURA 2. Robot inteligente de almacén de Amazon Robotics. Obtenido de: <https://www.amazon.science/research-areas/robotics>

MATERIALES Y MÉTODOS

En cuestión de los materiales tenemos los siguientes:

- Robot:** Se utilizó el Puzzlebot NVIDIA JETSON® Edition.
- Algoritmos de navegación:** Se implementaron los algoritmos Bug 0 y Bug 2 en Python, sin utilizar bibliotecas adicionales aparte de NumPy.
- Escenarios de prueba:** Se utilizó el mundo creado y propuesto por Manchester Robotics en Gazebo ("obstacle_avoidance_X.world").
- Filtro de Kalman:** Se utilizó para mejorar la navegación reactiva del robot permitiendo corregir su posición de acuerdo a los marcadores ArUco.
- Covarianza de la pose:** Se consideró la creciente covarianza alrededor de la pose del robot durante las simulaciones para mejorar la precisión de la navegación.

- Launch files:** Se definieron archivos de lanzamiento específicos para configurar y ejecutar las simulaciones.
- Tiempo de muestreo:** Se definió un tiempo de muestreo adecuado para garantizar la precisión y estabilidad de las simulaciones.
- Entorno simulado:** Gazebo, se utilizó para crear y ejecutar las simulaciones.
- Entorno físico:** Se diseñó y construyó un entorno físico similar a un laberinto en el que se instalaron metas para que el Puzzlebot llegue a ellas.

Por otro lado, es importante resaltar que utilizamos la siguiente arquitectura de nuestro Puzzlebot, como se muestra en la Figura 3.

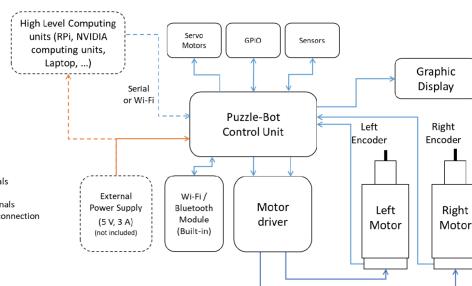


FIGURA 3. Arquitectura del Puzzlebot. Obtenido de: https://github.com/ManchesterRoboticsLtd/TE3003B_Integration_of_Robotics_and_Intelligent_Systems

La unidad de control mostrada en la Figura 3 funciona de la siguiente manera:

- Las unidades de computación de alto nivel ejecutan el software que controla al robot. Este software incluye el algoritmo de navegación, el controlador de motores y el controlador de sensores.
- Los sensores proporcionan información al software sobre el entorno del robot. Esta información incluye la posición del robot, la distancia a los obstáculos y la presencia de objetos.
- El software utiliza la información de los sensores para determinar cómo debe moverse el robot. El software envía comandos a los motores para controlar su movimiento.
- Los motores controlan el movimiento del robot. Los motores giran las ruedas o los brazos del robot para moverlo en la dirección deseada.
- La pantalla gráfica muestra información al usuario sobre el estado del robot y la ubicación de los obstáculos. Esta información puede ser útil para depurar el robot o para monitorizar su progreso.
- El módulo Wi-Fi/Bluetooth permite al robot comunicarse con un ordenador u otro dispositivo. Esta comunicación se puede utilizar para enviar comandos al robot, recibir datos del robot o actualizar el software del robot.

Por otro lado, para la implementación de algoritmos fue muy importante la máquina de estados del algoritmo Avoid Obstacles. Dicha máquina se utiliza para navegar por un entorno con obstáculos y llegar a un objetivo. La máquina de estados consta de tres estados, como se observa en la figura 4:

1. **Stop:** Este estado se activa cuando el robot está en el objetivo o cuando la distancia al obstáculo más cercano (dclosest) es menor que la distancia de parada (dstop). En este estado, el robot se detiene.
2. **Go to goal:** Este estado se activa cuando dcosest es mayor que dstop y el objeto no está detrás del robot. En este estado, el robot se mueve hacia el objetivo.
3. **Avoid Obstacles:** Este estado se activa cuando dcosest es mayor que dstop y el objeto está detrás del robot. En este estado, el robot evita el obstáculo.

Para cada estado, hay una condición de transición que determina cuándo cambia el estado. Las condiciones de transición son las siguientes:

- **Stop:** El robot pasa al estado Stop si está en el objetivo o si dcosest es menor que dstop.
- **Go to goal:** El robot pasa al estado Go_to_goal si dcosest es mayor que dstop y el objeto no está detrás del robot.
- **Avoid Obstacles:** El robot pasa al estado Avoid Obstacles si dcosest es mayor que dstop y el objeto está detrás del robot.

El algoritmo funciona de forma que, supongamos que el robot está en el estado Go_to_goal y se mueve hacia el objetivo. De repente, el robot detecta un obstáculo en su camino. El robot calcula dcosest y determina que es mayor que dstop. Sin embargo, el robot también determina que el objeto está detrás de él. En este caso, el robot pasa al estado Avoid Obstacles y comienza a evitar el obstáculo.

El robot evitará el obstáculo girando a la izquierda o a la derecha. El robot seguirá girando hasta que dcosest sea menor que dstop o hasta que el objeto esté frente a él. Una vez que el robot ha evitado el obstáculo, vuelve al estado Go_to_goal y continúa moviéndose hacia el objetivo.

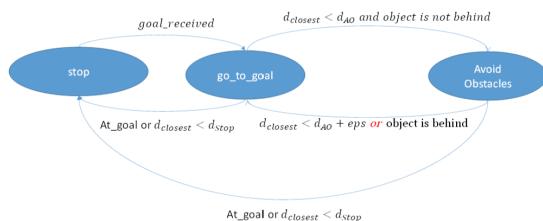


FIGURE 4. Máquina de estados del algoritmo de Avoid Obstacles. Obtenido de: OneNote 2024A Integración de Robótica y sistemas

Asimismo, las ecuaciones del diagrama del algoritmo Avoid Obstacles se utilizan para calcular la distancia al

obstáculo más cercano (dclosest), el ángulo del obstáculo más cercano (thetaclosest) y el error de ángulo (e_theta).

Este algoritmo fue de mucha utilidad para la implementación de los Bug 0 y 2, ya que sus condiciones ayudan a la navegación reactiva del Puzzlebot.

Para la implementación y prueba de los algoritmos de navegación Bug 0 y Bug 2, se utilizaron los siguientes diagramas de bloques, que básicamente describen su funcionamiento, como se muestra en las Figuras 5 y 6.

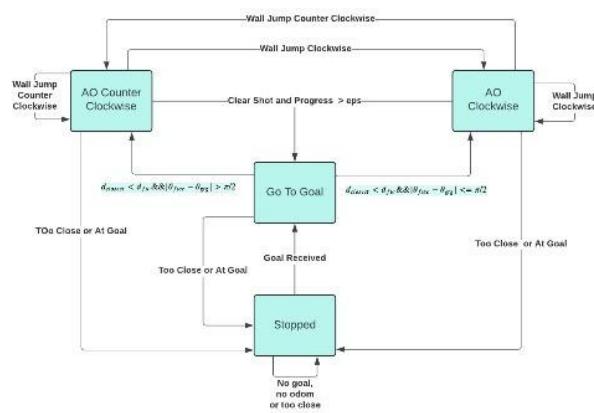


FIGURE 5. Diagrama de bloques con la explicación del algoritmo Bug 0.

El algoritmo Bug 0, como se muestra en la Figura 5, es una estrategia de navegación simple y efectiva para que robots móviles exploren y naveguen entornos desconocidos con obstáculos. Dicho algoritmo funciona de la siguiente manera:

1. El robot comienza en cualquier lugar y se mueve en una dirección aleatoria.
2. Si detecta un obstáculo, gira en la dirección opuesta a la pared.
3. Sigue la pared hasta encontrar una esquina o salida.
4. En una esquina, gira en la dirección opuesta a la pared actual y continúa.
5. Si encuentra una salida, se dirige hacia el objetivo.

Algunas de las ventajas de este algoritmo es que es simple, eficiente y robusto para cumplir su propósito, garantiza llegar al objetivo deseado si hay un camino libre. Por otro lado, algunas de las desventajas es que puede ser ineficiente en entornos abiertos o que puede quedar atrapado entre paredes circulares.

Sin embargo, tenemos un algoritmo que ha demostrado ser más eficiente, el cual es el Bug 2, como se muestra en el diagrama de bloques de la Figura 6.

El algoritmo Bug2, es una variante del algoritmo Bug 0 que mejora su rendimiento en entornos abiertos y reduce la

probabilidad de quedar atrapado en ciclos infinitos. Al igual que el algoritmo Bug 0, el algoritmo Bug2 se basa en la idea de seguir las paredes hasta encontrar una salida o un objetivo. Sin embargo, el algoritmo Bug 2 incorpora algunas mejoras que lo hacen más eficiente y robusto.

El funcionamiento del algoritmo se describe a continuación:

1. El robot se posiciona en cualquier lugar del entorno.
2. El robot comienza a moverse en una dirección aleatoria.
3. Si el robot detecta un obstáculo a su lado, gira en la dirección opuesta a la pared.
4. El robot sigue la pared manteniendo la mano derecha contra la pared.
5. Si el robot encuentra una esquina, gira en la dirección opuesta a la pared actual y continúa siguiendo la pared con la mano derecha contra la pared.
6. Si el robot detecta que ha completado una vuelta completa alrededor de un obstáculo, gira 90 grados en la dirección opuesta a la pared actual y continúa siguiendo la pared con la mano derecha contra la pared.
7. Si el robot encuentra una salida, se dirige hacia el objetivo.

El algoritmo Bug 2 es una estrategia de navegación simple, eficiente y robusta que mejora el rendimiento del algoritmo Bug 0 en entornos abiertos y reduce la probabilidad de quedar atrapado en ciclos infinitos, demostró ser una buena opción para robots que necesitan navegar por entornos desconocidos con obstáculos como el Puzzlebot y el entorno físico creado y que requieren una navegación más eficiente que la que ofrece el algoritmo Bug 0.

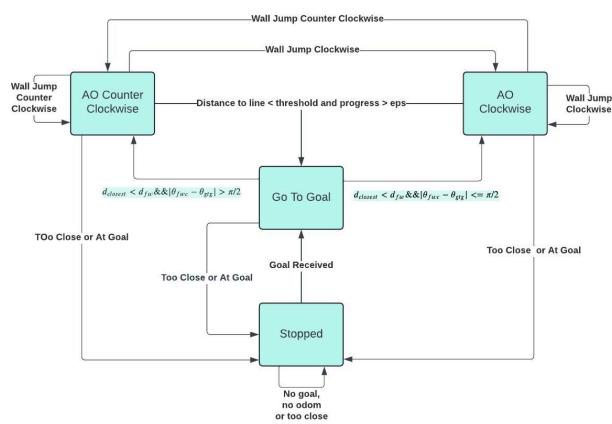


FIGURE 6. Diagrama de bloques con la explicación del algoritmo Bug 2.

Por otro lado, tenemos la implementación del filtro de Kalman para mejorar la navegación, la cual se describe a continuación.

El filtro de Kalman es un algoritmo recursivo que estima el estado de un sistema dinámico a partir de mediciones ruidosas. Se utiliza en una amplia variedad de aplicaciones, como la navegación de robots, el control de procesos y el seguimiento de radar. (MathWorks, 2023)

El filtro de Kalman funciona de la siguiente manera:

1. **Predicción:** El filtro predice el estado del sistema en el siguiente paso del tiempo utilizando las ecuaciones de estado del sistema.
2. **Corrección:** El filtro actualiza la predicción del estado utilizando las mediciones actuales del sistema.

Algorithm 1 EKF Localisation with known data association

```

1. function EKF-Localisation ( $M$ ,  $\mu_{k-1}$ ,  $\Sigma_{k-1}$ ,  $u_k$ ,  $z_{i,k}$ ,  $Q_k$ ,  $R_k$ )
2.  $\hat{\mu}_k \leftarrow h(\mu_{k-1}, u_k)$ 
3.  $H_k \leftarrow \nabla_{s_{k-1}} h(s_{k-1}, u_k)|_{s_{k-1}=\mu_{k-1}}$ 
4.  $\hat{\Sigma}_k \leftarrow H_k \Sigma_{k-1} H_k^T + Q_k$ 
5. if  $z_{i,k}$  corresponds to landmark  $m_i \in M$ 
6.    $\hat{z}_{i,k} \leftarrow g(m_i, \hat{\mu}_k)$ 
7.    $G_k \leftarrow \nabla_{s_k} g(m_i, s_k)|_{s_k=\hat{\mu}_k}$ 
8.    $Z_k \leftarrow G_k \hat{\Sigma}_k G_k^T + R_k$ 
9.    $K_k \leftarrow \hat{\Sigma}_k G_k^T \hat{\Sigma}_k^{-1}$ 
10.   $\mu_k \leftarrow \hat{\mu}_k + K_k(z_{i,k} - \hat{z}_{i,k})$ 
11.   $\Sigma_k \leftarrow (I - K_k G_k) \hat{\Sigma}_k$ 
12. return  $\mu_k, \Sigma_k$ 
  
```

FIGURE 7. Diagrama de bloques con la explicación del algoritmo Bug 2.

Las ecuaciones de estado del filtro de Kalman describen cómo evoluciona el estado del sistema con el tiempo. Se pueden representar mediante la siguiente ecuación:

$$(1) \quad x[k+1] = A * x[k] + B * u[k]$$

donde:

- $x[k]$ es el estado del sistema en el paso de tiempo k
- $x[k+1]$ es el estado del sistema en el paso de tiempo k+1
- A es la matriz de estado
- B es la matriz de control
- $u[k]$ es la entrada de control en el paso de tiempo k

Por otro lado, las ecuaciones de observación describen cómo las mediciones del sistema están relacionadas con el estado del sistema. Se pueden representar mediante la siguiente ecuación:

$$(2) \quad z[k] = H * x[k] + v[k]$$

donde:

- $z[k]$ es la medición del sistema en el paso de tiempo k

- H es la matriz de observación Jacobiana.
- $v[k]$ es el ruido de medición en el paso de tiempo k

Dichas ecuaciones son muy relevantes al momento de seguir el algoritmo del filtro de Kalman, mostrado en la Figura 7, para realizar la corrección de la posición de nuestro robot. De acuerdo con esto, realizamos la implementación del algoritmo del filtro de Kalman, mostrado en la Figura 8.

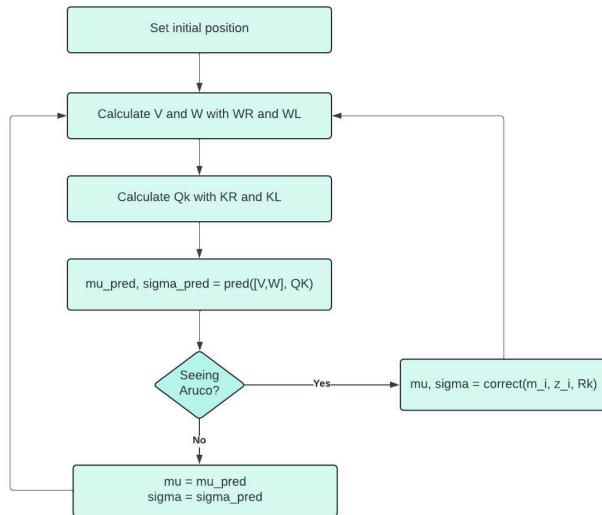


FIGURE 8. Diagrama de bloques con la explicación del algoritmo Bug 2.

Por otro lado, algunas de las aplicaciones más relevantes del filtro de Kalman son:

1. **Navegación de robots:** Los robots utilizan el filtro de Kalman para estimar su posición y orientación a partir de mediciones de sensores como giroscopios y acelerómetros.
2. **Control de procesos:** El filtro de Kalman se utiliza en el control de procesos para estimar el estado de un proceso a partir de mediciones de sensores como la temperatura y la presión.
3. **Seguimiento de radar:** El filtro de Kalman se utiliza en el seguimiento de radar para estimar la posición y la velocidad de un objeto a partir de mediciones de radar.

Por otro lado, para la detección de marcadores ArUco tenemos el algoritmo mostrado en la Figura 9. Para esto lo que realizamos principalmente fue realizar la calibración de la cámara utilizando un tablero de dama para a partir de esto, detectar los marcadores ArUco y posteriormente realizar la transformada de forma que al final nuestro robot es capaz de conocer la distancia y el ángulo al que se encuentra a partir del marcado ArUco identificado.

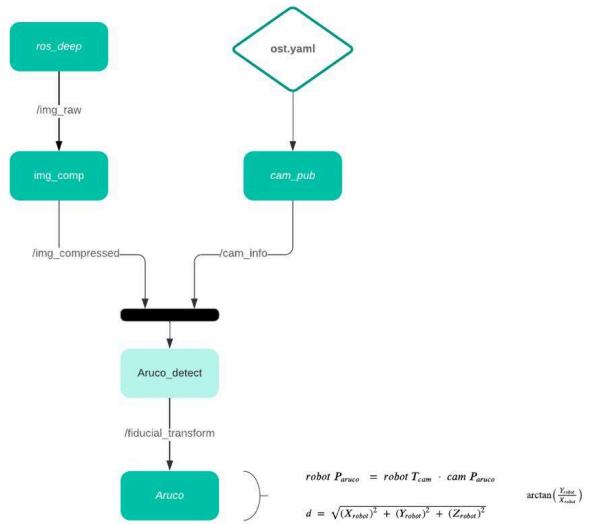


FIGURE 9. Diagrama de bloques con el algoritmo de detección de los marcadores ArUco.

EXTRA: CASOS ESPECIALES

Adicionalmente, nuestro equipo desarrolló excepciones especiales en los algoritmos para ciertos casos especiales que detectamos que causaban conflicto al Puzzlebot durante la navegación autónoma para lograr su objetivo.

Caso 1. "Jump"

El caso de "Jump" se presenta cuando el robot está siguiendo una pared y detecta que el objeto más cercano ("closest_object") parece "saltar" de una pared a otra, generalmente porque ambas paredes están a una distancia similar y muy cercana al robot. Para evitar que el robot se quede atrapado en un ciclo infinito de "saltos" entre las paredes en sentido horario y antihorario, se implementa un mecanismo de prevención de "Jump".

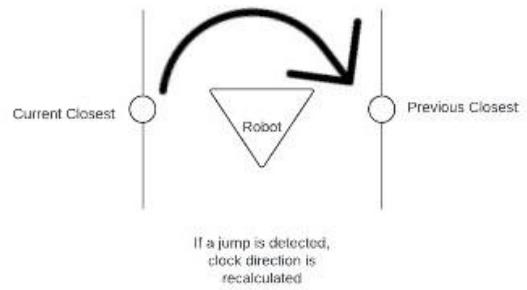


FIGURE 10. Caso especial "Jump".

El mecanismo de prevención de "Jump" se activa cuando se cumplen las siguientes dos condiciones:

1. **Diferencia significativa en el ángulo:** La diferencia entre el ángulo actual del robot y el ángulo anterior es significativa. Esto indica que el robot ha cambiado de dirección de manera

abrupta, lo que podría sugerir un "salto" entre las paredes.

2. **Distancia mínima:** La distancia entre el robot y el objeto más cercano es inferior a 35 cm. Esto confirma que el robot está realmente cerca de ambas paredes.

Caso 2. "Corner Detector"

Cuando el robot llega a una esquina, existe la posibilidad de que siga dando prioridad a mantenerse avanzando por la pared detectada, ignorando la presencia de un objeto cercano a la esquina. Esto podría ocasionar una colisión con el objeto al salir de la esquina. Es por ello que, se implementó una funcionalidad para detectar esto, de manera que el robot cambie su prioridad momentáneamente y continúe avanzando al salir de la esquina.

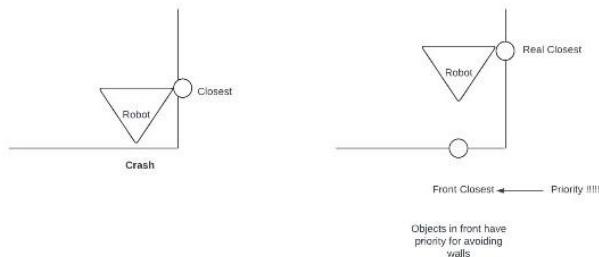


FIGURE 11. Caso especial "Corner detector".

Caso 3. "Escape"

Cuando el robot tiene una trayectoria directa hacia el objetivo, puede quedar atrapado en las paredes si se encuentra demasiado cerca de ellas. Esto podría retrasar o incluso impedir que el robot llegue a su destino.

Para evitar este problema, se implementa un radio de escape en el algoritmo Bug2. Este radio de escape funciona de la siguiente manera: El robot monitorea constantemente su distancia a las paredes y si el robot detecta que está demasiado cerca de una pared (dentro de un radio de escape predeterminado), se activa el modo de escape.

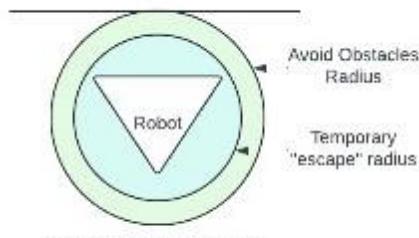


FIGURE 12. Caso especial "Escape".

V. PROJECT MANAGEMENT

En la parte de administración de proyectos, fue muy importante el uso de las siguientes herramientas para el éxito del proyecto.

1. Project Charter

El Project Charter es un documento que establece la autoridad del proyecto y asigna responsabilidades clave. Este documento nos ayudó a definir el alcance, los objetivos y los entregables del proyecto. Asimismo proporciona una dirección clara desde el principio, asegurando que todos los interesados estén alineados con los objetivos del proyecto y comprendan su propósito y alcance. (EAE Business School Barcelona, 2023)

Link: [W Project Charter - Aranchubots.docx](#)

2. RACI Matrix

La matriz RACI es una herramienta de gestión de proyectos que ayuda a definir roles y responsabilidades. Específicamente, define quién es responsable (R), quién es el que aprueba (A), quién debe ser consultado (C) y quién debe ser informado (I) en cada actividad o tarea del proyecto. (Hurtado, 2023) Esto nos ayudó a mejorar la asignación de responsabilidades claras y evitar la confusión sobre quién debe hacer qué en cada etapa del proyecto.

Link: [X Matriz RACI - Aranchubots.xlsx](#)

3. Planned Value and Gained Value

El Valor Planeado (PV) y el Valor Ganado (VG) son métricas utilizadas en la gestión del rendimiento del proyecto. El Valor Planeado representa el presupuesto autorizado para las actividades programadas en un momento dado, mientras que el Valor Ganado representa el valor del trabajo realizado hasta la fecha. (Ambriz, 2008) Estas métricas nos ayudaron a evaluar el progreso del proyecto en comparación con el plan original y a identificar desviaciones tempranas para que se puedan tomar medidas correctivas.

Link:

[X Valor Ganado y Valor Planeado Ultima Actualizació...](#)

4. Gantt Diagram

El diagrama de Gantt es una herramienta visual que muestra las tareas del proyecto en forma de barras horizontales a lo largo de un eje de tiempo. (Teamleader, 2024) Esta herramienta fue útil para planificar, coordinar y rastrear el progreso del proyecto conforme avanzaban las semanas. Nos permitió ver las dependencias entre las tareas,

identificar puntos importantes y gestionar el cronograma de manera efectiva.

Link: [Diagrama de Gantt - Aranchubots Final.xlsx](#)

Link a la carpeta con todos los entregables:

[Resultados Finales](#)

VII. RESULTADOS

Los resultados de las simulaciones y en el entorno real mostraron que ambos algoritmos, Bug 0 y Bug 2, lograron navegar con éxito y evitar obstáculos en la pista creada, asimismo la implementación de los casos especiales fue de gran ayuda y comprobamos que ambos algoritmos funcionan de forma correcta. En múltiples pruebas, el robot logró llegar al objetivo con un máximo de una colisión en la mayoría de los intentos. Asimismo, se observaron diferencias en la eficiencia entre los dos algoritmos, siendo Bug 0 generalmente más eficiente en términos de distancia recorrida y tiempo.

VIII. DISCUSIÓN Y ANÁLISIS DE RESULTADOS

Los resultados presentados revelan un desempeño excepcional de los algoritmos implementados en el proyecto. Con un promedio de error de tan solo 12 cm para el algoritmo Bug 0 y 20 cm para el Bug 2, estos resultados confirman la precisión y eficacia de ambos en la navegación del robot en entornos reales, como se muestra en la Figura 13, lo que indica una alta capacidad de los algoritmos para mantener una trayectoria precisa hacia los objetivos definidos. Siendo el algoritmo con mejor resultado el Bug 0.

Uno de los resultados más destacados es la ausencia de colisiones en todos los intentos realizados. Esto sugiere que los casos especiales implementados, como el algoritmo de escape y el algoritmo jump, fueron efectivos para evitar situaciones potencialmente peligrosas y garantizar la seguridad del robot durante la navegación. Esta característica es crucial para la aplicabilidad práctica de los algoritmos en entornos reales, donde la prevención de colisiones es una prioridad fundamental.

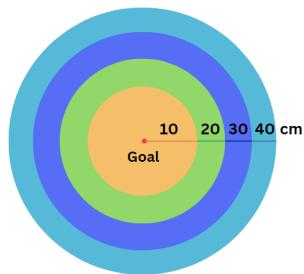


FIGURE 13. Estadísticas obtenidas a partir de los experimentos realizados en el PuzzleBot y el margen de error disponible en cada meta

establecida. Las posiciones en X y Y de las metas se especifican en la tabla.

Además, la diferencia en el error entre Bug 2 y Bug 0 podría atribuirse a la naturaleza más robusta y adaptable de este último algoritmo. Aunque ambos lograron mantenerse dentro del umbral de error establecido, la mayor eficiencia del Bug 0 en términos de distancia recorrida y tiempo empleado puede ser un factor a considerar en la elección del algoritmo en diferentes aplicaciones.

En conclusión, los resultados obtenidos respaldan la efectividad y confiabilidad de los algoritmos de navegación implementados, demostrando su capacidad para lograr una navegación precisa y segura en entornos simulados de robótica móvil. Estos resultados positivos proporcionan una base sólida para futuras investigaciones y aplicaciones en el campo de la robótica autónoma, con el potencial de mejorar la eficiencia y la seguridad en una variedad de entornos y situaciones.

IX. CONCLUSIONES

En conclusión, nuestros resultados del proyecto reflejan el éxito en la implementación y evaluación de los algoritmos de navegación y localización para robots móviles en el simulador Gazebo y en el entorno físico. Los casos especiales que se incorporaron adicionalmente demostraron comportamientos esperados por el robot y fueron de gran ayuda para evitar colisiones o pérdida en la navegación, lo cual enriqueció la capacidad de navegación en entornos desafiantes. Específicamente, la introducción de algoritmos para situaciones excepcionales, como el algoritmo de escape y el algoritmo jump, resultaron en una notable eficiencia y confiabilidad en la navegación del robot.

Además, el algoritmo Bug 0 demostró ser el que nos dió una mayor eficiencia en los entornos implementados, evidenciando su capacidad para resolver de manera efectiva situaciones complejas y dinámicas. Algo importante fue que, en la mayoría de los casos, el sistema logró evitar colisiones, lo cual fue posible gracias a la inclusión estratégica de los casos especiales diseñados para abordar escenarios potencialmente peligrosos.

Finalmente, los resultados obtenidos respaldan la efectividad y robustez de los algoritmos implementados, demostrando su capacidad para realizar una navegación precisa y segura en entornos simulados de robótica móvil. Este éxito sienta una base sólida para futuras investigaciones y aplicaciones en el campo de la robótica autónoma.

A continuación se presentan las conclusiones individuales del equipo:

Goal	x	y
initial position	2.3	1.04
Goal 1	0.50	1.55
Goal 2	1.38	1.03
Goal 3	2.05	0.35
Goal 4	2.97	0.40
Goal 5	0.50	1.55

- **Esteban Padilla Cerdio**

Este reto concluyente de la carrera de robótica brindó una variedad de desafíos interesantes y complicados, que requirieron de la unificación de todo nuestro conocimiento recaudado durante este semestre y los anteriores. A través del trabajo constante, la experimentación, y la prueba y error, logramos concluir nuestro proyecto con resultados positivos. Para esto, tuvimos que innovar nuestros algoritmos para adaptarlos a la circunstancia y a los retos específicos que enfrentamos.

Las estadísticas resultantes demostraron que nuestra solución fue acertada, estable y robusta, logrando llegar a la meta con un error menor a las dimensiones del robot. Esto se logró a través de la correcta inclusión del filtro de Kalman, las lecturas de los marcadores Aruco y los algoritmos mejorados de Bug0, Bug2 y Bug RayTracing.

En conclusión, el proyecto se llevó a cabo con éxito, y nos llevamos aprendizajes importantes en el área de robótica y manejo de proyectos.

- **Karen Cebreros López:**

A lo largo de este proyecto, tuvimos la oportunidad de aprender más a fondo sobre la navegación autónoma y sobre los algoritmos pueden ser usados en ésta, como el algoritmo de evasión de obstáculos, los de corrección y predicción, o el del detector de marcadores Aruco. Todo esto tomando en cuenta el uso de herramientas y sensores con los que el Puzzlebot ya cuenta, como el lidar y la cámara.

A pesar de que se nos presentaron diversos retos para la elaboración de la solución de nuestro proyecto, pudimos superarlos y completar con éxito todos los objetivos planteados por el socio formador.

Dichos retos fueron superados a través de la experimentación, investigación y observación, que en lo personal, se me hizo una parte no solo muy interesante para el desarrollo del proyecto, sino que sin duda algo muy necesario, para la correcta elaboración de los algoritmos.

Para la parte de resultados, en el caso de nuestro equipo, tras hacer una serie de pruebas en la pista propuesta por el salón, pudimos comprobar que para ésta, el algoritmo que mostró mayor eficiencia fue el Bug0, al presentar un promedio de error de la distancia menor, al del Bug2.

No obstante, creemos que el desempeño del robot podría mejorar, si llegásemos a utilizar y probar alguno de los otros algoritmos que desarrollamos, como el de RayTracing.

Finalmente, creo importante mencionar que considero que, gracias a estos nuevos conocimientos adquiridos a lo largo de este bloque, junto con los que ya habíamos adquirido en

los bloques anteriores de robótica, pudimos solucionar exitosamente el reto de este bloque.

Sin duda creo que todo lo visto a lo largo de estos años en robótica, tendrán aplicaciones muy útiles en proyectos de la vida real a futuro.

- **Naomi Estefanía Nieto Vega**

En conclusión, uno de los mayores retos enfrentados durante este reto fue el ajuste de los algoritmos para su correcto funcionamiento, en el que, finalmente se concluyó que el más eficiente fue el Bug 2, ya con todas las mejoras implementadas previamente para los casos especiales. Sin embargo, una dificultad específica que enfrentamos fue encontrar el balance correcto entre los parámetros que determinan cómo reacciona nuestro robot a los obstáculos para evitar comportamientos extraños o colisiones, pero algo que fue muy útil fue ajustar las tolerancias, ya que así teníamos el comportamiento deseado. Finalmente, a pesar de dichos retos enfrentados, el reto se logró concluir de manera satisfactoria.

Algunas oportunidades futuras de trabajo de las que me di cuenta, fue que tiene mucha aplicabilidad en nuestra vida real, como las aspiradoras robot. Aunque la robótica aún es un campo de estudio que tiene mucho por investigar y estudiar, en la actualidad ya podemos ver los avances significativos que trae a nuestras vidas. En nuestro caso, el PuzzleBot represa una implementación a escala, que es la base para muchas aplicaciones en la vida real.

- **Aranza Leal Aguirre:**

Los algoritmos de navegación implementados fueron eficientes pues el Puzzlebot pudo navegar de manera autónoma, evitando obstáculos y logrando corregir su trayectoria con el uso de Filtro de Kalman y Arucos. Además de estas implementaciones, logramos implementar soluciones innovadoras como la función de Jump o Escape, las cuales nos funcionaron para situaciones específicas, y magnificaron la eficiencia del recorrido cada vez.

Si bien el proyecto tiene grandes oportunidades de optimización como implementar SLAM y Navigation Stack, o bien, mejoras de hardware para mejorar el rendimiento de los recorridos; pudimos completar el reto con éxito y de manera efectiva.

ANEXOS

En este link podemos encontrar los archivos del código fuente y todos los archivos necesarios para compilar el proyecto.

- **Link al GitHub con todo el código fuente**

https://github.com/Esteb37/aranchubots_autonomousnavigation_puzzlebot/tree/master

Posteriormente, en este link podemos encontrar el video de la presentación final ante Manchester Robotics Ltd.

- **Link al video de la presentación final**

<https://youtu.be/cKHk5dFEsmY>

Además, en este link podemos encontrar las medidas y especificaciones de la pista ganadora realizada en físico.

- **Diagrama de la pista propuesta (ganadora)**

<https://docs.google.com/document/d/e/2PACX-1vRMPvz26VrfILOSPOYyBEG8Hc98UptnHO3Lr7vv9J5eTl0YLChE-1dyGIMHNCg4EAyyeLK56TAIdA4/pub>

Finalmente, en este último enlace podemos encontrar todos los archivos relacionados con la administración del proyecto.

- **Project Management**

https://drive.google.com/drive/folders/1_ALujGwsGWpsrotL1dV_P3dtM46EZy9?usp=sharing

AGRADECIMIENTOS

Nos complace expresar nuestro más profundo agradecimiento a nuestro valioso socio, Manchester Robotics, por su inquebrantable dedicación y valioso apoyo a lo largo de nuestro recorrido. Valoramos enormemente las innumerables horas invertidas en nuestras clases y la generosidad con la que compartieron su vasto conocimiento y experiencia.

En particular, deseamos extender un agradecimiento especial a los distinguidos profesores Dr. Jesús Arturo Escobedo Cabello, Dr. Josué González García, M en C. Benigno Muñoz Barrón, PM. Pablo Estefano Arroyo Garrido y M. Sc. Juan Manuel Ledesma Rangel de la UF TE3003B. Integración de Robótica y Sistemas Inteligentes. Sus aportes trascendentales han dejado una huella importante en nuestro desarrollo y comprensión del reto final.

REFERENCIAS

[1]Manchester Robotics Ltd. (2023). Obstacle Avoidance Worlds. Retrieved from https://github.com/ManchesterRoboticsLtd/TE3003B_Integration_of_Robotics_and_Intelligent_Systems/tree/main/Week%205/Challenge

[2]Demonstrating Large-Scale package Manipulation via learned metrics of pick success | Robotics: Science and Systems. (n.d.). Robotics: Science and Systems. <https://roboticsconference.org/program/papers/023/>

[3]Aspiradoras Roomba – iRobot Mexico. (n.d.). <https://www.irobotshop.mx/categoría-producto/robots/aspiradoras-roomba>

[4]Robotics, S. D. V. A. (2023, October 19). Amazon anuncia dos nuevas formas de utilizar robots para ayudar a los empleados a realizar entregas más rápidas. ES About Amazon.

<https://www.aboutamazon.es/noticias/innovacion/amazon-anuncia-dos-nuevas-formas-de-utilizar-robots-para-ayudar-a-los-empleados-a-realizar-entregas-mas-rapidas>

[5] 2024A Integración de Robotica y Sistemas. (2024) https://tecmx-my.sharepoint.com/:r/personal/arturo_escobedo_tec_mx/Documents/Class%20Notebooks/2024A%20Integraci%C3%B3n%20de%20rob%C3%B3tica%20y%20sistemas?d=wecdd37767ab849bb9d2fd2509773515e&csf=1&web=1&e=61kycv

[6] Filtros kalman. (n.d.). MATLAB & Simulink. <https://la.mathworks.com/discovery/kalman-filter.html>

[7] EAE Business School Barcelona (2023, September 4). ¿Qué es un project charter? Retos En Supply Chain | Blog Sobre Supply Chain. <https://retos-operaciones-logistica.eae.es/que-es-un-project-charter/>

[8] Hurtado, J. S. (2023, May 19). Matriz RACI ¿Para qué sirve y cómo hacerla? Thinking for Innovation. <https://www.jebschool.com/blog/matriz-raci-para-que-sirve-y-como-hacerla-agile-scrum/>

[9] Ambriz A., Rodolfo (13 agosto, 2008). La gestión del valor ganado y su aplicación. (n.d.). <https://www.pmi.org/learning/library/es-las-mejores-practicas-de-gestion-del-valor-ganado-7045>

[10] Teamleader. (2024, April 16). ¿Qué es y para qué sirve un diagrama de Gantt? Teamleader. <https://www.teamleader.eu/es/blog/diagrama-de-gantt>