Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Puebla

TE3003B.501

Integración de robótica y sistemas inteligentes

Reto Semanal 6 | Manchester Robotics

Frida Lizett Zavala Pérez A01275226

Diego Garcia Rueda A01735217

Alejandro Armenta Arellano A01734879

17 de Mayo del 2024

Índice

Navegación reactiva **4**
Sensor Lidar modelo rplidar A1 **5**
Algoritmos para evasión de obstáculos **6**
Bug 0 6
Bug 1 7
Solución del problema **10**
Resultados **10**
Conclusiones **11**
Referencias **11**

Resumen

Este reto se centra en el desarrollo e implementación de un sistema de navegación reactiva para un robot móvil, usando el sensor LIDAR RPLidar A1 y los algoritmos Bug 0, Bug 1 y Bug 2. Se genera la simulación del funcionamiento de los algoritmos en Gazebo probandolo en diversos escenarios.

Objetivos

Desarrollar e implementar un sistema de navegación reactiva para el Puzzlebot usando el sensor LIDAR RPLidar A1, los algoritmos Bug 0, Bug 1 y Bug 2 en el entorno de simulación Gazebo. Se busca evaluar el rendimiento de cada algoritmo en los diversos mundos proporcionados por Manchester Robotics, poniendo a prueba su seguridad y robustez al llegar al punto meta, esperando que los resultados obtenidos permitan analizar el comportamiento del sistema de navegación y los diferentes bugs y entender la toma de decisiones que emplea cada uno.

Introducción

En la robótica móvil, la navegación autónoma se ha convertido en un campo de estudio importante para el desarrollo de sistemas inteligentes capaces de desplazarse en entornos complejos de manera segura. La navegación reactiva ha surgido como una estrategia de movimiento para que los robots tengan la capacidad de reaccionar en tiempo real a obstáculos presentes en su entorno, permitiendo una navegación robusta y adaptable a cambios dinámicos.

Navegación reactiva

La navegación reactiva en la robótica móvil permite a los robots navegar en un escenario de manera autónoma en tiempo real. En este caso en lugar de generar previamente una planeación de ruta completa, se hace uso de la navegación reactiva, donde las decisiones de movimiento se ejecutan en función de la información que se recibe al momento del entorno a través de sus sensores.

Esto hace que la navegación reactiva resulte muy eficiente para entornos dinámicos e impredecibles, dónde una planeación preestablecida podría no funcionar positivamente.

Los principios básicos de la la navegación reactiva son los siguientes:

- Percepción del entorno: A partir de sensores, el robot recopila información sobre su entorno, tales como la detección de obstáculos, límites y la ubicación del objetivo.
- Toma de decisiones: Se procesa la información sensorial y se hace uso de algoritmos para tomar decisiones sobre el movimiento, tales como los algoritmos bug.
- Acción: El robot envía las instrucciones a sus motores para moverse de acuerdo a las decisiones tomadas.

Existen diferentes tipos de navegación reactiva, los dos grupos principales son la navegación basada en comportamientos y la navegación basada en el potencial de campo. La navegación basada en comportamientos se basa en la navegación por comportamientos simples, cómo seguir una pared, evitar obstáculos, o moverse hacia un objetivo, los robots que usan este tipo de navegación seleccionan y ejecutan comportamientos en función de la situación actual. Por otro lado, la navegación basada en el potencial de campo usa este para representar el entorno del robot. Los valores del campo de potencial indican que tan deseable es la posición en la

que se encuentra el robot, para así moverse a posiciones con valores de potencial más altos, evitando espacios con obstáculos y por lo tanto valores de potencial bajos.

Las ventajas de la navegación reactiva son su capacidad de adaptabilidad, ya que permite a los robots adaptarse a entornos de forma rápida y eficaz, su robustez, ya que los robots que usan navegación reactiva son menos propensos a fallar debido a errores en el mapeo del terreno, y la eficiencia, ya que desde el punto de vista computacional esta puede ser muy adecuada para robots con recursos limitados. Sin embargo también cuenta con algunas desventajas como la falta de planificación global de su entorno, lo que puede dificultar la navegación en entornos complejos, igualmente los robots pueden quedar atrapados en bucles ya que comienzan a repetir patrones de movimiento, no anticipa obstáculos futuros, igualmente tiene dependencia de la percepción lo que lleva a decisiones incorrectas, además del ruido que pueden tener los sensores.

Para implementar la navegación reactiva de una manera más óptima y mejorar considerablemente la ruta y movimiento de los robots, se puede combinar con planificaciones a largo plazo, procesamiento de datos robusto, o aprendizaje automático, además de añadir mecanismos para evitar bucles.

Sensor Lidar modelo rplidar A1

La tecnología LIDAR es el acrónimo del inglés "Light Detection and Ranging o Laser Imaging Detection and Ranging" y determina la distancia desde el emisor láser a un objeto o superficie utilizando un láser modulado por pulsos. La distancia al objeto se determina midiendo el tiempo de retraso entre la señal emitida y su detección a través de la señal reflejada. Es una tecnología que ofrece una extrema precisión en la medición de la distancia

en todo tipo de aplicaciones. Para este bloque se cuenta con el sensor LIDAR modelo rplidar A1, éste sensor LIDAR tiene una rotación constante de 360 grados que permite obtener un mapa de distancias de todo su alrededor, hasta 2000 muestras por segundo (5.5Hz) y una distancia de medición de hasta 6 metros. El ángulo de apertura es de 1 grado aproximadamente y tiene un margen de error de menos de 1% de la distancia medida.

Se puede configurar la velocidad de rotación por software mediante PWM. Dispone de un conector que facilita su uso y donde se pueden acceder a los diferentes pines TX/RX, señal PWM, alimentación (5V) y GND.

Algoritmos para evasión de obstáculos

En el ámbito de algoritmos de evasión de obstáculos el conjunto denominado"*BUG"* son los más utilizados debido a su gran capacidad de adaptación y eficacia, esto porque se basan en el principio fundamental en el que si un robot encuentra un obstáculo en su camino hacia el objetivo, puede rodear el obstáculo hasta encontrar una ruta clara hacia el objetivo.

En el conjunto "*BUG"* existen 3 variaciones principales donde el enfoque principal son la toma de decisiones posteriores a la detección de obstáculos como se describe a continuación.

Bug 0

Es el algoritmo más fácil de navegación reactiva , consiste en mover el robot directamente al objetivo deseado y si en el trayecto encuentra un obstáculo , el robot se verá forzado a rodear el contorno hasta que pueda retomar su camino. El algoritmo bug 0 no garantiza completar la trayectoria debido a que puede encontrarse atrapado en algunas decisiones donde es imposible alcanzar el punto final. Su algoritmo es el siguiente:

Entrada: Un robot en forma de punto con un sensor de contacto

Salida: Una trayectoria a G, sólo si existe.

1. mientras Siempre hacer

2. repetir moverse en línea recta con dirección a G

3. hasta G es alcanzada

4. si G es alcanzada entonces salir con éxito

5. repetir seguir la frontera

6. hasta G es alcanzada o es posible dirigirse hacia G

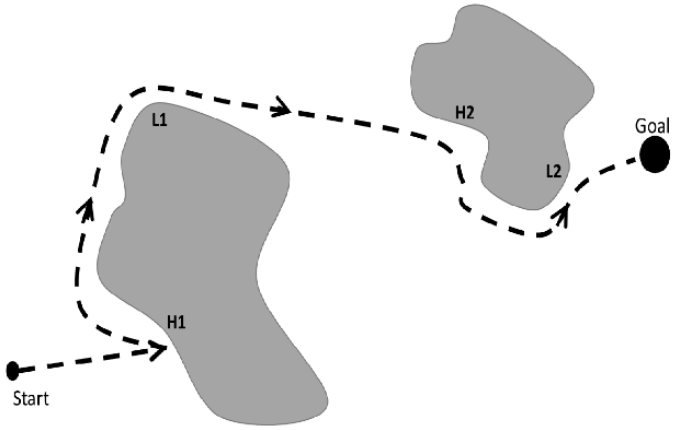


Fig 1. Bug 0

Bug 1

Este algoritmo es una evolución del Bug 0 pues agrega una función en la que una vez terminado de rodear el contorno del obstáculo el robot regresa al punto más cercano al punto deseado mientras rodeo el contorno generando que este algoritmo sea más eficiente al llegar al punto deseado , sin embargo aún es posible que se encuentre en situaciones donde no puede alcanzar el objetivo. El algoritmo es el siguiente:

Entrada: Un robot en forma de punto con un sensor de contacto

Salida: Una trayectoria a G o la conclusión de que tal trayectoria no existe.

1. L0 ← S; i ← 1

2. mientras Siempre hacer

3. repetir moverse en línea recta desde Li−1 hacia G

4. hasta G es alcanzada o un obstáculo es encontrado en Hi

5. si G es alcanzada entonces salir con éxito

6. repetir seguir la frontera

7. hasta G es alcanzada o Hi es reencontrado

8. Determinar el punto Li con la distancia más corta hacia G

9. Dirigirse hacia Li

10. si al moverse en línea recta desde Li hacia G un obstáculo es encontrado y Li == Hi

11. entonces concluir que G no es alcanzable y salir con fallo

12. sino i ← i + 1

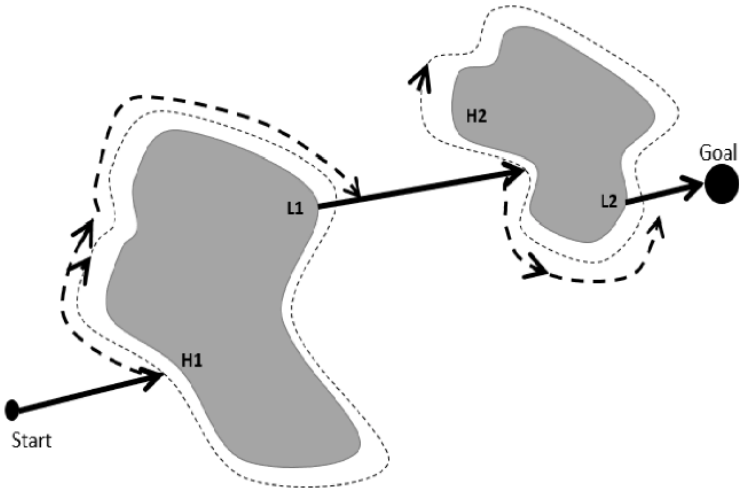


Fig 2. Bug 1

Bug 2

Este algoritmo es una evolución con respecto al Bug 1 debido a la introducción de tangentes para que posteriormente de encontrar un obstáculo el robot busca la tangente más cercana al obstáculo permitiendo encontrar la ruta más eficiente alrededor del obstáculo evitando caer

en situaciones en las que los anteriores algoritmos si fallaran. Su pseudocódigo es el siguiente:

Entrada: Un robot en forma de punto con un sensor de contacto

Salida: Una trayectoria a G o la conclusión de que tal trayectoria no existe.

1. L0 ← S; i ← 1

2. mientras Verdadero hacer

3. repetir moverse sobre la línea-m desde Li−1 hacia G

4. hasta G es alcanzada o un obstáculo es encontrado en Hi

5. si G es alcanzada entonces salir con éxito

6. repetir seguir la frontera

7. hasta

(a) G es alcanzada o

(b) Hi es reencontrado o

(c) la línea-m es reencontrada en un punto P tal que P ≠ Hi , d(P, G) < d(Hi , G)y la línea entre P y G no cruza el obstáculo actual en P

8. si G es alcanzada entonces salir con éxito

9. sino Hi es reencontrado entonces concluir que G no es alcanzable y salir con fallo

10. sino Li ← P; i ← i + 1

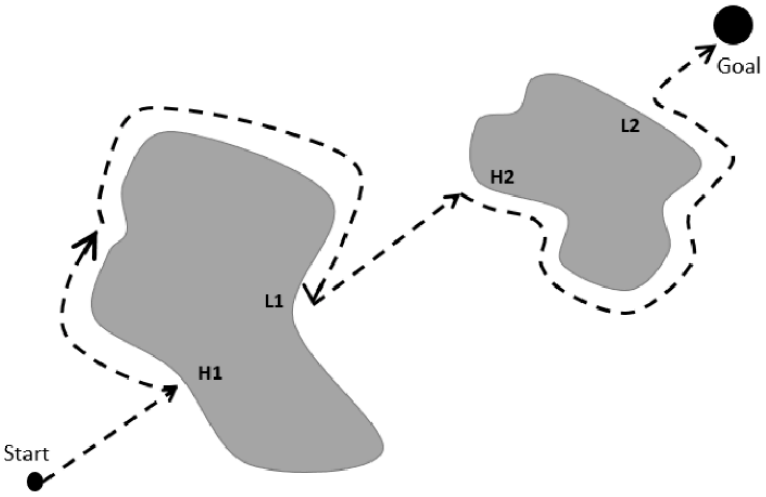


Fig 3. Bug 2

Solución del problema

Para la solución del reto semanal se desarrolló una nueva arquitectura de ROS implementando en nodos separados cada uno de los algoritmos Bug. Para la simulación de este reto se utilizó el paquete desarrollado por *Manchester Robotics* donde se simulaba el *Puzzlebot* en un entorno de Gazebo, el entorno simulado se puede observar en la figura 4.

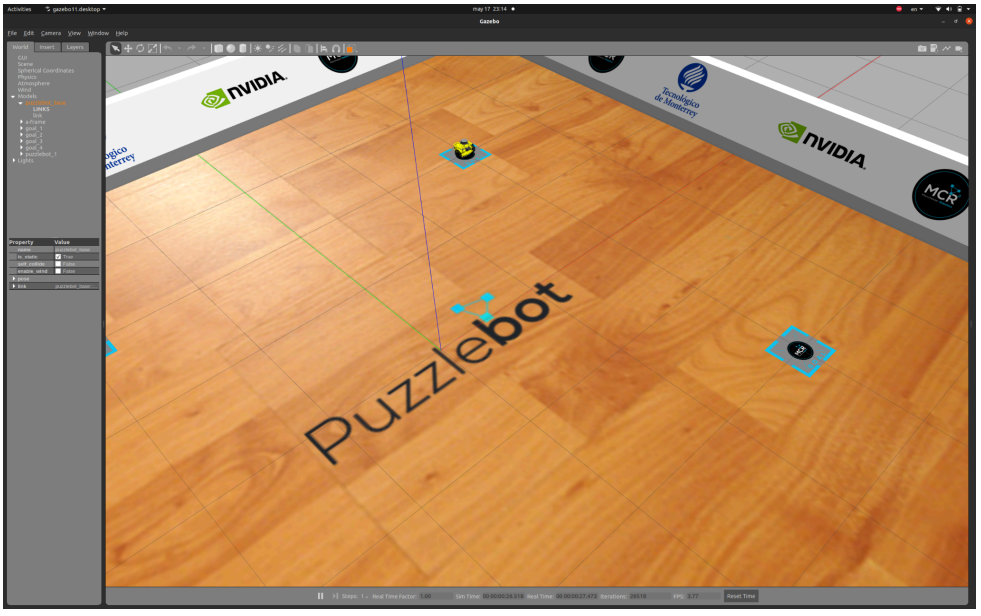
onada po 

Fig 4. Simulación proporcir Manchester Robotics

Cada uno de los nodos se conecta en este caso al topico */puzzlebot_1/wl* y *puzzlebot_1/wr* tópicos correspondientes a la simulación de Gazebo, es con estos tópicos que mediante el nodo *localisation* podemos generar la odometría que genera el sistema de retroalimentación para el controlador. Un ejemplo de esta arquitectura se puede observar en la figura (5).

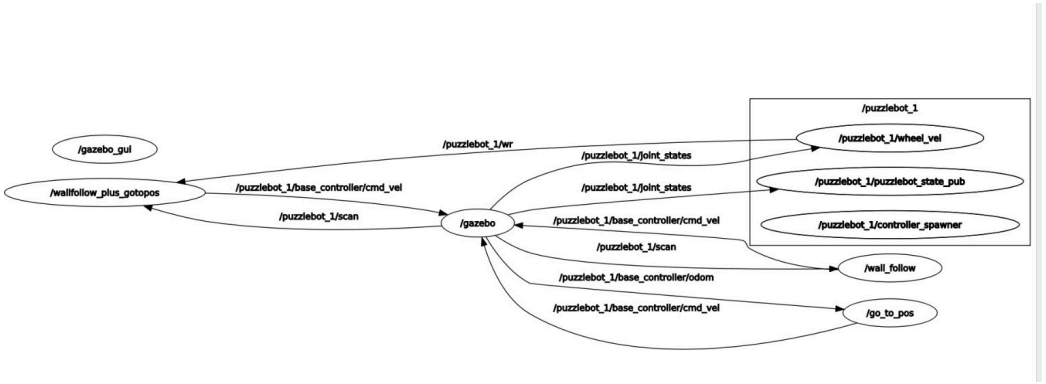


Fig 5. Arquitectura ROS acoplada a algoritmos Bug.

La implementación del Lidar para este reto orillo al equipo a modificar ciertas partes de la arquitectura desarrollada a lo largo de las semanas anteriores.

La nueva arquitectura la podemos convertir a un sistema de estados, estos estados dependen de la condición en la que se encuentre el *Puzzlebot.* Se agregan 2 nuevos nodos: *wall_follow* y *go_to_position*. A continuación se detallan las funcionalidades de estos nodos:

- ***wall_follow:*** Nodo encargado de seguimiento de la superficie del obstáculo. Para la detección de esto se dividen en regiones los datos captados por el LiDAR en 5 regiones (las medidas del lidar se dividen en 5 secciones de 72 elementos cada una, conformando asi los 360 datos enviados). Se establece una distancia respecto al obstáculo que funciona como threshold, en este caso es de 0.3 metros, si esta distancia o una menor es detectada en alguna de estas 5 zonas es que se define un estado de acción. Este estado es un valor global que es utilizado en el nodo correspondiente del algoritmo Bug.

- ***go_to_position:*** Mediante funciones como *go_straight, correct_heading*, etc. es que se calcula el error de posición, para obtener la distancia se utiliza una formula de pendiente que calcula el componente x,y de la posición actual y de la posición objetivo , mientras que el ángulo se calcula mediante el arcotangente igualmente utilizando el componente en x,y de la posición actual y de la posición objetivo. Para esto se utiliza la dinámica de estados que definirá la velocidad lineal y angular mediante un mensaje de tipo *Twist.* Este mensaje es el que genera el mensaje */puzzlebot_1/base_controller/cmd_vel* que genera la retroalimentación y actua como el controlador para la simulación en Gazebo.

Para la selección del bug correspondiente dependen del archivo *.launch* que define los 2 nodos anteriormente explicados con su propio nodo del algoritmo Bug que defines lo s estados que seguirá esta arquitectura.

Resultados

Video de la simulación del Bug 0

https://youtu.be/wc7GbBg0xTs

Video de la simulación del Bug 2

https://youtu.be/X-mSBan4WS0

Aspectos a mejorar:

En el desarrollo del proyecto se encontraron ciertos errores en la implementación de los algoritmo Bug, igualmente se intentó implementar estos algoritmos en el *Puzzlebot* creando una version analoga a la arquitectura mostrada en la figura (5) modificando los tópicos de comunicación para realizar las conexiones con el robot físico. En este proceso existieron complicaciones, siendo la primera el fallo del tópico */scan* que en diversas ocasiones no se

llegaba a generar correctamente. Adicionalmente al momento de ejecutar el Bug 0, se llegó a mover el robot a una posición marcada ligeramente relacionada a la posición deseada, sin embargo el error de posición era bastante notorio, además, al agregar obstáculos en la trayectoria no eran identificados por el sensor LiDAR lo que generaba colisiones con él. Se tiene la hipótesis que el tópico aún no funciona correctamente, o el controlador puede llegar a ser diferente al obtenido y utilizado en la simulación. Se esta trabajando aun en la solución de estos problemas.

Conclusiones

La navegación reactiva es una herramienta muy útil en la robótica autónoma, ya que permite a los robots navegar de forma segura y eficiente en entornos dinámicos y complejos. Tiene diversas cualidades tales como su simplicidad, adaptabilidad y robustez que la convierten en una buena opción para diversas aplicaciones.

Tras probar los algoritmos en los mundos de Gazebo, se muestra que sí bien los tres algoritmos son capaces de generar la trayectoria de manera correcta, en algunas situaciones es posible que no lleguen y tomen decisiones equivocadas, esto es debido a que el desempeño de los algoritmos depende de la posición inicial en la que se encuentre el robot, lo que interfiere con la toma de decisiones posteriores, ya que los valores que vaya percibiendo el sensor pueden variar. De manera general, hasta el momento en esta implementación el Bug 2 es el que muestra un comportamiento más consistente, además de que su lógica es más sencilla. Se comenzaron a evaluar los Bugs en el Puzzlebot para obtener resultados más reales, y se propone como tarea posterior para mejorar el funcionamiento, desarrollar un sistema que combine diferentes algoritmos en conjunto con los controladores, para así poder generar un sistema de navegación más robusto y preciso.

Referencias

Martínez, C. C., Badillo, I. A., Pimentel, J. J. A., Pérez, E. C., Acevedo, F. A., & Rosales, L. A. M. (2016). Sistema de Navegación Reactiva Difusa para Giros Suaves de Plataformas Móviles Empleando el Kinect. https://www.redalyc.org/journal/5122/512253114004/html/

Serna, M. (2018) Navegación de robots móviles utilizando los algoritmos bug extendidos. Retrieved from: https://hdl.handle.net/20.500.12371/7420.