Universidad de La Coruña

Facultad de Informática

Mapeado y Localización:

Arquitectura híbrida en simulador Webots

Xian Priego Martín
Samuel Otero Agraso

16 de abril de 2024

1 Introducción

La robótica es un campo en constante evolución que busca desarrollar sistemas autónomos capaces de interactuar con su entorno de manera inteligente y adaptativa. En este contexto, la práctica propuesta se centra en la implementación de una arquitectura híbrida para el robot Khepera IV, utilizando el simulador Webots, con el objetivo de llevar a cabo tareas de mapeo, localización de objetos y planificación de trayectorias.

El robot Khepera IV, representado por la instancia `self.robot = Robot()`, será el protagonista de esta práctica. Este modelo, conocido por su versatilidad y precisión, será utilizado para explorar un entorno discreto dividido en celdas de 25x25 cm, con dimensiones totales de 12x12 celdas. La tarea principal consistirá en mapear este entorno de manera eficiente, localizar un objeto ubicado junto a una pared en cada búsqueda y planificar una trayectoria para retornar al punto de partida.

La práctica se divide en dos etapas fundamentales:

1. **Etapa de Exploración:** En esta fase inicial, el robot se dedicará a explorar el entorno con el fin de crear un mapa detallado que refleje las características y obstáculos presentes en el área. Durante esta etapa, se emplearán sensores y algoritmos de navegación autónoma para garantizar una exploración completa y precisa del entorno.

2. **Etapa de Búsqueda y Retorno:** Una vez finalizada la etapa de exploración, el robot iniciará la búsqueda del objeto localizable, el cual estará siempre adyacente a una pared. Tras encontrar el objeto, se procederá a planificar una trayectoria óptima que permita al robot retornar de manera segura y eficiente a su posición inicial. Esta etapa se subdivide en dos partes: la localización del objeto en el mapa y la planificación de la trayectoria de retorno.

El desarrollo de esta práctica permitirá a los participantes adquirir habilidades en programación de robots, manejo de sensores, algoritmos de navegación y planificación de trayectorias, aspectos fundamentales en el campo de la robótica y la automatización.

2 Implementación

2.1 Estrategia de Exploración

Para la implementación de la primera fase del proyecto, se ha adoptado una estrategia de exploración que consiste en que el robot Khepera IV explore el entorno manteniendo siempre a su izquierda una pared. Esta técnica garantiza que, al completar una vuelta al mapa (retornar a la casilla de partida), el robot habrá registrado la disposición completa de las paredes del entorno.

2.2 Sensores Infrarrojos para Navegación

Para facilitar la navegación y la detección de obstáculos, se han utilizado los sensores infrarrojos del robot. Estos sensores permiten al robot evitar colisiones con las paredes, asegurando así una exploración segura y eficiente del entorno.

2.3 Detección del Objeto Amarillo

Adicionalmente, se ha incorporado una cámara al robot para la detección del objeto amarillo. En lugar de procesar toda la imagen captada por la cámara, se ha optado por analizar únicamente los píxeles más centrales de la imagen. Esta estrategia minimiza la posibilidad de detectar el objeto en una casilla lateral y asegura que el robot se encuentre en la casilla adyacente al objeto cuando lo detecta. La detección del objeto amarillo se realiza procesando la imagen de la cámara. Se recorta la imagen para centrarse en la región de interés y se convierte al espacio de color HSV. Se aplica un rango de color amarillo para crear una máscara y se cuenta el número de píxeles amarillos. Si supera un umbral, se detecta el objeto amarillo.

2.4 Optimización del Mapa

Durante la primera vuelta de exploración, al detectar el objeto, que siempre estará adyacente a una pared, se almacena esta pared en la matriz del mapa, incluso antes de que sea detectada por los sensores infrarrojos. Este enfoque permite una detección anticipada del objeto y facilita la planificación de la trayectoria en etapas posteriores.

Una vez finalizada la primera vuelta de exploración y mapeo del entorno, se implementa una función para reducir el tamaño de la matriz del mapa. Esta optimización tiene como objetivo minimizar el uso de memoria y reducir la complejidad computacional en la planificación de la trayectoria de retorno.

2.5 Planificación del Camino de Vuelta

Para la planificación del camino de vuelta, se ha implementado una técnica de "frente de onda". Esta técnica consiste en crear un mapa secundario, conocido como "mapa de distancias", a partir de la casilla de salida. En este mapa, se propaga una señal desde la casilla de salida a través de las celdas adyacentes,

marcando en cada una la distancia discreta respecto a la salida. Una vez generado este mapa de distancias, se emplea en conjunto con un montículo de mínimos y una lista de celdas visitadas para llevar a cabo la búsqueda de un camino óptimo de regreso. Esta combinación de herramientas permite al robot determinar la ruta más corta y eficiente para retornar a la base, evitando bucles y optimizando el trayecto de vuelta.

2.6 Estructura del Código y Manejo de Time Steps

La estructura general del código se fundamenta en un bucle principal que maneja los tres comportamientos clave del robot: mapear, patrullar y volver a casa. Estos comportamientos se controlan mediante una variable de estado que determina la acción a ejecutar en cada iteración del bucle. Es importante destacar que este bucle principal también gestiona los 'time step' de la simulación. Gracias a una cuidadosa implementación de los comportamientos, se garantiza que no acaparen excesivamente la CPU, permitiendo así que la simulación se ejecute de manera eficiente y fluida.

3 Problemas y Conclusiones

Durante el desarrollo e implementación de la práctica, nos enfrentamos a diversos desafíos que influyeron en la precisión y eficiencia del robot. Uno de los problemas más notables fue la acumulación de errores al moverse y girar entre celdas. Los giros realizados por el robot no son siempre exactos, lo que lleva a una posible acumulación de desviaciones. Esta acumulación puede resultar en que el robot se desvíe de su trayectoria planeada y, en situaciones extremas, no detecte adecuadamente paredes en celdas adyacentes, provocando colisiones.

Para mitigar este problema, se optó por reducir la velocidad de desplazamiento del robot, buscando minimizar los errores acumulativos a expensas de aumentar el tiempo total de ejecución. Esta solución, aunque efectiva en cierta medida, no aborda completamente el problema subyacente y solo actúa como una medida paliativa.

Una solución alternativa considerada fue llevar un registro de los giros realizados hacia cada lado. Cuando se detectase una diferencia significativa en la cantidad de giros se propuso realizar un giro de 360 grados en la dirección opuesta para compensar el error acumulado. Sin embargo, esta solución presenta desafíos tanto en eficiencia como en intuición, ya que requeriría que el robot girase sobre sí mismo regularmente, lo que no solo es ineficiente sino que tampoco garantiza una corrección completa del error.

En conclusión, a pesar de los desafíos encontrados, la implementación de la práctica permitió adquirir un conocimiento valioso sobre las arquitecturas híbridas y la planificación de trayectorias, así como habituarse con el uso de simulación y el procesado de imágenes.

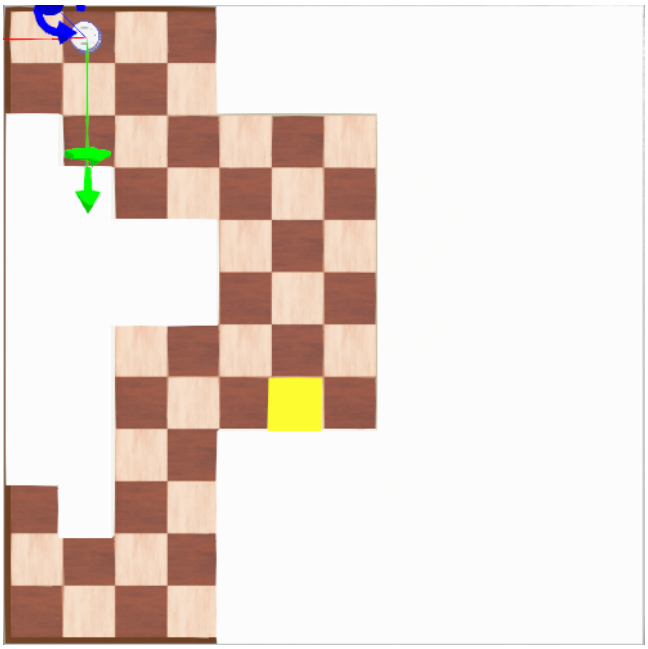

Figure 1: Mapa de prueba en webots

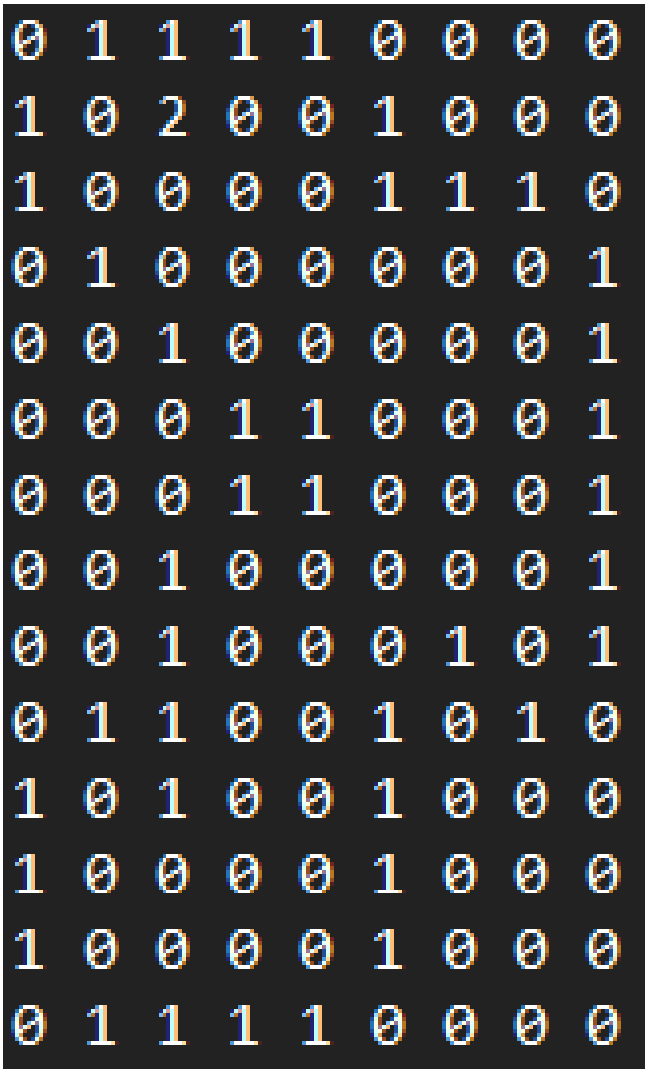

Figure 2: Matriz resultante de aplicar el código sobre el mapa de la Figura 1, los ceros representan celdas vacías, los unos representan paredes(el objeto amarillo se representa como una pared), el número dos representa a la casilla de salida, donde se ha de regresar una vez se encuentre el objeto amarillo.