PRÁCTICA 2 : MAPEADO Y LOCALIZACIÓN

Nuria Iglesias Traviesa
Xoel González Pereira

Grupo de prácticas: 1.2

Robótica
Universidade da Coruña
Curso 2022/23

Índice

1. Introducción

Esta práctica consiste en la implementación en el lenguaje Python, usando la plataforma de Webots con el robot Khepera IV de una arquitectura híbrida. Se añadiran funcionalidades de mapeado y localización a través del mapeado del terreno a través de seguir las paredes, mostrando 1 en los lugares ocupados (paredes) o 1 si está vacío. También contará con la planificación de trayectorias, el cual consistirá en seguir paredes para después volver a base.

2. Descripción del problema

Se plantea un problema a resolver en el que el robot Khepera IV sale de una celda base, explorando y guardando en el primer recorrido el mapa. Después, una vez ha guardado el mapa y ha llegado a la posición de la que ha partido, busca el objeto de color amarillo en el mapa, y cuando lo encuentra calcula la ruta óptima usando el algoritmo de A* desde esa posición a la celda base, volviendo así a esta y terminando el recorrido.

En cuanto al entorno en el que se mueve el robot, se trata de un entorno discreto, dividido en celdas de 25x25 cm, y con dimensiones conocidas, de 12x12 celdas. Los obstáculos nunca estarán a medias entre dos celdas.

El robot tiene acceso a varios sensores, de los cuales han sido usados los infrarrojos y la cámara:

1. Sensor de infrarrojos: para medir la distancia hasta la pared o hasta ciertos obstáculos. Se ha usado para los casos en los que el robot camina recto, gira a la izquierda o gira a la derecha (quedándose así a cierta distancia de la pared para realizar el giro y no chocarse).

2. Cámara: Se utiliza para detectar el color del objeto.

3. Decisiones de diseño

Este diseño se divide en dos partes. Siendo la primera la exploración del entorno y la generación del mapeado a través de esta exploración. La segunda, sigue recorriendo paredes pero cuando encuentra el objeto amarillo, usa el mapa generado anteriormente para volver al punto de partida.

1. Seguir paredes:

    a) Creamos la implementación a través del uso de los sensores infrarrojos. Si no detecta una pared a su lado, el robot girará a su izquierda, ya que implicará que se ha encontrado con una esquina. Mientras no

se encuentre con una pared en frente, seguirá caminando recto y por último, si tiene la pared en frente, este girará a la derecha para evitar chocarse con él.También añadimos el uso de un valor booleano, llamado giroL para tener en cuenta cuándo gira a la izquierda, y así en caso de que no haya girado a la izquierda anteriormente y no tenga pared al lado, implicará que tiene que girar a su izquierda. Hemos creado las funcionalidades de giro e ir recto, que se aplican en estos casos como funciones separadas para su cómodo uso.

b) Por otro lado, según recorre el terreno, va almacenando los valores de su alrededor a través de saber la posición y la orientación en todo momento, a través del uso de la odometría. El mapa es creado a través de los valores de los sensores de alrededor, si estos valores son superiores a 180, implica que tiene una pared, y se marcará como uno en la matriz, en caso contrario, será un cero. Estos valores luego son guardados en un fichero llamado mapa.txt en función de la ruta asignada. Este mapa se usará más adelante para poder volver a la base.

2. Volver inicial: En esta funcionalidad, el robot una vez ha mapeado el terreno y ha sido guardado en un fichero con unos y ceros, representando los unos las paredes y obsáculos, y el resto de celdas libres representadas por ceros, primero se sigue el mismo procedimiento que en seguir paredes, caminando recto y girando cuando sea necesario. Cuando encuentre y detecte el objeto amarillo con la función de procesamiento de imagen, el robot pasará a calcular la ruta óptima con la función creada para el cálculo del algoritmo A*(A star), en la que se le pasa la posición de la celda de la que se parte, la meta que será la posición inicial y el mapa guardado. Así, una vez el robot calcule la ruta, para cada nodo de la ruta calculará que movimientos tiene que hacer en función de los valores del nodo, teniendo en cuenta que la posición se va actualizando y que además la orientación también lo hace. Así, irá recorriendo la ruta haciendo las comprobaciones y movimientos necesarios en cada nodo hasta llegar a la meta, terminando de esta forma la funcionalidad.

El mapa será almacenado en un fichero .txt en la ruta que se especifique en la función guardarmapa, para así poder utilizarlo posteriormente. Los mapas que guarda tanto para el primer mapa como para el segundo serán almacenando como ceros las celdas libres y las paredes u obstáculos como unos:

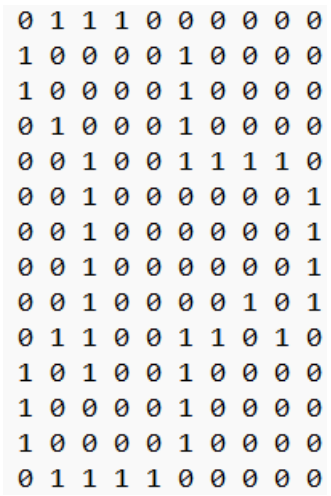

Figura 1: Mapa 1 almacenado

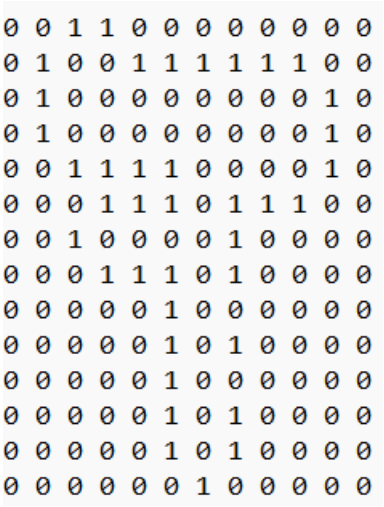

Figura 2: Mapa 2 almacenado

4. Ventajas e incovenientes frente a arquitectura reactiva

Los robots que utilizan comportamientos reactivos son atrayentes ya que son computacionalmente muy económicos. Estos robots se basan en una conexión directa entre los sensores y los actuadores, lo que permite ejecutar la acción óptima en cada momento. Es decir, solo tienen la capacidad de percibir y actuar en función de los datos obtenidos en tiempo real.

Sin embargo, en la arquitectura híbrida, se añade una capa de planificación, permitiendo que el robot pueda tomar decisiones que no son exclusivos de los datos obtenidos en tiempo real, sino que además se basan en la información almacenada en la memoria. Esto se puede ver en el caso en el que el robot tiene que regresar a su base después de haber encontrado un objeto, las arquitecturas híbridas pueden mejorar los movimientos del robot debido a los datos que han sido guardados.

En conclusión, las arquitecturas híbridas combinan lo mejor de ambos mundos: por un lado, la capacidad de reacción rápida de los robots con comportamientos reactivos, y por otro lado, la capacidad de selección en los compartimentos más complicados que proporciona la planificación.

5. Peculiaridades de la práctica

En la función implementada para guardar el mapa cada uno tendrá que añadir su propia ruta en la que quiere guardar el fichero .txt.
Además, sabemos que ciertas veces cuando ejecutamos el código, pensamos que debido a la acumulación de error, el robot Khepera IV falla después de ser ejecutado unas cuantas veces.
Por último, debido a ciertas zonas de iluminación del mapa, no siempre detecta correctamente el color que es.