Estructuras de Datos y Algoritmos I

Trabajo Práctico Final: Path Planning

Agustín López

Carrera: **Licenciatura en Ciencias de la Computación**

Facultad de Ciencias Exactas Ingeniería y Agrimensura

1 Actividad 1

En esta actividad, se presenta el desarrollo e implementación de un algoritmo de navegación para un robot en un entorno desconocido. En este entorno, el robot se mueve a ciegas y solo puede detectar los obstáculos al "chocarse" con ellos. El objetivo es que el robot llegue a la posición objetivo de manera eficiente, evitando obstáculos y explorando nuevas áreas de forma sistemática.

A continuación, se detallan los pasos del algoritmo y su implementación.

1.1 Descripción del Algoritmo

Problem Algoritmo
El algoritmo para la navegación del robot sigue los siguientes pasos:

1. **Paso 1:** El robot comienza moviéndose en una dirección vertical hasta intentar alcanzar la altura de la posición objetivo. Cuando el robot logra llegar a la esa altura o se encuentra con algun obstaculo intenta moverse en dirección horizontal siguiendo el mismo principio. Este proceso se repite en bucle hasta que el robot no pueda acercarse más al objetivo utilizando este método.
2. **Paso 2:** Cuando el robot llega a una posición en la que no puede avanzar más, debe dirigirse hacia una celda que no haya sido visitada previamente. Este paso asegura que el robot explora nuevas áreas del mapa y evita el movimiento redundante en regiones ya descubiertas. Después, el robot vuelve al paso 1.
3. **Paso 3:** Si no hay celdas no visitadas disponibles, el robot retrocede una posición por la ruta previamente recorrida y regresa al paso 1.

Este enfoque permite al robot explorar el entorno sin quedar atascado en un bucle infinito y llegar a la posición objetivo de manera eficiente, maximizando la cobertura del área y minimizando el tiempo de búsqueda.

1.2 Estructuras de Datos Utilizadas

Estructura `Robot`: Para la implementación del algoritmo, una de las estructuras más importantes es `Robot`, que representa la información con la que cuenta el robot y que luego utiliza para seguir el algoritmo descripto.

```
typedef struct {
    int i;
    int j;
} _Punto;
typedef _Punto* Punto;

typedef enum {
    LEFT,
    RIGHT,
    UP,
    DOWN
} Direccion;

typedef struct {
    Punto pos;
    Punto dest;
    Pila movimientos;
    TablaHash visitados;
} _Robot;
typedef _Robot *Robot;
```

Descripción de los campos:

- **Punto `pos` y `dest`:** Representan la posición actual y la posición destino del robot, respectivamente.
- **Pila `movimientos`:** Una pila implementada con una lista enlazada que almacena elementos de tipo `*Direccion`. Cuando el robot realiza un movimiento que no sea un retroceso (por ejemplo, hacia la izquierda), se agrega el movimiento a la pila. De este modo, al realizar un retroceso, el movimiento que se ejecute será el opuesto al almacenado en el tope de la pila.
- **TablaHash `visitados`:** Una tabla hash que almacena las posiciones visitadas por el robot. Esta tabla se utiliza para evitar que el robot visite una celda más de una vez y para determinar si hay celdas sin visitar en el mapa. La elección de una tabla hash se basa en su eficiencia en la búsqueda en tiempo constante y en el hecho de que no se necesita eliminar elementos de la tabla.

2 Actividad 2

En esta actividad el objetivo sigue siendo el mismo que en la actividad anterior, pero en este caso el robot cuenta con un sensor que le permite detectar los obstáculos en las cuatro direcciones cardinales (arriba, abajo, izquierda, derecha). El robot se desplaza en un mapa representado por una matriz de dimensiones $N \times M$ y debe llegar a un destino específico desde una posición inicial, evitando obstáculos y utilizando el sensor para obtener información sobre su entorno e intentando optimizar el uso del mismo dado que se considera costoso.

2.1 Descripción del Algoritmo

Problem Algoritmo
El algoritmo para la navegación del robot con sensor sigue los siguientes pasos:

1. **Paso 0:** En este momento el robot no cuenta con nada de información, por lo que usa el sensor para detectar obstaculos y/o casillas libres a su alrederdor.
2. **Paso 1:** El robot siempre conoce su posicón inicial y la posicion destino, entonces utilizando el algoritmo A* calcula la ruta mas corta hacia el destino asumiendo que las casillas desconocidas estan libres pero tienen un costo mayor que las casillas conocidas para optimizar el uso del sensor eligiendo una ruta quizás más larga pero que necesite menos usos del sensor.
3. **Paso 2:** Una vez que se construye el camino optimo, el robot se mueve a lo largo de este camino mientras las casillas sean conocidas, si la siguiente casilla del camino es desconocida utiliza el sensor para asegurarse de que no hay un obstaculo y luego continua moviendose, hasta llegar al destino o que efectivamente haya un obstaculo en la casilla desconocida.
4. **Paso 3:** Cuando encuentra un obstaculo en la ruta calculada vuelve al paso 1.

Este enfoque permite al robot explorar el entorno de manera más eficiente y segura, utilizando la información del sensor para evitar obstáculos y encontrar el camino más corto hacia el destino.

2.2 Estructuras de Datos Utilizadas

2.3 Descripción del Código

El código desarrollado se divide en varias funciones clave:

1. **Estructuras de Datos:** Se definen estructuras para representar el estado del robot (`_Robot`) y la información de cada celda del mapa (`CeldaInfo`). La estructura `_Nodo` se utiliza para manejar los nodos en la cola de prioridad, esencial para la implementación del algoritmo A*.

2. **Creación y Destrucción del Robot:** La función `robot_crear` inicializa la posición y el destino del robot, así como la memoria para la matriz que representa el mapa. La función `robot_destruir` libera la memoria utilizada por el robot, asegurando una correcta gestión de los recursos.

3. **Uso del Sensor:** La función `usar_sensor` simula la interacción con un sensor que proporciona las distancias a los obstáculos en las cuatro direcciones cardinales (arriba, abajo, izquierda, derecha). Esta función actualiza la matriz del mapa con la información obtenida, marcando las celdas exploradas y los obstáculos detectados.

4. **Algoritmo A*:** La función `calcular_ruta` implementa una variante del algoritmo A* para encontrar el camino más corto desde la posición actual del robot hasta el destino. Se emplea una cola de prioridad para explorar las celdas con el menor costo estimado y se actualiza la información de cada celda en función de los costos de movimiento y la heurística (distancia Manhattan al destino).

5. **Movimiento del Robot:** La función `ir_a_destino` reconstruye el camino óptimo encontrado por el algoritmo A* y mueve el robot a lo largo de ese camino, actualizando su posición en el mapa y gestionando las celdas visitadas.

El enfoque adoptado en esta actividad permite al robot navegar en un entorno parcialmente desconocido utilizando un algoritmo eficiente y adaptativo, maximizando la cobertura y minimizando el tiempo de búsqueda.