Universidad del Valle de Guatemala

Facultad de Ingeniería

Ciencias de la Computación y Tecnologías de la Información

Proyecto 1

Inteligencia Artificial

José Luis Gramajo Moraga, Carné 22907

Angel Andres Herrarte Lorenzana, Carné 22873

Jorge Luis Lopez ,Carné 221038

22 de febrero de 2026, Guatemala, Guatemala

1. Introducción y Escenario

El objetivo de esta fase fue diseñar el motor de búsqueda para un robot de entrega capaz de navegar en un entorno representado por imágenes satelitales. El sistema debe ser capaz de interpretar un laberinto en formato de imagen (.png o .bmp) y encontrar la ruta más eficiente desde un punto de inicio (rojo) hasta una meta (verde).

2. Task 1.1 - Discretización del Mundo

Para evitar el procesamiento ineficiente de imágenes de alta resolución píxel por píxel, se implementó una estrategia de **agrupación por "batches" o "tiles"**.

- **Metodología de Grid:** La imagen se dividió en celdas de 15*15 píxeles. Se calculó el color promedio de cada celda para clasificarla.
- **Identificación de Terreno:**
    - **Paredes (Negro):** Celdas con valores RGB cercanos a [0, 0, 0] se marcaron como intransitables.
    - **Inicio (Rojo):** Se localizó la coordenada única donde predomina el canal rojo.
    - **Meta (Verde):** Se identificaron todas las celdas con predominancia de canal verde como estados de éxito (goalTest).

3. Task 1.2 - Framework de Búsqueda (POO)

Se diseñó una arquitectura siguiendo el paradigma de **Programación Orientada a Objetos**, utilizando una clase abstracta SearchProblem para definir la interfaz genérica del problema formal.

- **Modelo Formal:** La clase MazeProblem hereda de la interfaz y deduce automáticamente las acciones posibles (norte, sur, este, oeste), el estado inicial y los costos de paso basándose en la matriz discreta generada previamente.
- **Algoritmos Implementados:**
    - **BFS (Breadth-First Search):** Implementado para garantizar el camino con la menor cantidad de pasos, asumiendo un costo uniforme.
    - **DFS (Depth-First Search):** Implementado para explorar rápidamente las rutas, aunque no garantice la optimización de distancia.

4. Task 1.3 - Búsqueda A*(A-star)

Se implementó el algoritmo A* utilizando una **heurística de distancia Manhattan** para guiar la búsqueda de forma más eficiente que BFS.

- **Heurística Seleccionada:** $h(n) = | x_1 - x_2 | + | y_1 - y_2 |$

- **Costo de Paso:** Inicialmente, cada movimiento tiene un costo de 1.
- **Visualización:** Los resultados se muestran gráficamente sobre la matriz discreta, dibujando la ruta final desde el inicio hasta la meta.

5. Análisis de Resultados (Demo)

En las pruebas realizadas con el archivo Prueba Lab1.bmp, el sistema identificó correctamente:

- **Punto de Inicio:** (25, 25).
- **Punto de Meta:** (3, 2).
- **Comportamiento:** Tanto BFS como A* lograron evadir la pared diagonal negra y encontrar el camino libre sobre las áreas blancas (caminos libres).

6. Task 2.1 - Entrenamiento de la Red Neuronal

Para dotar al mapa de "semántica", se integró la capacidad de clasificar los tipos de terreno basándose en su color RGB.

Preparación del Dataset: Se utilizó el dataset "Associated RGBs and Basic Color Names" descargado mediante la API de Kaggle. Las etiquetas de texto devueltas por el modelo se mapearon a costos numéricos para el robot de la siguiente manera:

- Green (Grama) → Costo 3
- Blue (Agua) → Costo 10
- Grey (Pavimento) → Costo 1
- Yellow (Arena) → Costo 5

Modelo "Desde Cero": Se implementó una Red Neuronal (Multilayer Perceptron) exclusivamente utilizando la librería `numpy` sin frameworks de alto nivel.

- Arquitectura: Capa de entrada de 3 neuronas (R, G, B normalizados) → Capa oculta de 64 neuronas (activación ReLU) → Capa de salida de 11 neuronas (activación Softmax para clasificar 11 colores básicos).

- Entrenamiento: El modelo fue entrenado utilizando Descenso de Gradiente (Batch GD / SGD) y Backpropagation durante 800 épocas.

Métrica: Tras dividir el dataset (80% entrenamiento, 20% prueba), el modelo logró un Accuracy (Precisión) del 87.54% sobre datos nunca vistos.

7. Task 2.2 - Inferencia en vivo y Búsqueda A

Se modificó la lógica de costo de paso g(n) dentro de la iteración de A* para aplicar pesos condicionales.

Inferencia en vivo: Al evaluar los nodos vecinos, el algoritmo extrae el color RGB promedio de dicha baldosa en la imagen. Este vector se introduce en tiempo real a la Red Neuronal previamente entrenada, y

la predicción (ej. "Blue") se convierte en su costo asociado (ej. 10). Dicho costo se emplea para calcular los pesos acumulados.

Experimento de Demostración (Blue vs Grey): Para comprobar la funcionalidad, se diseñó un laberinto en mapa de bits (demo_maze.bmp) donde existen dos rutas a la meta:

- Camino A: Muy corto, pero compuesto enteramente de casillas Azules (Agua, penalización de 10 por paso).
- Camino B: Un desvío geométricamente más largo, pero compuesto de casillas Grises (Pavimento, costo de 1 por paso).

Resultado: El modelo A* con inferencia viva detectó la masiva penalización de la ruta A, forzando la cola de prioridad a explorar e inclinarse por completo por el largo (pero barato) Camino B. Esto valida la integración exitosa de la topografía neuronal sobre la heurística geométrica pura.

Repositorio: https://github.com/Abysswalkr/terrain-cost-pathfinder.git