Project I Report for COM S 4/5720 Spring 2025: A* Path Planning in Grid Environments

Nicholas Morrow[1]

***Abstract*— This report presents the implementation of an A* (A-star) path planning algorithm for grid-based navigation tasks. The algorithm efficiently finds optimal paths from start to destination positions, avoiding obstacles in a discrete environment. The implementation combines Dijkstra's algorithm's completeness with a heuristic-guided search, The solution utilizes Euclidean distance as the heuristic function, as well as 8-directional movement.**

I. INTRODUCTION

This project addresses the problem of path planning in a discrete, grid-based environment. The objective is to develop an algorithm capable of finding a valid path for an agent to navigate from a given start location to a specified end location, while circumventing obstacles. The environment is represented as a two-dimensional grid, where each cell is either traversable or blocked (obstacle). The algorithm must efficiently find the shortest path whenever one exists.

II. ALGORITHM DESCRIPTION

The implemented solution employs the A* search algorithm [1], a well-established best-first search method known for its efficiency and optimality in pathfinding. A* combines the cost to reach a node ($g(n)$) with an estimated cost from that node to the goal ($h(n)$), forming the evaluation function $f(n) = g(n) + h(n)$.

A. A* Algorithm Steps

1) **Initialization:**
   - A priority queue, Q, stores nodes to be explored, prioritized by their f-score. Q is initialized with the start node, where $f(\text{start}) = h(\text{start})$.
   - A dictionary, g_score, tracks the cost to reach each node from the start. It is initialized with $g\_score[\text{start}] = 0$.
   - A dictionary, $parent$, stores the predecessor of each node, enabling path reconstruction. It is initialized with $parent[\text{start}] = \text{None}$.
2) **Iteration:** While Q is not empty:
   - Remove node n with the lowest f-score from Q.
   - If n is the end node, reconstruct and return the path (Section II-B).
   - For each neighbor n' of n:
     - Check if n' is within grid bounds and not an obstacle.
     - Calculate the tentative g-score for n' as $g(n') = g(n) + d(n, n')$, where $d(n, n')$ is the cost of moving from n to n'. In an 8-connected grid, this value is 1.
     - Calculate the heuristic $h(n')$ (Section II-C).
     - Calculate $f(n') = g(n') + h(n')$.
     - If n' is not in g_score or $g(n')$ is lower than the existing $g\_score[n']$:
       * Update $parent[n'] = n$.
       * Update $g\_score[n'] = g(n')$.
       * Add n' to Q with priority $f(n')$.
3) **No Path Found:** If Q is empty before reaching the end node, return `None`.

B. Path Reconstruction

If the end node is reached:

1) Start from the end node.
2) Repeatedly follow the $parent$ pointers back to the start node, adding each node to the path.
3) Reverse the path to obtain the correct order (start to end).

C. Heuristic Function

The heuristic function, $h(n)$, is the Euclidean distance:

$$h(n) = \sqrt{(n_x - \text{end}_x)^2 + (n_y - \text{end}_y)^2}$$

where (n_x, n_y) are the coordinates of node n, and $(\text{end}_x, \text{end}_y)$ are the coordinates of the end node. This heuristic is admissible and consistent, guaranteeing the optimality of A*.

D. Allowed Movements

The algorithm considers eight-connected neighbors, allowing diagonal movements in addition to orthogonal movements.

III. IMPLEMENTATION DETAILS

The core logic resides within the 'planner.py' file. The 'plan-path' function serves as the entry point, accepting the grid ('world'), start position, and end position as inputs. It returns a NumPy array representing the path, or 'None' if no path exists. The 'a-star' function encapsulates the A* algorithm, and the 'heuristic' function computes the Euclidean distance. The implementation utilizes only the Python standard libraries, NumPy, and heapq.

REFERENCES

[1] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[1]Nicholas Morrow is with the Department of Computer Science, Iowa State University, Ames, IA 50011, USA `nmorrow@iastate.edu`