TraceEscape: Gesture-Based Maze Design and AI-Powered Solving on a Virtual Canvas

Nhan Tran Ishaq Mustafa Khan Rohail Alam

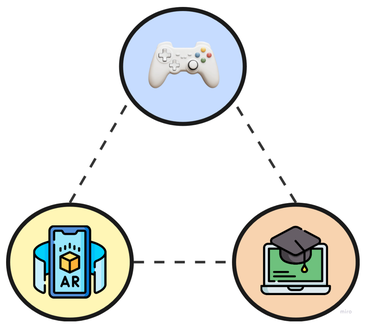
ComS 5750: Computation Perception

Abstract

This project introduces an interactive application that enables users to draw mazes through hand gestures and have them solved with Artificial Intelligence. The virtual canvas allows real-time maze creation with full control over structure and complexity. Hand gestures are captured using MediaPipe's hand landmark detection, which extracts 21 keypoints from the user's hand. These keypoints are fed into a trained gesture classification model to interpret drawing commands mapped to a pen tool, an eraser tool, and a panning tool to name a few. Once a maze is constructed, the A* pathfinding algorithm is used to compute the shortest solution path. The system is designed to be intuitive and responsive, minimizing the learning curve for new users. We evaluate the application across three key areas: the classification accuracy of hand gestures, user experience during maze design, and the computational performance of the maze-solving process. Results demonstrate that the graphical user interface (GUI) effectively bridges gesture-based interaction and classical AI algorithms for a seamless experience.

Introduction

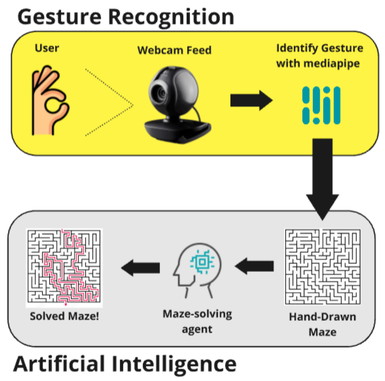
Mazes are classic spatial problem-solving tasks that require logical reasoning and efficient path traversal. With advancements in computer vision and AI, maze-solving can now be approached more dynamically and interactively. This project introduces a system that allows users to draw mazes using hand gestures and solve them in real time using AI.



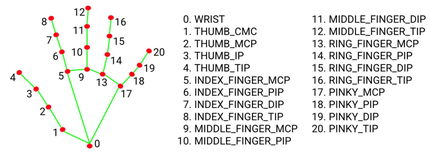
This project targets a wide range of use cases: educators seeking interactive tools to demonstrate pathfinding, visually-oriented users requiring spatial assistance, and casual users via a gameified mode where players aim to challenge the AI in a time-based setting. The ultimate goal is to eliminate dependence on traditional input devices or external tools, relying entirely on gesture-based control for both design and execution.

Methodology

Our system is divided into two components: Gesture Recognition and AI-based Pathfinding.

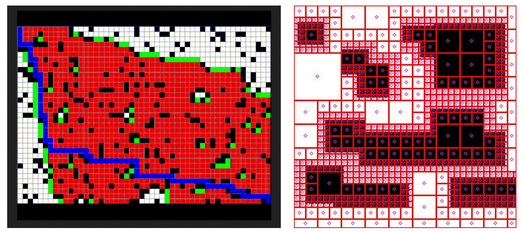


Gesture Recognition relies on a camera feed processed with Mediapipe, which detects 21 hand landmarks. These keypoints are passed to a TensorFlow-based Multi-Layer Perceptron (MLP) model trained to classify eight distinct gestures. Users interact with a canvas via these gestures—to draw, erase, or navigate maze segments.



21 Keypoints detected by Mediapipe

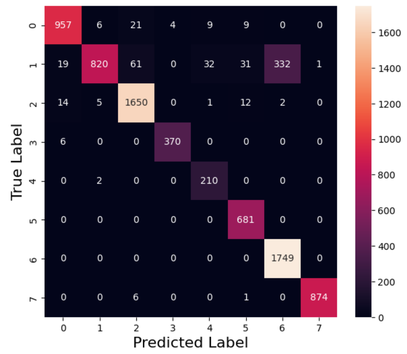
Pathfinding uses the A* algorithm to find an optimal path through the user-drawn maze. The maze is treated as a graph where each pixel is a node, and non-traversable pixels (walls) are excluded. A min-heap prioritizes exploration using Euclidean distance as a heuristic. To optimize performance, we implement a quad-tree structure that hierarchically divides the space, reducing unnecessary computation.



(i) A-Star and (ii) Quadtree path finding on a 2-D grid

Results – Gesture Recognition

The classifier is able to generally perform well and overall has a high accuracy. False positives are handled through a modified confidence thresholds to provide for a seamless drawing experience.



Confusion Matrix for the 8 Classes

One can infer from the confusion matrix in Figure 10 that there are only a few values present in the non-diagonal positions which are non-zero, and the ones which are are significantly lesser than that on the diagonals. This is an indication that the classifier is doing its job well, with rarely occurring false detections

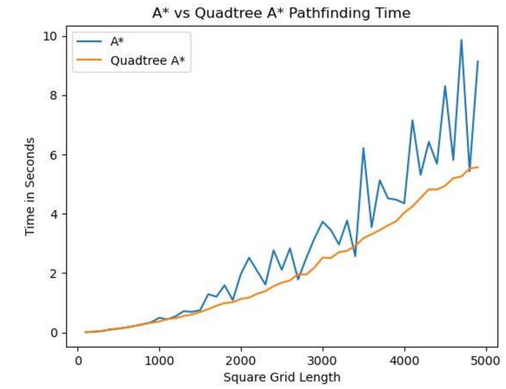
Class	Precision	Recall	F1-Score	Support
0	0.96	0.95	0.96	1006
1	0.98	0.63	0.77	1296
2	0.95	0.98	0.96	1684
3	0.99	0.98	0.99	376
4	0.83	0.99	0.91	212
5	0.93	1.00	0.96	681
6	0.84	1.00	0.91	1749
7	1.00	0.99	1.00	881
Accuracy		**0.93**		7885
Macro Avg	0.94	0.94	0.93	7885
Weighted Avg	0.93	0.93	0.92	7885

Classification Metrics for MLP

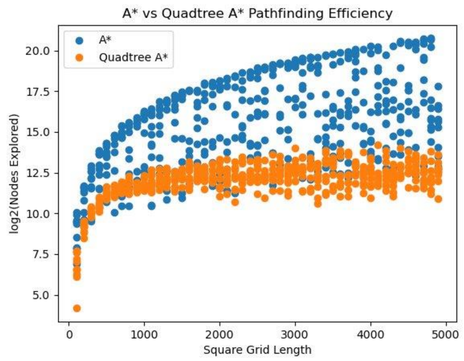
The F1 scores are consistently high for all but one of the classes, indicating reliable predictions with minimal class imbalance or outlier behavior in terms of recognition accuracy. To handle the false predictions of class 1, its confidence threshold has been tweaked to be lower.

Results – Path Finding

A* algorithm and Quadtree A* algorithm is run on square grids with sides length ranging from 100 to 4900 in increments of 100. For each side lengths, A* algorithm ran on 10 different randomly generated environment as described above and average the time that it takes to find a solution.



Speed of A* vs Quadtree averaged across 10 runs per length



Logarithmic graph of the efficiency of A* vs Quadtree on square grid with 10 runs for each square grid.

Quadtree A* is consistent in the time that it takes to find a solution than A*. This is likely due to the fact that quadtree construction is inherently consistent in term of the max number of subdivisions required in each environment. For n is the length of the grid in terms of pixels, quadtree need to be subdivided for a maximum of ⌈log(n)⌉ times. Quadtree also significantly reduces the number of nodes needed to be searched by A*, leading to a faster algorithm overall.