README for Pathfinding Algorithm Repository

Overview

This repository contains multiple pathfinding algorithms designed to find efficient routes through obstacle-rich environments. The primary focus is on comparing and enhancing three core algorithms: an optimized version of the Probabilistic Roadmap (PRM), the Rectangles algorithm, and a novel algorithm named APPATTrectangles. Each algorithm is benchmarked against various scenarios to assess computational efficiency and path accuracy.

Key Contributions

1. Optimized PRM Algorithm

An optimized version of the Probabilistic Roadmap (PRM) algorithm has been implemented to improve its performance in navigating obstacle-dense areas. This version of PRM reduces the time complexity by optimizing the process of selecting random nodes and connecting them in a graph. It has been integrated into the overall system to allow comparative analysis with other algorithms.

- **File**: `PRMwithLoop.py`(PRMwithLoop)

2. Rectangles Algorithm

The Rectangles algorithm is based on research that identifies the corners of obstacles as interest points for pathfinding. The algorithm connects these corner points to form a graph, creating minimal and efficient paths between the start and end points. This algorithm is particularly effective at navigating complex obstacle layouts.

- **File**: `Rectangles.py`(Rectangles)

3. APPATTrectangles Algorithm (Novel Contribution)

The APPATTrectangles algorithm is a novel development that combines the efficiency principles of the APPATT and Rectangles algorithms. It finds near-optimal paths by leveraging obstacle corner connections while incorporating optimizations for computational efficiency. In testing, APPATTrectangles outperformed the other algorithms in terms of both speed and accuracy.

- **File**: `APPATTrectangles.py`(APPATTrectangles)

4. Algorithm Testing and Analysis

Extensive testing and benchmarking have been conducted across multiple scenarios. The comparative analysis highlights the superior performance of APPATTrectangles in terms of computational efficiency and path accuracy, followed by the optimized PRM and Rectangles algorithms.

- **Files**:
  - `ShortesPathRandomObstacles.py`(ShortesPathRandomObstac…)
  - `ShortestPathRandomObstaclesWithEdges.py`(ShortestPathRandomObsta…)

Files and Directories

- **`PRMwithLoop.py`**: Contains the implementation of the optimized PRM algorithm (PRMwithLoop).
- **`Rectangles.py`**: Implements the Rectangles algorithm for pathfinding using obstacle corners(Rectangles).
- **`APPATTrectangles.py`**: Implements the APPATTrectangles algorithm, combining the strengths of APPATT and Rectangles(APPATTrectangles).
- **`SaveGraph.py`** & **`SaveGraphObstaclesNotOverlapping.py`**: Scripts to generate and save graphs with obstacle information(SaveGraph)(SaveGraphObstaclesNotOv…).
- **`ShortesPathRandomObstacles.py`** & **`ShortestPathRandomObstaclesWithEdges.py`**: Scripts to test algorithms with random obstacles and visualize results (ShortesPathRandomObstac…)(ShortestPathRandomObsta…).

How to Use

1. **Run the algorithms**: To test the performance of each algorithm, run the corresponding Python script (`PRMwithLoop.py`, `Rectangles.py`, or `APPATTrectangles.py`).
2. **Visualization**: Each script contains functions to visualize the graph, obstacles, nodes, and paths. The function `display_array_with_graph_and_path()` is used to display the shortest path calculated by each algorithm(APPATTrectangles)(PRMwithLoop) (Rectangles).
3. **Benchmarking**: Use the provided random obstacle generation scripts (`ShortesPathRandomObstacles.py`, `ShortestPathRandomObstaclesWithEdges.py`) to generate test environments and compare the algorithms(ShortesPathRandomObstac…) (ShortestPathRandomObsta…).

Future Work

- Further optimization of the PRM and Rectangles algorithms.
- Expansion of the APPATTrectangles algorithm to handle dynamic obstacles.

This repository is a comprehensive exploration of efficient pathfinding in complex environments, with an emphasis on balancing computational efficiency and path accuracy.