AL-2002 PROJECT24

Note: Carefully read the following instructions (*Each instruction contains a weightage*)

1. There must be a block of comments at start of every question's code by students; the block should contain brief description about functionality of code.
2. Comment on every function and about its functionality.
3. Use understandable name of variables.
4. Write a code in PYTHON language and you may use any of IDE or Notebook environment.
5. First think about the problems statements then you may start your programming.
6. At the end when you done your tasks, attached .py or .ipynb files on google classroom. Paste your complete code in word file along with output **(Make sure your submission is completed)**. In case of missing any file, marks will be deducted.
7. The project can be completed **with maximum 4 memebers in group**, but the evaluation will be based upon individual effort and VIVA.
8. Please submit your file in this format **19F1234, 19F1235, 19F1236**.
9. Do not submit your project after deadline. Late and email submission is not accepted.
10. Do not copy code from any source otherwise you will be penalized with negative marks.
11. **YOUR MARKING WILL BE BASED ON PRIOR SUBMISSION OF YOUR CODE BEFORE DEADLINE AND VIVA.**

PROBLEM 1 Autonomous Delivery Robot

An autonomous delivery robot for your online grocery orders, needs to navigate through an area in the city to deliver packages to designated locations. Its environment is filled with obstacles like buildings, houses, and vehicles on roads. Your task is to guide the robot to deliver the order safely to the customer. The following is a detailed description of your task.

Environment Representation:

- Design a representation of some city area, possibly as a grid or a graph.
- Include information about buildings, houses, delivery points, and vehicles on roads.
- A grid size of around 15 x 15 should be created as a graph.
- The initial "start location" for robot should be fixed.

Algorithm Implementation:

- Implement informed search algorithms like best first and A* for robot motion planning, which you should think is the best for this problem. For this, you have to make a comparison between both algorithms and choose the best.
- Develop a heuristic function using Euclidean distance that considers the distance to the goal, and the presence of obstacles.
- If applicable, the cost from one location to another location should be randomly generated but should be between 1 to 20 and the cost from the current location to the goal should be calculated by Euclidean distance.

Dynamic Environment Handling:

- Simulate dynamic changes in the environment such as changing the start, goal state and vehicles positions after delivering an item.
- Ensure that the robot can adapt its path planning in real-time to handle these changes efficiently.

Path Execution and Control:

- Develop a path execution module that allows the robot to follow the planned path while avoiding collisions.
- The location of goal should be generated randomly within the grid. You must create 5 different locations for delivering items.
- In the start robot waits for delivery, then you should assign delivery items to it and destination.

After the robot has done its job the previous delivery location should be the start point and then assign the next delivery location and it will do the same for the remaining 4 deliveries.

User Interface and Visualization:

- Create a user interface to interact with the simulation environment.
- Visualize the robot's path, obstacles, and delivery points in real time.

Performance Evaluation:

- Evaluate the performance of the motion planning algorithm in terms of path optimality, execution time, and adaptability to dynamic changes.

Implementation Steps:

Environment Representation:

You can use matplotlib to visualize the city area, with obstacles and delivery points represented graphically.

Algorithm Implementation:

Implement informed search algorithms for an autonomous delivery robot in Python.
Use an animation library to animate the robot's movement and visualize the planning process.

Python Libraries for Animation and GUI:

- **matplotlib:** It provides functionality for creating animations using its animation module.
- **Pygame:** This library is well-suited for game development and includes features for creating animations and interactive simulations.
- **Tkinter:** It's Python's built-in library for creating simple GUI applications.
- **PyQt or PySide:** These libraries provide more advanced GUI capabilities with extensive widget sets and support for modern GUI design.

Note: You can use any other python library if required.

Project Deliverables:

- Source code implementing the motion planning algorithm, dynamic environment handling, path execution, and simulation components.
- A user-friendly interface allows users to interact with the environment, visualize the robot's movement, and control the simulation.
- You can code in .ipynb or .py format.

Bonus Task:

Multi-Robot Coordination:

Extend the project to handle multiple autonomous robots navigating in the same environment simultaneously. Implement coordination strategies to prevent collisions and optimize delivery routes.