Section II: Methodology

Role of Student vs. Mentor

All work was done by me over the course of the past 4 months (November 2023- February 2024). Mentors guided the overall presentation and organization of this project.

Equipment and Materials

All algorithms were coded and visually represented using Python. A remote API was established to facilitate communication between the Coppelia Robotics Simulation software and a Python Integrated Development Environment (IDE). Through this remote API, various data, including readings from ultrasonic sensors and Lidar sensors in the simulation, was exchanged. Furthermore, the drone within the simulation responded to commands dispatched from the Python IDE, enabling controlled movement. All algorithms and the Coppelia Simulation were executed remotely on a Dell Precision Tower 3420 computer.

Mathematical Problem Statement

In an unknown 3D space, an energy-efficient coverage path capable of effectively navigating and clearing obstacles within the space. The mathematical problem is formulated as an optimization task, where the goal is to minimize the total energy cost associated with traversing the coverage path. The unknown 3D space is represented as R in a three-dimensional coordinate system, with a set of obstacles denoted as O. The optimization objective is expressed as the integral of the energy function E along the coverage path P where an optimal P is found such that the total energy cost, $\int_P E\, ds$ is minimized. In addition to that, path P is subject to the condition that $P \subseteq R, P \cap O = \emptyset$ and P is a continuous and feasible coverage path in R.

This paper will address the stated optimization problem.

Objective 1: Obstacle avoidance

The drone's movements were performed through predefined movement functions moving 1 unit front, back, left, right and $\sqrt{2}$ units in the other 4 directions (Fig 1) . The drone was set to navigate a grid-based landscape with a representation of 50x50 units and a grid size of 1 $unit^2$. The drone was also equipped with 8 ultrasonic sensors for the 8 directions (Fig 2). The ultra sonic sensors were equipped with a range of 1.05 m and a search angle of 45 degrees. Once the simulation environment was initialized, each movement in Fig. 1 was assigned an index i. The optimal movement was chosen such that $i_{optimal} = argmin_i(J_i)$ where $J_i = d_i(P_i, T) + 1000\alpha_i$, where $d_i(P_i, T)$ is the expected distance between the position of the drone and the target after movement i, and α_i is 1 if an object is detected in the ultrasonic sensor corresponding to movement i or 0 otherwise. This process was repeated until the target position was reached.

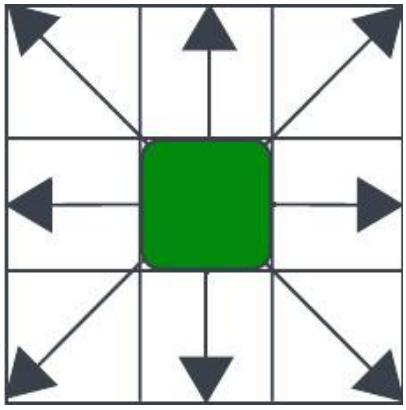

Figure 1: Drone's possible movements. The 4 corner movements have a length of $\sqrt{2}$ units while the other movements have a length of 1 unit. Image generated in Lucid.

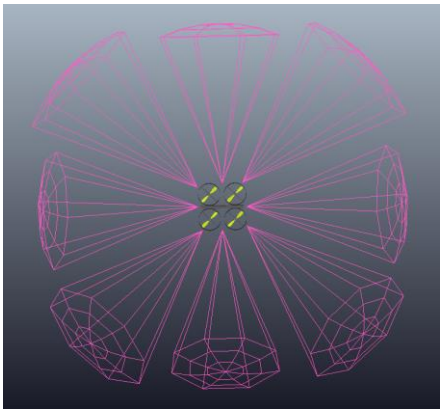

Figure 2: An image of the drone in the middle surrounded by 8 ultrasonic sensors for the 8 movement directions.

Objective 2: Mapping of the environment

A Hokuyo URG-04LX-UG01 LiDAR sensor with a 3% margin of error was used in the simulation to map the environment. The sensor transmitted a set of detected points, $\{S: S = (x, y, z)\}$, in 3D space to the IDE through a remote API. The algorithm then converted the set of points using the following logic (see Fig x). A unique rounding operation was performed on each element of S. The rounding operation was defined to be the following $\lfloor x \rceil = \lfloor x \rfloor + 0.5$ if $x - \lfloor x \rfloor \geq 0.5$, otherwise $\lfloor x \rfloor$. Once this operation was performed, each element of the rounded set with exactly 1 integer coordinate was used as grid coordinates. These coordinates were then colored black to show that it was occupied using a colormap.

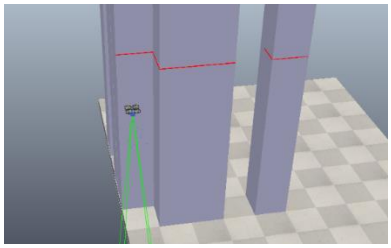

Figure 3i: An image of the environment in Coppelia Sim

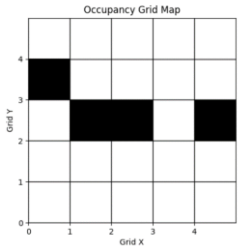

Figure 3ii: Corresponding occupancy grid map

Objective 3: Optimal Area Division

An n x n grid was split into n^2 cells. Two drones were initialized in 2 distinct cells $(0,0,z)$ and (n,n,z). The algorithm iterated through each cell and calculated a cost A which is the distance between each drone's current position and the current cell. The cell was then assigned to the drone to the with the minimum cost A. However, if the cell was already assigned to another drone, a penalty β was added to the cost. β was set to be 0.25. This algorithm was implemented in python and executed on a Dell Precision Tower 3420 computer.

However, this algorithm did not account for the issue of spatial disconnection. There were other obstacles or another drone's assignment in between the current drone's assignment as illustrated in

Figure 4i. To fix this issue, any assigned path, or a section of assigned path that does not contain the drone's current position was penalized by increasing the penalty β by the proportion of already assigned cells of n^2. β remained this value until updated. By incorporating a dynamic penalty β, the optimal area assignment matrix ensured spatial connectivity (Fig. 4ii).

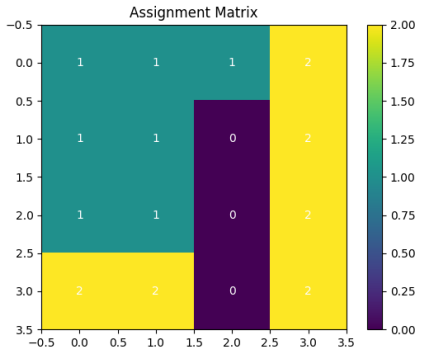

Figure 4i: Depiction of suboptimal area assignment with spatial disconnection. "1"s represent drone 1's assignment, "2"s represent drone 2's assignment and "0"s represent obstacles. Image generated by the algorithm.

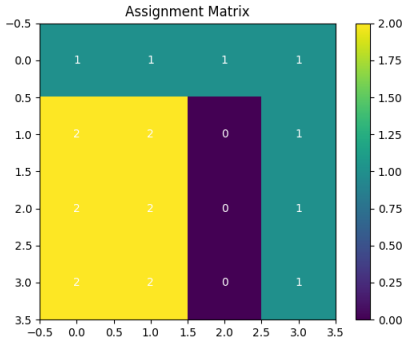

Figure 4ii: Depiction of the area assignment ensuring spatial connectivity. Image generated by the algorithm.

Energy-aware Coverage path generation

Coverage path generation

Coverage paths are defined as paths that enable the robot to visit every cell of a given area (Wu et al., 2019).

The coverage path for a given drone for a given area is a path that wraps around the Minimum Spanning Tree of said area (Kapoutsis et al., 2017). Given the properties of MST, it could also be proven that travelling through every point of the MST with an appropriate width ensures a coverage path (Fig 5). The coverage path in this paper was defined to be a path moved by a $1x1$ square through all the given points of a generated energy-efficient MST. While a cyclic tree could be used for the drone to traverse across assigned points, the time complexity for the algorithm is $O(n!)$(Appendix 1) whereas the suggested MST

generation algorithm has a time complexity of $O(n^2 + E + log(E))$(Appendix 2) where n is the number of input points and E is the minimum number of edges required to connect the given n points

To simulate the drone, the drone's height was fixed at 5.5 m in the simulation environment. A camera facing downwards was attached to the bottom of the drone (Fig 6). The camera's view angle was set to 10.37 degrees given by $2\,arctan\left(\frac{0.5}{5.5}\right)$.

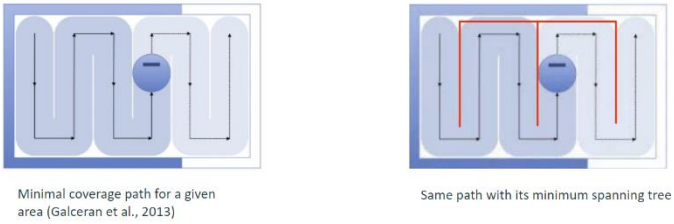

Figure 5: Depiction of a minimum spanning tree incorporated in the minimal coverage path for a given area.

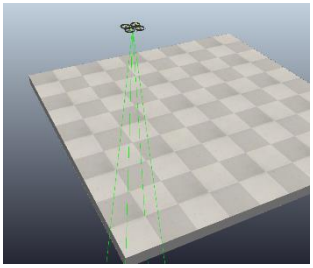

Figure 6: Image of a camera view of the drone covering 1 unit square.

Generating an energy efficient MST

A modified Kruskal's algorithm was used to generate a Minimum Spanning Tree (MST) for the assigned cells. Kruskal's algorithm is renowned for its efficiency in constructing the most economical connections between points in a graph. The implementation involved sorting edges based on their weights. The weight, W, of an edge is calculated as follows:

$$W = \lambda * degrees\ of\ turn\ from\ previous\ edge + \mu \text{ if the edge is a straight line}$$

$$\text{else } W = \lambda * degrees\ of\ turn\ from\ previous\ edge + \sqrt{(2)}\mu$$

where λ is the joules per degree turn of the drone and μ is the joules per unit traveled of the drone. λ and μ were set to 0.01 and 0.1, respectively. These experimental values of energy consumption are

derived in “UB-ANC planner: Energy efficient coverage path planning with multiple drones” (Modares et al., 2017). The algorithm then proceeded by iteratively considering edges in ascending order of weight, ensuring the inclusion of edges that connect cells without forming cycles. To achieve this, a Union-Find data structure was utilized, allowing for the efficient tracking of connected components. This approach facilitated the creation of a tree structure that is both acyclic and connects all assigned cells. Furthermore, incorporating the energy variables and minimizing the cost will make this MST energy aware.

Integration of methods

With the different objectives implemented, they were integrated together which was then tested. The 4 different parts were integrated together as per the algorithmic flowchart in Figure 7. The algorithm was then implemented in 4 different environments for both a single drone and a double drone instance.

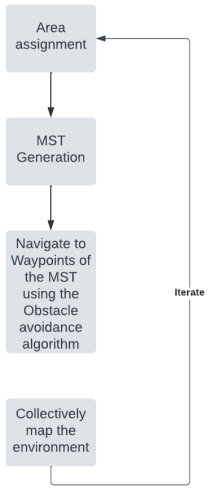

Figure 7: Algorithmic flowchart of the integration of methods.

Statistical Tests

Linear regressions were employed as a primary analytical tool to compare relationships within the experimental dataset. Given the deterministic nature of the algorithm, the reliance on statistical tests that traditionally account for randomness, such as randomization tests or Monte Carlo simulations, was deemed unnecessary for direct comparisons. Unlike processes influenced by chance, the algorithm's consistent and systematic operation allowed for a focused examination of linear relationships without the need for additional stochastic considerations.