An Approach for Mapping Unknown Environments using Frontier-Led Swarms

Kumara Ritvik Oruganti
A James Clark School of Engineering
University of Maryland
College Park, Maryland 20742
Email: okritvik@umd.edu

***Abstract*—Robots have become increasingly important in many aspects of life, from manufacturing to health care, logistics to entertainment. The basic building blocks for the robot or agent to function in an environment are Perception, Navigation and Control. To navigate in an environment, it is necessary for the robot to require a map in which they operate. This project proposes an implementation of a novel swarm-based algorithm for the exploration and coverage of unknown environments while maintaining the formation of the swarm and a frontier-based search for effective exploration of the unknown environment.**

***Index Terms*: Navigation, Control, Perception, Swarm, Frontier-Based Search**

1. Introduction

In the realm of robotics applications, it is commonly assumed that the robot possesses prior knowledge about the environment. State-of-the-art planning algorithms such as Dijkstra, A*, Randomly Exploring Rapid Trees [**?**], and various flavors of RRT rely on the assumption that the obstacle space is well-known to the agent. However, in scenarios where the environment is unknown, exploration becomes a critical task to maximize the coverage of the search area.

Traditional coverage algorithms, such as Voronoi-based approaches and graph-based methods, typically assume independent agents. To address the challenge of a robot lacking prior knowledge of the environment, this project proposes the implementation of a frontier-based exploration strategy using multiple robots. This strategy draws inspiration from biological and physicomimetic emergent behaviors observed in bird flocks, ants, and repulsion between charged particles.

Frontiers refer to the boundary regions between known and unknown space. To acquire knowledge about the unknown areas, robots need to navigate near these frontiers and explore them. This process is repeated until no frontiers remain, indicating complete exploration of the environment. Different approaches exist for the frontier-based search algorithm, such as Wavefront Frontier Detection and Fast Frontier Detection. In this project, the concept of Reynolds' boids is employed to realize the search strategy. Additionally, conventional algorithms for Simultaneous Localization and Mapping (SLAM) are considered, which update environment information while simultaneously exploring and avoiding obstacles on platforms like TurtleBot and Robot Operating System (ROS).

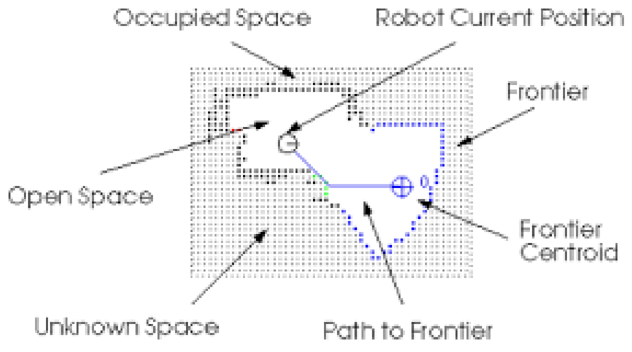

Figure 1: Frontier exploration [7]

The subsequent sections of this paper delve into a comprehensive discussion of related work, the problem statement elucidated through mathematical notations, a detailed explanation of the technical approach, challenges encountered, and an analysis of the results obtained along with performance metrics. Throughout the paper, the terms "robot" and "agent" are used interchangeably to refer to the autonomous entities involved in the exploration process.

2. Related Work

Titus Cieslewski et al. [2] introduced an innovative approach that leverages version control systems to efficiently retrieve and update an agent's map. This method effectively reduces inter-robot communication bandwidth in distributed networks. Alejandro Puente-Castro et al. [3] proposed a swarm path planning technique based on reinforcement learning. This approach enables efficient field prospecting regardless of the field's size. Syed Irfan Ali Meerza et al. [**?**] presented a dynamic obstacle avoidance and coverage technique utilizing particle swarm optimization.

Coverage in robotic systems can be classified into two types: static coverage [4] and dynamic coverage. Static coverage involves forming a swarm in a way that ensures complete sensor coverage of the environment. However, practical constraints such as limited sensor range and communication bandwidth often require robots to move and sample the environment at a reasonable resolution to achieve complete coverage [5].

Collaborative exploration and coverage require dynamic obstacle avoidance and situational awareness. Cheng et al. [6] proposed a dynamic coverage algorithm that employs a leader-follower framework and utilizes flocking techniques for swarm formation. Dario Albani et al. [16] employed a UAV swarm to perform distributed mapping of agricultural fields for weed detection.

For post-flood assessment in disaster scenarios, Leighton Collins et al. [17] developed a novel multi-robot coverage path planning algorithm. This algorithm efficiently allocates time and ensures a balanced workload for the UAV swarm, enabling effective assessment of flood-affected areas.

3. Problem Statement

3.1. Map Building

A 2D grid defined by G with a coverage grid cell location $G_{i,j}$ is said to be covered by the agent A_i when the position P of the agent A_i is $P_i = G^i_{i,j}$ and $G^i_{i,j} = 3$.

When the agent A_i receives the coverage grid cell from the neighbouring agent A_j, then the agent A_i is said to have covered the cell $G^i_{i,j}$ indirectly.

If there is an obstacle, then $G_{i,j} = 1$ and $G_{i,j} = 0$ if the area is unexplored.

3.2. Map Stitching

An agent $A_{i,j}$ starts at an initial position $P^i_{initial}$ and explores its own grid map. When it receives a grid map from another agent A_j, The maps are updated by taking the mathematical formulation of comparing each cell of the agent's grid map.

$$G^i_{i,j} = \begin{cases} 3, & \text{if } G^i_{i,j} = 0, G^j_{i,j} = 3 \\ \\ 0, & \text{if } G^i_{i,j} = 0, G^j_{i,j} = 0 \\ \\ -1, & \text{if } G^i_{i,j} = 0, G^j_{i,j} = -1 \end{cases}$$

Similarly, the Map of the agent A^j is updated by comparing the values with the grid G^i of the agent A_i

3.3. Agents within range

We can define set of agents

$$N_a = A^k | k \neq i, \quad \|P^k - P^i\| < R$$

3.4. Average position

The agent moves towards the average position of the nearby neighbours. The average position of the agents within the range of communication can be calculated as

$$\hat{P} = \frac{\Sigma_k P^k}{|N_a|}$$

The direction of the agent is towards the average heading of its neighbours. Similarly the average orientation can be calculated as

$$\hat{\theta} = \frac{\Sigma_k \theta^k}{|N_a|}$$

3.5. Velocity of the agent

The velocity of the agent A_i is given as

$$v^i_{flock(t+1)} = v^i_{flock(t)} + v_{c(t)} + v_{a(t)} + v_{s(t)}$$

where v_c, v_a, v_s are cohesion, alignment and separation velocity vectors. The alignment velocity is calculated as

$$v_a = \frac{\Sigma_k v^k_{flock}}{|N_a|}$$
$$N_a = A^k | k \neq i, \quad \|P^k - P^i\| < R_a$$

The cohesion velocity is calculated as

$$v_a = P^i - \frac{\Sigma_k P^k}{|N_a|}$$
$$N_a = A^k | k \neq i, \quad \|P^k - P^i\| < R_c$$

The agents move away from the other agents to avoid collision as in the repulsion in the likely charged particles. The separation velocity is calculated as

$$v_a = P^i - \frac{\Sigma_k P^k}{|N_a|}$$
$$N_a = A^k | k \neq i, \quad \|P^k - P^i\| < R_s$$

To limit the velocities, the velocity updation will be in the range of [V_{min}, V_{max}].

3.6. Frontiers

The coverage problem is dealt with frontiers in an unknown environment by specifying the boundary conditions of the environment. This approach uses a distributed solution that has a coverage matrix in each agent that gets shared with its neighbours. The map is decomposed into cells with a resolution and those cells are initially set as unexplored. When the agent passes through that cell, the cell is set as explored and when there is an obstacle, the value of the cell is set with a value that denotes an obstacle. A Breadth-First-Search or connected components algorithm is then used to detect the frontier regions and explore nearby frontier centroids from the robot's present position. The flowchart is given in figure 2

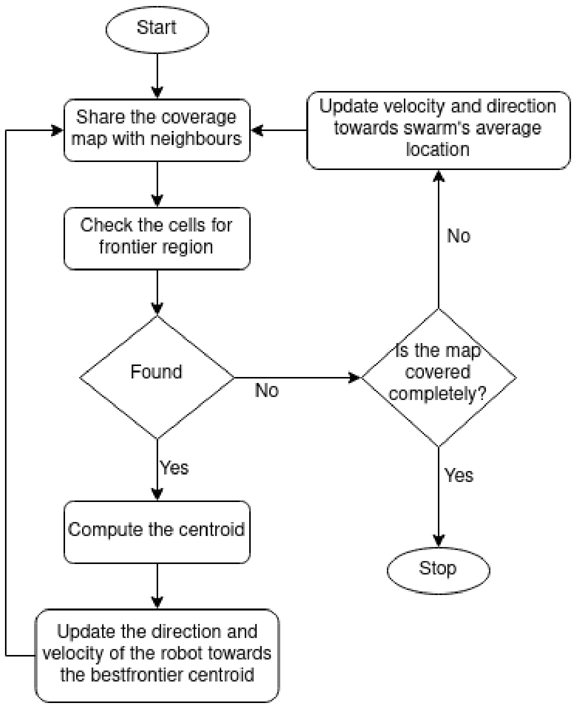
Start
Share the coverage map with neighbours
Update velocity and direction towards swarm's average location
Check the cells for frontier region
No
Found
No
Is the map covered completely?
Yes
Yes
Compute the centroid
Stop
Update the direction and velocity of the robot towards the bestfrontier centroid

Figure 2: An overview of the implementation

Any cell that is in between the explored cell $G_{i,j} = 3$ and unexplored cell $G_{i,j} = 0$ is added to the list of cell F. Given the set of frontier cells F, we find the frontier regions F_r. The centroids of all the frontier regions F_r are calculated and a position p_r is chosen such that

$$p_r = min(\|\hat{F_r^i} - p^i\|)$$

where $\hat{F_r^i}$ is the centroid of the i^{th} frontier region

3.7. Obstacle avoidance

Obstacle avoidance is achieved using repulsion force between the robots by utilizing the LIDAR sensors on the robots. The cohesion and alignment velocities are updated by the dynamic distance-based repulsive scale.

$$\delta_i = \frac{D - p^i}{R_c}$$

where D is the obstacle distance.

$$v_a^{'} = v_a * max((1 - 2\delta_i), 0)$$
$$v_c^{'} = v_c * max((1 - 2\delta_i), 0)$$

Problem 1: *Coverage of an environment*
Given a group of robot agents $A_i \; i = 1, 2, ...n$ and an unknown environment G with known boundaries, find an effective way of sharing the obstacle map asynchronously while finding frontier regions.
Problem 2: *Formation of the swarm*
Given the individual obstacle map (occupancy grid) G_i of each robot, find neighbouring agents and compute the effective heading angle and velocity towards a common frontier centroid.

4. Technical approach

A 2D map of random size is generated, featuring obstacles placed at randomized locations. Robot agents are then positioned in non-colliding, randomized locations within the map. An occupancy grid is assigned to each robot, initially representing unexplored areas. The robots initiate their exploration by searching for frontiers, initially starting at the center of their respective occupancy grids . The connected components algorithm is utilized to identify frontiers, and the robots move towards the centroid of the closest frontier, progressing one step at a time. The search region comprises eight connected sets, allowing the robots to move in any of the eight directions based on the centroid of the closest frontier. When there are no obstacles within the robot's vicinity (eight cells surrounding the robot), the occupancy grid is updated, assigning a value of 3 to denote the absence of obstacles as shown in figure 3.

Collisions between robots are effectively avoided by considering the heading action of other agents and their respective locations within the vicinity. If a robot encounters an obstacle during its search (similar to a lidar sensing), it updates its occupancy grid accordingly. In this experiment, the robot's vicinity is defined as the grid cell adjacent to its current location. The obtained results are presented in the figures below.

To prevent robots from getting stuck at the same location or redundantly exploring the same area, a random unexplored location within the frontier region is selected as the target exploration location for the agents. This accounts for the possibility of the centroid of unexplored frontiers aligning with previously explored regions.

When other agents are within proximity, the robots share their occupancy grid maps, thereby reducing the number of iterations required to explore the environment. In this experiment, if an agent is located in any of the grids surrounding another agent, they exchange information. This not only enhances exploration efficiency but also promotes cohesion and alignment by assigning the same frontier region centroid to each agent.

It is worth noting that developing the code involved integrating multiple concepts, randomizing obstacles, and creating a robust code capable of accepting user input while comparing various scenarios with different numbers of agents. The Python implementation of the 2D visualization, featuring the frontier-based search strategy for exploring unknown environments in accordance with the Boids swarm formation algorithm, can be found in the GitHub repository [18].

5. Advantages

- The algorithm effectively addresses the challenge of decentralized communication among agents operating in communication-restricted environments.

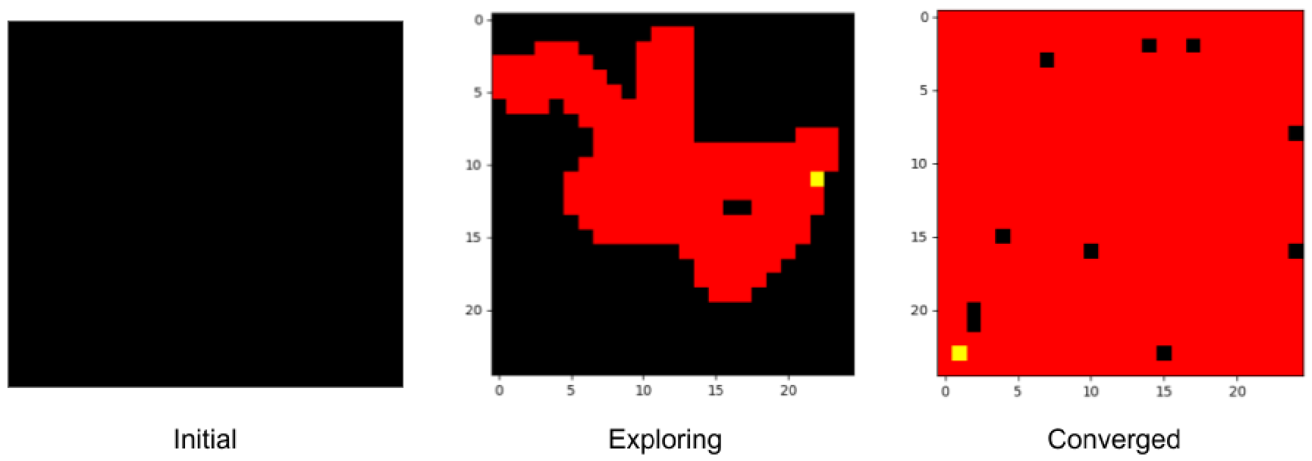

Figure 3: Grid map during different stages of exploration

- The algorithm exhibits robustness by enabling the creation of random environments tailored to user specifications, including the number of agents and obstacles on the map.
- Furthermore, the algorithm incorporates elements of the Boids algorithm to ensure cohesive swarm formation.
- The integration of the Boids algorithm within the exploration strategy contributes to cohesive movement and coordination among agents, facilitating efficient exploration.
- Notably, the algorithm achieves exploration of the environment in significantly fewer iterations compared to traditional methods such as BFS or DFS.
- The algorithm's efficiency in terms of exploration iterations contributes to its overall effectiveness and time-saving benefits.
- By leveraging decentralized communication, the algorithm enables efficient coordination and collaboration among agents, resulting in expedited exploration and increased overall performance.
- The algorithm's adaptability to user-defined environments allows for customizable simulations and experimental scenarios, enhancing its versatility and practicality.
- With its reduced iteration requirements, the algorithm offers a promising solution for time-sensitive exploration tasks, optimizing resource utilization and minimizing time-to-completion.
- The algorithm's ability to navigate through various environments, with varying numbers of agents and obstacles, showcases its adaptability and resilience in real-world applications.

6. Disadvantages

- The integration of the Boids algorithm introduces constraints on other agents, compelling them to align their heading angle and speed with the swarm. This constraint ensures that agents collectively explore new areas, leading to faster convergence.
- By incorporating the Boids algorithm and utilizing frontier regions, this algorithm may require more iterations compared to traditional methods like BFS or DFS. The random selection of unexplored regions often results in agents traversing previously visited locations before reaching their target destinations.
- It is important to note that the algorithm's convergence time is not directly compared to other exploration algorithms. Instead, the metric for evaluating performance is the number of iterations required for the set of agents to fully explore the environment.
- Comparing the algorithm's convergence time to that of other exploration algorithms may not be a direct measure of its effectiveness. Instead, the focus on iteration count acknowledges the algorithm's unique approach.

7. Problems Encountered

- The agents struck at the same location if only frontier centroids are used as the mean of the x,y coordinates of the unexplored region can result in the already explored region. This is mitigated by choosing a random location from the unexplored region.
- The random choice of the unexplored grid might have an obstacle and hence the robot has to acknowledge the target location vicinity and move to a new location.
- The visualization was difficult for each iteration and it was tackled using matplotlib animation package.

8. Results

Figure 4 provides an overview of the environment in which the agents are situated. The environment is represented by a 25x25 grid, with red regions indicating the presence of obstacles and yellow regions denoting the locations of the robots.

Moving forward, figure 5, figure 6, figure 7, and figure 8, showcase the occupancy grid of each individual agent at an intermediate step. As depicted in Figure 2, the agents begin exploring the map independently, without sharing their respective maps or adhering to the Boids algorithm due to their initial spatial separation. The different locations of the robots and their corresponding occupancy grids reflect their distinct exploratory paths. The color red represents the explored regions, white signifies obstacles, and yellow signifies the current location of each agent.

In figure 9 the individual occupancy grids of the agents are displayed. The red color represents the explored regions, white indicates obstacles, and black represents unexplored areas. Notably, this intermediate result is obtained after multiple iterations of exploration, during which the agents have moved closer to one another. Consequently, their occupancy grids exhibit a higher degree of similarity. The agents have reached a state of proximity that allows for the sharing of grid maps, thereby satisfying the cohesion condition of the Boids algorithm. As a result, the closest frontier centroid becomes nearly identical for each robot.

Figure 10 portrays the environment after the successful completion of exploration. The red color now represents the fully explored regions, while the black color denotes the obstacles present in the environment. This map can serve as a foundation for the deployment of advanced planning algorithms such as Dijkstra, A*, RRT*, Distributed Dijkstra, etc., enabling global path planning.

It is worth noting that this paper does not address the incorporation of a sensory and situational aware algorithm to handle dynamic obstacles in the environment. However, such considerations hold potential for future research and development.

8.1. Performance Evaluation

Figure 11 illustrates a grid map measuring 20x20, featuring 12 randomly placed obstacles. This particular map serves as a benchmark to assess the algorithm's convergence to a fully explored state.

In Figure 12, we observe a comparison between the number of iterations required for the algorithm to converge and the number of agents involved in map exploration. Generally, an intriguing pattern emerges, showcasing a decreasing curve as the number of agents increases. However, some exceptions arise when 3, 6, and 9 agents are engaged in exploration.

It is noteworthy that algorithms like Breadth-First Search (BFS) or Depth-First Search (DFS) demand 400 iterations to fully explore the map. In contrast, a single agent navigating this specific map necessitates over 400 iterations to converge. Yet, the number of iterations significantly decreases when employing 2 agents, outperforming BFS and DFS. Promising results can be achieved by employing 10 agents for optimistic exploration on this map.

Figure 13 displays a grid map of size 40x40, incorporating 15 randomly positioned obstacles. Similar to the previous case, this map serves as an additional reference to evaluate the algorithm's convergence.

In Figure 14, we observe the comparison between the algorithm's convergence iterations and the number of agents involved in map exploration. Similarly, the curve exhibits a decreasing trend as the number of agents increases, with

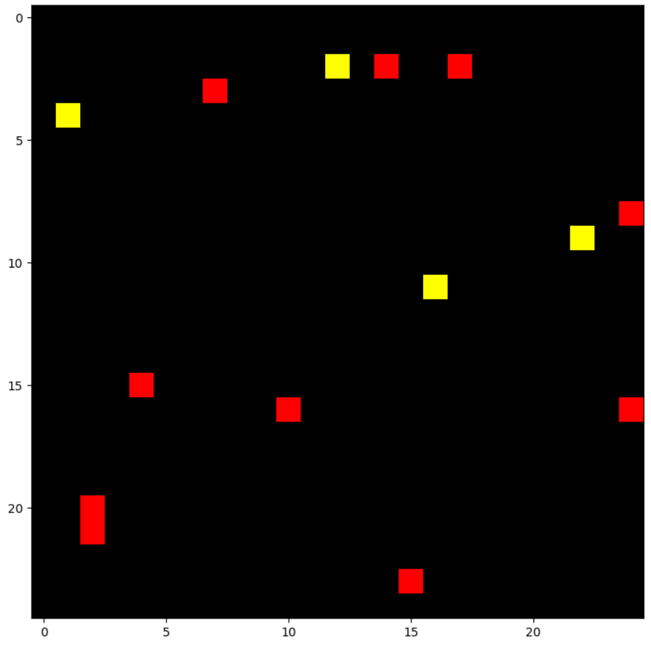

Figure 4: Initial Map that has obstacles and starting positions of the agents

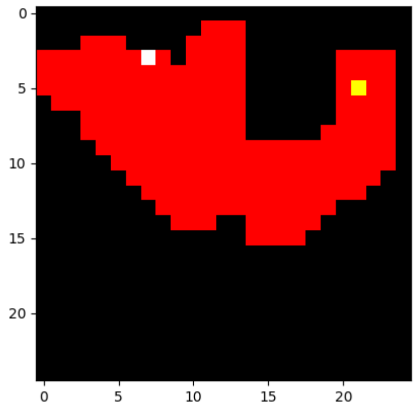

Figure 5: Agent 1 Occupancy Grid

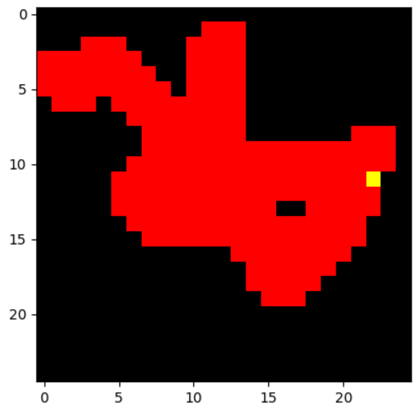

Figure 6: Agent 2 Occupancy Grid

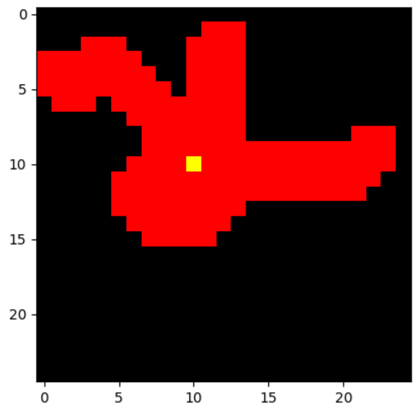

Figure 7: Agent 3 Occupancy Grid

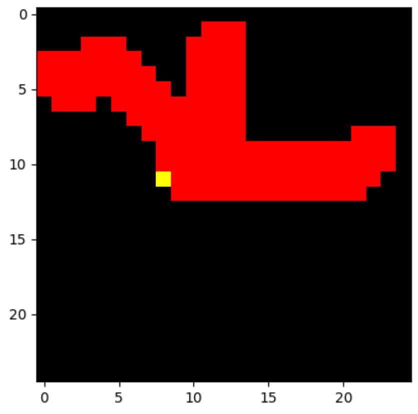

Figure 8: Agent 4 Occupancy Grid

exceptions occurring when 3 and 10 agents explore the map.

Notably, algorithms like BFS or DFS require 1600 iterations to fully explore this larger map. Conversely, a single agent navigating this map requires over 2000 iterations to converge. However, employing 2 agents drastically reduces the number of iterations, surpassing the efficiency of BFS and DFS. For this specific scenario, employing 9 agents represents the most optimistic approach.

It is essential to acknowledge that the relationship between the number of iterations and the number of agents cannot be strictly claimed as inversely proportional. The optimal number of agents required for faster convergence depends on individual cases, and the randomized initial locations of robots and obstacles significantly influence performance. However, through conducting numerous experiments with different starting locations, valuable insights can be gleaned.

Moreover, it is important to note that the comparison between this algorithm and DFS or BFS is solely based on the number of iterations, not on the computational complexity. Internally, the proposed algorithm employs BFS to identify frontier centroids.

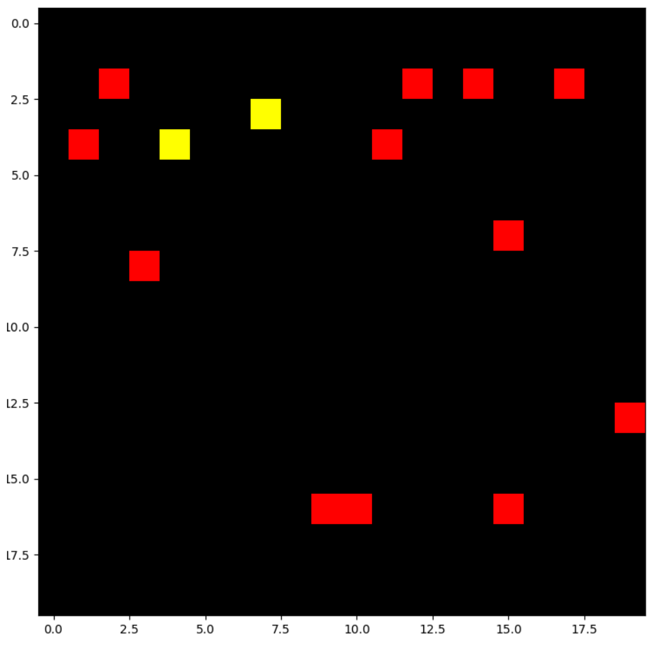

Figure 11: 20x20 Map for evaluating the performance of the algorithm using a variable number of agents

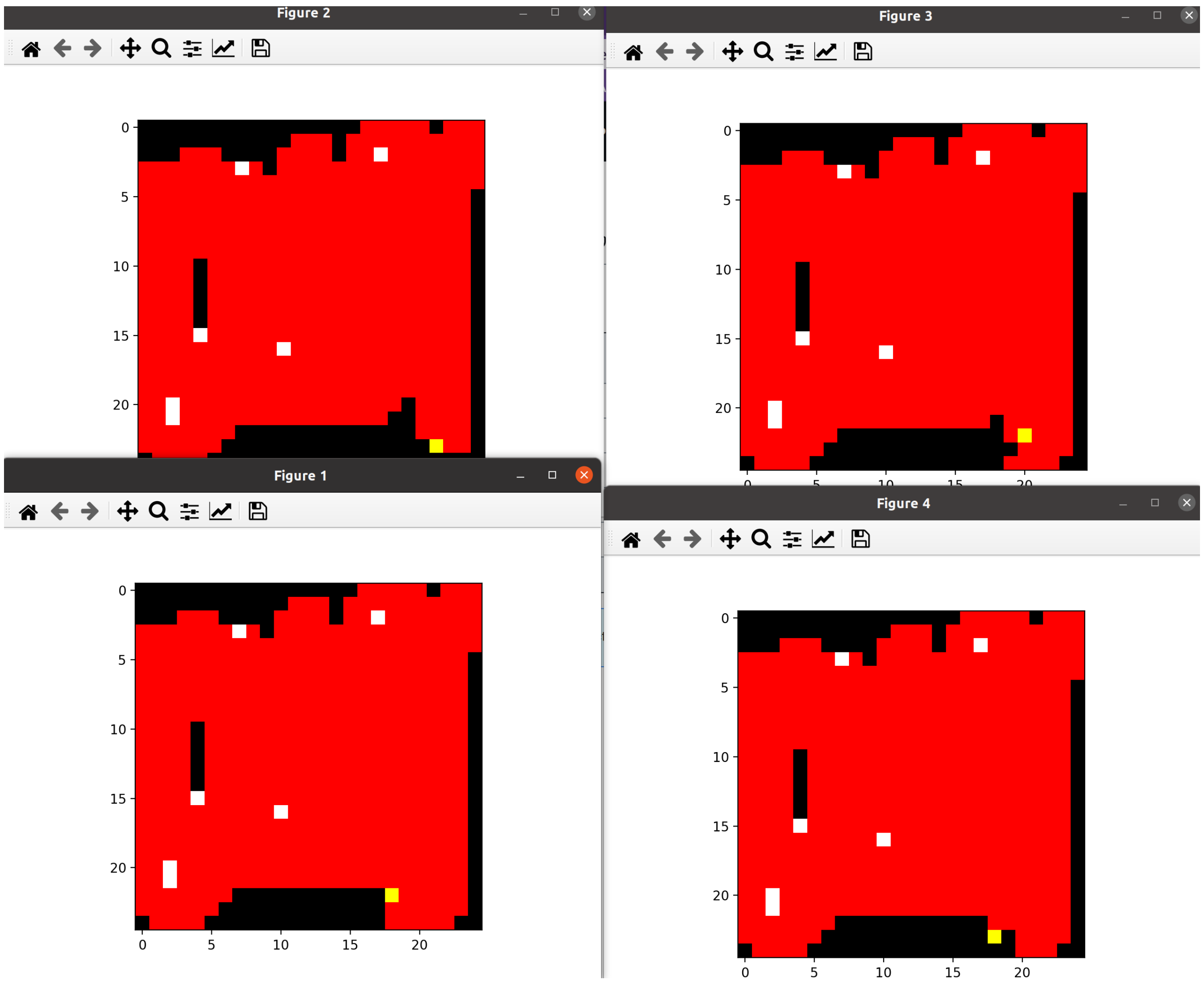

Figure 9: All the agents exploring the environment by adhering to the Boids swarm algorithm

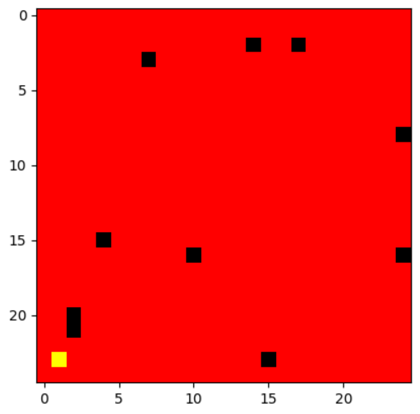

Figure 10: Occupancy grid after exploration

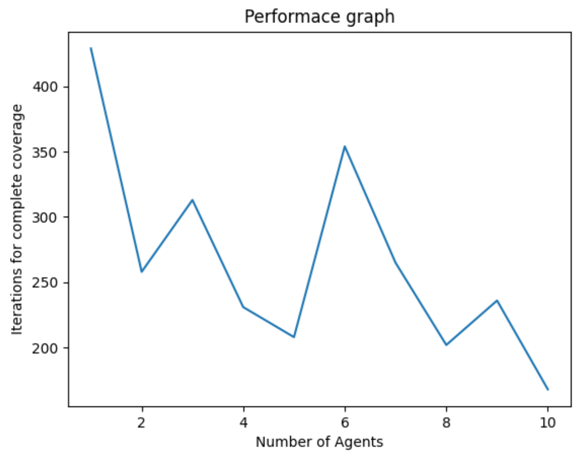

Figure 12: Performance of the algorithm for different agents on the map in figure 11

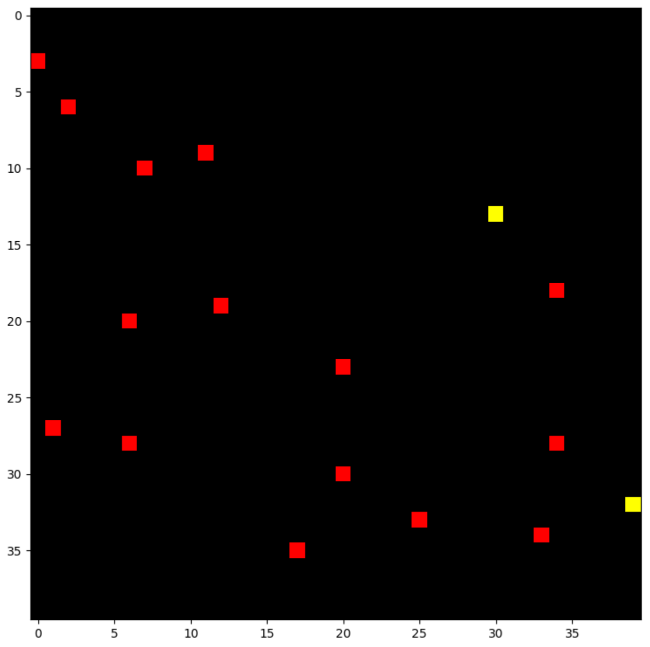

Figure 13: 40x40 Map for evaluating the performance of the algorithm using a variable number of agents

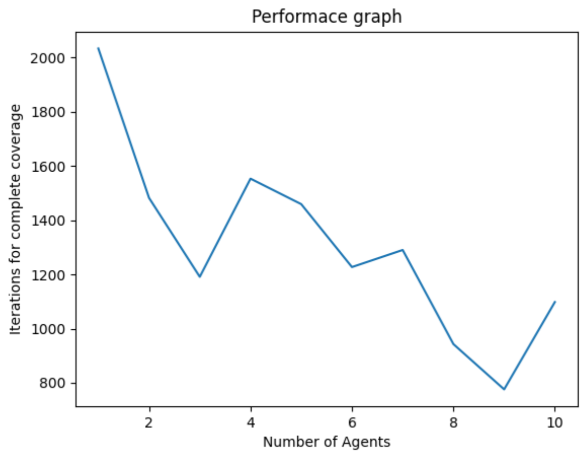

Figure 14: Performance of the algorithm for different agents on the map in figure 13

9. Future Work

The future work of this paper aims to enhance the efficiency of environment exploration by ensuring that no two agents visit the same unexplored location. To achieve this, a mechanism will be developed to facilitate asynchronous information sharing among agents when they come into proximity or within a communicable region.

To further evaluate and quantify the relationship between the time taken and the number of iterations required for convergence, a comparative analysis will be conducted using different environments and a higher number of agents. This exploration will provide valuable insights into the scalability and performance of the algorithm as the number of deployed agents increases.

Additionally, the results obtained from the proposed approach will be compared with existing state-of-the-art exploration algorithms. This comparative assessment will shed light on the strengths and weaknesses of the frontier-led swarm algorithm in relation to other established techniques.

Furthermore, the system will be optimized to accommodate a larger number of agents, surpassing the current limit. Special attention will be given to enhancing the efficiency of sharing larger maps among multiple agents to ensure smooth and effective coordination.

In order to validate the practicality and real-world applicability of the proposed approach, the algorithm will be deployed on physical robots or implemented in a simulation environment. This practical implementation will provide valuable insights into the algorithm's behavior in real-world scenarios, moving beyond the constraints of point robots and accounting for the complexities and challenges encountered in real-world robotic systems.

10. Conclusion

The decision to employ a 2D visualization approach for the implementation of the paper by V.P. Tran et al. [1] was motivated by the author's intent to construct the algorithms from the ground up, rather than relying on pre-existing packages within the Robot Operating System (ROS) and obtaining immediate results. This choice allowed for a deeper understanding of the underlying principles and mechanics of the frontier-led swarm algorithm.

With Python as the programming language of choice, the frontier-led swarm algorithm was meticulously developed from scratch. This hands-on implementation enabled the authors to gain comprehensive insights into the intricacies of the algorithm and its behavior in different scenarios. By conducting a comparative analysis, a valuable understanding of the relationship between the time taken and the number of agents was established, shedding light on the scalability and efficiency of the proposed approach.

A key focus of this project was to create a robust codebase capable of accommodating user input and generating random obstacle and agent placements for the initial exploration phase. This versatility empowers users to adapt the algorithm to various environments and scenarios, promoting flexibility and applicability across different settings.

By taking a comprehensive approach to the implementation process, integrating user inputs, and employing Python for the development, the authors successfully contributed to the advancement of frontier-led swarm algorithms and gained valuable insights into their performance. This work serves as a foundation for further research and exploration in the field of multi-agent exploration and highlights the significance of meticulous algorithmic implementation and comparative analysis.

References

[1] V. P. Tran, M. A. Garratt, K. Kasmarik, and S. G. Anavatti, “Robust Multi-Robot Coverage of Unknown Environments using a Distributed Robot Swarm,” CoRR, vol. abs/2111.14295, 2021, [Online]. Available: https://arxiv.org/abs/2111.14295

[2] T. Cieslewski, S. Lynen, M. Dymczyk, S. Magnenat and R. Siegwart, ”Map API - scalable decentralized map building for robots,” 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 2015, pp. 6241-6247, doi: 10.1109/ICRA.2015.7140075.

[3] Puente-Castro, A., Rivero, D., Pazos, A. et al. UAV swarm path planning with reinforcement learning for field prospecting. Appl Intell 52, 14101–14118 (2022). https://doi.org/10.1007/s10489-022-03254-4

[4] M. Zhong and C. G. Cassandras. Distributed coverage control and data collection with mobile sensor networks. IEEE Transactions on Automatic Control, 56(10):2445–2455, 2011.

[5] M. Masár. A biologically inspired swarm robot coordination algorithm for exploration and surveillance. In 2013 IEEE 17th International Conference on Intelligent Engineering Systems (INES), pages 271–275, 2013.

[6] K. Cheng, Y. Wang, and P. Dasgupta. Distributed area coverage using robot flocks. In 2009 World Congress on Nature Biologically Inspired Computing (NaBIC), pages 678–683, 2009.

[7] A. Topiwala, P. Inani, and A. Kathpal, “Frontier Based Exploration for Autonomous Robot,” CoRR, vol. abs/1806.03581, 2018, [Online]. Available: http://arxiv.org/abs/1806.03581

[8] Reynolds, Craig W. ”Flocks, herds and schools: A distributed behavioral model.” Proceedings of the 14th annual conference on Computer graphics and interactive techniques. 1987.

[9] Bhattacharya, Priyadarshi, and Marina L. Gavrilova. ”Roadmap-based path planning-using the voronoi diagram for a clearance-based shortest path.” IEEE Robotics Automation Magazine 15.2 (2008): 58-66.

[10] Wang, Huijuan, Yuan Yu, and Quanbo Yuan. ”Application of Dijkstra algorithm in robot path-planning.” 2011 second international conference on mechanic automation and control engineering. IEEE, 2011.

[11] Karaman, Sertac, et al. ”Anytime motion planning using the RRT.” 2011 IEEE international conference on robotics and automation. IEEE, 2011.

[12] Virágh, Csaba, et al. ”Flocking algorithm for autonomous flying robots.” Bioinspiration biomimetics 9.2 (2014): 025012.

[13] Meerza, Syed Irfan Ali, Moinul Islam, and Md Mohiuddin Uzzal. ”Optimal Path Planning Algorithm for Swarm of Robots Using Particle Swarm Optimization Technique.” 2018 3rd International Conference on Information Technology, Information System and Electrical Engineering (ICITISEE). IEEE, 2018.

[14] Y. Zheng, L. Wang and P. Xi, ”Improved Ant Colony Algorithm for Multi-Agent Path Planning in Dynamic Environment,” 2018 International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC), Xi’an, China, 2018, pp. 732-737, doi: 10.1109/SDPC.2018.8664885.

[15] David Paez, Juan P. Romero, Brian Noriega, Gustavo A. Cardona, Juan M. Calderon, Distributed Particle Swarm Optimization for Multi-Robot System in Search and Rescue Operations, IFAC-PapersOnLine, Volume 54, Issue 4, 2021, Pages 1-6, ISSN 2405-8963, https://doi.org/10.1016/j.ifacol.2021.10.001.

[16] D. Albani, J. IJsselmuiden, R. Haken and V. Trianni, ”Monitoring and mapping with robot swarms for agricultural applications,” 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Lecce, Italy, 2017, pp. 1-6, doi: 10.1109/AVSS.2017.8078478.

[17] D. Albani, J. IJsselmuiden, R. Haken and V. Trianni, ”Monitoring and mapping with robot swarms for agricultural applications,” 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Lecce, Italy, 2017, pp. 1-6, doi: 10.1109/AVSS.2017.8078478.

[18] K. R. Oruganti, https://github.com/okritvik/Frontier-based-coverage-using-multiple-agents