CMPE 297: Final Project Report: Multi-Robot Task Assignment

Kapil Wanaskar [016649880]

May 2024

1 INTRODUCTION

1.1 Motivation and Objective

The rapidly advancing field of robotics has seen an increasing integration of multi-robot systems in various applications, from industrial automation to surveillance and exploration tasks. The challenge lies in efficiently managing and coordinating these robots to perform complex tasks collaboratively without human intervention. This project addresses the multi-robot task assignment problem, specifically focusing on optimizing path planning and ensuring collision avoidance. The primary objectives of this project are:

- To develop a system that can dynamically assign tasks and plan paths for multiple robots in a two-dimensional space, with an extension to three-dimensional spaces as a bonus feature.
- To ensure that these paths are optimized for both efficiency and safety, incorporating robust collision avoidance mechanisms.
- To provide real-time online visualization of the robots' movements, enhancing the system's usability and monitoring capabilities.

The ultimate goal is to create a framework that not only supports effective task allocation among multiple robots but also facilitates smooth step-by-step movement, ensuring that operational constraints such as collision avoidance are met.

1.2 Related Work

The conceptual framework of this project is supported by several key pieces of literature in the domains of path planning, multi-robot coordination, collision avoidance, visualization, and practical applications. Here are some of the notable contributions:

1.2.1 Path Planning

- Kuhn's (1955) seminal work on the Hungarian method provides a foundational algorithm for assignment problems, which is crucial for optimal task allocation in multi-robot systems [1].
- Koenig and Likhachev (2002) introduced the D* Lite algorithm, an essential contribution to real-time path planning and navigation, which can be adapted for dynamic multi-robot environments [2].

1.2.2 Multi-Robot Coordination

- Gerkey and Matarić (2004) offer a formal analysis and taxonomy of task allocation that helps in understanding the various strategies that can be implemented in multi-robot systems [3].
- Dias et al. (2006) discuss market-based approaches for multi-robot coordination, providing insights into the complexities of decentralized decision-making in robotic swarms [4].

1.2.3 Collision Avoidance

- Van Den Berg et al. (2011) describe an n-body collision avoidance system, which is directly applicable to ensuring the safety of robots while navigating shared spaces [5].
- Alonso-Mora et al. (2018) focus on efficient shared control mechanisms for collision avoidance in autonomous multi-robot systems, highlighting strategies for dynamic environment adaptation [6].

1.2.4 Visualization

- Kritzinger et al. (2015) explore the role of the Digital Twin in manufacturing, which parallels the need for accurate simulation and visualization in multi-robot systems [7].
- Crick et al. (2017) provide a comprehensive review of multi-robot systems with a focus on the challenges and frameworks, including those related to visualization [8].

1.2.5 Applications

- Choudhury et al. (2018) and Coppola et al. (2020) discuss the practical applications of multi-robot systems in service robots and micro air vehicles respectively, underscoring the broad applicability and potential challenges of implementing these systems in real-world scenarios [9] [10].

These works collectively inform the design and development of the proposed multi-robot task assignment system by providing proven methodologies and highlighting existing gaps in technology that this project aims to address.

2 System Design & Implementation Details

2.1 Algorithm(s) Considered/Selected

The mathematical model employs a step-by-step approach to move robots towards their targets while avoiding collisions. This implementation draws inspiration from path planning and collision avoidance methodologies:

2.1.1 Generate Random Position

- $\mathbf{p}$: Position vector of a point, where the first element represents the x-coordinate and the second element represents the y-coordinate.
- $x_{\min}, x_{\max}$: Minimum and maximum limits for the x-coordinate.
- $y_{\min}, y_{\max}$: Minimum and maximum limits for the y-coordinate.
- rand(): A function that generates a random number between 0 and 1.

A random position $\mathbf{p}$ within specified limits $[x_{\min}, x_{\max}]$ and $[y_{\min}, y_{\max}]$ is defined as:

$$\mathbf{p} = \begin{bmatrix} x_{\min} + (x_{\max} - x_{\min}) \cdot \text{rand}() \\ y_{\min} + (y_{\max} - y_{\min}) \cdot \text{rand}() \end{bmatrix}$$

Example:

$$\mathbf{p} = \begin{bmatrix} 0 + (100 - 0) \cdot 0.5 \\ 0 + (100 - 0) \cdot 0.7 \end{bmatrix} = \begin{bmatrix} 50 \\ 70 \end{bmatrix}$$

2.1.2 Position Validity

- $\mathbf{p}_{\text{new}}$: The new position being checked for validity.
- $\mathbf{p}$: Existing positions against which the new position is being checked.
- $d_{\min}$: The minimum required distance between any two positions.

A position $\mathbf{p}_{\text{new}}$ is valid if it is at least a minimum distance $d_{\min}$ away from any existing position $\mathbf{p}$ in a set of positions:

$$\text{valid}(\mathbf{p}_{\text{new}}) = \begin{cases} 1 & \text{if } \|\mathbf{p}_{\text{new}} - \mathbf{p}\| \geq d_{\min} \text{ for all } \mathbf{p} \\ 0 & \text{otherwise} \end{cases}$$

Example: Given $\mathbf{p}_{\text{new}} = \begin{bmatrix} 10 \\ 10 \end{bmatrix}$, $\mathbf{p} = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$, and $d_{\min} = 10$:

$$\|\mathbf{p}_{\text{new}} - \mathbf{p}\| = \sqrt{(10-5)^2 + (10-5)^2} = \sqrt{25+25} = \sqrt{50} \approx 7.07$$

Thus, $\mathbf{p}_{\text{new}}$ is not valid.

2.1.3 Move Towards Target

- $\mathbf{p}$: Current position of an object.
- $\mathbf{t}$: Target position to which the object is moving.
- s: Step size, the maximum distance the object can move in one update.

To move a position $\mathbf{p}$ towards a target $\mathbf{t}$ with a step size s, the updated position is:

$$\mathbf{p}_{\text{updated}} = \mathbf{p} + \frac{s}{\|\mathbf{t} - \mathbf{p}\|} \min(s, \|\mathbf{t} - \mathbf{p}\|)(\mathbf{t} - \mathbf{p})$$

Example: $\mathbf{p} = \begin{bmatrix} 20 \\ 20 \end{bmatrix}$, $\mathbf{t} = \begin{bmatrix} 25 \\ 25 \end{bmatrix}$, $s = 5$:

$$\mathbf{p}_{\text{updated}} = \begin{bmatrix} 20 \\ 20 \end{bmatrix} + 5 \cdot \frac{\begin{bmatrix} 25-20 \\ 25-20 \end{bmatrix}}{\sqrt{(25-20)^2 + (25-20)^2}} = \begin{bmatrix} 20 \\ 20 \end{bmatrix} + \begin{bmatrix} 5 \\ 5 \end{bmatrix} = \begin{bmatrix} 25 \\ 25 \end{bmatrix}$$

2.1.4 Collision Avoidance

- $\mathbf{p}$: Position of one object.
- $\mathbf{q}$: Position of another object which might be too close to the first.
- $d_{\min}$: Minimum allowable distance between any two objects to avoid collision.

To adjust $\mathbf{p}$ for collision avoidance when too close to another position $\mathbf{q}$:

$$\mathbf{p} = \mathbf{p} - 0.5 \times \frac{d_{\min}}{\|\mathbf{q} - \mathbf{p}\|}(\mathbf{q} - \mathbf{p})$$

$$\mathbf{q} = \mathbf{q} + 0.5 \times \frac{d_{\min}}{\|\mathbf{q} - \mathbf{p}\|}(\mathbf{q} - \mathbf{p})$$

Example: $\mathbf{p} = \begin{bmatrix} 20 \\ 20 \end{bmatrix}$, $\mathbf{q} = \begin{bmatrix} 21 \\ 21 \end{bmatrix}$, $d_{\min} = 5$:

$$\mathbf{p} = \begin{bmatrix} 20 \\ 20 \end{bmatrix} - 0.5 \cdot \frac{5}{\sqrt{(21-20)^2 + (21-20)^2}} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 19.5 \\ 19.5 \end{bmatrix}$$

$$\mathbf{q} = \begin{bmatrix} 21 \\ 21 \end{bmatrix} + 0.5 \cdot \frac{5}{\sqrt{(21-20)^2 + (21-20)^2}} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 21.5 \\ 21.5 \end{bmatrix}$$

2.1.5 Check All Reached Targets

- $\mathbf{p}$: Current position of a robot or moving object.
- $\mathbf{t}$: Target position that the robot is supposed to reach.
- ϵ: A small threshold distance within which the position is considered as having reached the target.

All robots have reached their targets if:

$$\text{reached} = \begin{cases} 1 & \text{if } \|\mathbf{p} - \mathbf{t}\| \leq \epsilon \text{ for all } (\mathbf{p}, \mathbf{t}) \\ 0 & \text{otherwise} \end{cases}$$

Example: If $\mathbf{p} = \begin{bmatrix} 30 \\ 30 \end{bmatrix}$ and $\mathbf{t} = \begin{bmatrix} 30.4 \\ 30.4 \end{bmatrix}$, and $\epsilon = 0.5$, then:

$$\|\mathbf{p} - \mathbf{t}\| = \sqrt{(30.4 - 30)^2 + (30.4 - 30)^2} = \sqrt{0.16 + 0.16} \approx 0.4$$

Thus, the target is reached.

The reason for selecting these algorithms is their straightforward applicability to the multi-robot motion simulation, allowing for real-time adjustments based on the current state of the robot positions and their targets.

2.2 Technologies & Tools Used

- **Python:** Chosen for its extensive libraries and ease of use in numerical calculations, simulations, and data handling.
- **NumPy:** Utilized for efficient numerical computations, especially for handling arrays and matrix operations crucial in the calculations of positions and distances between robots.
- **Matplotlib:** Used for creating static, interactive, and animated visualizations in Python, making it an ideal choice for visualizing the paths and movements of robots.
- **Streamlit:** Selected for building the interactive application because of its ability to rapidly develop and deploy data applications. It also supports real-time manipulation of data through widgets like sliders and buttons.
- **FFmpeg (via Matplotlib Animation):** Used for saving animations, which allows the visualization of robot movements over time in a video format.

2.3 Handling Large Datasets

Not mentioned in given sources: The provided materials do not cover specific methods for dealing with large datasets.

2.4 System Design/Architecture/Data Flow/Workflows

2.4.1 System Design

The core of the system design involves initializing the simulation environment, updating robot positions based on set algorithms, and checking for the completion of tasks.

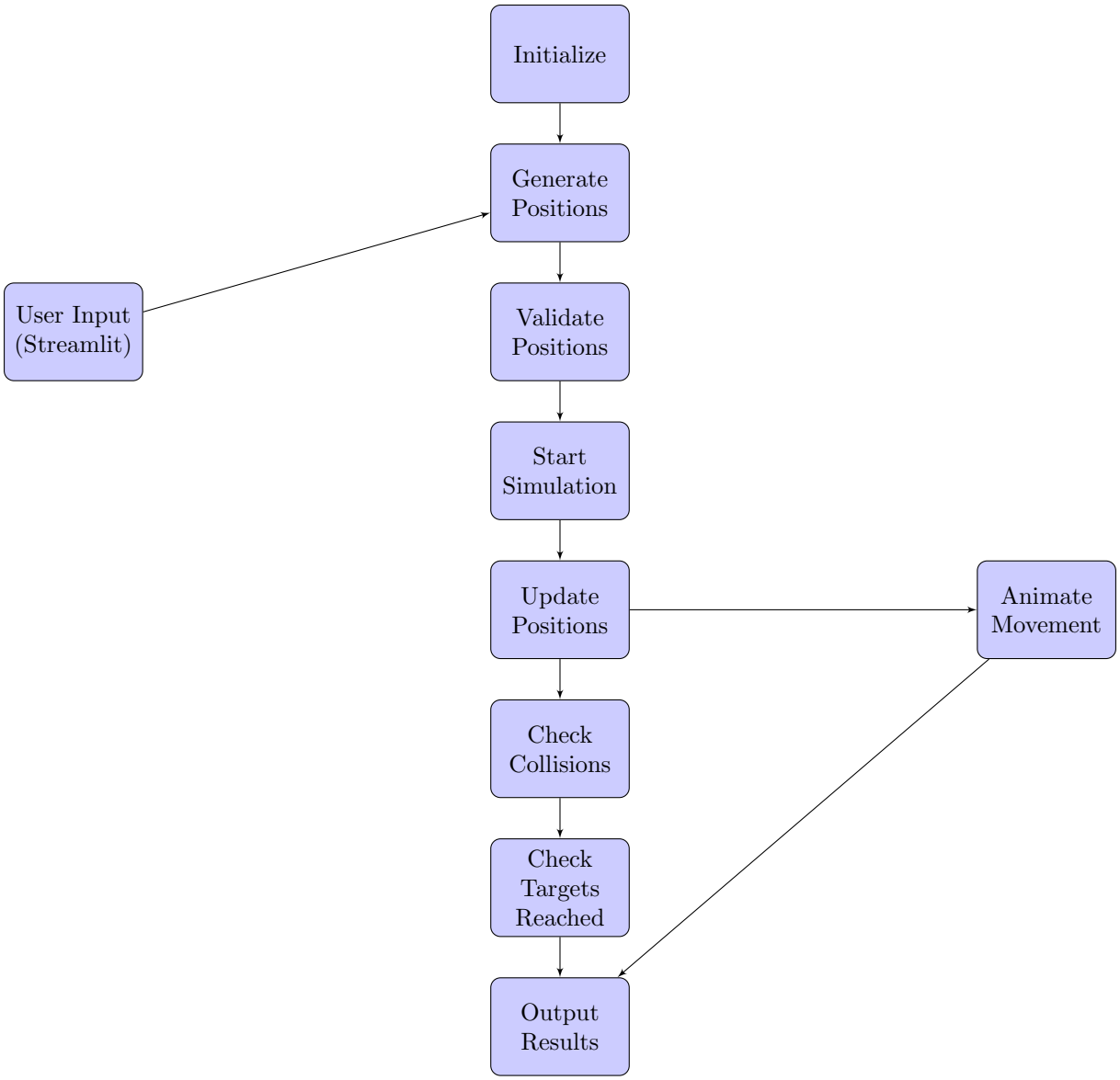

Figure 1: System Architecture of the Multi-Robot Simulation

2.4.2 Data Flow

Initialization: Set the initial and target positions for all robots using a random generation method that ensures no initial collisions.

Simulation Loop:

1. For each time step, calculate the next position for each robot based on their target positions while employing collision avoidance mechanisms.
2. Update the positions in the visualization.
3. Check if all robots have reached their targets.

Completion: Once all targets are reached, the simulation ends, and the results are displayed.

2.5 Visualization/UI/GUI/Screenshots

2.5.1 Streamlit Application Interface

The UI is designed using Streamlit, featuring sliders for real-time parameter adjustments (e.g., number of robots, space limits, and movement radius). Buttons are provided to start the 2D or 3D simulations. The simulations can be visualized directly in the Streamlit interface through embedded Matplotlib figures or through generated video files displayed in the UI.

2.5.2 Screenshots/Visualization

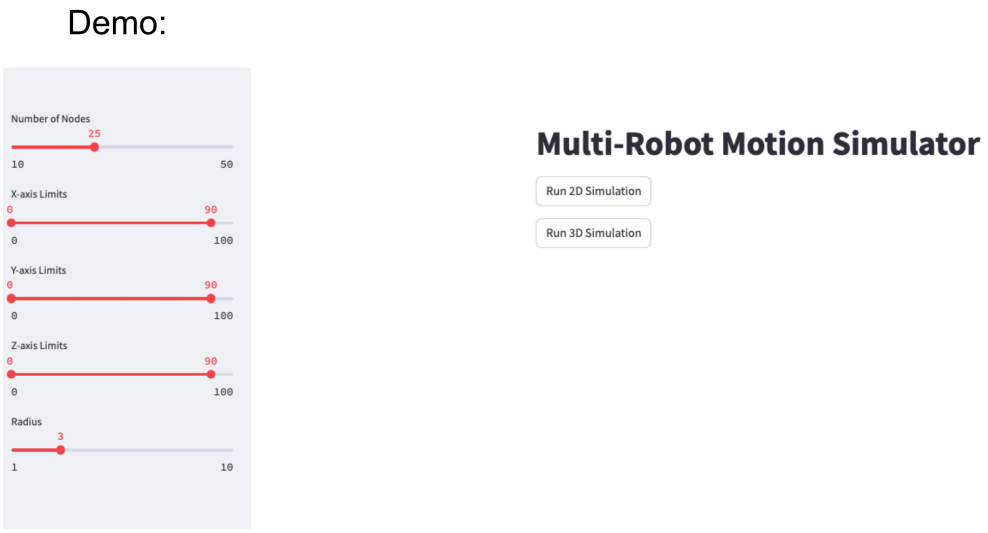

Figure 2: Step 1: Initial Setup in Streamlit showing parameter adjustments for simulation.

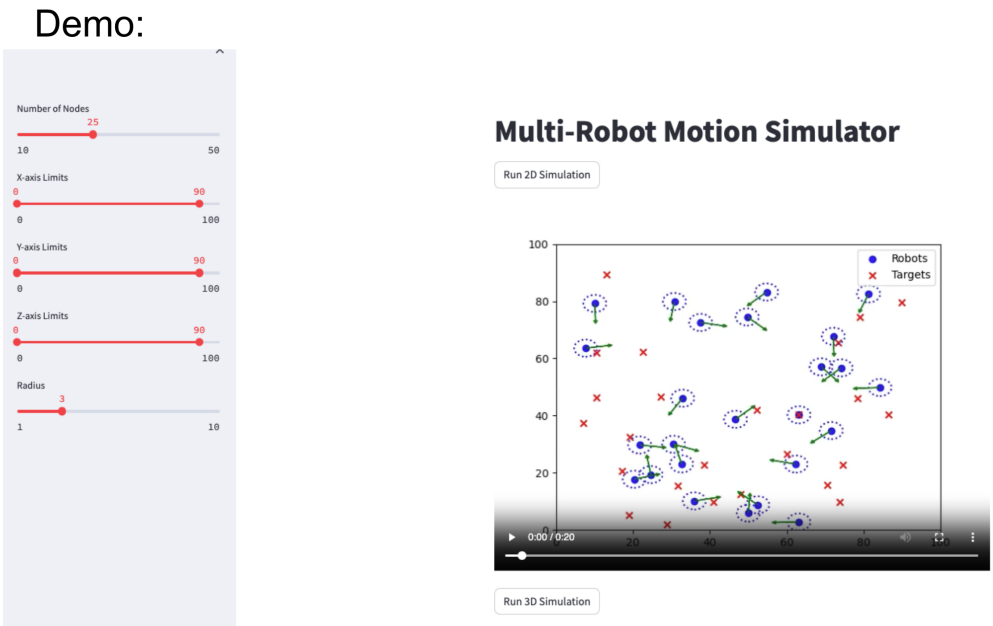

Figure 3: Step 2: Running a 2D simulation in Streamlit, demonstrating the interactive simulation environment.

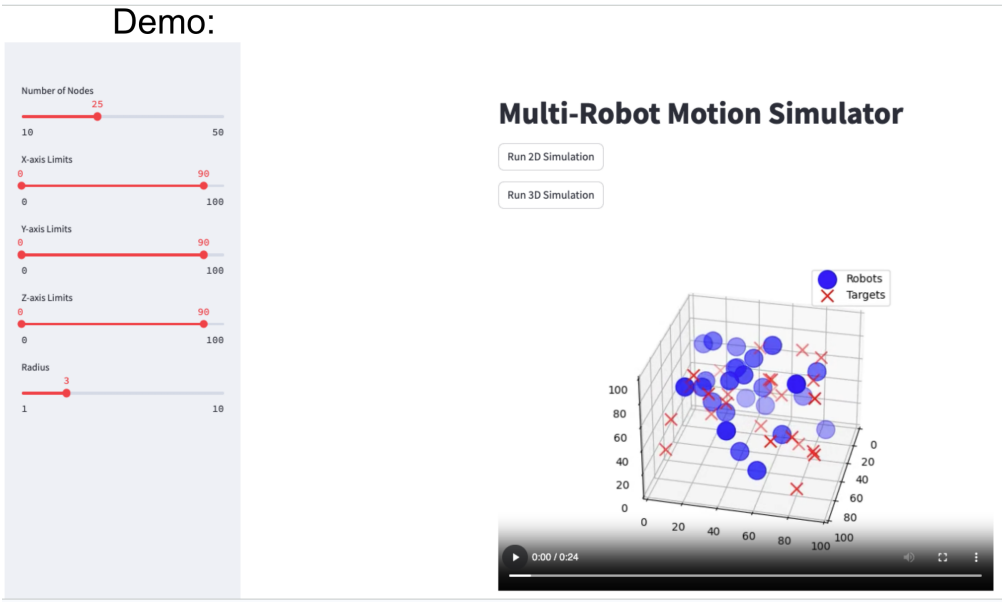

Figure 4: Step 3: Running a 3D simulation in Streamlit, showcasing complex spatial interactions between robots.

By integrating these elements, the designed system efficiently manages the task of simulating multi-robot path planning and collision avoidance in a visually engaging and user-interactive manner. This approach not only serves the functional requirements but also enhances the user experience by providing real-time control and feedback on the simulation parameters and outcomes.

3 Experiments / Proof of Concept Evaluation

3.1 Dataset(s) Used

Not mentioned in given sources: The sources do not provide information about specific datasets used. The system generates its data dynamically for simulation purposes, such as initial and target positions of robots within specified spatial limits.

3.2 Datasets Details/Patterns

Not mentioned in given sources: There is no mention of specific patterns or statistical details of datasets since the project generates random positions for simulation.

3.3 Data/Results Visualization, Online Interactive (Preferred)

The visualization and user interaction are primarily handled through the Streamlit application. This setup allows users to manipulate simulation parameters like the number of robots, space limits, and radius for collision avoidance through sliders. The movements of robots are visualized in both 2D and 3D formats. Here are the stages of development demonstrating the capabilities of the simulation:

Result 1: 2D Robots as Single Point

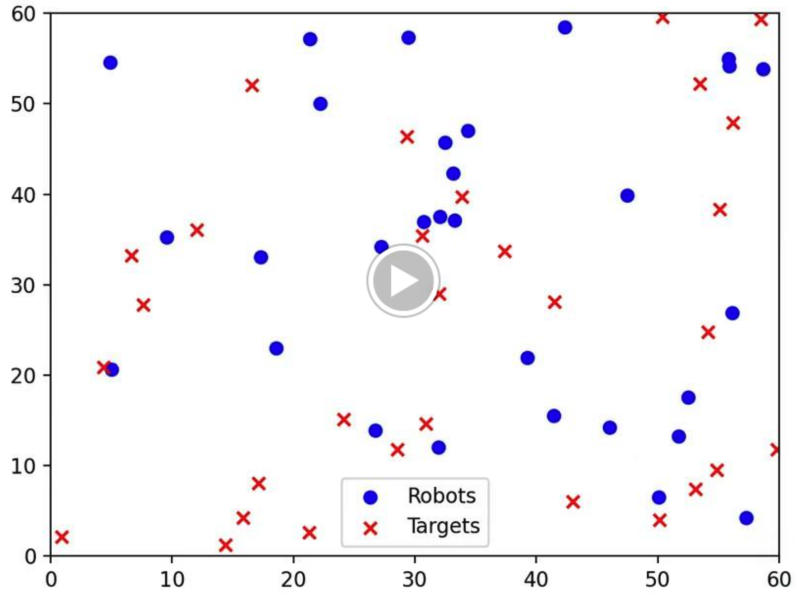

Figure 5: Initial visualization in 2D where each robot is represented as a single point. View the video here.

Result 2: 2D Robots with Circular Area

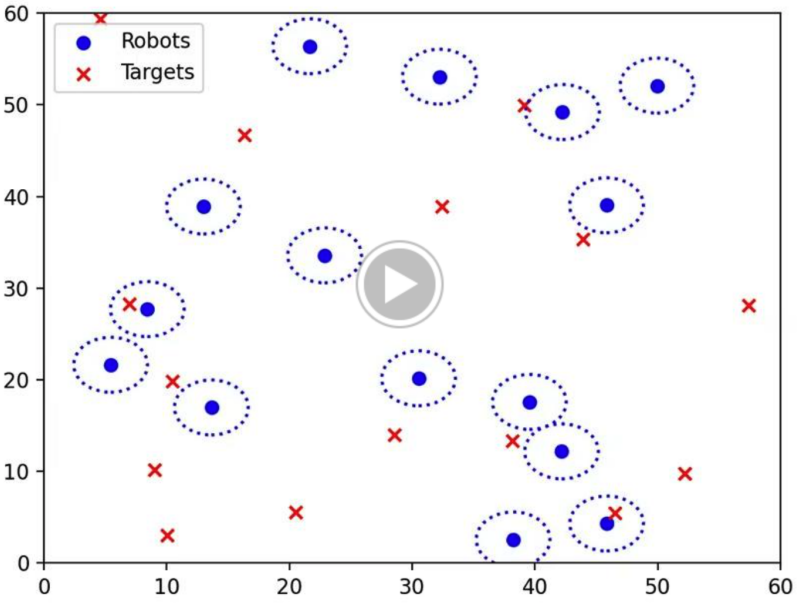

Figure 6: Enhanced visualization in 2D showing robots with a defined circular area indicating their collision radius. View the video here.

Result 3: 2D Robots with Vector Showing Current Direction

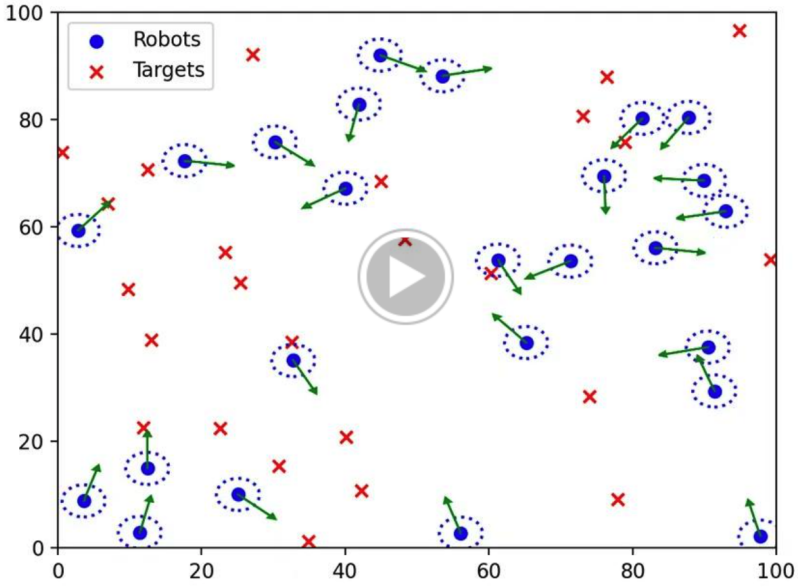

Figure 7: 2D visualization with vectors showing the current movement direction and intended path of each robot. View the video here.

Result 4: 3D Robots as Sphere

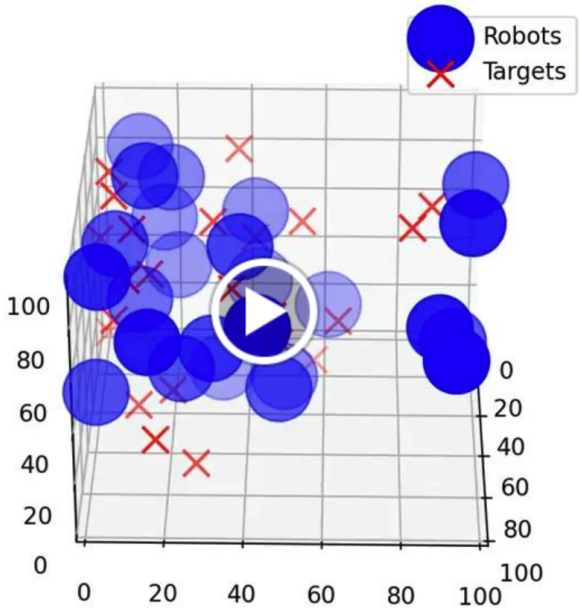

Figure 8: 3D visualization where each robot is represented as a sphere, providing a spatial view of their distribution. View the video here.

Result 5: 3D Spherical Robots with Vector Showing Current Direction

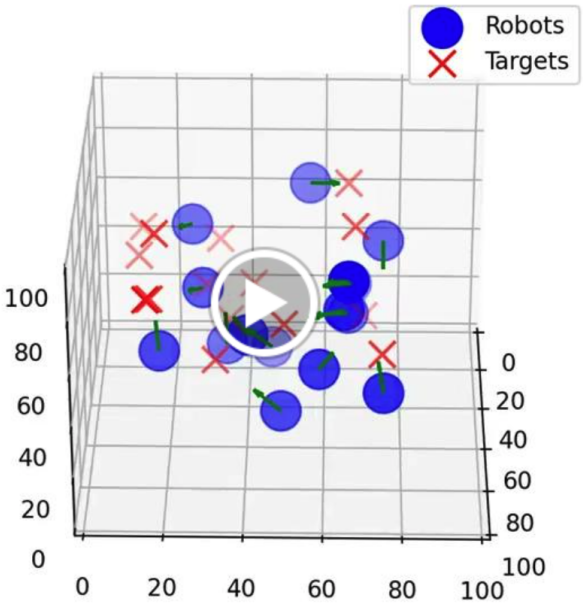

Figure 9: Advanced 3D visualization illustrating both the spherical shape of the robots and vectors indicating the direction of movement. View the video here.

3.4 Data Preprocessing Decisions

Not mentioned in given sources: There are no explicit preprocessing decisions described as the data required for the simulations (positions of robots) are generated in real-time and used directly without preprocessing.

3.5 Evaluation Methodology

Not mentioned in given sources: The sources do not detail specific methodologies like n-fold cross-validation as the project does not involve traditional machine learning datasets or model training processes.

3.6 Graphs Showing Different Parameters/Algorithms Evaluated in a Comparative Manner

Not mentioned in given sources: There are no comparative graphs or analysis provided about different algorithms or parameters. The implementation directly applies the designed algorithms in the simulation without a comparative evaluation setup.

3.7 Analysis of Results

Not mentioned in given sources: There is no analytical discussion provided on the outcomes of the simulations. The primary focus is on demonstrating the functionality of the path planning and collision avoidance algorithms through the visualizations generated by the Streamlit app.

In summary, the experiment section revolves around the real-time simulation of multi-robot systems using the developed application. The evaluation is visual and interactive rather than based on statistical analysis or comparison of various algorithms. The main proof of concept is demonstrated through the

effective visualization of robot movements and the successful avoidance of collisions, confirming the practical applicability of the algorithms in controlled settings.

4 Discussion & Conclusions

4.1 Decisions Made/Things That Worked

The project involved several key decisions that significantly contributed to its success:

- **Algorithm Selection:** The choice of using straightforward path planning and collision avoidance algorithms allowed for a clear demonstration of the system's capabilities without the complexities of more sophisticated algorithms. This approach made it easier to understand and visualize the fundamental dynamics of multi-robot coordination and obstacle navigation.
- **Technology Stack:** Utilizing Python, NumPy, Matplotlib, and Streamlit provided a robust framework for developing and visualizing the simulation. Python and NumPy facilitated efficient computations, while Matplotlib enabled the creation of dynamic visualizations. Streamlit was particularly effective for creating an interactive user interface, allowing users to adjust parameters in real-time and immediately see the impact of these changes in the simulation.
- **Interactive Visualization:** Implementing the system in Streamlit with real-time interactive controls (sliders and buttons) for simulation parameters proved to be very effective. It enhanced user engagement and provided a clear visual understanding of how different parameters affect the movement and coordination of robots.

4.2 Difficulties Faced/Things That Didn't Work Well

- **Complexity in 3D Visualization:** While the 2D simulations were straightforward and effective, extending these visualizations to 3D brought additional complexity. Ensuring that the 3D movements were as clear and comprehensible as the 2D animations proved challenging, particularly in maintaining clarity in the web-based interface provided by Streamlit.
- **Real-time Performance:** As the number of robots increased, the real-time performance of the simulation began to lag, especially when running more complex scenarios with many robots and narrow collision avoidance thresholds. This performance issue highlighted the need for further optimization of the simulation algorithms or the use of more powerful computational resources.
- **Algorithm Scalability:** The basic algorithms used for path planning and collision avoidance, while effective for a small number of robots, did not scale well with increasing numbers. This limitation could impact the applicability of the system in more practical, larger-scale operations.

4.3 Conclusions

The project successfully demonstrated a basic framework for simulating multi-robot task assignment and collision avoidance in both 2D and 3D environments. The use of an interactive web application for visualization proved to be an effective way to demonstrate the capabilities and to interactively explore the effects of different system parameters.

Key Conclusions:

- **Proof of Concept:** The system effectively showed that even simple algorithms could manage basic task assignments and collision avoidance in a simulated multi-robot environment.
- **Scalability and Performance:** There are notable challenges with scalability and real-time performance that need addressing to enhance the system's applicability to more extensive and complex scenarios.

- **Potential for Further Development:** There is substantial scope for integrating more advanced algorithms, optimizing performance, and extending the system's capabilities to handle larger datasets and more robots.

In summary, this project lays the groundwork for further research and development in multi-robot systems, highlighting both the potential and the challenges of current methodologies and technologies. Future work could focus on optimizing existing algorithms, exploring new methods for handling larger scales, and enhancing the user interface to accommodate more complex simulations.

5 Future Plan / Task Distribution

5.1 Pros and Cons of the Current Method & Future Plan

5.1.1 Pros

- **Simplicity and Clarity:** The current method's straightforward approach allows for easy understanding and implementation, which is particularly beneficial for educational and demonstration purposes.
- **Interactive Visualization:** Utilizing Streamlit for real-time parameter adjustments and visualization provides an engaging way to interact with and understand the system's dynamics.
- **Scalability in Concept:** The basic framework is designed to be scalable in concept, providing a foundation upon which more complex features can be built.

5.1.2 Cons

- **Scalability in Practice:** While the concept is scalable, the actual performance and efficiency in handling larger numbers of robots or more complex environments are limited.
- **Limited Real-World Application:** The current algorithms are basic and may not perform well under the varied and unpredictable conditions found in real-world applications.
- **Performance Issues:** The system can experience performance lags as the complexity of the task increases, which could be a significant hindrance in deploying this system for real-time applications.

5.1.3 Future Plan

To address the limitations noted above and enhance the system's capabilities, the future plan includes:

- **Algorithm Enhancement:** Integrating more advanced path planning and collision avoidance algorithms that are better suited for dynamic and unpredictable environments.
- **Performance Optimization:** Investigating more efficient data structures and optimizing the code to improve the system's performance, especially for larger simulations.
- **Extended Testing and Validation:** Implementing comprehensive testing scenarios to validate the system under various conditions and ensuring robustness and reliability.
- **3D Visualization Improvements:** Enhancing the 3D visualization tools to make them more intuitive and clearer, which will help in better understanding and analyzing the robots' movements in three-dimensional space.

5.2 Task Distribution

Since I am the sole member of this project, the task distribution was as follows:

5.2.1 Assigned Tasks

- **Algorithm Development:** I was responsible for designing and implementing the path planning and collision avoidance algorithms.
- **System Implementation:** I handled all aspects of coding the simulation, including setting up the Streamlit interface and integrating the back-end algorithms with the front-end display.
- **Testing and Validation:** I conducted initial tests to ensure that the system performed as expected under various parameter settings.
- **Documentation and Reporting:** I prepared all project documentation, including writing the chapters of the project report and ensuring that all aspects of the system were well-documented.

5.2.2 Completed Tasks

All the tasks listed were completed by me, given that I am working alone on this project. This included everything from the initial research and design phase through to the final testing and documentation.

As the project moves forward, I will continue to handle all aspects of its development, from refining the algorithms to enhancing the user interface. The solo nature of this project allows for cohesive and consistent progress but also requires careful management of time and resources to ensure all aspects of the system are developed to their full potential.

6 REFERENCES

References

[1] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83-97, 1955.

[2] S. Koenig and M. Likhachev, "D* Lite," in *AAAI/IAAI*, pp. 476-483, 2002.

[3] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939-954, 2004.

[4] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1257-1270, 2006.

[5] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research*, Springer, Berlin, Heidelberg, pp. 3-19, 2011.

[6] J. Alonso-Mora, S. Bransen, P. W. Lieths, A. Sanfeliu, and R. Siegwart, "Efficient shared control for fully autonomous multi-robot coordination," *arXiv preprint arXiv:1809.08830*, 2018.

[7] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital Twin in manufacturing: A categorical literature review and classification," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 1016-1022, 2015.

[8] C. Crick, E. Munoz, and S. Osentoski, "A Comprehensive Review of Multi-Robot Systems: Taxonomies, Frameworks, and Challenges," *arXiv preprint arXiv:1708.08615*, 2017.

[9] S. Choudhury, A. Kapoor, G. Ranade, D. Arora, and S. Narayanan, "Adaptive formation control for multiple service robots," in *Robot World Cup*, Springer, Cham, pp. 3-14, 2018.

[10] M. Coppola, K. N. McGuire, C. De Wagter, and G. C. de Croon, "A survey on swarming with micro air vehicles: Fundamental challenges and constraints," *Frontiers in Robotics and AI*, vol. 7, p. 18, 2020.