Autonomous Exploration of Unknown Environments Using Dubins Path-Oriented RRT*

Hands-on Planning Course Project

Mohamed Khaled Othman, Goitom Abrha Leaku, Eliyas Kidanemariam Abraha,

***Abstract*—This paper presents a enhanced software architecture designed for autonomous exploration and navigation of differential drive robots in unknown environments. The architecture employs a frontier-based exploration algorithm that integrates Rapidly-exploring Random Trees (RRT*) with a Dubins state space, enabling the robot to generate collision-free paths that respect dynamic constraints. Utilizing high-resolution 2D Lidar, the robot achieves accurate obstacle detection and mapping. The path planning module optimizes global exploration through a comprehensive cost function and post-processing techniques, while a greedy algorithm efficiently selects exploration frontiers to minimize movement costs. The low-level control is handled by an Adaptive Pure Pursuit Controller, ensuring smooth and precise navigation. The versatility of the architecture is demonstrated through extensive testing in both real-world and simulated environments, with results showing significant improvements in exploration efficiency and path quality. This research advances the field of autonomous robotics by addressing the complexities of navigating uncharted environments and providing practical solutions for real-world applications.**

***Index Terms*—Frontier-Based Exploration, RRT*, Dubins Path, Adaptive Pure Pursuit Controller.**

I. Introduction

THE Autonomous exploration and navigation in unknown environments pose significant challenges for robotics, particularly for differential drive robots that must navigate dynamically and efficiently in real time. This paper introduces an enhanced software architecture tailored for such robots, integrating advanced technologies to enhance their autonomous capabilities. At the core of this system is a frontier-based exploration strategy that leverages Rapidly-exploring Random Trees (RRT*), customized with a Dubin's state space to ensure the generation of feasible and collision-free paths.

The robot's perception is powered by a high-resolution 2D Lidar, enabling precise detection and mapping of obstacles. The path planning component of the architecture utilizes RRT* for global exploration, optimized through a comprehensive cost function and post-processing methods to refine path quality. A greedy algorithm is employed to select the next frontier to explore, focusing on reducing movement costs by prioritizing locally interesting locations. This strategic approach ensures efficient coverage of unknown areas.

For navigation, the Adaptive Pure Pursuit Controller is used to provide smooth and accurate movement, adapting to the dynamic conditions of the environment. The versatility of this software architecture is validated through rigorous testing in both simulated and real-world scenarios, demonstrating its robustness and effectiveness.

By addressing the intricate challenges of autonomous exploration in uncharted territories, this research contributes valuable insights and practical solutions to the field of autonomous robotics, paving the way for more advanced and reliable exploration systems.

II. Related Works

Several research efforts have focused on enhancing autonomous exploration and navigation in unknown environments, integrating advanced exploration algorithms, sensors, path-planning, and control strategies.The concept of frontier-based exploration was pioneered by Brian Yamauchi, who implemented it on a Nomad 200 mobile robot using a laser rangefinder and sonar sensors, with a Breadth-First Search (BFS) algorithm to navigate frontiers efficiently [1].

Liu et al. advanced this approach with an incremental caching topology-grid hybrid map (TGHM), combining topology and grid maps for optimized exploration and navigation. The approach further improved frontier-based exploration by incorporating robot heading information and coarse graph representation, which increased efficiency in cluttered environments [2].

Johnson et al. applied frontier-based strategies to autonomous drones, using the Rapidly-exploring Random Trees (RRT) algorithm and sensors like LiDAR and RGB-D cameras for effective mapping and navigation in complex terrains[3]. Additionally, Chen et al. integrated frontier-based exploration with Adaptive Monte Carlo Localization (AMCL) for robust navigation in underground mines, utilizing multi-modal sensors for precise obstacle detection and mapping [4].

On the aspect of low-level control, various strategies have been employed to ensure smooth and accurate navigation. For instance, Proportional-Integral-Derivative (PID) controllers are commonly used to maintain stable and responsive control of the robot's movements. These controllers are crucial for implementing the planned paths and achieving reliable real-time navigation in dynamic and unpredictable environments. These advancements highlight the ongoing evolution and effectiveness of frontier-based exploration methods across various robotic platforms and applications.

III. Methodology

Autonomous exploration in unknown environments involves expanding a robot's map by guiding it towards unexplored regions using frontier-based strategies. This section outlines the techniques and algorithms used in our approach illustrated in Figure 4, including efficient frontier detection, clustering, and selection, as well as path planning using RRT* with Dubins curves and low-level control for precise navigation.

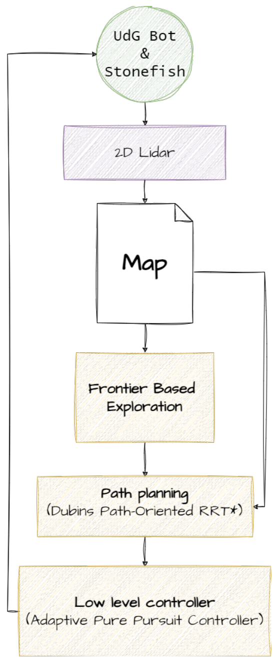

UdG Bot & Stonefish
2D Lidar
Map
Frontier Based Exploration
Path planning (Dubins Path-Oriented RRT*)
Low level controller (Adaptive Pure Pursuit Controller)

Fig. 1: Exploration Workflow

A. Frontier Based Exploration

Exploration seeks to expand the map by directing robots towards unexplored regions using frontier-based strategies. Frontier edges, which separate known and unknown areas on the map, guide the robot's exploration. When a frontier edge is identified, the robot is assigned a point, typically the centroid, to explore. However, processing the entire map to extract frontier edges can be computationally intensive, especially as the map grows in size. Therefore, efficient methods are being developed to detect frontier edges and optimize computational resources.Figure 2 shows the overview of frontier detection , clustering , and viewpoint selection processed on image.

1) Frontier Detection: convolution search had been used to efficiently detect frontiers in a grid map, and this process is preceded by applying dilation to the map. Dilation expands the boundaries of obstacles to account for the robot's size, ensuring that the frontiers are safely navigable. This reduces computation time by using the convolution operation, which works by applying a convolution kernel to the dilated grid map to examine each cell's neighbors. The kernel sums the values of the surrounding cells to identify those that border unknown areas. Cells that are free spaces and have at least one adjacent unknown cell are marked as frontiers. By leveraging dilation followed by convolution, this approach quickly identifies these boundary cells, optimizing the robot's exploration by focusing on the most informative regions while minimizing computational overhead.Figure 2b shows detected frontiers using convolution on image.

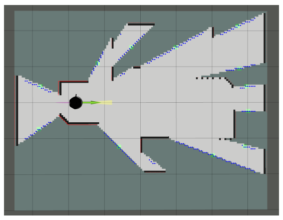
2) Frontier Clustering: After detecting frontiers the *scikit-image* library had been used to cluster the detected frontiers. The clustering process involves labeling the frontier map to identify distinct frontier regions. This is done using functions that assign unique labels to each connected component in the map, helping to differentiate separate frontier areas. Next, the properties of these labeled regions, such as size and shape, had been extracted to filter out insignificant frontiers. Only regions with an area larger than a specified threshold are considered valid 2c. The viewpoints(2d) for each valid frontier were determined by selecting the median point of the connected frontier cells.This method ensures that the chosen viewpoint is not only part of the frontier but also representative of its spatial distribution.

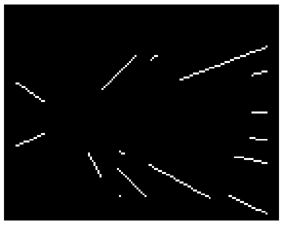
3) Frontier Selection: Frontier selection plays a crucial role in autonomous robot exploration, determining the trajectory and efficiency of the robot's path through its environment. By strategically identifying and prioritizing frontiers, the robot can navigate effectively, maximizing the coverage of unexplored areas while minimizing redundant movements.In our implementation, we employ a comprehensive cost function to select frontiers, considering factors such as distance, orientation, size, and density of clustered frontiers. Here's an overview of the selection process Initially, frontiers beyond a certain distance threshold from the robot are prioritized for selection. If there are no frontiers within this threshold distance, all frontiers are considered for selection.

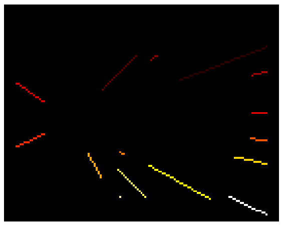
- Distance Cost: The cost associated with navigating to the frontier, which involve factors such as path length.
- Size of the Cluster: Larger clusters of frontiers are assigned a lower cost, as they potentially offer more exploration opportunities and information gain.
- Orientation to the Cluster: Frontiers that align well with the robot's current orientation are favored, as they facilitate smoother navigation and reduce the need for excessive turns.

B. Path Planning

In our approach for path planning in a differential constraint system, the Rapidly-exploring Random Tree Star (RRT*) algorithm is integrated with the Dubins Curve algorithm. This hybrid method, known as RRT*-based Dubins-Curve-connecting RRT*, constructs a tree by connecting new sample points to the existing tree using predetermined Dubins curves. Once the allotted time for tree construction has elapsed, the established tree serves as a roadmap from the root to the goal. From this tree, we extract the best path from the root to the goal and output it. This path represents a feasible trajectory that satisfies the system's constraints and navigates from the starting point to the desired goal. Our planning


(a) grid map


(b) detected frontiers


(c) clustered frontiers

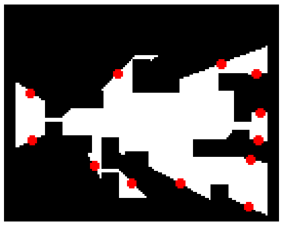

(d) frontier viewpoint

Fig. 2: Frontier Detection and Clustering

Implementation contains the following sub-implementations.

1) State Validity Checker : To ensure the validity of positions or paths concerning the environment, we utilize the StateValidityChecker class. This class examines the validity of a given position or path by utilizing an occupancy map, a 2D array that characterizes the environment's occupancy. The map's resolution and origin are specified. The StateValidityChecker class determines collision-freeness by converting world positions to map coordinates and evaluating the occupancy in the surrounding area. Moreover, the check path() method discretizes a path and verifies the validity of each position along the path.

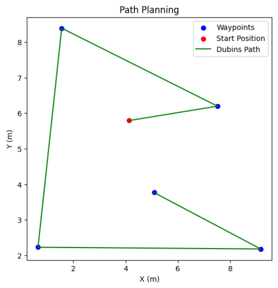
2) RRT Star Planner: A rapidly exploring random tree (RRT) is an algorithm designed to efficiently search nonconvex, high-dimensional spaces by randomly building a space-filling tree. The tree is constructed incrementally from samples drawn randomly from the search space and is inherently biased to grow towards large unsearched areas of the problem.RRT* (Rapidly-exploring Random Tree Star) improves upon the basic RRT algorithm by focusing on path optimality. Unlike RRT, which quickly finds feasible paths, RRT* refines paths over time to converge on the optimal solution through a rewiring mechanism. This mechanism allows the algorithm to reconnect nodes to shorter or more cost-effective paths, continuously optimizing the tree structure.

3) Dubins Curve : A Dubins curve is a path segment that a vehicle can follow while maintaining a constant forward velocity and a minimum turning radius, without reversing direction. These curves consist of a combination of straight line segments and circular arcs, specifically designed for vehicles that cannot make sharp turns or go backward.

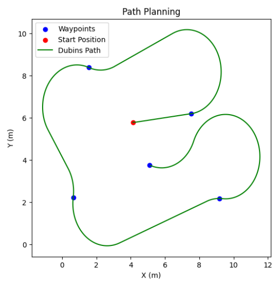
C. Dubins Path optimization

By implementing the angle to the next waypoint based on the slope (i.e., the direction from the current point to the next waypoint) makes resulting paths are smooth , continuous and easy robot to follow.

Now we have RRT and Dubins curve, next step is to combine them together so that it generates the paths that car-like robot can run on shown as in figure 3b. In the RRT algorithm, points in tree are connected by straight line figure 3a and the obstacle collision detection is running on the staight line too.But in this project, there is not straight line. Every points are connected by Dubins curve. So,when we build the tree, we add curves as edges into the graph.



(a) Straight line connection



(b) Dubins curve connection

D. Low-level Control

A low-level controller executes commands to follow a planned trajectory, managing the robot's motion along a path generated by a high-level controller, specifically using RRT* with Dubins path. The Pure Pursuit controller and its extension, Adaptive Pure Pursuit, are implemented to control the robot's angular velocity while maintaining constant linear speed. Using the robot's current pose and a goal point, the Pure Pursuit controller iterates to steer the robot toward a look-ahead point as illustrated by [5], while the Adaptive Pure Pursuit also adjusts linear speed based on orientation. Tuning the look-ahead distance impacts performance: a shorter distance improves tracking and quick recovery but may cause oscillations, whereas a longer distance provides a smoother path following with larger curvatures near corners.

E. Experimental Setup

The Kobuki TurtleBot3 is equipped with various components, including a Raspberry Pi microcontroller, a 2D lidar sensor for environment sensing, an IMU for orientation estimation, and wheel encoders for odometry. We used Kobuki TurtleBot3 to test the simulation results that we obtained in the real experiment. Before trying on the real robot, the Stonefish simulator was utilized in simulation experiments to replicate the robot's movement, physical environment, and sensor readings. In addition, for visualization, RVIZ was used.

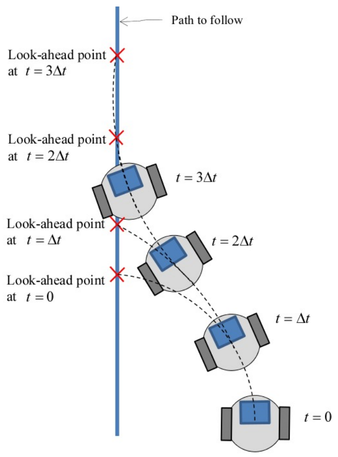
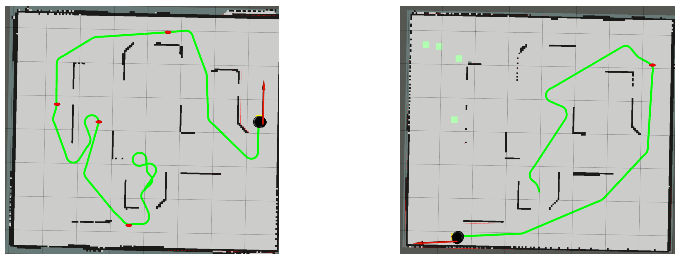

Path to follow
Look-ahead point at $t = 3\Delta t$
Look-ahead point at $t = 2\Delta t$
Look-ahead point at $t = \Delta t$
Look-ahead point at $t = 0$
$t = 3\Delta t$
$t = 2\Delta t$
$t = \Delta t$
$t = 0$

Fig. 4: Pure Pursuit controller

IV. Results and Discussion

A. Frontier selection

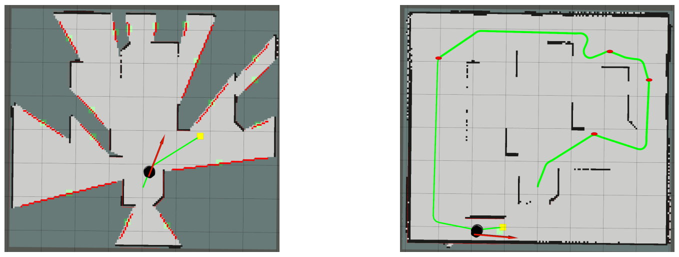
- Figure 5a illustrates frontier selection based on minimum distance, where the robot explores the nearest frontiers, resulting in four distinct viewpoints and requiring at least four planning steps plus intermediate replanning, completing the exploration in 240 seconds. This approach allows efficient local coverage but involves frequent replanning. In contrast, Figure 5b shows frontier selection based on maximum distance, where the robot targets the most distant frontier, resulting in only one primary planning viewpoint but extending the total exploration time to 350 seconds due to frequent intermediate replanning as obstacles are encountered. While this method reduces initial planning steps, it increases exploration time due to the dynamic need for path adjustments. Balancing these strategies is crucial: minimum distance selection enhances local efficiency, whereas maximum distance selection promotes broader coverage but requires more dynamic replanning, suggesting a hybrid approach may optimize overall exploration performance.



(a) Frontier selection based on minimum distance (b) Frontier selection based on longer distance

Fig. 5: Effect of selecting frontier based on distance

- As figure 6 shows in addition to minimum distance, by including orientation, size (information gain), we enhance exploration efficiency. This approach directs the robot to frontiers aligned with its current orientation, prioritizes clusters with higher information gain. This method improves exploration time by optimizing the robot's path and decision-making process, ensuring more efficient and comprehensive coverage of the area.

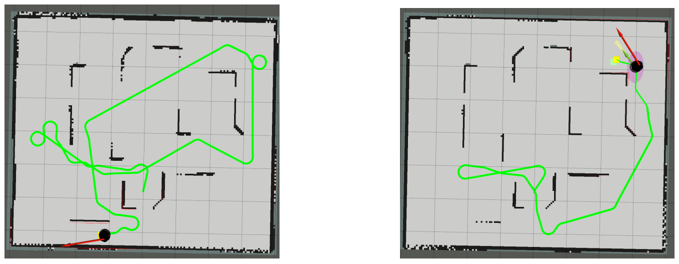


(a) before exploration (b) efficient exploration

Fig. 6: Incorporate different frontier selection method to enhance exploration

B. Planning Parameters

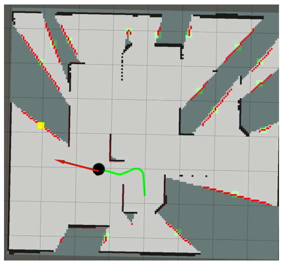
- Number Of Iterations:Increasing the number of iterations in the RRT* algorithm results in better path quality and closer approximation to the optimal path but at the cost of higher computational time and memory usage. Fine-tuning the number of iterations in our RRT* algorithm implementation is crucial for balancing path length and computation time.Due to the randomness inherent in the RRT* algorithm, results can vary between runs. However, generalized conclusions can be drawn from multiple tests.With 1000 iterations, we achieved a path length of 28.38 units in 239 seconds (Figure 3a), whereas increasing to 4000 iterations reduced the path length to 20.96 units but required 338 seconds (Figure 3b). These results highlight the trade-off between improved path quality and increased computation time, emphasizing the need for careful adjustment of iterations to meet specific application constraints.

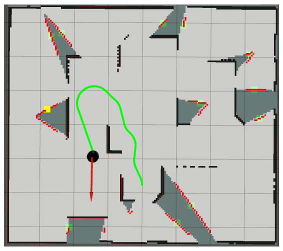


(a) 1000 number of iteration (b) 4000 number of iteration

Fig. 7: Effect of Number of iteration in path planning

- Dubins Turning Radius : Testing different Dubins radius reveals that the turning radius significantly affects path feasibility, length, and smoothness. Smaller radii offer better maneuverability in tight spaces but result in sharper turns, while larger radius provide smoother paths more suitable for open areas and higher speeds. Balancing these factors is essential for effective path planning, considering both the environment and vehicle capabilities. Figure 8a shows a path using a turning radius of 0.2 and figure 8b shows a path using 0.4 turning radius to the same goal point, shows the difference in sharpens and smoothness of those paths. Figure 8c shows that using a minimum turning radius of 0.4 units results in a smooth path but fails to find feasible paths in environments with sharp turns.


(a) 0.2 turning radius


(b) 0.4 turning radius

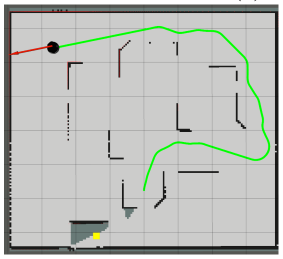

(c) 0.4 turning rad difficult to find path to the next goal

Fig. 8: Effect of dubins turning radius in path planning

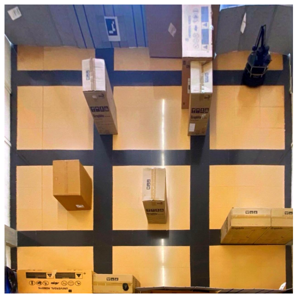
C. Real-time Implementation

The methodology outlined in the previous sections was successfully applied to a real-world robotic platform, yielding impressive results. Figure 9 demonstrates the accuracy of the map generated by the real robot when compared to ground truth data. However, transitioning from simulation to the real robot presented several challenges, including:

- Controller Frequency: The controller frequency was increased to handle fast real-world dynamics, as the initial settings were adequate for simulation but insufficient for real environments, requiring faster response times for accurate and stable navigation.
- Speed Tuning: Adjusting the robot's speed was essential, as higher speeds degraded lidar map quality, leading to mapping inaccuracies. Balancing speed with the need for reliable sensor data was crucial for effective exploration.
- Handling Unreachable Frontiers: The system occasionally detected unreachable frontiers due to environmental noise or errors outside map boundaries. Filtering out these false positives ensured efficient path planning focused on valid exploration targets.

V. Conclusion and Future Work

This research introduces an enhanced software architecture for autonomous exploration and navigation of differential drive robots in unknown environments. It integrates frontier-based exploration with RRT* and Dubins state space for optimal path


(a) Map top view

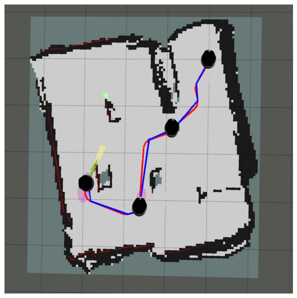

(b) RVis Explored Map

Fig. 9: Real-time implementation environment

planning. Using a high-resolution 2D Lidar for accurate obstacle detection and mapping, the system shows significant improvements in exploration efficiency and path quality through extensive testing in both simulated and real-world scenarios. The Adaptive Pure Pursuit Controller ensures smooth and precise navigation, demonstrating the architecture's versatility and robustness. This work advances autonomous robotics by addressing complex navigation challenges and offering practical solutions for real-world applications.
Future work will integrate advanced sensor fusion techniques to improve obstacle detection accuracy and develop adaptive algorithms to dynamically adjust exploration strategies based on real-time environmental feedback.

VI. Acknowledgment

The authors thank their supervisor Prof. Palomeras Rovira, Narcissus for his invaluable support and guidance throughout this research.

VII. References

[1] B. Yamauchi, ”A frontier-based approach for autonomous exploration,” Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on, Monterey, CA, 1997, pp. 146- 151. doi: 10.1109/CIRA.1997.613851

[2] W. Gao, M. Booker, A. Adiwahono, M. Yuan, J. Wang, and Y. Yun, ”An improved Frontier-Based Approach for Autonomous Exploration,” 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), pp. 292-297, 2018, doi: 10.1109/ICARCV.2018.8581245

[3] Johnson, A. et al. ”Autonomous Exploration of Unknown Terrains Using Frontier-Based Exploration Strategy.” IEEE Robotics and Automation Letters, vol. 5, no. 2, 2020.

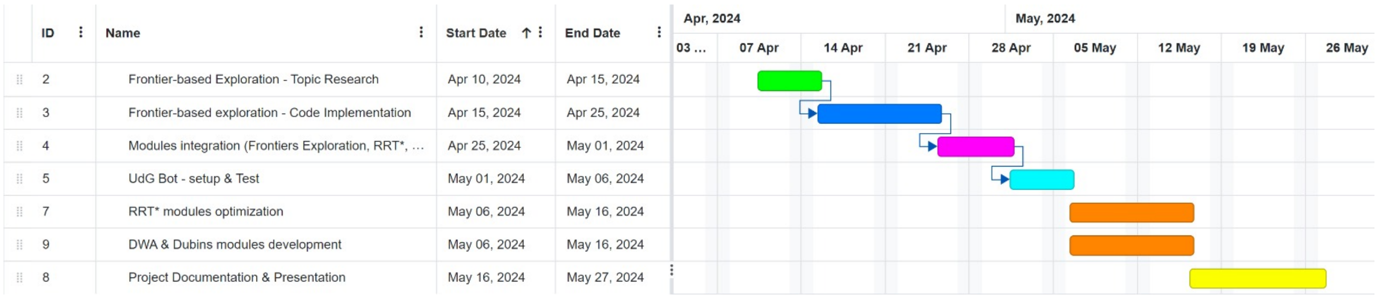
[4]Chen, B. et al. ”Autonomous Exploration in Underground Mines Using a Mobile Robot.” IEEE Transactions on Robotics, vol. 36, no. 4, 2020.

[5]Kumar, Dr. Neerendra & Vamossy, Zoltan. (2018). Robot navigation with obstacle avoidance in unknown environment. International Journal of Engineering & Technology. 7. 2410-2417. 10.14419/ijet.v7i4.14767.

VIII. Appendix

Link to simulation and real time exploration video result

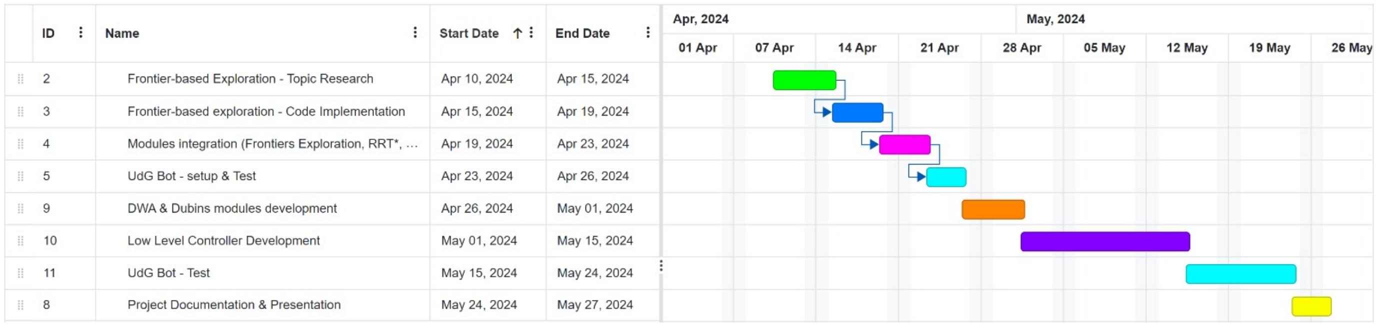
A. Expected Gantt Chart



ID	Name	Start Date	End Date
2	Frontier-based Exploration - Topic Research	Apr 10, 2024	Apr 15, 2024
3	Frontier-based exploration - Code Implementation	Apr 15, 2024	Apr 25, 2024
4	Modules integration (Frontiers Exploration, RRT*, ...	Apr 25, 2024	May 01, 2024
5	UdG Bot - setup & Test	May 01, 2024	May 06, 2024
7	RRT* modules optimization	May 06, 2024	May 16, 2024
9	DWA & Dubins modules development	May 06, 2024	May 16, 2024
8	Project Documentation & Presentation	May 16, 2024	May 27, 2024

Fig. 10: Old Gantt chart

B. Actual Gantt Chart



ID	Name	Start Date	End Date
2	Frontier-based Exploration - Topic Research	Apr 10, 2024	Apr 15, 2024
3	Frontier-based exploration - Code Implementation	Apr 15, 2024	Apr 19, 2024
4	Modules integration (Frontiers Exploration, RRT*, ...	Apr 19, 2024	Apr 23, 2024
5	UdG Bot - setup & Test	Apr 23, 2024	Apr 26, 2024
9	DWA & Dubins modules development	Apr 26, 2024	May 01, 2024
10	Low Level Controller Development	May 01, 2024	May 15, 2024
11	UdG Bot - Test	May 15, 2024	May 24, 2024
8	Project Documentation & Presentation	May 24, 2024	May 27, 2024

Fig. 11: Recent Gantt chart