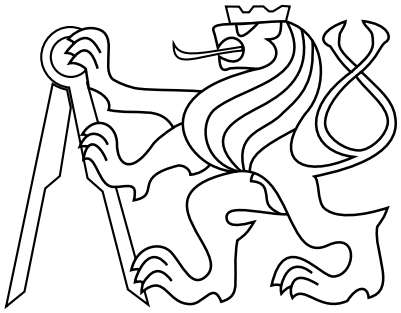
Czech Technical University in Prague

Faculty of Electrical Engineering
Department of Cybernetics
Multi-robot Systems



Development of a Safe Flocking Algorithm for UAVs Using 3D Lidar and Collaborative Multi-Robot Coordination

Bachelor's Thesis

Michal Kamler

Prague, May 2025

Study programme: Electrical Engineering and Information Technology
Branch of study: Cybernetics and Robotics

Supervisor: MSc. Manuel Boldrer, Ph.D.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, MSc. Manuel Boldrer, Ph.D., for his valuable guidance, helpful feedback, and for providing direction throughout the development of this thesis. His expertise was irreplaceable in shaping this work.

My appreciation also extends to the members of the Multi-robot Systems (MRS) group. I am particularly grateful for their support and practical assistance during the experimental phase, especially concerning safety during experiments and for helping me understand various technical aspects. Their willingness to share knowledge was very beneficial.

Finally, I would like to extend my thanks to my family for their support, patience, and encouragement throughout my studies and the writing of this thesis. Their belief in me has been a constant source of motivation.

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Kamler Michal** Personal ID number: **516070**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Development of a Safe Flocking Algorithm for UAVs Using 3D Lidar and Collaborative Multi-Robot Coordination

Bachelor’s thesis title in Czech:

Vývoj bezpečného algoritmu pro koordinaci UAV pomocí 3D lidaru a koordinace více robotů

Guidelines:

(1) Develop a safe flocking algorithm for UAVs in C++ within the MRS system framework . The algorithm must ensure collision-free and efficient movement of UAV.
(2) Compare your solution to some of the most promising algorithms in the literature within the MRS simulator, in particular [1],[2],[3].
(3) The novel contributions of the thesis will include
i) Extend existing multi-robot algorithm from 2d to 3d.
ii) Use 3d lidar to localize, sense, process, and react to environments such as a forest.
iii) Research on possible enhancements of the algorithm for example using neural networks or learning-based techniques.
(4) Conduct an experimental campaign to validate the theoretical findings. Include different real-world scenarios, such as navigating through forests or crowded spaces.

Bibliography / sources:

[1] Boldrer, M., Serra-Gomez, A., Lyons, L., Alonso-Mora, J., & Ferranti, L. (2024). Rule-Based Lloyd Algorithm for Multi-Robot Motion Planning and Control with Safety and Convergence Guarantees. arXiv preprint arXiv:2310.19511v2 (2023).
[2] Mezey, D., Bastien, R., Zheng, Y., McKee, N., Stoll, D., Hamann, H., & Romanczuk, P. (2024). Purely vision-based collective movement of robots. arXiv preprint arXiv:2406.17106.
[3] Ahmad, A., Licea, D. B., Silano, G., Báča, T., & Saska, M. (2022). PACNav: a collective navigation approach for UAV swarms deprived of communication and external localization. Bioinspiration & Biomimetics, 17(6), 066019

Name and workplace of bachelor’s thesis supervisor:

Manuel Boldrer, Ph.D. Multi-robot Systems FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **30.01.2025** Deadline for bachelor thesis submission: _____

Assignment valid until: **20.09.2026**

_____
prof. Dr. Ing. Jan Kybic
Head of department’s signature

_____
prof. Mgr. Petr Páta, Ph.D.
Vice-dean´s signature on behalf of the Dean

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt

Student's signature

FAKULTA ELEKTROTECHNICKÁ
FACULTY OF ELECTRICAL ENGINEERING
Technická 2
166 27 Praha 6

DECLARATION

I, the undersigned

Student's surname, given name(s): Kamler Michal
Personal number: 516070
Programme name: Cybernetics and Robotics

declare that I have elaborated the bachelor's thesis entitled

Development of a Safe Flocking Algorithm for UAVs Using 3D Lidar and Collaborative Multi-Robot Coordination

independently, and have cited all information sources used in accordance with the Methodological Instruction on the Observance of Ethical Principles in the Preparation of University Theses and with the Framework Rules for the Use of Artificial Intelligence at CTU for Academic and Pedagogical Purposes in Bachelor's and Continuing Master's Programmes.

I declare that I used artificial intelligence tools during the preparation and writing of this thesis. I verified the generated content. I hereby confirm that I am aware of the fact that I am fully responsible for the contents of the thesis.

In Prague on 21.05.2025

Michal Kamler
.....
student's signature

Abstract

Unmanned Aerial Vehicles (UAVs) are increasingly transitioning from open-sky operations to complex, Global Navigation Satellite System (GNSS)-denied environments, presenting significant challenges for multi-agent coordination and single-agent autonomous navigation. This thesis addresses these issues by first extending the Rule-based Lloyd (RBL) algorithm from two to three dimensions for multi-Unmanned Aerial Vehicle (UAV) communication-free coordination. New rules were developed to manage vertical movement and enhance 3D coordination efficiency. Secondly, the thesis investigates the practical application of this 3D RBL for single UAV navigation in a challenging forest environment, utilizing onboard Light Detection and Ranging (LiDAR) sensing.

The methodology involved adapting the core RBL for 3D space and integrating it with real-time perception systems. For the single-agent navigation task, this included LiDAR-based point cloud processing, state estimation using Point-LIO, and voxel-based environmental mapping with Bonxai. Strategies to minimize limitations from anisotropic sensors, such as restricted LiDAR Field of View (FOV), were also developed. The proposed algorithm was evaluated through extensive simulations in the multi-agent scenario and validated with real-world experiments in a forest environment for the single-agent case.

Simulations confirmed the successful 3D extension of the RBL algorithm, demonstrating effective multi-agent coordination. Real-world experiments validated the capability of the 3D RBL-based system for autonomous point-to-point navigation in a forest. However, limitations were identified, particularly in the mapping system's handling of dynamic environmental elements, which occasionally led to navigation failures.

Keywords Unmanned Aerial Vehicles, Distributed Multi-Agent Coordination, Anisotropic Sensor

Abstrakt

Výzkum na poli autonomních bezpilotních prostředků (UAV) se stal významným oborem mobilní robotiky.

Klíčová slova TODOBezpilotní Prostředky, Automatické Řízení

Abbreviations

FOV Field of View

GNSS Global Navigation Satellite System

IMU Inertial Measurement Unit

LiDAR Light Detection and Ranging

MPC Model Predictive Control

MRS Multi-robot Systems Group

SLAM Simultaneous Localization And Mapping

UAV Unmanned Aerial Vehicle

UAVs Unmanned Aerial Vehicles

UGV Unmanned Ground Vehicle

RBL Rule-based Lloyd

ToF Time-of-Flight

PCL Point Cloud Library

EMI Electromagnetic Interference

RL Reinforcement Learning

NN Neural Network

Point-LIO Point Light Detection and Ranging Inertial Odometry

Contents

1 Introduction

The field of robotics has witnessed transformative advancements in recent decades, with Unmanned Aerial Vehicles (UAVs) emerging as particularly versatile platforms. Their applications are rapidly expanding beyond traditional open-sky operations (agricultural surveying [1], powerline inspection [2], photogrammetry [3]) into increasingly complex scenarios, underscoring a significant trend towards autonomy. This growing relevance is highlighted by initiatives ranging from commercial use cases, such as Amazon's development of Unmanned Aerial Vehicle (UAV)s for package delivery and return services [4], to pioneering scientific missions like the Ingenuity helicopter's collaboration with the Perseverance rover on Mars [5], which demonstrated the potential of aerial robotic support in extraterrestrial exploration.

However, the widespread and effective deployment of UAVs, particularly in coordinated multi-agent systems and challenging environments, presents substantial research challenges. One critical area is multi-robot coordination, which requires robust methods for agents to perceive and react to each other efficiently. While distributed swarms can achieve scalability even with explicit communication, for instance, through the implementation of ad-hoc networks, developing strategies that minimize dependence on such communication is important. This is because communication channels are often prone to unreliability, including delays, failures, and packet losses, which can decrease performance or even compromise safety Simultaneously, autonomous navigation in complex, unstructured environments, often characterized by the absence or unreliability of Global Navigation Satellite System (GNSS) signals, remains a significant barrier.

Addressing these navigational challenges often involves advanced perception systems. Light Detection and Ranging (LiDAR) technology offers distinct advantages in providing accurate and fast spatial data compared to alternatives like depth cameras or computationally expensive neural network-based depth estimation from monocular images. While the precise real-time state estimation of the UAV within such GNSS-denied environments is a complex problem in itself (often addressed by dedicated Simultaneous Localization And Mapping (SLAM) algorithms, which are utilized as existing tools in this work rather than being a primary research focus), perception and navigation through complex cluttered spaces is the main objective. This thesis explores several challenges by first investigating approaches for UAV coordination in simulated multi-agent scenarios with predefined configurations - such as formations involving multiple UAVs crossing in a circular or spherical pattern, first in convex environments (e.g. without obstacles) then in more complex cluttered scenarios. Subsequently, it focuses on actual implementation on a single UAV of the proposed algorithm in complex enviroments such as forests, using perceived information from onboard sensors.

1.1 Related Works

This research primarily addresses multi-agent coordination and autonomous UAV navigation in complex environments. This section briefly reviews key literature to contextualize the thesis.

Foundational Work

This thesis directly extends the Rule-based Lloyd (RBL) algorithm presented in [6]. While the original work showed promise for distributed multi-agent navigation, it was limited

to two-dimensional (2D) scenarios. This thesis addresses the need to adapt RBL for three-dimensional (3D) UAV operations, which involve vertical movement and spatial complexity.

Multi-Agent Coordination and Navigation Approaches

Enabling multiple robots to navigate and coordinate effectively is a significant challenge that has led to a variety of research directions. A fundamental distinction lies in the system's architecture: centralized approaches rely on a global controller for optimal, system-wide decisions, often suitable for structured settings like warehouses [7], but face scalability and communication bottlenecks. In contrast, distributed or distributed methods allow individual agents to make decisions based on local information. This approach generally improves scalability, but do not provide an optimal solution for each agent. Distributed methods can be further categorized into three main types:

- **Reactive Methods:**
  These methods [8], [9] make robots react immediately to what their sensors see in their current local area. They are generally fast and simple, but because they don't look far ahead, robots using them can sometimes get stuck in deadlocks (e.g., becoming completely immobilized) or trapped in livelocks (e.g., engaging in repetitive, non-progressive loops), preventing them from reaching their goals.
- **Predictive Planning Methods:**
  These techniques [10], [11] aim to make robots smarter by using information about nearby robots or the environment to plan better routes. This can lead to improved performance and help avoid getting stuck. However, these methods often need more computational power and sometimes they need to heavily rely on communication between robots.
- **Learning-Based Methods:**
  These newer approaches [12], [13] use techniques like Reinforcement Learning to teach robots how to navigate by learning from experience or data. They can be very good at handling complex situations. However, it's often hard to guarantee they will always be safe or reach their goal, and they might not work well in situations very different from what they were trained on.

RBL algorithm, which is a core focus of this thesis, is fundamentally a reactive approach that has been enhanced with specific rules designed to prevent common deadlock situations.

1.2 Problem Statement

The increasing deployment of UAVs in diverse and complex scenarios requires robust autonomous navigation and coordination capabilities. While foundational algorithms like the RBL algorithm, as presented in [6], have shown promise for distributed multi-agent navigation, their application has primarily been explored in 2D planar environments. This thesis identifies and addresses two key problem areas emerging from these limitations and the demands of real-world applications:

(i) **Limitations of 2D Algorithms for 3D Multi-Agent Coordination:**
The direct application of 2D coordination strategies to three-dimensional space can be incomplete. Adding the vertical dimension makes the navigation problem harder, as it introduces more agent interactions and demands new rules for efficient goal convergence. Existing 2D algorithm lacks the methods to effectively control vertical exploration and coordination.

- **Problem 1:**
  Given N UAVs operating in a shared 3D environment, there is a need to develop a distributed coordination algorithm that extends planar navigation principles to efficiently steer each agent from its initial position to a goal region in three dimensions, while ensuring safety and operating without communication. This involves the challenge of designing rules that effectively manage the vertical dimension without compromising the scalability and deadlock-avoidance properties of the foundational algorithm.

(ii) **Transitioning Navigation Algorithms to Complex, GNSS-Denied Real-World Environments:**
Another challenge lies in applying navigation algorithms to UAVs operating in complex, unstructured, and GNSS-denied environments, such as dense forests. For successful operation in these settings, UAVs must effectively perceive their surroundings, accurately estimate their own state, and adequately represent the environment.

- **Problem 2:**
  Given a single UAV equipped with sensors like LiDAR, operating in a cluttered, GNSS-denied environment (e.g., a forest), the problem is to enable reliable autonomous point-to-point navigation. This requires not only a suitable 3D navigation algorithm (such as an extension of RBL) but also its effective integration with real-time sensor data processing for obstacle detection and accurate state estimation, all while addressing additional challenge of navigating with anisotropic onboard sensors and their inherent limitations (e.g., restricted fields of view).

1.3 Contributions

This thesis presents four key contributions to the field of autonomous UAV navigation and multi-agent coordination.

- Extension of the RBL algorithm to three dimensions, development of new rules to enhance multi-UAV coordination and efficiency in 3D space.
- Development and practical integration of navigation strategy that effectively address the challenges posed by anisotropic onboard sensors (e.g., limited Field of View (FOV)). This work advances beyond common isotropic sensing assumptions, often assumed in foundational algorithms like RBL, to enable navigation in realistic environments.
- Experimental validation of the developed algorithm through simulations and real-world trials in challenging forested environments.

1.4 Mathematical Notation

$\mathbf{x}$, $\boldsymbol{\alpha}$	vector or tuple
$\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$	elements of the standard basis
$\mathbf{X}, \boldsymbol{\Omega}$	matrix
$\mathbf{y} = \mathbf{A}\mathbf{x}$	matrix-vector product of matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and vector $\mathbf{x} \in \mathbb{R}^n$
$\mathbf{C} = \mathbf{A}\mathbf{B}$	matrix product of matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and matrix $\mathbf{B} \in \mathbb{R}^{n \times p}$
$s = \mathbf{a} \cdot \mathbf{b}$	dot product (scalar product) of $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$
$\dot{x}$	1^{st} time derivative of x
SO(3)	3D special orthogonal group of rotations
$\mathbf{R}_x, \mathbf{R}_y, \mathbf{R}_z$	rotation matrices about the x, y, and z axes, $\mathbf{R}_x, \mathbf{R}_y, \mathbf{R}_z \in \mathrm{SO}(3)$

Table 1.1: Mathematical notation, nomenclature and notable symbols.

2 Extension of the Rule-Based Lloyd Algorithm to 3D

2.1 Chapter Overview

The principles of the RBL algorithm, presented in [6], form the basis of the work in this chapter. These principles are clarified and modified to address the challenges of 3D extension. The chapter details the fundamental principles of RBL in 2D and 3D spaces, describes the specific rules applied to enhance convergence, and concludes with a series of simulation scenarios designed to demonstrate the performance of the proposed solution. The chapter concludes with the presentation and evaluation of simulation experiments designed to assess the efficiency of the proposed 3D extension and the impact of the introduced vertical exploration rules.

2.2 Basic principles of Rule-Based Lloyd Algorithm

Overview

The algorithm is a communication-less approach designed to navigate agents from point A to point B. The algorithm's ability to determine each agent's position relative to its goal depends on the availability of positioning data. This can be either global positioning data, such as from GPS, or relative positioning data obtained through techniques like SLAM in GNSS-denied environments, combined with sensor inputs that provide information about the agent's surrounding environment. These sensors can include LiDAR, depth cameras, or standard cameras combined with estimation techniques, allowing the agent to detect and avoid obstacles and other agents. The algorithm enables autonomous navigation without requiring direct communication between agents, making it suitable for scalable and distributed applications.

Applications and Limitations in 2D

Modern robotics relies on the capability to navigate from point A to point B. Navigation plays a crucial role in various robotic applications, such as Unmanned Ground Vehicle (UGV)s, which are commonly used in manufacturing and logistics. UGVs typically follow predefined 2D trajectories guided by visual [14], magnetic [15], or LiDAR-based navigation [16]. Additionally, 2D navigation is widely used in robotic vacuum cleaners, enabling them to systematically cover an area while avoiding obstacles.

An obvious limitation for algorithms in 2D is efficiency. As the number of agents in a system increases, the complexity of managing their movements and coordination also grows significantly. Obstacle avoidance in 2D can also be less efficient compared to 3D environments, as agents have fewer options for evading obstacles. In 3D, agents can change their altitude in addition to their horizontal trajectory, giving them more freedom to maneuver around obstacles.

Key Principles in 2D

It should be noted that most of the information presented here is adapted from the work detailed in [6]. RBL ensures convergence to the goal and provides sufficient conditions

for achieving it. The problem involves individual control of N agents from their initial position $\mathbf{p}_i(0)$ toward a goal region, represented as a circle. This goal region is denoted as $G(\mathbf{g}_i, r_g)$, where $\mathbf{g}_i$ is the center and r_g is the radius of the goal region. The agent is progressing towards its designated destination, $\mathbf{d}_i$. Each agent knows its current position $\mathbf{p}_i$, encumbrance δ_i, which determines the safe space around the agent. Additionally, each agent also knows the positions and encumbrances of its neighboring agents $\mathcal{N}_\mathbf{i}$, an agent $j \in \mathcal{N}_\mathbf{i}$ if $||\mathbf{p}_i - \mathbf{p}_j|| \leq \frac{r_{s,i}}{\eta}$, where $r_{s,i}$ is sensing radius of the i-th agent and η is a scaling parameter of cells, as demonstrated in Equation (2.3). For simplicity $r_{s,i}$ is considered to be the same for all agents, therefore $r_{s,i} = r_s$.

The core objective of the algorithm is to minimize the coverage cost function, which accounts for the distribution of agents and obstacles over the environment. This function is expressed as:

$$J_{\text{cov}}(\mathbf{p}) = \sum_{i=1}^{N} \int_{\mathcal{V}_i} \|\mathbf{q} - \mathbf{p}_i\|^2 \varphi_i(\mathbf{q}) d\mathbf{q}, \tag{2.1}$$

where $\mathbf{p}_i$ is the position of agent i, $\mathcal{V}_i$ is the Voronoi cell of the i-th robot, $\|\mathbf{q} - \mathbf{p_i}\|^2$ is squared Euclidian distance between a point $\mathbf{q} \in \mathcal{Q}$ in the mission space and agent's position $\mathbf{p}_i$, and $\varphi_i(\mathbf{q})$ is the weighting function.

The Voronoi cell $\mathcal{V}_i$ is defined as:

$$\mathcal{V}_i = \{q \in \mathcal{Q} | \|\mathbf{q} - \mathbf{p}_i\| \leq \|q - \mathbf{p}_j\|, \forall j \neq i\}. \tag{2.2}$$

For visual representation see Fig. 2.2.

However, this standard definition of Voronoi cells does not take into account the physical space occupied by the agents and their encumbrances. To address this, a Modified Voronoi cell is introduced, which takes into account the encumbrances of the agents. This modified version adjusts the boundaries of each Voronoi cell to account for the encumbrances of neighboring agents. Also to enhance the algorithm performance, the Voronoi cells are scaled using a scaling parameter $\eta \in [0, 1]$. The modified Voronoi cell definition is as follows:

$$\tilde{V}_i = \begin{cases} \{\mathbf{q} \in Q \mid \mathbf{n}_{\mathbf{p}_j} \cdot (\mathbf{q} - \mathbf{a}_{\mathbf{p}_j}) \leq 0\} & \text{if } \|\mathbf{p}_i - \tilde{\mathbf{p}}_i\| \leq \|\mathbf{p}_i - \tilde{\mathbf{p}}_j\| \\ \{\mathbf{q} \in Q \mid \mathbf{n}_{\mathbf{p}_j} \cdot (\mathbf{q} - \mathbf{a}_{\mathbf{p}_j}) \geq 0\} & \text{otherwise,} \end{cases} \tag{2.3}$$

$\forall j \in \mathcal{N}_i$, where $\mathcal{N}_i$ is set of all neighbors, $\mathbf{n}_{\mathbf{p}_j} = \tilde{\mathbf{p}}_j - \tilde{\mathbf{p}}_i$ is norm that defines a line, $\mathbf{a}_{\mathbf{p}_j} = \eta\ \mathbf{n}_{\mathbf{p}_j} + \tilde{\mathbf{p}}_i$ is a point defining line, $\tilde{\mathbf{p}}_i = \Delta_{i,j} \frac{\mathbf{p}_j - \mathbf{p}_i}{\|\mathbf{p}_j - \mathbf{p}_i\|} + \mathbf{p}_i$ is a point on the agent closest to the neighbor j, $\tilde{\mathbf{p}}_j = \Delta_{i,j} \frac{\mathbf{p}_i - \mathbf{p}_j}{\|\mathbf{p}_i - \mathbf{p}_j\|} + \mathbf{p}_j$ is a point on neighbor closest to the agent i, and $\Delta_{i,j} = \delta_i + \delta_j$. The parameter η controls the degree of overlap or empty space between cells. Specifically, for the classical Voronoi cell definition ($\eta = 0.5$ for uniform agent encumbrances), each agent needs to detect its neighbors within a distance of $2r_s$. In general, each agent needs to know its neighbors within a distance of $\frac{r_s}{\eta}$.

Together with the cell $\mathcal{S}_i$ defined as:

$$\mathcal{S}_i = \{\mathbf{q} \in \mathcal{Q} | \|\mathbf{q} - \mathbf{p}_i\| \leq r_{s,i}\}, \tag{2.4}$$

the cell $\mathcal{A}_i$ is obtained as:

$$\mathcal{A}_i = \tilde{V}_i \cap \mathcal{S}_i. \tag{2.5}$$

The main parameters and variables are depicted in Fig. 2.1 for improved clarity.

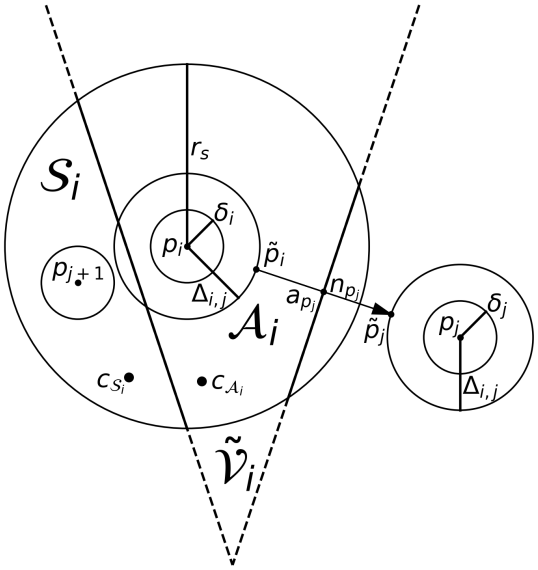


Figure 2.1: Main parameters associated with agent i.

The convergence to goal region $G(\mathbf{g}_i, r_g)$ depends on the choice of weighting function that assigns weights to points $\mathbf{q}$ in the mission space $\mathcal{Q}$. The weighting function $\varphi_i(\mathbf{q})$ is defined as follows:

$$\varphi_i(\mathbf{q}) = \exp\left(-\frac{\|\mathbf{q} - \mathbf{d}_i\|}{\beta_i}\right), \tag{2.6}$$

where β_i is the weighting factor for points $\mathbf{q}$, and $\mathbf{d}_i$ represents the current destination of the agent. The destination is computed as follows:

$$\mathbf{d}_i = \mathbf{p}_i + R(\theta)(\mathbf{g}_i - \mathbf{p}_i), \tag{2.7}$$

where R is the azimuthal rotation matrix. The rules for changing the weighting function β_i and the rotation θ in the azimuthal rotation matrix are defined as follows:

- **Weighting rule**

$$\dot{\beta}_i(A_i) = \begin{cases} -k & \text{if } \beta_i > \beta_{\min} \wedge \|\mathbf{c}_{A_i} - \mathbf{p}_i\| < d_1 \wedge \|\mathbf{c}_{A_i} - \mathbf{c}_{\mathcal{S}_i}\| > d_2 \\ 0 & \text{if } \beta_i \leq \beta_{\min} \wedge \|\mathbf{c}_{A_i} - \mathbf{p}_i\| < d_1 \wedge \|\mathbf{c}_{A_i} - \mathbf{c}_{\mathcal{S}_i}\| > d_2 \\ -(\beta_i - \beta_i^D) & \text{otherwise,} \end{cases} \tag{2.8}$$

  where the first case decreases β_i over time, the second case ensures that β_i does not decrease below its minimum threshold $\beta_{\min}$ (saturation) and the third case provides a general update rule when the previous conditions are not met.
- **Azimuth update rule**

$$\dot{\theta}_i = \begin{cases} k & \text{if } \theta < \frac{\pi}{2} \wedge \|\mathbf{c}_{\mathcal{A}_i} - \mathbf{c}_{\mathcal{S}_i}\| > d_4 \wedge \|\mathbf{p}_i - \mathbf{c}_{\mathcal{A}_i}\| > d_3 \\ -k & \text{if } \theta > 0 \wedge \neg(\|\mathbf{c}_{\mathcal{A}_i} - \mathbf{c}_{\mathcal{S}_i}\| > d_4 \wedge \|\mathbf{p}_i - \mathbf{c}_{\mathcal{A}_i}\| > d_3) \\ 0 & \text{otherwise,} \end{cases} \tag{2.9}$$

  where the first case increases θ_i over time, the second case ensures that θ_i converges back when the distance constraints are not satisfied, and the third case keeps θ_i unchanged.

- **Azimuth reset rule**

$$\theta = 0 \quad \text{if } \theta = \frac{\pi}{2} \wedge \|\mathbf{p}_i - \overline{\mathbf{c}}_{\mathcal{A}_i}\| > \|\mathbf{p_i} - \mathbf{c}_{\mathcal{A}_i}\|, \tag{2.10}$$

  where $\overline{\mathbf{c}}_{\mathcal{A}_i}$ represents the centroid computed from the cell $\mathcal{A}_i$, which is weighted using the unrotated destination, meaning $\mathbf{d}_i = \mathbf{g}_i$.

The weighted centroid is computed as follows:

$$\mathbf{c}_{\mathcal{A}_i} = \frac{\int_{\mathcal{A}_i} \mathbf{q}\varphi_i(\mathbf{q})\, d\mathbf{q}}{\int_{\mathcal{A}_i} \varphi_i(\mathbf{q})\, d\mathbf{q}}, \tag{2.11}$$

where $\mathbf{q}$ represents the a point in mission the space, and $\varphi_i(\mathbf{q})$ is a weighting function. The centroids for other relevant cells are computed using a similar approach, but over different sets. The primary weighted centroid, $\mathbf{c}_{\mathcal{A}_i}$, is particularly important as it serves as immediate navigational target for agent i. The RBL strategy relies on each agent i to continuously moving towards this dynamically updated centroid $\mathbf{c}_{\mathcal{A}_i}$ within cell $\mathcal{A}_i$.

By applying the previously defined rules and computations, the RBL algorithm successfully guides agent movement toward the goal. This guidance is inherently safe because each agent i is always restricted within its convex cell $\mathcal{A}_i$. These cells are constructed to be free of other agents, thereby ensuring collision-free navigation. However, these rules focus on rotating the centroid $\mathbf{c}_{\mathcal{A}_i}$ in the azimuthal plane. To enhance performance in a fully three-dimensional space, additional rules are required to account for elevation adjustments.

2.3 Extension of the RBL algorithm to 3D

Motivation for 3D Extension

Extending agent navigation to three dimensions is crucial for many practical applications where movement is not restricted to a planar surface. The ability to utilize this vertical dimension enhances agent capabilities by enabling more energy-efficient path optimization—such as ascending or descending to avoid obstructions or access more advantageous conditions.

Differences between 2D and 3D

The primary distinction in the 3D extension is that each goal region is now represented as a sphere rather than a circle. Similarly, the sensing cell $\mathcal{S}_i$ is also modeled as a sphere instead of a circle, allowing for a more accurate representation of the UAV's perception in three-dimensional space. This change introduces a key modification when defining $\tilde{V}_i$. While in the 2D case, a line was sufficient to slice the sensing region, whereas in 3D, a plane must be computed to properly segment the spherical sensing cell.

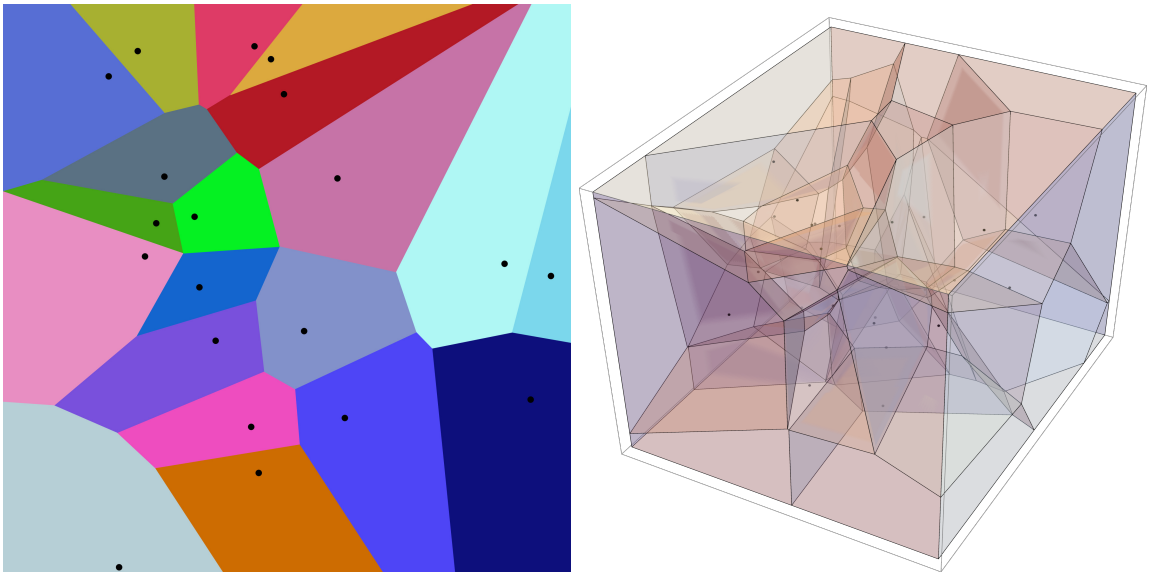


Figure 2.2: Left: An example of 20 Voronoi cells in 2D [17]. Right: 25 Voronoi cells in 3D [18].

Additional Constraints and Modifications

Extending the approach to three dimensions necessitates several adjustments. The weighting rule (2.8) remains unchanged. However, in both the azimuth update rule (2.9) and the azimuth reset rule (2.10), only the displacement of centroids projected onto the xy-plane is taken into account. The modified formulations for the 3D case are as follows:

- **Azimuth update rule 3D**

$$\dot{\theta}_i = \begin{cases} k & \text{if } \theta < \frac{\pi}{2} \wedge \|\mathbf{c}_{\mathcal{A}_i} - \mathbf{c}_{\mathcal{S}_i}\|_{xy} > d_4 \wedge \|\mathbf{p}_i - \mathbf{c}_{\mathcal{A}_i}\|_{xy} > d_3 \\ -k & \text{if } \theta > 0 \wedge \neg(\|\mathbf{c}_{\mathcal{A}_i} - \mathbf{c}_{\mathcal{S}_i}\|_{xy} > d_4 \wedge \|\mathbf{p}_i - \mathbf{c}_{\mathcal{A}_i}\|_{xy} > d_3) \\ 0 & \text{otherwise.} \end{cases} \tag{2.12}$$

- **Azimuth reset rule 3D**

$$\theta = 0 \quad \text{if } \theta = \frac{\pi}{2} \wedge \|\mathbf{p}_i - \overline{\mathbf{c}}_{\mathcal{A}_i}\|_{xy} > \|\mathbf{p_i} - \mathbf{c}_{\mathcal{A}_i}\|_{xy}. \tag{2.13}$$

The notation $\|\cdot\|_{xy}$ denotes the Euclidean norm computed in the xy-plane only, defined as:

$$\|\mathbf{p}_i - \mathbf{p}_j\|_{xy} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}. \tag{2.14}$$

Additionally, to enforce operational altitude limits for agent movement and interaction (for instance, to prevent excessive vertical exploration and focus movement within a defined vertical range) Z_{clipping} is applied to each sensing cell $\mathcal{S}_i$, constraining it within the vertical limits defined by $\min_z$ and $\max_z$:

$$\mathcal{S}_i = \left\{ \mathbf{q} \in \mathcal{Q} \mid \|\mathbf{q} - \mathbf{p}_i\| \leq r_{s,i}, \quad \min_z \leq q_z \leq \max_z \right\}, \tag{2.15}$$

where $\min_z$ and $\max_z$ define the vertical bounds within which the sensing region $\mathcal{S}_i$ is restricted. This ensures that the agent cannot exceed these limits, as it follows the computed centroid $\mathbf{c}_{\mathcal{A}_i}$. By constraining the sensing radius, the agent remains confined within the specified region, preventing it from moving outside the vertical interval $\min_z$ to $\max_z$.

The destination rotation rule Z_{rule} is introduced to enhance agent avoidance by rotating the computed destination $\mathbf{d}_i$ by an angle ϕ. For vertical rotation angle ϕ, the following condition is introduced

$$\Omega = (\|\mathbf{c}_{\mathcal{A}_i} - \mathbf{c}_{\mathcal{S}_i}\|_z < d_6) \wedge (\|\mathbf{p}_i - \mathbf{c}_{\mathcal{A}_i}\|_z) \vee (|\|\mathbf{p}_i - \mathbf{c}_{\mathcal{S}_i}\|_{xy} - \|\mathbf{p}_i - \mathbf{c}_{\mathcal{A}_i}\|_{xy}| > d_7). \tag{2.16}$$

This condition has two main parts. The first part evaluates the vertical displacement of the centroid of the partitioned cell $\mathbf{c}_{\mathcal{A}_i}$ from the centroid of the sensing cell $\mathbf{c}_{\mathcal{S}_i}$ and the displacement of the agents position $\mathbf{p}_i$ from $\mathbf{c}_{\mathcal{A}_i}$. The second part considers the difference in the horizontal plane between the agent's position and the centroids $\mathbf{c}_{\mathcal{A}_i}$ and $\mathbf{c}_{\mathcal{S}_i}$.

To improve agent distribution in the vertical dimension, a directional influence on vertical exploration is introduced. Given that each agent's position $\mathbf{p}_i$ and final goal $\mathbf{g}_i$ are known, the directional influence can be included into the update rule for ϕ_i.

First, the global heading angle, θ_{goal}, towards the goal is calculated:

$$\theta_{\text{goal}} = \text{atan2}(g_{i,y} - p_{i,y}, g_{i,x} - p_{i,x}), \tag{2.17}$$

where $g_{i,x}$, $g_{i,y}$ represent the x and y coordinates of the goal, and $p_{i,x}$, $p_{i,y}$ represent the x and y coordinates of the agent. The resulting angle is in the range $[-\pi, \pi]$. Next, the heading angle is linearly mapped to a direction influence $D_{\text{influence}}$ value in the range [-1, 1]:

$$D_{\text{influence}} = \frac{\theta_{goal}}{\pi}. \tag{2.18}$$

This mapping ensures that agents traveling directly northward (from South to North) have a directional influence close to one, which will promote upward vertical exploration. Similarly, agents traveling southward have a direction influence close to -1, promoting downward exploration. Agents moving primarily eastward or westward have a directional influence close to 0, resulting in minimal vertical exploration. This strategy encourages a more uniform distribution of agents throughout the 3D space.

To determine the adjustment to the agent's vertical orientation, the directional influence $D_{\text{influence}}$ is combined with the relative vertical displacement of the agent and the centroid $\mathbf{c}_{\mathcal{S}_i}$ A weighted average is used to combine them:

$$C_{\text{influence}} = \frac{w_1 \cdot (\mathbf{c}_{\mathcal{S}_i,z} - \mathbf{p}_{i,z}) + w_2 \cdot D_{\text{influence}}}{w_1 + w_2}, \tag{2.19}$$

where w_1 and w_2 are weighting factors. The resulting combined influence $C_{\text{influence}}$ is then used to update the agent's rotation of its destination $\mathbf{d}_i$ by ϕ_i.

Based on condition (2.16) and combined influence $C_{\text{influence}}$ (2.19), an update rule for ϕ_i can be constructed as follows:

$$\dot{\phi}_i = \begin{cases} k & \text{if } \phi_i < \frac{\pi}{4} \wedge C_{\text{influence}} > 0 \wedge \Omega \\ 0 & \text{if } \phi_i > \frac{\pi}{4} \wedge C_{\text{influence}} > 0 \wedge \Omega \\ -k & \text{if } \phi_i > -\frac{\pi}{4} \wedge C_{\text{influence}} \leq 0 \wedge \Omega \\ 0 & \text{if } \phi_i < -\frac{\pi}{4} \wedge C_{\text{influence}} \leq 0 \wedge \Omega \\ -k & \text{if } \phi_i > 0 \wedge \neg\Omega \\ k & \text{if } \phi_i < 0 \wedge \neg\Omega \\ 0 & \text{otherwise,} \end{cases} \tag{2.20}$$

where the first four cases control the rotation of the agent's destination $\mathbf{d}_i$ and the final three cases handle the smooth convergence of ϕ_i back to 0, when the condition Ω is not met.

After the application of the rules for θ_i and ϕ_i, the agent's destination is updated as:

$$\mathbf{d}_i = \begin{pmatrix} p_{i,x} + \|\mathbf{p}_i - \mathbf{g}_i\| \cdot \sin(\phi_{\mathbf{g}_i} + \phi_i) \cdot \cos(\theta_{\mathbf{g}_i} - \theta_i) \\ p_{i,y} + \|\mathbf{p}_i - \mathbf{g}_i\| \cdot \sin(\phi_{\mathbf{g}_i} + \phi_i) \cdot \sin(\theta_{\mathbf{g}_i} - \theta_i) \\ p_{i,z} + \|\mathbf{p}_i - \mathbf{g}_i\| \cdot \cos(\phi_{\mathbf{g}_i} + \phi_i) \end{pmatrix}, \tag{2.21}$$

where $\phi_{\mathbf{g}_i} = \arccos(\frac{\mathbf{g}_{i,z} - \mathbf{p}_{i,z}}{\|\mathbf{p}_i - \mathbf{g}_i\|})$ is the polar angle from the z-axis to the goal, $\theta_{\mathbf{g}_i} = \text{atan2}(g_{i,y} - p_{i,y}, g_{i,x} - p_{i,x})$ is the azimuthal angle in the xy-plane to the goal.

Algorithm Walkthrough

At each iteration, the proposed algorithm performs a sequence of actions to control agent movement. These actions can be summarized as follows:

(i) **Cell Generation:**
For each agent, two cells are generated: a partitioned cell, denoted as $\mathcal{A}_i$ (2.5), and a sensing cell, denoted as $\mathcal{S}_i$ (2.4).

(ii) **Centroid Computation:**
- The centroid of cell $\mathcal{A}$, $\mathbf{c}_\mathcal{A}$ (2.11) is computed using a weighted function (2.6) that considers both the agent's goal position, $\mathbf{g}_i$, and its current destination, $\mathbf{d}_i$ (2.7).
- The centroid of cell $\mathcal{S}$, $\mathbf{c}_\mathcal{S}$ (2.11) is computed using a weighted function (2.6) that considers only the agent's current destination, $\mathbf{d}_i$ (2.7).

(iii) **Rule Application and Destination Update:**
A set of rules is applied to:
- Adjust weighting function used in the centroid computations (2.8).
- Rules to update θ_i (2.12) and ϕ_i (2.20) are applied.
- Update the agent's destination $\mathbf{d}_i$ (2.21).

(iv) **Agent movement**
Finally, the agent is guided towards a centroid location $c_\mathcal{A}$. This guidance is achieved by feeding the centroid's position as a reference setpoint to a Model Predictive Control (MPC). The MPC, a well-established control methodology already implemented within the Multi-robot Systems Group (MRS) system, then accounts for the robot's dynamics to generate the required low-level control inputs. Since this MPC is a standard component, its specific design and implementation are not a primary focus of this thesis.

This process is repeated until all agents reach their goals.

2.4 Simulation and Results Analysis

Computational Implementation Details

The continuous algorithm presented in the previous section can be easily discretized for implementation in a computational environment. This discretization is primarily necessary because the integral required to compute the centroids position, $\mathbf{c}_{\mathcal{A}_i}$ (2.11), generally cannot be solved analytically. Therefore, instead of continuous sets, the cells $\mathcal{A}_i$ and $\mathcal{S}_i$ are approximated using a set of discrete points. Discrete points are partitioned out when definition (2.3) is not met.

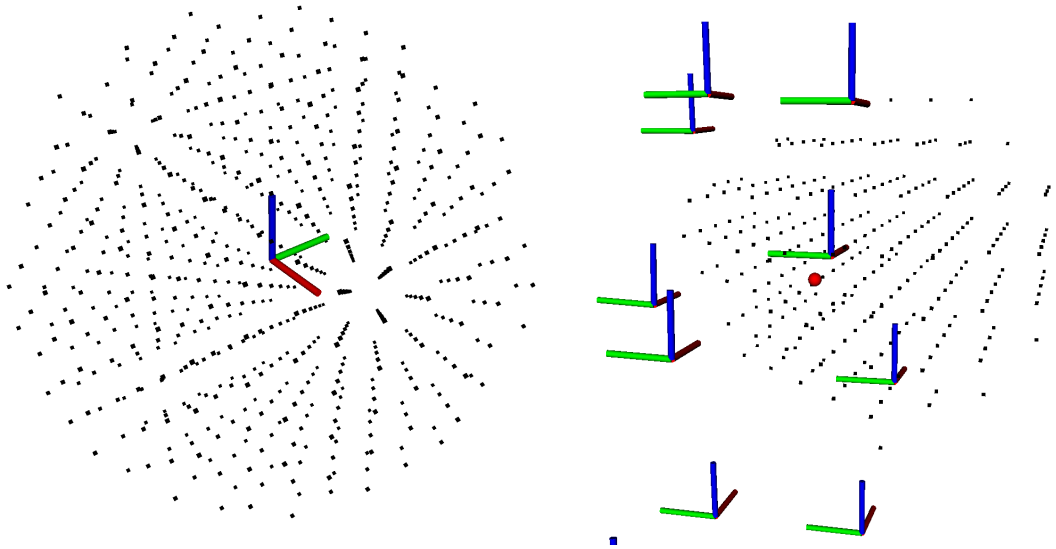


Figure 2.3: Discrete approximation of Voronoi cells, with centroid $\mathbf{c}_{\mathcal{A}}$ shown as a red dot.(a) Example where the partitioned cell $\mathcal{A}$ and the sensing cell $\mathcal{S}$ are the same. (b) Example where the partitioned cell $\mathcal{A}$ is partitioned by other UAVs.

Simulation Environment

In this simulation, the term 'agent' refers to the UAV, which was simulated using the framework provided by MRS [19]. Each UAV obtains its global ground truth position from the ROS simulator. The positions of neighboring UAVs were estimated using simulated blinking ultraviolet markers, following the method described in [20] and [21]. The following constraints are relevant for each UAV:

Parameter	**Value**
Maximal horizontal velocity $[\mathrm{m\,s^{-1}}]$	4.0
Horizontal acceleration $[\mathrm{m\,s^{-2}}]$	2.0
Maximal ascending velocity $[\mathrm{m\,s^{-1}}]$	2.0
Vertical ascending acceleration $[\mathrm{m\,s^{-2}}]$	1.0
Maximal descending velocity $[\mathrm{m\,s^{-1}}]$	2.0
Vertical descending acceleration $[\mathrm{m\,s^{-2}}]$	1.0

Table 2.1: Motion constraints of the UAV.

Simulation Scenarios

Predefined agent formations are utilized to establish a controlled experimental environment. These consisted of both circular and spherical arrangements where UAVs were evenly distributed, either along the circumference of circle or the surface of a sphere, each with a 5 meter radius. In all formation-based experiments, the goal position located on the opposite side

To evaluate the safety and performance of the UAVs during interactions, a series of experiments was conducted. Experiments were performed primarily using circular formations,

with UAV counts of N = 5, 10, and 15. Each of these circular formation experiments was repeated 10 times. Additionally, to extend the analysis to 3D and assess the algorithm's performance, a set of spherical formation experiments was conducted with N = 10, also repeated 10 times. After each UAV was flown from the ground to its starting position, the RBL algorithm was activated.

The following metrics were measured across the 10 repetitions of each experiment, along with their standard deviations: success rate SR [%], defined as the percentage of simulations where all UAVs successfully converged to their goals, average trajectory length $\overline{L}$ [m] for those that successfully reached their goal, average time to reach the goal $\bar{t}$ [s], average time for last UAV to reach the goal $\bar{t}_{\max}$ [s], and average velocity $\overline{v}$ [$\mathrm{m\,s^{-1}}$].

The specific values of the parameters used in the experiments are summarized in the following table 2.2:

Table 2.2: Parameters Used in Experiments

Parameter	r_s [m]	δ_i [m]	$d_1 = d_3 = d_5$ [m]	$d_2 = d_4 = d_6$ [m]	d_7 [m]
Value	3.5	0.5	0.5	1.0	0.2

Parameter	$\min_z$ [m]	$\max_z$ [m]	Update rate [Hz]	β_i^D [ ]	η [ ]	w_1 [ ]	w_2 [ ]
Value	1.0	10.0	10.0	1.5	0.9	0.7	0.3

Simulation Results

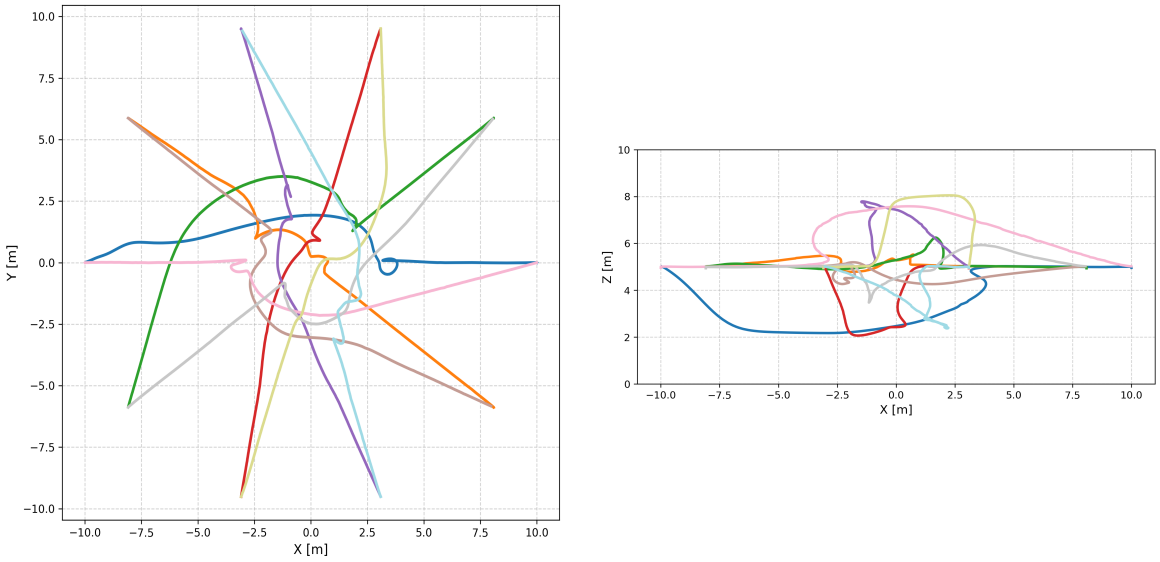
Full simulation results and trajectory visualizations are presented in Chapter A. The following tables highlight key results for the N=10 agent scenarios:

- $N = 10$ **Circular Formation:**

	SR [%]	$\overline{L}$ [m]	$\bar{t}$ [s]	$\bar{t}_{\max}$ [s]	$\overline{v}$ [m/s]
RBL 2D	100.00	22.95 ± 1.64	30.79 ± 2.28	34.73 ± 0.77	0.74 ± 0.07
RBL 3D	100.00	**22.22 ± 0.89**	30.05 ± 2.61	34.39 ± 4.19	0.73 ± 0.05
RBL $3D_{clipped}$	100.00	22.31 ± 0.69	30.22 ± 1.83	33.22 ± 0.93	0.73 ± 0.04
RBL $3D_{rule}$	100.00	22.38 ± 0.88	**28.80 ± 2.30**	**32.35 ± 1.25**	**0.77 ± 0.05**

- $N = 10$ **Spherical Formation:**

	SR [%]	$\overline{L}$ [m]	$\bar{t}$ [s]	$\bar{t}_{\max}$ [s]	$\overline{v}$ [m/s]
RBL 3D	100.00	14.32 ± 1.52	27.76 ± 3.06	32.48 ± 2.26	0.51 ± 0.05
RBL $3D_{rule}$	100.00	**14.06 ± 0.97**	**27.17 ± 2.66**	**31.86 ± 1.24**	**0.55 ± 0.06**



(a) Top-Down View (X-Y)

(b) Side View (X-Z)

Figure 2.4: Effect of the Z-axis rule on the circular formation.

The simulation results, with key performance metrics for N=5, 10, 15 UAVs detailed in Tables A.1, A.2, A.3 (circular formation) and Table A.4 (spherical formation), consistently demonstrate a 100% success rate (SR) across all tested RBL algorithm variants, underscoring fundamental reliability. Evaluations with circular formations, conducted for N=5, N=10, and N=15 UAVs, reveal a significant trend: while the purely 2D RBL version showed slightly better performance at N=5, the advantages of the 3D RBL variants, particularly RBL $3D_{rule}$, become increasingly improved as the number of agents increases through N=10 and N=15. This is clearly shown in the N=15 circular results presented in Table A.3, where RBL $3D_{rule}$ notably outperforms RBL 2D across all key metrics: achieving shorter average path lengths $\overline{L}$, shorter average $\bar{t}$ and maximum $\bar{t}_{max}$ completion times, and higher average speed $\overline{v}$.

The spherical formation experiments with N=10 UAVs Table A.4 were particularly important as they successfully demonstrated the 3D RBL system's capability to manage safe agent movement in all directions. As intended, this $3D_{rule}$ promote a distributed utilization of the vertical dimension, some UAVs strategically ascend, others descend, while some make little to none vertical adjustments.

2.5 Comparison with State-of-the-Art Method

As an alternative approach developed within the MRS framework, the 3D RBL algorithm presented here can be compared to other state-of-the-art methods like PACNav [22]. Both algorithms target communication-less UAV coordination, a feature desirable for reliability. Furthermore, implementations described for both approaches have utilized UVDAR to determine the positions of neighboring UAVs.

The approaches differ notably in their coordination strategy. PACNav utilizes a leader-follower dynamic: 'uninformed' UAVs identify and follow 'informed' (or otherwise reliable) leaders based on observed motion characteristics like path persistence and similarity, encouraging group cohesion. However, such behavior-based strategies often require careful tuning of numerous parameters to ensure reliable leader identification, stable following, and correct

group behavior without introducing undesirable oscillations. These parameters may also need recalibration for different scenarios, such as varying numbers of robots or environmental conditions. In contrast, the RBL algorithm assigns an individual goal to each UAV. Coordination, including inherent collision avoidance and thus operational safety, emerges as each agent calculates its path based on its local modified Voronoi cell, moving towards its computed centroid while implicitly accounting for neighbors. This design lowers the need for behavior-specific parameter tuning to ensure safety.

Another distinct approach in distributed, communication-less coordination is the purely vision-based model by Mezey et al. [23]. Unlike RBL, which relies on explicit position information, the Mezey et al. model derives control actions from a 1D visual projection field generated by detecting neighbors via onboard camera. Coordination emerges from low-level visual attraction-repulsion forces, resulting in patterns like flocking or swarming rather than directed motion towards individual goals. While effectively demonstrated in 2D with UGVs, extending this purely vision-based model to complex 3D environments presents significant challenges. Achieving consistent behavior typically depends on accurate tuning of its underlying attraction-repulsion parameters, which can be scenario-dependent and prone to producing oscillatory or unpredictable group movements rather than reliable goal achievement. Furthermore, the 2D visual field representation poses reliability issues for robust 3D perception and navigation. The use of toroidal simulation environments in their work, which can reduce group fragmentation and artificially encourage specific emergent patterns, further limits the direct applicability of some observed behaviors to unconstrained real-world 3D scenarios. For instance, the documented swarming can be highly dependent on specific parameters like low steering gains (beta value), which, when combined with the toroidal space, may produce an alignment that is more a direct result of these conditions than a robust, generalizable coordination strategy. The 3D RBL, by utilizing explicit position information, aims for more predictable goal-oriented navigation and guaranteed safety in 3D, reducing the dependence on fine-tuning reactive parameters and overcoming the limitations of environment-specific emergent behaviors seen in model like Mezey et al.

2.6 Summary and Key Insights

This chapter detailed the extension of RBL algorithm, originally tested for 2D coordination, to operate in a full three-dimensional space. Key modifications included adapting the Voronoi cell partitioning for 3D, utilizing projections for already existing azimuth update rules, and introducing a new elevation rotation angle. This new rule dynamically adjusts the agent's vertical destination based on relative centroid positions and directional influence towards the goal, aiming to improve distribution. Note that for the directional influence to work the UAVs need to have information about a frame of reference between each other at the beginning of the algorithm, such as Earth's magnetic field. Vertical movement has also been constrained within a defined interval ($\min_z$, $\max_z$).

The proposed 3D RBL extensions were evaluated through simulations within the MRS framework, comparing the performance UAVs crossing circle and sphere. The results indicate that the extension variants successfully enabled multi-agent coordination in three dimensions. Specifically, analysis of the simulation results reveals a trend where the performance of the 3D approach with applied rules compared to the basic 2D appears to increase with the scaling number of agents. For instance, based on the average completion times, the 3D variant was calculated to be approximately 1.99% faster than the 2D variant for N=5 agents. This performance scales for larger swarms. For N=15 the 3D approach was calculated to be 4.22%

faster than the 2D. This suggests that the extension becomes increasingly beneficial in more crowded scenarios.

While the magnitude of this observed improvement was perhaps less than initially anticipated, it is important to consider the conservative motion constraints for UAVs on the Z-axis (velocity/acceleration), reflecting realistic UAV dynamics but limiting vertical exploration. Furthermore, the parameters used for the testing simulation were also conservative, suggesting that finer tuning - particularly of vertical exploration weights and weighting function aggressiveness - could potentially yield better performance. Nonetheless, the trend indicated by these results supports the conclusion that the 3D extension effectiveness in crowded scenarios rises. Furthermore, simulations in the spherical formation scenario confirmed the algorithm's capability to manage 3D navigation and coordination.

In conclusion, the simulations presented in this chapter demonstrate the feasibility of extending the RBL algorithm to 3D. The introduced elevation control rule appears particularly beneficial, enhancing the efficiency and convergence speed of the agents towards their goals in simulated 3D environments. This provides a promising foundation for applying the 3D RBL algorithm to real-world UAV navigation tasks, explored further in the subsequent chapter.

2.7 Future Work

Several ideas for future research and development emerge from the work presented in this chapter:

- **Real-World Multi-Agent Testing:**
  While the subsequent chapter explores single-agent navigation in a real forest, the multi-agent coordination aspects of the 3D RBL algorithm lack real-world validation. Initial plan to conduct such tests was set back by practical challenges, including the lack of full $360° \cdot 180°$ sensor coverage, which resulted in blind spots, particularly above and below the UAV. Specifically, hardware limitations on the MRS UAV for swarming [24], including emitter and camera placement that created blind spots (especially directly above and below), restricting the UVDAR system [21] from providing the complete spherical coverage needed for 3D coordination experiments. Resolving these hardware and sensing limitations to facilitate robust multi-agent 3D experiments remains a key step for future research. Constraining UAVs to a defined vertical region using Z_{clipping} offers another potential approach to partially address this sensor coverage issue.
- **Parameter Tuning using Reinforcement Learning:**
  While safety does not depend on parameter tuning, the performance of goal convergence is directly influenced by it. Reinforcement Learning (RL) could be used to automatically tune these parameters for optimal performance in specific environments or scenarios such as crossing circles. Parameters suitable for RL-based tuning include the vertical exploration weights (w_1, w_2), the scale parameter (η), aggressiveness parameter (β_i^D for $\varphi_i(\mathbf{q})$), the sensing radius (r_s), and potentially the distance thresholds (d_1 through d_7) for centroid displacement.
- **Neural Network-Based Navigation Policy:**
  An alternative direction would be to replace the rule-based approach entirely with a learned policy using Neural Network (NN).
  - Input Representation: The primary NN input could be a fixed-size 3D tensor representing a discretized voxel grid centered on the UAV's current position (e.g., $n \cdot n \cdot n$, where n is an odd integer defining the sensing region). Each element in this tensor would hold a binary value: '1' indicating free space and '0' indicating occupied or

unknown space. This provides the network with a structured representation of the current surroundings. Additionally, the goal position (x_g, y_g, z_g) would serve as a separate input, directing the policy towards the desired destination.

- Output: The NN's output could directly be the next desired local waypoint (a point within the free-space part of the input array).
- Training: Training such an NN, likely using RL, would require a suitable simulation environment. This environment should feature dynamic obstacles (other agents) and static obstacles, along with randomized initial and goal positions to promote generalization. The reward structure should be designed to guide the learning effectively by penalizing collisions (with obstacles or other agents), penalizing the selection of invalid next waypoints (those outside the determined free-space), and rewarding progress towards the goal.

3 UAV Navigation Using the RBL Algorithm and LiDAR Sensing

3.1 Introduction

Motivation

UAVs are increasingly being deployed in challenging, unstructured environments like dense forests, moving beyond their traditional use in open areas. This shift is driven by a growing demand for autonomous solutions in sectors like environmental monitoring, forestry management (health assessment [25]), and search and rescue, where ground-based access is difficult. Furthermore, this work aligns with the broader trend of deploying UAVs for increasingly complex tasks, such as detailed infrastructure inspection (e.g., power lines, where autonomous navigation in cluttered, potentially high Electromagnetic Interference (EMI) environments is essential - Fly4Future [26]). Enabling UAVs to reliably navigate point-to-point within these challenging settings is a foundational step towards realizing these advanced applications safely and efficiently.

While the primary objective of this work is to enable successful point-to-point navigation within a forest using the RBL algorithm, a significant secondary benefit emerges from this process. By successfully navigating from point A to point B while simultaneously building a map of the traversed environment, the UAV generates valuable spatial data about the forest structure. This generated map can then serve a vital purpose: enabling enhanced planning for future missions within the same area. Once a map exists, subsequent UAV operations could potentially transition from purely reactive navigation strategies to more efficient methods with globally informed path planning. Leveraging prior knowledge of obstacle locations could significantly improve efficiency of future routine tasks.

Problem Statement

Operating UAVs effectively within complex, three-dimensional environments like forests presents significant navigational challenges that slow down autonomous deployment. The primary problems addressed in this work originate from:

(i) **GNSS-Denied Conditions:**
Within forests, the dense canopy and other obstructions frequently block or scatter GNSS signals, leading to unreliable or completely absent reception. This necessitates reliance on onboard sensors and algorithms (like SLAM based on LiDAR and Inertial Measurement Unit (IMU) data) for accurate localization and state estimation.

(ii) **Cluttered and Unpredictable Environments:**
Forests are inherently cluttered with numerous static obstacles (trees, trunks, branches) and potentially dynamic ones (animals, leafs, falling branches). The navigation system must be capable of perceiving these obstacles in real-time and planning safe paths around them without prior knowledge of their exact layout.

Therefore, the core problem is to develop and validate a robust autonomous navigation system that allows a UAV to reliably traverse between specified points in a cluttered, GNSS-denied forest environment using only onboard sensing and computation.

Objectives

The primary objectives within this chapter are:

- **Integrate the RBL with Onboard Sensing:**
  Adapt the core of the RBL to utilize real-time sensor data. This includes modifications to sensing cell $\mathcal{S}$ due to the introduction of an anisotropic sensor. The sensing cell needs to remain convex so the centroid computed from partitioned cell $\mathcal{A}$ remains within the set, to guarantee a safe UAV motion.
- **Process Point Cloud Data:**
  Implement necessary filtration to refine the raw point cloud data acquired from the LiDAR.
- **Integrate external packages on the UAV:**
  Integrate and configure external software packages for simultaneous environmental mapping and robust state estimation, suitable for operation within the GNSS-denied forest environment.
- **Experimental Validation:**
  Conduct experiments to evaluate the performance, robustness, and effectiveness of the complete navigation solution. This validation will be performed through real-world flight tests in an actual forest.

Chapter Overview

This chapter covers the LiDAR perception process, covering data acquisition, preprocessing, mapping, and the method used to integrate mapped voxel data into the RBL algorithm. Subsequently, the practical challenges of implementing the system on a UAV with sensor limitations are discussed, along with the used software solutions. The chapter proceeds to present the experimental validation, outlining the simulation setup and results, followed by the real-world forest experiment approaches, challenges, and performance analysis, illustrated with videos of successful flights [27], [28] and a mapping failure case [29]. Finally, a comparative analysis, discussion of limitations, and summary of key findings are presented.

3.2 LiDAR-Based Perception and Point Cloud Processing

Overview of LiDAR for UAV Navigation

LiDAR is a crucial sensing technology widely used in applications such as SLAM [30], autonomous vehicles [31], and precision agriculture [32]. It provides high-resolution spatial data about the surrounding environment, making it a valuable tool for perception and navigation in dynamic and complex environments. For UAV applications, LiDAR serves several essential functions:

- **3D Mapping** – Capturing a detailed representation of terrain, structures, and obstacles.
- **Obstacle Detection** – Identifying objects and estimating their position relative to the UAV for collision avoidance.
- **Autonomous Path Planning** – Assisting navigation algorithms by providing spatial information for decision-making.
- **Localization** – Helping the UAV maintain a safe altitude by detecting variations in ground elevation.

LiDAR offers several benefits that make it an attractive choice for UAV-based navigation:

- **High Accuracy** – Provides precise distance measurements, crucial for obstacle avoidance and localization.
- **Environment Reliability** – Functions effectively in various conditions, including low-light environments and featureless terrain where cameras may fail.
- **Fast Data Acquisition** – Captures thousands to millions of points per second, enabling real-time processing.
- **Rich Depth Information** – Unlike cameras that provide only 2D images, LiDAR generates accurate depth data, improving spatial awareness and 3D perception.

Despite its advantages, LiDAR also presents certain challenges and limitations:

- **Computational Complexity** – Processing large point clouds in real-time requires significant computational power, which may be a limitation for UAVs with low processing resources.
- **Sensor Noise** – External factors such as vibrations and the motion of UAV can introduce errors in point cloud data.
- **Limited Field of View (FoV)** – The placement of the LiDAR sensor on the UAV affects its coverage, requiring strategies to compensate for blind spots.
- **Environmental Interference** – Performance may degrade in challenging conditions such as fog, rain, or dense vegetation due to light deviation.
- **Power Consumption** – LiDAR sensors can consume a significant amount of power, which reduces the overall flight time of the UAV.
- **Interference with Other LiDARs** – LiDAR sensors can experience interference when multiple units are used nearby, potentially leading to faulty measurements.

Point Cloud Data Acquisition

LiDAR sensors determine object distances by emitting laser pulses and measuring the time it takes for the reflected light to return. This process, known as Time-of-Flight (ToF), involves scanning the environment with laser beams directed at varying horizontal and vertical angles. The reflected light, modulated in intensity, phase, or frequency, is captured by a receiver, which uses a lens to focus the signal onto a photodetector. This detector converts the light into an electrical signal via the photoelectric effect [33].

The system calculates distance based on the light's travel time, considering its near-light-speed propagation. Subsequent signal processing filters and analyzes the electrical signal, accounting for surface material and environmental variations. The output is a 3D point cloud representing the scanned environment, along with reflected laser energy intensities.

Preprocessing Techniques

To efficiently process LiDAR data and reduce computational complexity, the raw point cloud undergoes downsampling and filtering. The point cloud density is reduced using a voxel grid filter. Subsequently, points associated with the UAV's structure are removed based on its known encumbrance.

- **Voxel Grid Downsampling** – The raw LiDAR point cloud often contains a large number of points, which can be computationally expensive to process in real-time. To address this, we apply a voxel grid filter using the Point Cloud Library (PCL) [34]. This method partitions the 3D space into a grid of voxels with a given resolution (leafSize) and retains a single representative point per voxel. The filtering process reduces the number of points while preserving the overall structure of the environment.

- **Filtering Points Corresponding to the UAV Structure** – LiDAR sensors mounted on UAVs can capture unwanted points originating from the UAV itself, such as reflections from its frame or rotor rods. To prevent these points from interfering with navigation, additional filtering step was been applied. Points falling outside a specified distance range (closer than a minimum threshold or farther than a maximum threshold) are filtered out.

The resulting filtered point cloud contains only relevant environmental features while eliminating unnecessary points, improving efficiency for future processes. The point cloud is fed into the Point Light Detection and Ranging Inertial Odometry (Point-LIO) [30] state estimator and the bonxai mapping [35].

Point-LIO State Estimation

Point-LIO [30] is a robust, high-bandwidth LiDAR-inertial odometry system designed to accurately estimate rapid and aggressive robotic motions. Its core innovations address common limitations in traditional frame-based LIO approaches.

Key characteristics of Point-LIO include:

- **Point-by-Point Processing:**
  Unlike methods that accumulate LiDAR points into frames (scans) before processing, Point-LIO updates the system's state with each individual LiDAR point measurement as it arrives. This point-wise solution allows for extremely high-frequency odometry output that typically ranges from 4 to 8 kHz.
- **Motion Distortion Removal:**
  By processing points at their true sampling times, Point-LIO fundamentally eliminates the in-frame motion distortion that often affects frame-based systems, especially during fast movements.
- **IMU Modeling:**
  To better handle aggressive motions, Point-LIO features a kinematic model augmented with a stochastic process. This design treats IMU data as an output (a measurement of the true motion) rather than a direct input. This approach allows for accurate localization even when the robot undergoes aggressive motions that cause IMU measurements to saturate.

In essence, Point-LIO uses an Extended Kalman Filter to fuse each LiDAR point and IMU measurement at their respective sampling times. This filter is specifically an 'on-manifold' type, meaning it is designed to correctly handle the underlying geometry of the system's state, particularly for 3D orientation, ensuring more accurate and consistent estimation. This enables high-rate, accurate state estimation, even under severe vibrations or when angular velocities and accelerations exceed the IMU's measuring range.

Bonxai Mapping

As will be shown later in this chapter, prior terrain knowledge is beneficial for navigation. To achieve this, environmental mapping was performed by leveraging the Bonxai mapping approach [35], which the MRS group forked and adapted specifically for integration and compatibility with the MRS system. The resulting map is represented as a voxel grid, which is then used by the RBL algorithm for navigation planning.

Initially, an approach involving surface reconstruction from this voxel data was investigated. This involved estimating surface normals from the point cloud combined with the

Greedy Projection Triangulation (GP3) method from the PCL library to generate a polygonal mesh approximating the environment's surface. While this method could produce a relatively accurate surface representation, it turned out to be computationally expensive and too slow for real-time execution on the UAV's onboard computer.

Due to these limitations, the surface reconstruction approach was abandoned. Instead a simpler and more efficient method was adopted, which involves directly using the information about the environment stored in the voxel grid map. This direct voxel usage approach is detailed in the subsequent section. Consequently, further development of the mesh-based surface reconstruction method is not recommended for this application.

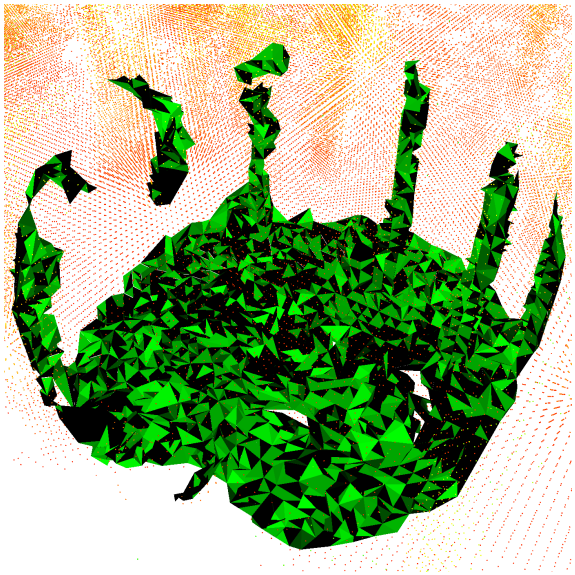


Figure 3.1: Surface approximation using Greedy Projection Triangulation.

Voxel-Based Modification of Cell $\mathcal{A}$

Given that the size of the voxels is known, this information can be used to refine the partitioning of cell $\mathcal{A}$ by treating voxels as obstacles. First, it is necessary to determine which voxels are relevant to the RBL algorithm. This can be achieved by considering the scaling parameter η. A voxel is considered relevant if it lies within a radius of $\frac{r_s}{\eta}$ of the agent. For each discrete point in the cell $\mathcal{S}$, the nearest voxel is identified using a k-d tree algorithm from the PCL library [36]. Because the found point is the voxel's center, the closest point within that voxel's boundaries is calculated to accurately partition the cell $\mathcal{A}$ from cell $\mathcal{S}$.

Given a point $\mathbf{p}_\mathcal{S}$, the voxel center $\mathbf{v_c}$ and the voxel edge length $\mathbf{e}$, $\mathbf{p}_{\text{closest}}$ is computed as:

$$\mathbf{p}_{\text{closest}} = \begin{pmatrix} \text{f}(p_{\mathcal{S},x}, v_{c,x} - \frac{\mathbf{e}}{2}, v_{c,x} + \frac{\mathbf{e}}{2}) \\ \text{f}(p_{\mathcal{S},y}, v_{c,y} - \frac{\mathbf{e}}{2}, v_{c,y} + \frac{\mathbf{e}}{2}) \\ \text{f}(p_{\mathcal{S},z}, v_{c,z} - \frac{\mathbf{e}}{2}, v_{c,z} + \frac{\mathbf{e}}{2}) \end{pmatrix}, \tag{3.1}$$

where the function f(x, a, b) constrains the value x to the range [a, b]. This constrains $\mathbf{p}_\mathcal{S}$ to the voxel boundaries.

Considering the point $\mathbf{p}_{\text{closest}}$ as an obstacle, the sensing cell $\mathcal{S}$ is partitioned to form cell $\mathcal{A}$ using the same procedure as (2.3) (considering the closest point as another agent) encumbrance of the obstacle is not needed, because the closest point has been already found.

The sensing cell $\mathcal{S}$ is partitioned to form the operational cell $\mathcal{A}$ by treating $\mathbf{p}_{\text{closest}}$ (the nearest point on an obstacle) as an agent and applying the procedure from Equation (2.3), the obstacle's full encumbrance is not required for this step because $\mathbf{p}_{\text{closest}} = \tilde{\mathbf{p}}_j$ provides the relevant point for partitioning $\mathcal{A}$, therefore $\delta_j = 0$.

3.3 Implementation and Integration on UAV

Challenges in Integration

The algorithm's performance is influenced by the limitations of the LiDAR sensor used for environmental sensing. While the algorithm works optimally with a full 360° horizontal and 180° vertical FOV, which would require multiple sensors, the experiments used a single Livox Mid 360 LiDAR [37]. This LiDAR provides a 360° horizontal FOV and only a 59° vertical FOV, resulting in a sensing blind spot.

To solve this limitation, several modifications were implemented. The LiDAR was mounted at an angle γ, as shown in Figure Fig. 3.2, to enhance ground sensing and increase forward visibility.

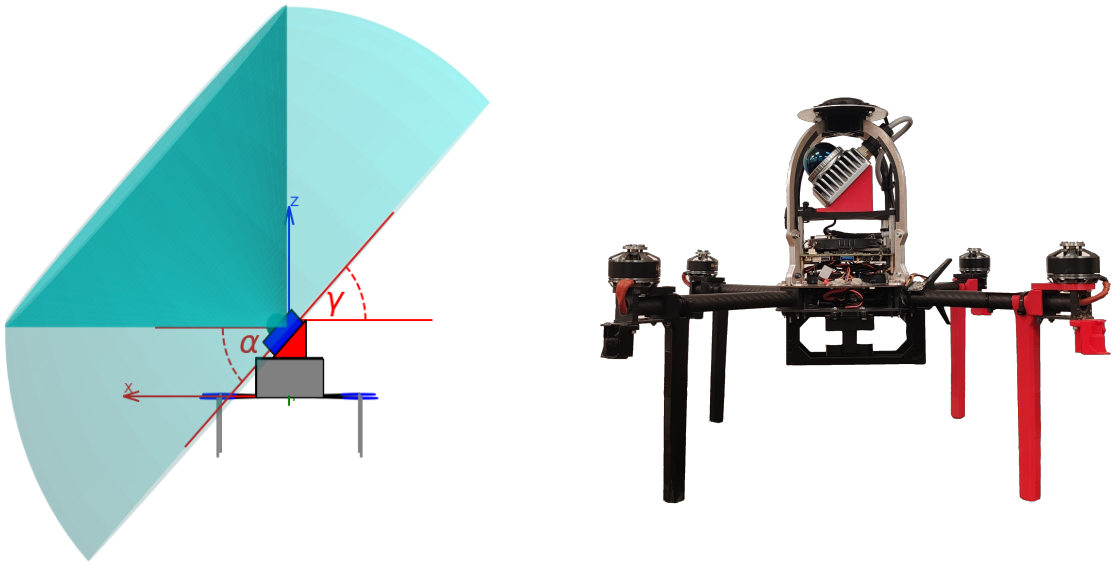


Figure 3.2: UAV and LiDAR mounting scheme. Please note that a different mounting angle was used in practical experiments, so the configuration shown here is purely for visualization purposes. Subfigure on the left shows a modeled UAV with visualized LiDAR mounting parameters, including the LiDAR elevation field of view α and tilt angle γ. Subfigure on the right displays the real UAV used in the experiment with the LiDAR mounted according to the same configuration.

In addition to the LiDAR mounting angle, a software modification is implemented to account for the LiDAR's limited FOV. The algorithm uses the map to partition cell $\mathcal{S}$ into cell $\mathcal{A}$, from which the centroid is computed to guide UAV movement. To ensure safety, the UAV should only move within its visible FOV. However, the map information from areas outside the current FOV can still be valuable.

Therefore, the following constraint is introduced - the UAV maintains its yaw rotation towards the current centroid $\mathbf{c}_{\mathcal{A}}$ and moves towards the centroid only when the centroid is

positioned within a $\pm\frac{\pi}{4}$ rad angular range from the direction the LiDAR is tilted towards (the UAV's x-axis). This $\frac{\pi}{2}$ rad forward sector for movement is determined by the UAV's physical structure, specifically to prevent the UAV from moving into areas where mounting rods for GPS obstruct active sensing.

To integrate the LiDAR's FOV into cell $\mathcal{S}$, two planes are defined based on the LiDAR's mounting configuration and vertical FOV. Let $\mathbf{e}_z$ be a unit vector along the z-axis, $R_\gamma \in SO(3)$ the tilt of the LiDAR, and $R_{rpy} \in SO(3)$ the roll-pitch-yaw rotation. The normal vectors, $\mathbf{n}_1$ and $\mathbf{n}_2$ defining two planes are given by:

$$\mathbf{n}_1 = R_{rpy} R_y(\gamma) \mathbf{e}_z \tag{3.2}$$

$$\mathbf{n}_2 = R_{rpy} R_y(\gamma - \alpha) \mathbf{e}_z, \tag{3.3}$$

where α is the LiDAR's FOV and γ is the mounting angle of the LiDAR.

Cell $\mathcal{S}$ is then modified by excluding points that lie outside the LiDAR's FOV, defined by these two planes:

$$\mathcal{S}_i' = \{\mathbf{q} \in \mathcal{S}_i \mid \mathbf{n}_1 \cdot (\mathbf{q} - \mathbf{p}_L) \geq 0 \wedge \mathbf{n}_2 \cdot (\mathbf{q} - \mathbf{p}_L) \leq 0\}, \tag{3.4}$$

where $\mathcal{S}_i'$ is the modified cell $\mathcal{S}_i$ and $\mathbf{p}_L$ is the exact position of the LiDAR sensor.

This modification effectively restricts the points considered in the calculation of the centroid to only those within the LiDAR's FOV.

To integrate map information from the surrounding area outside of the LiDAR's current FOV, the following procedure is used. Firstly, both cells $\mathcal{S}_i$ and $\mathcal{S}_i'$ are created. These cells are partitioned into cells $\mathcal{A}_i$ and $\mathcal{A}_i'$ using voxels. Cell $\mathcal{A}_i'$ is partitioned using voxels that are actively sensed, while cell $\mathcal{A}_i$ is partitioned using the voxels from the whole surroundings.

If the centroid computed from cell $\mathcal{A}_i$, $\mathbf{c}_{\mathcal{A}_i}$, is within the cell $\mathcal{A}_i'$, the UAV is commanded to move towards $\mathbf{c}_{\mathcal{A}_i}$ while simultaneously rotating its yaw towards it. However, if $\mathbf{c}_{\mathcal{A}_i}$ lies outside cell $\mathcal{A}_i'$, it is projected onto the closest point on the boundary of cell $\mathcal{A}_i'$. As the UAV only moves if the centroid is within a certain angle in front of it, this projection onto the boundary of $\mathcal{A}_i'$ ensures that the UAV primarily rotates towards the centroid until the centroid is within an angle in front of the UAV. This control logic is formally defined as follows:

$$\begin{cases} \text{move and rotate towards } \mathbf{c}_{\mathcal{A}_i} & \text{if } \mathbf{c}_{\mathcal{A}_i} \in \mathcal{A}_i' \wedge \alpha \in [-\frac{\pi}{4}, \frac{\pi}{4}] \\ \text{only rotate, do not move towards } \mathbf{c}_{\mathcal{A}_i} & \text{otherwise,} \end{cases} \tag{3.5}$$

where $\alpha = \text{atan2}((\mathbf{c}_{\mathcal{A}_i} - \mathbf{p}_i)_y, (\mathbf{c}_{\mathcal{A}_i} - \mathbf{p}_i)_x) - \text{yaw}_i$

This procedure effectively utilizes the information from the map while ensuring that the UAV moves only within its actively sensed FOV.

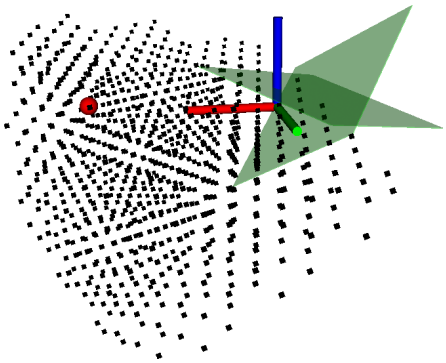


Figure 3.3: Rviz visualization of LiDAR field-of-view (FOV) planes. The UAV is represented by the XYZ coordinate axes. The green squares represent the two planes, defined by normal vectors $\mathbf{n}_1$ and $\mathbf{n}_2$, used to slice cell $\mathcal{S}_i$ and determine the modified cell $\mathcal{S}_i'$. The red sphere represents the centroid $\mathbf{c}_{\mathcal{A}_i}$.

3.4 Experimental Results

This section presents the findings from both simulated and real-world experiments conducted to evaluate the proposed approach.

Simulation Setup

In the simulations, a virtual forest environment was created. Initially, a raycasting approach from the UAV to the simulated trees was considered for obstacle detection. However, due to its computational cost, a static point cloud representation of the entire forest was generated and published instead. The forest was modeled as a collection of trees, with trunks represented by cylinders and crowns by spheres. The forest was generated within a rectangular area defined by the opposing corner points (-20.0 m, -32.0 m) and (20.0 m, 32.0 m) in the x-y plane, with its sides aligned parallel to the x and y axes. A total of 100 trees were randomly generated within this area, with a minimum separation distance of 3 meters between them. The trees were generated with the following parameter variances:

- Trunk radius: 0.5 ± 0.2 m
- Tree height: 7.5 ± 2.0 m
- Crown radius: 2.5 ± 0.5 m

The UAV's starting position was set at one corner of the rectangular area (-22.0, -33.0, 2.0), and the goal position was set at the opposite corner (22.0, 33.0, 5.0).

Note that the right-hand rule and z rule, described in the previous chapter, are not required for effective coordination in this static forest environment. Also, the Point-LIO estimator and Bonxai mapping were not simulated.

A separate simulation was also run to compare the 2D RBL algorithm with its 3D extension, using a circle crossing setup in the forest.

The specific values of the parameters used in the experiments are summarized in the following table 3.1:

Table 3.1: Parameters Used in Experiments

Parameter	r_s [m]	δ_i [m]	$d_1 = d_3 = d_5$ [m]	$d_2 = d_4 = d_6$ [m]
Value	3.5	0.5	0.5	1.0

Parameter	Update rate [Hz]	γ [°]	β_i^D [ ]	η [ ]
Value	10.0	20.0	0.5	0.9

The UAV dynamics are the same as described in the previous chapter and detailed in Table 2.1. Each simulation was run 10 times with the same forest.

- **Performance in Simulated Environments**

The first simulation results highlight the proposed solution capability in navigating inside dense forest environments. The UAV demonstrated effective obstacle avoidance, reacting appropriately to simulated trees while maintaining an efficient trajectory without unnecessary detours. Convergence towards the designated goal was consistently achieved in a stable manner.

	SR [%]	$\overline{L}$ [m]	$\overline{t}$ [s]	$\overline{v}$ [m/s]
Results	100.00	91.99 ± 0.96	150.89 ± 0.88	0.85 ± 0.00

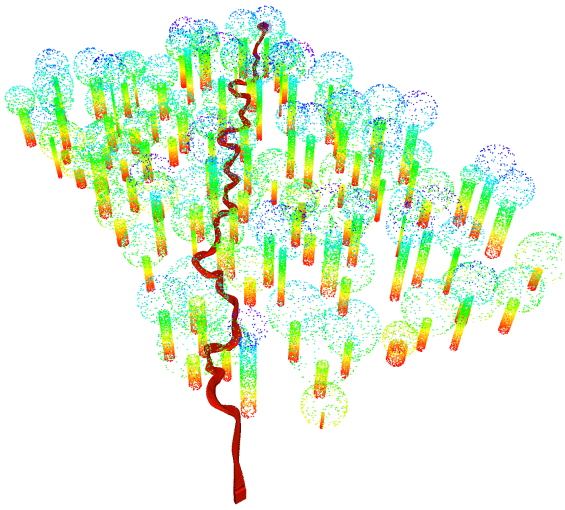


Figure 3.4: Visualization of simulated forest and the corresponding path of the UAV, indicated by a red line.

Additional simulations directly compared the 2D and extended 3D RBL algorithms in the static forest. These results confirmed that UAVs using the 3D RBL were capable of consistently reaching the goal by exploring vertical dimensions, navigating above or below tree crowns. The 2D RBL, however, sometimes failed by becoming stuck between these obstacles.

	SR [%]	$\overline{L}$ [m]	$\bar{t}$ [s]	$\bar{t}_{\max}$ [s]	$\bar{v}$ [m/s]
RBL 2D	0.00	30.27 ± 0.12	∞	∞	0.00 ± 0.00
RBL $3D_{rule}$	**100.00**	**28.40 ± 10.68**	**50.90 ± 33.51**	**99.86 ± 16.01**	**0.54 ± 0.27**

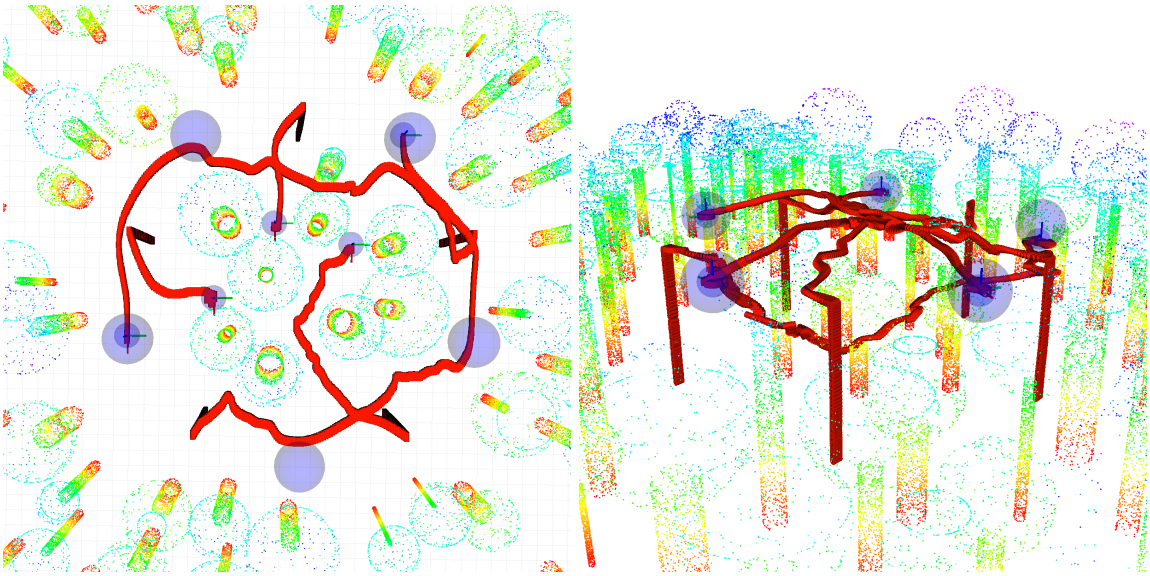


Figure 3.5: Illustrative comparison of 2D and 3D RBL navigation capabilities in a simulated forest environment. Left: The 2D algorithm is shown in a deadlock. Right: The 3D algorithm demonstrates its ability to utilize vertical space.

Real-World Experiments in a Forest Environment

Following the successful simulation validation, the RBL algorithm was tested through real-world experiments conducted in a forest environment.

- **Location:**
  The forest site for initial real-world experiments was chosen for its relatively open characteristics, allowing for straightforward system validation. Subsequent flight tests were then conducted in more challenging and densely obstructed environments. To increase the navigational challenge and better test the obstacle avoidance capabilities, additional branches were placed near trees within the flight area after a few successful flights.
- **UAV Platform:**
  A custom-built multirotor UAV from the MRS group, the X500, was used for these experiments. It was equipped with LiDAR mounted on top for environmental perception and state estimation. While other sensors like GPS (unreliable in the forest), a Garmin altimeter, and barometer were present on the platform, they were not used.

Performance in Real-World Experiment

For the real-world experiment I picked 2 flights to describe and one fail flight. For these flights the point cloud filtration parameters were set to discard points closer than 1.0 meter (to avoid detecting the UAV's own propellers) and farther than 40 meters. For both flights a video has been created to show the performance aggressive flight [27], conservative flight[28].

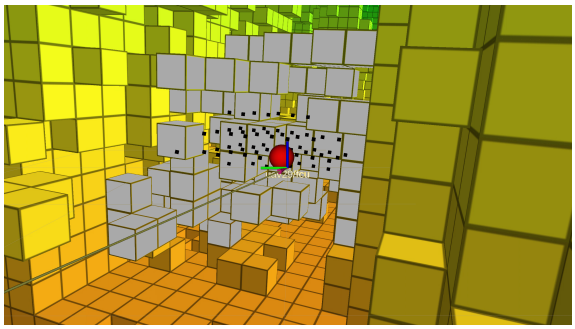


Figure 3.6: Visualization of a moment where the UAV incorrectly mapped leaves as obstacles directly in its path.

Figure 3.7: The UAV navigating in a forest during an experiment.

Table 3.2: Parameters Used in Experiments

Parameter	Values Conservative	Values Aggressive
r_s [m]	2.0	2.0
Update rate [Hz]	10	10
δ_i [m]	0.5	0.5
$d_1 = d_3 = d_5$ [m]	0.5	0.5
$d_2 = d_4 = d_6$ [m]	1.0	1.0
γ [°]	20.0	20.0
β_i^D [ ]	0.5	0.1
η [ ]	0.8	0.8

	L [m]	t [s]	v [m/s]
Conservative Flight	53.84	116.40	0.46
Aggressive Flight	35.56	51.64	0.69

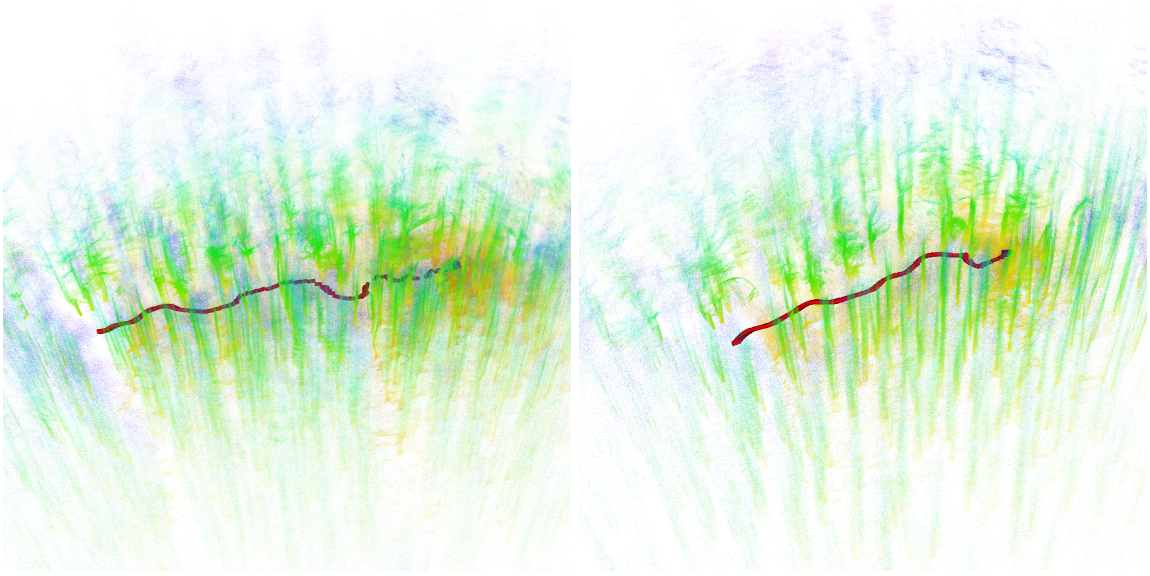


Figure 3.8: UAV flight trajectories through the forest test environment under different navigation settings. Left: Conservative parameter settings. Right: Aggressive parameter settings.

- **Comparative Analysis**

The real-world flight experiments conducted in the forest environment successfully validated the navigation capabilities of the proposed setup. The UAV generally performed as expected based on simulation results, demonstrating its ability to navigate autonomously between points A and B while avoiding static obstacles.

- **Performance and Limitations**

A primary limitation observed during real-world testing originated from the environmental mapping component, specifically the Bonxai mapping package used. While effective

for static elements, the system occasionally struggled with dynamic obstacles, such as moving leaves. These obstacles were sometimes included in the voxel grid map and, due to the map's update policy (configured conservatively for safety), were not always removed immediately even when no longer present. This could lead to situations where the UAV perceived itself as blocked (as shown in the failure case video [29]), requiring the safety pilot to land. Addressing this mapping behavior is crucial for future testing.

Comparison with 2D Baseline

A meaningful comparison can be made with the original 2D RBL implementation [6]. The key differences lie in the sensing and dimensionality:

- The 2D version relied on a 2D LiDAR for planar obstacle detection and a separate optical distance measurement sensor to maintain a constant flight altitude. Remapping obstacles in 2D is often simpler.
- The 3D implementation utilizes a 3D LiDAR for perception in three dimensions and must actively manage altitude based on the perceived environment, treating the ground itself as an obstacle.

A notable behavior observed in the 3D tests was the UAV's tendency to initially increase altitude after takeoff, effectively moving away from the ground obstacle, before potentially descending again as it got closer to the goal. This behavior highlights the algorithm's capability to manage its vertical position based on the 3D environment. Regarding computational efficiency, while the 3D RBL algorithm inherently involves greater complexity than its 2D counterpart (needing more points for 3D cell representation and more computational power for their partitioning), it nonetheless demonstrated manageable efficiency throughout the simulations and real-world flights. Illustrative examples of successful flights showcasing different parameter settings (aggressive vs. conservative) are provided in [27] and [28].

Future Work

The mapping limitations highlight key areas for future work. Potential solutions include:

- **Dynamic Obstacle Handling:**
  Implementing techniques to segment and filter out dynamic objects from the map representation such as [38].
- **Reactive Navigation:**
  Exploring strategies that rely more on direct sensor input for immediate obstacle avoidance, removing dependence on the map, though this introduces challenges in perceiving areas outside the current sensor view (e.g. behind the UAV).
- **Map-Based Strategic Replanning:**
  Future work could include developing a higher-level path replanner that leverages the collected map to navigate agents around crowded areas, thereby supplementing the RBL algorithm's local collision avoidance and enhancing overall efficiency.
- **Map Update Logic:**
  Investigating alternative mapping packages or implementing mechanisms like voxel decay instead of mapping (where unoccupied voxels gradually fade if not persistently observed) into the existing framework. While probabilistic mapping should ideally handle this, the developmental stage of the Bonxai package currently limits this capability.

Resolving these mapping challenges, particularly the robust handling of dynamic elements, is a prerequisite for advancing to more complex multi-agent experiments utilizing only onboard

lidar sensing.

3.5 Conclusion

This chapter detailed the implementation and real-world validation of the RBL algorithm, adapted for autonomous UAV navigation within a cluttered, GNSS-denied forest environment using onboard 3D LiDAR sensing. The primary objective was to implement RBL principles with real-time perception, state estimation (Point-LIO), and mapping (Bonxai) to achieve reliable point-to-point navigation.

Key technical contributions included adapting the RBL algorithm's cell partitioning to directly utilize voxelized map data created from processed LiDAR point clouds. Modifications were also introduced to effectively handle the practical limitations of a single LiDAR sensor with a restricted vertical field of view, ensuring safe convergence by constraining movement based on the actively sensed area while still leveraging map information. An alternative approach using surface reconstruction via mesh generation was investigated but considered unsuitable due to computational complexity on the UAV onboard computer.

The proposed solution's effectiveness was demonstrated through both simulation and real-world experiments conducted in a forest. The UAV successfully navigated between designated start and goal points, avoiding static obstacles like trees and adapting its altitude based on the LiDAR's perception.

Despite the overall success, the experiments identified a key limitation related to the Bonxai mapping package's handling of dynamic environmental elements, such as moving leaves, which occasionally led to navigation deadlocks. This emphasizes the importance of robust mapping with dynamic obstacles.

In conclusion, this chapter successfully demonstrated the practical application and feasibility of using a 3D RBL algorithm integrated with LiDAR sensing for autonomous UAV navigation in a challenging forest setting. This capability can be observed in these videos [27], [28].

4 Conclusion

This thesis set out to address key challenges in autonomous UAV operations. The work focused on two primary objectives: first, to extend the established two-dimensional RBL algorithm for effective multi-agent coordination in three-dimensional space, introducing new rules to enhance its performance. The second objective was validating the practical application of this extended framework for single UAV navigation in a complex, GNSS-denied forest environment using onboard LiDAR sensing.

A fundamental core contribution of this work is the successful extension of the RBL algorithm to 3D. This involved key modifications such as adapting Voronoi cell partitioning for three dimensions, modifying existing rules, and introducing a new elevation rotation angle designed to adjust the agent's vertical destination and improve spatial distribution. Simulations of multi-agent scenarios, including crossing circle and sphere formations, demonstrated that these extensions successfully enabled coordination in three dimensions. Notably, analysis revealed a trend where the performance advantage of the 3D RBL approach with these rules, compared to the basic 2D version, increased with the number of agents.

Building upon this, the second major contribution was the practical implementation and real-world validation of the 3D RBL algorithm for single UAV navigation. This involved integrating the algorithm with LiDAR-based perception, Point-LIO for state estimation, and the Bonxai voxel-based mapping. Key technical advancements included adapting the RBL cell partitioning to directly utilize voxelized map data and implementing software modifications to effectively handle the practical limitations of a single LiDAR sensor with a restricted vertical field of view, ensuring safe movement based on actively sensed areas while still leveraging map information. Real-world experiments in a forest environment successfully demonstrated the UAV's ability to navigate autonomously between designated start and goal points, avoiding static obstacles like trees and adapting its altitude based on LiDAR perception, as shown in the provided flight videos [27], [28].

The significance of these findings lies in advancing distributed communication-less control strategies for UAVs operating in three dimensions. The successful 3D extension of RBL offers a scalable approach for multi-agent systems, while the demonstration of LiDAR-based navigation in a real forest highlights the potential for autonomous operations in GPS-denied settings.

For the multi-agent 3D RBL, while simulations indicated advantages, the degree of improvement was occasionally less noticeable, possibly a result of conservative motion constraints and parameter settings that could be refined through further tuning. For optimal performance of certain directional rules, the algorithm needs an initial shared frame of reference, which can be established using common onboard sensors such as a magnetometer. In the single-agent forest navigation, the primary challenge encountered was the Bonxai mapping package, particularly its handling of dynamic environmental elements like moving leaves, which occasionally led to navigation deadlocks [29]. This highlights the dependence of the current navigation approach on an accurate map representation.

Future work should therefore focus on improving the system's ability to handle dynamic obstacles. This could involve implementing another mapping algorithm or more sophisticated map update and filtering techniques, or developing strategies that allow the UAV to react more directly to sensor perceptions of the environment, reducing dependence on the map.

In conclusion, this thesis has successfully demonstrated the extension of the RBL algorithm to 3D for multi-agent coordination in simulation and its practical application for single UAV navigation in a real-world environment using LiDAR. While challenges, particularly in perception and mapping in dynamic settings, remain, the presented work provides a valuable contribution and a solid foundation for future advancements in autonomous UAV navigation and coordination.

5 References

[1] M. R. Fikri, T. Candra, K. Saptaji, A. N. Noviarini, and D. A. Wardani, *A review of implementation and challenges of unmanned aerial vehicles for spraying applications and crop monitoring in indonesia*, 2023. arXiv: 2301.00379 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2301.00379.

[2] J. Xing, G. Cioffi, J. Hidalgo-Carrió, and D. Scaramuzza, *Autonomous power line inspection with drones via perception-aware mpc*, 2023. arXiv: 2304.00959 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2304.00959.

[3] J. Zhong, Q. Zhou, M. Li, A. Gruen, and X. Liao, *A novel solution for drone photogrammetry with low-overlap aerial images using monocular depth estimation*, 2025. arXiv: 2503.04513 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2503.04513.

[4] Insider Intelligence, *Why Amazon, UPS and even Domino's is investing in drone delivery services*, [Online]. [Online]. Available: https://www.insiderintelligence.com/insights/drone-delivery-services/.

[5] National Aeronautics and Space Administration (NASA), *Ingenuity mars helicopter*, [Online], Accessed: 2025-05-13. [Online]. Available: https://science.nasa.gov/mission/mars-2020-perseverance/ingenuity-mars-helicopter/.

[6] M. Boldrer, A. Serra-Gomez, L. Lyons, V. Kratky, J. Alonso-Mora, and L. Ferranti, "Rule-based lloyd algorithm for multi-robot motion planning and control with safety and convergence guarantees," *arxiv*, 2024. [Online]. Available: https://arxiv.org/abs/2310.19511.

[7] A. Krnjaic, R. D. Steleac, J. D. Thomas, *et al.*, *Scalable multi-agent reinforcement learning for warehouse logistics with robotic and human co-workers*, 2022. arXiv: 2212.11498 [cs.LG].

[8] R. Zheng and S. Li, "Mcca: A decentralized method for collision and deadlock avoidance with nonholonomic robots," *IEEE Robotics and Automation Letters*, vol. PP, pp. 1–8, Mar. 2024. DOI: 10.1109/LRA.2024.3358623.

[9] X. Zhou, J. Zhu, H. Zhou, C. Xu, and F. Gao, *Ego-swarm: A fully autonomous and decentralized quadrotor swarm system in cluttered environments*, 2021. arXiv: 2011.04183 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2011.04183.

[10] Y. Chen, M. Guo, and Z. Li, *Deadlock resolution and recursive feasibility in mpc-based multi-robot trajectory generation*, 2024. arXiv: 2202.06071 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2202.06071.

[11] A. Thomas, F. Mastrogiovanni, and M. Baglietto, *Probabilistic collision constraint for motion planning in dynamic environments*, 2021. arXiv: 2104.01659 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2104.01659.

[12] R. Han, S. Chen, S. Wang, *et al.*, *Reinforcement learned distributed multi-robot navigation with reciprocal velocity obstacle shaped rewards*, 2022. arXiv: 2203.10229 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2203.10229.

[13] Q. Tan, T. Fan, J. Pan, and D. Manocha, *Deepmnavigate: Deep reinforced multi-robot navigation unifying local & global collision avoidance*, 2020. arXiv: 1910.09441 [cs.MA]. [Online]. Available: https://arxiv.org/abs/1910.09441.

[14] R. Jin, Y. Wang, Z. Gao, X. Niu, L.-T. Hsu, and J. Liu, "Dynavig: Monocular vision/ins/gnss integrated navigation and object tracking for agv in dynamic scenes," 2022. arXiv: 2211.14478 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2211.14478.

[15] T. Takebayashi, R. Miyagusuku, and K. Ozaki, "Development of magnetic-based navigation by constructing maps using machine learning for autonomous mobile robots in real environments," *Sensors*, vol. 21, no. 12, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/12/3972.

[16] Z. Liu, A. Amini, S. Zhu, S. Karaman, S. Han, and D. Rus, "Efficient and robust lidar-based end-to-end navigation," 2021. arXiv: 2105.09932 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2105.09932.

[17] B. Ertl, *3d voronoi mesh of 25 random points with 0.3 opacity and points*, Own work, licensed under CC BY-SA 4.0, Accessed: 2025-02-27, 2023. [Online]. Available: https://commons.wikimedia.org/w/index.php?curid=38534275.

[18] WalkingRadiance, *3d voronoi mesh of 25 random points with 0.3 opacity and points*, Own work, licensed under CC BY-SA 4.0, Accessed: 2025-02-27, 2023. [Online]. Available: https://commons.wikimedia.org/w/index.php?curid=127768757.

[19] C. i. P. Multi-Robot Systems Group, *Mrs uav system*, Accessed: 2025-03-10, 2024. [Online]. Available: https://github.com/ctu-mrs/mrs_uav_system.

[20] V. Walter, M. Saska, and A. Franchi, "Fast mutual relative localization of uavs using ultraviolet led markers," *2018 International Conference on Unmanned Aircraft System (ICUAS 2018)*, 2018. [Online]. Available: https://dronument.cz/sites/default/files/uploaded/mutual_localizations_leds_naki.pdf.

[21] V. Walter and M. Group, *Uvdar_multirobot_simulator*, https://github.com/ctu-mrs/uvdar_multirobot_simulator, Accessed: 2025-04-24, 2024.

[22] A. Ahmad, D. Bonilla Licea, G. Silano, T. Báča, and M. Saska, "Pacnav: A collective navigation approach for uav swarms deprived of communication and external localization," *Bioinspiration & Biomimetics*, no. 6, 2022. DOI: 10.1088/1748-3190/ac98e6. [Online]. Available: http://dx.doi.org/10.1088/1748-3190/ac98e6.

[23] D. Mezey, R. Bastien, Y. Zheng, *et al.*, *Purely vision-based collective movement of robots*, 2024. arXiv: 2406.17106. [Online]. Available: https://arxiv.org/abs/2406.17106.

[24] *F4f robofly swarm*, Fly4Future. [Online]. Available: https://shop.fly4future.com/product/f4f-robofly-swarm/.

[25] F. e. České vysoké učení technické v Praze, *Detekce kůrovce, obrana proti nepřátelskému převzetí dronů a roje úzce spolupracujících vzdušných robotů.* 2023. [Online]. Available: https://fel.cvut.cz/import/tz/2023-0426-tz_fel_cvut_mrs_temesvar.pdf.

[26] Fly4Future, *Drones for inspection of power lines*. [Online]. Available: https://fly4future.com/custom-drones/drones-for-inspection-of-power-lines/.

[27] M. Kamler, *Drone forest navigation – aggressive parameters (best performance)*, 2025. [Online]. Available: https://www.youtube.com/watch?v=DFt222gnA_w&ab_channel=MichalKamler.

[28] M. Kamler, *Drone forest navigation - conservative parameters (slow and safe flight)*, 2025. [Online]. Available: https://www.youtube.com/watch?v=AJPk0yVCPUo&ab_channel=MichalKamler.

[29] M. Kamler, *Drone forest navigation – mapping failure (remapping error)*, 2025. [Online]. Available: https://www.youtube.com/watch?v=JVmW0qfwP3c&ab_channel=MichalKamler.

[30] D. He, W. Xu, N. Chen, F. Kong, C. Yuan, and F. Zhang, "Point-lio: Robust high-bandwidth light detection and ranging inertial odometry," *Advanced Intelligent Systems*, vol. 5, no. 7, p. 2 200 459, 2023. DOI: https://doi.org/10.1002/aisy.202200459. eprint: https://advanced.onlinelibrary.wiley.com/doi/pdf/10.1002/aisy.202200459. [Online]. Available: https://advanced.onlinelibrary.wiley.com/doi/abs/10.1002/aisy.202200459.

[31] Y. Li and J. Ibanez-Guzman, "Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems," *IEEE Signal Processing Magazine*, vol. 37, no. 4, 50–61, Jul. 2020, ISSN: 1558-0792. DOI: 10.1109/msp.2020.2973615. [Online]. Available: http://dx.doi.org/10.1109/MSP.2020.2973615.

[32] S. Debnath, M. Paul, and T. Debnath, "Applications of lidar in agriculture and future research directions," *Journal of Imaging*, vol. 9, no. 3, 2023, ISSN: 2313-433X. DOI: 10.3390/jimaging9030057. [Online]. Available: https://www.mdpi.com/2313-433X/9/3/57.

[33] N. Lopac, I. Jurdana, A. Brnelić, and T. Krljan, "Application of laser systems for detection and ranging in the modern road transportation and maritime sector," *Sensors*, vol. 22, no. 16, 2022, ISSN: 1424-8220. DOI: 10.3390/s22165946. [Online]. Available: https://www.mdpi.com/1424-8220/22/16/5946.

[34] B. S. Radu B. Rusu, *The voxelgrid class, point cloud library*, Accessed: 2025-03-28. [Online]. Available: https://pointclouds.org/documentation/classpcl_1_1_voxel_grid.html.

[35] D. Faconti, *Bonxai: Fast, hierarchical, sparse voxel grid*, Accessed: 2025-05-20, 2025. [Online]. Available: https://github.com/facontidavide/Bonxai.

[36] P. C. L. (PCL), *Pcl::kdtree class reference*. [Online]. Available: https://pointclouds.org/documentation/classpcl_1_1_kd_tree.html.

[37] Livox Technology Company Limited, *Mid-360 LiDAR Sensor*, Accessed: 2025-05-02, 2023. [Online]. Available: https://www.livoxtech.com/mid-360.

[38] Y. Jia, T. Wang, X. Chen, and S. Shao, *Trlo: An efficient lidar odometry with 3d dynamic object tracking and removal*, 2024. arXiv: 2410.13240 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2410.13240.

A Experimental Results

- $N = 5$ **Circular Formation:**

Table A.1: Simulation Results for N=5 Circular Formation.

	SR [%]	$\overline{L}$ [m]	$\overline{t}$ [s]	$\overline{t}_{\max}$ [s]	$\overline{v}$ [m/s]
RBL 2D	100.00	21.06 ± 0.10	**25.15 ± 0.21**	**25.15 ± 0.19**	**0.83 ± 0.01**
RBL 3D	100.00	20.77 ± 0.29	26.04 ± 0.51	26.79 ± 0.27	0.79 ± 0.02
RBL $3D_{clipped}$	100.00	**20.60 ± 0.24**	26.73 ± 0.47	27.39 ± 0.28	0.77 ± 0.02
RBL $3D_{rule}$	100.00	20.97 ± 0.52	25.54 ± 0.97	26.72 ± 0.60	0.81 ± 0.03

- $N = 10$ **Circular Formation:**

Table A.2: Simulation Results for N=10 Circular Formation.

	SR [%]	$\overline{L}$ [m]	$\overline{t}$ [s]	$\overline{t}_{\max}$ [s]	$\overline{v}$ [m/s]
RBL 2D	100.00	22.95 ± 1.64	30.79 ± 2.28	34.73 ± 0.77	0.74 ± 0.07
RBL 3D	100.00	**22.22 ± 0.89**	30.05 ± 2.61	34.39 ± 4.19	0.73 ± 0.05
RBL $3D_{clipped}$	100.00	22.31 ± 0.69	30.22 ± 1.83	33.22 ± 0.93	0.73 ± 0.04
RBL $3D_{rule}$	100.00	22.38 ± 0.88	**28.80 ± 2.30**	**32.35 ± 1.25**	**0.77 ± 0.05**

- $N = 15$ **Circular Formation:**

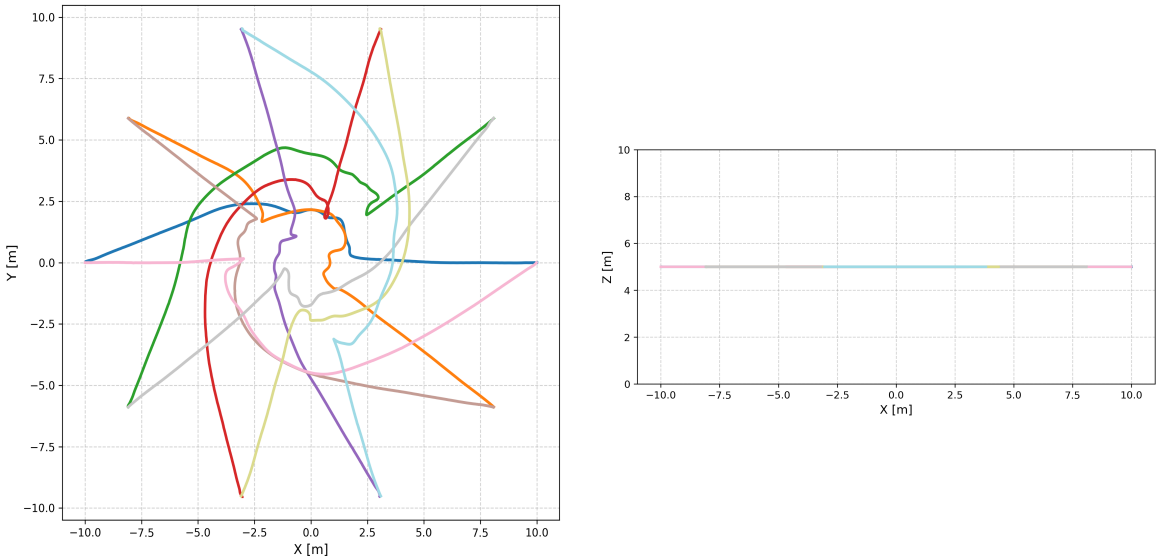
Table A.3: Simulation Results for N=15 Circular Formation.

	SR [%]	$\overline{L}$ [m]	$\overline{t}$ [s]	$\overline{t}_{\max}$ [s]	$\overline{v}$ [m/s]
RBL 2D	100.00	23.56 ± 1.75	33.89 ± 2.48	38.75 ± 1.45	0.69 ± 0.06
RBL 3D	100.00	22.36 ± 0.97	30.65 ± 3.07	35.77 ± 3.72	0.72 ± 0.07
RBL $3D_{clipped}$	100.00	**22.32 ± 0.81**	30.69 ± 2.61	34.50 ± 0.47	0.72 ± 0.06
RBL $3D_{rule}$	100.00	22.64 ± 0.95	**29.67 ± 2.54**	**34.27 ± 0.88**	**0.76 ± 0.06**

- $N = 10$ **Spherical Formation:**

Table A.4: Simulation Results for N=10 Spherical Formation.

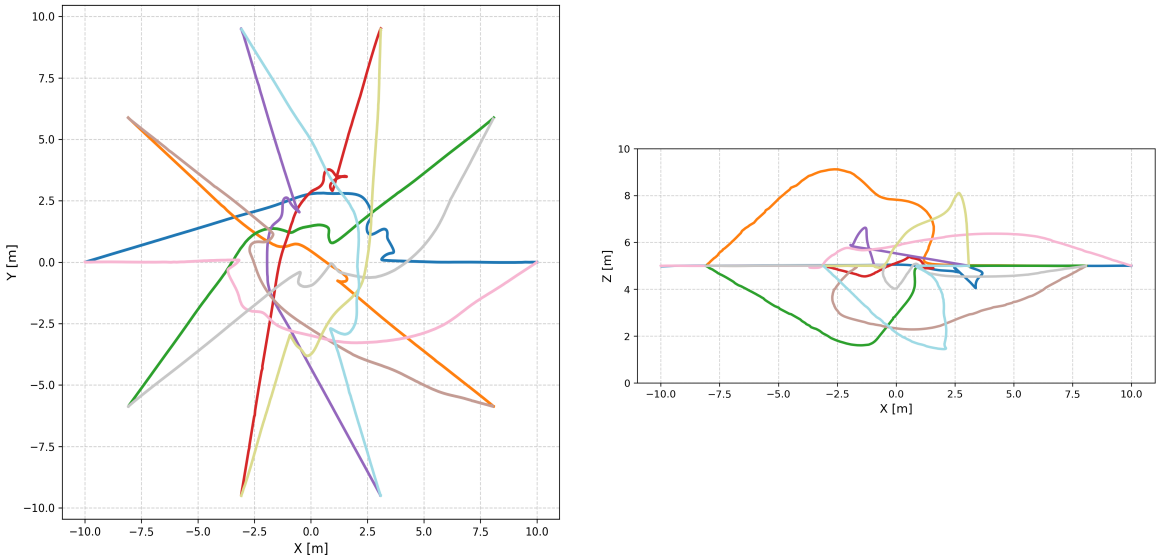
	SR [%]	$\overline{L}$ [m]	$\overline{t}$ [s]	$\overline{t}_{\max}$ [s]	$\overline{v}$ [m/s]
RBL 3D	100.00	14.32 ± 1.52	27.76 ± 3.06	32.48 ± 2.26	0.51 ± 0.05
RBL $3D_{rule}$	100.00	**14.06 ± 0.97**	**27.17 ± 2.66**	**31.86 ± 1.24**	**0.55 ± 0.06**



(a) Top-Down View (X-Y)

(b) Side View (X-Z)

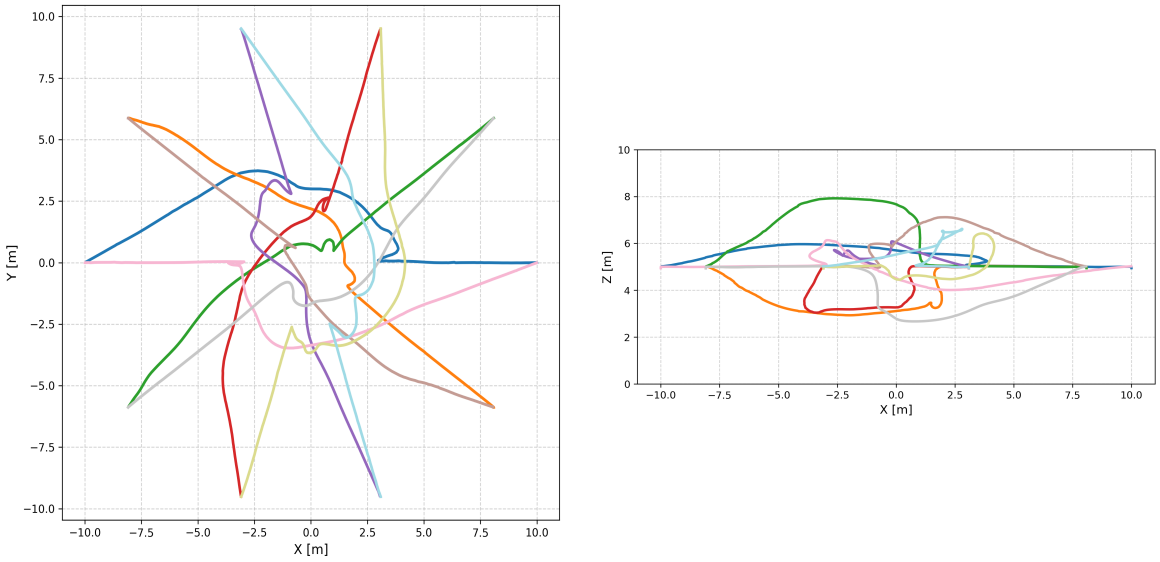
Figure A.1: Trajectories in a 2D circular crossing.



(a) Top-Down View (X-Y)

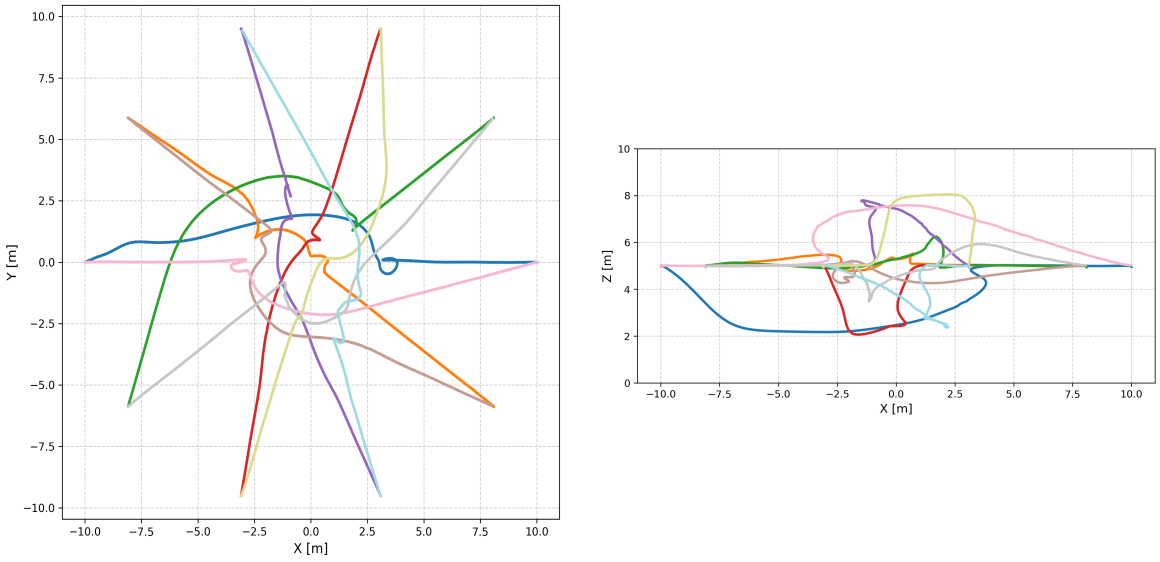
(b) Side View (X-Z)

Figure A.2: Trajectories in a 3D circular crossing.



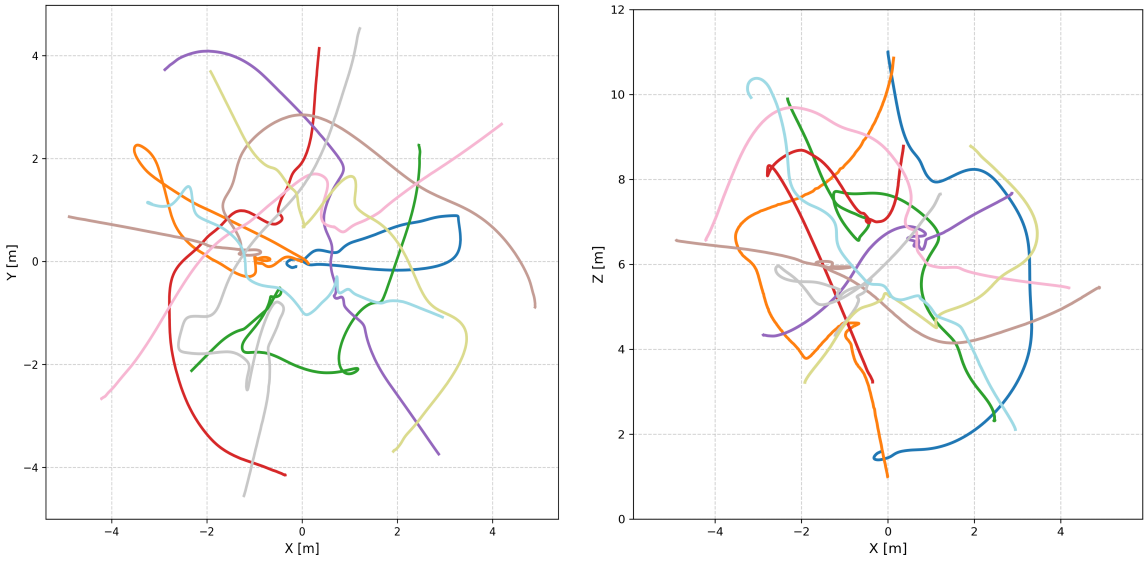
(a) Top-Down View (X-Y) (b) Side View (X-Z)

Figure A.3: Effect of Z-axis clipping on the circular crossing.



(a) Top-Down View (X-Y) (b) Side View (X-Z)

Figure A.4: Effect of the Z-axis rule on the circular crossing.



(a) Top-Down View (X-Y)

(b) Side View (X-Z)

Figure A.5: Trajectories in a 3D spherical crossing.