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: Ing. Manuel Boldrer, Ph.D.

Acknowledgments

Firstly, I would like to express my gratitude to my supervisor.

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

Declaration

I declare that presented work was developed independently, and that I have listed all sources of information used within, in accordance with the Methodical instructions for observing ethical principles in preparation of university theses.

Date ....

Abstract

The study of autonomous Unmanned Aerial Vehicles (UAVs) has become a prominent sub-field of mobile robotics.

Keywords TODOUnmanned Aerial Vehicles, Automatic Control

Abbreviations

UAV Unmanned Aerial Vehicle

Contents

1 Introduction **1**
1.1 Related works . . . 1
1.2 Contributions . . . 1
1.3 Mathematical notation . . . 1

2 Extension of the Rule-Based Lloyd Algorithm to 3D **2**
2.1 Introduction to the RBL . . . 2
2.1.1 Overview . . . 2
2.1.2 Applications and Limitations in 2D . . . 2
2.1.3 Key principles . . . 2
2.2 Extension of the RBL algorithm to 3D . . . 4
2.2.1 Motivation for 3D Extension . . . 4
2.2.2 Differences between 2D and 3D . . . 4
2.2.3 Additional Constraints and Modifications . . . 5
2.3 Simulation and Results Analysis . . . 6
2.3.1 Simulation Enviroment . . . 6
2.3.2 Simulation Scenarios . . . 6
2.3.3 N = 5 circular . . . 7
2.3.4 N = 10 circular . . . 7
2.3.5 N = 15 circular . . . 7
2.3.6 N = 10 sphere . . . 7
2.3.7 Practical Experiment Setup . . . 7
2.3.8 Comparisons . . . 7
2.4 Summary and Key Insights . . . 7

3 Environment Perception Using 3D LiDAR **8**
3.1 Introduction . . . 8
3.2 3D LiDAR Sensor Model and Simulation Setup . . . 8
3.3 Object Detection and Approximation . . . 8
3.4 Integration with RBL Algorithm . . . 8
3.5 Simulation Results . . . 8
3.6 Discussion . . . 8
3.7 Summary . . . 8

4 Researching on possible enhancements of the algorithm for example using neural networks or learning-based techniques. **9**

5 Conclusion **10**

6 References **11**

A Appendix A **12**

1 Introduction

TODO

1.1 Related works

TODO

1.2 Contributions

TODO

1.3 Mathematical notation

TODO

$\mathcal{N}_i$	set of neighboring agents for the i-th agent
$\mathbf{x}$, $\boldsymbol{\alpha}$	vector, pseudo-vector, or tuple
$\hat{\mathbf{x}}$, $\hat{\boldsymbol{\omega}}$	unit vector or unit pseudo-vector
$\hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2, \hat{\mathbf{e}}_3$	elements of the *standard basis*
$\mathbf{X}, \boldsymbol{\Omega}$	matrix
$\mathbf{I}$	identity matrix
$x = \mathbf{a}^\intercal \mathbf{b}$	inner product of $\mathbf{a}$, $\mathbf{b} \in \mathbb{R}^3$
$\mathbf{x} = \mathbf{a} \times \mathbf{b}$	cross product of $\mathbf{a}$, $\mathbf{b} \in \mathbb{R}^3$
$\mathbf{x} = \mathbf{a} \circ \mathbf{b}$	element-wise product of $\mathbf{a}$, $\mathbf{b} \in \mathbb{R}^3$
$\mathbf{x}_{(n)} = \mathbf{x}^\intercal \hat{\mathbf{e}}_n$	n^{th} vector element (row), $\mathbf{x}, \mathbf{e} \in \mathbb{R}^3$
$\mathbf{X}_{(a,b)}$	matrix element, (row, column)
x_d	x_d is *desired*, a reference
$\dot{x}, \ddot{x}, \dddot{x}, \ddddot{x}$	1^{st}, 2^{nd}, 3^{rd}, and 4^{th} time derivative of x
$x_{[n]}$	x at the sample n
$\mathbf{A}, \mathbf{B}, \mathbf{x}$	LTI system matrix, input matrix and input vector
SO(3)	3D special orthogonal group of rotations
SE(3)	*SO(3)* $\times \mathbb{R}^3$, special Euclidean group

Table 1.1: Mathematical notation, nomenclature and notable symbols.

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

2.1 Introduction to the RBL

TODO icnlude motivation of why extending rbl in 3d is good idea and simulation+experiment compare the time.

Overview

The algorithm is a communication-less approach designed to navigate agents from point A to point B. It relies on global positioning data, such as GPS or alternative methods for obtaining global coordinates, combined with sensor inputs that provide information about the agent's environment. These sensors can include LiDAR, depth cameras, or standard cameras with estimation techniques, allowing the agent to detect and avoid obstacles or other agents. The algorithm enables autonomous navigation without requiring direct communication between agents, making it suitable for scalable and decentralized 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 Automated Guided Vehicles (AGVs), which are commonly used in manufacturing and logistics. AGVs typically follow predefined 2D trajectories guided by visual [5], magnetic [7], or LiDAR-based navigation [6]. Additionally, 2D navigation is widely employed in robotic vacuum cleaners, enabling them to systematically cover an area while avoiding obstacles.

An obvious limitation for algorithms in 2D is scalability. 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

RBL, as presented in the original paper [1], 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 circle. This goal region is denoted as $B(\mathbf{e}_i, \epsilon)$, where $\mathbf{e}_i$ is center and ϵ is radius of goal region. Each agent is knows of its current position $\mathbf{p}_i$, encumbrance δ_i, which determines safe space around agent. Additionally, each agent also knows the positions and encumbrances of its neighboring agents $\mathcal{N}_\mathbf{i}$, agent $j \in \mathcal{N}_\mathbf{i}$ if $||\mathbf{p}_i - \mathbf{p}_j|| \leq 2r_{s,i}$, where $r_{s,i}$ is denoted as half of the sensing radius of the i-th agent. For simplicity $r_{s,i}$ is considered to be 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:

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

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 point in the mission space $\mathbf{q} \in \mathcal{Q}$ and agent's position p_i, and $\varphi_i(\mathbf{q})$ is the weighting function.

Voronoi cell 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. 2.1a. However, this standard definition of Voronoi cells does not take into account the physical space occupied by the agents, or their encumbrances. To address this, a Modified Voronoi cell is introduced, which takes into account the encumbrances of agents. This modified version adjusts the boundaries of each Voronoi cell to account for the encumbrances of neighboring agents. The modified Voronoi cell definition is as follows:

$$\tilde{V}_i = \begin{cases} \{\mathbf{q} \in Q \mid \|\mathbf{q} - \mathbf{p}_i\| \leq \|\mathbf{q} - \mathbf{p}_j\|\}, & \text{if } \Delta_{ij} \leq \frac{\|\mathbf{p}_i - \mathbf{p}_j\|}{2} \\ \{\mathbf{q} \in Q \mid \|\mathbf{q} - \mathbf{p}_i\| \leq \|\mathbf{q} - \tilde{\mathbf{p}}_j\|\}, & \text{otherwise}, \end{cases} \tag{2.3}$$

$\forall j \in \mathcal{N}_i$, where $\Delta_{ij} = \delta_i + \delta_j$ and $\tilde{\mathbf{p}}_j = \mathbf{p}_j + 2(\Delta_{ij} - \frac{\|\mathbf{p}_i - \mathbf{p}_j\|}{2}) \frac{\mathbf{p}_i - \mathbf{p}_j}{\|\mathbf{p}_i - \mathbf{p}_j\|}$. Together with 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 $\mathcal{A}_i = \tilde{V}_i \cap \mathcal{S}_i$.

Convergence to goal region $B(\mathbf{e}_i, \epsilon)$ 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:

$$\varphi_i(\mathbf{q}) = \exp\left(-\frac{\|\mathbf{q} - \bar{\mathbf{p}}_i\|}{\beta_i}\right), \tag{2.5}$$

where

$$\dot{\beta}_i(A_i) = \begin{cases} -\beta_i & \text{if } \|\mathbf{c}_{A_i} - \mathbf{p}_i\| < d_1 \wedge \|\mathbf{c}_{A_i} - \mathbf{c}_{S_i}\| > d_2, \\ -(\beta_i - \beta_i^D) & \text{otherwise}. \end{cases} \tag{2.6}$$

$$\dot{\bar{\mathbf{p}}}_i = \begin{cases} -(\bar{\mathbf{p}}_i - R^{\mathbf{p}_i}(\frac{\pi}{2} - \epsilon)\mathbf{e}_i) & \text{if } \|\mathbf{c}_{A_i} - \mathbf{p}_i\| < d_3 \wedge \|\mathbf{c}_{A_i} - \mathbf{c}_{S_i}\| > d_4, \\ -(\bar{\mathbf{p}}_i - \mathbf{e}_i) & \text{otherwise}, \end{cases} \tag{2.7}$$

$$\bar{\mathbf{p}}_i = \begin{cases} \mathbf{e}_i & \text{if } \|\mathbf{p}_i - \mathbf{c}_{\mathcal{A}_i}\| > \|\mathbf{p}_i - \mathbf{c}_{\mathcal{A}_i}\|, \\ R^{\mathbf{p}_i}(\frac{\pi}{2} - \epsilon)\mathbf{e}_i & \text{otherwise}. \end{cases}$$

The position of the centroid $\mathbf{c}_{V_i}$ of a region V_i, weighted by the function $\varphi_i(\mathbf{q})$, is used to guide the control. It is defined as:

$$\mathbf{c}_{V_i} = \frac{\int_{V_i} \mathbf{q} \varphi_i(\mathbf{q}) \, d\mathbf{q}}{\int_{V_i} \varphi_i(\mathbf{q}) \, d\mathbf{q}}, \tag{2.8}$$

where $\mathbf{q}$ represents the position vector, and $\varphi_i(\mathbf{q})$ is a weighting function. The centroid $\mathbf{c}_{V_i}$ serves as the target for the control system, directing the system toward the weighted center of the region.

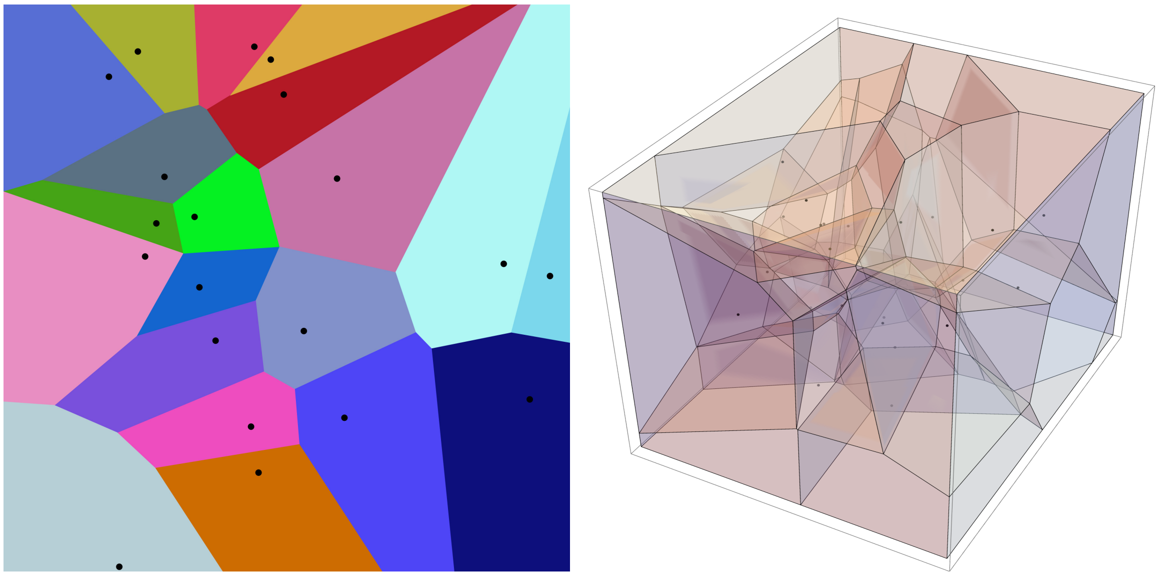
2.2 Extension of the RBL algorithm to 3D

Motivation for 3D Extension

In many practical applications, agents must operate in three-dimensional spaces, considering not only horizontal movement but also vertical positioning. A 3D extension is necessary to navigate complex environments that feature obstacles in all directions. In 2D, agents are restricted to a flat plane, which simplifies navigation but limits the ability to interact with objects and environments that exist in the third dimension. The ability to utilize vertical space can enhance energy efficiency, as agents can optimize their paths by ascending or descending to avoid obstacles or to find more favorable environmental conditions, such as discovering more open space. The transition from 2D to 3D also opens up possibilities for more advanced movement strategies, such as navigating through multi-level environments or optimizing trajectories by utilizing vertical space. Moreover, the use of 3D models allows for more accurate representations of real-world scenarios, where elevation plays a crucial role in decision-making and task execution.

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 $\tilde{V}_i$: 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.



(a) Euclidean Voronoi Diagram in 2D (b) Euclidean Voronoi Diagram in 3D

Figure 2.1: (a) an example of 20 Voronoi cells in 2D [3] (b) 25 Voronoi cells in 3D [4]

Additional Constraints and Modifications

For the 3D case, several modifications are introduced. Firstly, $Z_{clipping}$ is applied to each sensing cell $\mathcal{S}_i$, constraining it within the vertical limits defined by $\min_z$ and $\max_z$:

$$\mathcal{S}_i = \{\mathbf{q} \in \mathcal{Q} \mid \|\mathbf{q} - \mathbf{p}_i\| \leq r_{s,i}, \quad \min_z \leq q_z \leq \max_z\} \tag{2.9}$$

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

Secondly, Z_{rule} is intorduced to enhance UAV avoidance. Rule rotates computed centroid by ϕ

$$\dot{\phi}_i(A_i) = \begin{cases} \mathrm{sgn}(\omega_i) \cdot dt, & \text{if } \|\mathbf{c}_{A_i} - \mathbf{p}_i\|_z < d_6 \wedge \|\mathbf{c}_{S_i} - \mathbf{c}_{A_i}\|_z > d_5 \\ & \vee(\|\mathbf{p}_i - \mathbf{c}_{S_i}\|_{xy} - \|\mathbf{p}_i - \mathbf{c}_{A_i}\|_{xy}) > d_7, \\ -\mathrm{sgn}(\phi_i) \cdot dt, & \text{otherwise.} \end{cases} \tag{2.10}$$

where the directional influence ω_i is given by a weighted combination:

$$\omega_i = \frac{w_1 \cdot \|\mathbf{c}_{S_i} - \mathbf{p}_i\|_z + w_2 \cdot \left(\frac{\theta_i}{\pi} - 1\right)}{w_1 + w_2}, \tag{2.11}$$

with

$$\theta_i = \mathrm{atan2}(g_x - p_{ix}, g_y - p_{iy}), \tag{2.12}$$

ensuring it remains in $[0, 2\pi]$. The weights w_1 and w_2 balance the contribution of vertical distance and directional influence.

Convergence Back to Zero If no condition for modification is met, ϕ converges back to zero:

$$\dot{\phi}_i = \begin{cases} -dt, & \phi_i > 0, \\ dt, & \phi_i < 0, \\ 0, & \phi_i = 0. \end{cases} \tag{2.13}$$

Constraint on ϕ A constraint ensures that vertical avoidance does not increase separation:

$$\phi_i = 0, \quad \text{if } |\phi_i| = \frac{\pi}{4} \wedge \|\mathbf{p}_i - \mathbf{c}_{S_i}\|_z > \|\mathbf{p}_i - \mathbf{c}_{A_i}\|_z. \tag{2.14}$$

This ensures that if vertical modification makes the drone farther from the reference, it is reset.

2.3 Simulation and Results Analysis

Describtion of simulation enviroment, used tools, Describtion of few simulation scenarios and result analysis.

Simulation Enviroment

The term 'agent' refers to the UAVs used in the simulation. For this simulation, I relied on the framework provided by [2] Each UAV obtains its global position from the ROS simulator, RViz. The positions of other UAVs were estimated using blinking LEDs mounted on each UAV, based on the method outlined in [8]. Below, I list some relevant constraints 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

Several experiments were conducted to evaluate and ensure the safe behavior of the UAVs during interactions. Different sets of UAVs were used in these experiments, with N = 5, 10, and 15, and data was collected from them for analysis. Each UAV was first flown from ground to its initial position, and once all UAVs were in their starting positions, the RBL algorithm was initiated.

For most agent interactions, experiments were conducted in both circular and spherical formations, with the circular formation being initially conducted in [1]. The circular formation was chosen as it promotes more predictable and consistent interactions between agents, as opposed to random initial positions and goal locations, which could lead to less structured or less frequent interactions. Similarly, the spherical formation was introduced to extend the experiment into 3D, providing a more comprehensive test scenario. Both structured formations help to better evaluate the performance of the RBL algorithm in environments where agents are more likely to encounter each other. The UAVs were evenly distributed along the perimeter of the circle or the surface of the sphere. The goal position was placed on the opposite side of the circle or sphere. The radius of both the circle and the sphere was set to 5 meters.

TODO table of parameters used for this experiment. Sensing radius r_s = 4.5m, update rate 10 Hz, encumbrance 0.5 m, d1 = d3 = d5 = 0.5, d2 = d4 = d6 = 1.0. For $z_{clipping}$ - min_z = 1.0 m, max_z = 10.0

In the next tables

N = 5 circular

	SR [%]	$\overline{L}$ [m]	$\bar{t}$ [s]	$\bar{t}_{\max}$ [s]	$\bar{v}$ [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 $3D_{\text{clipped}}$	100.00	20.60 ± 0.24	26.73 ± 0.47	27.39 ± 0.28	0.77 ± 0.02
RBL $3D_z$	100.00	20.97 ± 0.52	25.54 ± 0.97	26.72 ± 0.60	0.81 ± 0.03

N = 10 circular

	SR [%]	$\overline{L}$ [m]	$\bar{t}$ [s]	$\bar{t}_{\max}$ [s]	$\bar{v}$ [m/s]
RBL 2D	100.00	23.52 ± 2.10	24.52 ± 1.91	27.23 ± 0.94	0.96 ± 0.11
RBL 3D	100.00	23.32 ± 1.94	25.21 ± 2.74	27.40 ± 1.92	0.87 ± 0.14
RBL $3D_{\text{clipped}}$	100.00	23.00 ± 1.43	24.94 ± 2.42	27.54 ± 1.03	0.88 ± 0.13
RBL $3D_z$	100.00	22.79 ± 1.06	23.23 ± 2.51	26.99 ± 1.24	0.96 ± 0.10

N = 15 circular

	SR [%]	$\overline{L}$ [m]	$\bar{t}$ [s]	$\bar{t}_{\max}$ [s]	$\bar{v}$ [m/s]
RBL 2D	100.00	00.00 ± 0.00	00.00 ± 0.00	00.00 ± 0.00	0.00 ± 0.00
RBL 3D	100.00	00.00 ± 0.00	00.00 ± 0.00	00.00 ± 0.00	0.00 ± 0.00
RBL $3D_{\text{clipped}}$	100.00	00.00 ± 0.00	00.00 ± 0.00	00.00 ± 0.00	0.00 ± 0.00
RBL $3D_z$	100.00	00.00 ± 0.00	00.00 ± 0.00	00.00 ± 0.00	0.00 ± 0.00

N = 10 sphere

	SR [%]	$\overline{L}$ [m]	$\bar{t}$ [s]	$\bar{t}_{\max}$ [s]	$\bar{v}$ [m/s]
RBL 2D	100.00	00.00 ± 0.00	00.00 ± 0.00	00.00 ± 0.00	0.00 ± 0.00
RBL 3D	100.00	00.00 ± 0.00	00.00 ± 0.00	00.00 ± 0.00	0.00 ± 0.00
RBL $3D_{\text{clipped}}$	100.00	00.00 ± 0.00	00.00 ± 0.00	00.00 ± 0.00	0.00 ± 0.00
RBL $3D_z$	100.00	00.00 ± 0.00	00.00 ± 0.00	00.00 ± 0.00	0.00 ± 0.00

Practical Experiment Setup

Comparisons

2.4 Summary and Key Insights

Recap of modifications. Faced challenges and solutions applied

3 Environment Perception Using 3D LiDAR

3.1 Introduction

Motivation for using 3d lidar and challenges

3.2 3D LiDAR Sensor Model and Simulation Setup

Overview of LiDAR used. Simulation configuration

3.3 Object Detection and Approximation

Methods for extracting objects from LiDAR point clouds. Approximating detected objects with simple shapes. Handling noisy or incomplete data

3D Surface Approximation from Point Clouds ← This one seems promising. I would like to try this.

Alpha Shape: A Generalization of the Convex Hull

3.4 Integration with RBL Algorithm

Modifications to ensure safe navigation and how approximated objects influence Voronoi cells

3.5 Simulation Results

Example scenarios - in rviz with drone and also in real life - me holding a branch with leafs or something like that

3.6 Discussion

Limitations and possible improvements

3.7 Summary

4 Researching on possible enhancements of the algorithm for example using neural networks or learning-based techniques.

tune the paramaters of RBL to suit 3D

I will have to read something first, but what I found is that I could try object detection and classification using models like PointNet, PointNet++ or VoxelNet

Pointcloud denoising

Segmentation of LiDAR data. RangeNet++, SalsaNext to differentiate between terrain trees and free space

NN approaches for approximating point clouds with simple convex shapes:

CvxNet: Learnable Convex Decomposition by Deng et al. (CVPR 2020)

Label-Efficient Learning on Point Clouds using Approximate Convex Decompositions by Gadelha et al. (Arxiv 2020)

Region Segmentation via Deep Learning and Convex Optimization by Sonntag and Morgenshtern (Arxiv 2019)

Point Density-Aware Voxels for LiDAR 3D Object Detection

5 Conclusion

Summarize the achieved results. Can be similar as an abstract or an introduction, however, it should be written in past tense.

6 References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

A Appendix A