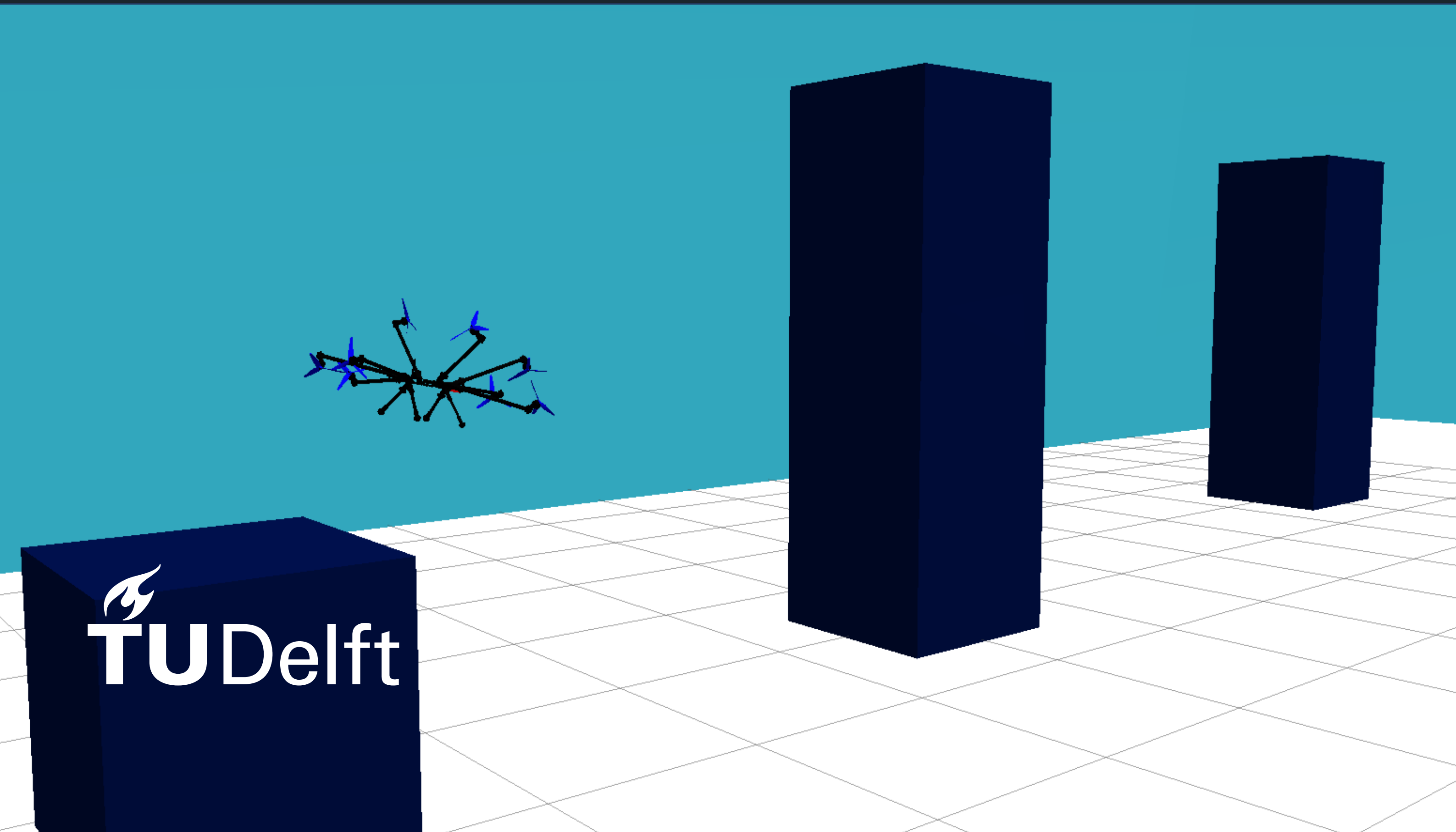
Local Path Planning and Obstacle Avoidance for an Omnicopter

A 6D-DWA Algorithm for Real-time Omnidirectional Navigation

Mikołaj Heliński



Local Path Planning and Obstacle Avoidance for an Omnicopter

A 6D-DWA Algorithm for Real-time Omnidirectional Navigation

Thesis report

by

Mikołaj Heliński

to obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on March 5, 2026 at 13:15

Faculty of Aerospace Engineering · Delft University of Technology

Preface

This thesis marks the conclusion of my efforts toward obtaining a Master of Science in Aerospace Engineering at Delft University of Technology. The thesis project was conducted between February 2025 and February 2026, as a collaboration project between TU Delft and New York University Abu Dhabi, which developed the omnicopter and provided the simulation environment.

First and foremost, I would like to thank my supervising professors, Dr. Marija Popovic and Dr. Spilios Theodoulis, for the opportunity to work on this thesis project. Their guidance facilitated right-path following and pushed me to complete this work to the best of my ability. I would also like to express my gratitude to Dr. Mahmoud Hamandi and Abdullah Ali from New York University Abu Dhabi for their assistance while working with their platform. Furthermore, I thank Professor Anthony Tzes for facilitating the collaboration.

I would also like to thank my parents and my sister for their unwavering support in every way possible. Finally, I am grateful for all the friends I made during my time at TU Delft, with a special mention to all the housemates and honorary housemates.

Mikołaj Heliński
Delft, February 2026

Summary

Autonomous UAVs are currently deployed in a rapidly expanding range of applications, where the omnidirectionality of aerial platforms vastly augments the operational capabilities. Path planning algorithms aim to facilitate autonomous, collision-free and efficient traversal of obstacle-filled environments. The increased dimensionality poses a significant computational challenge, necessitating further research on computationally-efficient algorithm implementations, facilitating platforms with limited computational capabilities. Several global and local path planning algorithms have been successfully implemented for UAV traversal of cluttered and dynamic environments. The limited research on the integration of path planning algorithms necessitates investigation into potential performance enhancement through algorithm hybridisation.

Within the framework of New York University Abu Dhabi's research project, an omnicopter platform was developed alongside a Gazebo simulation and a global path planner implementation. As the implemented global path planning RRT* algorithm is not real-time feasible, a computationally-efficient algorithm facilitating the global path following and local replanning must be integrated. This study proposes a 6D Dynamic Window Approach (6D-DWA) algorithm, a high-dimensional expansion of the 2D DWA algorithm. This algorithm determines favourable commands through sampling achievable velocity spaces, evaluating collision with the environment and scoring the trajectories based on defined metrics. Unmapped obstacle avoidance is achieved through depth-camera detection of unknown obstacles and context-aware adjustment of the 6D-DWA scoring function. To maintain real-time feasibility, the planner utilises environment voxelisation, omnicopter structure approximation and adaptive velocity sampling. A separate environment-aware evasion logic was implemented via a state machine for fast-moving obstacle evasion, compensating for the foresight limitations of the 6D-DWA planner.

Experimental results from simulation testing demonstrate that the 6D-DWA consistently performs below the critical 0.2 s loop time threshold, allowing for real-time replanning. The adaptive velocity sampling significantly outperformed random sampling, thereby limiting the required number of samples and reducing the computational load. The environment simplification enabled real-time obstacle-aware replanning. The context-aware adjustment of 6D-DWA weights facilitated prioritising unknown obstacle avoidance over path following, enhancing the achievable behaviour. The static-aware evasion strategy was successfully integrated within the local path planner.

The 6D-DWA algorithm is a feasible real-time local path planning algorithm. The desired behaviour of the omnicopter can be achieved through scoring function definition and context-aware weight alteration. The 6D-DWA metrics and weights should be defined based on the intended goals of the particular real-world applications. The algorithm's capabilities are limited by its foresight and computational performance. Further investigation into integration of different path planning algorithms can potentially mitigate the algorithm's limitations and enhance replanning capabilities.

Contents

Preface ii

Summary iii

List of Figures vii

List of Tables viii

1 Introduction 1
- 1.1 Literature Review and Research Gap 1
- 1.2 Research Objective 2
- 1.3 Methodology 2
- 1.4 Thesis Structure 2

2 Literature Review 3
- 2.1 Omnicopter Classification and Applications 3
- 2.2 Motion Planning 4
- 2.3 Global Path Planning Algorithms 6
- 2.4 Local path planning algorithms 9
- 2.5 Sensor-based Obstacle Detection and Avoidance 10
- 2.6 Summary and Knowledge Gap 11

3 System Architecture and Implementation 13
- 3.1 Omnicopter and Simulation 13
- 3.2 Local Path Planner 14
- 3.3 Obstacle Detection and Avoidance 22
- 3.4 Code Architecture 26

4 Experimental Results 29
- 4.1 Computational Performance 29
- 4.2 Velocity Sampling Evaluation 31
- 4.3 6D-DWA Weight Tuning 33
- 4.4 Path Following Evaluation 35
- 4.5 Static Obstacle Avoidance Evaluation 36
- 4.6 6D-DWA Agile Mode Weight Tuning 40
- 4.7 Evasion Evaluation 44

5 Conclusions & Recommendations 48
- 5.1 Conclusions 48
- 5.2 Recommendations 49

A Appendix A: Baseline Parameters 54

Nomenclature

List of Abbreviations

6D-DWA	Six-Dimensional Dynamic Window Approach
ANN	Artificial Neural Network
APF	Artificial Potential Field
CNN	Convolutional Neural Network
DOF	Degrees of Freedom
DRL	Deep Reinforcement Learning
DWA	Dynamic Window Approach
FOV	Field of View
GA	Genetic Algorithm
KD-Tree	k-Dimensional Tree
LIDAR	Laser Imaging Detection and Ranging
ML	Machine Learning
NYUAD	New York University Abu Dhabi
PCL	Point Cloud Library
PRM	Probabilistic Roadmap
RL	Reinforcement Learning
ROS	Robot Operating System
RQ	Research Question
RRT	Rapidly Exploring Random Tree
RRT*	Rapidly Exploring Random Tree Star
SA	Simulated Annealing
SLAM	Simultaneous Localization and Mapping
STL	Stereolithography File Format
UAV	Unmanned Aerial Vehicle
VFH	Vector Field Histogram

$(\cdot)^{\text{agile}}$	Parameter value utilised during Agile Mode operations
$(\cdot)_{\triangle}$	Discrete increment or change in a value
$(\cdot)_{\text{b}}$	Property defined relative to the body frame
$(\cdot)_{\text{curr}}$	Value associated with the current state
$(\cdot)_{\text{c}}$	Property defined relative to the camera frame
$(\cdot)_{\text{face}}$	Incentive to orient the sensors toward an obstacle
$(\cdot)_{\text{goal}}$	Value associated with the local goal
$(\cdot)_{\text{head}}$	Property related to heading or orientation alignment
$(\cdot)_{\text{limit}}$	Hardware or software boundary constraint
$(\cdot)_{\text{look}}$	Property related to the lookahead pointing objective
$(\cdot)_{\text{max}}$	Maximum allowable value
$(\cdot)_{\text{min}}$	Minimum allowable value
$(\cdot)_{\text{msg}}$	Data received via a message
$(\cdot)_{\text{path}}$	Property related to the global path or cross-track error
$(\cdot)_{\text{sim}}$	Predicted state value from internal simulation
$(\cdot)_{\text{wa}}$	Reference to waypoint-related parameters
$(\cdot)_{\text{w}}$	Property defined relative to the world frame

Constants

$(\cdot)^T$	Transpose operator
$\otimes$	Hamilton product (quaternion multiplication)
C	Individual cost term component
J	Total weighted cost value

List of Symbols

$\boldsymbol{\alpha}_{\text{max}}$	Maximum angular acceleration limits [rad/s^2]
$\boldsymbol{\omega}_{\text{b}}$	Angular velocity vector in the body frame [rad/s]
$\boldsymbol{\omega}_{\text{limit}}$	Maximum angular velocity limits [rad/s]
Δt_{step}	Internal simulation prediction step duration [s]
$\hat{\mathbf{e}}_{\text{opt}}$	Optimal evasion unit vector [-]
$\mathbf{a}_{\text{max}}$	Maximum linear acceleration limits [m/s^2]
$\mathbf{o}_i$	Body-frame offset for a collision approximation sphere [m]
$\mathbf{p}_k$	Position vector at discrete time step k [m]
$\mathbf{p}_{\text{o,w}}$	Omnicopter position in the world frame [m]
$\mathbf{q}_k$	Orientation unit quaternion at discrete time step k [-]
$\mathbf{q}_{\text{o,w}}$	Omnicopter orientation in the world frame [-]
$\mathbf{r}$	Relative position vector to a moving threat [m]
$\mathbf{R}(\mathbf{q})$	Rotation matrix representation of the quaternion $\mathbf{q}$ [-]
$\mathbf{R}_{\text{c}\to\text{b}}$	Rotation matrix from the camera frame to the body frame [-]
$\mathbf{t}_{\text{c}\to\text{b}}$	Translation vector from the camera frame to the body frame [m]
$\mathbf{v}_{\text{limit}}$	Maximum linear velocity limits [m/s]
$\mathbf{v}_{\text{rel}}$	Relative velocity vector to a moving threat [m/s]
$\mathcal{E}$	Set of candidate evasion vectors [-]
$\mathcal{O}_{\text{obs}}$	Set of points representing the mapped static environment [-]
$\mathcal{O}_{\text{unknown}}$	Set of points representing unmapped obstacles [-]
σ	Standard deviation for focused velocity search [-]
b_r	Boundary-Search sampling ratio [-]
d_{miss}	Shortest predicted distance to a moving obstacle [m]
d_{stuck}	Displacement threshold for 'Stuck' state detection [m]
e_r	Exploration sampling ratio [-]
f_r	Focused-Search sampling ratio [-]
n_{samples}	Number of velocity samples per planning cycle [-]
N_{wa}	Waypoint index offset for local goal selection [-]
t_p	Time horizon for trajectory prediction [s]
t_{cpa}	Calculated time to the closest point of approach [s]
t_{stuck}	Time duration threshold for 'Stuck' state activation [s]
t_{tti}	Time-to-impact threshold for evasion triggering [s]
v_{evade}	Constant velocity magnitude used during evasion [m/s]
w_{clear}	Weight assigned to the obstacle clearance cost [-]
w_{face}	Weight assigned to the obstacle facing cost [-]
w_{goal}	Weight assigned to the goal distance cost [-]
w_{head}	Weight assigned to the orientation error cost [-]
w_{look}	Weight assigned to the lookahead alignment cost [-]
w_{path}	Weight assigned to the cross-track error cost [-]

List of Figures

2.1 Examples of environment representation methods . . . 5
2.2 Further examples of environment representation methods . . . 5
2.3 Examples of applications of the potential field method and RRT* algorithm . . . 7
2.4 Visualisation of the fuzzy inference system . . . 8
2.5 The basic logic of the different metaheuristic algorithms . . . 8
2.6 Visualisations of the DWA and VFH algorithms . . . 10
2.7 Overview of the framework for the UAV obstacle detection and avoidance process[5] . . . 10
2.8 Breakdown of approaches assessed in research studies . . . 12

3.1 The NYUAD-developed omnicopter and the model within the simulation . . . 14
3.2 Visualisation of the 6-DOF Dynamic Window . . . 15
3.3 Visualisation of the effect of different search ratios in Adaptive Sampling . . . 16
3.4 Visualisation of the 10-sphere approximation encompassing the omnicopter's structure . . . 17
3.5 Visualisation of the voxelisation method . . . 18
3.6 Visualisation of the worst-case voxel centroid spread . . . 18
3.7 Visualisation of the sphere-surface interaction . . . 19
3.8 RViz view of the generated depth camera point cloud . . . 23
3.9 RViz view of the identified unknown obstacle points . . . 24
3.10 Flowchart of the finite state machine . . . 28

4.1 Average computation time vs a number of obstacles for varying number of samples . . . 30
4.2 Average computation time vs a number of obstacles for varying number of samples with regression lines . . . 30
4.3 Averaged path-following performance for varying number of samples . . . 31
4.4 Average cross-track and orientation performance of the Adaptive Sampling without Boundary-Search, Adaptive Sampling with Boundary-Search, and random sampling configurations . . . 32
4.5 6D-DWA weight tuning run-to-run scoring variations . . . 34
4.6 Averaged best performing weight configurations for each scoring configuration . . . 35
4.7 Resulting trajectories for a curved global path . . . 36
4.8 The side view of trajectories resultant from testing scenario 5.1 . . . 37
4.9 The side view of trajectories resultant from testing scenario 5.2 . . . 38
4.10 Trajectories resulting from testing scenario 5.3 . . . 38
4.11 Altitude and roll angle variation during gap traversal . . . 39
4.12 Proximity to obstacles during gap traversal for w_{head} of 15 . . . 39
4.13 Gazebo visualisation of the Agile Mode testing scenario . . . 40
4.14 The impact of unknown obstacle clearance weight $w_{\text{clear}}^{\text{agile}}$ on trajectory with an off-centre placed unknown obstacle . . . 41
4.15 The effect of unknown obstacle clearance weight $w_{\text{clear}}^{\text{agile}}$ on the trajectory with a centrally placed unknown obstacle . . . 41
4.16 The trajectories for $w_{\text{clear}}^{\text{agile}}$ of 1.0 with an off-centre placed unknown obstacle . . . 42
4.17 The trajectories for $w_{\text{clear}}^{\text{agile}}$ of 1.0 with a centred unknown obstacle . . . 43
4.18 The impact of unknown obstacle facing weight $w_{\text{face}}^{\text{agile}}$ on the trajectory with an off-centre placed unknown obstacle . . . 43
4.19 Evasion caused by a threat offset by 0.5 m . . . 44
4.20 Evasion manoeuvre in the presence of static obstacles . . . 45
4.21 Distance from the simulated threat path vs time for different body approximations . . . 46
4.22 3D trajectories for the 6D-DWA path following with an induced evasion manoeuvre . . . 46

List of Tables

4.1 Computational evaluation testing scenario parameters 29
4.2 Velocity sampling evaluation testing scenarios 32
4.3 6D-DWA weight tuning weight configurations 33
4.4 Scoring weight configurations for the sensitivity analysis 33
4.5 6D-DWA Baseline weights 35
4.6 Path following evaluation testing scenarios 35
4.7 Testing scenarios for the evaluation of static obstacle avoidance 37
4.8 Agile Mode testing parameters 40
4.9 Agile Mode with $w_{\text{clear}}^{\text{agile}}$ of 1.0 testing parameters 42
4.10 Testing scenarios for moving obstacle evasion evaluation. 44

A.1 Baseline parameters of the 6D-DWA Planner 54

1 Introduction

The following chapter outlines the background into the current state of path planning algorithms and presents the identified research gap. The research objective and research questions are defined, followed by a description of the methodology and the outline of the thesis structure.

1.1. Literature Review and Research Gap

Recent years have witnessed significant advancements in the area of autonomous multicopter platforms [1]. The ability of multicopter UAVs to perform precise hovering and operate at low horizontal velocities makes them particularly useful in applications such as search and rescue, delivery or surveillance [2]. Octocopters possess the ability to achieve omnidirectionality and full actuation, allowing for the full in-flight control of the attitude [3]. Omnidirectionality allows for a wider range of applications, such as scenarios requiring high manoeuvrability or aerial manipulation [4]. The advantages of utilising the omnidirectional platforms, as well as the benefits of system autonomy, warrant further research and development of efficient methods for autonomous traversal of cluttered spaces for omnidirectional platforms.

To autonomously traverse an obstacle-filled space, the omnicopter has to be able to generate and follow a path in the free space. Path planning algorithms generate an optimised collision-free path in the defined space. The algorithms are categorised into classical, soft-computing and hybrid [1]. Global path planning algorithms are used to generate an optimised path from the start point to the desired end point. Local path algorithms are often integrated with global path planning algorithms for local path optimisation and real-time replanning [5]. Real-life environments are often dynamic, requiring the UAV to be able to respond to the changes in the local environment. In order to avoid collision with dynamic obstacles the omnicopter has to be able to perceive the environment, detect an obstacle and perform an avoidance manoeuvre [6]. Obstacle perception is accomplished through integrating sensor data.

There are many commonly used path planning algorithms for 3D space traversal. These approaches, however, exhibit advantages and disadvantages, necessitating a trade-off on complexity and performance [7]. Dynamic environment applications are significantly more challenging for the current algorithms [8]. Obstacle detection and avoidance methods are commonly used but are poorly integrated with the path planning [5]. The performance limitations of an individual algorithm may be mitigated by the integration with another algorithm. However, the increased complexity entails greater required computational effort. Real-life implementation poses constraints on the possible methods. The computational limitations and the need for real-time execution limit the allowable complexity of the algorithms. A computationally efficient hybrid path planning algorithm integrated with obstacle detection and collision avoidance could improve the performance and applicability of omnidirectional platforms in cluttered and dynamic spaces. The omnidirectionality of the multicopter can be utilised by incorporating the attitude control in the local path planning and obstacle avoidance strategies.

The literature review identified a clear research gap. There is a need for further research on hybrid computationally-efficient path planning algorithms. The trade-off between the algorithm's capabilities and computational complexity necessitates investigation into possible computationally efficient path planning algorithms that facilitate autonomous traversal of dynamic environments.

1.2. Research Objective

The research objective and research questions investigated in this thesis project are defined based on the identified research gap.

Research Objective

The primary objective of this research project is to develop an autonomous algorithm that enables an omnicopter to efficiently navigate and traverse a dynamic and cluttered environment.

Throughout this thesis, the term *dynamic environment* refers to any environment containing obstacles not present in the initial STL-defined map. In order to achieve the research objective two research questions are defined. Research Question 1 addresses the computational feasibility of the algorithm, as well as the ability to perform replanning in dynamic environments.

Research Question 1

How can a computationally efficient online local path planning algorithm be developed to facilitate real-time 6-DOF navigation for an omnicopter in a dynamic environment?

As the DWA algorithm is of limited foresight, the replanning capabilities in dynamic environments are limited. The focus on the mitigation of these limitations through sensor integration is highlighted by Research Question 2.

Research Question 2

How can a real-time obstacle avoidance strategy using sensor input be integrated within a local planner to mitigate the foresight limitations of the 6D-DWA?

The thesis project aims to answer the defined research questions.

1.3. Methodology

The local path planner aims to enhance the existing framework developed by the New York University Abu Dhabi [9]. The local path planner must be integrated with the global path planner, allowing for local replanning and traversal of dynamic environments. A Dynamic Window Approach ([10]) based 6D-DWA algorithm was engineered. The evaluation of the local path planning algorithm was performed in the Gazebo simulation environment[11]. The thesis aims to evaluate the computational feasibility of the implemented algorithm, the ability to avoid static collision and dynamic obstacle avoidance. Additionally, the integration of a moving obstacle evasion strategy within the local path planner is investigated.

1.4. Thesis Structure

The thesis is structured as follows. The literature review is presented in Chapter 2. The chapter contextualises the research, presents the currently used path planning algorithms and identifies the research gap. The architecture of the implemented local path planner is presented in Chapter 3. Firstly, the implementation of the 6D-DWA environment-aware algorithm is discussed. Processing of the depth-camera sensor data, as well as the unmapped static obstacle avoidance strategy is then described. Additionally, the chapter outlines the code architecture and the state machine, implemented to aid state transition and act as a robustness layer. The implemented planner is evaluated in simulation testing, as presented in Chapter 4. The computational performance of the core 6D-DWA planner is first assessed. The effect of the implemented adaptive velocity sampling is subsequently evaluated. The core 6D-DWA weight tuning is performed, followed by the path following performance evaluation. The implemented static and dynamic obstacle avoidance methods, as well as the static-aware evasion strategy are then assessed. Lastly, the conclusions and recommendations for further research are discussed in Chapter 5.

2 Literature Review

The literature review chapter aims to provide an extensive overview and synthesis of state-of-the-art solutions in path planning and obstacle detection and avoidance methods relevant to omnicopter UAVs. The topics addressed by the literature are evaluated and discussed in terms of relevance to the thesis project. The literature review establishes the necessary background and focuses on identifying the relevant research gaps.

Firstly, the omnicopter classification and current applications are briefly addressed in Section 2.1. The path planning problem and environment representation are then explained in Section 2.2. The global and local path planning algorithms are presented in Section 2.3 and Section 2.4 respectively, followed by the evaluation of currently used obstacle detection and avoidance methods in Section 2.5.

2.1. Omnicopter Classification and Applications

The flight capabilities of multicopters depend on the level of actuation. The level of actuation is quantified by the rank of the mapping matrix, a mapping of the allowable propeller-thrust space to the allowable aerodynamic-control wrench space[3]. Under-actuated systems have more possible degrees of freedom (DoF) than actuated degrees of freedom. The fully-actuated multicopters have exactly six actuated degrees of freedom, the number of possible DoF in 3D space. The over-actuated systems are fully actuated with redundancy. The tilt-rotor designs can convert between the actuation classes, as actuation is configuration-dependent. A multicopter is considered omnidirectional if the thrust is sufficient to counteract gravity in any direction[3].

Multicopter UAVs have a wide range of applications ranging from combat, surveillance and intelligence collection systems [2]. Multicopter UAV solutions are beneficial for numerous civilian contexts, because of their high manoeuvrability, low cost, and precise hovering capabilities. Multicopter UAVs have been evaluated in various domains such as medical emergency response applications (e.g. Everdrone [12]), surveillance [13], delivery (e.g. Amazon Prime Air[14]), wildlife monitoring (e.g. Poaching Tracking [15]), agriculture applications (e.g. [16]) as well as search and rescue missions (e.g. [17]). The applications benefit from the ease of deployment, as well as the modularity of the available sensors.

The scope of applications for multicopters significantly increases for fully actuated platforms. Omnidirectional multicopters are able to move smoothly with high agility, which is crucial in traversing cluttered environments. The omnidirectional multicopter platforms are highly beneficial for search and rescue missions in hard-to-reach areas [18]. Full actuation allows these platforms to function as aerial manipulators, executing complex tasks such as grasping, transport, and object assembly [4]. The aerial manipulation applications are highly beneficial in terms of mitigating human safety risks. The full actuation has additional potential in collaborative manipulation tasks for multi-robot operations, performing tasks such as collaborative object manipulation[2].

While the fully-actuated platforms have a number of benefits and potential applications, they are significantly more challenging in implementation due to their novelty and complexity [19]. The expanded manoeuvrability of these platforms significantly complicates the motion planning problem, as the search space must account for the full 6D pose. The challenges associated with higher energy consumption and the increased complexity of control algorithms warrant future research.

2.2. Motion Planning

Multicopter UAVs are highly advantageous in applications requiring manoeuvring through cluttered environments. The multicopters' ability to hover and rapidly alter the flight direction facilitates superior manoeuvrability compared to fixed-wing and hybrid UAV platforms. The shift from remotely operated multicopter UAVs toward autonomous systems necessitated the development of robust and efficient motion planning algorithms. Autonomous traversal of an obstacle-filled environment requires the implementation of both path-planning algorithms and obstacle avoidance methods.

2.2.1. Path Planning Problem Description

Path planning is the generation of a feasible collision-free route optimised for a defined metric. A path is optimised based on the intended goal, with regard to a defined metric such as flight time, distance or energy consumption [1]. Autonomous multicopters traversing a cluttered environment must determine the optimal path to the end-goal point and diverge from the path when needed. The path planning problem thus comprises two distinct aspects, path generation and obstacle detection and avoidance.

Path generation entails two distinct domains, global path planning and local path planning. The global path planning algorithms are applied to predefined and mapped environments. The global path is generated offline, based on a pre-traversal environment model, resulting in potential infeasibility of paths in dynamic environments. Local path planning and obstacle avoidance are performed online and allow for the traversal of dynamic environments. Path planning for UAVs faces the challenge of dimensionality. The path planning needs to account for the 6D search space, as well as the path-planning time and time horizon [3]. The increased dimensionality significantly increases the computational complexity, imposing constraints on algorithms intended for real-time execution.

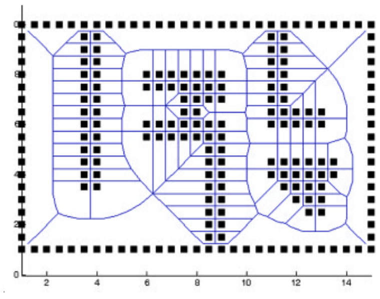
The performance of a path planning algorithm is commonly benchmarked by two criteria, the completeness and optimality. The completeness criterion ensures that, given a feasible path is possible, that path is identified by the algorithm. The criterion guarantees that the algorithm finds a solution for any environment configuration. The completeness criterion is categorised into probabilistic completeness and resolution completeness. The probabilistic completeness states that, given infinite time and resources, an existing solution will always be discovered. The resolution completeness guarantees the discovery of an existing solution in finite time. The second criterion is the optimality criterion. Path generation is optimised for a desired metric, such as distance, smoothness, or completion time. The desired metric is defined in terms of constraint or cost functions, allowing for the optimisation and evaluation of the path. A system can be both single-objective, as well as aiming to optimise multiple objectives simultaneously [1]. The evaluation of a path planning algorithm implementation depends on the definition of a desired behaviour and the determined performance requirements.

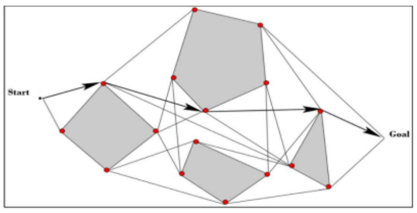
2.2.2. Environment representation

Effective path planning requires accurate environmental modelling to identify collision-free spaces. Environment simplification can minimise the required computational effort and aid the feasibility of real-time algorithms. The appropriate environment representation must be selected to balance accuracy and computational complexity. The following sections evaluate common world representation methods.

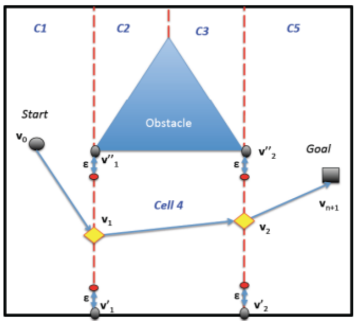
The Voronoi diagram is a mathematical graph-based method of partitioning an object-filled space. This representation is created from points in space equidistant from two or more object features. The space is partitioned into regions with one object feature each, for any point in a specific region this region's feature is always the closest object. The Voronoi diagram representation is presented in Figure 2.1. While commonly applied to path planning problems, the Voronoi diagram representation performs inefficiently for applications beyond three-dimensional [20]. The visibility graph method constructs one-dimensional lines connecting the feature of an object to another 'visible' feature. The features are the vertices of polygonal representation of the object. This method is presented in Figure 2.1. The generalised visibility graph method is an extension of the method, allowing the objects to be defined as more complex generalised polygons. The A* algorithm, further discussed in Section 2.3, is commonly applied to visibility graphs, although primarily in 2D cases [20].

The cell decomposition method partitions the free space into convex polygon cells. The partitioning comprises the same shape, such as triangles or trapezoids, and are interconnected. The cell shape must facilitate accurate environment representation. A connectivity graph is then constructed to represent the connections between the cells. The method poses a complex task of defining the optimal crossing points between the adjacent cells based on the created graph [21]. This representation can be used for both 2D and 3D environments [22]. The cell decomposition representation is visualised in Figure 2.1.


(a) Voronoi diagram [20]



(b) Visibility graph [1]



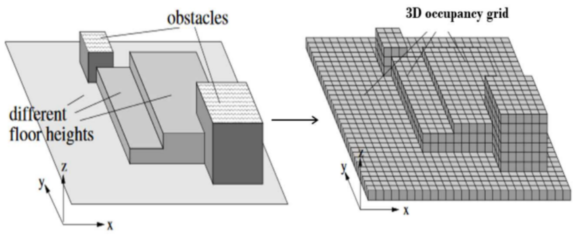
(c) Cell decomposition [21]

Figure 2.1: Examples of environment representation methods

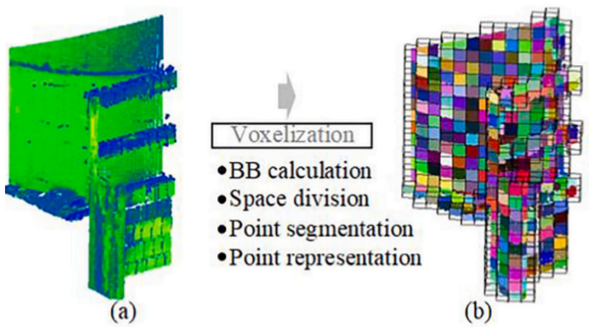
The occupancy grid map method segments the multidimensional space into cells. Each cell is assigned a value determining whether the cell is considered occupied or a part of the free space. The strength of the method is both the ease of visualisation and implementation, as well as the ability to implement stochastic reasoning for handling uncertainty in cell occupation statuses. The cells can store the probabilistic estimate of the state, allowing for mitigating allocation uncertainty and improved path planning performance[23]. The occupancy grid map is visualised in Figure 2.2.

The Point Cloud is a 3D environment representation method. A point cloud is a generated set of points in a three-dimensional space. The 3D object geometry is created by combining a large number of individual points in a common system frame. The density of the points determines the level of detail of the 3D object. These points can be derived from raw sensor data, typically utilising LiDAR or vision-based sensing to capture environmental geometry. The point cloud representation is highly advantageous due to its detailed environment description [24]. High point density increases the computational complexity for environment processing.

The voxelisation method creates a 3D representation using cuboid cells. The voxel grid representation is a discrete and resolution-flexible solution, bearing resemblance to the occupancy grid map. Voxelisation is a process of converting a point cloud into a voxel grid. The voxelisation requires the calculation of a bounding box of the point cloud, partitioning the space into certain-sized cuboids, point cloud segmentation and voxel representation of the segmented point sets. The voxel grid method approximates the point cloud lowering the accuracy but significantly decreasing the computational effort [24]. The accuracy of the voxel grid depends on the selected voxel dimensions. The voxelisation of a point cloud is visualised in Figure 2.2.



(a) Occupancy Grid [25]



(b) Voxelisation of a point cloud [24]

Figure 2.2: Further examples of environment representation methods

The performance of the path planning algorithms depends on the space definition, especially in cluttered and complex environments. Notably, the simplification of the object geometry is commonly used in path planning to reduce the complexity of the problem. The environment simplification can necessitate the inclusion of buffer space around the obstacles to account for the used approximation [1].

2.3. Global Path Planning Algorithms

There are several algorithms currently applicable for the task of path generation in 3D environments. The path planning methods will be classified as either classical, soft-computing or hybrid [1]. The currently used methods are presented in the following sections.

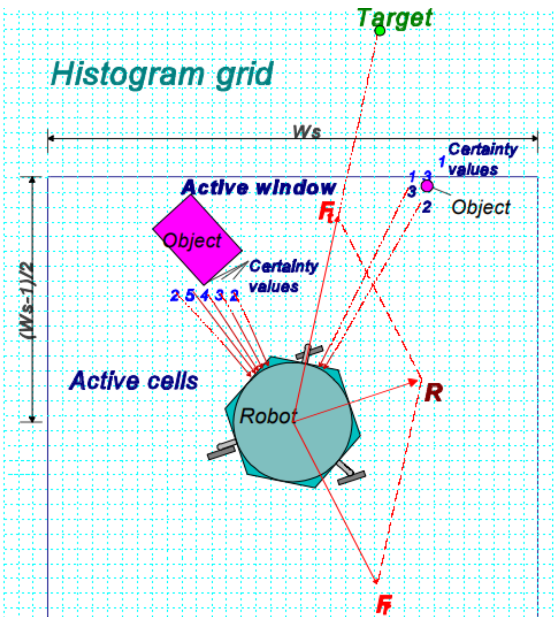
Classical methods

The classical methods are comparatively computationally efficient, the performance, however, declines with the scale of the evaluated space. The complexity of 6D search spaces restricts the applicability of several classical planning methods for multicopter systems [1].

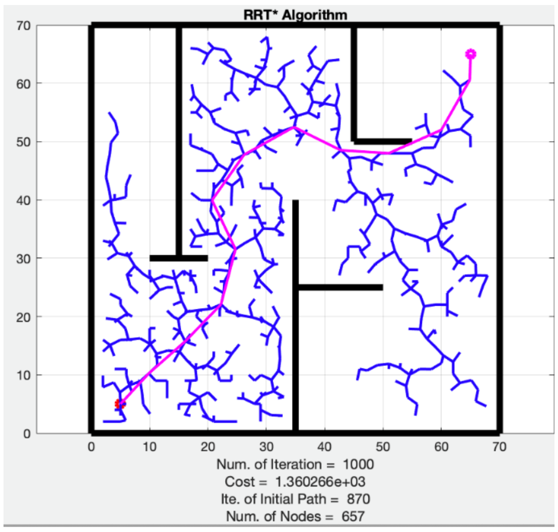
Dijkstra's algorithm is a graph-search algorithm used for UAV applications. The graph-search methods generate paths within a created graph, connecting the nodes via the edge lines [26]. Dijkstra's algorithm always explores the closest unexplored node. The algorithm is computationally demanding for high complexity problems [1]. The A* algorithm is a graph-search method based on Dijkstra's algorithm and best-first search algorithms. The A* algorithm explores nodes based on the distance from the start and the heuristic distance to the end-node [27]. The greedy best-first search can be applied, resulting in the generation of a suboptimal but feasible path. The A* is advantageous, as it prioritises reaching the end node over exploring all nodes. Notably, there are multiple variations of the A* algorithm, such as D*, D*-Lite, field D*, FSA*, or GAA* [1].

In the artificial potential field algorithm, each point in space is assigned a value based on the artificial potential function. The agent is treated as a particle reacting to the attractive forces of the end goal position and the repulsive forces from the obstacles [28]. The vehicle traverses the space, navigating towards the lowest potential at the endpoint. The space is tessellated into cells, but only the centre points of each cell are considered during path planning. The method is prone to inducing oscillations near obstacles and not allowing the agent between two closely-placed objects [20]. For complex spaces the algorithm can fail to escape local minima. A potential solution is the inclusion of artificially added obstacles in the areas of expected local minima. Obstacle avoidance can be accomplished by integrating the sensor inputs in the Vector Force Field (VFF) or the Vector Field Histogram (VFH) methods [29]. The concept is visualised in Figure 2.3.

The sampling-based methods analyse the random sampling points of the free space. These methods do not require the complete exploration of a configuration space and are computationally efficient even in high-dimensional space, although there is no guarantee of the optimality of the generated path. Probabilistic Roadmap (PRM) is a sampling-based path planning method. The probabilistic roadmap method firstly creates a connectivity graph, which is then used for the path generation. The PRM method generates random collision-free sampling point placements in the configuration space and applies a simple local planner to create connections between the points. The generated connections are then evaluated in terms of collision. The created connections form a graph that can be solved using graph-search methods [30]. Rapidly exploring random tree method (RRT) is another sampling-based method commonly used in path planning. The algorithm incrementally builds a tree by creating paths from the root node to randomly sampled points, and only the collision-free connections and nodes are added. The algorithm has been proven successful in autonomous vehicle applications in complex environments [31]. The RRT has been extended in several algorithms in an attempt to improve the issue of generating suboptimal solutions. The extension algorithm RRT*, shown in Figure 2.3, improves the quality of the generated path over time. As more nodes are added, the solution asymptotically converges towards the optimal path [32].



(a) The Virtual Force Field concept [33]



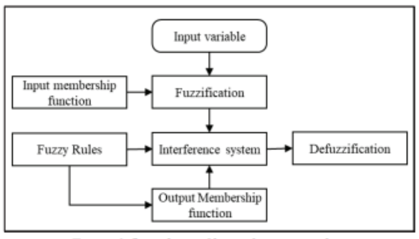
(b) Rapidly exploring random tree RRT* algorithm visualisation [34]

Figure 2.3: Examples of applications of the potential field method and RRT* algorithm

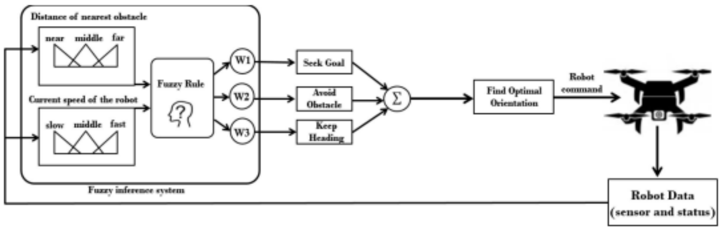
Soft-computing methods
The term 'soft-computing' encompasses machine learning, fuzzy approaches and metaheuristic algorithms. The methods are categorised together due to their shared characteristics in uncertainty handling, mimicking human reasoning and decision-making [1].

The machine learning methods comprise neural networks, supervised learning, unsupervised learning and reinforcement learning. Neural networks mimic the biological neural networks, creating connections between processing units. The Artificial Neural Network (ANN) application allows for fast convergence and reduced computational complexity due to parallel processing. The Deep Neural Networks (DNN) are deep architectures comprising of multiple layers ANNs and are able to handle more complex data patterns, allowing for applications in more complex scenarios [35]. The neural networks are suitable for dynamic environments, can handle multi-objective path planning and exhibit fast convergence but have a high computational complexity for complex environments. Both the ANNs and DNNs have been applied successfully in UAV path generation and collision avoidance [8]. In reinforcement learning (RL) the UAV learns by interacting with the environment, receiving feedback and rewards and updating its strategies accordingly. The model effectively learns through trial and error. Reinforcement learning has been applied to the UAV path planning problem in applications focusing on improving path optimization, collision avoidance, real-time decision-making, and energy efficiency. RL-based methods have proven efficient in handling manoeuvrability but are computationally extensive and yield suboptimal results [8]. In supervised learning, the UAV learns based on labelled training input, allowing the model to recreate learned patterns. In unsupervised learning, the algorithm identifies patterns without supervision, meaning that the labelled training data is not required [35].

The fuzzy inference system utilises a non-binary truth/false logic. A decision is made based on the degree of truth or falsity [36]. A 'fuzzy' variable is non-binary, accounting for the uncertainty. A solution is determined based on applying a defined if-then ruleset to the 'fuzzy' input. Relevant terminology includes *fuzzification*, a process of converting the input variable by the membership function, *inference* outputting based on the fuzzy rules, *crisp value* a definite numerical value, and *defuzzification*, the conversion of fuzzy output to crisp values. The basic fuzzy logic is presented in Figure 2.4. The fuzzy inference system is particularly advantageous in dealing with undefined environments, showing potential in obstacle avoidance in navigation. The requirement for a predefined ruleset can pose a significant challenge in high-dimensional scenarios [1].



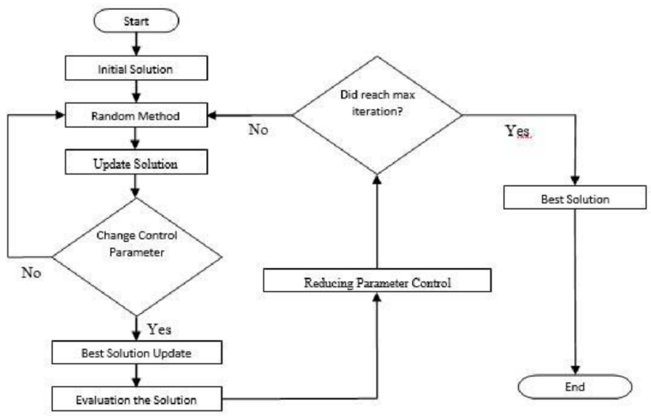
(a) A flowchart of basic fuzzy logic control system [36]



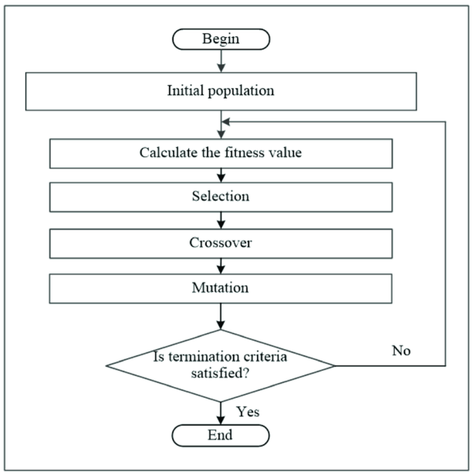
(b) Example of fuzzy inference system for path planning [1][37]

Figure 2.4: Visualisation of the fuzzy inference system

The metaheuristic algorithms are a group of algorithms used for optimisation problems that are challenging for the exact methods. In navigation tasks, metaheuristic algorithms input the mapped environment and generate initially random solutions. The paths are evaluated until a path is found to reach the goals and constraints defined in the objective function. Simulated Annealing (SA) is a common metaheuristic algorithm. The SA algorithm performs random adjustments to randomly generated paths, upon which each variation is evaluated based on a probability function. The design of the probability function aims to ensure extensive exploration of the problem space and allow for escaping the local minima [38]. The UAV applications of the algorithm have been extensively proven to be feasible and advantageous, allowing for optimisation in terms of multiple objectives [1]. The simplified logic of the algorithm is presented in Figure 2.5. The Genetic Algorithm (GA) is a metaheuristic algorithm that emulates the survival of the fittest mechanism as seen in nature. A population of solutions is generated and evaluated using an objective function. The best-performing solutions undergo either individual mutation or part combination, ultimately resulting in a new improved set of solutions [39]. The process has been shown to be applicable to UAV applications[1]. The principle of the algorithm is visualised in Figure 2.5. The Differential Evolution algorithm is an evolutionary algorithm. The algorithm, similarly to the GA algorithm, modifies the best-scoring solutions. As opposed to the random changes to the solutions in the GA algorithm, the alterations in the DE algorithm are based on other solutions [40]. The DE algorithm is applicable to UAV applications in both defined and undefined environments. The DE algorithm is advantageous due to the ease of implementation, fast and efficient computation and the quality of the proposed solution[1]. Other metaheuristic algorithms can be applied to the problem of UAV path planning, such as Teaching-Learning-Based Optimization, Grey Wolf Optimization or Fruit-fly optimisation [7].



(a) Simulated Annealing algorithm simplified [41]


Begin
Initial population
Calculate the fitness value
Selection
Crossover
Mutation
Is termination criteria satisfied?
No
Yes
End

(b) Genetic algorithm [42]

Figure 2.5: The basic logic of the different metaheuristic algorithms

Hybrid methods
The hybrid methods combine multiple algorithms to achieve improved performance, unattainable for the individual algorithms. The fusion of algorithms with different logic and characteristics can mitigate the limitations of the algorithms while benefitting from the advantageous characteristics. Numerous recently developed hybrid algorithms are applicable for the UAV path planning problem. The hybridised methods tend to outperform the base algorithms in terms of path length, energy consumption or computational time [8]. The soft-computing methods are integrated with both the classical methods and other soft-computing methods. Metaheuristic algorithms have been successfully integrated with both classical methods (Multi-frequency Vibrational Genetic Algorithm and Hybrid Genetic Algorithm), machine learning methods (Reinforcement Learning-based Grey Wolf Optimizer) and other metaheuristic algorithms (Ant Colony Optimization-DE algorithm) [8][7]. Hybrid solutions based on classical methods can be developed as well, such as the fused artificial potential field method and RRT-Connect algorithm [1]. The hybridisation of the path planning algorithms offers significant potential for optimisation of the path planning in dynamic environments [8].

Global path planning algorithms are often not feasible for real-time local replanning to the high computation time. The algorithms applicable for online replanning are discussed in Section 2.4.

2.4. Local path planning algorithms

Local path planning algorithms generate a path near the current location, allowing for the traversal of dynamic environments and real-time decision making. The local path planning algorithms are implemented alongside the global path planning algorithms. The goal of the local path planning algorithms is to react to the sensed changes in the environment and avoid potential collisions. The computational performance is thus a priority over the optimality [5]. The commonly used local path planning algorithms are presented below.

The Rapidly-Exploring Random Tree (RRT) algorithm discussed in Section 2.3 is also applicable for local path planning. To improve the performance of the RRT algorithm, RRT* and RRT-Connect extensions have been developed. The improved algorithm RRT* improves the performance by analysing nearby nodes for lower cost and rewiring the tree at each step. The RRT-Connect improves the computation time by generating two trees from the start and end nodes, resulting in faster convergence. The performance can be further improved by implementing fuzzy control for random node generation. [5] The algorithm can be applied to cluttered and dynamic environments. Real-time replanning is challenging due to the algorithm's complexity.

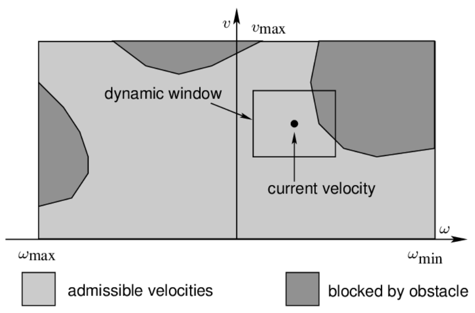
The artificial potential field method, as discussed in Section 2.3, utilises artificial forces that repel the drone from obstacles and attract it toward the desired end point. The algorithm can be further improved by optimising the repulsive force function definitions. The algorithm can encounter difficulties in complex environments with high obstacle density. The method can be improved in terms of avoiding oscillations and escaping local minima by incorporating potential field guide points at the cost of increased algorithm complexity. The algorithm's capability may be limited in highly dynamic environments [5].

The dynamic window approach algorithm (DWA) evaluates discrete sampled velocities within the allowable range. The feasible velocity commands and their effect on immediate movement are evaluated based on the defined cost function, allowing for flexibility in the desired behaviour definition. The direct evaluation of the velocity control inputs is highly beneficial for dynamic environment applications. The algorithm is advantageous in terms of efficiency and real-time capability. The dynamic window approach has been successfully integrated with the A* algorithm, resulting in promising smooth path generation [43]. The computational effort significantly increases with the dimensionality of the velocity search space. The dynamic window approach velocity search space is visualised in Figure 2.6. There has been a successful DWA implementation capable of real-time replanning in 3D environments [44].

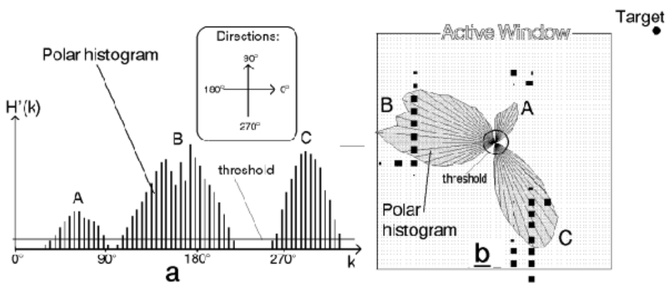
Fuzzy logic, as discussed in Section 2.3, is a decision-making framework that utilises a non-binary truth/false logic. The fuzzy logic allows for navigation and decision-making in uncertain and dynamic environments. The fuzzy logic itself is unsuitable for real-time dynamic obstacle avoidance. To improve local path planning application performance, the method has been integrated with other path planning algorithms, such as the genetic algorithm or ant colony optimisation algorithm. The hybrid approaches yield improved performance at the cost of high computational effort, which can cause difficulties in real-time applications [43].

The Vector Field Histogram (VFH) method is a local path planning method primarily used for obstacle avoidance. The algorithm represents the environment with a vector field histogram generated from the sensor data. The histogram, as presented in Figure 2.6, represents the density of obstacles in the given direction. The decision on the direction of the movement is based on the generated histogram, the heading direction of the UAV and the direction toward the end point. The VFH* method, an extension to the algorithm, has been successfully applied for local path planning [45]. The algorithm is computationally efficient, applicable for dynamic environments and real-time applications but can exhibit difficulties in cluttered environments.

As detailed in Section 2.3, the neural networks are a machine learning method. In local path planning algorithms, neural networks offer high optimisation capabilities, adaptability, and effectiveness in complex environments [5]. For example, a residual convolutional neural network (RCNN) has been successfully applied to the local path planning problem, resulting in good performance in terms of path accuracy and collision avoidance [46]. The computational challenges are relevant in real-time applications. The reinforcement learning is a learning method utilising trial-and-error for cumulative reward optimisation. The deep reinforcement learning (DRL) frameworks have been researched for UAV applications to enhance the clarity and efficiency of decision-making[5]. The multi-layer reinforcement learning method has been applied for UAV traversal of a complex environment. The complexity of the method impedes its applicability in real-time applications [47]. The reinforcement learning methods allow for path optimisation in large-scale and complex environments but require high processing effort.



(a) The search space for the dynamic window approach [48]



(b) The Vector Field Histogram [49]

Figure 2.6: Visualisations of the DWA and VFH algorithms

2.5. Sensor-based Obstacle Detection and Avoidance

Autonomous systems require measures of collision avoidance to facilitate the traversal of dynamic environments. The environment perception, obstacle identification and collision avoidance approaches are addressed in this section.

The UAV must possess the capability to perform a diversion based on the detection of an obstacle on the UAV's course. The obstacle avoidance control algorithms utilise the sensor inputs and a combination of rule-based and optimisation-based control methods [5]. The generalised flow of the obstacle decision and avoidance process is presented in Figure 2.7.

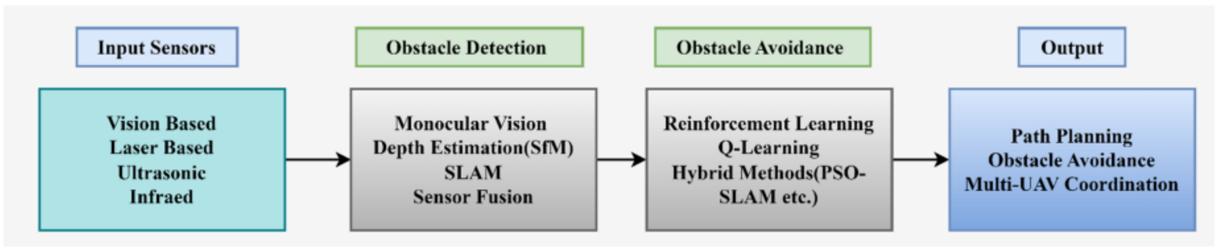


Figure 2.7: Overview of the framework for the UAV obstacle detection and avoidance process[5]

The detection of obstacles within the UAV's immediate environment is crucial for collision-free dynamic space traversal. Environment perception utilises the available sensor data. The most commonly used sensing methods for multicopter UAVs are addressed in Figure 2.7. The perception characteristics depend on the type, number and configuration of the available sensors. Single sensor configurations limit the data available to the planner to a singular data type and, in cases such as fixed monocular vision or depth-camera configurations, limited Field of View (FOV). Remarkably, omnicopter UAVs have the ability to decouple attitude from lateral movement, thus possibly maintaining obstacle pointing orientations regardless of the movement direction. The omnidirectionality enables the mitigation of the FOV limitations.

Notably, multi-sensor integration can mitigate the perceptual limitations of individual sensors. There are multiple successful sensor integrations of different sensor types, such as stereo vision and IMUs [50], stereo vision and radar [51] or vision and laser fusion [52]. A multi-sensor strategy combining ultrasonic, infrared, and laser data was proposed to improve the performance in dynamic environments [53]. Multi-sensor solutions can be unfeasible in systems with weight and computation limitations [5]. The obstacle state estimation can additionally be augmented by estimation algorithms, such as the Kalman filtering [54].

Local path planning algorithms require preprocessed sensing data, aligned with the specific environment representation. The classical local path planning algorithms addressed in Section 2.4 utilise the preprocessed obstacle information in trajectory generation. Such solutions can facilitate real-time obstacle detection and avoidance, at the cost of requiring intensive raw sensor data processing [5]. The Simultaneous Location and Mapping (SLAM) based methods enable obstacle avoidance by providing an estimated environment map based on the sensing data [55]. SLAM-based methods have been successfully integrated in multi-sensor and hybrid algorithm applications[5]. The Visual-SLAM implementations can become unfeasible for limited-computation platforms [55].

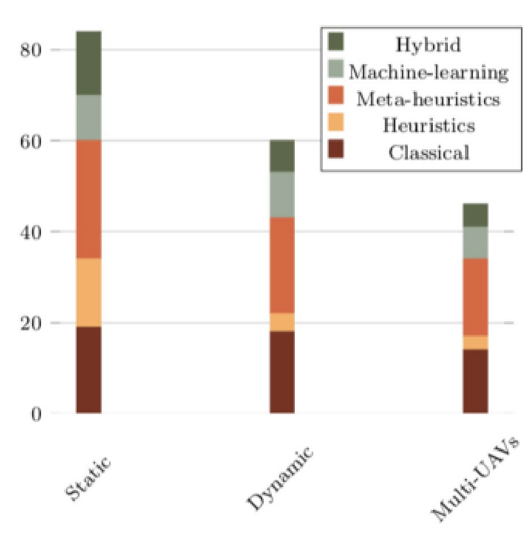
Soft-computing methods can facilitate the integration of obstacle avoidance and detection as well. There have been successful real-time-feasible implementations of real-time obstacle avoidance using CNN [56]. Notably, some CNN-based applications are limited to environments with predefined obstacles[5]. A deep Q-learning network DQN-based algorithm was applied successfully facilitating real-time obstacle avoidance [57]. Deep learning algorithms can exhibit high computational complexity in dynamic environments.

The task of real-time obstacle avoidance poses a challenge to individual methods. Multi-sensor solutions and data filtering allow for the mitigation of some of the challenges at the cost of computational complexity. There is a noticeable need for further research into computationally-efficient hybrid methods, integrating the different sensing methods and algorithms.

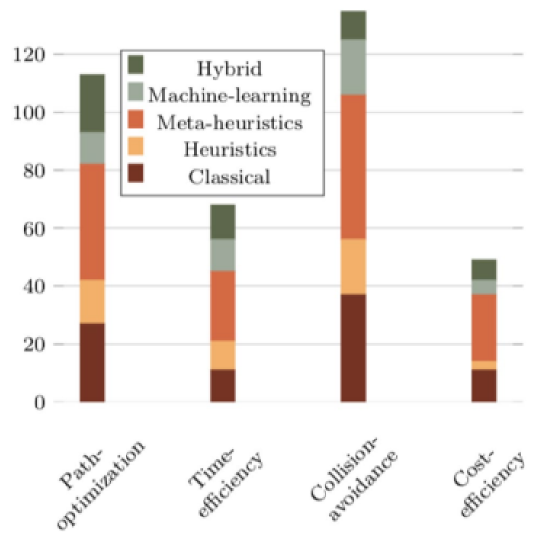
2.6. Summary and Knowledge Gap

Numerous studies investigated the multicopter path planning problem, applying and evaluating the aforementioned classical, soft computing and hybrid methods. There is a noticeable need for further development of path planning algorithms facilitating traversal of highly dynamic environments [7]. As seen in Figure 2.8, hybrid path planning approaches have not been the primary focus of path planning research efforts. Hybrid methods can facilitate good performance in terms of global path planning, as well as enabling traversal of dynamic environments. The hybrid methods are currently often considered computationally demanding. Furthermore, the algorithm's feasibility declines with the increased dimensionality of the search-space. Multicopter hardware limitations require further research into computationally efficient strategies [1].

Local path planning algorithms are integrated with global path planning algorithms, enabling local on-route adjustments. While essential for traversal of cluttered spaces, current online methods frequently exhibit performance limitations in dense environments [1]. The currently applied algorithms fail to maintain real-time performance in dynamic and complex environments with imposed computational limitations[5]. Obstacle detection and avoidance methods utilise sensor data to detect and predict the trajectories of dynamic obstacles, facilitating collision-free traversal of the dynamic environments. Obstacle avoidance methods perform inefficiently for applications with limited computational capabilities in highly dynamic scenarios. These limitations highlight the need for further investigation into the integration of local path planning, sensing and obstacle avoidance [5].



(a) Experimented environment of UAV path planning [8]



(b) Optimization results of UAV path planning [8]

Figure 2.8: Breakdown of approaches assessed in 150 research studies in the time period of 2000-2022 showing limited research into algorithm hybridisation [8].

In conclusion, a clear research gap can be identified. The algorithm limitations are particularly relevant for omnicopter applications due to the significantly increased complexity of the 6D search-space. The current state of UAV path planning research necessitates a focus on hybrid algorithms enabling offline global path planning and online local path planning in dynamic cluttered environments. The integration of sensing, adaptable obstacle avoidance methods and local path planning algorithms facilitating the traversal of dynamic environments requires further research. Lastly, there is a need for the algorithm to be computationally efficient, allowing for real-time implementation. The high dimensionality of the omnicopter applications imposes a significant challenge on the real-time performance of the algorithm.

3 System Architecture and Implementation

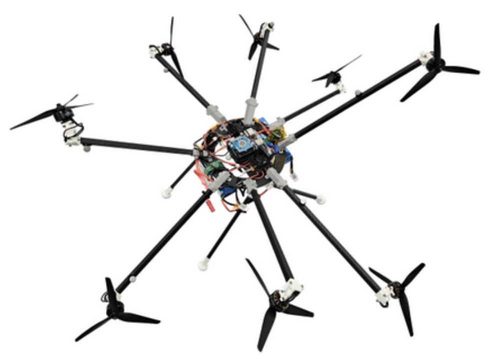
This chapter contextualises the developed local path planner within the greater framework of the omnicopter project of New York University Abu Dhabi in Section 3.1. Subsequently, the section presents the developed architecture and implementation of the local path planner, to address the research questions defined in Section 1.2. The description of the local path planning algorithm implementation in Section 3.2 precedes the explanation of the unmapped obstacle detection and avoidance strategy in Section 3.3. Finally, the computational optimisation steps taken to achieve real time performance and the implemented state machine are addressed in Section 3.4.

3.1. Omnicopter and Simulation

New York University Abu Dhabi (NYUAD) has developed a novel omnidirectional octocopter [58]. The purpose of the omnicopter project is to deepen academic research in the area of omnidirectional UAVs. To enable the omnicopter to autonomously traverse the cluttered environment, NYUAD has implemented an RRT* based global path planner [9]. The global path planner inputs the defined STL map of the testing environment and outputs a list of waypoints defining the desired 6D poses. The algorithm implementation has been successfully tested by commanding the omnicopter through the offline-generated 6D waypoints in simulation. Due to the significant computational overhead the RRT* algorithm cannot be used for online local replanning.

Strict adherence to the global path poses threats, such as potential collision with the environment due to the presence of unmapped obstacles in the environment space or unfeasible global path definitions. Implementation of a collision-aware local path planner would allow for autonomous traversal of dynamic environments. There is a need for a real-time feasible local path planner integrated within the omnicopter's framework.

The implementation of the Local Path Planner must be compatible with the previous developments within the NYUAD's project. The local path planner testing is thus performed in the Gazebo simulation environment developed by NYUAD for global path planner testing[9]. The simulation environment is widely used within the area of UAV development and academic research. Gazebo facilitates realistic physics simulation, as well as the integration of simulated sensing [11]. The simulated omnicopter is commanded within the Gazebo environment via the Robot Operating System (ROS) [59]. The Robot Operating System is an open-source framework facilitating the node communication within the simulation. ROS is widely used in robotics and is highly compatible with the Gazebo simulation environment.


(a) Omnicopter developed by NYUAD [58]

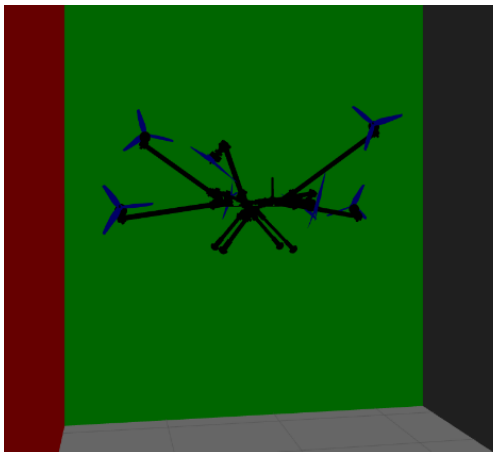

(b) Omnicopter flying in the Gazebo simulation environment

Figure 3.1: The NYUAD-developed omnicopter and the model within the simulation

3.2. Local Path Planner

The local path planner autonomously determines optimal pose and velocity commands leading to efficient collision-free following of the global path and ultimately reaching the global goal. The following section presents the measures taken by the path planning's core algorithm to determine feasible and efficient commands.

3.2.1. Path Planning Algorithm

The core path planning algorithm must facilitate real-time replanning. The selected local path planning algorithm utilises the Dynamic Window Approach [10]. At each time cycle, the Dynamic Window Approach algorithm evaluates sampled velocities that are physically reachable at the given state. Each resulting pose is assessed for potential collision with the environment, with the collision-free poses being scored based on the defined metrics. The algorithm limits the evaluation to only the feasible trajectories, improving computational performance. The trajectory deemed the most favourable is commanded to the controller. The scoring is determined by a multi-objective cost function, allowing for adjustable definition of desired behaviour. As aforementioned in Chapter 2, the DWA is primarily applied in 2D cases, due to the computational effort increasing for higher dimension search spaces.

Adapting the algorithm for the needs of an omnicopter requires consideration of the 6 velocity components, namely V_x, V_y, V_z and $\omega_x, \omega_y, \omega_z$. The 6D-DWA exploits the omnicopter's ability to decouple attitude from translation, facilitating simultaneous path progression and sensor-pointing manoeuvres. A crucial challenge is developing a real-time feasible implementation given the highly increased computational complexity in a 6 dimensional search space. The computational limitations impose a limit on the sampling of the search space, as well as require a computationally efficient collision evaluation. A crucial consideration in the selection of the algorithm is the requirement of real-time performance. The target processing time is 5 Hz or 0.2 s for each planning cycle.

The DWA algorithm is relatively short-sighted, the planner provides only the current best found command with no information on future commands. That characteristic constrains the exploration capabilities of the algorithm. The algorithm must be expanded to improve the obstacle avoidance performance. Computational constraints pose a significant challenge in expanding the replanning capabilities of the planner. The algorithm expansion is further addressed in Section 3.3.

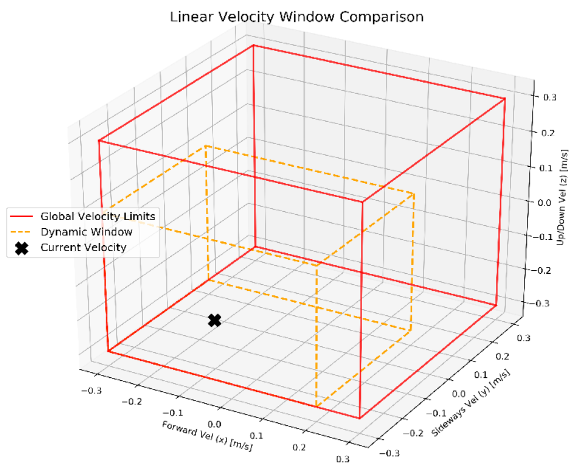
3.2.2. Velocity Sampling

The 6D-DWA evaluates the sampled velocities within the Dynamic Window. The Dynamic Window describes velocities that are achievable given a particular state of the omnicopter. The achievable linear and angular velocities are determined by the current state, hardware-dictated limitations on the linear and angular velocities and accelerations, as well as the defined time step (Δt), as presented in Equation 3.1 and Equation 3.2.

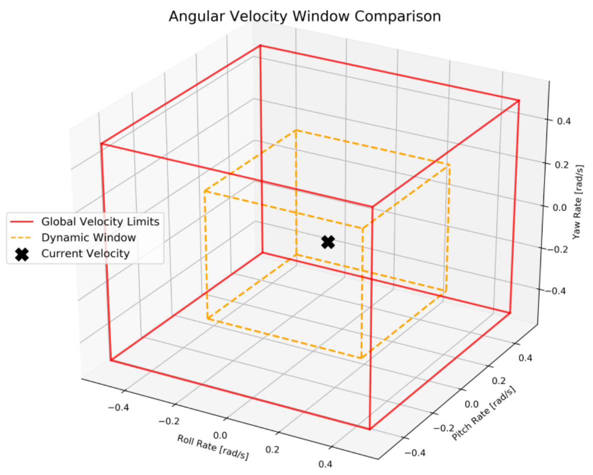
$$\max\left\{\begin{matrix} v_{\text{limit},i}^{\text{min}} \\ v_{\text{curr},i} - a_i^{\text{max}}\Delta t \end{matrix}\right\} \leq v_i \leq \min\left\{\begin{matrix} v_{\text{limit},i}^{\text{max}} \\ v_{\text{curr},i} + a_i^{\text{max}}\Delta t \end{matrix}\right\}, \quad \forall i \in \{x, y, z\} \tag{3.1}$$

$$\max\left\{\begin{matrix} \omega_{\text{limit},j}^{\text{min}} \\ \omega_{\text{curr},j} - \alpha_j^{\text{max}}\Delta t \end{matrix}\right\} \leq \omega_j \leq \min\left\{\begin{matrix} \omega_{\text{limit},j}^{\text{max}} \\ \omega_{\text{curr},j} + \alpha_j^{\text{max}}\Delta t \end{matrix}\right\}, \quad \forall j \in \{x, y, z\} \tag{3.2}$$

In Equation 3.1, a_i^{max} corresponds to the defined hardware linear acceleration limitations. Similarly, in Equation 3.2, α_j^{max} represents the limits of angular acceleration. The dynamic window is visualised in Figure 3.2.



(a) Linear Velocity Window visualisation



(b) Angular Velocity Window visualisation

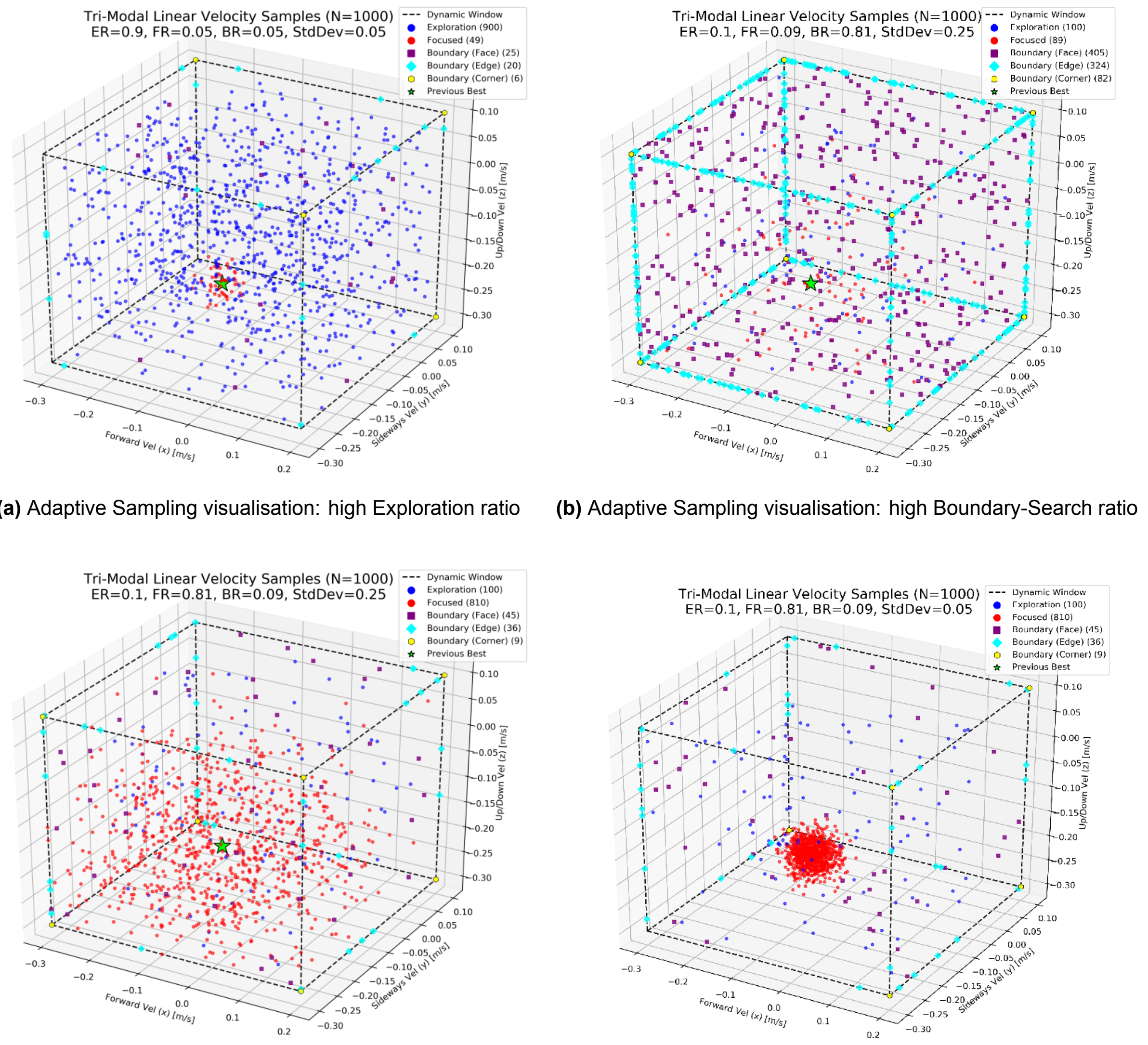
Figure 3.2: Visualisation of the 6-DOF Dynamic Window, only the velocities achievable at the current state that are within the global hardware velocity limits are sampled.

A significant challenge in adapting DWA to 6D-DWA is the vastly increased complexity of the velocity search space. In order for the 6D-DWA to yield satisfactory results, the velocity space must be explored thoroughly and efficiently. The number of samples determines the computational effort required at each 6D-DWA loop, as 6D-DWA has to evaluate and score each velocity sample. To limit the number of samples that need to be evaluated, Adaptive Sampling is implemented. Adaptive Sampling is designed to optimise sampling by performing focused sampling on a part of the total sampling. This ensures sampling in areas with a higher likelihood of high-scoring velocity inputs.

Sampling thus comprises three separate sampling stages determined by configurable ratios that must sum up to 1.0. The first stage is the *Exploration*, in which a fraction (Exploration ratio e_r) is allocated to uniform random distribution sampling. Setting the Exploration ratio to 1.0 thus equates to disabling adaptive sampling. The Exploration stage yields the generation of vastly different samples, allowing for an unbiased coverage of the velocity space.

The second stage is the *Focused-Search*. The Focused-Search ratio (f_r) determines what portion of the total samples is generated around the best-scoring velocity determined in the previous 6D-DWA cycle. This is particularly beneficial for maintaining smooth trajectories. The samples are generated using normal distribution, the standard deviation (σ) of the distribution is scaled relative to the size of the Dynamic Window. This stage is skipped at first iteration, the samples are allocated to the Exploration stage.

Lastly, the *Boundary-Search* stage forces sample generation at the limits of the dynamic window. The Boundary-Search ratio (b_r) determines the number of samples used in the Boundary-Search stage. The samples are first uniformly randomly generated, upon which either one, two or three velocity components are shifted to the nearest Dynamic Window boundary value. This results in forced search of respectively faces, edges and corners of the dynamic window. Boundary Search aims to improve performance in cases in which the most beneficial behaviour is achieving velocity limits. The Adaptive Sampling is visualised in Figure 3.3. The computational impact of the number of samples, as well as the evaluation of the implementation and impact of adaptive sampling is presented in Section 4.2.



(a) Adaptive Sampling visualisation: high Exploration ratio **(b)** Adaptive Sampling visualisation: high Boundary-Search ratio

(c) Adaptive Sampling visualisation: high Focused-Search ratio, high standard deviation **(d)** Adaptive Sampling visualisation: high Focused-Search ratio, low standard deviation

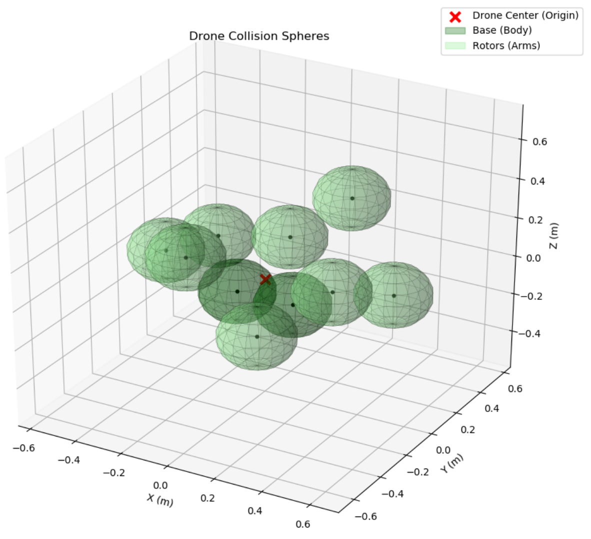
Figure 3.3: Visualisation of the effect of different search ratios in Adaptive Sampling; The chosen sampling ratios impact the sampled velocity values. For high-dimensional search spaces and limited number of samples this can impact the sample quality.

3.2.3. Static Collision Detection

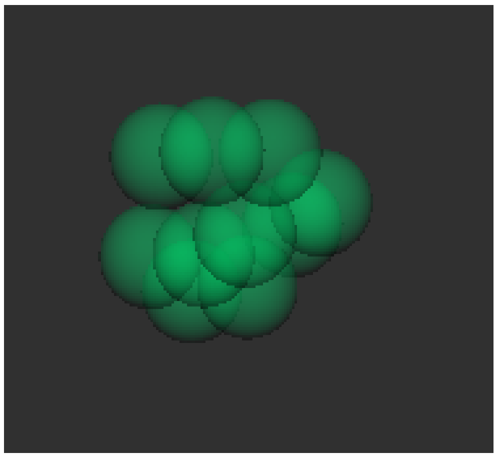
The local path planner must be environment aware, forbidding any collision with the environment. The term 'known environment' is used throughout the paper to indicate static environment indicated in the STL file. In 6D-DWA each trajectory must be evaluated for potential interactions with the environment surfaces.

For large number of samples evaluating each point within the global map is not feasible in real-time. In this implementation, computationally-efficient static environment collision detection is accomplished through structure simplification, environment voxelisation and local map generation.

The omnicopter is simplified and covered by 10 spheres, 8 located at the centres of the rotors and 2 covering the base. Approximating the omnicopter's geometry with spheres minimises computational overhead, as the distance calculation becomes invariant under rotation. The 10-sphere approximation is visualised in Figure 3.4. The centre placement and selected sphere radius (r) of 0.15 m ensures that the omnicopter's structure is encompassed.



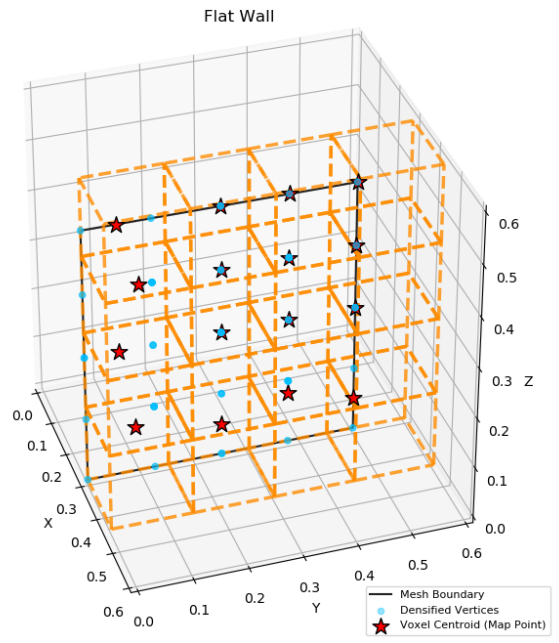
(a) 10-sphere approximation visualisation



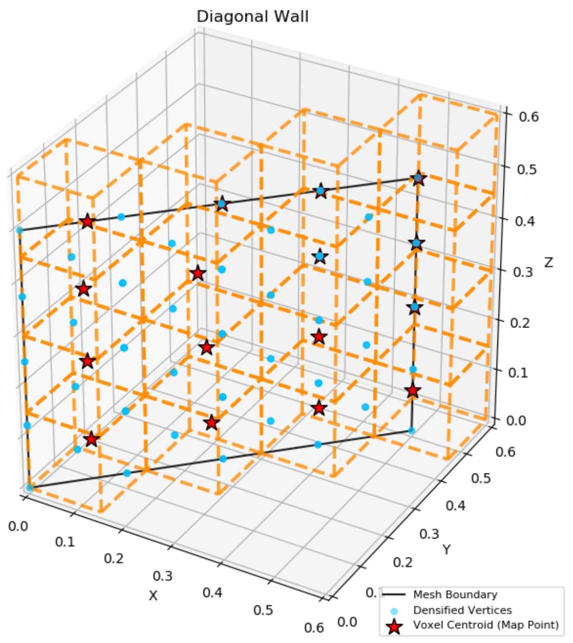
(b) 10-sphere approximation, as seen in RViz [60]

Figure 3.4: Visualisation of the 10-sphere approximation encompassing the omnicopter's structure; The 10-sphere approximation is sufficient encompassing the omnicopter's structure, while maintaining the approximate shape and minimising computational effort.

To guarantee a desired density of points within the obstacle surfaces the loaded STL mesh undergoes densification. The mesh densification applies a recursive method, in which the triangles defined by the vertices are evaluated for edge length. The triangles with edge length over the threshold are bisected by creating a new vertex in the midpoint of the longest edge. The process continues until no triangle has an edge over the defined mesh densification parameter value. To minimise the number of points having to be processed by the algorithm the environment is voxelised. The environment space is partitioned into identical cubic voxels in a regular grid pattern. If at least one vertex point is found within the voxel, the voxel substitutes the points with a single new point (centroid). The location of the new point is an average of the locations of all points found within the voxel. The effect of voxelisation on the environment points is visualised in Figure 3.5. This step is performed prior to take-off.



(a) Voxelisation of a flat wall segment



(b) Voxelisation of a wall angled with respect to the voxel grid

Figure 3.5: Visualisation of the voxelisation method; The voxel centroids substitute multiple points while maintaining desired maximum distances.

To remove the possibility of contact between the defined omnicopter spheres and environment wall faces the omnicopter sphere radius must be artificially increased. The *inflated* radius (r_{inf}) parameter defines the size of a larger concentric sphere, as seen in Figure 3.7. The parameters of mesh densification and voxel size shall be chosen so that no empty voxel can be generated between the densified points. Setting the mesh densification parameter to less or equal to the voxel size ensures that two points will be in the same or adjacent voxels. The maximum distance between centroids corresponds to the diagonal length between the farthest corners of adjacent voxels. This worst-case centroid spread is defined in Figure 3.6.

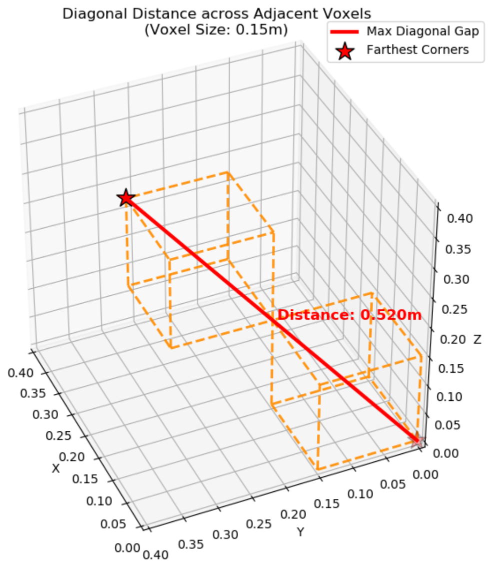
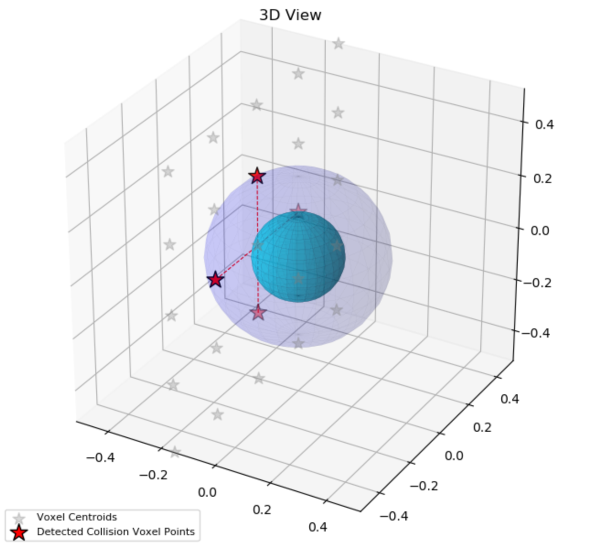
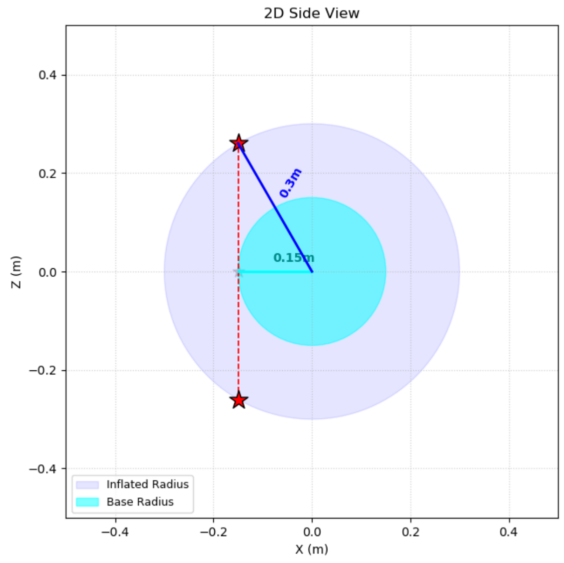


Figure 3.6: Visualisation of the worst-case voxel centroid spread for mesh densification of 0.15 m and 0.15 m voxel edge; This maximum spread must be accounted for in collision evaluation.

For the selected parameters of 0.15 m voxel edge and mesh densification of 0.15 m that distance is 0.52 m, resulting in a required inflated radius of 0.3 m, as shown in Figure 3.7.



(a) Visualisation of the worst-case voxel centroid spread



(b) Effect of inflated radius in case of the worst-case voxel centroid spread

Figure 3.7: Visualisation of the sphere-surface interaction in the worst-case voxel centroid spread for mesh densification of 0.15 m and 0.15 m voxel edge; Increasing the maximum centroid spread necessitates increasing the inflated radius.

The chosen parameters are a result of a trade-off between conflicting goals of minimising the number of points and the minimising the omnicopter's artificial dimensions. The current implementation parameters limit the minimum gap width the omnicopter is able to pass through. The farthest body z-axis protruding sphere centres and the used sphere radius result in a minimum required width 0.98 m. This can be minimised up to the physical limit of 0.68 m through higher environment refinement and lowering the inflation radius. The aforementioned values are the theoretical minima resulting from geometric analysis and do not account for controller overshoot or positional inaccuracies.

An edge case should be acknowledged. If a voxel includes two parallel outside faces of a wall, the centroids will be created inside of the wall. The within-wall shift of the face points should be additionally accounted for. This edge case is not possible given the used testing environment and can be easily mitigated by increasing the inflation radius or lowering the voxel size.

To further limit the number of points having to be processed in each 6D-DWA cycle a local map is generated. The local map filters all the points available in the global map and creates a subset of points within the radius around the centre of the omnicopter. Only the local subset of points is evaluated for collision. The local map updates every time the omnicopter travels farther than the selected rebuild trigger distance. In all tests presented in Chapter 4 the parameter is set to 0.1 m. In summary, the 6D-DWA algorithm evaluates whether the velocity input results in a pose in which a voxel centroid within the local map coincides with at least one of the 10 defined spheres.

3.2.4. Trajectory Propagation

Trajectory propagation entails simulating each velocity sample forward in time to evaluate its kinematic feasibility and potential for collision. For each velocity sample the trajectory must be simulated forward in time. The parameter (t_p) defines the time prediction horizon, while the simulation step (Δt_{step}) defines the time steps taken in the simulation. The planner simulates discretised steps leading to the final pose of the omnicopter. This measure ensures trajectory validity estimation by avoiding validating trajectories passing through an obstacle. The prediction horizon exceeds the planner's time step ΔT, allowing to enhance the planning foresight of the planner. Notably, in highly cluttered environments, high t_p can potentially lead to unnecessary invalidation of trajectories.

As the angular velocities are defined in the body frame, the resulting rotation must be transformed into the world frame for trajectory evaluation. Let the state of the omnicopter at time step k be the position $\mathbf{p}_k \in \mathbb{R}^3$ in the world frame. The orientation is denoted by the unit world-frame state quaternion $\mathbf{q}_k \in \mathbb{H}$. Let $\boldsymbol{\omega}_\text{b}$ be the constant body-frame angular velocity vector in the interval Δt_step. The implementation assumes constant velocity within the time horizon and instantaneous attainment the sampled velocities. For high accelerations this can cause discrepancy between the predicted and actual trajectories. The rotation magnitude $\Delta\theta$ and the rotation axis $\mathbf{u}$ are computed in Equation 3.3 and Equation 3.4 [61].

$$\Delta\theta = \|\boldsymbol{\omega}_\text{b}\| \cdot \Delta t_\text{step} \tag{3.3}$$

$$\mathbf{u} = \frac{\boldsymbol{\omega}_\text{b}}{\|\boldsymbol{\omega}_\text{b}\|} \tag{3.4}$$

If $\omega_\text{b} = [0, 0, 0]^T$, an identity quaternion is used as there is no rotation. If ω_b is non-zero, the magnitude of rotation $\Delta\theta$ and the rotation axis $\mathbf{u}$ are used to compute the rotation quaternion, as presented in Equation 3.5 [61].

$$\mathbf{q}_\Delta = \left[\cos\left(\frac{\Delta\theta}{2}\right),\ \mathbf{u}^T \sin\left(\frac{\Delta\theta}{2}\right)\right]^T \tag{3.5}$$

The subsequent orientation quaternion $\mathbf{q}_{k+1}$ is calculated via quaternion multiplication of the current state quaternion $\mathbf{q}_k$ and the rotation quaternion $\mathbf{q}_\Delta$., as shown in Equation 3.6. The quaternion is normalised to counteract numerical integration drift [61].

$$\mathbf{q}_{k+1} = \frac{\mathbf{q}_k \otimes \mathbf{q}_\Delta}{\|\mathbf{q}_k \otimes \mathbf{q}_\Delta\|} \tag{3.6}$$

Similarly, the linear velocity samples are generated in the body-frame and must be transformed into the world-frame. The body-frame velocity $\mathbf{v}_\text{b}$ is first rotated into the world frame using the current orientation quaternion $\mathbf{q}_k$. The subsequent positional vector $\mathbf{p}_{k+1}$ is then calculated using the Euler method for computational efficiency, as presented in Equation 3.7. The $\mathbf{R}(\mathbf{q}_k)$ denotes the rotation matrix representation corresponding to the orientation state quaternion $\mathbf{q}_k$ [61].

$$\mathbf{p}_{k+1} = \mathbf{p}_k + (\mathbf{R}(\mathbf{q}_k) \cdot \mathbf{v}_\text{b}) \cdot \Delta t_\text{step} \tag{3.7}$$

To perform collision checking, the world-frame positions of the centres of the collision approximation spheres must be determined. At every simulation step k, the global position $\mathbf{p}_{i,k}$ of each sphere centre, where $i \in \{1, 2, \ldots, 10\}$, is calculated by transforming the static body-frame offsets, as presented in Equation 3.8. The $\mathbf{o}_i$ represents the body-frame offset of the centre of a sphere.

$$\mathbf{p}_{i,k} = \mathbf{p}_k + \mathbf{R}(\mathbf{q}_k) \cdot \mathbf{o}_i \tag{3.8}$$

A trajectory is then considered invalid if, for any time step k and any sphere i, the distance to the closest local map obstacle point $\mathcal{O}_\text{obs}$ violates the condition presented in Equation 3.9. The parameter r_inf represents the inflated radius.

$$\exists\, i, k : \quad \text{dist}(\mathbf{p}_{i,k}, \mathcal{O}_\text{obs}) < r_\text{inf} \tag{3.9}$$

The invalid trajectories are discarded. For computational efficiency an early-exit strategy is implemented, if the trajectory is deemed as collision-inducing at a given time step, the trajectory is invalidated without the need to compute further steps. The trajectories deemed valid are evaluated, as explained in Section 3.2.5. Notably, if no trajectories are deemed valid, the omnicopter is commanded to hover at the previously commanded pose, as valid trajectories can potentially be found in subsequent 6D-DWA cycles.

3.2.5. Trajectory Evaluation

The valid trajectories must be evaluated, as the objective is to output solely the best-scoring velocity input. The following section describes the chosen evaluation criteria and the scoring methodology.

The optimal trajectory is the trajectory with the lowest value of the defined cost function. The weighted cost function used by the implemented 6D-DWA is as presented in Equation 3.10.

$$J = w_{\text{goal}}C_{\text{goal}} + w_{\text{path}}C_{\text{path}} + w_{\text{head}}C_{\text{head}} + w_{\text{look}}C_{\text{look}} + w_{\text{clear}}C_{\text{clear}} + w_{\text{face}}C_{\text{face}} \tag{3.10}$$

The first term incentivises rewarding trajectories progressing the omnicopter towards the local goal. The C_{goal} is the distance-based cost function, as presented in Equation 3.11, while the w_{goal} is the corresponding weight of the cost.

$$C_{\text{goal}} = \sqrt{(x_{\text{sim}} - x_{\text{goal}})^2 + (y_{\text{sim}} - y_{\text{goal}})^2 + (z_{\text{sim}} - z_{\text{goal}})^2} \tag{3.11}$$

The C_{goal} is calculated as a distance from the simulated world-frame end-position of the omnicopter's centre $((x_{\text{sim}}, y_{\text{sim}}, z_{\text{sim}}))$ to the position of the local goal $(x_{\text{goal}}, y_{\text{goal}}, z_{\text{goal}})$. The local goal is selected based on the current location of the omnicopter. The index of closest defined global path waypoint to the current location is determined. The local goal is created by applying an index offset of N_{wa} to the closest index, resulting in targeting points further down the global path. If there are not sufficient waypoints ahead of the omnicopter, the local goal is set to be the global goal. Increasing the corresponding weight results in the algorithm favouring progression towards the defined local goal.

The subsequent metric promotes trajectories maintaining the global path. The path-distance cost term C_{path} represents the cross-track error, defined as the minimum perpendicular distance between the predicted position and a local subset of the discretised global path. Consecutive waypoints create K path segments denoted by $\mathcal{S}_i$. The predicted position $\mathbf{p}_{\text{sim}}$ is projected onto a line extending each segment. If the projection is within a segment the Euclidean distance between the simulated position and the projection is calculated. If the projection is outside of the segment the closest segment point to the projection is used in the distance calculation. The C_{path} is thus the lowest calculated distance to a local segment, as presented in Equation 3.12. Lowering the w_{path} would result in lesser incentive to closely follow the path, increasing the likelihood of path diversions.

$$C_{\text{path}} = \min_{i \in K} \left(\text{dist}(\mathbf{p}_{\text{sim}}, \mathcal{S}_i) \right) \tag{3.12}$$

The following term of the weighted cost function J is the heading-error weight w_{head} and cost C_{head}. This metric aims to penalise the orientation error between the simulated pose and the local goal pose. The cost calculation is shown in Equation 3.13.

$$C_{\text{head}} = 2 \arccos \left(|\mathbf{q}_{\text{sim}} \cdot \mathbf{q}_{\text{goal}}| \right) \tag{3.13}$$

The expression $\mathbf{q}_{\text{sim}} \cdot \mathbf{q}_{\text{goal}}$ denotes the dot product of the components of the simulated orientation quaternion $\mathbf{q}_{\text{sim}}$ and desired local goal quaternion $\mathbf{q}_{\text{goal}}$. The absolute value ensures the calculation accounts for the double cover property of quaternions, thus ensuring the shortest angular distance $\theta \in [0, \pi]$ is computed [61]. If the omnicopter is to follow the orientation of the global waypoints more accurately, the w_{head} should be increased.

The subsequent metric within the weighted cost function is a soft-constraint measure implemented due to the single front-facing depth camera setup of the omnicopter. As further described in Section 3.3.1, the current sensing capabilities are limited. For true omnidirectionality in a dynamic environment the planner should have complete sensing coverage of the local environment. To enable obtaining sensing information needed for dynamic obstacle detection the single-camera omnicopter should be facing the desired path segment. This is further addressed in Section 3.3. A 'lookahead' point is identified analogously to the local goal identification, using an independent index offset.

To calculate the 'lookahead' cost C_{look}, the normalized unit vector $\hat{\mathbf{n}}_{fwd}$ defining the pointing of the depth camera is first computed. The computation requires world-frame orientation of the simulated end-pose.

As the camera is pointing towards the omnicopter's positive x-axis, the x-axis is selected, as shown in Equation 3.14. The normalised unit vector $\hat{\mathbf{t}}_{\text{path}}$ pointing toward the 'lookahead' point is then computed. The $\hat{\mathbf{t}}_{\text{path}}$ is calculated based on the positions of the end-pose $\mathbf{p}_{\text{sim}}$ and the 'lookahead point' $\mathbf{p}_{\text{look}}$, as presented in Equation 3.14.

$$\hat{\mathbf{n}}_{\text{fwd}} = \mathbf{R}(\mathbf{q}_{\text{sim}}) \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \hat{\mathbf{t}}_{\text{path}} = \frac{\mathbf{p}_{\text{look}} - \mathbf{p}_{\text{sim}}}{||\mathbf{p}_{\text{look}} - \mathbf{p}_{\text{sim}}||} \tag{3.14}$$

The cost is thus derived using the dot product of the normalised unit vectors $\hat{\mathbf{t}}_{\text{path}}$ and $\hat{\mathbf{n}}_{\text{fwd}}$. The C_{look} calculation is presented in Equation 3.15. The perfect alignment of the camera axis with the 'lookahead' results in zero cost. Increasing the w_{look} prioritises poses where the camera is aligned with the 'lookahead' point, ensuring continuous environmental perception.

$$C_{\text{look}} = 1.0 - (\hat{\mathbf{n}}_{\text{fwd}} \cdot \hat{\mathbf{t}}_{\text{path}}) \tag{3.15}$$

The last two metrics are implemented to aid path diversion caused by unmapped static obstacles. The w_{clear} and C_{clear} correspond to the obstacle clearance weight and cost, respectively. The goal of the metric is to reward trajectories farther from obstacle points. The w_{face} and C_{face} relate to respectively the unmapped obstacle facing weight and cost. This metric is implemented to ensure sufficient environment awareness during diversion manoeuvres. Those two metrics are only applied in a separate context-aware mode, as further described Section 3.3. In presence of only static points, the obstacle clearance weights are set to zero.

Notably, the individual cost terms in Equation 3.10 are calculated using raw physical values with disparate units, such as meters for distance and radians for heading orientation. Consequently, the weights both define the relative priority of each objective and act as scaling factors to account for the different orders of magnitude between the cost terms. The appropriate weight tuning is thus crucial for the 6D-DWA performance. The tuning of the weights is performed empirically. Increasing one metric's weight will subsequently decrease the impact of other metrics on the selection of the optimal command. The weight tuning is presented in Chapter 4. The cost function relies on the relative ratios of the weights rather than their absolute values. The weights are evaluated as percentages of the total weight sum.

The valid velocity sample leading to the lowest weighted cost J is commanded by the planner finalising the core 6D-DWA cycle. Further evaluation metrics could be included depending on the desired behaviour of the system. The inclusion of additional metrics ,such as rewarding velocity consistency or high linear velocity, was considered not relevant for the goals of this particular implementation. Additional metrics can significantly improve algorithm's performance in particular applications, at the cost of increased computational effort.

3.3. Obstacle Detection and Avoidance

The following section describes the measures taken to ensure the local path planner allows collision-free traversal of an environment differing from the STL definition. Static obstacles not defined in the used STL map are further referred to as *unknown*.

3.3.1. Perception

To account for the unmapped obstacles in the planning cycle, the local planner has to integrate sensing data into the 6D-DWA pipeline. The following section describes the sensing setup and sensing data processing.

As of the time of development, the configuration of the NYUAD's omnicopter consisted of a single front-facing depth camera setup. This implementation uses a Gazebo simulated depth camera [60]. The camera is mounted at a body-frame offset of $[0.15, 0, 0]$ m, resulting in the alignment of the depth camera with the positive x-axis direction. The offset from the omnicopter centre is accounted for in the transformations discussed in this section. The camera output point cloud feed is limited to data from the range 0.4 m and

12 m from the camera, allowing to avoid self-detection and minimise the number of points to be processed. The simulation applies Gaussian noise with a standard deviation of 0.007 m to the depth measurements. The inclusion of simulated noise improves the fidelity of the perception pipeline.

The raw output of the simulated depth camera is a point cloud in the camera-reference frame. The point cloud is visualised in Figure 3.8. To facilitate unmapped obstacle detection, the point cloud must be transformed from camera-reference frame to the body frame, and then into the world frame. The rotation matrix $\mathbf{R}_{c\to b}$ describes the transformation from the standard optical frame into the omnicopter's body frame. The vector $\mathbf{t}_{c\to b}$ accounts for the mounting offset, as shown in Equation 3.16.

$$\mathbf{R}_{c\to b} = \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}, \quad \mathbf{t}_{c\to b} = \begin{bmatrix} 0.15 \\ 0 \\ 0 \end{bmatrix} \tag{3.16}$$

The world-frame positions of the points within the point cloud are thus calculated. In Equation 3.17, the $\mathbf{p}_c$ is the raw input points position in the camera-frame, and $\mathbf{p}_w$ is the output position of a point in the world reference frame. The terms $\mathbf{p}_{o,w}$ and $\mathbf{q}_{o,w}$ refer to the omnicopter's position and orientation, respectively, in the world frame.

$$\mathbf{p}_w = \mathbf{p}_{o,w} + \mathbf{R}(\mathbf{q}_{o,w})\left(\mathbf{R}_{c\to b}\mathbf{p}_c + \mathbf{t}_{c\to b}\right) \tag{3.17}$$

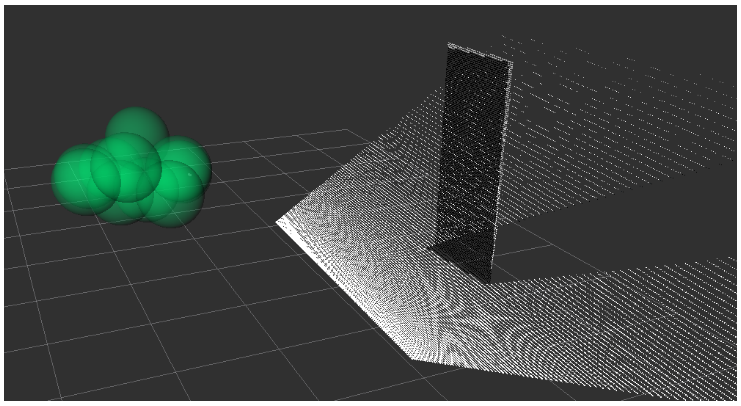


Figure 3.8: RViz view of the generated depth camera point cloud; The excessive number of data points necessitates downsampling.

This transformation framework ensures that all detected environmental features are represented in a unified coordinate system, enabling the local planner to execute accurate avoidance manoeuvres.

3.3.2. Unmapped Obstacle Avoidance

The static-obstacle processing pipeline first utilises a range filter to only include the points within the specified range of interest. The filtered points are then randomly downsampled to a desired total number of points. The downsampled point cloud is then voxelised, analogously to the voxelisation method described in Section 3.2.3, to further minimise the computational effort. The new voxel centroids are added to the local static point map, as described in Section 3.2.3. The inclusion of the processed depth-camera points allows the 6D-DWA to evaluate the trajectories based on a more accurate environment definition. The static-obstacle pipeline outputs obstacle points that are either a part of the map-defined obstacles, unknown static obstacles or moving obstacles. Due to the foresight and computational limitations of the 6D-DWA the moving obstacle points are treated as static at each processing cycle. The 6D-DWA implementation does not facilitate moving obstacle point simulation in the collision evaluation.

The behaviour of the 6D-DWA local path planner is highly dependent on the defined weighted cost function, as discussed in Section 3.2.5. Context-aware *Agile Mode* is introduced to enhance the planners ability to find trajectories around path-interfering unknown obstacles. The core 6D-DWA weights are referred to as *Standard Mode*.

The context-aware Agile Mode activates when an unknown obstacle is detected. An obstacle point is categorised as an unknown if the distance from the obstacle point to the closest point within the STL-defined obstacle map exceeds a defined threshold. This implementation limits the unknown obstacle recognition to obstacles placed farther than the defined threshold from the mapped obstacles. In cases of the unknown obstacle not protruding beyond the threshold, the collision will still be avoided, as the detected obstacle points are included in the static collision checking pipeline. The detection of an unknown obstacle is visualised in Figure 3.9.

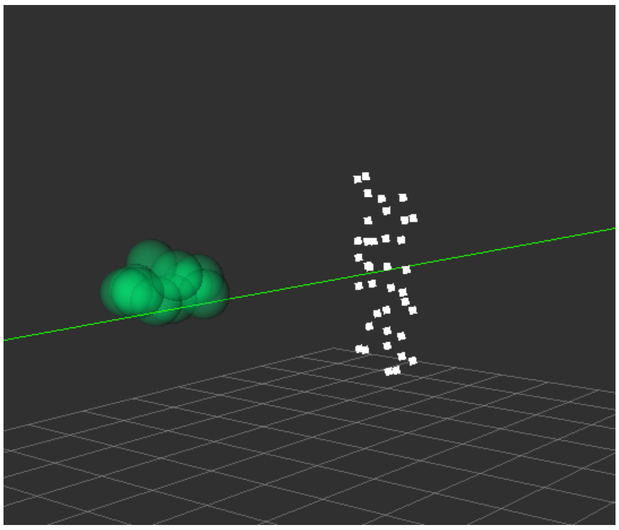


Figure 3.9: RViz view of the identified unknown obstacle points; As the obstacle is placed away from mapped obstacles the planner classifies the obstacle as unknown.

To facilitate the unknown obstacle avoidance two metrics uninitialised in the Standard Mode are present in the 6D-DWA weighted cost function J, as described in Section 3.2.5. The first metric is the obstacle clearance cost C_{clear}. Let the unknown obstacle points be denoted by $\mathcal{O}_{\text{unknown}}$. For a simulated position $\mathbf{p}_{\text{sim}}$, the cost is the sum of inverse-square distances to all k obstacle points' positions $\mathbf{o}_k$ within a defined radius r_{field}. The method is inspired by the Artificial Potential Fields algorithms [62]. The expression for the cumulative clearance cost is presented in Equation 3.18.

$$C_{\text{clear}} = \sum_{\mathbf{o}_k \in \mathcal{O}_{\text{unknown}}} C_{\text{clear},k}(\mathbf{p}_{\text{sim}}, \mathbf{o}_k) \tag{3.18}$$

The individual cost term $C_{\text{clear},k}$ is defined in Equation 3.19.

$$C_{\text{clear},k}(\mathbf{p}_{\text{sim}}, \mathbf{o}_k) = \begin{cases} \frac{1}{\|\mathbf{p}_{\text{sim}} - \mathbf{o}_k\|^2} & \text{if } \|\mathbf{p}_{\text{sim}} - \mathbf{o}_k\| < r_{\text{field}} \\ 0 & \text{otherwise} \end{cases} \tag{3.19}$$

The second metric is the incentive to point the camera towards the unknown obstacle, as defined by the C_{face} cost term. The unknown obstacle 'facing' cost is calculated analogously to the 'lookahead' cost component C_{look} described in Section 3.2.5. Let $\mathbf{o}_{\text{min}}$ be the distance to the nearest point in $\mathcal{O}_{\text{unknown}}$. The $\mathbf{p}_{\text{sim}}$ and $\mathbf{q}_{\text{sim}}$ are respectively the simulated position and orientation of the omnicopter. The calculation of the camera pointing vector $\hat{\mathbf{n}}_{\text{fwd,face}}$ and the relative obstacle vector $\hat{\mathbf{v}}_{\text{obs,face}}$ is presented in Equation 3.20.

$$\hat{\mathbf{n}}_{\text{fwd,face}} = \mathbf{R}(\mathbf{q}_{\text{sim}}) \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \hat{\mathbf{v}}_{\text{obs,face}} = \frac{\mathbf{o}_{\text{min}} - \mathbf{p}_{\text{sim}}}{\|\mathbf{o}_{\text{min}} - \mathbf{p}_{\text{sim}}\|} \tag{3.20}$$

The cost is derived from the dot product, as shown in Equation 3.21

$$C_{\text{face}} = 1.0 - (\hat{\mathbf{n}}_{\text{fwd,face}} \cdot \hat{\mathbf{v}}_{\text{obs,face}}) \tag{3.21}$$

While in the Standard Mode the planner does not include the clearance and obstacle facing costs in the scoring, weights w_{clear} and w_{face} are set to zero. Additionally, to allow for the planner to prioritise unknown obstacle clearance over strict global path following, the other metric weights can be adjusted as well. The context-aware Agile Mode allows for lowering metrics constraining the exploration of the space, such as path adherence. The defined weights consist thus of two sets, standard weights and Agile Mode weights, as shown in Equation 3.22. In Chapter 4, the path adherence weight $w_{\text{goal}}^{\text{agile}}$ is minimised to enable path deviation. The evaluation of the effect of adjusting the 6D-DWA weights is presented in Chapter 4.

$$J = \begin{cases} J(w_{\text{goal}}^{\text{agile}}, w_{\text{path}}^{\text{agile}}, w_{\text{head}}^{\text{agile}}, w_{\text{look}}^{\text{agile}}, w_{\text{clear}}^{\text{agile}}, w_{\text{face}}^{\text{agile}}) & \text{if } |\mathcal{O}_{\text{unknown}}| > 0 \\ J(w_{\text{goal}}, w_{\text{path}}, w_{\text{head}}, w_{\text{look}}, 0, 0) & \text{if } |\mathcal{O}_{\text{unknown}}| = 0 \end{cases} \tag{3.22}$$

3.3.3. Evasion of Moving Obstacles

The point cloud processing pipeline described in Section 3.3.1 is not suitable for the identification and state estimation of fast-moving obstacles. A separate point cloud processing pipeline must be implemented to facilitate the detection of fast-moving obstacles. The processing should allow for high-frequency clustering of the depth-camera data, tracking of the obstacle clusters and estimation of the obstacles' state. Such estimation can be achieved through state-estimation algorithms allowing for noise handling, such as the Kalman Filtering [54]. The detection of a moving obstacle should be communicated to the planner via a threat message. The threat message should contain information on the identified threat's estimated position and velocity at a specific timestamp. Throughout testing of the static-aware evasion, artificially generated threat messages are received by the planner node.

The trajectory estimation calculation accounts for the latency in communication between the perception node and the planner node. The latency $\Delta t_{\text{latency}}$ is calculated as the time difference of the time stamp of the received message and the current time. The node structure and communication is further explained in Section 3.4. Let $\mathbf{p}_{\text{est}}$ denote the estimated current position of the moving obstacle, and the $\mathbf{p}_{\text{msg}}$ and $\mathbf{v}_{\text{msg}}$ be respectively the received obstacle position and velocity. The calculation of $\mathbf{p}_{\text{est}}$ is shown in Equation 3.23. Due to the real-time state estimation limitations, obstacle velocity is assumed to be constant, potentially limiting the estimation accuracy. Higher-order motion models can be employed given sufficient confidence in obstacle acceleration estimation at the cost of additional computational load.

$$\mathbf{p}_{\text{est}} = \mathbf{p}_{\text{msg}} + (\mathbf{v}_{\text{msg}} \cdot \Delta t_{\text{latency}}) \tag{3.23}$$

In order to evaluate whether an evasion manoeuvre should be initiated, the time to the closest point of approach t_{cpa} is determined. If the time determined exceeds the defined threshold, evasion is not initiated. This measure forbids premature evasion. The t_{cpa} is calculated using the relative position vector **r** and the received position and velocity vectors, as shown in Equation 3.26. The relative position vector **r** and relative velocity vector $\mathbf{v}_{\text{rel}}$ are defined in Equation 3.24 and Equation 3.25 respectively.

$$\mathbf{r} = \mathbf{p}_{\text{est}} - \mathbf{p}_{\text{omnicopter}} \tag{3.24}$$

$$\mathbf{v}_{\text{rel}} = \mathbf{v}_{\text{msg}} - \mathbf{v}_{\text{omnicopter}} \tag{3.25}$$

The time to closest point of approach is calculated as presented in Equation 3.26.

$$t_{\text{cpa}} = \max\left(0, -\frac{\mathbf{r} \cdot \mathbf{v}_{\text{rel}}}{\|\mathbf{v}_{\text{rel}}\|^2}\right) \tag{3.26}$$

In order to evaluate whether a detected moving obstacle will collide with the omnicopter, the shortest distance between the obstacle and the omnicopter to happen throughout the obstacle's movement must be determined. The shortest distance calculation is presented in Equation 3.27. The omnicopter's structure is encompassed by a single sphere of radius $r_{\text{bounding-sphere}}$ for collision detection. The moving obstacle radius r_{obstacle} is a set estimate of the moving obstacle. If the shortest determined distance is lower than the sum of defined radii collision is detected.

$$d_{\text{miss}} = \|\mathbf{r} + \mathbf{v}_{\text{rel}} \cdot t_{\text{cpa}}\| \tag{3.27}$$

A detected collision activates a planning pipeline separate to the 6D-DWA. The implemented 6D-DWA is not suitable for fast evasion of moving obstacles. A fast evasion command must be generated, accounting for the static environment to ensure collision-free evasion. The optimal evasion vector $\hat{\mathbf{e}}_{\text{opt}}$ is found based on the received estimated fast-moving obstacle velocity vector.

$$\hat{\mathbf{e}}_{opt} = \frac{\mathbf{v}_{\text{msg}} \times \mathbf{r}}{||\mathbf{v}_{\text{msg}} \times \mathbf{r}||} \tag{3.28}$$

This cross-product formulation ensures that the primary evasion direction is perpendicular to the threat's velocity and the relative position vector. As the optimal evasion might result in a collision, a set of candidate evasion vectors $\mathcal{E}$ is generated by rotating the optimal evasion vector $\hat{\mathbf{e}}_{\text{opt}}$ around threat velocity unit vector $\hat{\mathbf{v}}_{\text{msg}}$. The rotation angle θ_i is determined based on the desired number of evasion candidates n_{vectors}, as shown in Equation 3.29

$$\mathcal{E} = \left\{ \mathbf{R}(\theta_i, \hat{\mathbf{v}}_{\text{msg}})\hat{\mathbf{e}}_{\text{opt}} \;\middle|\; \theta_i = \frac{2\pi i}{n_{\text{vectors}}},\; i = 0, \ldots, n_{\text{vectors}} - 1 \right\} \tag{3.29}$$

Increasing the number of candidate vectors n_{vectors} increases the likelihood of finding a feasible evasion trajectory in cluttered environments, at the cost of computational time due to potential evaluation of more unfeasible trajectories. The resulting set of candidate evasion vectors is evaluated for collision with the local static map points, using the 10-sphere body approximation as described in Section 3.2.3. The trajectory is simulated as a movement along the evasion vector with the set evasion velocity. A defined number of steps is evaluated for each vector to ensure a trajectory passing through an obstacle is not validated. The first found evasion vector resulting in a collision-free trajectory is commanded. If no candidate vector results in a collision-free trajectory the omnicopter does not perform the evasion manoeuvre. The planner prioritises mapped static environment collision avoidance over evasion due to a potential collision with a fast-moving obstacle.

As the evasion manoeuvre finishes, the 6D-DWA activates, facilitating the identification of a collision-free trajectory leading back to the global path waypoints. The transition logic is facilitated through the state machine. The state machine logic is explained in Section 3.4.1.

3.4. Code Architecture

The following section includes specification on the used computational measures as well as the outline of the established node communication. The local path planner with sensing processing is implemented within the Robot Operating System using C++17. The implemented system comprises two nodes, the planner node and the perception node.

The perception node performs the frame transformation and clustering, as described in Section 3.3. The perception node subscribes to the Gazebo-published raw depth point cloud and the state topics. The node outputs the processed point cloud to be integrated with the local static map, as well as a threat messages containing the ID, position, and velocity of tracked moving obstacles.

The planner node encapsulates the 6D-DWA algorithm, static environment processing and the state machine. Three separate C++ classes are established. The 6D-DWA Simulator class performs the trajectory simulation and scoring. The Map Manager class processes the static environment data. The state machine and node communication are facilitated through the Planner class. The implemented state machine is further discussed in Section 3.4.1. The planner node outputs a message to the flight controller, containing the desired pose and velocities of the omnicopter.

In order to achieve high computational efficiency several computational optimisations are implemented. Both the static map management and the clustering algorithms utilise k-dimensional trees (KD-Trees) via the Point Cloud Library (PCL [63]). The KD-trees implementation reduces the complexity of nearest-neighbour searches from linear time $\mathcal{O}(N)$ to logarithmic time $\mathcal{O}(\log N)$. The implementation utilises C++ multi-threading for parallelisation of the simulation loop across available CPU cores. All transformations, quaternion and matrix operations are performed using the highly optimised Eigen3 library [64]. Eigen utilises Single Instruction, Multiple Data (SIMD) vectorisation instructions, which offer significant improvements in computational efficiency compared to standard array operations.

3.4.1. Finite State Machine

The implemented finite state machine acts as a robustness and safety layer, managing high-level mission logic and compensating for the limitations of the planner. The finite state machine facilitates the transition between the following operational states.

- ***Initialisation and Stabilisation***: Upon initialisation, the omnicopter awaits the completion of required preprocessing steps and the reception of the Gazebo-published state messages. The state transitions into the Take-off state.
- ***Take-off***: The omnicopter is commanded to ascend to a defined altitude and enter hover. That measure is implemented as the implemented omnicopter structure overestimation in collision checking would result in detected collision with ground surface, invalidating any trajectory. The state transitions into the Running (DWA Loop) state.
- ***Running (6D-DWA Loop)***: The 6D-DWA operating state. The planner performs steps described in Section 3.2. It is to be noted that if the 6D-DWA does not find a valid command or the processing time exceeds a defined time threshold the omnicopter is commanded to maintain hover at previously validated pose, transitioning into the Post Failure Hover state.
- ***Post-Failure Hover***: A hover state is defaulted to in case the 6D-DWA fails to complete the cycle and find a valid trajectory within the specified time. It transitions back into the 6D-DWA operating state
- ***Evading***: High-priority state initiated by collision detection of a moving obstacle on a collision-track. The state performs the evasion manoeuvre as described in Section 3.3.3. The Evading state overrides the Running state and transitions into the Post-Evasion Hover state.
- ***Post-Evasion Hover***: After the evasion manoeuvre is completed the omnicopter enters a hover state in order to stabilise the omnicopter. The state transitions into the Running (6D-DWA) state.
- ***Landing / Goal Reached***: If the omnicopter is within a defined radius of the global goal, the Running (6D-DWA) state is overridden and landing is initiated, upon which completion the run is finalised.
- ***Stuck***: If the omnicopter fails to travel a minimal distance within a given time frame the omnicopter is considered to be stuck and lands. This measure is necessary to account for the omnicopter being trapped in local minima.

The flowchart of the finite state machine is presented in Figure 3.10, with the colour red indicating the priority of the evasion pipeline over the Running (6D-DWA) state.

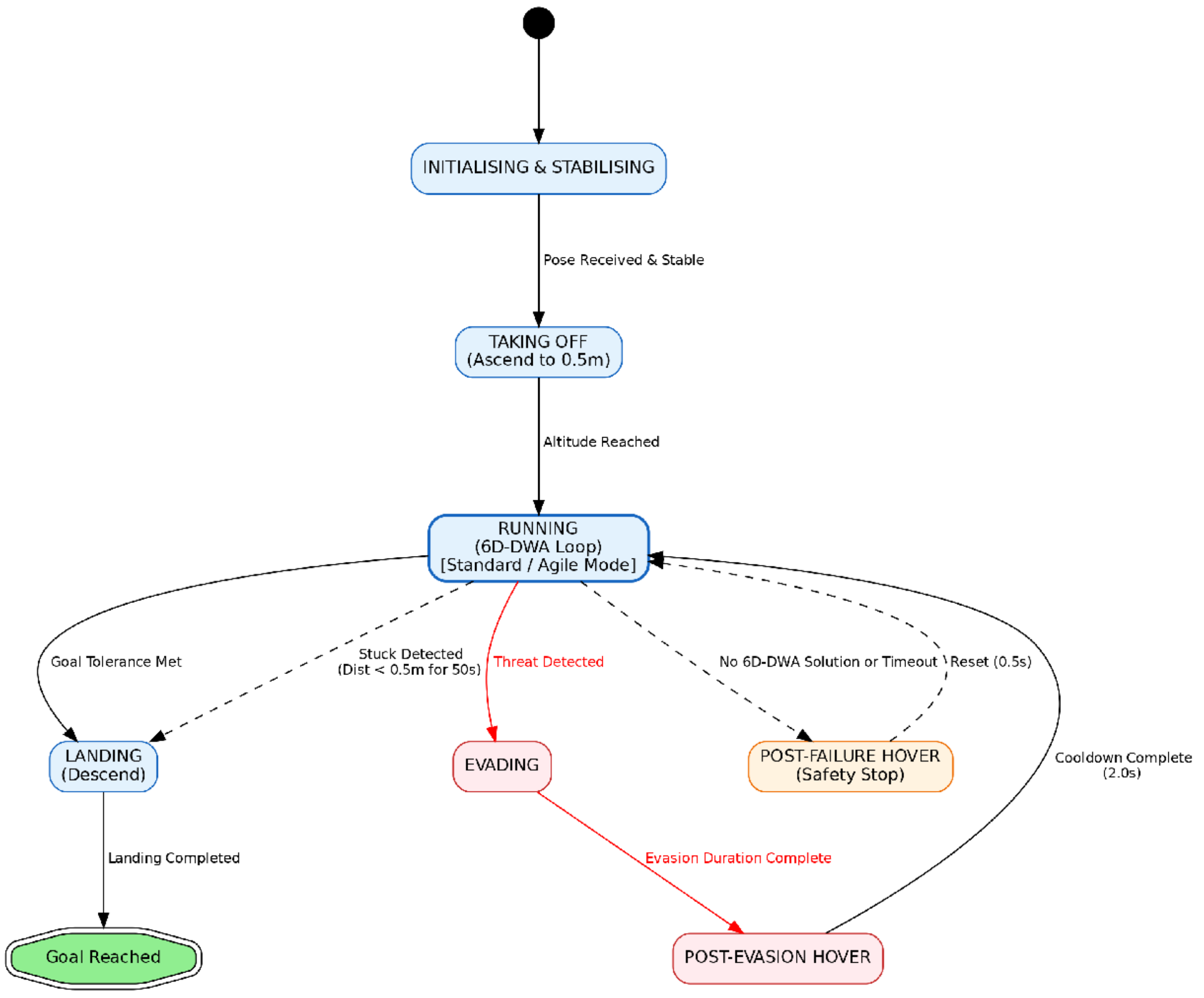

INITIALISING & STABILISING
Pose Received & Stable
TAKING OFF (Ascend to 0.5m)
Altitude Reached
RUNNING (6D-DWA Loop) [Standard / Agile Mode]
Goal Tolerance Met
Stuck Detected (Dist < 0.5m for 50s)
Threat Detected
No 6D-DWA Solution or Timeout
Reset (0.5s)
Cooldown Complete (2.0s)
LANDING (Descend)
EVADING
POST-FAILURE HOVER (Safety Stop)
Landing Completed
Evasion Duration Complete
Goal Reached
POST-EVASION HOVER

Figure 3.10: Flowchart of the finite state machine; The moving obstacle evasion takes priority over the 6D-DWA. The planner transitions between states based on the defined criteria.

4 Experimental Results

The following chapter describes the testing frameworks and presents the results of the performed simulation experiments. As discussed in Section 3.1, the high fidelity of the Gazebo simulation environment allows for accurate verification and validation of the developed planner. These experiments are structured to systematically address the research questions defined in Section 1.2, specifically evaluating computational feasibility (RQ1) and the robustness of obstacle avoidance (RQ2).

The chapter first presents the evaluation of the computational performance in Section 4.1, followed by an assessment of the implemented sampling strategy in Section 4.2. The weight tuning of the core 6D-DWA and Agile Mode 6D-DWA is then presented in Section 4.3 and Section 4.6, respectively. The path-following performance is discussed in Section 4.4. Lastly, the implemented moving obstacle evasion strategy is evaluated in Section 4.7.

Throughout the chapter, tables are used to highlight parameters relevant to a given testing scenario. If a parameter used in a testing scenario differs from a set baseline parameter configuration, this parameter is presented in a corresponding table. The set baseline parameter values are presented in Appendix A.

4.1. Computational Performance

This section evaluates the computational performance of the core 6D-DWA planner. To ensure real-time feasibility the effect of variation of relevant parameters on the computation time needs to be examined. The computational performance is evaluated for different numbers of samples in an environment with varying number of static obstacle points. The testing scenario 1.1 is established using parameters described in Table 4.1.

Table 4.1: Computational evaluation testing scenario parameters

Test:	Testing Scenario	Trials per configuration	Number of samples
1.1	Static floor, varying altitude	5	1000, 5000, 10000, 20000

The testing scenario 1.1 yielded results plotted in Figure 4.1. As expected, increasing the number of samples results in an increase of the average computation time of the 6D-DWA loop. Additionally, the number of obstacle points in the local map has a direct effect on the computation time. For 20000 samples the average computation time exceeds the defined threshold if more than 54 obstacle points are present.

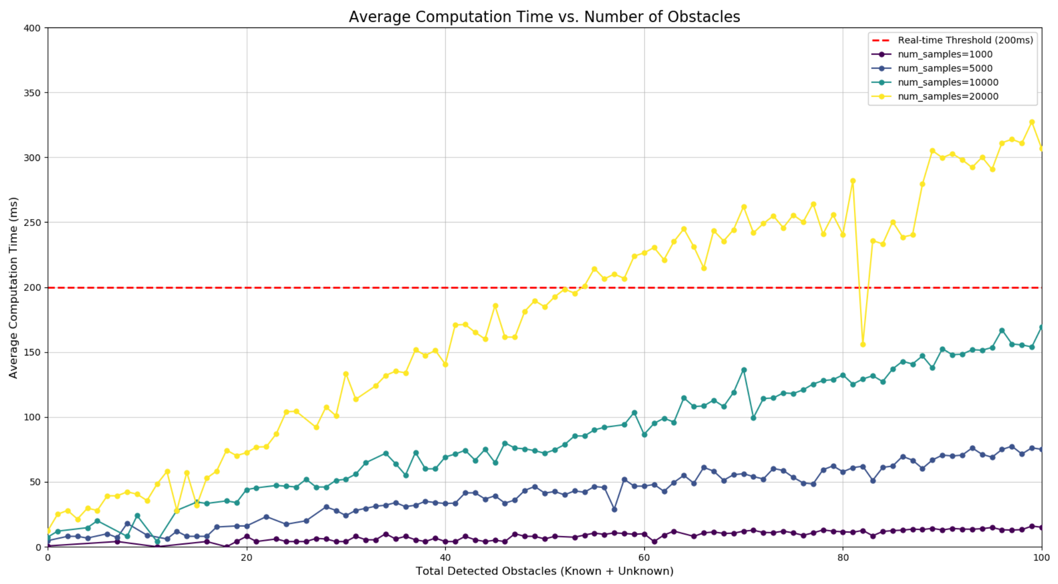


Figure 4.1: Average computation time vs a number of obstacles for varying number of samples; The computational effort increases for higher number of obstacle points in the local map and for higher number of samples.

Linear regression has been applied to estimate the allowable number of obstacles for each sampling configuration. The resulting regression lines are shown in Figure 4.2. The relationship between the computation time and number of obstacles is linear for all sets, allowing for the estimation of the maximum allowable number of obstacle points for given number of samples. The estimated allowable number of obstacle points processed at each 6D-DWA loop is 142 for 10000 samples, 332 for 5000 samples and 1704 for 1000 samples.

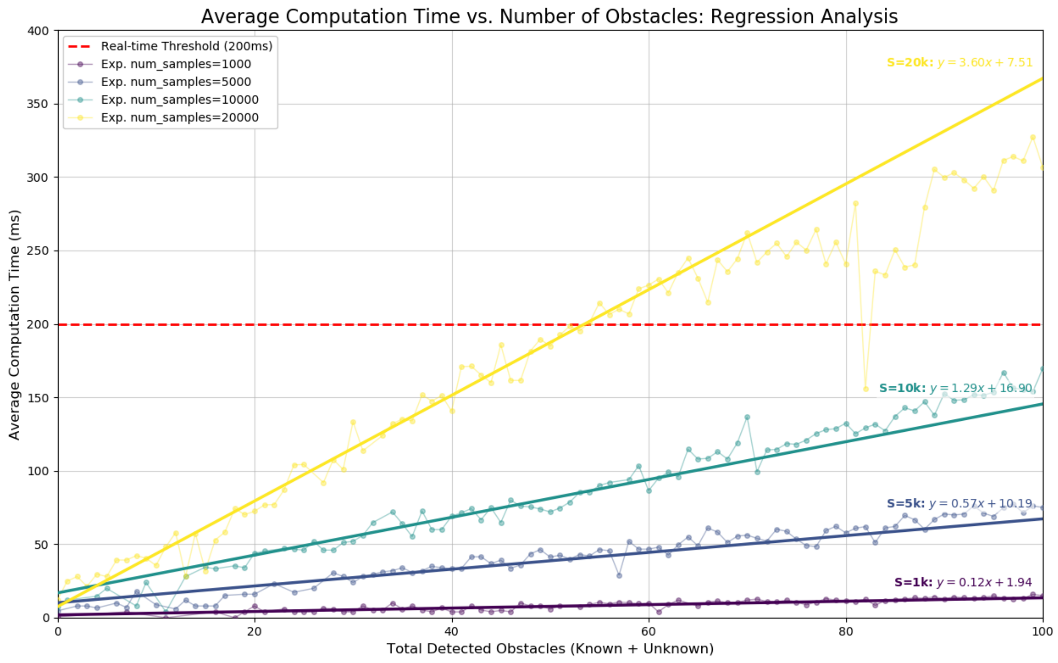


Figure 4.2: Average computation time vs a number of obstacles for varying number of samples with regression lines; All configurations exhibit linear relationship between the number of obstacles and computation time.

The path-following performance of each configuration must be evaluated to determine a desired number of samples to be used. The evaluation of the average cross-track error and the heading orientation error is presented in Figure 4.3. The cross-track error and the orientation error calculations follow from Section 3.2.5.

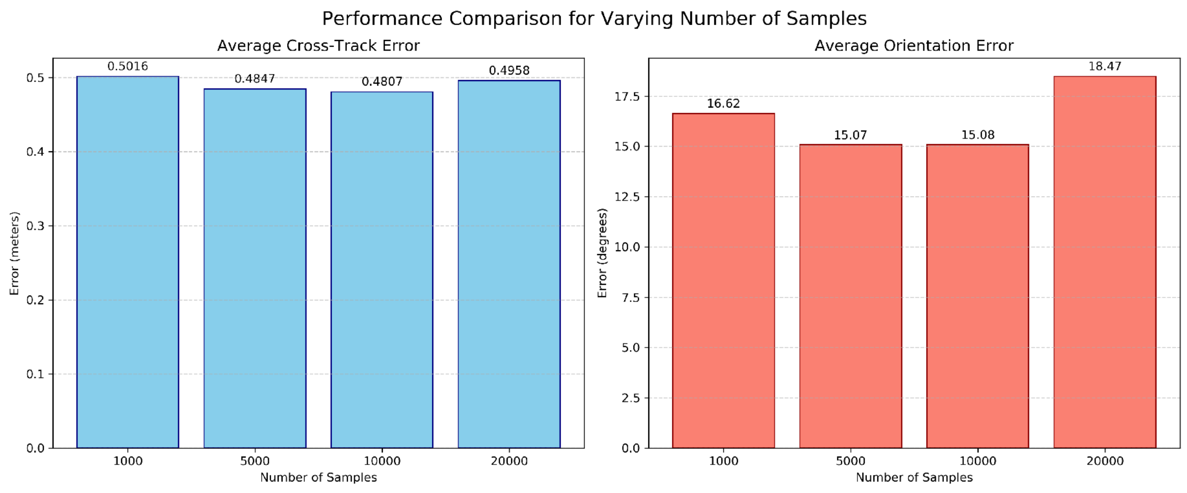


Figure 4.3: Averaged path-following performance for varying number of samples; The errors decrease for higher sample numbers only if the computation time is consistently below the defined threshold.

As expected, the errors decrease with the increasing number of samples for both the heading orientation error and the cross-track error. The performance degradation for the configuration of 20000 samples, as compared to the 10000 sample configuration, can be attributed to the computation time performance. As the 20000 sample configuration yields times above the desired threshold, as seen in Figure 4.1, the planner fails to establish a favourable command thus degrading the performance. Performance can thus be optimised by increasing the number of samples, only if the resulting computation time falls consistently below the time threshold.

The number of static points to be processed is dependent on the parameters defined in Section 3.2.3. Increasing the local map radius or lowering the voxel dimension results in a higher number of points in the static map. The real-time feasibility can then be maintained by reducing the number of samples, this can however degrade the performance. In further testing, the number of samples used is 5000, as presented in Appendix A, unless otherwise specified.

The 6D-DWA architecture attains the required real-time performance by consistently operating below the loop-time threshold. Exceeding that threshold results in degradation of the quality of the generated path, requiring limitation of the computational load. The computational load is governed by the number of velocity samples and the number of obstacle points. Consequently, the algorithm necessitates a trade-off between the accuracy of the environment representation and the number of velocity samples. While the computational performance can potentially differ in real-world hardware implementations, real-time feasibility can be preserved through parameter adjustment, at the potential expense of path optimality.

4.2. Velocity Sampling Evaluation

The following section assesses the implemented adaptive sampling. The parameter configurations used in testing are presented in Table 4.2. Only the configurations in which the sum of Exploration ratio, Focused-Search ratio and Boundary-Search ratio is equal to 1.0 are used in testing.

The outcomes of the aforementioned testing configurations are presented in Figure 4.4. As expected, the random sampling shows lower errors for higher number of samples. All configurations with the adaptive sampling implemented perform better in both metrics than the purely randomly sampled 1000-sample configuration. The implementation of adaptive sampling successfully augments the performance of the planner given a set number of velocity samples. High performance differences are a direct result of a 6D velocity search space. Due to the high-dimensionality of the space, random sampling is less likely to yield

Table 4.2: Velocity sampling evaluation testing scenarios

Test	Testing Scenario	Trials per config.	Number of samples	Explor. ratio	Focus. ratio	Bound. ratio	Std. dev.
2.1	Random sampling	5	1000, 5000	1.0	N/A	N/A	N/A
2.2	Adaptive sampling	5	1000	0.1, 0.5	0.9, 0.5	N/A	0.1, 0.2
2.3	Adaptive sampling with boundary search	5	1000	0.1, 0.5	0.1, 0.25, 0.4	0.1, 0.45, 0.8	0.1, 0.2

well-performing velocity samples. The impact of adaptive sampling should decrease for lower velocity limits, as well as for higher numbers of samples.

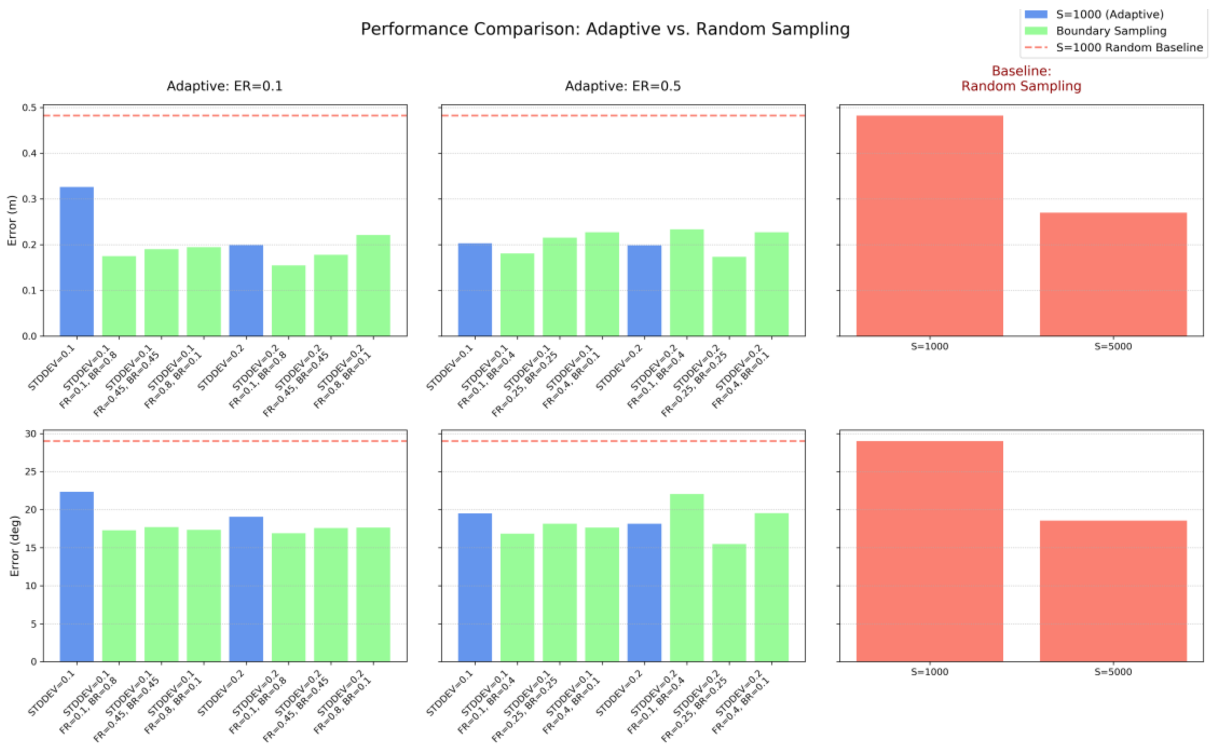


Figure 4.4: Average cross-track and orientation performance of the Adaptive Sampling without Boundary-Search, Adaptive Sampling with Boundary-Search, and random sampling configurations; The Adaptive Sampling strategy enables the planner to achieve comparable or superior accuracy with only 20% of the samples required for the same performance using random distribution.

The observed performance degradation of adaptive sampling for Exploration ratio of 0.1 and σ of 0.1 can be attributed to insufficient exploration of the velocity search space. As 90% of the samples are focused closely around the previous best solution, the exploration of other velocities is limited. The variations of Focused-Search and Boundary-Search ratios yielded no clear trend. It is possible that the sampling parameters would have a higher variation in performance given higher velocity limits. The configurations with the Exploration ratio of 0.5 and standard deviation of 0.1 consistently exhibit performance on-par with or superior to the random 5000-sample configuration. The baseline sampling parameters are thus set to Exploration ratio of 0.5, Focused-Search ratio of 0.25, Focused-Search sampling deviation of 0.1 and Boundary-Search ratio of 0.25. The baseline parameters are presented in Appendix A.

The significantly increased search-space dimensionality and limited computational capabilities limit the possible coverage necessitating an improvement in sample quality. The adaptive sampling implementation facilitates this improvement, yielding consistently higher performance over the purely random sampling for the same number of samples. The variation in sampling ratios demonstrated negligible impact on performance for Adaptive Sampling with Boundary-Search.

4.3. 6D-DWA Weight Tuning

The following section describes the weight tuning of the core 6D-DWA weights, as defined in Section 3.2.5. To find the best performing weight distribution a grid search using weights presented in Table 4.3. In each trial, the omnicopter follows the same curved global path of varying altitude in an empty simulation environment.

Table 4.3: 6D-DWA weight tuning weight configurations

Test:	**Trials per config.**	w_{goal}	w_{head}	w_{path}	w_{look}
3.1	3	20, 30, 40, 50, 60	10, 20, 30, 40, 50	0, 10, 20, 30, 40	0, 10, 20, 30, 40

To determine the best-performing 6D-DWA weight configurations a scoring function is defined. Each trial is scored based on the average cross-track error, average orientation heading error, total covered distance and average 'lookahead' error, as discussed in Section 3.2.5. The final score is calculated via a weighted score function, with the corresponding metric weights presented in Table 4.4.

Table 4.4: Scoring weight configurations for the sensitivity analysis

Scoring Scenario	**CTE Metric** $w_{\text{CTE}}^{\text{score}}$	**Heading Metric** $w_{\text{ori}}^{\text{score}}$	**Distance Metric** $w_{\text{dist}}^{\text{score}}$	**Lookahead Metric** $w_{\text{look}}^{\text{score}}$
Baseline	0.45	0.45	0.10	0.00
Slight-Facing	0.40	0.40	0.10	0.10
Strong-Facing	0.30	0.30	0.10	0.30
Facing-Focused	0.20	0.20	0.10	0.50

As presented in Table 4.3, each of the resulting configurations is evaluated in 3 trials. To ensure that the number of trials is sufficient for accurate evaluation, the differences between scores of different trials within each configuration are calculated. The run-to-run performance variation is plotted in Figure 4.5. The scores are normalised to where 1.0 is the score obtained by the best scoring trial of Test 3.1.

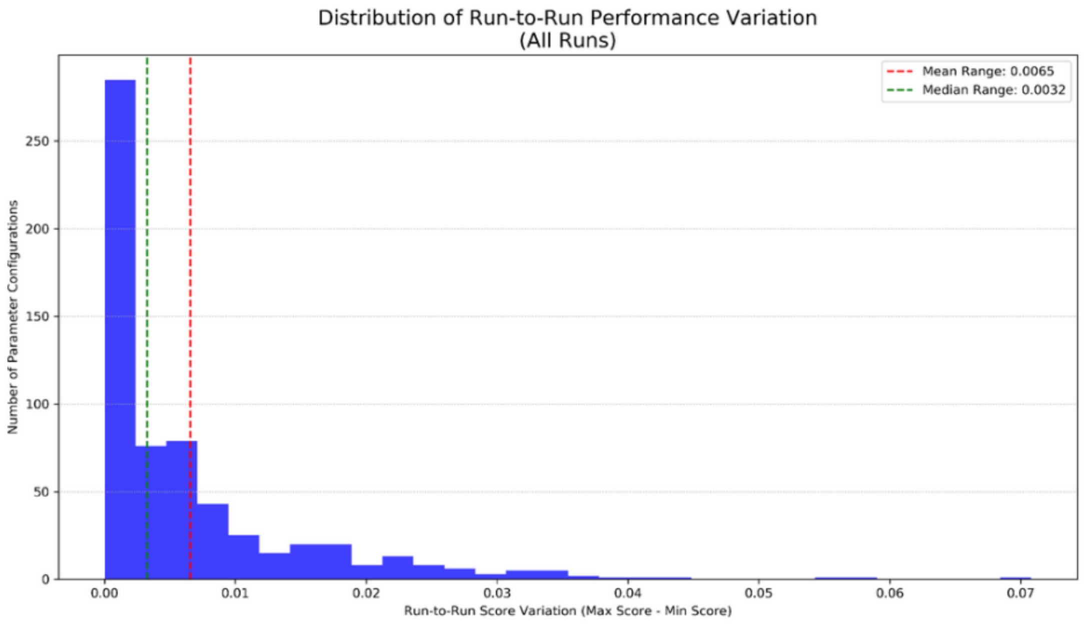


Figure 4.5: 6D-DWA weight tuning run-to-run scoring variations; There are negligible differences between the scores of trials within the same configuration.

As seen in Figure 4.5, the median scoring range between the trials is less than 0.1%. The trial scores are consistent for each configuration, providing confidence in the configuration evaluation. The total score distribution is presented in Figure 4.5. The top 2% best scoring configurations for each scoring scenario are averaged. The averages, as well as the ranges of the best performing configurations are presented in Figure 4.6. While the weights used in testing are as in Table 4.3, the weight relative distribution should be evaluated. The weights are included in the 6D-DWA scoring function, as defined in Section 3.2.5. The effective percentage of total weight sum determines each metric's impact on the score, as opposed to the weight value. The 6D-DWA weights are thus represented as the percentage of total weight sum. The low run-to-run variation ensures that there is no bias towards over-represented or under-represented relative weight distributions.

As shown in Figure 4.6, the best performing runs for all scoring scenarios result in the best performing relative average w_{goal} of approximately 37%. The consistency across different scoring scenarios suggests that the metric is less sensitive to specific behavioral trade-offs between path precision and environmental perception. Similarly, the various scoring configurations exhibited negligible influence on the average best performing relative w_{head}, averaging approximately 31%. As expected, the scoring scenarios with higher contribution of lookahead scoring metric favour the configurations of higher w_{look} on average. The relative average w_{look} decreases for scoring configurations with lower score contribution of cross-track errors.

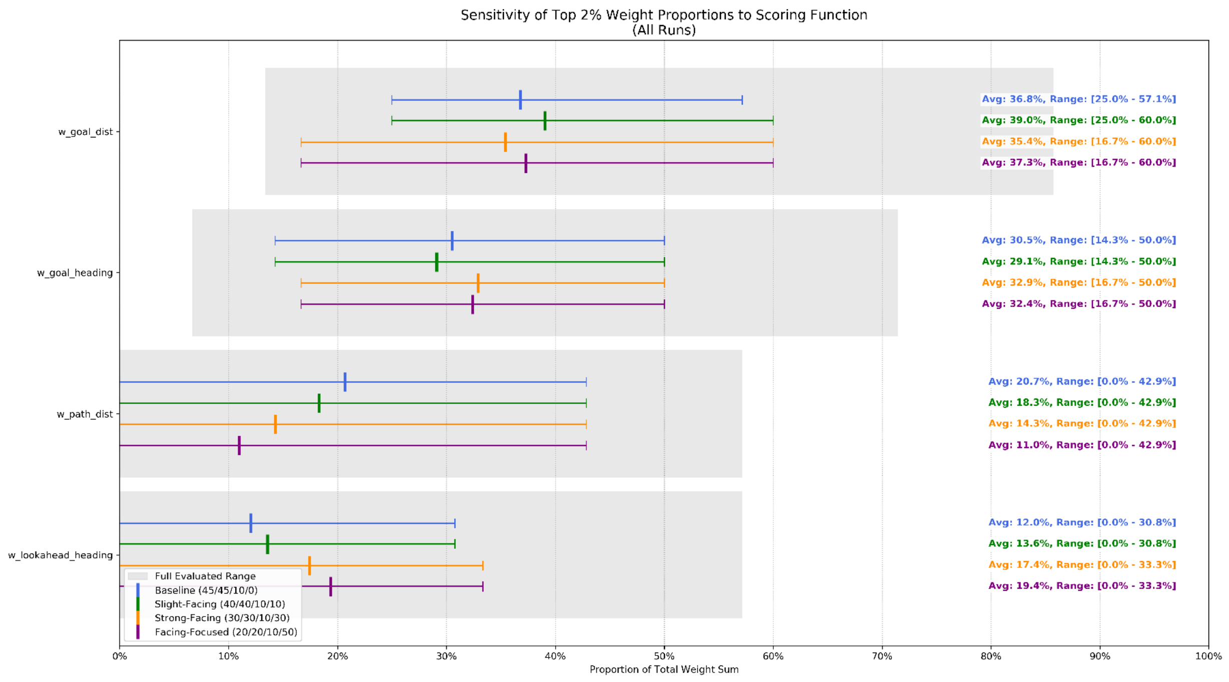


Figure 4.6: Averaged best performing weight configurations for each scoring configuration. The averaged weight values offer a good benchmark for the 6D-DWA. Increasing the importance of looking ahead in the grading results in higher corresponding average weight. The weights must be thus adjusted for the desired objective.

The weights should be tuned for the desired behaviour in a particular application. As shown in Figure 4.6, different priorities and scoring metrics can yield different average weights. The weights used in further testing are presented in Table 4.5. The evaluation of the path-following performance under the baseline 6D-DWA weights is presented in Section 4.4.

Table 4.5: 6D-DWA Baseline weights

w_{goal}	w_{head}	w_{path}	w_{look}
40	30	20	10

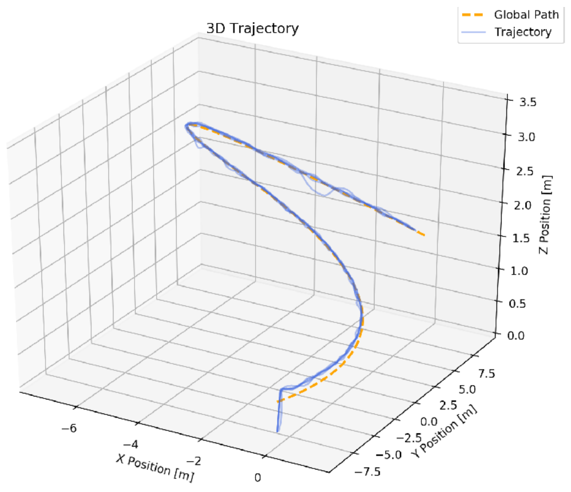
4.4. Path Following Evaluation

The following section evaluates the path following of the core 6D-DWA using the baseline parameters, as defined in Appendix A. To assess the ability of the planner to command the omnicopter through the empty testing space, two distinct testing scenarios are established, as shown in Table 4.6. The test scenario 4.1 is performed to determine the path-following performance of the 6D-DWA given a waypoint-dense global path plan, such as the paths output by the global RRT* path planning algorithm. The second testing scenario 4.2 investigates the behaviour of the planner in scenarios with limited waypoint information.

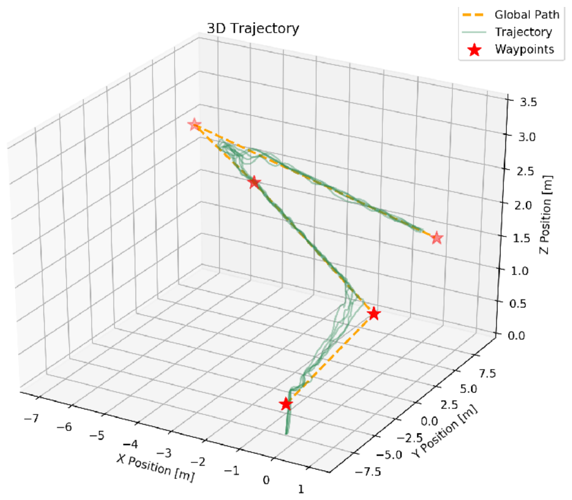
Table 4.6: Path following evaluation testing scenarios

Test:	Testing Scenario	Trials per configuration
4.1	Dense global path waypoints	5
4.2	Sparse global path waypoints (5 waypoints)	5

The global path, as well as the resultant trajectories are shown in Figure 4.7.



(a) Trajectories given the dense global path



(b) Trajectories given the 5-waypoint sparse global path

Figure 4.7: Resulting trajectories for a curved global path; The 6D-DWA is able to command the omnicopter through free space for both RRT*-like global path definition and sparse waypoints.

As seen in Figure 4.7, given a waypoint-dense global path the omnicopter accurately follows the path. The omnicopter manages to maintain an average cross-track distance to the global path of under 0.1 m. The average orientation error is approximately $13°$, and the average lookahead error is approximately $10°$. The orientation direction and lookahead direction can differ to varying degree. The difference is dependent on the shape of the path, waypoint poses, omnicopter position, as well as the waypoint horizons. The parameters used in the tests are shown in Appendix A.

To maintain the logic of lookahead and local-goal waypoints, the planner linearly interpolates the path, ensuring that sufficient waypoint density is provided. Upon the path densification, the planner is able to command the omnicopter through the test space. The resulting path following errors are calculated with respect to the dense global path. The sparse-path following results in an average cross-track distance to the global path of approximately 0.55 m. The average orientation error is approximately $27°$, and the average lookahead error is approximately $40°$. The errors are significantly greater, which is to be expected given the path estimation through interpolation. The total covered distance is however lower than for the dense global path trajectories.

The evaluated implementation of 6D-DWA is designed to be integrated with the global path planner. If feasible, the global path is assumed to be the optimal path through the environment. As shown in Figure 4.7, the planner turns prematurely and undershoots the global waypoints. This behaviour is a direct result of the local goal selection methodology, as explained in Section 3.2.5. The waypoint horizon was introduced to smoothen the 6D-DWA trajectories, which can be seen in the resulting trajectories. The waypoint undershooting can be mitigated by lowering the waypoint horizon, at the cost of trajectory smoothness.

The 6D-DWA with the baseline scoring function weights is able to facilitate highly accurate global path following. The RRT*-like waypoints are utilised in determining favourable trajectories. If the global path is sparsely defined, the planner is able to maintain its waypoint logic by linearly interpolating the provided waypoints.

4.5. Static Obstacle Avoidance Evaluation

The following section assesses the detection and avoidance of mapped static obstacles. The goal is to evaluate that the omnicopter is able to traverse an environment with mapped static obstacles without collisions. The obstacles in the testing space are defined in the STL map but do not possess defined collision in the Gazebo world. The poses of the omnicopter are thus limited not by the physical obstacles, but solely by the planner's understanding of the environment based on the defined STL map. This allows for the assessment of the collision based on the logged locations of the body-approximation spheres. The testing scenarios used in the assessment of the static collision avoidance are presented in Table 4.7. The number of samples has been decreased from the baseline value to ensure no performance degradation, following the analysis in Section 4.1.

Table 4.7: Testing scenarios for the evaluation of static obstacle avoidance

Test	Testing scenario	Trials per config.	Number of samples	w_{head}
5.1	Path fully blocked by a mapped static obstacle	5	1000	30
5.2	Path-obstructing mapped static obstacle	5	1000	30
5.3	Static mapped obstacles requiring orientation deviation	5	2000	15, 30

In the testing scenario, a static wall is positioned obstructing the path. As seen in Figure 4.8, the planner correctly does not command positions that would result in collision. As the planner fails to find a feasible collision-free command that would progress the omnicopter along the global path, the planner eventually transitions into the 'Stuck' state, as explained in Section 3.4.1.

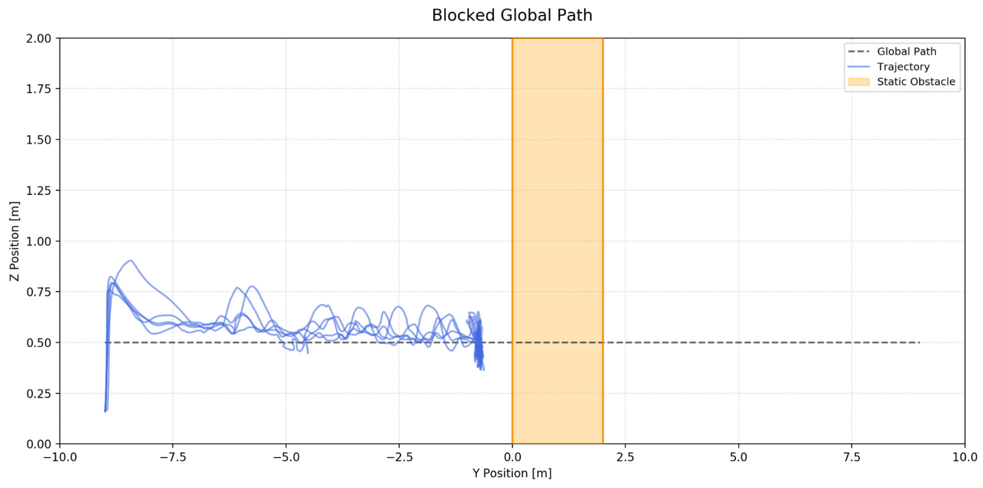


Figure 4.8: The side view of trajectories resultant from testing scenario 5.1; The planner correctly recognises an infeasible path and does not command the omnicopter along the global path based on the static map definition.

The testing scenario 5.2 is shown in the Figure 4.9. A path-obstructing obstacle is defined in the static map, necessitating a deviation from the global path. As seen in Figure 4.9, all trajectories successfully detected infeasibility of the global path and continued progressing along the path at a collision-free distance. Beyond the static obstacle, at 'y' positions greater than 3.0 m, the omnicopter correctly returns to the feasible global path.

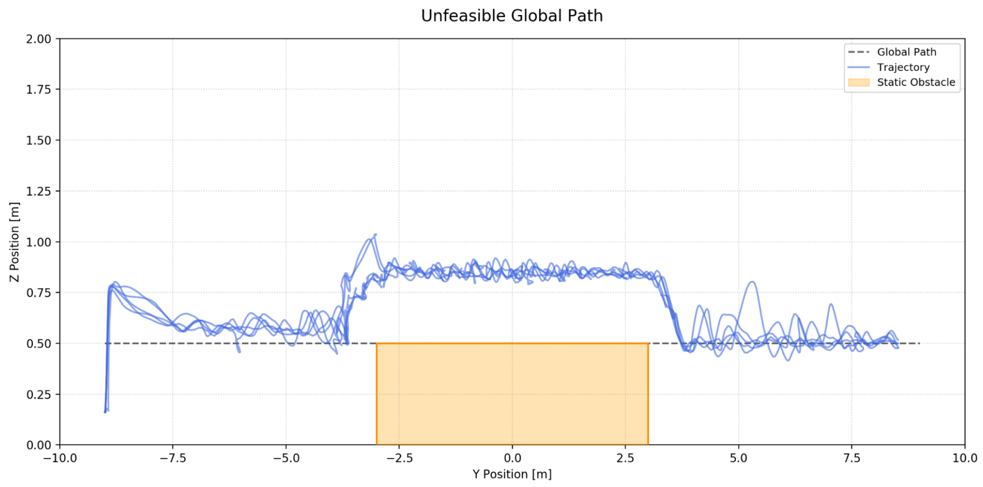
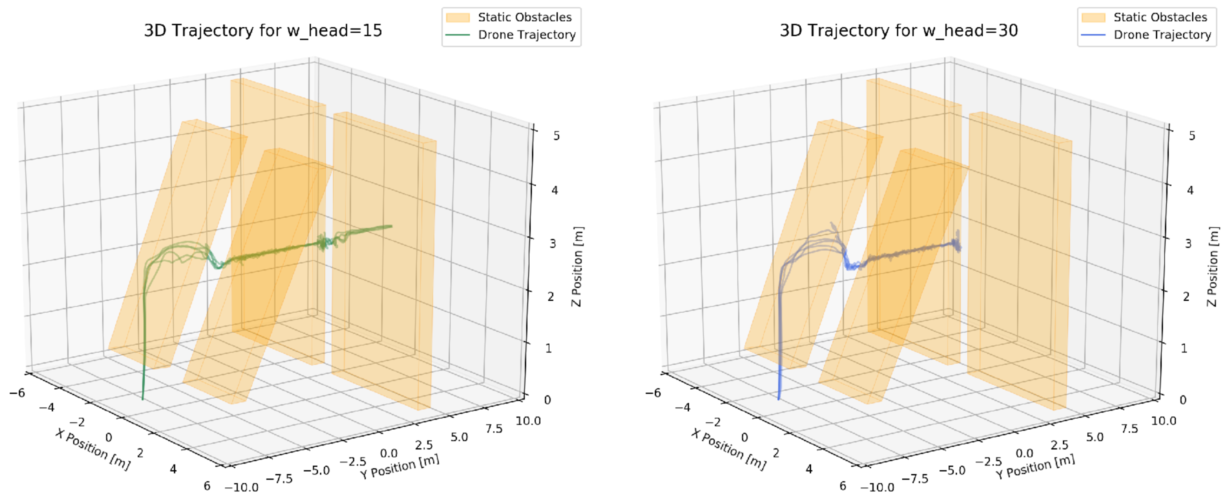


Figure 4.9: The side view of trajectories resultant from testing scenario 5.2; The omnicopter correctly deviates from the global path when the path becomes infeasible.

The following test 5.3 provides insight into the planner's performance given an infeasible global path plan requiring orientation diversion. Two static obstacle gaps of 1.3 m width are defined in the STL map, as seen in Figure 4.10. The first gap is at a $45°$ angle, the second gap is at a $90°$ angle with respect to the global waypoint orientations.



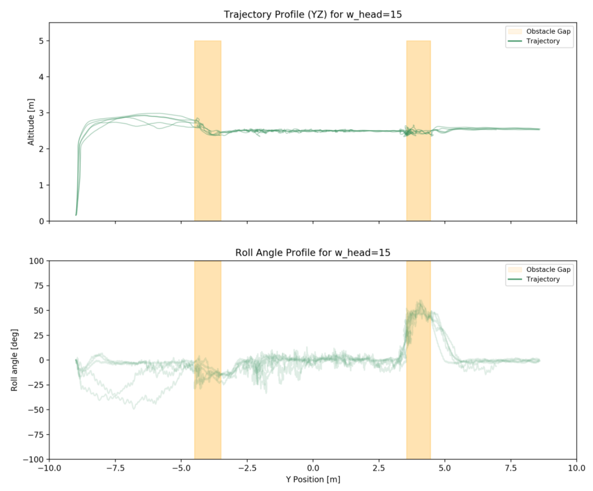
(a) Trajectories for the w_{head} of 15

(b) Trajectories for the w_{head} of 30

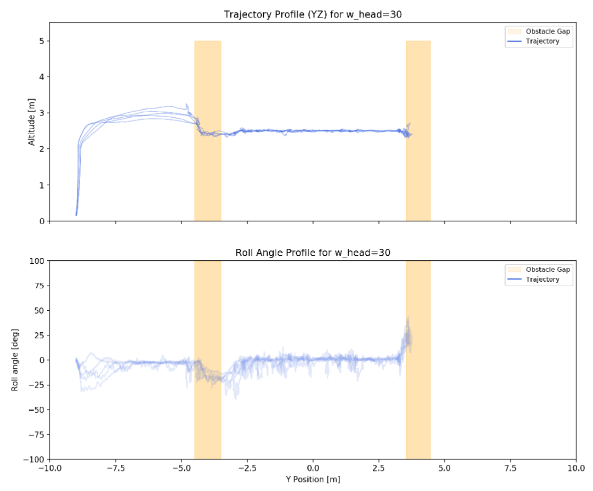
Figure 4.10: Trajectories resulting from testing scenario 5.3; In all trials for w_{head} of 15, the omnicopter successfully reaches the global goal. In all trials for w_{head} of 30, none of the trials result in global path completion. The w_{head} constrains the possible deviation from the global waypoint orientations.

As seen in Figure 4.10, all configurations manage to command the omnicopter through the $45°$ gap. The trials with the baseline w_{head} of 30 fail to reach the end goal, terminating at the $90°$ gap. Lowering the w_{head} results in the planner being more likely to select trajectories diverting from the waypoint orientations. The w_{head} constrains the possible orientation deviation by favouring adherence to global waypoint orientations. All trials with the lower w_{head} configurations manage to command the omnicopter through the $90°$ gap.

The altitude and roll angle of the resultant trajectories are plotted in Figure 4.11. For the w_{head} of 15, the planner correctly diverts from the waypoint orientation to enable collision-free traversal of the gaps. The used gap width exceeds the geometrical minimum for the given parameters. The roll angle thus does not need to reach exactly $45°$ and $90°$ for the corresponding gaps.



(a) Altitude and roll angle for the trajectories with a w_{head} value of 15



(b) Altitude and roll angle for the trajectories with a w_{head} value of 30

Figure 4.11: Altitude and roll angle variation during gap traversal; For higher w_{goal} the planner fails to command orientations sufficiently deviating from the waypoint-defined $0°$.

The proximity of each sphere centre for the configuration of w_{goal} of 15 is determined for when the centre falls within the defined obstacle gap. The proximity graph is shown in Figure 4.12. The spheres are consistently farther than the sphere radius of 0.15 m, resulting in no collision. It is to be however noted, that for the used inflated radius of 0.35 m the planner should only command positions farther than 0.235 m from the obstacle surface, following the logic explained in Section 3.2.3. As seen in Figure 4.12, the sphere centres exhibit closer proximity than the maximum allowed command distance. This is likely due to the controller overshooting the commanded position. Consequently, an inflation radius greater than the theoretical geometric minimum has to be used to ensure collision-free traversal.

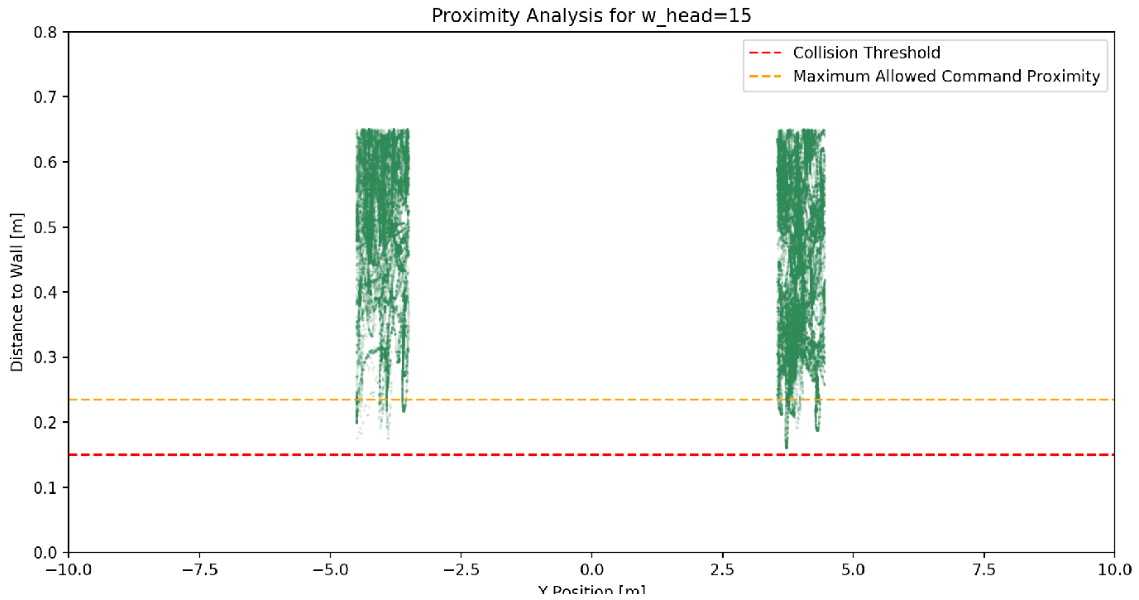


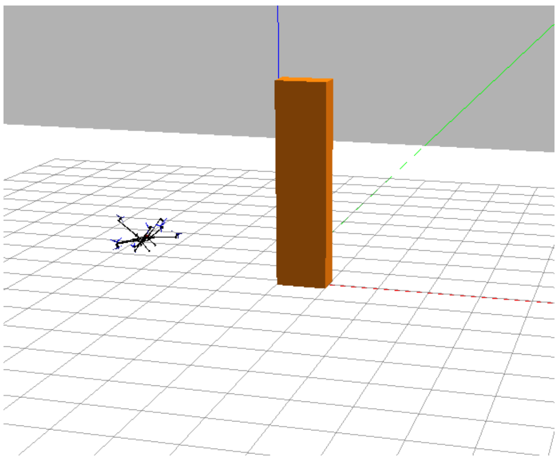
Figure 4.12: Proximity to obstacles during gap traversal for w_{head} of 15; The sphere centres occasionally exhibit closer proximity to the obstacle surface than the maximum allowed command distance, necessitating a safety margin.

The implemented environment simplification and omnicopter's structure approximation facilitated the avoidance of mapped static environments. The ability to deviate from the defined global path definition depends on the relative magnitude of the corresponding adherence metrics' weights, necessitating a trade-off between close global path adherence and deviation capabilities.

4.6. 6D-DWA Agile Mode Weight Tuning

To address the robustness of obstacle avoidance (RQ2), this section evaluates the planner's ability to avoid static unknown obstacles. This section entails the evaluation of the implemented unknown obstacle detection strategy, as well as the weight tuning of the Agile Mode weights, as described in Section 3.3.2.

The testing pipeline comprises two distinct testing scenario environments. In both scenarios, the mapped environment consists of only the STL-defined floor. The unknown cuboid is placed to interfere with the global path. As the cuboid is farther than the detection threshold from a static mapped obstacle, it should be recognised by the planner as an unknown obstacle. In the first scenario, the unknown obstacle is adjacent to the global path, requiring diversion from the global path. In the second scenario, the obstacle is placed centrally on the global path. The visualisation of the testing environment is presented in Figure 4.13.


(a) Visualisation of the unknown obstacle detection testing scenario

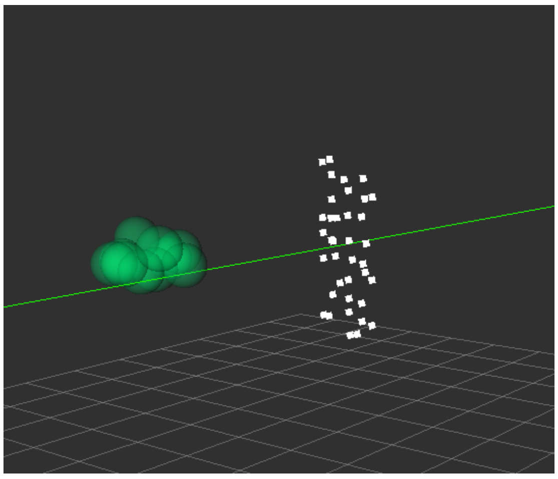

(b) RViz visualisation of the detected unknown obstacle point clusters

Figure 4.13: Gazebo visualisation of the Agile Mode testing scenario; As the detected obstacle is sufficiently far from mapped obstacles, the planner classifies it as an unknown obstacle.

The relevant parameters for the Agile Mode testing are presented in Table 4.8. The $w_{\text{path}}^{\text{agile}}$ is set to zero, as adherence to the global path is not desired during the diversion manoeuvre. The variations of $w_{\text{clear}}^{\text{agile}}$, $w_{\text{face}}^{\text{agile}}$ and $w_{\text{head}}^{\text{agile}}$ are applied and evaluated for their impact on the shape of generated trajectories and the collision-free path completion 'success' rate.

Table 4.8: Agile Mode testing parameters

Test:	Testing Scenario	Number of trials per configuration	$w_{\text{clear}}^{\text{agile}}$	$w_{\text{face}}^{\text{agile}}$	$w_{\text{path}}^{\text{agile}}$	$w_{\text{head}}^{\text{agile}}$
6.1	Off-centre unknown obstacle	3	1,2,3,4,5	10,20,30	0	0,10
6.2	Centred unknown obstacle	3	1,2,3,4,5	10,20,30	0	0,10

The resulting trajectories of test 6.1 are plotted in Figure 4.14. All trials resulted in collision-free traversal of the space. In all trials, the unknown obstacle was correctly identified resulting in the activation of the Agile Mode, causing repulsion away from the obstacle. The success rate decreases with the $w_{\text{clear}}^{\text{agile}}$, from 72.2% at $w_{\text{clear}}^{\text{agile}}$ of 1.0 to 0% at $w_{\text{clear}}^{\text{agile}}$ of 5.0. The trials with higher $w_{\text{clear}}^{\text{agile}}$ result in trajectories farther away from the obstacle. Increasing the $w_{\text{clear}}^{\text{agile}}$ eventually forbids progressing along the global path. This is particularly evident in testing scenario 6.2, as seen in Figure 4.15. For all configurations with successful runs, the success rate is higher for $w_{\text{head}}^{\text{agile}}$ of 0.0. Considering only the trials with $w_{\text{clear}}^{\text{agile}}$ of 1.0 and $w_{\text{head}}^{\text{agile}}$ of 0.0 , the success rate rises to 77.8%. For the $w_{\text{clear}}^{\text{agile}}$ of 3.0, all successful trials occur under no goal heading incentive. The $C_{\text{head}}^{\text{agile}}$ soft-constrains the possible orientations, limiting the exploration ability of the planner.

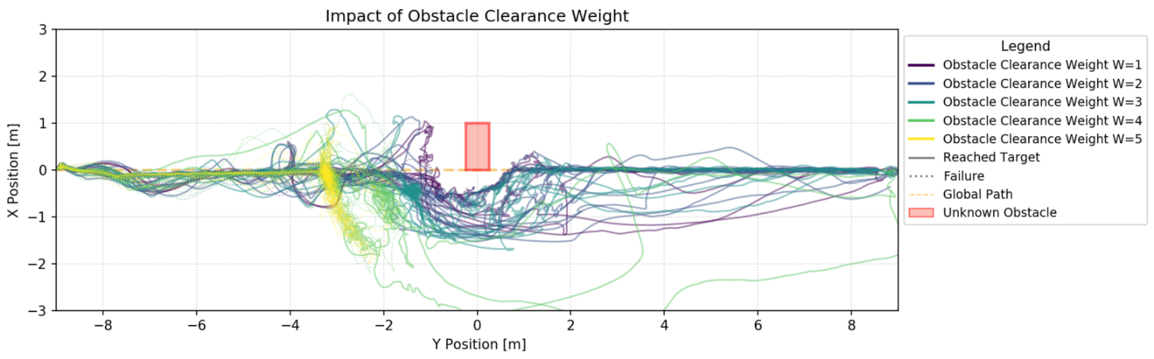


Figure 4.14: The impact of unknown obstacle clearance weight $w_{\text{clear}}^{\text{agile}}$ on trajectory with an off-centre placed unknown obstacle; The higher $w_{\text{clear}}^{\text{agile}}$, the higher the incentive to command trajectories away from the unknown obstacle

The resulting trajectories of the testing scenario 6.2 are plotted in Figure 4.15. All trials resulted in collision-free traversal of the testing space. Similarly to testing scenario 6.1, higher $w_{\text{clear}}^{\text{agile}}$ result in higher distance from the unknown obstacle. The only weight allowing for finding a path around the unknown obstacle is $w_{\text{clear}}^{\text{agile}}$ of 1.0, higher weights result in too strong of a repulsive field around the obstacle. In such cases, the planner favours remaining on the global path at a distance from the unknown obstacle, resulting in eventual transition into the 'Stuck' operational state. Consistent with the results of scenario 6.1, the only successful trials occur for $w_{\text{head}}^{\text{agile}}$ of 0.0. The evaluation of the behaviour under $w_{\text{clear}}^{\text{agile}}$ of 1.0 and $w_{\text{head}}^{\text{agile}}$ of 0.0 is enhanced by the additional testing scenarios 6.3 and 6.4.

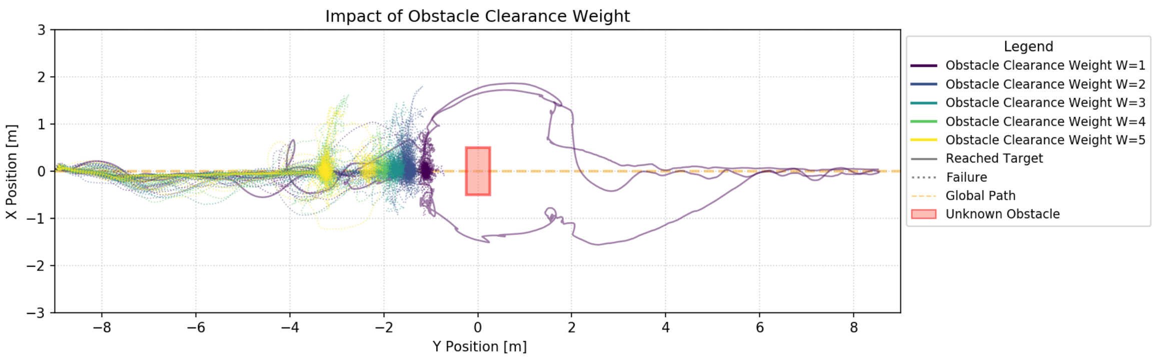


Figure 4.15: The effect of unknown obstacle clearance weight $w_{\text{clear}}^{\text{agile}}$ on the trajectory with a centrally placed unknown obstacle; High $w_{\text{clear}}^{\text{agile}}$ forbids exploration close to the centrally-placed obstacle.

To further evaluate the performance, additional trials were performed with a set $w_{\text{clear}}^{\text{agile}}$ of 1.0 and $w_{\text{head}}^{\text{agile}}$ of 0.0. The used parameters for the tests are presented in Table 4.9. The testing data of 6.3 and 6.4 is combined for analysis with the trials of the same $w_{\text{clear}}^{\text{agile}}$ from the testing scenarios 6.1 and 6.2 respectively.

The trajectories resulting from the combined trials are presented in Figure 4.16. The testing of configurations with the $w_{\text{clear}}^{\text{agile}}$ of 1.0 with an off-centre placed unknown obstacle yielded a success rate of 79.3%.

Table 4.9: Agile Mode with $w_{\text{clear}}^{\text{agile}}$ of 1.0 testing parameters

Test:	**Testing Scenario**	**Trials per configuration**	$w_{\text{clear}}^{\text{agile}}$	$w_{\text{face}}^{\text{agile}}$	$w_{\text{path}}^{\text{agile}}$	$w_{\text{head}}^{\text{agile}}$
6.3	Off-centre unknown obstacle	10	1	10,30	0	0
6.4	Centred unknown obstacle	10	1	10,30	0	0

The success rate could potentially be higher given higher testing time. Unsuccessful runs were terminated as a result of either the detection of lack of progress ('Stuck' operational state) or testing timeout of 200 s. Given a higher testing timeout it is possible that more trials could eventually become successful.

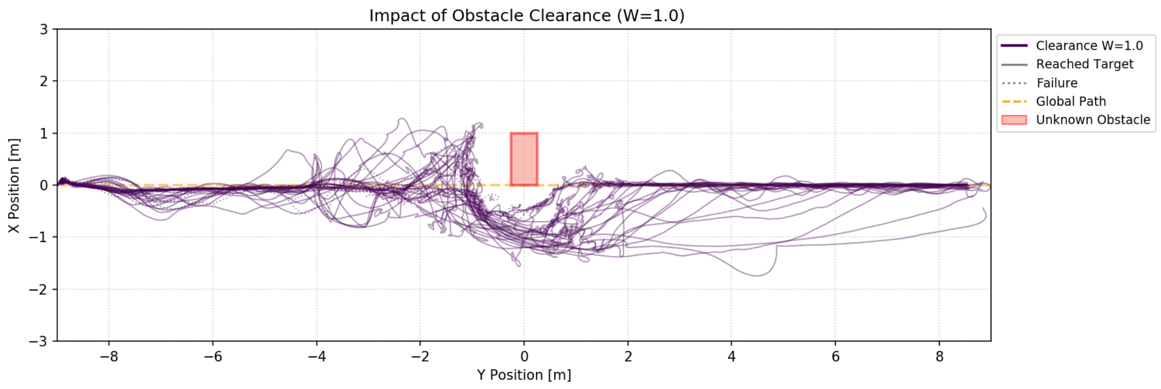


Figure 4.16: The trajectories for $w_{\text{clear}}^{\text{agile}}$ of 1.0 with an off-centre placed unknown obstacle; The success rate is 79.3%.

The trajectories of the trials with the centred unknown obstacles are plotted in Figure 4.17. The central placement of the obstacle resulted in a significantly lower success rate of 41.4%. The difference between the success rates demonstrates that centred obstacles pose a significantly higher risk of entrapment in local minima, as the repulsive forces act against the goal-seeking incentive. Notably, the majority of the successful trials result in diversion manoeuvres in the positive x coordinates. There is no algorithm incentive to favour poses with positive x coordinates. This tendency can be a result of the common starting orientation of the omnicopter. The 6D-DWA is initiated pointing towards the positive x-axis for all trials. The trials are likely to follow a similar pattern of overshooting the global path and initiating the diversion manoeuvres at poses with similar offsets from the global path. Due to the calculation of the simulated repulsive force, as described in Section 3.3, the diversion manoeuvre starting pose is likely to determine the direction of the diversion.

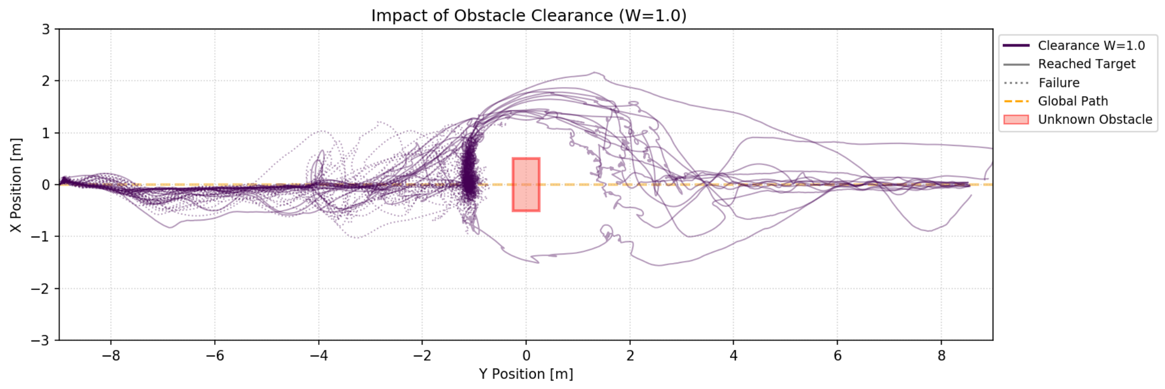


Figure 4.17: The trajectories for $w^{\text{agile}}_{\text{clear}}$ of 1.0 with a centred unknown obstacle; The success rate is 41.4%, significantly lower than for the off-centre obstacle.

As seen in Figure 4.18, the variation in the facing weight $w^{\text{agile}}_{\text{face}}$ does not impact the ability to avoid collision with the unknown obstacle. The facing weight $w^{\text{agile}}_{\text{face}}$ of 10 is sufficient to forcing the camera pointing towards the unknown obstacle throughout the diversion manoeuvre. The facing weight does, however, impact how soon the omnicopter transitions from the diversion manoeuvre in Agile Mode back to the default 6D-DWA parameters. As the obstacle is being passed, the Agile Mode 6D-DWA begins to have conflicting goals of the lookahead heading cost $C^{\text{agile}}_{\text{look}}$ and the unknown obstacle facing cost $C^{\text{agile}}_{\text{face}}$. As the 'looking ahead' is to be favoured, the camera starts pointing away from the obstacle. The omnicopter eventually no longer perceives the unknown obstacle and exits the Agile Mode. The change of focus occurs sooner when the $C^{\text{agile}}_{\text{face}}$ has less of an impact on the 6D-DWA scoring function. As seen in Figure 4.18, the lower $w^{\text{agile}}_{\text{face}}$ of 10 results in convergence onto the global path earlier than for trajectories with $w^{\text{agile}}_{\text{face}}$ of 30.

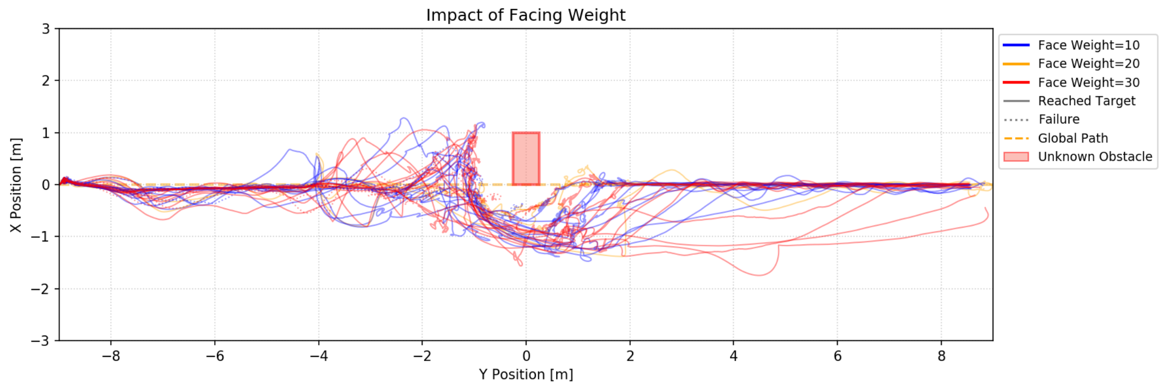


Figure 4.18: The impact of unknown obstacle facing weight $w^{\text{agile}}_{\text{face}}$ on the trajectory with an off-centre placed unknown obstacle; The $w^{\text{agile}}_{\text{face}}$ has no significant impact on the success rate. Lower $w^{\text{agile}}_{\text{face}}$ results in faster convergence to the global path.

The introduction of context-aware weight adjustment allows for unknown obstacle avoidance, facilitating close global path following and necessary path deviation using the different weight sets. The limited foresight of the 6D-DWA significantly decreases the planner's ability to command the omnicopter around centrally placed unknown obstacles. The replanning performance is sensitive to initial conditions. The reliance on fixed threshold for unknown obstacle recognition, while computationally efficient, limits the capability to recognise unknown obstacles and thus dynamic environment adaptability.

4.7. Evasion Evaluation

The following section evaluates the fast evasion strategy implemented to mitigate the 6D-DWA's foresight limitations, aiming to answer the research question RQ2. The used testing scenarios are described in Table 4.10. The evasions are triggered by simulated threat messages, with a simulated point-mass moving obstacle.

Table 4.10: Testing scenarios for moving obstacle evasion evaluation.

Test	Testing Scenario	Trials per Config.	Threat Offset [m]	Body Approximation
7.1	Evasion of an offset threat	5	0.5, 1.5	10-sphere
7.2	Evasion of an offset threat with static obstacles	5	0.1	1-sphere, 10-sphere
7.3	Evasion of an offset threat during 6D-DWA path following	5	0.1	10-sphere

The testing scenario 7.1 evaluates the evasion triggering logic. Two threat messages are generated. The threat message with an offset of 0.5 m results in a threat path going through the established in Section 3.3.3 approximation sphere. The second threat message with an offset of 1.5 m results in a threat path not passing through the sphere. The 1.5 m offset messages correctly yield no reaction from the path planner. The moving obstacle is determined to not cause collision with the omnicopter. The threat message with an offset of 0.5 m results in evasion manoeuvre for all trials, as seen in Figure 4.19. The planner correctly estimates the desired direction of evasion, based on the location of the closest point on the estimated threat path to the omnicopter's centre.

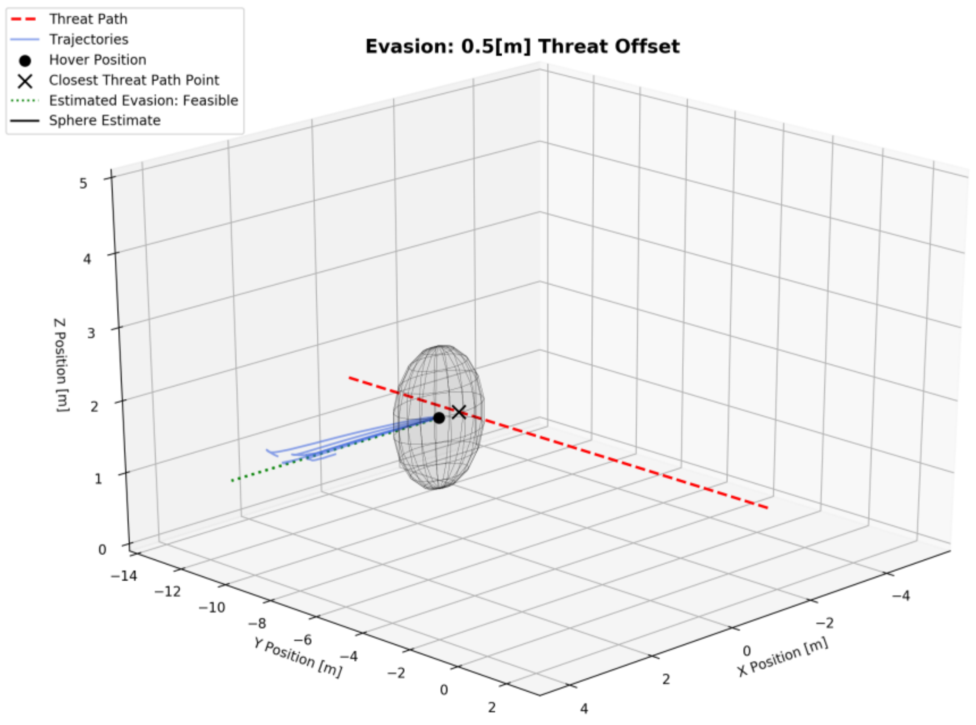


Figure 4.19: Evasion caused by a threat offset by 0.5 m; The evasion logic correctly identifies the optimal evasion direction based on the simulated threat message.

In testing scenario 7.2, the threat message uses an offset triggering an infeasible evasion direction, as presented in Figure 4.20. As the first evasion direction is not possible, next evasions directions are evaluated, according to the evasion logic explained in Section 3.3.3. As seen in Figure 4.20, each of the trials results in a collision-free evasion.

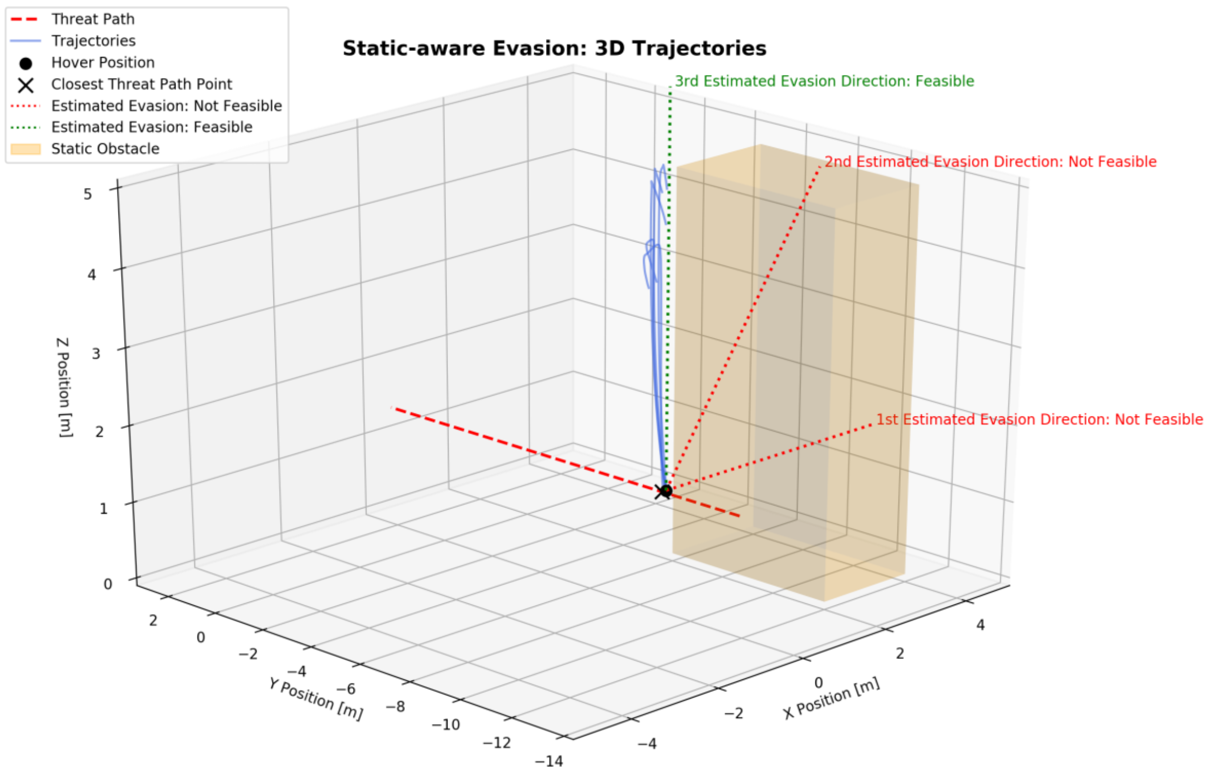


Figure 4.20: Evasion manoeuvre in the presence of static obstacles; The planner correctly commands the first feasible evasion direction found.

The 10-sphere approximation is used for the evasion. To showcase the computational feasibility of the static-aware evasion, 1-sphere approximation encapsulating the entire omnicopter geometry was additionally tested. The resulting evasion manoeuvre are compared in Figure 4.21. Notably, there is no significant difference between the two evaluations. The negligible performance difference between the 1-sphere and 10-sphere models in Figure 4.21 indicates that the KD-Tree-based collision checking is highly optimised. The reaction time is thus more dependent on the hardware limitations and the controller latency.

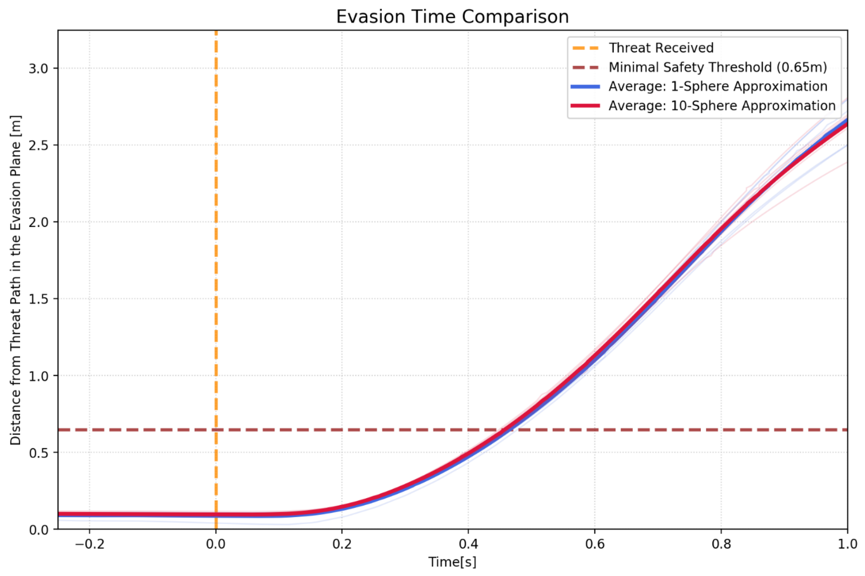


Figure 4.21: Distance from the simulated threat path vs time for different body approximations; The more accurate 10-sphere approximation exhibits similar reaction time to the 1-sphere approximation.

The last testing scenario evaluates the integration of the fast evasion and 6D-DWA. The evasion is triggered during 6D-DWA straight path following. As shown in the Figure 4.22, all trials resulted in successful evasion manoeuvres, upon which the omnicopter transitioned back into 6D-DWA and continued along the global path. The state machine is implemented as intended.

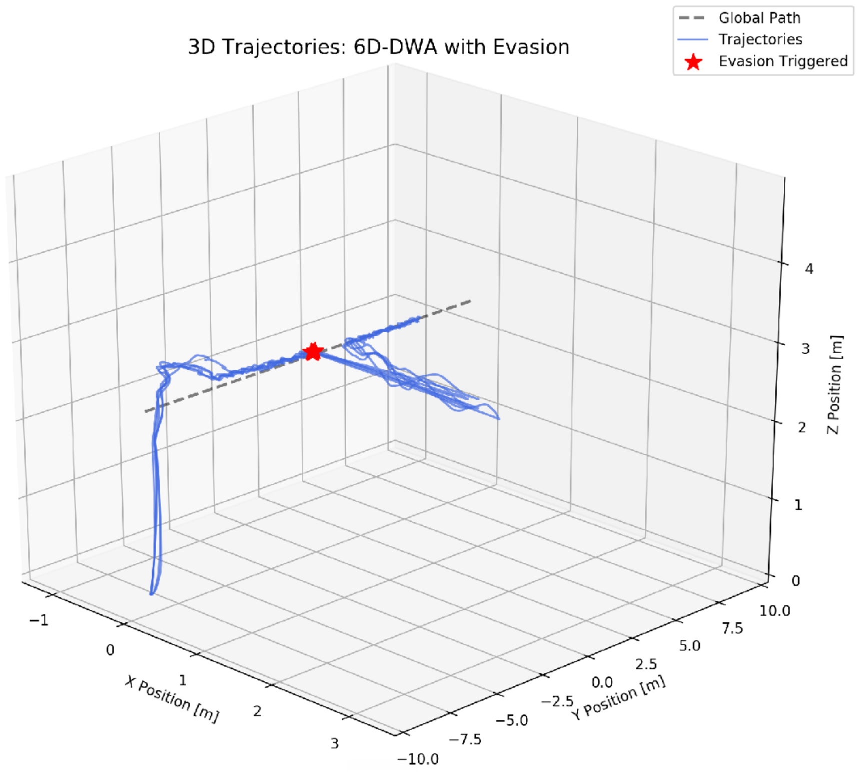


Figure 4.22: 3D trajectories for the 6D-DWA path following with an induced evasion manoeuvre; The planner correctly transitions from the 6D-DWA to the evasion state, upon which completion the 6D-DWA path following resumes.

The fast evasion strategy attains high computational efficiency by leveraging the same environment simplification and trajectory propagation methods as the core 6D-DWA. The evasion logic facilitates the accurate identification of collision threats and the determination of optimal, collision-safe evasion directions. Furthermore, the state machine ensures that safety-critical evasion manoeuvres are prioritised over standard path following. Notably, while the current evaluation relies on simulated messages, the integration of the fast-moving obstacle detection pipeline is required for hardware deployment. Its subsequent analysis would define the operational constraints of the system, specifically the required time-to-impact and velocity thresholds necessary for successful avoidance in real-world scenarios.

5 Conclusions & Recommendations

The following chapter contains the conclusions and recommendations resulting from the implementation and testing of the autonomous local path planning algorithm.

5.1. Conclusions

Regarding RQ1, the 6D-DWA local path planning algorithm has been proven feasible in facilitating the 6D movement of the omnicopter. The local path planner is integrated with the offline global path planner, facilitating environment-aware global path following. The DWA-based 6D-DWA allows for real-time local path planning, performing consistently below the 0.2 s threshold. The implementation of adaptive sampling significantly improved the performance of the planner, allowing for minimising the required number of velocity samples in the 6D search space. The local path-planner performs collision evaluation for each examined velocity sample. The computational feasibility of collision checking is achieved through space voxelisation and omnicopter's geometry sphere approximation. The planner augments the path-following abilities by ensuring that the collisions will not occur for infeasible global paths.

In response to the research question RQ2, the planner can facilitate replanning based on the sensing input. The depth-camera sensor data was successfully processed and integrated with the static map, facilitating unknown obstacle identification and static avoidance enhancement. The 6D-DWA allows for high adaptability to the desired behaviour. The weighted score function allows for the inclusion of advantageous trajectory evaluation metrics, such as imposing a soft forward facing constraint. The implementation of context-aware 6D-DWA demonstrated the ability of the 6D-DWA to achieve different behaviour characteristics through variation of the 6D-DWA weights. The context-aware adjustment of the weights provided the ability to closely follow the global path, as well as divert from the path given the detection of unmapped obstacles.

The 6D-DWA implementation imposes certain limitations on the system. The computational limitations result in the need for voxelisation of the search space, as well as body-approximation. Those approximation methods, as well as the controller overshoot, limit the minimal gap width able to be traversed. Additionally, 6D-DWA is not suitable for fast-moving obstacle avoidance. A separate processing pipeline is suggested with the implemented static-obstacle-aware evasion strategy. The integration of the two pipelines is successfully achieved through a state machine. The finite state machine serves as a safety layer by accounting for the local planner's limitations. The 6D-DWA utilises a single depth camera setup, limiting the amount of information available to the planner. In order to facilitate a fully omnidirectional movement, full field of view should be analysed in the sensing processing.

Notably, this 6D-DWA implementation suggests a computationally-feasible DWA-based framework. The algorithm must be adapted for the intended real-world application. The behaviour of the planner depends on the defined scoring function, which should be revised for the defined goals of a particular implementation. For example, for applications such as surveillance, motion smoothness can be critical. A relevant metric should thus be devised and implemented within the algorithm. Conflicting objectives may limit the planner's capabilities. The inclusion of multiple metrics allows for inclusion of different objectives, at the cost of increased computational complexity.

5.2. Recommendations

Several recommendations for further research were derived from the implementation evaluation. Some of the limitations of the current implementations can be mitigated by expanding the algorithm. The 6D-DWA has limited exploration foresight, limiting the capability to find trajectories leading out of local minima obstacle entrapments. An additional more computationally expensive algorithm could be integrated, activating when the omnicopter enters hover due to lack of path-progressing trajectories. Such further hybridisation could enhance the replanning capabilities of the system.

A beneficial strategy can be further expanding on the context-aware 6D-DWA. The inclusion of additional metrics can facilitate achieving different desired planner behaviours. The autonomous identification of other context scenarios would allow for additional weight adjustment and thus potentially performance improvement. Such a context scenario could be the identification of gap traversal. Upon identification, the waypoint orientation adherence weight could be minimised to prioritise global path progression. Furthermore, the implementation of context-classification algorithms, such as neural networks, could significantly enhance the context identification capabilities.

Notably, the implementation assumes all STL-defined obstacles are actually present in order to avoid collision. A potential feature providing the ability to identify whether a ’known’ obstacle is actually in the sensed environment, effectively updating the static map, would expand the replanning ability of the planner. The current implementation is limited by the sensing capabilities. In order to ensure full awareness of the omnicopter’s surroundings, the integration of data from multiple depth cameras should be investigated. The depth-camera based detection of the fast-moving obstacles should be integrated into the system. Lastly, real-world experiments would provide great insight into actual real-time performance of the planner.

References

[1] S. Ghambari, M. Golabi, L. Jourdan, J. Lepagnot, and L. Idoumghar. “UAV Path Planning Techniques: A Survey”. In: *RAIRO - Operations Research* 58 (2024), pp. 2951–2989. DOI: 10.1051/ro/2024073.

[2] M. H. Sabour, P. Jafary, and S. Nematiyan. “Applications and Classifications of Unmanned Aerial Vehicles: A Literature Review with Focus on Multi-rotors”. In: *The Aeronautical Journal* 127.1309 (2023), pp. 466–490. DOI: 10.1017/aer.2022.75.

[3] R. Rashad, J. Goerres, R. Aarts, J. B. C. Engelen, and S. Stramigioli. “Fully Actuated Multirotor UAVs: A Literature Review”. In: *IEEE Robotics & Automation Magazine* 27.3 (2020), pp. 97–107. DOI: 10.1109/MRA.2019.2955964.

[4] F. Ruggiero and V. Lippiello. “Aerial Manipulation: A Literature Review”. In: *IEEE Robotics and Automation Letters* 3 (July 2018), pp. 1957–1964. DOI: 10.1109/LRA.2018.2808541.

[5] D. Debnath, F. Vanegas, J. Sandino, A. F. Hawary, and F. Gonzalez. “A Review of UAV Path-Planning Algorithms and Obstacle Avoidance Methods for Remote Sensing Applications”. In: *Remote Sensing* 16.21 (2024). DOI: 10.3390/rs16214019.

[6] E. Balestrieri, P. Daponte, L. De Vito, F. Picariello, and I. Tudosa. “Sensors and Measurements for UAV Safety: An Overview”. In: *Sensors* 21.24 (2021). DOI: 10.3390/s21248253.

[7] H. Yahia and A. Mohammed. “Path Planning Optimization in Unmanned Aerial Vehicles Using Meta-Heuristic Algorithms: A Systematic Review”. In: *Environmental Monitoring and Assessment* 195 (Oct. 2022). DOI: 10.1007/s10661-022-10590-y.

[8] A. Ait Saadi, A. Soukane, Y. Meraihi, A. Benmessaoud Gabis, S. Mirjalili, and A. Ramdane-Cherif. “UAV Path Planning Using Optimization Approaches: A Survey”. In: *Archives of Computational Methods in Engineering* 29.6 (2022), pp. 4233–4284. DOI: 10.1007/s11831-022-09742-7.

[9] A. M. Ali, M. Hamandi, and A. Tzes. “Efficient Safe Trajectory Planning for an Omnidirectional Drone”. In: *2025 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2025, pp. 785–792. DOI: 10.1109/ICUAS65942.2025.11007909.

[10] D. Fox, W. Burgard, and S. Thrun. “The Dynamic Window Approach to Collision Avoidance”. In: *IEEE Robotics & Automation Magazine* 4 (Apr. 1997), pp. 23–33. DOI: 10.1109/100.580977.

[11] N. Koenig and A. Howard. “Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Sendai, Japan, Sept. 2004, pp. 2149–2154.

[12] L. K. Jakobsen, J. Kristine Bang Gram, A. J. Grabmayr, A. Højen, A. M. Hansen, M. Rostgaard-Knudsen, A. Claesson, and F. Folke. “Semi-autonomous Drone Delivering Automated External Defibrillators for Real Out-of-hospital Cardiac Arrest: A Danish Feasibility Study”. In: *Resuscitation* 208 (2025), p. 110544. DOI: 10.1016/j.resuscitation.2025.110544.

[13] M. Amami, A. El-Turki, A. Rustum, I. El-Amaari, and T. Jabir. “Topographic Surveying using Low-Cost Amateur Drones & 4K Ultra-High-Definition Videos”. In: *Open Access Research Journal of Science and Technology* 4 (Apr. 2022), pp. 72–082. DOI: 10.53022/oarjst.2022.4.2.0040.

[14] Y. Lu. “Mastering Complexity: Amazon’s Innovative Approach to Global Supply Chain Challenges”. In: *Proceedings of the 2024 8th International Seminar on Education, Management and Social Sciences (ISEMSS 2024)*. Atlantis Press, 2024, pp. 752–762. DOI: 10.2991/978-2-38476-297-2_92.

[15] R. G. Sangeetha, Y. Srivastava, C. Hemanth, H. Sankar Naicker, A. P. Kumar, and S. Vidhyadharan. “Unmanned Aerial Surveillance and Tracking System in Forest Areas for Poachers and Wildlife”. In: *IEEE Access* 12 (2024), pp. 187572–187586. DOI: `10.1109/ACCESS.2024.3514941`.

[16] R. Parmar. “Decryption and Design of a Multicopter Unmanned Aerial Vehicle (UAV) for Heavy Lift Agricultural Operations”. In: Mar. 2021, pp. 189–221. DOI: `10.1002/9781119769231.ch10`.

[17] J. Scherer, S. Yahyanejad, S. Hayat, E. Yanmaz, T. Andre, A. Khan, V. Vukadinovic, C. Bettstetter, H. Hellwagner, and B. Rinner. “An Autonomous Multi-UAV System for Search and Rescue”. In: *Proceedings of the First Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use*. DroNet ’15. Florence, Italy: Association for Computing Machinery, 2015, pp. 33–38. DOI: `10.1145/2750675.2750683`.

[18] J. Peksa and D. Mamchur. “A Review on the State of the Art in Copter Drones and Flight Control Systems”. In: *Sensors* 24 (May 2024), p. 3349. DOI: `10.3390/s24113349`.

[19] A. Keipour, M. Mousaei, A. T. Ashley, and S. Scherer. *Integration of Fully-Actuated Multirotors into Real-World Applications*. 2021. DOI: `10.48550/arXiv.2011.06666`.

[20] E. Masehian and M. R. Amin-Naseri. “A Voronoi Diagram-visibility Graph-potential Field Compound Algorithm for Robot Path Planning”. In: *J. Field Robotics* 21 (June 2004), pp. 275–300. DOI: `10.1002/rob.20014`.

[21] M. Kloetzer, C. Mahulea, and R. Gonzalez. “Optimizing cell decomposition path planning for mobile robots using different metrics”. In: (Oct. 2015), pp. 565–570. DOI: `10.1109/ICSTCC.2015.7321353`.

[22] S. M. LaValle. “Combinatorial Motion Planning”. In: *Planning Algorithms*. Ed. by Steven M. LaValle. Chapter 6 – Combinatorial Motion Planning. Cambridge, UK: Cambridge University Press, 2006, pp. 206–256. DOI: `10.1017/CBO9780511546877.008`.

[23] A. Elfes. “Using Occupancy Grids for Mobile Robot Perception and Navigation”. In: *Computer* 22 (July 1989), pp. 46–57. DOI: `10.1109/2.30720`.

[24] Y. Xu, X. Tong, and U. Stilla. “Voxel-based Representation of 3D Point Clouds: Methods, Applications, and Its Potential Use in the Construction Industry”. In: *Automation in Construction* 126 (2021), p. 103675. DOI: `https://doi.org/10.1016/j.autcon.2021.103675`.

[25] S. Gutmann, M. Fukuchi, and M. Fujita. “A Floor and Obstacle Height Map for 3D Navigation of a Humanoid Robot.” In: Jan. 2005, pp. 1066–1071. DOI: `10.1109/ROBOT.2005.1570257`.

[26] M. Al-Ibadi, J. Abdul-Jabbar, and M. Alwan. “A New Hardware Architecture for Parallel Shortest Path Searching Processor Based-on FPGA Technology”. In: *International Journal of Electronics and Computer Science Engineering (IJECSE)* 1 (Nov. 2012), pp. 2572–2582.

[27] M. Wayahdi, S. Ginting, and D. Syahputra. “Greedy, A-Star, and Dijkstra’s Algorithms in Finding Shortest Path”. In: *International Journal of Advances in Data and Information Systems* 2 (Feb. 2021), pp. 45–52. DOI: `10.25008/ijadis.v2i1.1206`.

[28] M. Radmanesh, M. Kumar, P. Guentert, and M. Sarim. “Overview of Path Planning and Obstacle Avoidance Algorithms for UAVs: A Comparative Study”. In: *Unmanned Systems* 6 (Apr. 2018), pp. 1–24. DOI: `10.1142/S2301385018400022`.

[29] A. Woods and H. La. “Dynamic Target Tracking and Obstacle Avoidance using a Drone”. In: Dec. 2015. DOI: `10.1007/978-3-319-27857-5_76`.

[30] R. Geraerts and M. H. Overmars. “A Comparative Study of Probabilistic Roadmap Planners”. In: *Algorithmic Foundations of Robotics V*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 43–57. DOI: `10.1007/978-3-540-45058-0_4`.

[31] M. Kothari and I. Postlethwaite. “A Probabilistically Robust Path Planning Algorithm for UAVs Using Rapidly-Exploring Random Trees”. In: *Journal of Intelligent & Robotic Systems* 71.2 (2013), pp. 231–253. DOI: `10.1007/s10846-012-9776-4`.

[32] J. Nasir, F. Islam, and Y. Ayaz. “Adaptive Rapidly-Exploring-Random-Tree-Star (RRT*) -Smart: Algorithm Characteristics and Behavior Analysis in Complex Environments”. In: *Asia-Pacific Journal*

of Information Technology and Multimedia 02 (Dec. 2013), pp. 39–51. DOI: 10.17576/apjitm-2013-0202-04.

[33] J. Borenstein and Y. Koren. “Real-Time Obstacle Avoidance for Fast Mobile Robots”. In: *IEEE Transactions on Systems, Man and Cybernetics* 19 (Oct. 1989), pp. 1179–1187. DOI: 10.1109/21.44033.

[34] H. Suwoyo, A. Adriansyah, J. Andika, A. Ubaidillah, and M. F. Zakaria. “An Integrated RRT*SMART-A* Algorithm for solving the Global Path Planning Problem in a Static Environment”. In: *IIUM Engineering Journal* 24 (Jan. 2023), pp. 269–284. DOI: 10.31436/iiumej.v24i1.2529.

[35] Y. Zhao, Z. Zheng, and Y. Liu. “Survey on Computational-intelligence-based UAV Path Planning”. In: *Knowledge-Based Systems* 158 (2018), pp. 54–64. DOI: https://doi.org/10.1016/j.knosys.2018.05.033.

[36] S. Agrawal, B. Patle, and S. Sanap. “Path Planning of UAV in an Uncertain Static Environment Using Probability Fuzzy Logic”. In: Sept. 2023, pp. 97–104. DOI: 10.1109/ICUIS60567.2023.00025.

[37] H. Chang and T. Jin. “Command Fusion Based Fuzzy Controller Design for Moving Obstacle Avoidance of Mobile Robot”. In: *Future Information Communication Technology and Applications: ICFICE 2013*. Dordrecht: Springer Netherlands, 2013, pp. 905–913. DOI: 10.1007/978-94-007-6516-0_99.

[38] R.A. Rutenbar. “Simulated Annealing Algorithms: An Overview”. In: *IEEE Circuits and Devices Magazine* 5.1 (1989), pp. 19–26. DOI: 10.1109/101.17235.

[39] A. Lambora, K. Gupta, and K. Chopra. “Genetic Algorithm - A Literature Review”. In: *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*. 2019, pp. 380–384. DOI: 10.1109/COMITCon.2019.8862255.

[40] S. Das and P. N. Suganthan. “Differential Evolution: A Survey of the State-of-the-Art”. In: *IEEE Transactions on Evolutionary Computation* 15.1 (2011), pp. 4–31. DOI: 10.1109/TEVC.2010.2059031.

[41] D. Fajri, T. H. Saragih, A. Hamdianah, W. Mahmudy, and Y. P. Anggodo. “Optimized Fuzzy Neural Network for Jatropha Curcas Plant Disease Identification”. In: Nov. 2017. DOI: 10.1109/SIET.2017.8304152.

[42] M. Albadr, S. Tiun, M. Ayob, and F. Al-Dhief. “Genetic Algorithm Based on Natural Selection Theory for Optimization Problems”. In: *Symmetry* 12 (Oct. 2020), pp. 1–31. DOI: 10.3390/sym12111758.

[43] Y. He, T. Hou, and M. Wang. “A New Method for Unmanned Aerial Vehicle Path Planning in Complex Environments”. In: *Scientific Reports* 14.1 (2024), p. 9257. DOI: 10.1038/s41598-024-60051-4.

[44] J. Bes, J. Dendarieta, L. Riazuelo, and L. Montano. “DWA-3D: A Reactive Planner for Robust and Efficient Autonomous UAV Navigation in Confined Environments”. In: *Robotics and Autonomous Systems* 195 (2026), p. 105196. DOI: https://doi.org/10.1016/j.robot.2025.105196.

[45] W. Chen, N. Wang, X. Liu, and C. Yang. “VFH* Based Local Path Planning for Mobile Robot”. In: Sept. 2019, pp. 18–23. DOI: 10.1109/CCHI.2019.8901916.

[46] Y. Liu, Z. Zheng, F. Qin, X. Zhang, and H. Yao. “A Residual Convolutional Neural Network Based Approach for Real-time Path Planning”. In: *Knowledge-Based Systems* 242 (2022), p. 108400. DOI: https://doi.org/10.1016/j.knosys.2022.108400.

[47] Zhengyang Cui and Yong Wang. “UAV Path Planning Based on Multi-Layer Reinforcement Learning Technique”. In: *IEEE Access* 9 (2021), pp. 59486–59497. DOI: 10.1109/ACCESS.2021.3073704.

[48] B. Thibodeau, S. Hart, D. Karuppiah, J. Sweeney, and O. Brock. “Cascaded Filter Approach to Multi-objective Control.” In: vol. 4. May 2004, pp. 3877–3882. DOI: 10.1109/ROBOT.2004.1308872.

[49] J. Borenstein and Y. Koren. “The Vector Field Histogram - Fast Obstacle Avoidance For Mobile Robots”. In: *IEEE Transactions on Robotics and Automation* 7 (July 1991), pp. 278–288. DOI: 10.1109/70.88137.

[50] L. Heng, L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys. “Autonomous Obstacle Avoidance and Maneuvering on a Vision-guided MAV Using On-board Processing”. In: June 2011, pp. 2472–2477. DOI: 10.1109/ICRA.2011.5980095.

[51] S. Wu, S. Decker, P. Chang, T. Camus, and J. Eledath. “Collision Sensing by Stereo Vision and Radar Sensor Fusion”. In: *IEEE Transactions on Intelligent Transportation Systems* 10.4 (2009), pp. 606–614. DOI: 10.1109/TITS.2009.2032769.

[52] Y. Watanabe, C. Lesire, A. Piquereau, P. Fabiani, M. Sanfourche, and G. Le Besnerais. “The ONERA ReSSAC Unmanned Autonomous Helicopter : Visual Air-to-Ground Target Tracking in an Urban Environment”. In: *Annual Forum Proceedings - AHS International* 2 (Jan. 2010).

[53] Faying Li, J. Xiao, W. Huang, and S. Cai. “Research on the Intelligent Obstacle Avoidance and Path Planning Strategy of UAV based on Multi-Sensor Fusion”. In: *2022 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*. 2022, pp. 628–632. DOI: 10.1109/AEECA55500.2022.9919008.

[54] G. Welch and G. Bishop. *An Introduction to the Kalman Filter*. Tech. rep. TR 95-041. University of North Carolina at Chapel Hill, 2006.

[55] A. Gupta and X. Fernando. “Simultaneous Localization and Mapping (SLAM) and Data Fusion in Unmanned Aerial Vehicles: Recent Advances and Challenges”. In: *Drones* 6.4 (2022). DOI: 10.3390/drones6040085.

[56] S. Back, G. Cho, J. Oh, X. Tran, and H. Oh. “Autonomous UAV Trail Navigation with Obstacle Avoidance Using Deep Neural Networks”. In: *Journal of Intelligent & Robotic Systems* 100.3 (2020), pp. 1195–1211. DOI: 10.1007/s10846-020-01254-5.

[57] G. Tu and J. Juang. “UAV Path Planning and Obstacle Avoidance Based on Reinforcement Learning in 3D Environments”. In: *Actuators* 12.2 (2023). DOI: 10.3390/act12020057.

[58] M. Hamandi, A. M. Ali, K. Kyriakopoulos, A. Tzes, and F. Khorrami. “An Omnidirectional Non-Tethered Aerial Prototype with Fixed Uni-Directional Thrusters”. In: *2025 IEEE International Conference on Robotics and Automation (ICRA)*. 2025, pp. 8649–8655. DOI: 10.1109/ICRA55743.2025.11128060.

[59] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, B. Wheeler, and A. Ng. “ROS: An Open-Source Robot Operating System”. In: vol. 3. Jan. 2009.

[60] Gazebo Tutorials. *ROS Depth Camera Integration*. https://classic.gazebosim.org/tutorials?tut=ros_depth_camera. Accessed: 2025-12-21. 2014.

[61] J. Solà. “Quaternion Kinematics for the Error-state Kalman Filter”. In: *arXiv preprint arXiv:1711.02508* (2017).

[62] O. Khatib. “Real-time Obstacle Avoidance for Manipulators and Mobile Robots”. In: *Proceedings. 1985 IEEE International Conference on Robotics and Automation*. Vol. 2. 1985, pp. 500–505. DOI: 10.1109/ROBOT.1985.1087247.

[63] R. B. Rusu and S. Cousins. “3D Is Here: Point Cloud Library (PCL)”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China: IEEE, May 2011.

[64] G. Guennebaud, B. Jacob, et al. *Eigen v3: C++ Template Library for Linear Algebra*. https://libeigen.gitlab.io. Accessed: 2026-01-25. 2010.

A

Appendix A: Baseline Parameters

This appendix provides the relevant parameters used in the implementation in Table A.1.

Table A.1: Baseline parameters of the 6D-DWA Planner

Parameter	Symbol	Value	Unit	Description
6D-DWA Algorithm				
Simulation Timestep	Δt_{step}	0.2	s	Internal prediction step duration
Prediction Horizon	t_p	0.5	s	Total time horizon for trajectories
Number of Samples	n_{samples}	5000	—	Velocity samples per cycle
Max Linear Velocity	$\mathbf{v}_{\text{limit}}^{\text{max}}$	[0.3, 0.3, 0.3]	m/s	Linear velocity limits $[x, y, z]$
Max Angular Velocity	$\boldsymbol{\omega}_{\text{limit}}^{\text{max}}$	[0.5, 0.5, 0.5]	rad/s	Angular velocity limits $[r, p, y]$
Max Linear Accel.	$\mathbf{a}^{\text{max}}$	[3.0, 3.0, 3.0]	m/s^2	Linear acceleration limits
Max Angular Accel.	$\boldsymbol{\alpha}^{\text{max}}$	[3.0, 3.0, 3.0]	rad/s^2	Angular acceleration limits
Adaptive Sampling Ratios	e_r, f_r, b_r	0.50, 0.25, 0.25	—	Exploration / Focused / Boundary ratios
Focused Std. Dev.	σ	0.10	—	Focused-search standard deviation
Stuck Distance	d_{stuck}	0.5	m	Displacement threshold for ’Stuck’ state
Stuck Timeout	t_{stuck}	50.0	s	Time period before ’Stuck’ state activation
6D-DWA Cost Function Weights (Baseline / Agile)				
Goal Distance Weight	w_{goal}	40.0 / 40.0	—	Weight for distance to the local goal
Path Distance Weight	w_{path}	20.0 / 0.0	—	Weight for the cross-track error
Goal Heading Weight	w_{head}	30.0 / 30.0	—	Weight for waypoint orientation deviation
Lookahead Weight	w_{look}	10.0 / 20.0	—	Reward for camera/path alignment
Clearance Weight	w_{clear}	0.0 / 1.0	—	Unknown obstacle repulsive weight
Obstacle Facing Weight	w_{face}	0.0 / 10.0	—	Incentive to face unknown obstacles
Perception & Evasion				
Sphere Radius	r	0.15	m	Radius of the 10-sphere approximation
Static Inflation Radius	r_{inf}	0.35	m	Sphere inflation for collision avoidance
Local Map Radius	r_{local}	1.5	m	Radius of the local static map
Voxel Size (Static)	—	0.15	m	Resolution for global STL map
Voxel Size (Dynamic)	—	0.4	m	Resolution for depth point cloud
Detection Range	—	0.3 – 3.5	m	Dynamic obstacle sensing range
Unknown Threshold	—	0.4	m	Unknown obstacle classification distance
TTI Threshold	t_{tti}	2.0	s	Time-to-impact trigger for evasion
Evasion Velocity	v_{evade}	5.0	m/s	Velocity used in the evasion manoeuvre