University of Tartu

Faculty of Science and Technology

Institute of Technology

Kwasi Akuamoah Boateng

Risk-Aware Planning on Point Clouds

Master's thesis (30 ECTS)
Robotics and Computer Engineering

Supervisors:

Associate Prof. Arun Kumar Singh
Basant Sharma

Tartu 2025

Abstract

Risk-Aware Planning on Point Clouds

Navigating complex environments safely is a critical challenge for autonomous drones. This thesis introduces a novel risk-aware planning framework that empowers drones to make smarter, safer decisions. At its core, the framework utilizes an innovative ensemble of neural networks (integrating PointNet, a point cloud processing network, and Gaussian policy-based Multi-Layer Perceptron (MLP) structures) to deliver probabilistic predictions of obstacle distances and their associated uncertainties. Coupled with a jerk-controlled trajectory model, the system leverages Conditional Value-at-Risk (CVaR) and a cross-entropy optimization method to intelligently quantify and mitigate risky trajectories. This allows the drone to confidently navigate by optimally balancing mission objectives against this principled risk measure. Simulation results demonstrate the effectiveness of the proposed framework.

CERCS: T125 Automation, robotics, control engineering; P170 Computer science, numerical analysis, systems, control

Keywords: risk-aware planning, probabilistic robotics, neural networks, trajectory optimization, uncertainty quantification, autonomous navigation, obstacle avoidance, jerk-controlled model, cross-entropy optimization, motion primitives, simulation.

Resümee

Riskiteadlik Planeerimine Punktipilvedel

Ohutu navigeerimine keerukates keskkondades on autonoomsete droonide jaoks oluline väljakutse. Lõputöö tutvustab uudset riskiteadlikku planeerimise raamistikku, mis võimaldab droonidel teha arukamaid ja ohutumaid otsuseid. Raamistiku tuum hõlmab uuenduslike närvivõrkude ansambel mudelit (koosneb PointNetist, mis on punktipilvede töötlemise võrk, ja Gaussi jaotusel põhinevatest mitmekihilistest tajuritest (MLP)), et pakkuda tõenäosuslikke ennustusi takistuste kauguste ja nendega seotud määramatuste kohta. Koos kiirenduse muutumise juhitud trajektoori mudeliga, kasutab süsteem tingimuslikku riski väärtust (CVaR) ja rist-entroopia optimeerimise meetodit riskantsete trajektooride intelligentseks kvantifitseerimiseks ja leevendamiseks. See võimaldab droonil enesekindlalt navigeerida, tasakaalustades optimaalselt missiooni eesmärke selle põhimõttelise riskimõõdikuga. Simulatsiooni tulemused kinnitavad esitatud raamistiku tõhusust.

CERCS: T125 Automatiseerimine, robootika, control engineering; P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Märksõnad: riskiteadlik planeerimine, trajektoori optimeerimine, määramatuse kvantifitseerimine, äratundmisvõrgud, autonoomne navigeerimine.

Contents

Abstract **2**

Resümee **3**

List of Figures **6**

List of Tables **7**

List of Abbreviations **8**

1 Introduction **10**
1.1 Background . . . 10
1.2 Problem Statement . . . 10
1.3 Aim and Contributions . . . 11
1.4 Organization of Thesis . . . 11

2 Literature Review **13**
2.1 Risk Assessment in Robotics . . . 13
2.1.1 Worst-Case . . . 13
2.1.2 Expected Cost . . . 13
2.1.3 Mean–Variance . . . 14
2.1.4 Chance Constraints . . . 14
2.1.5 Value-at-Risk (VaR) . . . 15
2.1.6 Conditional Value-at-Risk (CVaR) . . . 15
2.2 Motion Planning Under Uncertainty . . . 17
2.2.1 Motion Primitives . . . 17
2.2.2 Sampling-Based Planning . . . 17
2.2.3 Optimization-based methods . . . 18
2.3 Neural Network Approaches . . . 19
2.3.1 Neural Networks for Pointcloud processing . . . 19
2.3.2 Uncertainty Estimation in Deep Learning . . . 20
2.3.3 Predicting Collision through Neural Networks . . . 20
2.3.4 End-to-End Learning . . . 21
2.3.5 Hybrid Approaches . . . 23

3 Methodology **24**
3.1 Motion Primitive Library . . . 25
3.2 Uncertainty-Aware Distance Prediction . . . 26

3.3 Risk-Aware Planning Policy 27

4 Implementation **31**
4.1 System Requirements 31
4.1.1 Hardware Configuration 31
4.1.2 Software Environment 31
4.2 Simulation Framework 31
4.2.1 Simulation Environments 32
4.2.2 Drone Simulation Models 32
4.3 Data Collection and Processing 33
4.3.1 Data Collection 33
4.3.2 Data Processing 34
4.4 Model Training 35
4.4.1 Deep Ensemble Strategy 35
4.4.2 Loss Function 35
4.4.3 Hyperparameters and Training Procedure 35

5 Experimental Results & Discussion **37**
5.1 Performance Evaluation of the Distance Prediction Model 37
5.2 Simulation Results 37
5.3 Planner Performance and Resource Consumption 41

6 Conclusion **43**
6.1 Future Work and Improvements 43

Bibliography **46**

Appendices **55**
I. Code Repository and Video Demos 55
II. Planner Parameters 55

Non-exclusive license **56**

List of Figures

2.1 Illustration comparing risk metrics [15]. 16
2.2 PointNet Architecture [46] 19
2.3 The DroNet Architetecture[63] 21
2.4 Comparison of end-to-end deep learning-based approach with the conventional approach. 22

3.1 The Proposed Risk-Aware Planning Framework Architecture. 24
3.2 3D jerk-based motion primitives. We sample a total of 50 dynamically feasible trajectories. 25
3.3 The proposed POINTNET-GaussianMLP architecture for uncertainty-aware distance prediction. 26

4.1 Simulation environments 33

5.1 Drone navigates in a simulated world (env4). 39
5.2 Drone navigates safely while avoiding obstacles to reach the goal. 40
5.3 Illustration of a failure during flight. 41
5.4 Illustration of the resource utilization profile of the planner. 42

List of Tables

4.1 PC Hardware specifications 31
4.2 Intel RealSense D455 depth camera specifications 32

5.1 Comparison of results on the distance prediction model 37
5.2 Comparison of collision counts between deterministic and uncertainty-aware prediction methods within the risk-aware planner framework across various simulation environments. 38
5.3 Statistical summary of the planner resource consumption and performance . . . 42

List of Abbreviations

2D two-dimensional

3D three-dimensional

AUC Area Under the Curve

BNN Bayesian Neural Network

CC-RRT Chance Constrained Rapidly Exploring Random Trees

CEM Cross Entropy Method

CNN Convolutional Neural Network

CPU Central Processing Unit

CVaR Conditional Value-at-Risk

DGCNN Dynamic Graph CNN

ES Expected Shortfall

FOV Field of View

GPU Graphics Processing Unit

JIT Just-In-Time

KPConv Kernel Point Convolution

LiDAR Light Detection and Ranging

MLP Multi-Layer Perceptron

MPC Model Predictive Control

MPL Motion Primitive Library

MPPI Model Predictive Path Integral

NLL Negative Log-Likelihood

NN Neural Network

PI Path Integral

RAM Random Access Memory

RGB Red-Green-Blue

RRT Rapidly Exploring Random Trees

RNN Recurrent Neural Network

ROC Receiver-operating Characteristic Curve

ROS Robot Operating System

SBP Sampling-Based Planning

SITL Software-In-The-Loop

TTC Time to Collision

UAV Unmanned Aerial Vehicle

VaR Value-at-Risk

1 Introduction

1.1 Background

The advent of Unmanned Aerial Vehicles (UAVs), particularly quadrotors, has opened up new possibilities across various sectors such as logistics, surveillance, search and rescue, and environmental monitoring [1, 2]. However, the full potential of these aerial platforms is still limited by challenges in autonomous navigation [3, 4], especially in ensuring robustness and reliability [5]. A crucial element of these platforms is the quadrotor's capacity to sense and react to its surroundings instantaneously [4].

Autonomous quadrotors employ various sensors, such as cameras [1, 6], ultrasonic sensors [4], and radar [7], to gather environmental data. While these sensors provide valuable insights, they often encounter limitations related to range, resolution, and performance across diverse environmental conditions [1, 7]. Consequently, Light Detection and Ranging (LiDAR) systems [3], which produce dense point cloud data, have assumed a pivotal role in advanced robotic perception due to their capacity to furnish a detailed three-dimensional (3D) representation of the environment. This data facilitates precise obstacle detection, terrain mapping, and spatial awareness [8]. Nonetheless, the substantial volume and complexity of point cloud data introduce a significant computational challenge, with real-time processing being essential for timely decision-making to uphold navigation accuracy and ensure flight safety [1, 4].

Autonomous quadrotors conventionally adhere to predetermined trajectories, rendering them vulnerable to unanticipated obstacles and environmental variations [7]. Conventional path-planning algorithms demonstrate limitations in accommodating such dynamic conditions, particularly in scenarios involving mobile obstacles or uncertainties in sensor data [5, 9, 10]. A genuinely robust navigation system must possess the ability to detect and respond to obstacles while concurrently addressing these uncertainties and managing the associated flight safety risks [11, 12].

The challenge intensifies when these drones operate in unknown environments, where prior mapping information is unavailable [13]. In these scenarios, real-time perception and adaptive planning are essential to avoid collisions and navigate safely, further underscoring the need for efficient and effective decision-making.

1.2 Problem Statement

A significant constraint within current autonomous quadrotor navigation systems utilizing point cloud data is the absence of effective mechanisms to quantify and address the impacts of sensor

measurement uncertainties and environmental variations on the trajectories of planned flights [14]. This limitation impedes the formulation of risk-aware trajectories that can adapt to unpredictable obstacles and unforeseen environmental conditions, thereby undermining the safety and reliability of quadrotor operations in real-world settings.

1.3 Aim and Contributions

This thesis seeks to address the challenges mentioned above by developing a risk-aware planning framework for autonomous quadrotors. This system will incorporate point cloud data, robot partial state data, neural networks, optimization-based algorithms and a Conditional Value-at-Risk (CVaR) assessment metric.

To achieve its aim, the thesis makes the following novel contributions:

- It develops a neural network-based action-conditioned collision prediction model for predicting the distribution of the distance to the closest obstacle. That is, for a given action, the neural model predicts the distance to the closest obstacle and the uncertainty around these predictions. To this end, the thesis trains an ensemble of Gaussian Multi-Layer Perceptron (MLP).
- The thesis develops a risk-aware planner for computing actions that are optimal and safe with respect to the distance prediction from the neural collision model. More precisely, the planner uses the distance prediction from the neural collision model for a chosen action to estimate the safety through Conditional Value at Risk (CVaR). It then iteratively modifies the action to bring the collision risk under a particular threshold
- The proposed pipeline is tested in simulation in highly dense environments for a quadrotor navigation task. The thesis particularly highlights the importance of incorporating uncertainty estimates in the neural collision model.

1.4 Organization of Thesis

The thesis is organized into six chapters:

- **In the current chapter**: This chapter elaborates on the motivation, problem statement, aim, and objectives of the thesis.
- **Literature Review**: This chapter reviews related work in risk assessment, motion planning under uncertainty, and relevant neural network approaches for robotics.
- **Methodology**: This chapter explains the core design of the proposed risk-aware planning framework, including the motion primitives, uncertainty-aware distance prediction network, and risk-aware policy.
- **Implementation**: This chapter details the simulation setup, data handling procedures, and model training specifics used in this work.

- **Experimental Results & Discussion**: This chapter presents the evaluation results for the distance prediction model and the overall planning framework in simulation.
- **Conclusion**: This chapter summarizes the key findings, addresses the achievement of objectives, and discusses future work.

2 Literature Review

This chapter introduces related concepts used in autonomous navigation, specifically in the area of obstacle avoidance, risk assessment and motion planning.

2.1 Risk Assessment in Robotics

Risk awareness is a crucial aspect of motion planning for autonomous robots operating in uncertain and dynamic environments, aiming to systematically assess and mitigate potential hazards. The increasing ubiquity of autonomous systems in safety-critical applications like self-driving vehicles and service robots has amplified the need for risk-aware approaches. The choice of metric heavily depends on the specific requirements of the robotic application, the nature of the uncertainties involved, and the acceptable level of risk. The following subsections will define and discuss some of the most commonly used methods of risk assessment in robotics.

2.1.1 Worst-Case

The worst-case method also called "MiniMax" serves as a deterministic risk assessment approach in robotics, evaluating system performance and safety under the most adverse conditions. This approach aims to identify the maximum potential negative impact or the minimum guaranteed performance, irrespective of the probability of encountering such extreme scenarios [15].

Risk is quantified by assessing outcomes in the most challenging situations, such as maximum collision force, longest task completion time, minimum safe distance from obstacles, or maximum execution time for safety-critical algorithms [16]. Ultimately, this method defines the boundaries of safe operational parameters, ensuring robotic system robustness even in highly adverse circumstances.

2.1.2 Expected Cost

Expected cost minimization represents the most straightforward approach to risk assessment and decision-making under uncertainty, aiming to optimize the average performance across all possible scenarios. This approach is mathematically expressed as:

$$\min_{x \in X} \mathbb{E}[f(x, \xi)] \tag{2.1}$$

where x represents the decision variables (e.g., robot trajectory), ξ denotes the random variables capturing uncertainty, and $f(x, \xi)$ is the cost function representing task-relevant metrics.

The cost function can be tailored to represent various factors relevant to the robotic task, such as task completion time, energy consumption, distance travelled, or penalties for task failure. However, expected cost minimization exhibits significant limitations when applied to safety-critical robotic systems. By focusing solely on average performance, it remains insensitive to low-probability, high-consequence events that are particularly relevant in robotic navigation [15]. For point cloud-based planning, where sensor noise and environmental uncertainty can lead to catastrophic failures, expected cost minimization may underestimate critical risks in sparse regions of the environment.

2.1.3 Mean–Variance

In the mean-variance method, risk is quantified by the variance (or standard deviation) of the performance metric being considered. A higher variance indicates a greater spread of possible outcomes, and thus a higher level of uncertainty or risk. The method typically involves formulating an objective function that combines the mean and variance:

$$\rho(Z) = \mathbb{E}[Z] + \lambda \cdot \text{Var}[Z] \tag{2.2}$$

where $\lambda > 0$ is a risk aversion parameter that controls the trade-off between expected outcome and uncertainty.

Sharma *et al.* [17] implemented mean-variance as a risk-aware cost measure within an A* planner to plan safer routes for ground vehicles in various terrains. In their work, the algorithm was incentivised to find paths that not only had a low expected travel cost but also passed through areas where the terrain classification was more certain (low variance). Similarly, in reinforcement learning applications, mean-variance optimizes policies that balance expected returns against their variability [18].

Despite its computational simplicity and popularity, mean-variance has significant limitations as a risk measure in robotics applications. Majumdar and Pavone [15] demonstrate that it does not satisfy several key axioms for coherent risk measures, including monotonicity. This can lead to counterintuitive decision-making where options with uniformly worse outcomes might be preferred solely because they exhibit slightly lower variance. For safety-critical robotics applications, this limitation can result in suboptimal or potentially unsafe decision-making.

2.1.4 Chance Constraints

Chance constraints provide a foundational approach for incorporating risk awareness into planning under uncertainty by limiting the probability of constraint violation. This is typically formulated as

$$P(g(x_i, u_i) \leq 0) \leq \delta \tag{2.3}$$

where $g(x_i, u_i) \leq 0$ represents a nonlinear inequality constraint (e.g., for obstacle avoidance), x_i and u_i are the system states and controls, and δ is the confidence level, representing the maximum allowed probability of constraint violation.

Luders *et al.* [19] introduced Chance Constrained Rapidly Exploring Random Trees (CC-RRT), extending traditional Rapidly Exploring Random Trees (RRT) to handle systems with process noise and uncertain obstacles by formulating probabilistic constraints that limit collision likelihood below specified thresholds. By assuming Gaussian noise, the algorithm converts these probabilistic constraints into deterministic bounds on the conditional mean state, creating a

computationally tractable approach that balances conservatism with risk. Blackmore *et al.* [20] have also explored similar chance-constrained methodologies for optimal path planning around obstacles.

In risk-aware control contexts, chance constraints are often employed to limit the probability of unsafe events [12]. However, recognising certain limitations of this approach (discussed below), researchers have explored alternative risk measures. For instance, Hakobyan *et al.* [21] proposed methods based on CVaR to offer different perspectives on risk management, particularly concerning the severity of potential violations [21]. Despite the development of such alternatives, chance constraints remain a relevant technique for providing basic probabilistic guarantees on safety.

It is worth noting that chance constraints, while intuitively appealing, face limitations: they often yield conservative solutions by treating constraints binarily without considering the magnitude of potential violations [22], and solving chance-constrained problems exactly is generally computationally intractable, necessitating approximation methods [19].

2.1.5 Value-at-Risk (VaR)

Value-at-Risk (VaR) provides a measure of risk by quantifying the potential loss threshold for a given confidence level. In the context of robotics, VaR at confidence level α for a cost random variable Z is formally defined as the $(1-\alpha)$-quantile of that variable:

$$\text{VaR}_\alpha(Z) := \min\{z \mid P[Z > z] \leq \alpha\} \tag{2.4}$$

This can be interpreted as defining a threshold cost z where the probability of the actual cost Z exceeding this value is no more than α.

An example of its application is found in the work by Hunt *et al.* [23], who explored the use of VaR to quantify potential losses over a defined time period and confidence interval in robot swarms operating in hazardous environments. In their agent-based simulations, individual robots calculated VaR in real-time and broadcast alerts to neighbours when their VaR limit was breached, helping the swarm collectively avoid hazardous areas. This minimal communication strategy proved effective in reducing the overall exposure of the swarm to hazards while still allowing individual agents to undertake risk-weighted explorations [23].

Despite such applications, VaR has several notable weaknesses relevant to robotics. A significant limitation is that it provides no information about the magnitude of costs (or losses) that might occur beyond the VaR threshold; it only indicates the probability of exceeding that threshold [15]. This disregard for the tail of the distribution can lead to a false sense of security, as it focuses on a specific quantile and ignores the potential severity of more extreme, albeit less likely, outcomes.

Due to these shortcomings, particularly the disregard for the magnitude of costs beyond the threshold, VaR is often deemed insufficient for ensuring safety in autonomous systems where extreme events can have severe consequences [15]. This has motivated the adoption of more comprehensive risk metrics like CVaR within the robotics community [10, 12, 21].

2.1.6 Conditional Value-at-Risk (CVaR)

Conditional Value-at-Risk (CVaR), also known as Expected Shortfall (ES), provides a more comprehensive risk measure than VaR by considering the magnitude of tail events. Mathematically, for a cost random variable Z and confidence level α, CVaR can be defined as the

expected value of the costs that exceed the VaR threshold, or formally via an integral involving VaR:

$$\text{CVaR}_\alpha(Z) := \frac{1}{\alpha}\int_{1-\alpha}^{1} \text{VaR}_{1-\tau}(Z)d\tau \quad \text{or equivalently} \quad E[Z \mid Z \geq \text{VaR}_\alpha(Z)] \tag{2.5}$$

Intuitively, CVaR measures the expected value of the cost within the worst α portion (the α-tail) of the cost distribution [24]. Unlike VaR, CVaR captures "how bad is bad" by considering the average severity of outcomes in this tail [15].

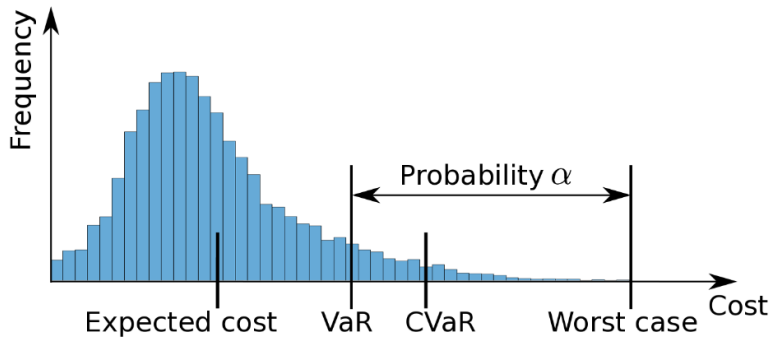


Figure 2.1: Illustration comparing risk metrics [15].

CVaR is considered a 'coherent' risk measure [25], meaning it satisfies properties desirable for rational risk evaluation, including subadditivity and convexity. Subadditivity reflects the principle that diversification (combining risks, e.g., across different parts of a plan or different agents) should not increase the overall assessed risk disproportionately (i.e., $\text{Risk}(A+B) \leq \text{Risk}(A) + \text{Risk}(B)$). Convexity ensures that the risk of a weighted average of potential outcomes is no more than the weighted average of their individual risks. This property is particularly advantageous because it often leads to well-behaved optimization problems; minimizing CVaR or incorporating CVaR constraints typically results in convex optimization problems, which can be solved efficiently and reliably [26]. CVaR has gained favour in recent studies partly due to its beneficial theoretical properties as a coherent and convex risk measure.

In risk-aware motion planning, especially in environments with uncertain or randomly moving obstacles, CVaR is used to minimize the expected magnitude of the worst-case collision risks (or other costs), offering a more robust safety measure than just limiting the probability of collision [27]. This focus means that planning with CVaR aims to reduce the severity of the potential negative outcomes that might occur, rather than just their likelihood.

Bian *et al.* [10] incorporated CVaR constraints into A* path search, introducing a risk-based heuristic that prevents path selection in highly uncertain regions. This approach ensures that the planned path remains robust against unpredictable obstacle movements by considering the expected cost in high-risk scenarios.

Unlike previous studies [10, 21] that applied CVaR to single-agent systems, Yang *et al.* [27] formulated the first analytical integration of CVaR for large-scale robotic swarms. Their method ensures that robot trajectories account for rare but high-risk events by considering the expected cost in the tail of the distribution, making it suitable for autonomous navigation in cluttered spaces.

Incorporating CVaR into robotic applications, while beneficial for safety and robustness, poses computational difficulties, especially in complex, dynamic environments, due to the need for

accurate estimation or approximation of the tail distribution [24, 27, 28]. Nonetheless, the increased safety and robustness it delivers renders it an important asset in risk-aware motion planning.

2.2 Motion Planning Under Uncertainty

2.2.1 Motion Primitives

Motion primitives are a widely used trajectory generation technique in robotics. In quadrotors, these primitives define dynamically feasible trajectories in 3D space. Given the current state of a UAV, the desired end pose, and duration, one can create smooth and optimal MPs by minimizing the jerk [29–31]. These primitives often take the form of polynomial functions of time, $\mathbf{p}(t) = [x(t), y(t), z(t)]^T$, where each component (e.g., $x(t)$) is a polynomial of order n:

$$x(t) = \sum_{i=0}^{n} a_i t^i \tag{2.6}$$

The coefficients a_i are determined by imposing boundary conditions on the state (position, velocity, acceleration, etc.) at the start and end of the primitive's duration. Higher-order polynomials facilitate smoother trajectories, and criteria like minimizing jerk are commonly used for optimization, with a typical cost function being:

$$J = \int_0^T (\dddot{p}(t))^2 dt \tag{2.7}$$

where $\dddot{p}(t)$ represents the jerk.

The effectiveness of motion primitives for risk-aware planning stems from their ability to discretise the continuous action space into segments that can be individually assessed for risk. By evaluating risk metrics for each primitive, a robot can make informed trajectory selection decisions, even with significant perceptual uncertainty [32].

2.2.2 Sampling-Based Planning

Sampling-Based Planning (SBP) algorithms represent a cornerstone of modern robotic navigation, particularly valued for their efficiency in exploring complex, high-dimensional configuration spaces. A key characteristic of these algorithms is their probabilistic completeness, meaning they will eventually find a feasible path, if one exists, given sufficient time [33]. Furthermore, SBPs offer the significant advantage of constructing connectivity roadmaps or trees without requiring an explicit, pre-defined model of the obstacle space. Consequently, in scenarios like open configuration spaces, their primary goal is often to rapidly identify an obstacle-free trajectory, though typically without guarantees on path optimality (See sections 5.5 - 5.6 in [33]).

To operate in uncertain environments, standard sampling-based planners must be augmented to account for risk. Accordingly, popular algorithms like RRT and RRT* [34] have been extended with various risk-aware mechanisms [35, 36]. For instance, chance-constrained RRT variants [19] manage risk by limiting the probability of violating constraints, proving effective when uncertainty distributions are well-characterised. However, a key limitation is their potential difficulty in handling the extreme tails of these distributions—regions often most critical for robust risk-aware planning.

More recently, risk-aware variants of sampling-based planners have integrated coherent risk measures to address this limitation. Safaoui *et al.*[12] incorporated CVaR constraints into a sampling-based framework, enabling the planner to account for not just the likelihood but also the severity of potential collisions. This approach is particularly relevant to point cloud-based planning, where the quality of environmental perception varies significantly across the workspace, leading to heterogeneous uncertainty distributions.

Beyond the choice of risk measure, a significant practical challenge when applying SBPs directly to point cloud data is ensuring computational efficiency [37]. Each sampled state or trajectory will need to be checked against a potentially large point cloud, which can be time-consuming. Our approach tackles this issue by utilizing neural network predictions. This method enables efficient risk evaluation without requiring explicit collision checks for each sample.

2.2.3 Optimization-based methods

Optimization-based methods are central to modern robotics, providing a structured way to determine optimal actions by minimizing costs subject to constraints. Model Predictive Control (MPC) is a key example, where control inputs are continuously optimized over a predictive horizon, enabling systems to handle complex dynamics and constraints [38].

In MPC, approaches to uncertainty management fall into three main categories: deterministic MPC (DMPC), which assumes no noise and lacks robustness to disturbances; robust MPC (RMPC), which handles worst-case scenarios but tends toward conservatism (though remains appropriate for systems requiring guaranteed stability, such as nuclear reactors); and stochastic MPC (SMPC), which employs chance constraints to balance cost-efficiency and robustness. These approaches typically rely on simplifying assumptions like system linearization or convexity [38].

SMPC offers a promising middle ground by balancing performance and safety, but faces significant limitations: its optimization often targets specific noise distributions, compromising robustness when these assumptions are violated; many implementations depend on linear dynamics or system linearization, which introduces computational overhead and restricts accuracy to small operating regions; and the chance constraints commonly employed only bound violation probability without accounting for violation severity or associated costs [39].

To address uncertainty with greater robustness, techniques such as CVaR are integrated into optimization frameworks, as seen in the work of Chu *et al.* [28] and Yin *et al.*[38].

In addition, Path Integral (PI) control has emerged as a compelling framework for stochastic optimal control and trajectory optimization, offering powerful sampling-based solutions, as surveyed by Kazim *et al.* [40]. PI control fundamentally transforms stochastic optimal control problems into the evaluation of path integrals, typically approximated using Monte Carlo methods. A widely adopted variant within this framework is Model Predictive Path Integral (MPPI) control, which is a derivative-free approach that utilizes predictive model-based rollouts in a receding horizon manner, making it well-suited for real-time trajectory generation [38, 40]. Importantly, to improve performance in uncertain environments, the PI control paradigm has been extended to explicitly incorporate risk sensitivity [40]. Collectively, these advanced optimization strategies, leveraging sophisticated risk measures and computationally efficient sampling-based control, are pivotal for enabling robust and reliable decision-making for autonomous systems in complex, real-world scenarios.

2.3 Neural Network Approaches

Neural Networks (NNs) have assumed a pivotal role in the enhancement of autonomous navigation capabilities [7, 41]. In this section, their application within robotic planning is examined, with a focus on fundamental architectures specifically devised for processing 3D point cloud data, methodologies for the quantification of predictive uncertainty, and prevalent system design strategies, encompassing both end-to-end and hybrid approaches.

2.3.1 Neural Networks for Pointcloud processing

Point clouds are a primary data representation for 3D environments, typically generated by sensors like LiDAR or depth cameras [42]. They consist of a collection of data points in 3D space, where each point minimally represents spatial coordinates (x, y, z) and may include additional attributes like colour, intensity, or surface normals [42]. Crucially, unlike images or voxel grids that possess a regular structure, point clouds are fundamentally unordered sets, i.e. the sequence in which points are listed does not alter the geometric information, and they are irregularly distributed in space, lacking the fixed connectivity and consistent local topology inherent in grid-based data formats.

These unique properties of irregularity and permutation invariance make point clouds incompatible with standard deep learning architectures like Convolutional Neural Networks (CNNs), which are designed to operate on regular grids and exploit fixed spatial neighbourhoods through kernel operations and weight sharing [43]. Early approaches attempted to bridge this gap by imposing structure, often converting point clouds into intermediate voxel grids or multi-view two-dimensional (2D) projections amenable to existing CNNs [44, 45]. However, such conversions frequently lead to significant drawbacks, including high computational and memory demands, loss of precision due to discretization (voxelization), or loss of 3D geometric information (multi-view projections) [46].

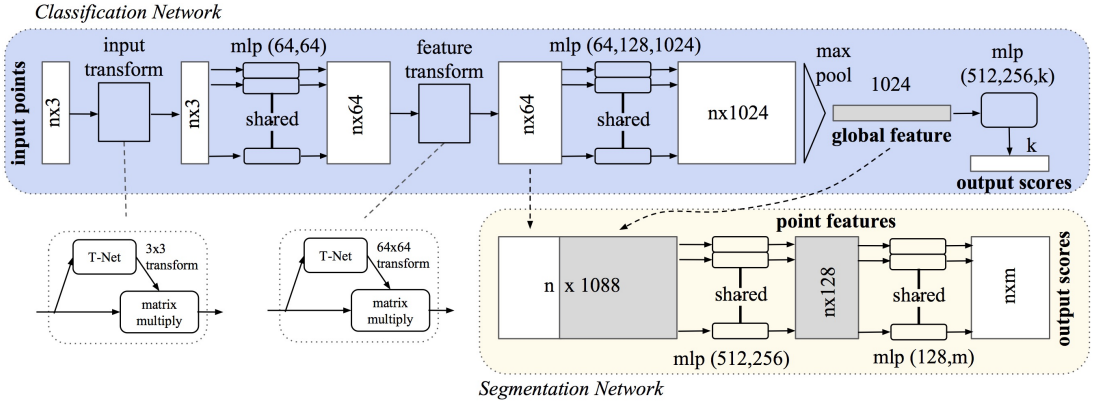


Figure 2.2: PointNet Architecture [46]

The development of PointNet [46] marked a significant advancement by enabling neural networks to process raw point sets directly. PointNet addresses the critical challenge of permutation invariance by applying shared Multi-Layer Perceptrons (MLPs) independently to each input point and then using a symmetric aggregation function, most notably max pooling, to combine point features into a fixed-size global representation [46]. This allows for efficient end-to-end learning on point clouds for tasks such as object classification and segmentation. While foundational, PointNet's reliance on global aggregation limits its ability to capture fine-grained local structures. Subsequent research has focused on architectures that learn richer local

and multi-scale features, including hierarchical networks (e.g., PointNet++ [47]), graph neural networks (e.g., Dynamic Graph CNN (DGCNN) [48]), specialized point-based convolutions (e.g., Kernel Point Convolution (KPConv) [49]), and transformer-based models [50], all aiming to provide more expressive representations for robust 3D perception in robotics.

2.3.2 Uncertainty Estimation in Deep Learning

Bayesian Neural Networks

Bayesian Neural Networks (BNNs) offer a probabilistic framework for modelling uncertainty by treating the network's weights and biases as probability distributions rather than fixed point values. Neal [51] highlights that BNNs can effectively reduce overfitting and estimate epistemic uncertainty through Bayesian inference.

Various studies [52–55] have used BNNs for planning and navigation, perception, control, and decision-making. For instance, Shi *et al.* [55] used a BNN for safe visual servoing in human-reobot interaction. The network's uncertainty estimates helped the robot predict repulsive poses to avoid collisions with human hands while still efficiently completing the visual servoing task.

While BNNs provide a principled approach to modelling epistemic uncertainty, they often come with a higher computational cost and can be more challenging to train compared to their standard counterparts due to the complexities of Bayesian inference in high-dimensional spaces [56, 57].

Deep Ensembles

Deep ensembles are a prominent method for estimating predictive uncertainty in neural networks. The core idea behind deep ensembles is to train multiple independent NNs, typically with the same architecture, but with different random initializations and potentially on different subsets of the training data[56].

Deep ensembles offer several advantages for uncertainty estimation. They are relatively simple to implement compared to other techniques like BNNs, which often require significant modifications to the training procedure [56].

While deep ensembles provide benefits, they also pose specific limitations. One major issue is the significant computational cost associated with training and performing inference with several neural networks [58].

2.3.3 Predicting Collision through Neural Networks

Enhancing safety is the fundamental motivation for collision prediction in autonomous systems. Robots operating in dynamic and complex environments must avoid collisions to ensure their integrity, prevent environmental damage, and guarantee the safety of humans and other agents [59, 60]. The process of detecting collisions, which involves verifying whether a robot's intended pose or movement interferes with its surroundings, frequently accounts for more than 90% of the time spent on motion planning, thus posing a major computational challenge for real-time operations [61, 62]. Neural network-based prediction addresses this challenge by efficiently forecasting whether a given robot configuration will result in collision before performing expensive geometric queries, thereby reducing overall computation [62].

The domain of collision prediction has seen the development of specialised neural architectures focusing on various dimensions of the problem. In contexts involving image-based inputs, convolutional neural networks (CNNs) are prevalent, exemplified by DroNet [63], FCPNet [64] and ORACLE [32]. DroNet, for instance, employs a fast 8-layer residual convolutional neural network (ResNet-8) that processes a single monocular camera image to output both a steering angle for navigation and a collision probability, as illustrated in Figure 2.3. This dual output allows a drone to autonomously navigate complex environments, such as city streets, by continuously adjusting its path and reacting to potential hazards; notably, DroNet is trained primarily using data collected from ground-based vehicles, enabling it to 'learn to fly by driving' and generalize navigation policies to the aerial domain without extensive drone-specific flight data in hazardous environments [63]. In the processing of LiDAR data, approaches generally make use of voxelization techniques or PointNet-based architectures [47] to effectively manage the unstructured nature of point cloud data. The temporal elements of collision prediction are addressed by recurrent architectures; for instance, Fan *et al.* [65] integrated PointNet++ [47] with Recurrent Neural Networks (RNNs) to forecast vehicle motion through sequential point cloud analysis. Numerous cutting-edge systems [66–68] adopt hybrid architectures that exploit the complementary advantages of disparate neural network types, such as CNNs for the extraction of spatial features, RNNs for the modeling of temporal sequences, and specialised modules for sensor fusion.

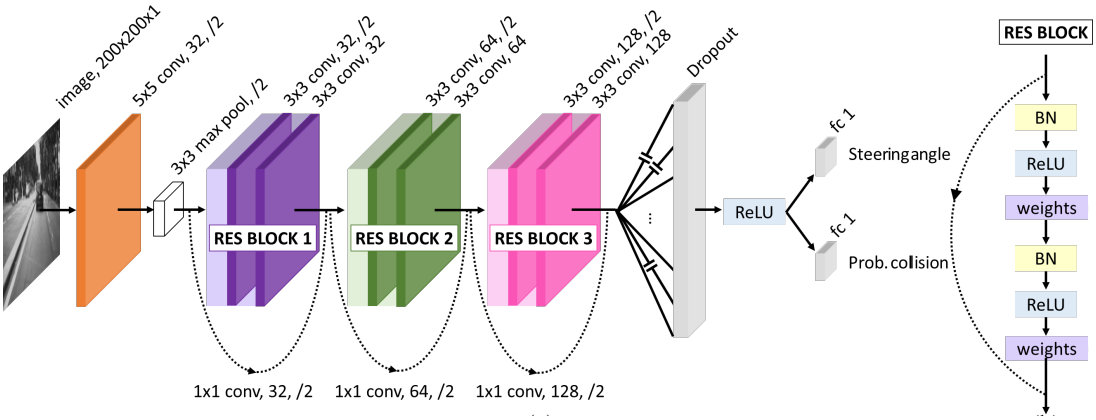


Figure 2.3: The DroNet Architetecture[63]

Integrating these predictive systems with motion planning frameworks offers notable computational benefits while still ensuring safety [52, 59, 68]. By minimizing the need for comprehensive geometric collision checks, neural collision predictors expedite planning cycles, allowing robots to respond more swiftly in complex and dynamic settings [62]. Neural networks for collision prediction are typically assessed using metrics such as Time to Collision (TTC), success rate, collision frequency, Receiver-operating Characteristic Curve (ROC), and Area Under the Curve (AUC). Despite the progress in the state-of-the-art architectures, challenges persist in achieving generalisation to new environments and maintaining reliability amidst sensor noise [69].

2.3.4 End-to-End Learning

End-to-end learning has gained traction as an influential method in robotic motion planning, in which neural networks are trained to create direct links between raw sensory inputs, such as images or LiDAR point clouds, and the resulting robot actions or trajectory plans [70, 71]. Un-

like traditional modular approaches that divide the planning process into several distinct stages, such as perception, mapping, prediction, and control, end-to-end learning bypasses the need for meticulously engineered intermediate representations (see Figure 2.4). This allows for simultaneous optimization of the complete task, which may result in more efficient and robust systems with fewer errors propagating between modules. For example, Yuan *et al.* [72] introduced DRAMA, an end-to-end motion planner for autonomous vehicles that integrates camera, LiDAR, and ego state information to produce a sequence of prospective ego trajectories.

In light of its potential, end-to-end learning in robotics faces several critical drawbacks. Primarily, these methods are notoriously data inefficient, requiring large datasets that are often impractical to acquire in real-world robotic scenarios [73]. Furthermore, the inherent "black box" nature of neural networks hinders interpretability, making it difficult to understand and debug failures [74]. This lack of transparency also raises safety concerns, as the robot's behaviour can become unpredictable in unfamiliar situations [75]. Finally, the challenge of generalizing from limited training data to diverse real-world conditions remains a significant hurdle, often resulting in poor performance when faced with novel environments or tasks [74, 76–78].

End-to-end learning in robotics is hindered by several challenges. It is data inefficient, necessitating large datasets that are difficult to obtain in real-world applications. The "black box" nature of neural networks reduces their interpretability, complicating failure analysis and safety assurance. Additionally, these methods struggle with generalization, often performing poorly in new environments or tasks due to limited training data.

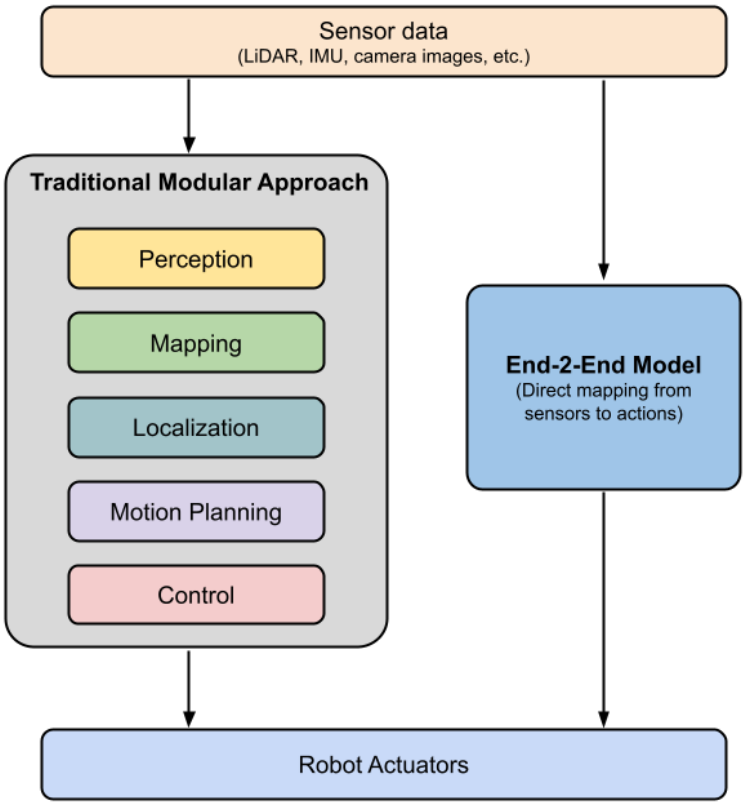


Figure 2.4: Comparison of end-to-end deep learning-based approach with the conventional approach.

2.3.5 Hybrid Approaches

Hybrid approaches that combine the strengths of traditional planning techniques with the learning capabilities of neural networks have gained significant attention in robotics [79]. Traditional planning algorithms, such as A*, RRT, and MPC, often provide guarantees of completeness and optimality in finding a solution, but they can become computationally intractable in high-dimensional configuration spaces or complex dynamic environments [80]. On the other hand, neural networks have the capability to learn efficient heuristics, approximate complex functions, and generalize from data, potentially speeding up the planning process and enabling robots to operate in scenarios where conventional methods struggle [41].

In applications where safety is paramount, hybrid systems typically integrate neural networks for perception tasks, such as object detection and scene analysis, while they depend on conventional and proven control methods to guarantee the reliable and safe execution of actions [79, 81]. For instance, a robot may employ a neural network to swiftly produce a feasible but potentially suboptimal path in a complex environment, and then utilize a traditional optimization algorithm to refine this path.

An illustrative case that serves as both an inspiration and a basis for our work is presented by Nguyen *et al.* [32]. They introduced a hybrid approach combining a learning-based network to predict collision costs for robots. The network evaluates action sequences from a library of motion primitives by using the current depth image and the robot's estimated velocities as inputs. It outputs collision costs for each primitive. These costs, accounting for uncertainty, are combined with goal direction from a global planner to select the optimal action sequence in a receding-horizon manner.

3 Methodology

This chapter details the methodology employed to develop and evaluate the risk-aware planning framework proposed in this thesis. Figure 3.1 provides a visual overview of the system's architecture, which integrates several key stages for processing information and making principled, risk-informed decisions for autonomous quadrotor navigation.

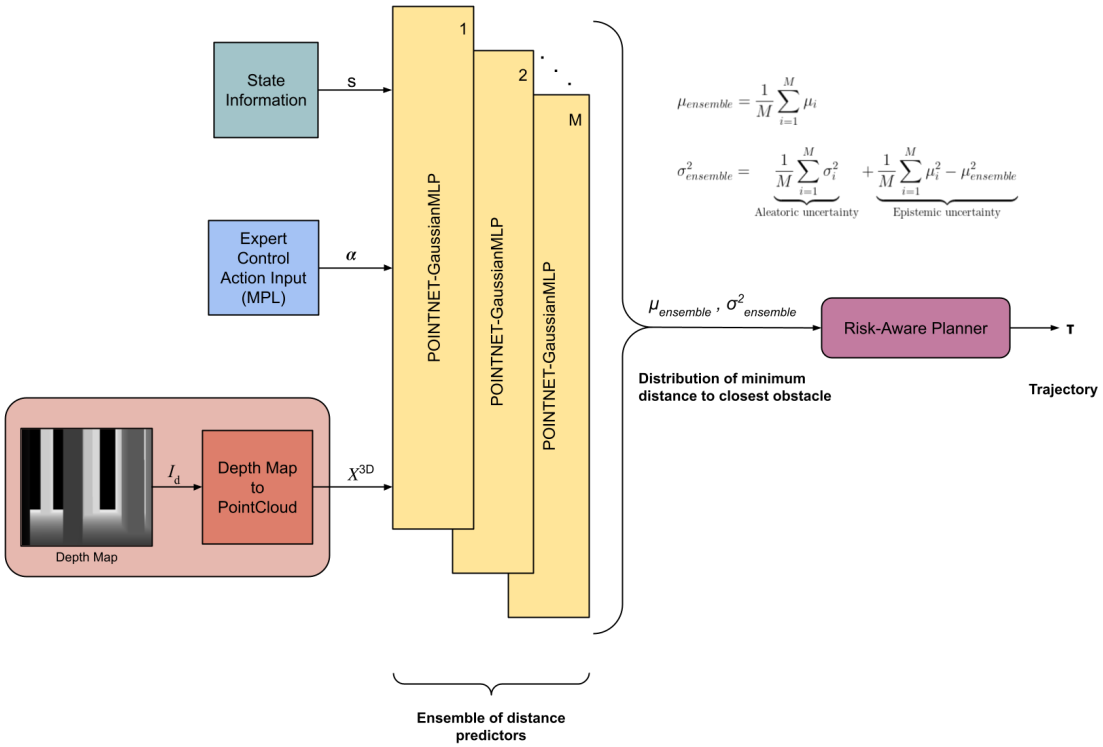


Figure 3.1: The Proposed Risk-Aware Planning Framework Architecture.

The core of the framework adopts a probabilistic approach to navigation. It processes environmental context derived from point cloud data (X^{3D}), fused with quadrotor state information (s) and potential actions defined by motion primitives (α). This information feeds an uncertainty-aware prediction module, built upon an ensemble of neural networks (integrating PointNet and Gaussian MLP structures), designed to estimate not just the minimum distance to obstacles (μ_d) but also the associated predictive uncertainty (σ_d^2) for each candidate primitive. Leveraging these probabilistic outputs, a risk-aware planning policy employs CVaR assessment to quantify the risk of each primitive. The planner employs Cross Entropy Method (CEM) to optimize and select the final trajectory (τ) and the associated control actions, effectively balancing mission objectives with the quantified risk, ensuring the navigation respects safety margins under uncertainty. The subsequent sections describe the specific design and implementation details of each

major component: the motion primitive library (MPL), the uncertainty-aware distance prediction network, the risk assessment formulation, the planning policy algorithm, and the simulation environment used for development and evaluation.

3.1 Motion Primitive Library

Fundamental to the risk-aware planner's operation is the set of potential actions it can evaluate and select from. This action space is defined by a library of dynamically feasible motion primitives, generated as described below. The motion primitive generation follows a cubic polynomial model based on jerk-controlled trajectories. For each dimension (x, y, and z), the position trajectory is defined by:

$$p(t) = p_0 + v_0 t + \frac{1}{2} a_0 t^2 + \frac{1}{6} J t^3 \tag{3.1}$$

where p_0, v_0, and a_0 represent the initial position, velocity, and acceleration respectively, while J is the applied jerk (rate of change of acceleration) and t is time. The corresponding velocity profile follows:

$$v(t) = v_0 + a_0 t + \frac{1}{2} J t^2 \tag{3.2}$$

This formulation ensures that the resulting trajectories maintain C^2 continuity (continuous up to the second derivative), which is essential for the realistic execution of trajectories by the quadrotor's low-level controller. To generate the primitive library, we sample jerk values within constrained bounds ($J \in [-1.68, 1.68]$ for each dimension) and compute the resulting trajectory over a finite time horizon ($T = 1$ second, discretized into 10 steps). Each primitive represents a potential control input that could be applied to the quadrotor. As shown in Figure 3.2, this sampling approach creates a diverse set of dynamically feasible trajectories that cover the reachable space of the quadrotor from its current state.

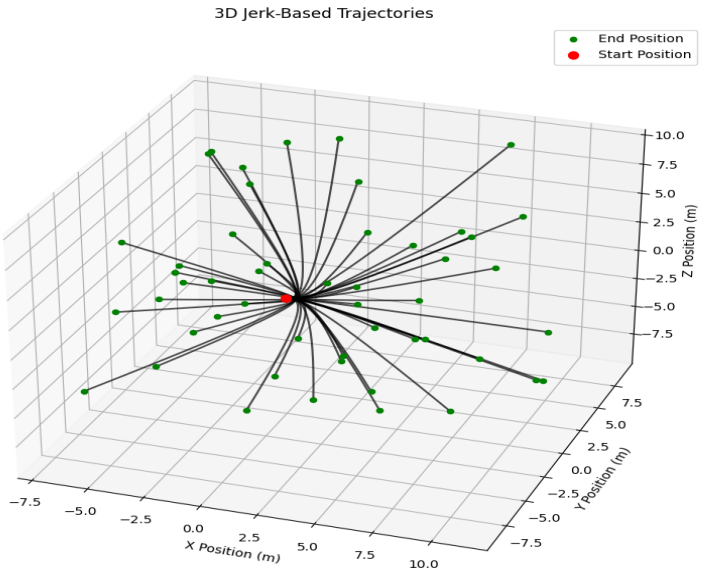


Figure 3.2: 3D jerk-based motion primitives. We sample a total of 50 dynamically feasible trajectories.

To enable the planner to choose safely among these trajectories requires predicting the potential minimum distances to the closest obstacle and the associated uncertainty of executing each primitive. This task is performed by the neural network detailed in the next section.

3.2 Uncertainty-Aware Distance Prediction

To enable risk-aware selection among the motion primitives defined previously, the framework requires predicting the minimum distance to obstacles and the associated uncertainty for each candidate trajectory. This crucial step is accomplished using a specialized neural network architecture, illustrated in Figure 3.3. The network is designed explicitly to output probabilistic predictions, estimating both the minimum distance and quantifying the inherent uncertainty in that estimation, thereby providing the necessary input for the downstream risk assessment process detailed in Section 3.3.

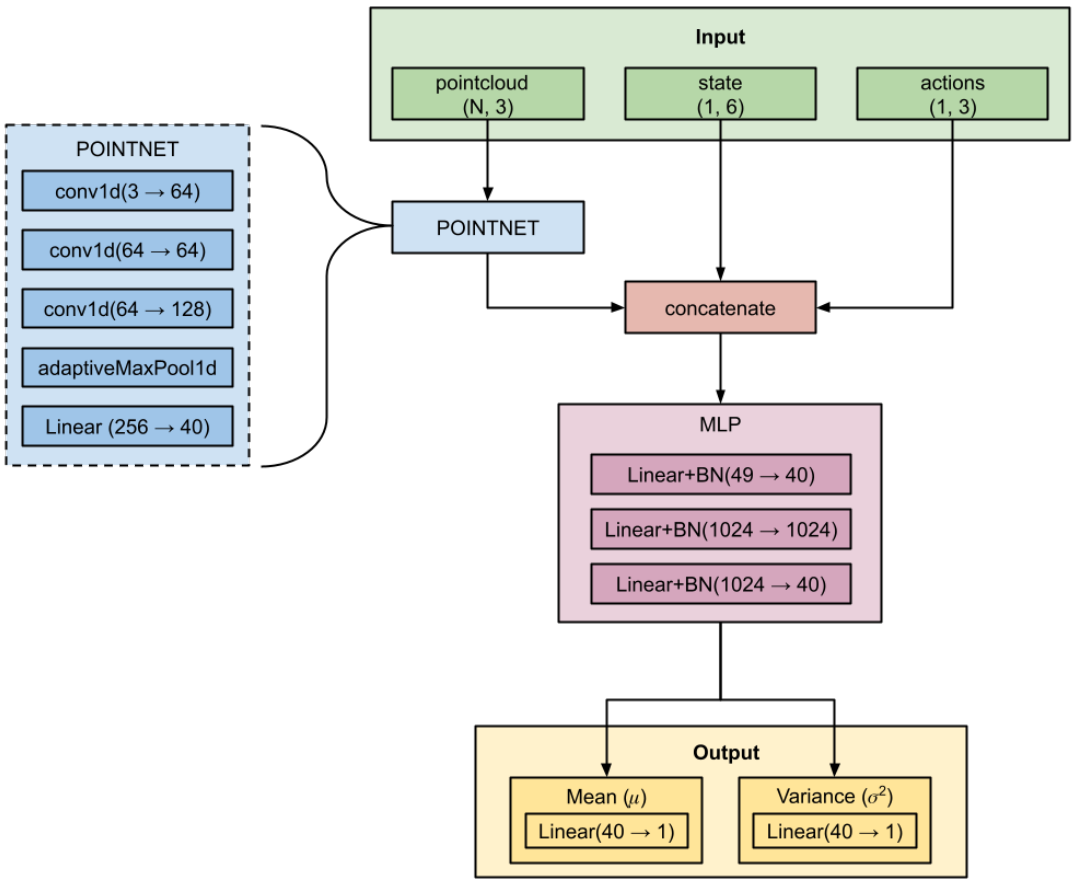


Figure 3.3: The proposed POINTNET-GaussianMLP architecture for uncertainty-aware distance prediction.

The network architecture combines principles from PointNet [46] for robust point cloud processing with a probabilistic output layer. It accepts 3D point cloud data (N points x 3 dimensions) as primary input. To handle the unordered nature of point clouds and extract meaningful spatial features, the input is first processed through a simplified PointNet block. This block con-

sists of a series of 1D convolutional layers (filters: 64, 64, 128) acting on each point, followed by an adaptive max pooling operation to aggregate global features into a fixed-size feature vector, ensuring permutation invariance [46]. This geometric feature representation (40 features) is then concatenated with the robot's current partial state information (6 features: velocities and accelerations) and parameters defining the candidate action being evaluated (3 features: 3D jerk values).

This combined 49-dimensional feature vector encapsulates the environmental context, robot state, and intended motion. It is subsequently processed by an MLP block designed for probabilistic output. The MLP uses fully connected layers (structure: 49, 1024, 40, with ReLU activations). Crucially, instead of a single output node, the final layer branches into two separate linear output heads: one (μ-Linear) predicts the mean (μ_d) of the estimated minimum distance, while the second (σ^2-Linear) predicts the log-variance ($\log(\sigma_d^2)$) associated with this estimate. Together, these two outputs parameterize a Gaussian distribution $\mathcal{N}(\mu_d, \sigma_d^2)$ over the predicted minimum distance. This probabilistic output provides not only a distance estimate but also a measure of the network's confidence (uncertainty), which is essential for the subsequent stages.

The details regarding network training are addressed later in Section 4.4. Since the network's main function is to predict distances, we will refer to it as a distance predictor, a term that will be used interchangeably in the subsequent sections.

3.3 Risk-Aware Planning Policy

The core of our planning framework is a risk-aware policy that integrates probabilistic distance predictions with principled risk assessment techniques. We employ a Cross Entropy Method (CEM) optimization approach combined with CVaR to select trajectories that minimize both task-oriented costs and collision risk. Algorithm 1 provides a detailed overview of our approach.

The risk-aware planner algorithm starts by obtaining the robot's current state (Line 1), which includes its velocity v_o and acceleration a_o. Next, the algorithm collects the filtered depth image I_d (Line 2). This image is then converted into a 3D point cloud X^{3D} (Line 3) and normalized to a range between [0, 1] to ensure consistent inputs for our distance predictor (Line 4).

The planner initializes a probability distribution over jerk control inputs by defining the mean μ_J and covariance matrix Σ_J (Line 5). Initial jerk samples J_i are then drawn from this distribution $\mathcal{N}(\mu_J, \Sigma_J)$ for a batch of size N_{batch} (Line 6), which will be used to generate and evaluate candidate trajectories.

The core optimization process begins with an iterative refinement loop (Lines 7-25). For each iteration, the algorithm processes every jerk sample J_i in the batch (Lines 8-12). For each jerk sample, a trajectory primitive τ_i is generated based on the jerk and current robot state (Line 9). The distance predictor then processes this primitive along with the normalized point cloud to predict a probabilistic distance distribution, represented by mean μ_d^i and standard deviation σ_d^i (Line 10). To account for the stochastic nature of distance prediction, samples d_k^i are drawn from this normal distribution (Line 11).

To systematically enforce safety during trajectory optimization, the algorithm utilizes *barrier functions* that encode state-dependent safety constraints in a smooth and differentiable manner. Barrier functions provide a continuous measure of constraint satisfaction and are widely used in

safety-critical control and learning systems due to their compatibility with gradient-based methods and their ability to shape feasible spaces without requiring hard constraints [82, 83].

In this context, a distance-based barrier function is defined for each trajectory i after all batched samples are processed (Line 13):

$$f^i_{\text{dist}} = d^i - d_{\text{safe}} \tag{3.3}$$

where d^i is the predicted distance to the closest obstacle, and d_{safe} is the predefined minimum safety margin. The barrier function f^i_{dist} quantifies how far the system is from violating the safety constraint; a positive value implies the trajectory remains safely within the allowable region, while a non-positive value indicates that the constraint is either active or violated.

To avoid sudden or unsafe changes in safety over time, the algorithm additionally establishes a temporal barrier constraint. Instead of guaranteeing safety at every individual time step, this constraint maintains that the safety margin does not diminish rapidly between successive steps. The violation of this constraint is expressed as follows:

$$h^i_{\text{dist}} = \max(0, -(f^i_{\text{dist}}(t+1) - f^i_{\text{dist}}(t) + \gamma \cdot f^i_{\text{dist}}(t))) \tag{3.4}$$

Here, the parameter $\gamma \in (0, 1)$ governs the conservativeness of the constraint: higher values of γ lead to stricter enforcement by allowing less degradation in the safety margin, whereas lower values permit more flexibility. The term $\gamma \cdot f^i_{\text{dist}}(t)$ defines a margin that scales with the current safety level, encouraging the system to preserve or improve safety over time.

The barrier constraints ensure that trajectories not only satisfy safety conditions at each point in time but also evolve in a smooth and predictable manner. By integrating these constraints into the learning and planning process, the algorithm achieves robust safety-aware behaviour without sacrificing computational tractability.

The algorithm then computes the CVaR risk assessment (Line 15) over the distribution of these violation values, focusing on the worst $1 - \alpha$ fraction of outcomes:

$$\text{CVaR}_\alpha(h^i_{dist}) = \mathbb{E}[h^i_{dist} | h^i_{dist} \geq \text{VaR}_\alpha(h^i_{dist})] \tag{3.5}$$

From the full batch, the algorithm selects the top N_{cost} primitives with the lowest risk (Line 16), where N_{cost} represents the number of elite samples selected based on their risk assessment before calculating the full planning cost. These risk-prioritized primitives proceed to the cost calculation phase (Lines 17-21). For each of these primitives, we calculate:

- The goal-oriented cost C^i_{goal}. This measures the distance from the robot's current position to the target position (Line 18)
- The smoothness cost C^i_{smooth}. This cost is based on jerk magnitude to encourage smooth motion (Line 19)
- The total cost that combines these individual costs with the risk assessment (Line 20):

$$C^i_{total} = w_g \cdot C^i_{goal} + w_s \cdot C^i_{smooth} + w_r \cdot \text{CVaR}_\alpha(h^i_{dist}) \tag{3.6}$$

  where w_g, w_s, and w_r are weights that balance the competing objectives.

After evaluating all primitives, the algorithm selects the K elite samples with the lowest total cost (Line 22). These elite samples are used to update the jerk distribution parameters μ_J and

Σ_J through weighted mean and covariance calculations (Line 23):

$$\mu_J = (1 - \alpha_\mu)\mu_J + \alpha_\mu \hat{\mu}_J \tag{3.7}$$

$$\Sigma_J = (1 - \alpha_\Sigma)\Sigma_J + \alpha_\Sigma \hat{\Sigma}_J + \epsilon I \tag{3.8}$$

where $\hat{\mu}_J$ and $\hat{\Sigma}_J$ are the weighted mean and covariance of the elite samples:

$$\hat{\mu}_J = \frac{\sum_{i=1}^{K} w_i a_i}{\sum_{i=1}^{K} w_i} \tag{3.9}$$

$$\hat{\Sigma}_J = \frac{\sum_{i=1}^{K} w_i (a_i - \mu_J)(a_i - \mu_J)^T}{\sum_{i=1}^{K} w_i} \tag{3.10}$$

where a_i are the elite action samples, $w_i = \exp(-\frac{1}{\lambda}(J_i - J_{min}))$ are weights derived from the costs J_i (with J_{min} being the minimum cost among elite samples), α_μ and α_Σ are learning rates, and ϵI adds a small regularization term to ensure exploration. New jerk samples are then drawn from the updated distribution for the next iteration (Line 24), continuing the refinement process.

After the final iteration, the algorithm selects the trajectory τ^* with the minimum total cost (Line 26). The velocity profile v^* associated with this optimal trajectory is extracted (Line 27), and the first velocity command v_1^* is executed (Line 28). The algorithm then returns to step 1, implementing a receding horizon control strategy that continuously replans based on updated sensor information.

Algorithm 1 Risk-Aware Planner

$s = [v_o, a_o] \leftarrow$ Get robot's state
$I_d \leftarrow$ Get filtered depth image
$\mathcal{P} \leftarrow$ Convert I_d to point cloud $\mathcal{X}^{3D}$
$\mathcal{P}_{scaled} \leftarrow$ Normalize point cloud to $[0, 1]$ range
$\mu_J, \Sigma_J \leftarrow$ Initialize jerk distribution
Sample initial jerks J_i $(i = 1, \ldots, N_{batch})$ from $\mathcal{N}(\mu_J, \Sigma_J)$
for $iter = 1$ to N_{max} **do**
  for $i = 1$ to N_{batch} **do**
    $\tau_i \leftarrow$ Generate primitive using J_i, s
    $(\mu_d^i, \sigma_d^i) \leftarrow$ DistancePredictor$(\mathcal{P}_{scaled}, s, J_i)$
    Generate samples $d_k^i \sim \mathcal{N}(\mu_d^i, \sigma_d^i)$
  end for
  Calculate barrier function $f_{dist}^i = d^i - d_{safe}$
  Calculate constraint violation h_{dist}^i by Eq. 3.4
  Calculate CVaR$_\alpha(h_{dist}^i)$ for all samples by Eq. 3.5
  Select top N_{cost} primitives with lowest risk
  for $i = 1$ to N_{cost} **do**
    Calculate C_{goal}^i using distance to goal
    Calculate C_{smooth}^i using jerk magnitude
    $C_{total}^i = w_g \cdot C_{goal}^i + w_s \cdot C_{smooth}^i + w_r \cdot \text{CVaR}_\alpha(h_{dist}^i)$
  end for
  Select K elite samples with lowest C_{total}^i
  Update μ_J, Σ_J using elite samples with weighted mean and covariance
  Sample new jerks from $\mathcal{N}(\mu_J, \Sigma_J)$
end for
Choose τ^* from primitives with minimum C_{total}
Extract velocity profile v^* for τ^*
Execute the first velocity ${v^*}_1$, then go to step 1

4 Implementation

This thesis project was conducted entirely in simulation due to the time and resource constraints associated with real-world flight tests of quadrotors. While simulation introduces certain limitations compared to physical implementation, it provides a controlled environment for rapid development and systematic evaluation of the proposed risk-aware planning framework.

4.1 System Requirements

4.1.1 Hardware Configuration

All simulations and neural network training were performed on a single workstation with the following specifications:

Table 4.1: PC Hardware specifications

Component	**Specification**
Processor	Intel Core i7 11th Generation
GPU	NVIDIA RTX 4070 (8GB VRAM)
Memory	32GB RAM
Storage	1TB

4.1.2 Software Environment

The software stack used in this research consists, but is not limited to:
Ubuntu 20.04 LTS, Robot Operating System (ROS) Noetic, Python 3.10, Jax 0.4.13, Equinox 0.12.1, PyTorch 2.6.0, Open3D 0.19.0.

4.2 Simulation Framework

The simulation environment was built using PX4 Software-In-The-Loop (SITL) [84] integrated with Gazebo 11.0 [85] and ROS [86]. This configuration offers significant advantages for UAV research, as the PX4 SITL runs the actual PX4 flight controller firmware in simulation, providing high-fidelity behaviour that closely matches real-world systems. As an industry standard widely used in both academic research and commercial UAV applications, PX4 SITL ensures our results have practical relevance. The framework includes accurate aerodynamic models, sensor simulation, including realistic LiDAR point cloud generation, and environmental interaction modelling [84].

Out of the box, PX4 SITL is equipped with the MAVROS package [87], which enables interaction between ROS and the PX4 autopilot via the MAVLink protocol, replicating the communication structure that would be applied in a real-world setup.

We also employ the FastPlanner framework [3] to autonomously control the quadrotor movements. FastPlanner is an advanced motion planning algorithm designed to generate efficient, collision-free trajectories in dynamic settings. The planner was configured to operate with nominal linear velocities of $0.5\,\mathrm{m\,s^{-1}}$, $0.75\,\mathrm{m\,s^{-1}}$, and $1.0\,\mathrm{m\,s^{-1}}$, with the specific velocity being set depending on the complexity of the environment and the need to generate varied data, while adapting its velocity as necessary to avoid obstacles and ensure smooth navigation.

4.2.1 Simulation Environments

To assess system performance in various operational scenarios, a series of simulation environments (see Figure 5.1) were developed in the Gazebo simulator. These environments were designed to systematically vary complexity, which is measured by factors such as obstacle density and navigational constraints. Obstacle representation, including pillars and walls used as collision meshes, was custom-designed within the Gazebo simulation platform.

4.2.2 Drone Simulation Models

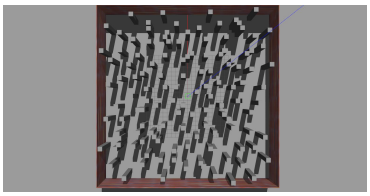
In this research, we utilised two distinct drone models within the simulation environment to gather varied sensor data. Firstly, we employed the Iris quadrotor model, available within the PX4 SITL package. Specifically, the *iris_depth_camera* variant was selected, which comes equipped with a simulated Intel RealSense D455 depth camera. While this sensor simulates both RGB and depth image outputs, the work presented in this thesis focused exclusively on using the depth data component for environmental perception and planning. Secondly, a customised Bebop2 drone simulation model, equipped with a Hokuyo LiDAR sensor positioned on top, was also utilised. The use of this second model allowed for the collection of simulated data directly from a LiDAR sensor, providing a different modality compared to the depth camera data obtained from the Iris model.

In the PX4 SITL environment, the simulated depth component of the camera operates with the default parameters, as shown in Table 4.2.

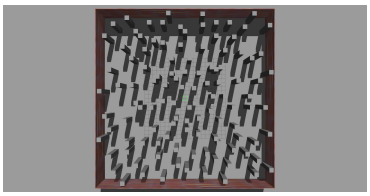
Table 4.2: Intel RealSense D455 depth camera specifications

Resolution	1280 x 720 pixels
Frame Rate	30 Hz
Depth Range	0.1 m to 62 m
Field of View (FOV)	87° (horizontal), 58° (vertical)

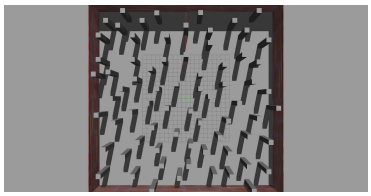
Depth cameras exhibit increased noise levels at extended ranges [5]. For this reason, constraining the maximum range to 3 meters proved adequate for acquiring point clouds of reduced noise within our environment.



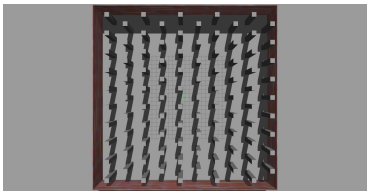
env1
Num Obstacles: 127
Avg. spacing: 2.37 m
Min. spacing: 0.42 m
Max. spacing: 4.73 m



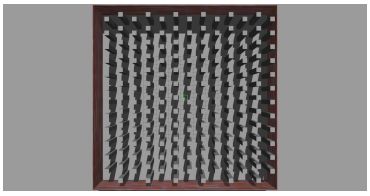
env2
Num Obstacles: 112
Avg. spacing: 2.69 m
Min. spacing: 1.11 m
Max. spacing: 4.58 m



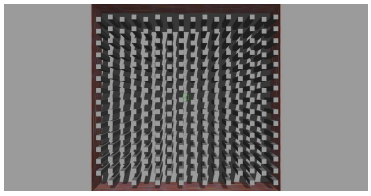
env3
Num Obstacles: 71
Avg. spacing: 3.67 m
Min. spacing: 2.54 m
Max. spacing: 5.61 m



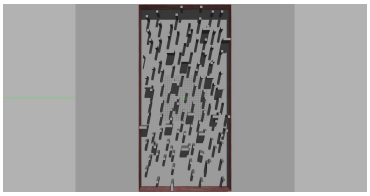
env4
Num Obstacles: 95
Avg. spacing: 4.00 m
Evenly spaced



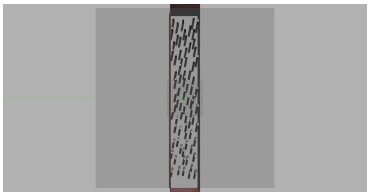
env5
Num Obstacles: 162
Avg. spacing: 3.00 m
Evenly spaced



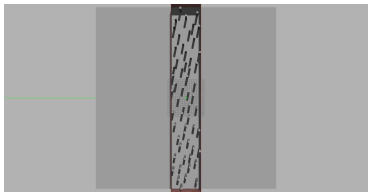
env6
Num Obstacles: 232
Avg. spacing: 2.50 m
Evenly spaced



env7
Num Obstacles: 125
Avg. spacing: 3.49 m
Min. spacing: 1.08 m
Max. spacing: 8.01 m



env8
Num Obstacles: 77
Avg. spacing: 3.09 m
Min. spacing: 1.77 m
Max. spacing: 4.70 m



env9
Num Obstacles: 57
Avg. spacing: 4.23 m
Min. spacing: 3.12 m
Max. spacing: 6.46 m

Figure 4.1: Simulation environments

4.3 Data Collection and Processing

We collected data in the custom environments shown in Figure 5.1 using the *iris_depth_camera* drone model. Our main goal was to create a detailed dataset for training and validating the risk-aware planning framework. We included point clouds and odometry data in this dataset, ensuring that all data were synchronised in time to maintain consistency across different sensor modalities.

4.3.1 Data Collection

We gathered data from the following sources:

- **Point Cloud Data:** Captured from the onboard depth camera to represent the surrounding environment.
  ROS topic: */camera/depth/points*

- **State Estimation Data:** Retrieved from odometry to obtain the quadrotor's velocity and acceleration.
  ROS topic: */mavros/local_position/odom*
  The velocity components (v_x, v_y, v_z) were directly obtained, while the acceleration components (a_x, a_y, a_z) were computed using numerical differentiation.

As mentioned earlier, we automated the data gathering process by flying the drone autonomously using FastPlanner. In order to collect a comprehensive dataset for training the distance predictor model, we randomly sampled the goal pose after each goal was reached. We had more than 10 fixed goal poses for each environment. This approach enabled us to gather over 120,000 data points in total. The derived dataset was then split into training, testing, and validation subsets.

4.3.2 Data Processing

To prepare the collected data for neural network training, the processing pipeline applies the following techniques:

1. **Point Cloud Transformation and Filtering:** This stage refines the raw point cloud data.
   - Transforms incoming point clouds into the drone's odometry frame for consistent spatial reference.
   - Downsamples the point cloud using voxel grid filtering (*voxel size = 0.15m*) to reduce computational load while preserving structural information.
   - Crops the point cloud vertically to focus only on obstacles within a ±0.5m height range relative to the drone's current altitude, removing irrelevant ground or ceiling points.
2. **Odometry Validation:** This stage filters out potentially corrupted or unsafe data instances based on odometry readings.
   - Removes data frames captured when the drone operates too close to the ground to ensure focus on relevant flight regimes.
   - Discards frames corresponding to collisions by assessing the minimum distance between the drone's position and the nearest obstacle point against a predefined threshold (*dist_to_obs = 0.15m*).
3. **Ground-Truth Minimum Distance Generation:** This stage computes the target labels for the neural network.
   - Calculates the Euclidean distances between each waypoint defining a candidate motion primitive trajectory (Note: the number of waypoints is defined by the horizon steps per primitive) and all points within the corresponding processed point cloud.
   - Extracts the minimum distance found across all waypoints for that specific primitive trajectory, establishing this value as the ground-truth safety label for network training.

The post-processing stage yields a final dataset comprising 104,299 data points, subsequently divided into training (80%), validation (10%), and testing (10%) subsets.

4.4 Model Training

The training of the uncertainty-aware distance predictor architecture follows a comprehensive approach designed to ensure robust performance in distance prediction tasks while providing reliable uncertainty estimates. This section details the training methodology, loss function, and evaluation metrics employed.

4.4.1 Deep Ensemble Strategy

Implementing a deep ensemble approach enhances the reliability of uncertainty estimates. Instead of relying on a single network instance, our approach follows [56]. We train $M = 3$ identical networks with different random initializations. Each network independently produces a mean and variance estimate, which are then combined to create a more robust predictive distribution.

The ensemble prediction combines individual network outputs as follows:

$$\mu_{ensemble} = \frac{1}{M} \sum_{i=1}^{M} \mu_i \tag{4.1}$$

$$\sigma^2_{ensemble} = \underbrace{\frac{1}{M} \sum_{i=1}^{M} \sigma_i^2}_{\text{Aleatoric uncertainty}} + \underbrace{\frac{1}{M} \sum_{i=1}^{M} \mu_i^2 - \mu^2_{ensemble}}_{\text{Epistemic uncertainty}} \tag{4.2}$$

This approach captures both the aleatoric uncertainty (data noise) through individual network variance predictions and epistemic uncertainty (model uncertainty) through the disagreement between ensemble members.

4.4.2 Loss Function

The model was trained using a Negative Log-Likelihood (NLL) loss function appropriate for probabilistic regression tasks:

$$\mathcal{L}_{NLL} = \frac{1}{2} \log(2\pi\sigma^2) + \frac{(y - \mu)^2}{2\sigma^2} \tag{4.3}$$

where y represents the ground truth minimum distance, and μ and σ^2 are the network's predicted mean and variance.

This loss function naturally balances the trade-off between accurate predictions and well-calibrated uncertainty estimates. It penalizes overconfident incorrect predictions (small σ^2 with large error) while allowing appropriate uncertainty (larger σ^2) in challenging cases.

4.4.3 Hyperparameters and Training Procedure

The model was trained using the AdamW optimizer with the following hyperparameters:

- Learning rate: 3×10^{-4}
- Batch size: 512

- Weight decay: 6×10^{-5}
- Training epochs: 50

The model's training employed a cosine annealing scheduler to modify the learning rate over time. We further integrated an early stopping feature into the training pipeline, which terminates the training process according to the validation set's performance, effectively reducing the risk of overfitting. To prevent variance collapse and ensure stable training, the networks were initialized with a higher output variance and gradually refined through training. In addition, gradient clipping (using a maximum norm of 1.0) was applied to prevent exploding gradients during the optimization process.

5 Experimental Results & Discussion

In this chapter, we assess the performance of our proposed risk-aware planner pipeline. The results reflect the performance on unseen testing data.

5.1 Performance Evaluation of the Distance Prediction Model

The distance predictor model was evaluated using a "success rate" metric, which measures the percentage of predictions where the absolute difference between predicted and ground truth minimum distance was within a 0.1m threshold. The evaluation revealed a stark contrast between deterministic and probabilistic approaches:

- **Deterministic prediction**: Using only the predicted mean (μ_d), the model achieved a success rate of 31.64% on the test set.
- **Uncertainty-aware prediction**: Using a sampling-based approach (i.e., drawing 100 samples from the predicted Gaussian distribution $\mathcal{N}(\mu_d, \sigma_d^2)$ for each test case), the success rate improved dramatically to 99. 18%.

The high success rate of the sampling-based method demonstrates that our ensemble network produces well-calibrated uncertainty estimates. This result reinforces the core hypothesis of our thesis: uncertainty quantification is critical for robust risk assessment. In contrast, deterministic predictions, which omit uncertainty information, are often inadequate for safety-critical scenarios where anticipating a range of possible outcomes is essential.

Table 5.1: Comparison of results on the distance prediction model

Evaluation Method	**Success Rate (%)**
Deterministic (Mean-only)	31.64
Uncertainty-aware (100 samples)	99.18

5.2 Simulation Results

We conducted extensive simulation experiments to evaluate the performance of the proposed risk-aware planning framework across multiple environments. The primary evaluation metric is the *collision count*, which records the number of times the drone collided with obstacles during 50 simulation trials in each environment, as presented in Table 5.2. A trial was considered successful when the drone reached its target without any collisions. The results indicate that our risk-aware planner using uncertainty predictions performed much better than the basic approach

that only uses the mean values. This improvement showed up in all nine simulation environments. The biggest improvement was in environment 3, where collisions dropped from 24 to just 1. Even in the hardest setting (environment 6 with narrow passages), collisions still went down by 63.8% , i.e., from 47 to 17.

Table 5.2: Comparison of collision counts between deterministic and uncertainty-aware prediction methods within the risk-aware planner framework across various simulation environments.

Environment	**Collision Count (out of 50)**	
	Deterministic (Mean-only)	**Uncertainty-aware**
env1	42	9
env2	38	7
env3	24	1
env4	34	2
env5	44	10
env6	47	17
env7	36	11
env8	35	11
env9	34	9
Average (%)	**74.2%**	**17.1%**

Figure 5.1 presents an example navigation scenario where the planner directs the drone towards a target point while ensuring safe distances from obstacles. In Figure 5.2, we demonstrate our navigation algorithm's effectiveness when confronted with a complex obstacle blockade. The sequence shows the drone initially taking a direct path toward the target before detecting the blockade. The yellow trajectory reveals how the algorithm balances goal-seeking with obstacle avoidance to find a safe path around the blockade. Throughout the trial, the drone maintains a safe distance from obstacles while successfully reaching its target.

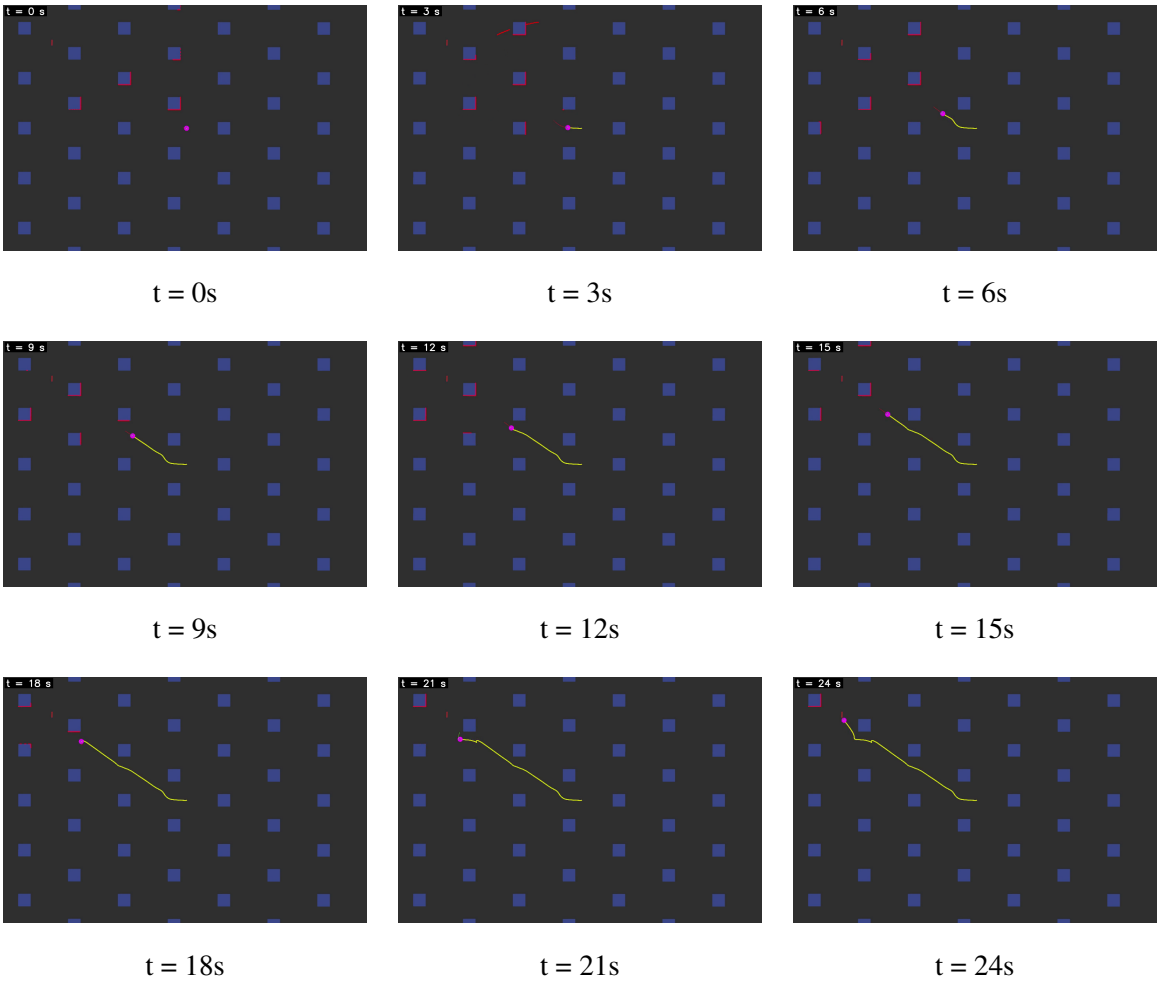


Figure 5.1: Drone navigates in a simulated world (env4).
The images depict the drone's (pink) state at 3-second intervals as it navigates towards the designated goal point (red arrow at the top left corner) while avoiding obstacles (blue squares). The yellow path represents the drone's trajectory history.

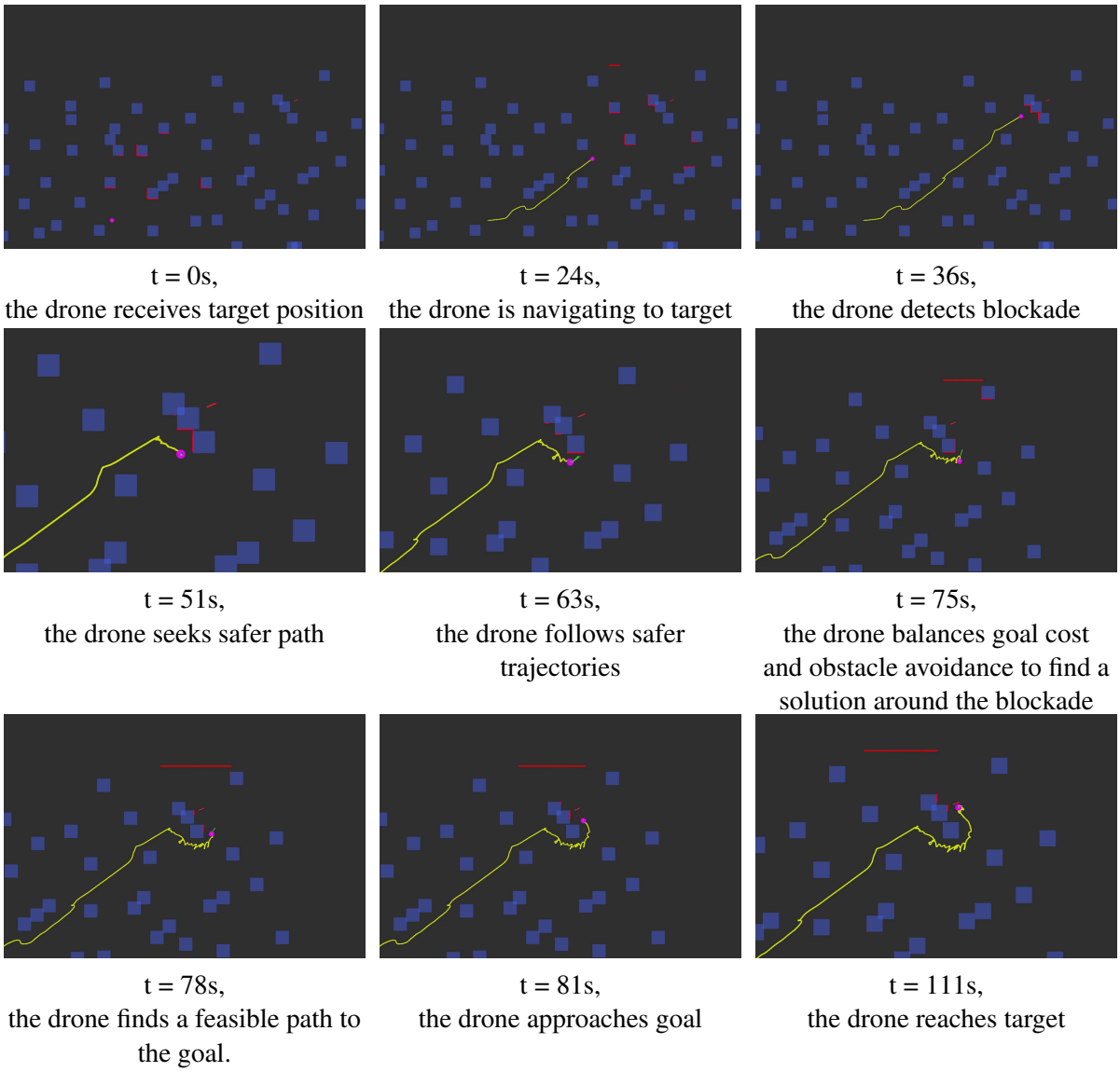
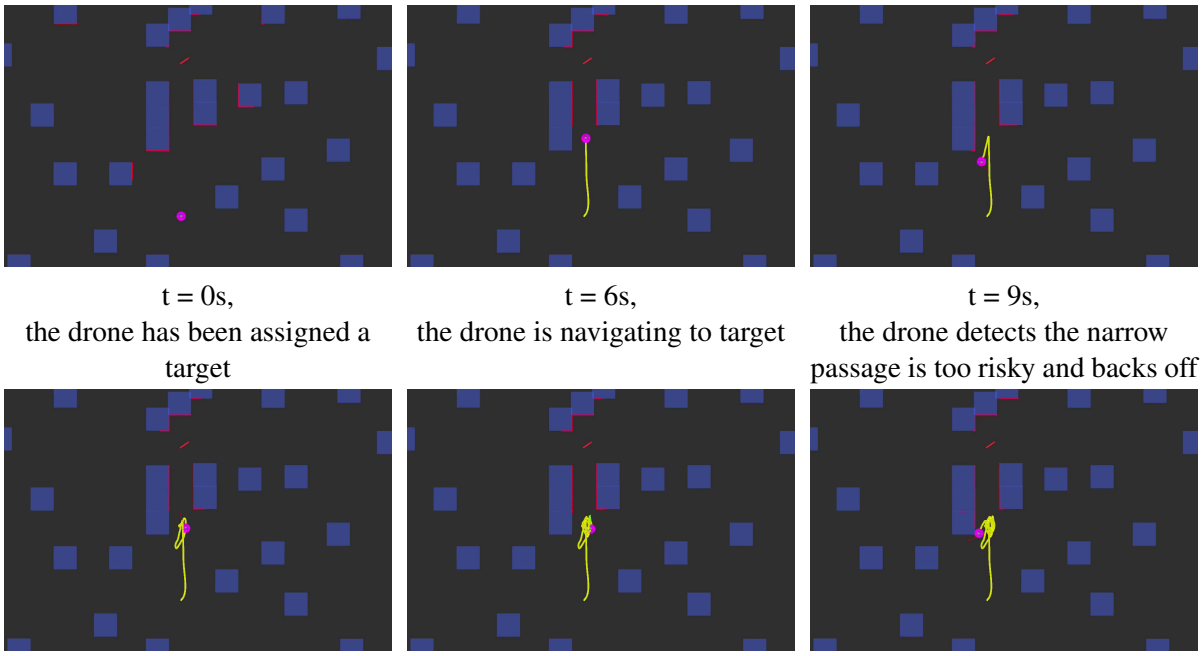


Figure 5.2: Drone navigates safely while avoiding obstacles to reach the goal.
The images depict the drone's (pink) state as it navigates towards the designated goal point (red arrow). The yellow path represents the drone's trajectory history, while the blue squares are the obstacles in the drone's path.

The systematic evaluation identified specific scenarios where collision-free navigation could not be guaranteed (See Figure 5.3). Analysis of these failure modes reveals important limitations:

- **Goal conflict near obstacles:** When the target goal was positioned in close proximity to an obstacle, the planner exhibited conflicting behaviours. The goal-attraction component incentivised movement towards the goal, while the risk-aversion component (driven by CVaR) penalised trajectories near the obstacle. In some instances, this conflict resulted in oscillatory motion parallel to the obstacle boundary, where the planner repeatedly attempted to approach the goal while simultaneously retreating due to perceived risk, occasionally leading to a collision during these oscillations.
- **Navigation in extremely narrow passages:** Failures were also observed during navigation through very confined passages. This behaviour appears linked to the limited Field of

View (FOV) of the simulated depth sensor. The planner primarily reacts to obstacles detected within its immediate FOV. In narrow spaces, obstacles just outside the FOV (e.g., directly to the sides) remain undetected until the drone is very close or oriented to them. Consequently, reactive manoeuvres to avoid obstacles ahead inadvertently cause lateral drift or oscillations within the confined space, sometimes resulting in collision with these momentarily unseen side obstacles.



t = 0s,
the drone has been assigned a target

t = 6s,
the drone is navigating to target

t = 9s,
the drone detects the narrow passage is too risky and backs off

t = 15s,
the drone is aware of the risk but tries to continue toward the target

t = 27s,
the drone experiences decision conflict and begins oscillating

t = 33s,
the drone collides due to oscillation and limited FOV

Figure 5.3: Illustration of a failure during flight.
The images depict the drone's (pink) state as it navigates towards the designated goal point (red arrow located above the narrow path). The yellow path represents the drone's trajectory history, while the blue squares are the obstacles in the drone's path.

5.3 Planner Performance and Resource Consumption

We profiled the planner to examine the resources consumed during planner execution. These results have been illustrated in Figure 5.4 and further summarized in Table 5.3.

Figure 5.4 presents the resource utilization profile (CPU, GPU and RAM) of our risk-aware planner throughout a 60s navigation task. The profile exhibits three distinct operational phases. Initially (0-5s), resources remain at baseline levels as the system prepares for execution. At approximately 5s, a significant spike occurs across all metrics, with the CPU usage surging to approximately 130%, while both system and GPU memory allocation increase substantially. This pronounced peak coincides with the planner's activation and primarily reflects the computational cost of JAX Just-In-Time (JIT) [88] compilation during first execution.

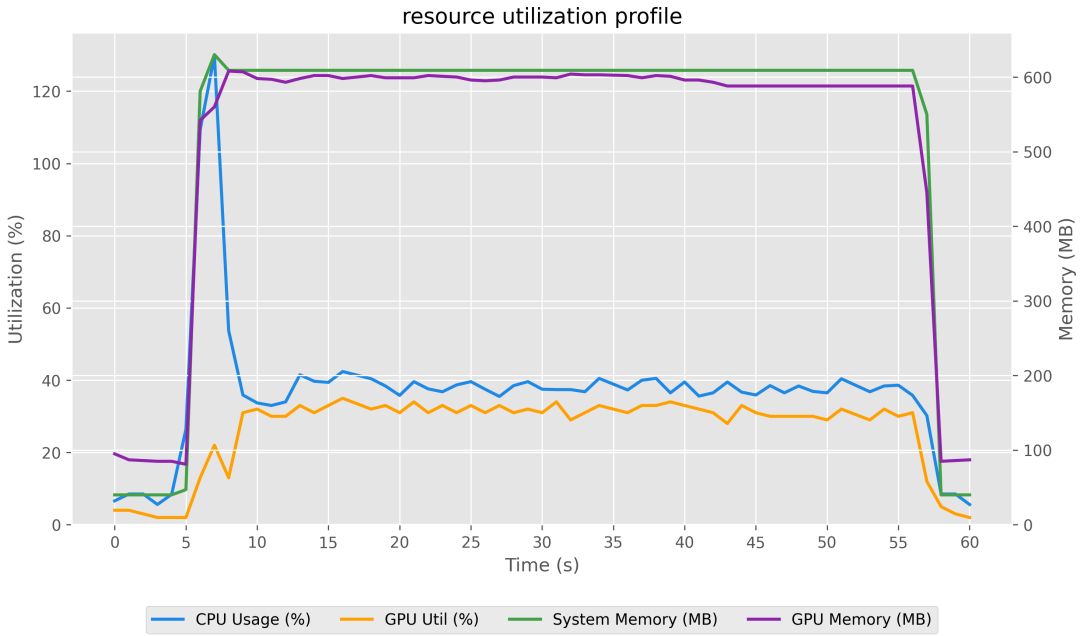


Figure 5.4: Illustration of the resource utilization profile of the planner.
It involves the total resources spent on the planning, execution and visualization modules of the planner framework.

After the compilation phase, the system shifts to a stable operation mode between 7 and 56 seconds, as the drone makes its way to the target. Throughout this phase, resource usage holds steady, with the CPU averaging around 36%, GPU usage at 26%, and memory allocation remaining at 630 MB for system memory (RAM) and 608 MB for GPU memory. Once the drone arrives at its destination (after 56 seconds), all resource usage quickly reverts to baseline as the planner remains on standby until issued another navigation task.

Furthermore, the planner exhibits remarkable computational performance. The average time for model inference is approximately 34 ms (about 29 Hz), while the entire planning cycle, which includes the inferencing process, averages less than 100 ms. The extended duration is mainly due to the CEM optimization phase, which iteratively evaluates numerous candidate trajectories to guarantee both safe and optimal paths.

Table 5.3: Statistical summary of the planner resource consumption and performance

Metric	**Min**	**Max**	**Avg**	**Std Dev**
CPU Usage (%)	5.6	129.8	36.4	19.2
RAM Usage (MB)	40	630	520	205
GPU Utilization (%)	2.0	35.0	26.0	10.8
GPU RAM Usage (MB)	81	608	513	184
Model Inference Time (ms)	31.77	43.18	34.33	2.05
Total Planning Time (ms)	81.24	134.21	90.36	10.75

6 Conclusion

This thesis aimed to develop a risk-aware planning framework for autonomous quadrotors that utilize pointclouds for perception. The specific objectives included neural collision prediction, risk-aware planning, and evaluation in simulation environments. All three objectives have been successfully met.

The first objective (developing a neural network-based action-conditioned collision prediction model) was accomplished by implementing an ensemble of Gaussian MLPs. This model effectively processes pointcloud data and robot state information to predict probability distributions of distances to obstacles, providing both the mean distance and uncertainty estimates as required.

The second objective (developing a risk-aware planner) was achieved by successfully implementing a planning system incorporating CVaR metrics for risk assessment. The planner demonstrates the ability to iteratively modify actions to maintain collision risk below specified thresholds while pursuing navigation goals.

The third objective (simulation testing) confirmed the efficacy of the integrated system, particularly within dense environments. Simulation results validate the hypothesis that incorporating uncertainty estimates in collision prediction significantly improves navigation performance compared to deterministic approaches. The experiments demonstrated that the system effectively balances safety constraints with mission objectives in challenging scenarios.

6.1 Future Work and Improvements

While the risk-aware planning framework shows promising results, several key improvements could enhance its performance and applicability. The uncertainty-aware prediction network could benefit from systematic hyperparameter tuning, which includes optimizing learning rates, architecture configurations, and training procedures. Additionally, retraining the model in diverse environments with mobile obstacles, various obstacle shapes, and sensor noise could improve its generalization, especially in complex scenarios that are not sufficiently represented in the current dataset.

Moreover, finding a better balance between mean distance prediction and variance estimation remains a significant challenge. In future work, methods that dynamically adjust this trade-off based on environmental complexity and sensor reliability should be explored. This could involve modified loss functions that adaptively weight the importance of mean accuracy versus uncertainty calibration, depending on the specific navigation context.

Finally, optimizing the computational efficiency of the planning framework for embedded systems is crucial for deployment on small aerial robots with limited processing power. To that end,

we should validate the entire system on physical hardware. Testing the risk-aware planner on a real drone navigating complex, unstructured environments would provide definitive evidence of its practical effectiveness beyond simulation results and represent the ultimate validation of this research approach.

The planner may also be adapted and improved to apply to mobile robots, with primitive actions expressible within a 2D framework.

Acknowledgements

I wish to express my heartfelt gratitude to all those who have supported and guided me throughout the completion of my master's thesis.

First and foremost, I am profoundly thankful to my family for their unwavering encouragement, understanding, and belief in my abilities. Their steadfast support has been my anchor during this journey.

I am deeply indebted to Prof. Arun K. Singh for his invaluable guidance, insightful feedback, and continuous support. His expertise and dedication have been instrumental in shaping this research.

I would also like to extend my sincere appreciation to the institution for providing the resources and environment conducive to my academic pursuits.

Special thanks to Dr. Lise D. Kakupa for her exceptional support in proofreading my work and for her encouragement during challenging times.

The author expresses gratitude to Prajyot Jadhav, Basant Sharma, and the entire AKS Lab team for their collaborative efforts to demystify complex topics, refine code, and contribute to the development of this project.

Finally, I would like to thank the team of the following AI applications: ChatGPT, Claude, Gemini, Writefull, and Grammarly for providing these useful tools to aid in enhancing the quality of my writing.

Author's signature

Bibliography

[1] Muhammad Yeasir Arafat, Muhammad Morshed Alam, and Sangman Moh. "Vision-Based Navigation Techniques for Unmanned Aerial Vehicles: Review and Challenges". en. In: *Drones* 7.2 (Feb. 2023). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, p. 89. ISSN: 2504-446X. DOI: 10.3390/drones7020089. URL: https://www.mdpi.com/2504-446X/7/2/89 (visited on 04/09/2025).

[2] Anusha Mujumdar and R. Padhi. "Nonlinear Geometric and Differential Geometric Guidance of UAVs for Reactive Collision Avoidance". In: *Journal of Guidance, Control, and Dynamics* 34 (July 2009), p. 69. DOI: 10.2514/1.50923.

[3] Boyu Zhou et al. "Robust and Efficient Quadrotor Trajectory Generation for Fast Autonomous Flight". In: *IEEE Robotics and Automation Letters* 4.4 (Oct. 2019). Conference Name: IEEE Robotics and Automation Letters, pp. 3529–3536. ISSN: 2377-3766. DOI: 10.1109/LRA.2019.2927938. URL: https://ieeexplore.ieee.org/document/8758904 (visited on 03/27/2025).

[4] Gopi Gugan. "Path Planning for Autonomous Drones: Challenges and Future Directions". In: *Drones* 7 (Feb. 2023), p. 169. DOI: 10.3390/drones7030169.

[5] Adrián González-Sieira et al. "Autonomous navigation for UAVs managing motion and sensing uncertainty". en. In: *Robotics and Autonomous Systems* 126 (Apr. 2020), p. 103455. ISSN: 09218890. DOI: 10.1016/j.robot.2020.103455. URL: https://linkinghub.elsevier.com/retrieve/pii/S0921889019307080 (visited on 04/08/2025).

[6] Sumedh Mannar et al. "Vision-based Control for Aerial Obstacle Avoidance in Forest Environments". In: *IFAC-PapersOnLine* 51 (Jan. 2018), pp. 480–485. DOI: 10.1016/j.ifacol.2018.05.081.

[7] Dipraj Debnath et al. "A Review of UAV Path-Planning Algorithms and Obstacle Avoidance Methods for Remote Sensing Applications". en. In: *Remote Sensing* 16.21 (Jan. 2024). Number: 21 Publisher: Multidisciplinary Digital Publishing Institute, p. 4019. ISSN: 2072-4292. DOI: 10.3390/rs16214019. URL: https://www.mdpi.com/2072-4292/16/21/4019 (visited on 04/09/2025).

[8] Luxin Han et al. *FIESTA: Fast Incremental Euclidean Distance Fields for Online Motion Planning of Aerial Robots*. arXiv:1903.02144 [cs]. July 2019. DOI: 10.48550/arXiv.1903.02144. URL: http://arxiv.org/abs/1903.02144 (visited on 09/13/2023).

[9] Sudarshan S. Harithas et al. *CCO-VOXEL: Chance Constrained Optimization over Uncertain Voxel-Grid Representation for Safe Trajectory Planning*. arXiv:2110.02904 [cs]. Apr. 2022. DOI: 10.48550/arXiv.2110.02904. URL: http://arxiv.org/abs/2110.02904 (visited on 09/13/2023).

[10] Jun Bian et al. “Risk-Aware Path Planning Using CVaR for Quadrotors”. In: *2023 6th International Symposium on Autonomous Systems (ISAS)*. June 2023, pp. 1–6. DOI: 10.1109/ISAS59543.2023.10164417. URL: https://ieeexplore.ieee.org/document/10164417 (visited on 03/06/2025).

[11] Junkai Jiang et al. *A Risk-aware Planning Framework of UGVs in Off-Road Environment*. arXiv:2402.02457 [cs]. Feb. 2024. DOI: 10.48550/arXiv.2402.02457. URL: http://arxiv.org/abs/2402.02457 (visited on 03/06/2025).

[12] Sleiman Safaoui and Tyler H. Summers. *Distributionally Robust CVaR-Based Safety Filtering for Motion Planning in Uncertain Environments*. arXiv:2309.08821 [cs]. Sept. 2023. DOI: 10.48550/arXiv.2309.08821. URL: http://arxiv.org/abs/2309.08821 (visited on 03/11/2025).

[13] Jesus Tordesillas, Brett T. Lopez, and Jonathan P. How. *FASTER: Fast and Safe Trajectory Planner for Flights in Unknown Environments*. en. Mar. 2019. URL: https://arxiv.org/abs/1903.03558v3 (visited on 03/03/2024).

[14] Taha Elmokadem and Andrey V. Savkin. “Towards Fully Autonomous UAVs: A Survey”. In: *Sensors (Basel, Switzerland)* 21.18 (Sept. 2021), p. 6223. ISSN: 1424-8220. DOI: 10.3390/s21186223. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8473245/ (visited on 04/09/2025).

[15] Anirudha Majumdar and Marco Pavone. *How Should a Robot Assess Risk? Towards an Axiomatic Theory of Risk in Robotics*. en. arXiv:1710.11040 [cs]. Nov. 2017. DOI: 10.48550/arXiv.1710.11040. URL: http://arxiv.org/abs/1710.11040 (visited on 03/14/2025).

[16] Andreas Labusch et al. “Worst case braking trajectories for robotic motion simulators”. en. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China: IEEE, May 2014, pp. 3297–3302. ISBN: 978-1-4799-3685-4. DOI: 10.1109/ICRA.2014.6907333. URL: http://ieeexplore.ieee.org/document/6907333/ (visited on 04/07/2025).

[17] Vishnu D. Sharma et al. “Risk-Aware Planning and Assignment for Ground Vehicles using Uncertain Perception from Aerial Vehicles”. en. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Las Vegas, NV, USA: IEEE, Oct. 2020, pp. 11763–11769. ISBN: 978-1-72816-212-6. DOI: 10.1109/IROS45743.2020.9341075. URL: https://ieeexplore.ieee.org/document/9341075/ (visited on 03/11/2025).

[18] Shangtong Zhang, Bo Liu, and Shimon Whiteson. *Mean-Variance Policy Iteration for Risk-Averse Reinforcement Learning*. en. arXiv:2004.10888 [cs]. Apr. 2022. DOI: 10.48550/arXiv.2004.10888. URL: http://arxiv.org/abs/2004.10888 (visited on 04/07/2025).

[19] Brandon D. Luders, Mangal Kothariyand, and Jonathan P. How. “Chance Constrained RRT for Probabilistic Robustness to Environmental Uncertainty”. In: *MIT web domain* (Aug. 2010). Accepted: 2011-12-13T20:37:01Z Publisher: American Institute of Aeronautics and Astronautics. URL: https://dspace.mit.edu/handle/1721.1/67648 (visited on 03/31/2025).

[20] Lars Blackmore, Masahiro Ono, and Brian C. Williams. “Chance-Constrained Optimal Path Planning With Obstacles”. In: *IEEE Transactions on Robotics* 27.6 (Dec. 2011), pp. 1080–1094. ISSN: 1941-0468. DOI: 10.1109/TRO.2011.2161160. URL: https://ieeexplore.ieee.org/document/5970128 (visited on 04/11/2025).

[21] Astghik Hakobyan, Gyeong Chan Kim, and Insoon Yang. “Risk-Aware Motion Planning and Control Using CVaR-Constrained Optimization”. In: *IEEE Robotics and Automation Letters* 4.4 (Oct. 2019). Conference Name: IEEE Robotics and Automation Letters, pp. 3924–3931. ISSN: 2377-3766. DOI: 10.1109/LRA.2019.2929980. URL: https://ieeexplore.ieee.org/document/8767973 (visited on 03/11/2025).

[22] Huan Xu, Constantine Caramanis, and Shie Mannor. “Optimization Under Probabilistic Envelope Constraints”. In: *Operations Research* 60.3 (June 2012). Publisher: INFORMS, pp. 682–699. ISSN: 0030-364X. DOI: 10.1287/opre.1120.1054. URL: https://pubsonline.informs.org/doi/10.1287/opre.1120.1054 (visited on 04/07/2025).

[23] Edmund R. Hunt, Conor B. Cullen, and Sabine Hauert. “Value at Risk strategies for robot swarms in hazardous environments”. In: *Unmanned Systems Technology XXIII*. Vol. 11758. SPIE, Apr. 2021, pp. 158–177. DOI: 10.1117/12.2585760. URL: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/11758/117580M/Value-at-Risk-strategies-for-robot-swarms-in-hazardous-environments/10.1117/12.2585760.full (visited on 04/11/2025).

[24] Prithvi Akella et al. *Risk-Aware Robotics: Tail Risk Measures in Planning, Control, and Verification*. arXiv:2403.18972 [cs]. Sept. 2024. DOI: 10.48550/arXiv.2403.18972. URL: http://arxiv.org/abs/2403.18972 (visited on 03/07/2025).

[25] Philippe Artzner et al. “Coherent Measures of Risk”. en. In: *Mathematical Finance* 9.3 (1999). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-9965.00068, pp. 203–228. ISSN: 1467-9965. DOI: 10.1111/1467-9965.00068. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-9965.00068 (visited on 04/02/2025).

[26] Sergey Sarykalin, Gaia Serraino, and Stan Uryasev. “Value-at-Risk vs. Conditional Value-at-Risk in Risk Management and Optimization”. en. In: *State-of-the-Art Decision-Making Tools in the Information-Intensive Age*. Ed. by Zhi-Long Chen et al. INFORMS, Sept. 2008, pp. 270–294. ISBN: 978-1-877640-23-0. DOI: 10.1287/educ.1080.0052. URL: http://pubsonline.informs.org/doi/abs/10.1287/educ.1080.0052 (visited on 04/11/2025).

[27] Xuru Yang et al. *Risk-Aware Non-Myopic Motion Planner for Large-Scale Robotic Swarm Using CVaR Constraints*. arXiv:2402.16690 [cs]. Aug. 2024. DOI: 10.48550/arXiv.2402.16690. URL: http://arxiv.org/abs/2402.16690 (visited on 03/06/2025).

[28] Jian Chu et al. “Risk-Aware Path Planning with Uncertain Human Interactions”. In: *2021 American Control Conference (ACC)*. ISSN: 2378-5861. May 2021, pp. 4225–4230. DOI: 10.23919/ACC50511.2021.9483314. URL: https://ieeexplore.ieee.org/document/9483314 (visited on 03/12/2025).

[29] Markus Ryll et al. “Efficient Trajectory Planning for High Speed Flight in Unknown Environments”. In: *2019 International Conference on Robotics and Automation (ICRA)*. ISSN: 2577-087X. May 2019, pp. 732–738. DOI: 10.1109/ICRA.2019.8793930. URL: https://ieeexplore.ieee.org/abstract/document/8793930 (visited on 04/09/2025).

[30] Aditya A. Paranjape et al. “Motion primitives and 3D path planning for fast flight through a forest”. en. In: *The International Journal of Robotics Research* 34.3 (Mar. 2015), pp. 357–377. ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364914558017. URL: https://journals.sagepub.com/doi/10.1177/0278364914558017 (visited on 03/31/2025).

[31] Mihir Dharmadhikari et al. “Motion Primitives-based Path Planning for Fast and Agile Exploration using Aerial Robots”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. ISSN: 2577-087X. May 2020, pp. 179–185. DOI: 10.1109/ICRA40945.2020.9196964. URL: https://ieeexplore.ieee.org/document/9196964 (visited on 03/31/2025).

[32] Huan Nguyen et al. *Motion Primitives-based Navigation Planning using Deep Collision Prediction*. arXiv:2201.03254 [cs]. May 2022. DOI: 10.48550/arXiv.2201.03254. URL: http://arxiv.org/abs/2201.03254 (visited on 03/28/2025).

[33] Steven M. LaValle. *Planning Algorithms*. en. 1st ed. Cambridge University Press, May 2006. ISBN: 978-0-521-86205-9 978-0-511-54687-7. DOI: 10.1017/CBO9780511546877. URL: https://www.cambridge.org/core/product/identifier/9780511546877/type/book (visited on 05/05/2025).

[34] Sertac Karaman and Emilio Frazzoli. “Sampling-based algorithms for optimal motion planning”. en. In: *The International Journal of Robotics Research* 30.7 (June 2011), pp. 846–894. ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364911406761. URL: https://journals.sagepub.com/doi/10.1177/0278364911406761 (visited on 03/31/2025).

[35] Artur Wolek et al. *Sampling-Based Risk-Aware Path Planning Around Dynamic Engagement Zones*. arXiv:2403.05480 [math]. Mar. 2024. DOI: 10.48550/arXiv.2403.05480. URL: http://arxiv.org/abs/2403.05480 (visited on 03/31/2025).

[36] Fei Meng et al. “NR-RRT: Neural Risk-Aware Near-Optimal Path Planning in Uncertain Nonconvex Environments”. In: *IEEE Transactions on Automation Science and Engineering* 21.1 (Jan. 2024). arXiv:2205.06951 [cs], pp. 135–146. ISSN: 1545-5955, 1558-3783. DOI: 10.1109/TASE.2022.3215562. URL: http://arxiv.org/abs/2205.06951 (visited on 03/31/2025).

[37] Chenning Yu and Sicun Gao. *Reducing Collision Checking for Sampling-Based Motion Planning Using Graph Neural Networks*. arXiv:2210.08864 [cs]. Oct. 2022. DOI: 10.48550/arXiv.2210.08864. URL: http://arxiv.org/abs/2210.08864 (visited on 05/05/2025).

[38] Ji Yin, Zhiyuan Zhang, and Panagiotis Tsiotras. *Risk-Aware Model Predictive Path Integral Control Using Conditional Value-at-Risk*. en. arXiv:2209.12842 [cs]. Sept. 2022. DOI: 10.48550/arXiv.2209.12842. URL: http://arxiv.org/abs/2209.12842 (visited on 05/16/2025).

[39] Ali Mohamed Ali, Hashim A. Hashim, and Chao Shen. *MPC Based Linear Equivalence with Control Barrier Functions for VTOL-UAVs*. en. arXiv:2404.09320 [eess]. Apr. 2024. DOI: 10.48550/arXiv.2404.09320. URL: http://arxiv.org/abs/2404.09320 (visited on 05/17/2025).

[40] Muhammad Kazim et al. “Recent advances in path integral control for trajectory optimization: An overview in theoretical and algorithmic perspectives”. In: *Annual Reviews in Control* 57 (Jan. 2024), p. 100931. ISSN: 1367-5788. DOI: 10.1016/j.arcontrol.2023.100931. URL: https://www.sciencedirect.com/science/article/pii/S1367578823000950 (visited on 05/16/2025).

[41] Yiming Ji et al. *Neural-Network-Driven Reward Prediction as a Heuristic: Advancing Q-Learning for Mobile Robot Path Planning*. en. arXiv:2412.12650 [cs]. Dec. 2024. DOI: 10.48550/arXiv.2412.12650. URL: http://arxiv.org/abs/2412.12650 (visited on 04/08/2025).

[42] Yulan Guo et al. “Deep Learning for 3D Point Clouds: A Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.12 (Dec. 2021), pp. 4338–4364. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2020.3005434. URL: https://ieeexplore.ieee.org/document/9127813 (visited on 04/24/2025).

[43] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[44] Daniel Maturana and Sebastian Scherer. “VoxNet: A 3D Convolutional Neural Network for real-time object recognition”. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2015, pp. 922–928. DOI: 10.1109/IROS.2015.7353481. URL: https://ieeexplore.ieee.org/document/7353481 (visited on 04/24/2025).

[45] Guan Pang and Ulrich Neumann. “3D point cloud object detection with multi-view convolutional neural network”. In: *2016 23rd International Conference on Pattern Recognition (ICPR)*. Dec. 2016, pp. 585–590. DOI: 10.1109/ICPR.2016.7899697. URL: https://ieeexplore.ieee.org/document/7899697 (visited on 04/24/2025).

[46] R. Qi Charles et al. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. en. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI: IEEE, July 2017, pp. 77–85. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.16. URL: http://ieeexplore.ieee.org/document/8099499/ (visited on 03/11/2025).

[47] Charles R. Qi et al. *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*. arXiv:1706.02413 [cs]. June 2017. DOI: 10.48550/arXiv.1706.02413. URL: http://arxiv.org/abs/1706.02413 (visited on 04/24/2025).

[48] Yue Wang et al. *Dynamic Graph CNN for Learning on Point Clouds*. arXiv:1801.07829 [cs]. June 2019. DOI: 10.48550/arXiv.1801.07829. URL: http://arxiv.org/abs/1801.07829 (visited on 04/24/2025).

[49] Hugues Thomas et al. *KPConv: Flexible and Deformable Convolution for Point Clouds*. arXiv:1904.08889 [cs]. Aug. 2019. DOI: 10.48550/arXiv.1904.08889. URL: http://arxiv.org/abs/1904.08889 (visited on 04/24/2025).

[50] Hengshuang Zhao et al. “Point Transformer”. en. In: 2021, pp. 16259–16268. URL: https://openaccess.thecvf.com/content/ICCV2021/html/Zhao_Point_Transformer_ICCV_2021_paper.html (visited on 04/24/2025).

[51] Radford Neal. “Bayesian Learning via Stochastic Dynamics”. In: *Advances in Neural Information Processing Systems*. Vol. 5. Morgan-Kaufmann, 1992. URL: https://papers.nips.cc/paper/1992/hash/f29c21d4897f78948b91f03172341b7b-Abstract.html (visited on 04/08/2025).

[52] Ánh Hoàng et al. “A Bayesian Neural Network-based Obstacle Avoidance Algorithm for an Educational Autonomous Mobile Robot Platform”. In: *Engineering, Technology and Applied Science Research* (Dec. 2023), pp. 12183–12189. DOI: 10.48084/etasr.6304.

[53] Ali Vaziri, Iman Askari, and Huazhen Fang. *Bayesian Inferential Motion Planning Using Heavy-Tailed Distributions*. arXiv:2503.22030 [cs] version: 1. Mar. 2025. DOI: 10.48550/arXiv.2503.22030. URL: http://arxiv.org/abs/2503.22030 (visited on 04/08/2025).

[54] Jinyeob Kim et al. *Belief Aided Navigation using Bayesian Reinforcement Learning for Avoiding Humans in Blind Spots*. arXiv:2403.10105 [cs]. Mar. 2024. DOI: 10.48550/arXiv.2403.10105. URL: http://arxiv.org/abs/2403.10105 (visited on 04/08/2025).

[55] Lei Shi, Cosmin Copot, and Steve Vanlanduit. “A Bayesian Deep Neural Network for Safe Visual Servoing in Human–Robot Interaction”. English. In: *Frontiers in Robotics and AI* 8 (June 2021). Publisher: Frontiers. ISSN: 2296-9144. DOI: 10.3389/frobt.2021.687031. URL: https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2021.687031/full (visited on 04/08/2025).

[56] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. *Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles*. arXiv:1612.01474 [stat]. Nov. 2017. DOI: 10.48550/arXiv.1612.01474. URL: http://arxiv.org/abs/1612.01474 (visited on 03/28/2025).

[57] Hao Mark Chen et al. *Enhancing Dropout-based Bayesian Neural Networks with Multi-Exit on FPGA*. en. arXiv:2406.14593 [cs]. June 2024. DOI: 10.48550/arXiv.2406.14593. URL: http://arxiv.org/abs/2406.14593 (visited on 04/08/2025).

[58] Wenbo Shao et al. *Uncertainty-Aware Prediction and Application in Planning for Autonomous Driving: Definitions, Methods, and Comparison*. arXiv:2403.02297 [cs] version: 1. Mar. 2024. DOI: 10.48550/arXiv.2403.02297. URL: http://arxiv.org/abs/2403.02297 (visited on 03/11/2025).

[59] Shiwei Lian and Feitian Zhang. *Improving Collision-Free Success Rate For Object Goal Visual Navigation Via Two-Stage Training With Collision Prediction*. en. arXiv:2502.13498 [cs]. Feb. 2025. DOI: 10.48550/arXiv.2502.13498. URL: http://arxiv.org/abs/2502.13498 (visited on 05/15/2025).

[60] Michał Czubenko and Zdzisław Kowalczuk. “A Simple Neural Network for Collision Detection of Collaborative Robots”. en. In: *Sensors* 21.12 (Jan. 2021). Number: 12 Publisher: Multidisciplinary Digital Publishing Institute, p. 4235. ISSN: 1424-8220. DOI: 10.3390/s21124235. URL: https://www.mdpi.com/1424-8220/21/12/4235 (visited on 05/15/2025).

[61] Ahmed Hussain Qureshi et al. “Motion Planning Networks: Bridging the Gap Between Learning-Based and Classical Motion Planners”. en. In: *IEEE Transactions on Robotics* 37.1 (Feb. 2021), pp. 48–66. ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2020.3006716. URL: https://ieeexplore.ieee.org/document/9154607/ (visited on 05/15/2025).

[62] Deval Shah and Tor M. Aamodt. "Collision Prediction for Robotics Accelerators". en. In: *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. Buenos Aires, Argentina: IEEE, June 2024, pp. 566–581. ISBN: 979-8-3503-2658-1. DOI: 10.1109/ISCA59077.2024.00048. URL: https://ieeexplore.ieee.org/document/10609658/ (visited on 05/15/2025).

[63] Antonio Loquercio et al. "DroNet: Learning to Fly by Driving". en. In: *IEEE Robotics and Automation Letters* 3.2 (Apr. 2018), pp. 1088–1095. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2018.2795643. URL: http://ieeexplore.ieee.org/document/8264734/ (visited on 05/15/2025).

[64] Hervé B. Olou et al. "FCPNet: A novel model to predict forward collision based-upon CNN". In: *2022 22nd International Conference on Control, Automation and Systems (ICCAS)*. ISSN: 2642-3901. Nov. 2022, pp. 1327–1332. DOI: 10.23919/ICCAS55662.2022.10003846. URL: https://ieeexplore.ieee.org/document/10003846 (visited on 05/15/2025).

[65] Hehe Fan and Yi Yang. *PointRNN: Point Recurrent Neural Network for Moving Point Cloud Processing*. en. arXiv:1910.08287 [cs]. Nov. 2019. DOI: 10.48550/arXiv.1910.08287. URL: http://arxiv.org/abs/1910.08287 (visited on 05/15/2025).

[66] Dário Pedro et al. "Collision Avoidance on Unmanned Aerial Vehicles Using Neural Network Pipelines and Flow Clustering Techniques". en. In: *Remote Sensing* 13.13 (Jan. 2021). Number: 13 Publisher: Multidisciplinary Digital Publishing Institute, p. 2643. ISSN: 2072-4292. DOI: 10.3390/rs13132643. URL: https://www.mdpi.com/2072-4292/13/13/2643 (visited on 05/15/2025).

[67] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. *ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst*. en. arXiv:1812.03079 [cs]. Dec. 2018. DOI: 10.48550/arXiv.1812.03079. URL: http://arxiv.org/abs/1812.03079 (visited on 05/15/2025).

[68] Aloukik Aditya et al. "Collision Detection: An Improved Deep Learning Approach Using SENet and ResNext". en. In: *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. arXiv:2201.04766 [cs]. Oct. 2021, pp. 2075–2082. DOI: 10.1109/SMC52423.2021.9659265. URL: http://arxiv.org/abs/2201.04766 (visited on 05/15/2025).

[69] Tuan Tran, Jory Denny, and Chinwe Ekenna. *Predicting Sample Collision with Neural Networks*. arXiv:2006.16868 [cs]. June 2020. DOI: 10.48550/arXiv.2006.16868. URL: http://arxiv.org/abs/2006.16868 (visited on 05/15/2025).

[70] Peng Wei et al. "Vision-Based 2D Navigation of Unmanned Aerial Vehicles in Riverine Environments with Imitation Learning". In: *Journal of Intelligent & Robotic Systems* 104 (Mar. 2022). DOI: 10.1007/s10846-022-01593-5.

[71] Lei Tai et al. "Socially Compliant Navigation Through Raw Depth Inputs with Generative Adversarial Imitation Learning". en. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, QLD: IEEE, May 2018, pp. 1111–1117. ISBN: 978-1-5386-3081-5. DOI: 10.1109/ICRA.2018.8460968. URL: https://ieeexplore.ieee.org/document/8460968/ (visited on 04/08/2025).

[72] Chengran Yuan et al. *DRAMA: An Efficient End-to-end Motion Planner for Autonomous Driving with Mamba*. en. arXiv:2408.03601 [cs]. Aug. 2024. DOI: 10.48550/arXiv.2408.03601. URL: http://arxiv.org/abs/2408.03601 (visited on 04/08/2025).

[73] Dhiraj Gandhi, Lerrel Pinto, and Abhinav Gupta. "Learning to fly by crashing". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ISSN: 2153-0866. Sept. 2017, pp. 3948–3955. DOI: 10.1109/IROS.2017.8206247. URL: https://ieeexplore.ieee.org/document/8206247 (visited on 04/08/2025).

[74] Gregory Kahn et al. *Uncertainty-Aware Reinforcement Learning for Collision Avoidance*. arXiv:1702.01182 [cs]. Feb. 2017. DOI: 10.48550/arXiv.1702.01182. URL: http://arxiv.org/abs/1702.01182 (visited on 04/07/2025).

[75] Dario Amodei et al. *Concrete Problems in AI Safety*. arXiv:1606.06565 [cs]. July 2016. DOI: 10.48550/arXiv.1606.06565. URL: http://arxiv.org/abs/1606.06565 (visited on 04/08/2025).

[76] Josh Tobin et al. *Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World*. en. arXiv:1703.06907 [cs]. Mar. 2017. DOI: 10.48550/arXiv.1703.06907. URL: http://arxiv.org/abs/1703.06907 (visited on 04/08/2025).

[77] Xue Bin Peng et al. "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization". en. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. arXiv:1710.06537 [cs]. May 2018, pp. 3803–3810. DOI: 10.1109/ICRA.2018.8460528. URL: http://arxiv.org/abs/1710.06537 (visited on 04/08/2025).

[78] Fereshteh Sadeghi and Sergey Levine. *CAD2RL: Real Single-Image Flight without a Single Real Image*. en. arXiv:1611.04201 [cs]. June 2017. DOI: 10.48550/arXiv.1611.04201. URL: http://arxiv.org/abs/1611.04201 (visited on 04/08/2025).

[79] Maymoonah Toubeh and Pratap Tokekar. *Risk-Aware Planning by Confidence Estimation using Deep Learning-Based Perception*. en. arXiv:1910.00101 [cs]. Sept. 2019. DOI: 10.48550/arXiv.1910.00101. URL: http://arxiv.org/abs/1910.00101 (visited on 04/08/2025).

[80] Jinglun Yu, Yuancheng Su, and Yifan Liao. "The Path Planning of Mobile Robot by Neural Networks and Hierarchical Reinforcement Learning". English. In: *Frontiers in Neurorobotics* 14 (Oct. 2020). Publisher: Frontiers. ISSN: 1662-5218. DOI: 10.3389/fnbot.2020.00063. URL: https://www.frontiersin.org/journals/neurorobotics/articles/10.3389/fnbot.2020.00063/full (visited on 04/08/2025).

[81] Ralf Römer et al. "Vision-Based Uncertainty-Aware Motion Planning based on Probabilistic Semantic Segmentation". In: *IEEE Robotics and Automation Letters* 8.11 (Nov. 2023). arXiv:2209.06936 [cs], pp. 7825–7832. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2023.3322899. URL: http://arxiv.org/abs/2209.06936 (visited on 03/31/2025).

[82] Chuyuan Tao et al. *Resilient Estimator-based Control Barrier Functions for Dynamical Systems with Disturbances and Noise*. en. arXiv:2407.00218 [eess]. June 2024. DOI: 10.48550/arXiv.2407.00218. URL: http://arxiv.org/abs/2407.00218 (visited on 05/17/2025).

[83] Richard Cheng et al. "End-to-End Safe Reinforcement Learning through Barrier Functions for Safety-Critical Continuous Control Tasks". en. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (July 2019). Number: 01, pp. 3387–3395. ISSN: 2374-3468. DOI: 10.1609/aaai.v33i01.33013387. URL: https://ojs.aaai.org/index.php/AAAI/article/view/4213 (visited on 05/17/2025).

[84] *Simulation — PX4 Guide (main)*. URL: https://docs.px4.io/main/en/simulation/ (visited on 04/14/2025).

[85] *Gazebo*. URL: https://classic.gazebosim.org/ (visited on 04/14/2025).

[86] *ROS: Home*. URL: https://www.ros.org/ (visited on 04/29/2025).

[87] *mavros - ROS Wiki*. URL: https://wiki.ros.org/mavros (visited on 04/29/2025).

[88] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018. URL: http://github.com/jax-ml/jax.

Appendices

I. Code Repository and Video Demos

The complete source code for reproducing all experiments presented in this thesis is available in a Google Drive repository. To access the code:

https://shorturl.at/FAZ7M

Please note that access permission is required. Kindly submit an access request through the link above.

Video demonstrations can be viewed from here:

https://shorturl.at/KsPnK

II. Planner Parameters

The table provided below details the planner parameters used in the experiments.

Configuration Parameters of the Risk-Aware Planner

Parameter	**Symbol**	**Value**
CEM iterations	N_{max}	20
Number of trajectories	N_{batch}	100
Number of elite samples	K	5
Number of cost-elite samples	N_{cost}	20
Safe distance threshold	d_{safe}	0.45 m
CBF decay rate	γ	0.95
Jerk limits	$[J_{min}, J_{max}]$	$[-1.68, 1.68]$ m/s^3
Time step	Δt	0.1 s
Planning horizon	T	1.0 s
Mean update rate	α_μ	0.6
Covariance update rate	α_Σ	0.6
Softmax temperature	λ	0.9
CVaR confidence level	α	0.98
Covariance regularization	ϵ	0.01

Non-exclusive licence to reproduce thesis and make thesis public

I, Kwasi Akuamoah Boateng

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   "Risk-Aware Planning on Pointclouds"

   supervised by Associate Prof. Arun Kumar Singh and Basant Sharma.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Kwasi Akuamoah Boateng
20.05.2025