SAFE ROBOT PLANNING AND CONTROL USING UNCERTAINTY-AWARE DEEP LEARNING

A Dissertation
Presented to
The Academic Faculty

By

David D. Fan

Towards the Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy in Robotics

Georgia Institute of Technology

August 2021

SAFE ROBOT PLANNING AND CONTROL USING UNCERTAINTY-AWARE DEEP LEARNING

Approved by:

Date Approved: July 14, 2021

By seeking and blundering we learn.

Johann Wolfgang von Goethe

To my parents and grandparents.

ACKNOWLEDGEMENTS

I would like to thank several individuals for their help and support in completing this thesis. First, I would like to thank my advisor Evangelos Theodorou, for his support, guidance, and patience throughout this journey. I am particularly grateful for his taking a chance on me, sending opportunities my way, and being patient and understanding in times of difficulty.

Heartfelt thanks goes to my co-advisor, Dr. Ali Agha, whose enthusiasm and mentorship has been invaluable in the latter part of my PhD, and who has helped to make my time spent at JPL most memorable. I would also like to thank the rest of the committee, Dr. Patricio Vela, Dr. Samuel Coogan, and Dr. Fumin Zhang for their comments, time, and insights.

I am indebted to the many collaborators and co-authors whom I have been lucky to know and learn from. I am grateful to Rohan Thakker for many interesting discussions and good advice, as well as to Kyohei Otsu, for his encouragement and guidance. I am grateful to the entire CoSTAR team at JPL, especially Ben Morrell, Angel Santamaria-Navarro, Brett Lopez, Sung Kim, Anushri Dixit, Amanda Bouman, Thomas Touma, Nikhilesh Alatur, Thomas Lew, Yuki Kubo, Jenn Nguyen, Jesus Tordesillas, Tara Bartlett, Leon Kim, Fadhil Ginting, and many others.

I would like to thank Dr. John Reeder for his support and mentorship in navigating the SMART program.

And, I am lucky to have known and worked with the members of the ACDS Lab, and am grateful for many interesting discussions and collaborations - Kaivalya Bakshi, Yunpeng Pan, Grady Williams, Keuntaek Lee, Marcus Pereira, Pat Wang, Bogdan Vlahov, Harleen Brar, and Manan Gandhi.

I would like to thank my friends and family for their love and support, especially my parents, Howard and Mary, and my brother, Philip, as well as Jason, for his encouragement and many taco runs.

Finally, I would like to thank Hannah, for always believing in me.

This research was partially carried out at the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the Year-round Internship Program and the National Aeronautics and Space Administration (80NM0018D0004).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

In order for robots to autonomously operate in novel environments over extended periods of time, they must learn and adapt to changes in the dynamics of their motion and the environment. Neural networks have been shown to be a versatile and powerful tool for learning dynamics and semantic information. However, there is reluctance to deploy these methods on safety-critical or high-risk applications, since neural networks tend to be black-box function approximators. Therefore, there is a need for investigation into how these machine learning methods can be safely leveraged for learning-based controls, planning, and traversability. The aim of this thesis is to explore methods for both establishing safety guarantees as well as accurately quantifying risks when using deep neural networks for robot planning, especially in high-risk environments. First, we consider uncertainty-aware Bayesian Neural Networks for adaptive control, and introduce a method for guaranteeing safety under certain assumptions. Second, we investigate deep quantile regression learning methods for learning time-and-state varying uncertainties, which we use to perform trajectory optimization with Model Predictive Control. Third, we introduce a complete framework for risk-aware traversability and planning, which we use to enable safe exploration of extreme environments. Fourth, we again leverage deep quantile regression and establish a method for accurately learning the distribution of traversability risks in these environments, which can be used to create safety constraints for planning and control.

CHAPTER 1
INTRODUCTION

1.1 Motivation

1.1.1 Current State of Practice

In 1970, Marvin Minsky, commonly regarded as the ”father of artificial intelligence”, told Life Magazine, “from three to eight years we will have a machine with the general intelligence of an average human being.” 50 years later, great advances have been made in various AI fields such as computer vision, natural language processing, and reinforcement learning. However, currently no field-deployed robot can be said to match the intelligence of a human being. Indeed, robots today tend to be over-engineered for safety, specificity, and reliability, and tend to be tele-operated by humans, rather than relying on intelligence to gracefully navigate hostile environments[1]. For robots to be truly autonomous in fields ranging from health care, exploration, rescue, monitoring, and surveying, they must possess enough intelligence to adapt and learn, grow from experience, yet at the same time assess risk and remain safe. Particularly when it comes to exploration of unknown environments, robots can not be pre-engineered for every situation and must adapt to new terrains, respond to changes in dynamics, and exercise caution when encountering novel dangers.

1.1.2 Neural Networks in Robotics

Neural networks have been shown to be a versatile and powerful tool for learning and adaptation, frequently out-performing any hand-engineered solutions [2, 3]. There is great potential, therefore, in using them to solve high-dimensional problems in robots, from learning dynamics to semantic understanding of the environment. However, there is currently reluctance to deploy these methods on safety-critical or high-risk applications, since neural networks tend to be black-box function approximators [4]. Therefore, with the motivation to outperform classical methods, there is a need for investigation into how neural networks can be used to solve safety-critical problems in robotics. One such way is to construct a network which *knows-what-it-knows*, i.e. it can give some measure of confidence or signal of trust that it is producing a reliable or meaningful output. Uncertainty quantification is the process of determining a quantitative description of the unknown of a system, and is a key element in the safe use of neural networks in safety critical applications [5].

Using neural networks to quantify uncertainty has been gaining recognition as an important field of study in its own right, perhaps beginning with Yarin Gal's seminal work on using dropout as a means for neural networks to perform Bayesian inference and model uncertainty [6]. Since then, a wide range of network architectures and training methods have been introduced, which offer varying degrees of efficiency and accuracy [7, 8, 9, 8, 10, 11, 6, 5, 12]. For any uncertainty-estimating network, the objective is for it to be *well-calibrated*, i.e., its quantification of uncertainty should match the true distribution of the underlying random variable it seeks to approximate [13].

As the field of uncertainty quantification in deep learning progresses in its own right, roboticists can make use of better and stronger tools to solve difficult problems that involve uncertainty. The immediate and gratifying result should be to endow robots with more adaptability, resilience, and intelligence in more challenging environments. The needs and challenges of this endeavor are elucidated by examining problems which lie at the boundaries of current robotic capability. Indeed, it is an aim of this thesis to address problems which lie at these boundaries, especially with respect to autonomous exploration of unknown, unstructured, and extreme environments, where both the magnitude and effects of uncertainty are strongly present (Figure 1.1).

1.1.3 Exploration of Extreme Environments

A good example of one such boundary is the DARPA Subterranean Challenge [14]. This three year challenge seeks to find an integrated solution for teams of heterogeneous robots to rapidly explore and search an unknown underground environment for objects of representative interest. The challenge presents three classes of underground environments to surmount: 1) tunnels, which are many kilometers in length and include highly constrained passages, multiple levels and vertical

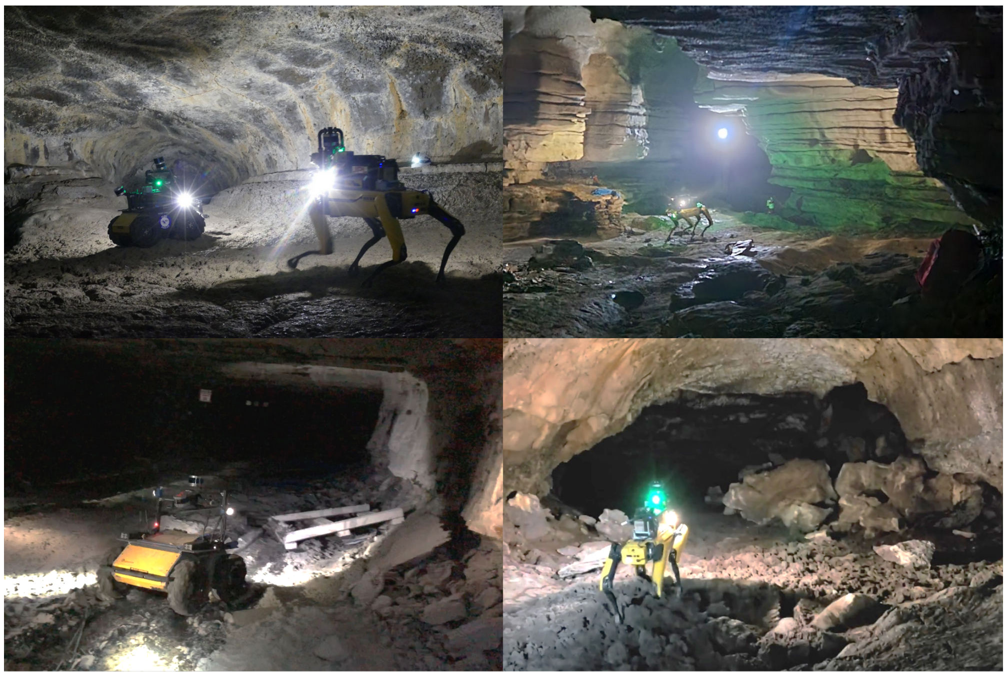


Figure 1.1: Robots used in this thesis autonomously exploring unknown and hazardous environments. Difficult-to-traverse terrain is often the cause of robot malfunction and damage. Top left: Clearpath's Husky and Boston Dynamic's Spot robots exploring an underground lava tube. Top right: Spot exploring a limestone cave. Bottom left: Husky exploring an industrial coal mine. Bottom right: Spot exploring a boulder field within a lava tube.

shafts; 2) urban environments, which include complex layouts, clutter, rubble, and hazards; and 3) caves, which contain a wide range of geological features which make navigation difficult for humans. Developing solutions to address novel problems presented by these environments forms the motivational backdrop to much of the work in this thesis.

Exploration of unknown, unstructured, and extreme environments is an important problem in robotics. Robots are meant to perform tasks which humans would rather not do. Such tasks include exploration of far-away worlds for furthering scientific understanding; search and rescue of humans from cave-ins, earthquakes, and natural disasters; surveying and mapping of ocean-beds for scientific or industrial purposes; and everyday navigation through a cluttered room to find the car keys. In all of these applications, certain common problems appear in which uncertainty plays a large role within localization, mapping, traversability, planning, and control. This thesis aims to point out and address some of these problems arising from uncertainty, with particular emphasis on control, planning, and traversability.

1.2 Objective and Scope of this Thesis

In this section, we introduce the research topics that will be covered in this thesis as well as the structure of this thesis.

1.2.1 Safety in Learning for Control

A main theme of this thesis is using learning methods to capture unknown dynamics, disturbances, and environmental factors in order to maintain safety. These objectives are often conflicting - using black box learning methods such as neural networks can lead to better representational power, but this often comes at a cost of safety guarantees. This thesis presents an approach to guaranteeing safety at the controls layer of a system, while enabling adaption and learning using neural networks. The notion of safety is considered from the perspective of Control Barrier Functions [15], which provide an approach to guaranteeing set invariance. The objective here is to construct a control policy which guarantees that the robot will never enter a predetermined, undesirable state. Because learned dynamics contain an element of uncertainty, this control policy must take a margin of safety proportional to this uncertainty into account [16].

Key to this approach is the use of uncertainty-aware Bayesian Neural Networks [17, 6], which are useful for capturing both *aleatoric* and *epistemic* uncertainty. Aleatoric uncertainty is uncertainty which is inherent to the variable of interest, while epistemic uncertainty arises from having incomplete information or insufficient data to characterize the variable of interest [11]. The use of Bayesian Neural Networks aims to characterize both types of uncertainty. As a result, we are able to make claims about the set-invariance of a controller even under insufficient data, as well as over the course of adaptation and learning.

1.2.2 Learning Uncertainty for Trajectory Optimization

Continuing the theme of robustness to uncertainty in control, this thesis also aims to apply a similar view to the level of trajectory optimization and planning. While controls in robotics tends to be concerned with one-timestep optimization at very high rates to track a given trajectory, the problem of trajectory optimization is concerned with finding the best trajectory to track. This thesis aims to leverage uncertainty-based learning methods to provide robust guarantees for a trajectory optimizer when optimizing over unknown or uncertain dynamics. Different from other methods which make parametric assumptions on the distribution of the underlying random variables, in this work, a non-parametric point of view is taken in which data is used to directly characterize the chance of exceeding some robust bounds with some probability [9].

Within the various classes of trajectory optimization methods, Model Predictive Control (MPC) plays a central role for its robustness, speed, efficiency, and ease of design [18]. When designing an MPC controller, one must take into account the notion of *recursive feasibility*. If an MPC controller is recursively feasible, then it is guaranteed to be able to find a feasible solution at each timestep, given a feasible solution at the previous timestep. When introducing neural network methods to characterize robust bound and uncertainty in the context of MPC therefore, we aim to take care to ensure recursive feasibility [19].

This thesis introduces a method for learning uncertainties in dynamics and/or controls using deep quantile regression, for use in an MPC controller, while maintaining recursive feasibility. Deep quantile regression is an approach to uncertainty quantification which uses a deep neural network to learn the quantiles of an underlying distribution [20]. This approach has advantages over other methods which propagate probability distributions through time by making strong assumptions on the distribution of the dynamics. In contrast, our method has no need for these assumptions, instead inferring margins of safety from the data directly.

1.2.3 Stochastic Traversability and Planning in Extreme Environments

As we move further up the planning stack, we arrive at the question of ”where” to move as opposed to ”how” to move, i.e. traversability. Classic methods for assessing traversability often rely on restrictive or simplistic assumptions of the environment: e.g. a known and well-mapped building, or a set of well-paved street roads. Moreover, these assessments are often binary, with no notion of risk or penalty to the robot’s health over different terrains. In reality, the situation is far more complex and requires careful treatment, especially in unknown, unmapped, and extreme terrain. Examples range from rovers exploring a new planet across varying terrain, to robots which search for trapped survivors in vast cave systems. In these types of environments, deciding ”where” to move involves a high degree of uncertainty and risk.

With this view, this thesis aims to establish a framework for risk-aware traversability and planning, in which the motion of a vehicle and its interaction with the terrain are considered jointly when assessing risk. Naturally, the concept of risk arises when thinking about robotic safety combined with uncertainty, and of particular interest is the concept of *tail-risk*. Tail-risks are important to consider because while we are concerned with keeping the robot within some safe set, we should also reason about what happens when the robot inevitably exits that safe set, and the consequences of doing so. One consistent and mathematically sound way to quantify and optimize for these consequences is through the concept of *risk measures* [21], and more specifically, the concepts of Value-at-Risk (VaR) and Conditional-Value-At-Risk (CVaR). Given a level of probability of risk $\alpha \in [0, 1]$, VaR is equivalent to the α-quantile for a given distribution, while CVaR is the expected value of the tail of the distribution outside the α-quantile bounds. Historically, these concepts come from the field of finance[1], but they find immediate and well-suited application to the evaluation of risk in robotics [23].

The contribution of this thesis is a hierarchical traversability assessment and planning framework which takes into account uncertainties arising from localization error, sensor sparsity and noise,

[1]As an aside, the 2008 financial crisis was perhaps triggered or exacerbated by the widespread use of VaR, which underestimated the risk of loss of portfolios of subprime mortgages [22].

and unknown ground-vehicle interactions. The framework is tested and field-hardened on a wide range of vehicles and in various extreme environments, including underground caves and mines, and large, hazardous industrial environments. This framework is used to compete in the DARPA Subterranean Challenge, as an integral part of the JPL Team CoSTAR's winning solution (2nd place Tunnel circuit, 1st place Urban circuit).

1.2.4 Learning Tail-risk Traversability

A central theme of this thesis is using learning to ensure safety for robot autonomy in challenging environments. The stochastic traversability and planning framework mentioned uses a model-based geometric risk assessment pipeline using 3D point cloud data. While this analysis performs very well on real systems in difficult environments, it does have a few drawbacks. First, it relies on having accurate estimation of localization uncertainty as well as having accurate sensor measurement models. This assumption is often difficult to satisfy in practice. Second, it is occasionally subject to certain artifacts from noise, occlusion, or localization error, and while these artifacts may occur infrequently, they can severely impede the progress of the robot. Third, it is computationally expensive, relying on 3D geometric analyses of hundreds of thousands of points. Fourth, it relies on assumptions of the underlying distribution of the traversability cost, which may be inaccurate or restrictive.

The final contribution of this thesis is to provide a method for addressing all these drawbacks through uncertainty-aware deep learning. We leverage the deep quantile regression theory mentioned previously and extend it to capturing CVaR traversability risk. The result is a CVaR-based traversability risk pipeline which directly maps point cloud data to the traversability risk, bypassing the need for computationally burdensome risk analyses. Moreover, this approach makes no assumptions on the distribution of traversability costs, gracefully handles errors in localization and sensors, and smooths out infrequently occurring artifacts.

1.2.5 Contributions and Outline

The aim of this thesis is to explore methods for both establishing safety guarantees as well as accurately quantifying risks when using deep neural networks for robot planning and control, especially in high-risk environments. The rest of the chapters are organized as follows:

- **Chapter 2: Bayesian Learning-Based Adaptive Control for Safety Critical Systems** This chapter presents the use of uncertainty-aware Bayesian Neural Networks for adaptive control, leveraging the theory of stochastic CLFs (Control Lyapunov Functions) and stochastic CBFs (Control Barrier Functions). Under reasonable assumptions, we guarantee stability and safety while adapting to unknown dynamics with probability 1. We demonstrate this architecture for high-speed terrestrial mobility targeting potential applications in safety-critical high-speed Mars rover missions. This chapter is based on the previously published article:
  - D. D. Fan, J. Nguyen, R. Thakker, N. Alatur, A.-a. Agha-mohammadi, and E. A. Theodorou, "Bayesian learning-based adaptive control for safety critical systems," in *International Conference on Robotics and Automation*, 2020
- **Chapter 3: Deep Learning Tubes for Tube MPC** This chapter presents the use of deep learning to obtain expressive and flexible models of how distributions of trajectories behave, which are then used for nonlinear Model Predictive Control (MPC). We introduce a deep quantile regression framework for control that enforces probabilistic quantile bounds and quantifies epistemic uncertainty. Using our method we explore three different approaches for learning tubes that contain the possible trajectories of the system, and demonstrate how to use each of them in a Tube MPC scheme. We prove these schemes are recursively feasible and satisfy constraints with a desired margin of probability. We present experiments in simulation on a nonlinear quadrotor system, demonstrating the practical efficacy of these ideas. This chapter is based on the previously published article:
  - D. D. Fan, A.-a. Agha-mohammadi, and E. A. Theodorou, "Deep learning tubes for

tube MPC," *Robotics: Science and Systems (RSS)*, 2020

- **Chapter 4: Stochastic Traversability Evaluation and Planning for Risk-aware Off-road Navigation** This chapter presents an approach for assessing traversability in challenging, off-road environments, and planning a safe, feasible, and fast trajectory in real-time. This approach relies on rapid uncertainty-aware mapping and traversability evaluation, tail risk assessment using the Conditional Value-at-Risk (CVaR), and efficient risk and constraint-aware kinodynamic motion planning using sequential quadratic programming-based (SQP) model predictive control (MPC). We analyze our method in simulation and validate its efficacy on wheeled and legged robotic platforms exploring extreme terrains including an underground lava tube. This chapter is based on the previously published article:

  – D. D. Fan, K. Otsu, Y. Kubo, A. Dixit, J. Burdick, and A.-A. Agha-Mohammadi, "STEP: Stochastic traversability evaluation and planning for safe off-road navigation," *Robotics: Science and Systems (RSS)*, 2021

- **Chapter 5: Costmap Learning for Risk-Aware Traversability in Challenging Environments** This chapter introduces a neural network architecture for robustly learning the distribution of traversability costs. The learning problem is presented from the perspective of learning tail-risks, i.e. the Conditional Value-at-Risk (CVaR). We show that this approach reliably learns the expected tail risk given a desired probability risk threshold between 0 and 1, producing a traversability costmap which is more robust to outliers, more accurately captures tail risks, and is more computationally efficient, when compared against baselines. We validate our method on data collected a legged robot navigating challenging, unstructured environments including an abandoned subway, limestone caves, and lava tube caves.

CHAPTER 2
BAYESIAN LEARNING-BASED ADAPTIVE CONTROL FOR SAFETY CRITICAL SYSTEMS

2.1 Summary

Deep learning has enjoyed much recent success, and applying state-of-the-art model learning methods to controls is an exciting prospect. However, there is a strong reluctance to use these methods on safety-critical systems, which have constraints on safety, stability, and real-time performance. We propose a framework which satisfies these constraints while allowing the use of deep neural networks for learning model uncertainties. Central to our method is the use of Bayesian model learning, which provides an avenue for maintaining appropriate degrees of caution in the face of the unknown. In the proposed approach, we develop an adaptive control framework leveraging the theory of stochastic CLFs (Control Lyapunov Functions) and stochastic CBFs (Control Barrier Functions) along with tractable Bayesian model learning via Gaussian Processes or Bayesian neural networks. Under reasonable assumptions, we guarantee stability and safety while adapting to unknown dynamics with probability 1. We demonstrate this architecture for high-speed terrestrial mobility targeting potential applications in safety-critical high-speed Mars rover missions.

2.2 Introduction

The rapid growth of Artificial Intelligence (AI) and Machine Learning (ML) disciplines has created a tremendous impact in engineering disciplines, including finance, medicine, and general cyber-physical systems. The ability of ML algorithms to learn high dimensional dependencies has expanded the capabilities of traditional disciplines and opened up new opportunities towards the development of decision making systems which operate in complex scenarios. Despite these recent successes [27], there is low acceptance of AI and ML algorithms to safety-critical domains,

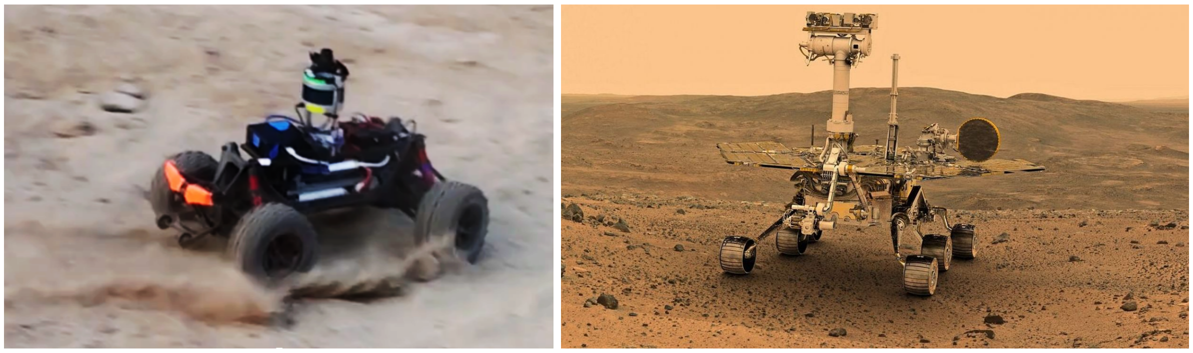


Figure 2.1: The left image depicts a 1/5th scale RC car platform driving at the Mars Yard at JPL; and the right is a platform from the Mars Explore Rover (MER) mission.

including human-centered robotics, and particularly in the flight and space industries. For example, both recent and near-future planned Mars rover missions largely rely on daily human decision making and piloting, due to a very low acceptable risk for trusting black-box autonomy algorithms. Therefore there is a need to develop computational tools and algorithms that bridge two worlds: the canonical structure of control theory, which is important for providing guarantees in safety-critical applications, and the data driven abstraction and representational power of machine learning, which is necessary for adapting the system to achieve resiliency against unmodeled disturbances.

Towards this end, we propose a novel, lightweight framework for Bayesian adaptive control for safety critical systems, which we call BALSA (BAyesian Learning-based Safety and Adaptation). This framework leverages ML algorithms for learning uncertainty representations of dynamics which in turn are used to generate sufficient conditions for stability using stochastic CLFs and safety using stochastic CBFs. Treating the problem within a stochastic framework allows for a cleaner and more optimal approach to handling modeling uncertainty, in contrast to deterministic, discrete-time, or robust control formulations. We apply our framework to the problem of high-speed agile autonomous vehicles, a domain where learning is especially important for dynamics which are complex and difficult to model (e.g., fast autonomous driving over rough terrain). Potential Mars Sample Return (MSR) missions are one example in this domain. Current Mars rovers (i.e., Opportunity and Curiosity) have driven on average $\sim$3km/year [28, 29]. In contrast, if MSR launches in 2028, then the rover has only 99 sols ($\sim$102 days) to complete potentially 10km [30, 31]. After factoring in the intermittent and heavily delayed communications to earth, the need for *adaptive*, high-speed autonomous mobility could be crucial to mission success.

Along with the requirements for safety and adaptation, computational efficiency is of paramount importance for real systems. Hardware platforms often have severe power and weight requirements, which significantly reduce their computational power. Probabilistic learning and control over deep Bayesian models is a computationally intensive problem. In contrast, we shorten the planning horizon and rely on a high-level, lower fidelity planner to plan desired trajectories. Our method then guarantees safe trajectory tracking behavior, even if the given trajectory is not safe. This frees up the computational budget for other tasks, such as online model training and inference.

2.2.1 Related Work

Machine-learning based planning and control is a quickly growing field. From Model Predictive Control (MPC) based learning [32, 33], safety in reinforcement learning [34], belief-space learning and planning [35], to imitation learning [36], these approaches all demand considerations of safety under learning [37, 38, 39, 40]. Closely related to our work is Gaussian Process-based Bayesian Model Reference Adaptive Control (GP-MRAC) [41], where modeling error is approximated with a Gaussian Process (GP). However, computational speed of GPs scales poorly with the amount of data ($\mathcal{O}(N^3)$), and sparse approximations lack representational power. Another closely related work is that of [42], who showed how to formulate a robust CLF which is tolerant to bounded model error. Extensions to robust CBFs were given in [43]. A stated drawback of this approach is the conservative nature of the bounds on the model error. In contrast, we incorporate model learning into our formulation, which allows for more optimal behavior, and leverage stochastic CLF and CBF theory to guarantee safety and stability with probability 1. Other related works include [44], which uses GPs in CBFs to learn the drift term in the dynamics $f(x)$, but uses a discrete-time, deterministic formulation. [45] combined L1 adaptive control and CLFs. Learning in CLFs and CBFs using adaptive control methods (including neuro-adaptive control) has been considered in several works, e.g. [46, 47, 48, 49].

2.2.2 Contributions

- Here we take a unique approach to address the aforementioned issues, with the requirements of 1) adaptation to changes in the environment and the system, 2) adaptation which can take into account high-dimensional data, 3) guaranteed safety during adaptation, 4) guaranteed stability during adaptation and convergence of tracking errors, 5) low computational cost and high control rates. Our contributions are fourfold: First, we introduce a Bayesian adaptive control framework which explicitly uses the model uncertainty to guarantee stability, and is agnostic to the type of Bayesian model learning used. Second, we extend recent stochastic safety theory to systems with *switched* dynamics to guarantee safety with probability 1. In contrast to adaptive control, switching dynamics are used to account for model updates which may only occur intermittently. Third, we combine these approaches in a novel online-learning framework (BALSA). Fourth, we compare the performance of our framework using different Bayesian model learning and uncertainty quantification methods. Finally, we apply this framework to a high-speed driving task on rough terrain using an Ackermann-steering vehicle and validate our method on both simulation and hardware experiments.

2.3 Safety and Stability under Model Learning via Stochastic CLF/CBFs

Consider a stochastic system with SDE (stochastic differential equation) dynamics:

$$\mathrm{d}x_1 = x_2 \mathrm{d}t, \quad \mathrm{d}x_2 = (f(x) + g(x)u)\mathrm{d}t + \Sigma(x)\mathrm{d}\xi(t) \tag{2.1}$$

where $x_1, x_2 \in \mathbb{R}^n$, $x = [x_1, x_2]^\intercal$, the controls are $u \in \mathbb{R}^n$, the diffusion is $\Sigma(x) \in \mathbb{R}^{n\times n}$, and $\xi(t) \in \mathbb{R}^n$ is a zero-mean Wiener process. For simplicity we restrict our analysis to systems of this form, but emphasize that our results are extensible to systems of higher relative degree [50], as well as hybrid systems with periodic orbits [51]. A wide range of nonlinear control-affine systems in robotics can be transformed into this form. In general, on a real system, f, g, and Σ may not be fully known. We assume $g(x)$ is known and invertible, which makes the analysis more tractable. It will be interesting in future work to extend our approach to unknown, non-invertible control gains, or non-control affine systems (e.g. $\dot{x} = f(x, u)$). Let $\hat{f}(x)$ be a given approximate model of $f(x)$. We formulate a pre-control law with pseudo-control $\mu \in \mathbb{R}^n$:

$$u = g(x)^{-1}(\mu - \hat{f}(x)) \tag{2.2}$$

which leads to the system dynamics being

$$\mathrm{d}x_1 = x_2\mathrm{d}t, \quad \mathrm{d}x_2 = (\mu + \Delta(x))\mathrm{d}t + \Sigma(x)\mathrm{d}\xi(t) \tag{2.3}$$

where $\Delta(x) = f(x) - \hat{f}(x)$ is the modeling error, with $\Delta(x) \in \mathbb{R}^n$.

Suppose we are given a reference model and reference control from, for example, a path planner:

$$\mathrm{d}x_{1rm} = x_{2rm}\mathrm{d}t, \quad \mathrm{d}x_{2rm} = f_{rm}(x_{rm}, u_{rm})\mathrm{d}t$$

The utility of the methods outlined in this work is for adaptive tracking of this given trajectory with guaranteed safety and stability. We assume that f_{rm} is continuously differentiable in x_{rm}, u_{rm}. Further, u_{rm} is bounded and piecewise continuous, and that x_{rm} is bounded for a bounded u_{rm}. Define the error $e = x - x_{rm}$. We split the pseudo-control input into four separate terms:

$$\mu = \mu_{rm} + \mu_{pd} + \mu_{qp} - \mu_{ad} \tag{2.4}$$

where we assign $\mu_{rm} = \mathrm{d}x_{2rm}$ and μ_{pd} to a PD controller:

$$\mu_{pd} = [-K_P - K_D]e \tag{2.5}$$

Additionally, we assign μ_{qp} as a pseudo-control which we optimize for and μ_{ad} as an adaptive element which will cancel out the model error. Then we can write the dynamics of the model error e as:

$$\begin{aligned}\mathrm{d}e = \begin{bmatrix}\mathrm{d}e_1 \\ \mathrm{d}e_2\end{bmatrix} &= \begin{bmatrix}0 & I \\ -K_P & -K_D\end{bmatrix} e\mathrm{d}t + \begin{bmatrix}0 \\ I\end{bmatrix} \big((\mu_{qp} - \mu_{ad} + \Delta(x))\mathrm{d}t + \Sigma(x)\mathrm{d}\xi(t)\big) \\ &= (Ae + G(\mu_{qp} - \mu_{ad} + \Delta(x)))\mathrm{d}t + G\Sigma(x)\mathrm{d}\xi(t) \end{aligned} \tag{2.6}$$

where the matrices A and G are used for ease of notation. The gains K_D, K_P should be chosen such that A is Hurwitz. When $\mu_{ad} = \Delta(x)$, the drift modeling error term is canceled out from the error dynamics.

Next, we require a method for learning or approximating the drift and diffusion terms $\Delta(x)$ and $\Sigma(x)$. Such methods include Bayesian SDE approximation methods [52], Neural-SDEs [53], or differential GP flows [54], to name a few. This model should *know what it doesn't know* [55], and should capture both the *epistemic* uncertainty of the model, i.e., the uncertainty from lack of data, as well as the *aleatoric* uncertainty, i.e., the uncertainty inherent in the system [56]. We expect that these methods will continue to be improved by the community. We can use the second equation in (2.3) to generate data points to use for learning these terms in the SDE. In discrete time, the learning problem is formulated as finding a mapping from input data $\bar{X}_t = x(t)$ to output data $\bar{Y}_t = (x_2(t+dt)-x_2(t))/dt-(\hat{f}(x(t))+g(x(t))u(t))$. Given the i^{th} dataset $\mathcal{D}_i = \{\bar{X}_t, \bar{Y}_t\}_{t=0,dt,\dots,t_i}$ with $i \in \mathbb{N}$, we can construct the i^{th} model $\{m_i(x), \sigma_i(x)\}$, where $m_i(x)$ approximates the drift term $\Delta(x)$ and $\sigma_i(x)$ approximates the diffusion term $\Sigma(x)$. Note that we do not require updating the model at each timestep, which significantly reduces computational load requirements and allows for training more expressive models (e.g., neural networks).

In practical terms, in this work we opt for an approximate method for learning $\{m_i(x), \sigma_i(x)\}$, in which we view each data point in $\mathcal{D}_i$ as an independently and identically distributed sample, and set up a single timestep Bayesian regression problem, in which we model $\Delta(x)$ as a multivariate Gaussian random variable, i.e. $\bar{\Delta}_i(x) \sim \mathcal{N}(m_i(x), \sigma_i(x))$. This approximation ignores the SDE nature of (2.3) and will not be a faithful approximation (See [57] for insightful comments on this problem). However, until Bayesian SDE approximation methods improve, we believe this approach to be reasonable in practice. Methods for producing reliable confidence bounds include a large class of Bayesian neural networks ([7, 58, 6]), Gaussian Processes or its many approximate variants ([59, 60]), and many others. We compare several methods in our experimental results. We leave a more principled learning approach using Bayesian SDE learning methods for future work.

After obtaining the joint model $\{m_i(x), \sigma_i(x)\}$, Equation (2.6) can be written as the following switching SDE:

$$\mathrm{d}e = (Ae + G(\mu_{qp} + \varepsilon_i^m(x))\mathrm{d}t + G\sigma_i(x)\mathrm{d}\xi(t) \tag{2.7}$$

with $e(0) = x(0) - x_{rm}(0)$ and where $\varepsilon_i^m(x) = m_i(x) - \Delta(x)$. $i \in \mathbb{N}$ is a switching index which updates each time the model is updated. The main problem which we address is how to find a pseudo-control μ_{qp} which provably drives the tracking error to 0 while simultaneously guaranteeing safety.

Since $\Delta(x)$ is not known a priori, one approach is to assume that $\|\varepsilon_i^m(x)\|$ is bounded by some known term. The size of this bound will depend on the type of model used to represent the uncertainty, its training method, and the distribution of the data $\mathcal{D}_i$. See [41] for such an analysis for sparse online Gaussian Processes. For neural networks in general there has been some work on analyzing these bounds [61, 62]. For simplicity, let us assume the modeling error $\epsilon_i^m(x) = 0$, and instead rely on $\sigma_i(x)$ to fully capture any remaining modeling error in the drift. Then we have the following dynamics:

$$\mathrm{d}e = (Ae + G\mu_{qp})\mathrm{d}t + G\sigma_i(x)\mathrm{d}\xi(t) \tag{2.8}$$

with $e(0) = x(0) - x_{rm}(0)$. This is valid as long as $\sigma_i(x)$ captures both the epistemic and aleatoric uncertainty accurately. Note also that if the bounds on $\|\varepsilon_i^m(x)\|$ are known, then our results are easily extensible to this case via (2.7).

2.3.1 Stochastic Control Lyapunov Functions for Switched Systems

We establish sufficient conditions on μ_{qp} to guarantee convergence of the error process $e(t)$ to 0. The result is a linear constraint similar to deterministic CLFs (e.g., [43]). The difference here is the construction a stochastic CLF condition for switched systems. The switching is needed to account for online updates to the model as more data is accumulated.

In general, consider a switched SDE of Itô type [63] defined by:

$$dX(t) = a(t, X(t))dt + \sigma_i(t, X(t))d\xi(t) \tag{2.9}$$

where $X \in \mathbb{R}^{n_1}$, $\xi(t) \in \mathbb{R}^{n_2}$ is a Wiener process, $a(t, X)$ is a $\mathbb{R}^{n_1}$-vector function, $\sigma_i(t, X)$ a

$n_1 \times n_2$ matrix, and $i \in \mathbb{N}$ is a switching index. The switching index may change a finite number of times in any finite time interval. For each switching index, a and σ must satisfy the Lipschitz condition $\|a(t,x) - a(t,y)\| + \|\sigma_i(t,x) - \sigma_i(t,y)\| \leq L\|x-y\|, \forall x, y \in D$ with D compact. Then the solution of (2.9) is a continuous Markov process.

Definition 2.3.1. *$X(t)$ is said to be* exponentially mean square ultimately bounded uniformly in *i if there exists positive constants K, c_0, τ such that for all t, X_0, i, we have that $\mathbb{E}_{X_0}\|X(t)\|^2 \leq K + c_0\|X_0\|^2 e^{-\tau t}$.*

We first restate the following theorem from [41]:

Theorem 2.3.1. *Let $X(t)$ be the process defined by the solution to (2.9), and let $V(t,X)$ be a function of class $\mathcal{C}^2$ with respect to X, and class $\mathcal{C}^1$ with respect to t. Denote the Itô differential generator by $\mathcal{L}$. If 1) $-\alpha_1 + c_1\|X\|^2 \leq V(t,X) \leq c_3\|X\|^2 + \alpha_2$ for real $\alpha_1, \alpha_2, c_1 > 0$; and 2) $\mathcal{L}V(t,X) \leq \beta_i - c_2 V(t,X)$ for real $\beta_i, c_2 > 0$, and all i; then the process $X(t)$ is exponentially mean square ultimately bounded uniformly in i. Moreover, $K = \frac{\alpha_2}{c_1} + \max_i(\frac{|\beta_i|}{c_1 c_2} + \frac{\alpha_1}{c_1})$, $c_0 = \frac{c_3}{c_1}$, and $\tau = c_2$.*

Proof. See [41] Theorem 1. □

We use Theorem 2.3.1 to derive a stochastic CLF sufficient condition on μ_{qp} for the tracking error $e(t)$. Consider the stochastic Lyapunov candidate function $V(e) = \frac{1}{2}e^\intercal Pe$ where P is the solution to the Lyapunov equation $A^\intercal P + PA = -Q$, where Q is any symmetric positive-definite matrix.

Theorem 2.3.2. *Let $e(t)$ be the switched stochastic process defined by (2.8), and let $\epsilon > 0$ be a positive constant. Suppose for all t, μ_{qp} and the relaxation variable $d_i^1 \in \mathbb{R}$ satisfy the inequality:*

$$\Psi_i^0 + \Psi^1 \mu_{qp} \leq d_i^1 \tag{2.10}$$

$$\Psi_i^0 = -\frac{1}{2}e^\intercal Qe + \frac{1}{\epsilon}V(e) + \frac{1}{2}\text{tr}(G\sigma_i\sigma_i^\intercal G^\intercal P)$$

$$\Psi^1 = e^\intercal PG.$$

Then $e(t)$ is exponentially mean-square ultimately bounded uniformly in i. Moreover if (2.10) is satisfied with $d_i^1 < 0$ for all i, then $e(t) \to 0$ exponentially in the mean-squared sense.

Proof. The Lyapunov candidate function $V(e)$ is bounded above and below by $\frac{1}{2}\lambda_{min}(P)\|e\|^2 \leq V(e(t)) \leq \frac{1}{2}\lambda_{max}(P)\|e\|^2$. We have the following Itô differential of the Lyapunov candidate:

$$\begin{aligned}\mathcal{L}V(e) &= \sum_j \frac{\partial V(e)}{\partial e_j} A e_j + \frac{1}{2}\sum_{j,k}[G\sigma_i\sigma_i^\intercal G^\intercal]_{jk}\frac{\partial^2 V(e)}{\partial e_k \partial e_j} \\ &= -\frac{1}{2}e^\intercal Q e + e^\intercal PG\mu_{qp} + \frac{1}{2}tr(G\sigma_i\sigma_i^\intercal G^\intercal P).\end{aligned} \tag{2.11}$$

Rearranging, (2.10) becomes $\mathcal{L}V(e) \leq -\frac{1}{\epsilon}V(e)$. Setting $\alpha_1 = \alpha_2 = 0, \beta_i = d_i^1, c_1 = \frac{1}{2}\lambda_{min}(P), c_2 = \frac{1}{\epsilon}, c_3 = \frac{1}{2}\lambda_{max}(P)$, we see that the conditions for Theorem 2.3.1 are satisfied and $e(t)$ is exponentially mean square ultimately bounded uniformly in i. Moreover,

$$\mathbb{E}_{e_0}\|e(t)\|^2 \leq \kappa(P)\|e_0\|^2 e^{-\frac{t}{\epsilon}} + \max_i\left(\frac{|d_i^1|}{4\lambda_{min}(P)\lambda_{max}(P)}\right) \tag{2.12}$$

where $\kappa(P)$ is the condition number of the matrix P. Therefore if $d_i^1 < 0$ for all i, $e(t)$ converges to 0 exponentially in the mean square sense. $\square$

The relaxation variable d_i^1 allows us to find solutions for μ_{qp} which may not always strictly satisfy a Lyapunov stability criterion $\mathcal{L}V \leq 0$. This allows us to incorporate additional constraints on μ_{qp} at the cost of losing convergence of the error e to 0. Fortunately, the error will remain bounded by the largest d_i^1. In practice we re-optimize for a new d_i^1 at each timestep. This does not affect the result of Theorem 2.3.2 as long as we re-optimize a finite number of times for any given finite interval.

One highly relevant set of constraints we want to satisfy are control constraints $Hu \leq b$, where $H \in \mathbb{R}^{n_c} \times \mathbb{R}^n$ is a matrix and $b \in \mathbb{R}^{n_c}$ is a vector. Let $\mu_d = \mu_{rm} + \mu_{pd} - \mu_{ad}$. Recall the pre-control law (2.2). Then the control constraint is:

$$Hg^{-1}(x)\mu_{qp} \leq H\hat{g}^{-1}(x)(\mu_d - \hat{f}(x)) + b. \tag{2.13}$$

Next we formulate additional constraints to guarantee safety.

2.3.2 Stochastic Control Barrier Functions for Switched Systems

We leverage recent results on stochastic control barrier functions [64] to derive constraints linear in μ_{qp} which guarantee the process $x(t)$ satisfies a safety constraint, i.e., $x(t) \in \mathcal{C}$ for all t. The set $\mathcal{C}$ is defined by a locally Lipschitz function $h : \mathbb{R}^n \to \mathbb{R}$ as $\mathcal{C} = \{x : h(x) \geq 0\}$ and $\partial\mathcal{C} = \{x : h(x) = 0\}$. We first extend the results of [64] to switched stochastic systems.

Definition 2.3.2. *Let $X(t)$ be a switched stochastic process defined by (2.9). Let the function $B : \mathbb{R}^n \to \mathbb{R}$ be locally Lipschitz and twice-differentiable on* $\text{int}(\mathcal{C})$*. If there exists class-K functions γ_1 and γ_2 such that for all X, $1/\gamma_1(h(X)) \leq B(X) \leq 1/\gamma_2(h(X))$, then $B(x)$ is called a* candidate control barrier function.

Definition 2.3.3. *Let $B(x)$ be a candidate control barrier function. If there exists a class-K function γ_3 such that $\mathcal{L}B(X) \leq \gamma_3(h(X))$, then $B(x)$ is called a* control barrier function (CBF).

Theorem 2.3.3. *Suppose there exists a CBF for the switched stochastic process $X(t)$ defined by (2.9). If $X_0 \in \mathcal{C}$, then for all t, $Pr(X(t) \in \mathcal{C}) = 1$.*

Proof. [64] Theorem 1 provides a proof of the result for non-switched stochastic processes. Let t_i denote the switching times of $X(t)$, i.e., when $t \in [0, t_0)$, the process $X(t)$ has diffusion matrix $\sigma_0(X)$, and when $t \in [t_{i-1}, t_i)$ for $i > 0$, the process $X(t)$ has diffusion matrix $\sigma_i(X)$. If $X_0 \in \mathcal{C}$, then $X_t \in \mathcal{C}$ for all $t \in [0, t_0)$ with probability 1 since the process $X(t)$ does not switch in the time interval $t \in [0, t_0)$. By similar argument for any $i > 0$ if $X_{t_{i-1}} \in \mathcal{C}$ then $X_t \in \mathcal{C}$ for all $t \in [t_{i-1}, t_i)$ with probability 1. This also implies that $X_{t_i} \in \mathcal{C}$, since $X(t)$ is a continuous Markov process. Then $X_t \in \mathcal{C}$ for all $t \in [t_i, t_{i+1})$ with probability 1. Then by induction, for all t, $Pr(X(t) \in \mathcal{C}) = 1$. $\square$

Next, we establish a linear constraint condition sufficient for μ_{qp} to guarantee safety for (2.8). Rewrite (2.8) in terms of $x(t)$ as:

$$\mathrm{d}x = (A_0 x + G(\mu_d + \mu_{qp}))\mathrm{d}t + G\sigma_i(x)\mathrm{d}\xi(t) \tag{2.14}$$

$$A_0 = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix}, \quad \mu_d = \mu_{rm} + \mu_{pd} - \mu_{ad}.$$

Theorem 2.3.4. *Let $x(t)$ be a switched stochastic process defined by (2.15). Let $B(x)$ be a candidate control barrier function. Let γ_3 be a class-K function. Suppose for all t, μ_{qp} satisfies the inequality:*

$$\Phi_i^0 + \Phi^1 \mu_{qp} \leq 0 \tag{2.15}$$

$$\Phi_i^0 = \frac{\partial B}{\partial x}^{\intercal}(A_0 x + G\mu_d) - \gamma_3(h(x)) + \frac{1}{2}\text{tr}(G\sigma_i\sigma_i^{\intercal}G^{\intercal}\frac{\partial^2 B}{\partial x^2})$$

$$\Phi^1 = \frac{\partial B}{\partial x}^{\intercal} G \tag{2.16}$$

Then $B(x)$ is a CBF and (2.16) is a sufficient condition for safety, i.e., if $x_0 \in \mathcal{C}$, then $x(t) \in \mathcal{C}$ for all t with probability 1.

Proof. We have the following Itô differential of the CBF candidate $B(x)$:

$$\mathcal{L}B(x) = \frac{\partial B}{\partial x}^{\intercal}(A_0 x + G(\mu_d + \mu_{qp})) + \frac{1}{2}tr(G\sigma_i\sigma_i^{\intercal}G^{\intercal}\frac{\partial^2 B}{\partial x^2}). \tag{2.17}$$

Rearranging (2.16) it is clear that $\mathcal{L}B(x) \leq \gamma_3(h(x))$. Then $B(x)$ is a CBF and the result follows from Theorem 2.3.3. □

2.3.3 Safety and Stability under Model Adaptation

We can now construct a CLF-CBF Quadratic Program (QP) in terms of μ_{qp} incorporating both the adaptive stochastic CLF and CBF conditions, along with control limits (Equation (2.18)):

$$\begin{aligned} \arg\min_{\mu_{qp}, d_1, d_2} \quad & \mu_{qp}^{\intercal}\mu_{qp} + p_1 d_1^2 + p_2 d_2^2 && (2.18) \\ s.t. \quad & \Psi_i^0 + \Psi^1 \mu_{qp} \leq d_1 && (\textbf{Adaptive CLF}) \\ & \Phi_i^0 + \Phi^1 \mu_{qp} \leq d_2 && (\textbf{Adaptive CBF}) \\ & Hg^{-1}(x)\mu_{qp} \leq Hg^{-1}(x)(\mu_d - \hat{f}(x)) + b \end{aligned}$$

In practice, several modifications to this QP are often made ([50],[65]). In addition to a relaxation term for the CLF in Theorem 2.3.2, we also include a relaxation term d^2 for the CBF. This helps to ensure the QP is feasible and allows for slowing down as much as possible when the safety constraint cannot be avoided due to control constraints, creating, e.g., lower impact collisions.

Safety is still guaranteed as long as the relaxation term is less than 0. For an example of guaranteed safety in the presence of this relaxation term see [43], also see [47] for an approach to handling safety with control constraints. The emphasis of this work is on guaranteeing safety in the presence of adaptation so we leave these considerations for future work. Our entire framework is outlined in Algorithm 1.

Algorithm 1: BAyesian Learning-based Safety and Adaptation (BALSA)

```
1  Require: Prior model $\hat{f}(x)$, known $g(x)$, reference trajectory $x_{rm}$, choice of modeling
      algorithm $\bar{\Delta}_i(x) \sim \mathcal{N}(m_i(x), \sigma_i(x))$, $dt$, $A$, $Hu \leq b$.
2  Initialize: $i = 0$, Dataset $\mathcal{D}_0 = \emptyset$, $t = 0$, solve $P$
3  while true do
4      Obtain $\mu_{rm} = \mathrm{d}x_{2rm}(t)$ and compute $\mu_{pd}$
5      Compute model error and uncertainty $\mu_{ad} = m_i(x(t))$, and $\sigma_i(x(t))$
6      $\mu_{qp} \leftarrow$ Solve QP (2.18)
7      Set $u(t) = g(x)^{-1}(\mu_{rm} + \mu_{pd} + \mu_{qp} - \mu_{ad} - \hat{f}(x))$
8      Apply control $u(t)$ to system.
9      Step forward in time $t \leftarrow t + dt$.
10     Append new data point to database:
11     $\bar{X}_t = [x(t)]$, $\bar{Y}_t = (x_2(t+dt) - x_2(t))/dt - (\hat{f}(x(t)) + g(x(t)u(t))$.
12     $\mathcal{D}_i \leftarrow \mathcal{D}_i \cup \{\bar{X}_t, \bar{Y}_t\}$
13     if updateModel then
14         Update model $\bar{\Delta}_i(x, \mu)$ with database $\mathcal{D}_i$
15         $\mathcal{D}_{i+1} \leftarrow \mathcal{D}_i$, $i \leftarrow i + 1$
```

2.4 Application to Fast Autonomous Driving

In this section we validate BALSA on a kinematic bicycle model for car-like vehicles. We model the state $x = [p_x, p_y, \theta, v]^\intercal$ as position in x and y, heading, and velocity respectively, with dynamics $\dot{x} = [v\cos(\theta), v\sin(\theta), v\tan(\psi)/L, a]^\intercal$. where a is the input acceleration, L is the vehicle length, and ψ is the steering angle. We employ a simple transformation to obtain dynamics in the form of (2.1). Let $z = [z_1, z_2, z_3, z_4]^\intercal$ where $z_1 = p_x$, $z_2 = p_y$, $z_3 = \dot{z}_1$, $z_4 = \dot{z}_2$, and $c = \tan(\psi)/L$. Let the controls $u = [c, a]^\intercal$. Then $\dot{z}$ fits the canonical form of (2.1). To ascertain the importance of learning and adaptation, we add the following disturbance to $[\dot{z}_3, \dot{z}_4]^\intercal$ to use as a "true" model:

$$\delta(z) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} -\tanh(v^2) \\ -(0.1 + v) \end{bmatrix} \tag{2.19}$$

This constitutes a non-linearity in the forward velocity and a tendency to drift to the right.

We use the following barrier function for pointcloud-based obstacles. Similar to [43], we design this barrier function with an extra component to account for position-based constraints which have a relative degree greater than 1. This is done by including the time-derivative of the position-based constraint as an additional term in the barrier function, which penalizes velocities (or higher order derivatives) leading to a decrease of the level set function h. Let our safety set $\mathcal{C} = \{x \in \mathbb{R}^n | h(x, x') \geq 0\}$, where x' is the position of an obstacle. Let $h(x, x') = \|(x - x')\|_2 - r$ where $r > 0$ is the radius of a circle around the obstacle. Then construct a barrier function $B(x; x') = 1/(\gamma_p h(x, x') + \frac{d}{dt} h(x, x'))$. As shown by [50], $B(x)$ is a CBF, where γ_p helps to control the rate of convergence. We chose $\gamma_1(x), \gamma_2(x) = x$ and $\gamma_3(x) = \gamma/x$.

2.4.1 Validation of BALSA in Simulation

One iteration of the algorithm for this problem takes less than $4ms$ on a 3.7GHz Intel Core i7-8700K CPU, in Python code which has not been optimized for speed. We make our code publicly available[1]. Because training the model occurs on a separate thread and can be performed anytime online, we do not include the model training time in this benchmark. We use OSQP [66] as our QP solver.

In Figure 2.2, we compare BALSA with several different baseline algorithms. We use a Neural Network trained with dropout and a negative-log-likelihood loss function for capturing the uncertainty [6]. We place several obstacles in the direct path of the reference trajectory. We also place velocity barriers for driving too fast or too slow. We observe that the behavior of the vehicle using our algorithm maintains good tracking errors while avoiding barriers and maintaining safety, while the other approaches suffer from various drawbacks. The adaptive controller (ad) and PD controller (pd) violate safety constraints. The (qp) controller with an inaccurate model also

[1]https://github.com/ddfan/balsa.git

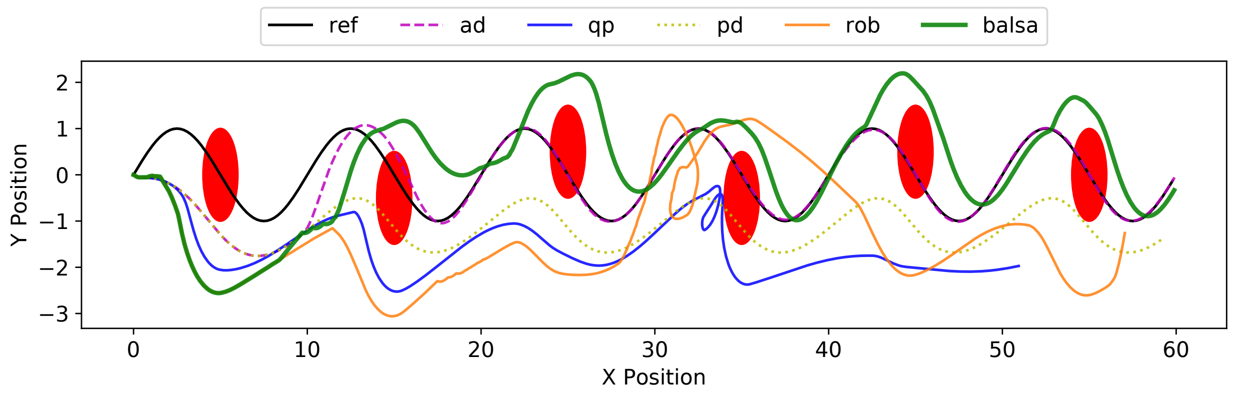


Figure 2.2: Comparison of the performance of four algorithms in tracking and avoiding barrier regions (red ovals). *ref* is the reference trajectory. *ad* is an adaptive controller $(\mu_{rm} + \mu_{pd} - \mu_{ad})$. *qp* is a non-adaptive safety controller $(\mu_{rm} + \mu_{pd} + \mu_{qp})$. *pd* is a proportional derivative controller $(\mu_{rm} + \mu_{pd})$. *rob* is a robust controller which uses a fixed $\sigma_i(x)$ to compensate for modeling errors. *balsa* is the full adaptive CLF-CBF-QP approach outlined in this paper and in Algorithm 1, i.e. $(\mu_{rm} + \mu_{pd} - \mu_{ad} + \mu_{qp})$.

violates constraints and exhibits highly suboptimal behavior (Figure 2.3). A robust (rob) formulation which uses a fixed robust bound which is meant to bound any model uncertainty [43], while not violating safety constraints, is too conservative and non-adaptive, has trouble tracking the reference trajectory. In contrast, BALSA adapts to model error with guaranteed safety. We also plot the model uncertainty and error in (Figure 2.3).

2.4.2 Comparing Different Modeling Methods in Simulation

Next we compared the performance of BALSA on three different Bayesian modeling algorithms: Gaussian Processes, a Neural Network with dropout, and ALPaCA [58], a meta-learning approach which uses a hybrid neural network with Bayesian regression on the last layer. For all methods we retrained the model intermittently, every 40 new datapoints. In addition to the current state, we also included as input to the model the previous control, angular velocity in yaw, and the current roll and pitch of the vehicle. For the GP we re-optimized hyperparameters with each training. For the dropout NN, we used 4 fully-connected layers with 256 hidden units each, and trained for 50 epochs with a batch size of 64. Lastly, for ALPaCA we used 2 hidden layers, each with 128 units, and 128 basis functions. We used a batch size of 150, 20 context data points, and 20 test data

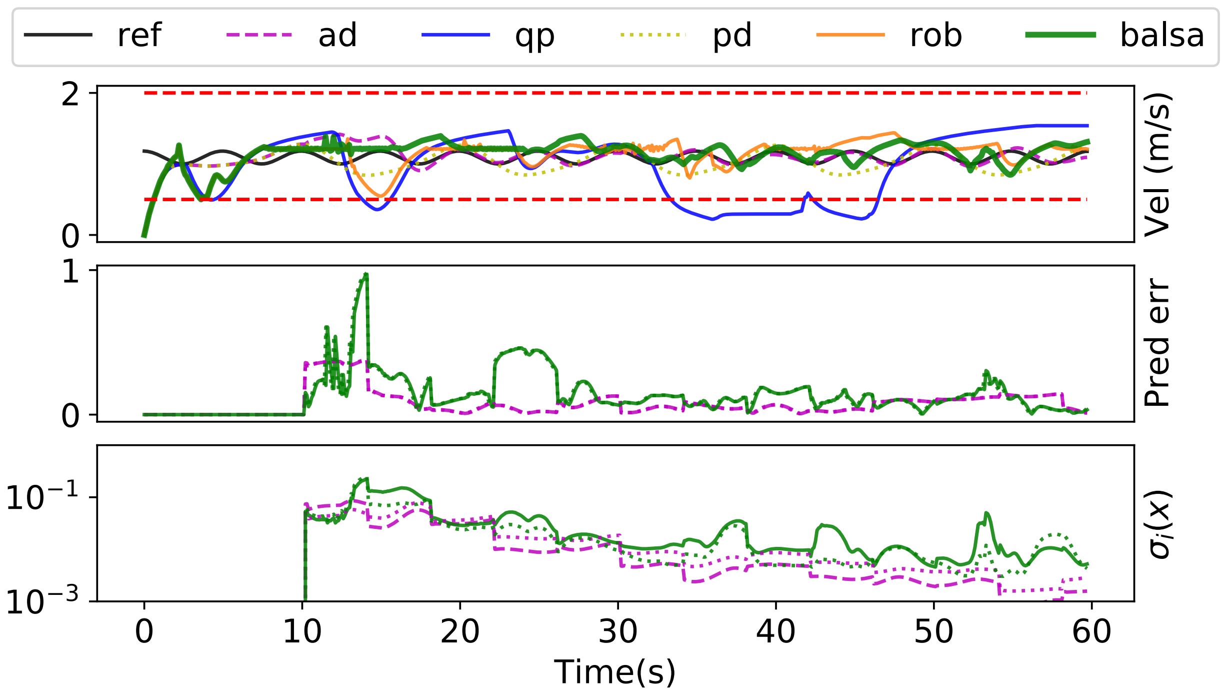
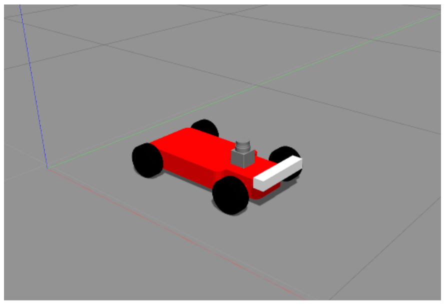


Figure 2.3: Top: Velocities of each algorithm. Red dotted line indicates safety barrier. Middle: Output prediction error of model, decreasing with time. Solid and dashed lines indicate both output dimensions. Bottom: Uncertainty $\sigma_i(x)$, also decreasing with time. Predictions are made after 10 seconds to accumulate enough data to train the network. During this time we choose an upper bound for $\sigma_0 = 1.0$.



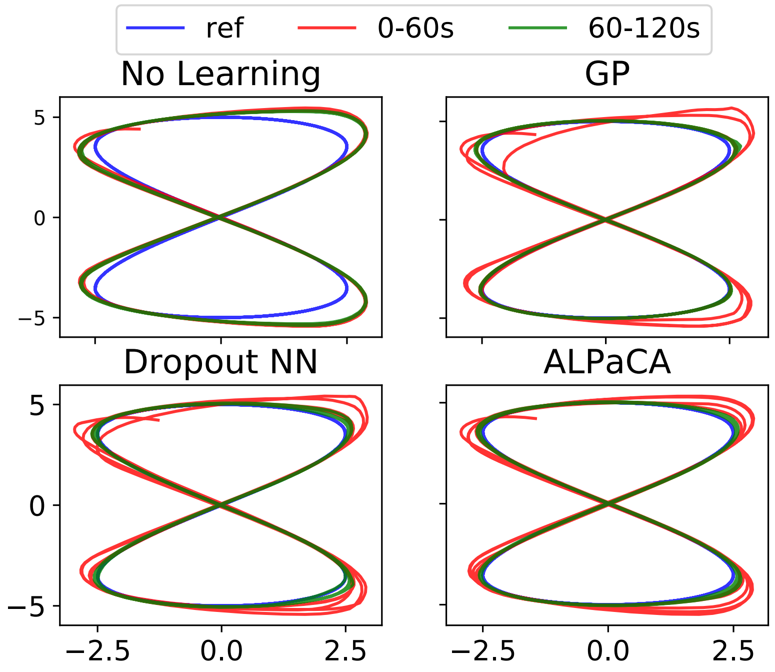
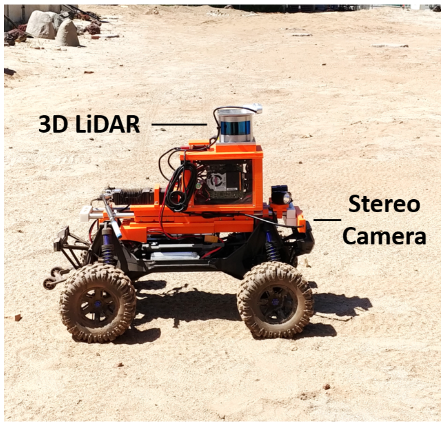


Figure 2.4: Comparison of adaptation performance in a Gazebo simulation using three different probabilistic model learning methods.

Table 2.1: Average tracking error in position for different modeling methods in sim, split into the first minute and second minute.

	No learn	GP	Dropout	ALPaCA
0-60s	0.580	0.3992	0.408	0.390
60-120s	0.522	0.097	0.105	0.110

points. The model was trained using 100 gradient steps and online adaption (during prediction) was performed using 20 of the most recent context data points with the current observation (see [58] for details of the meta-learning capabilities of ALPaCA). At each training iteration we retrain both the neural network and the last Bayesian linear regression layer. Figure (2.4) and Table (2.1) show a comparison of tracking error for these methods. We found GPs to be computationally intractable with more than 500 data points, although they exhibited good performance. Neural networks with dropout converged quickly and were efficient to train and run. ALPaCA exhibited slightly slower convergence but good tracking as well.



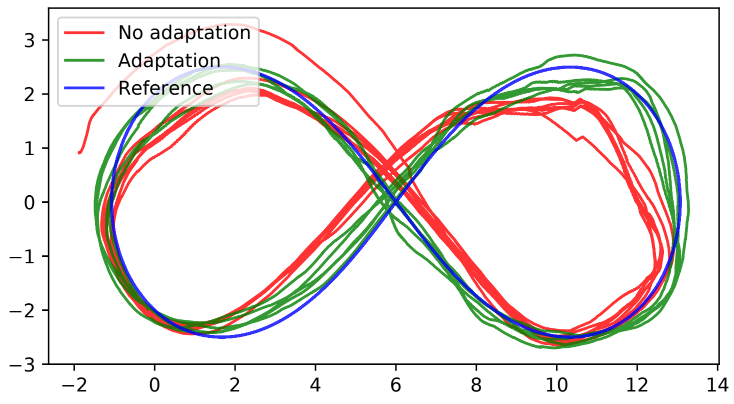


Figure 2.5: Left: A high-speed rover vehicle. Right: Figure-8 tracking on our rover platform on rough and sandy terrain, comparing adaptation vs. no adaptation.

2.4.3 Hardware Experiments on Martian Terrain

To validate that BALSA meets real-time computational requirements, we conducted hardware experiments on the platform depicted in Figure (2.5). We used an off-the shelf RC car (Traxxas Xmaxx) in 1/5-th scale (wheelbase 0.48 m), equipped with sensors such as a 3D LiDAR (Velodyne VLP-16) for obstacle avoidance and a stereo camera (RealSense T265) for on-board for state estimation. The power train consists of a single brushless DC motor, which drives the front and rear differential, operating in current control mode for controlling acceleration. Steering commands were fed to a servo position controller. The on-board computer (Intel NUC i7) ran Ubuntu 18.04 and ROS [67].

Experiments were conducted in a Martian simulation environment, which contains sandy soil, gravel, rocks, and rough terrain. We gave figure-eight reference trajectories at 2m/s and evaluated the vehicle's tracking performance (Figure 2.5). Due to large achieving good tracking performance at higher speeds is difficult. We observed that BALSA is able to adapt to bumps and changes in friction, wheel slip, etc., exhibiting improved tracking performance over a non-adaptive baseline (Table 2.2).

We also evaluated the safety of BALSA under adaptation. We used LiDAR pointclouds to create barriers at each LiDAR return location. Although this creates a large number of constraints, the QP solver is able to handle these in real-time. Figure 2.6 shows what happens when an obstacle is

Table 2.2: Mean, standard deviation, and max tracking error on our rover platform for a figure-8 task.

	Mean Err	Std Dev	Max
No Learn	1.417	0.568	6.003
Learning	0.799	0.387	2.310

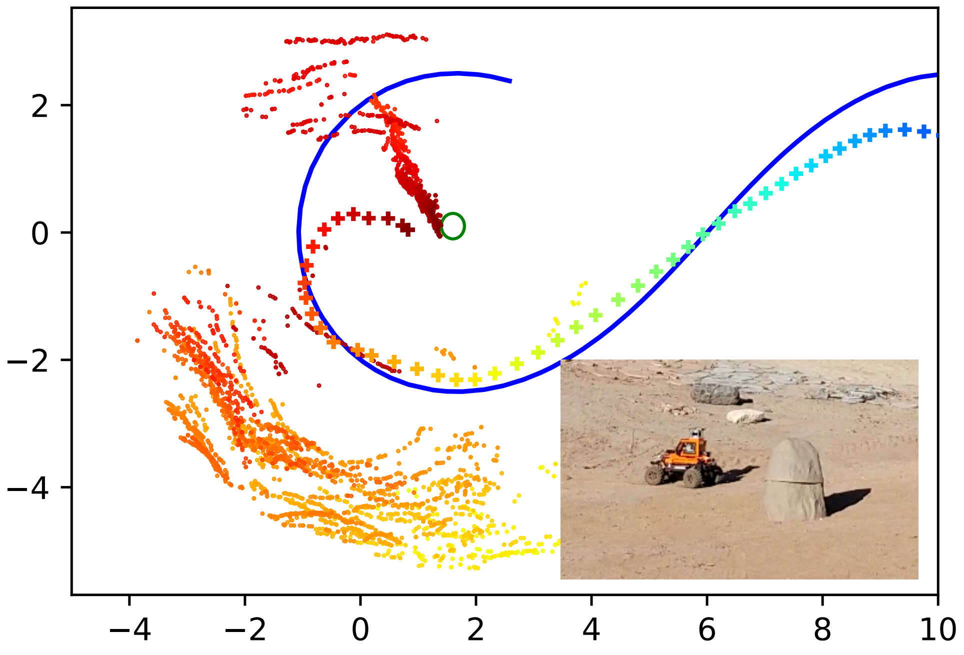


Figure 2.6: Vehicle avoids collision despite localization drift and unmodeled dynamics. Blue line is the reference trajectory, colored pluses are the vehicle pose, colored points are obstacles. Colors indicate time, from blue (earlier) to red (later). Note that localization drift results in the obstacles appearing to shift position. Green circle indicates location of the obstacle at the last timestep. Despite this drift the vehicle does not collide with the obstacle.

placed in the path of the reference trajectory. The vehicle successfully slows down and comes to a stop if needed, avoiding the obstacle altogether.

2.5 Conclusion

In this work, we have described a framework for safe, fast, and computationally efficient probabilistic learning-based control. The proposed approach satisfies several important real-world requirements and take steps towards enabling safe deployment of high-dimensional data-driven controls and planning algorithms.

CHAPTER 3
DEEP LEARNING TUBES FOR TUBE MPC

3.1 Summary

Learning-based control aims to construct models of a system to use for planning or trajectory optimization, e.g. in model-based reinforcement learning. In order to obtain guarantees of safety in this context, uncertainty must be accurately quantified. This uncertainty may come from errors in learning (due to a lack of data, for example), or may be inherent to the system. Propagating uncertainty forward in learned dynamics models is a difficult problem. In this work we use deep learning to obtain expressive and flexible models of how distributions of trajectories behave, which we then use for nonlinear Model Predictive Control (MPC). We introduce a deep quantile regression framework for control that enforces probabilistic quantile bounds and quantifies epistemic uncertainty. Using our method we explore three different approaches for learning tubes that contain the possible trajectories of the system, and demonstrate how to use each of them in a Tube MPC scheme. We prove these schemes are recursively feasible and satisfy constraints with a desired margin of probability. We present experiments in simulation on a nonlinear quadrotor system, demonstrating the practical efficacy of these ideas.

3.2 Introduction

In controls and planning, the idea of adapting to unknown systems and environments is appealing; however, guaranteeing safety and feasibility in the midst of this adaptation is of paramount concern. The goal of robust MPC is to take into account uncertainty while planning, whether it be from modeling errors, unmodeled disturbances, or randomness within the system itself [68]. In addition to safety, other considerations such as optimality, real-time tractability, scalability to high dimensional systems, and hard state and control constraints make the problem more difficult. In spite of these

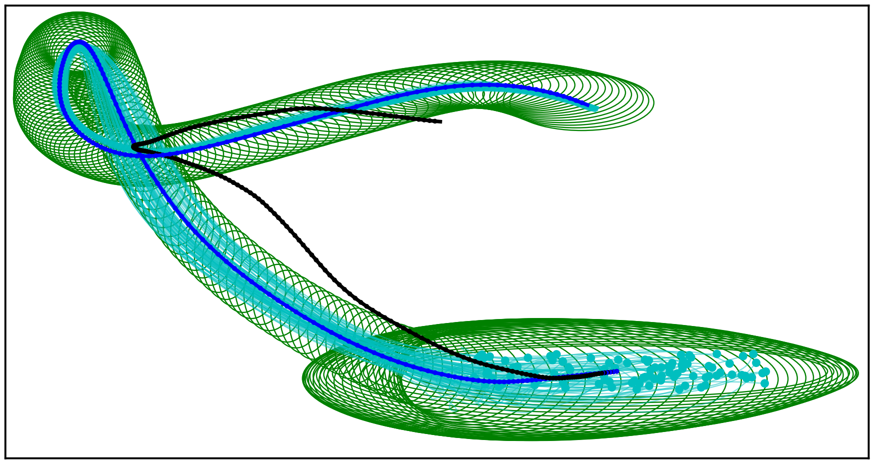


Figure 3.1: A learned tube (green) with learned mean (blue) that captures the distribution of trajectories (cyan) on a full quadrotor model tracking a target trajectory (black), propagated for 200 timesteps forward from the initial states (dots).

difficulties, learning-based robust MPC continues to receive much attention [69, 70, 71, 39, 72, 73, 37, 74, 75, 76]. However, in an effort to satisfy the many competing design requirements in this space, certain restrictive assumptions are often made, which include predetermined error bounds, restricted classes of dynamics models, or fixed parameterizations of the uncertainty.

Consider the following nonlinear dynamics equation that describes a real system:

$$x_{t+1} = f(x_t, u_t) + w_t \tag{3.1}$$

where $x \in \mathbb{X} \subseteq \mathbb{R}^n$ is the state, $u \in \mathbb{U} \subseteq \mathbb{R}^m$ are controls, and $w \in \mathbb{R}^n$ is noise or disturbance.

When attempting to find a model which captures the behavior of x_t, there will be error that results from insufficient data, lack of knowledge of w_t, or unknown or unobserved higher-dimensional dynamics not observed in x_t. One traditional approach has been to find robust bounds on the model error and plan using this robust model, i.e. $|w_t| \leq W$. However, this approach can be too conservative since it is not time or space varying and does not capture the distribution of the disturbance [75, 8]. To partially address this one could extend W to be time and state-varying, i.e. $W = W(x_t, u_t, t)$, as is commonly done in the robust MPC and control literature. For example, [77] takes this approach for feedback linearizable systems using boundary layer control, [16] leverages contraction theory and sum-of-squares optimization to find stabilizing controllers for nonlinear systems under uncertainty, and [78] solves for forward invariant tubes using min-max differential inequalities (See [79] for a recent overview of other related approaches). In this work we aim to learn this uncertainty directly from data, which allows us to avoid structural assumptions of the system of interest or restrictive parameterizations of uncertainty. We learn a *quantile* representation of the bounds of the distribution of possible trajectories, in the form of a tube around some nominal trajectory (Figure 1).

More closely related to our approach is the wide range of recent work in learning-based planning and control that seeks to handle model uncertainty probabilistically, where a model is constructed from one-step prediction measurements, and it is assumed that the true underlying distribution of the function is Gaussian [24, 80, 81, 12, 34]:

$$P(x_{t+1}|x_t, u_t) = \mathcal{N}(\mu(x_t, u_t), \sigma(x_t, u_t)). \tag{3.2}$$

where the mean function $\mu : \mathbb{X} \times \mathbb{U} \to \mathbb{X}$ and variance function $\sigma : \mathbb{X} \times \mathbb{U} \to \mathbb{X}^2$ capture the uncertainty of the dynamics for one time step. Various approaches for approximating this posterior distribution have been developed [82, 17]. For example, in PILCO and related work [80], moment matching of the posterior distribution is performed to find an analytic expression for the evolution of the mean and the covariance in time. The main problem with this approach is the one-step nature of this model, whereas for trajectory optimization we wish to propagate the distribution over multiple timesteps. Consider the two-timestep propagation:

$$\begin{aligned} P(x_{t+2}|x_t, u_t, u_{t+1}) &= \int P(x_{t+2}|x_{t+1}, u_{t+1}) P(x_{t+1}|x_t, u_t) dx_{t+1} \\ &= \int \big[\mathcal{N}(x_{t+2}|\mu(x_{t+1}, u_{t+1}), \sigma(x_{t+1}, u_{t+1}))\mathcal{N}(x_{t+1}|\mu(x_t, u_t), \sigma(x_t, u_t))\big] dx_{t+1} \end{aligned} \tag{3.3}$$

The two-timestep distribution will no longer be Gaussian, and marginalizing over the intermediate state x_{t+1} is intractable. As we do so, the true distribution may become multi-modal and highly non-Gaussian, and as the number of timesteps grows, the situation will grow worse. In order to arrive at these analytic expressions, assumptions must be made which lower the descriptive power for the model to capture the true underlying distribution, which may be multi-modal and highly

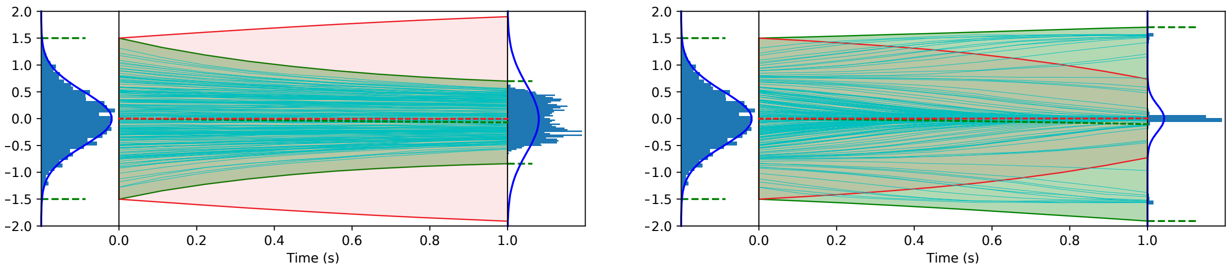


Figure 3.2: Comparison of 3-σ bounds on distributions of trajectories using GP moment matching (red) and the proposed quantile regression method (green). 100 sampled trajectories are shown (cyan) along with starting and ending distributions (blue, left and right histograms). Left: GP moment matching overestimates the distribution for the dynamics $\dot{x} = -x|x|$, while our method models it well. Right: GP moment matching underestimates the distribution for the dynamics $\dot{x} = -\sin(4x)$, while our method captures the tails of the distribution.

non-Gaussian. Furthermore, conservative estimates of the variance of the distribution will grow in an unbounded manner as the number of timesteps increases [83]. The result is that any chance constraints derived from these approximate models may be inaccurate. In Figure 3.2 we compare the classic GP-based moment matching approach for propagating uncertainty with our own deep quantile regression method on two different functions. While GP-moment matching can both underestimate and overestimate the true distribution of trajectories, our method is less prone to failures due to analytic simplifications or assumptions.

An alternative approach to Bayesian modeling for robust MPC has been to use quantile bounds to bound the tails of the distribution. This has the advantage that for planning in safety-critical contexts, we are generally not concerned with the full distribution of the trajectories, but the tails of these distributions only; specifically, we are interested in the probability of the tail of the distribution violating a safe set. A few recent works have taken this approach in the context of MPC; for example, [84] computes back-off sets with Gaussian Processes, and [76] uses an adaptive control approach to parameterize quantile bounds.

We are specifically interested in the idea of learning quantile bounds using the expressive power of deep neural networks. Quantile bounds give an explicit probability of violation at each timestep and allow for quantifying uncertainty which can be non-Gaussian, skewed, asymmetric, multimodal, and heteroskedastic [85]. Quantile regression itself is a well-studied field with the first results from

[86], see also [20, 87]. Quantile regression in deep learning has been also recently considered as a general statistical modeling tool [88, 89, 90, 91, 85]. Bayesian quantile regression has also been studied [92, 93]. Recently quantile regression has gained popularity as a modeling tool within the reinforcement learning community [94].

In addition to introducing a method for deep learning quantile bounds for distributions of trajectories, we also show how this method can be tailored to a tube MPC framework. Tube MPC [95, 96] was introduced as a way to address some of the shortcomings of classic robust MPC; specifically that robust MPC relied on optimizing over an open-loop control sequence, which does not predict the closed-loop behavior well. Instead, tube MPC seeks to optimize over a local policy that generates some closed-loop behavior, which has advantages of robust constraint satisfaction, computational efficiency, and better performance. The use of tube MPC allows us to handle high dimensional systems, as well as making the learning problem more efficient, tractable, and reliable. To the best of our knowledge, our work is the first to combine deep quantile regression with tube-based MPC, or indeed any learning-based robust MPC method.

The structure of the paper is as follows: In Section 3.3 we present our approach for learning tubes, which includes deep quantile regression, enforcing a monotonicity condition with a negative divergence loss function, and quantifying epistemic uncertainty. In Section 3.5 we present three different learning tube MPC schemes that take advantage of our method. In Section 3.6 we perform several experiments and studies to validate our method, and conclude in Section 3.7.

3.3 Deep Learning Tubes

3.3.1 Learning Tubes For Robust and Tube MPC

We propose learning time-varying invariant sets as a way to address the difficulties with propagating uncertainty for safety critical control, as well as to characterize the performance of a learned model

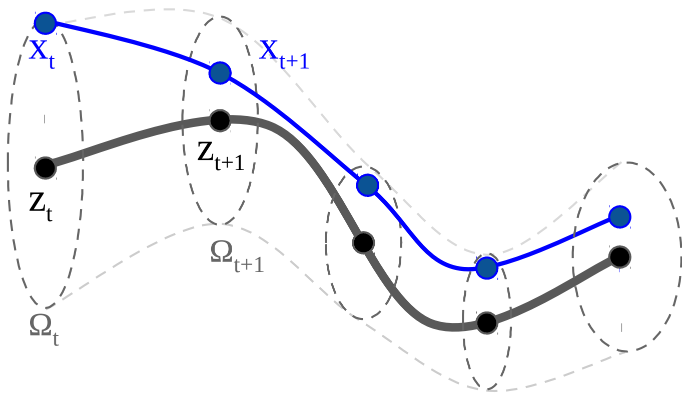


Figure 3.3: Diagram of a tube around the dynamics of z, within which x stays invariant. Note that the tube set Ω_t is time-varying.

or tracking controller. Consider the following *quantile* description of the dynamics:

$$
\begin{aligned}
x_{t+1} &= f(x_t, u_t) + w_t \\
z_{t+1} &= f_z(z_t, v_t) \\
\omega_{t+1} &= f_\omega(\omega_t, z_t, v_t, t) \\
P(d(x_t, z_t) &\leq \omega_t) \geq \alpha, \quad \forall t \in \mathbb{N}
\end{aligned}
\tag{3.4}
$$

where $z \in \mathbb{Z} \subseteq \mathbb{R}^{n_z}$ is a latent state of equal or lower dimension than x, i.e. $n_z \leq n$, and $v \in \mathbb{V} \subseteq \mathbb{R}^{m_z}$ is a pseudo-control input, also of equal or lower dimension than u, i.e. $m_z \leq m$. In the simplest case, we can fix $v_t = u_t$ and/or $z_t = x_t$. Also, $\omega \in \mathbb{R}^{n_z}$ is a vector that we call the *tube width*, with each element of $\omega > 0$.

This defines a "tube" around the trajectory of z within which x will stay close to z with probability greater than $\alpha \in [0, 1]$ (Figure 3.3). More formally, we can define the notion of closeness between some x and z by, for example, the distance between z and the projection of x onto $\mathbb{Z}$: $d(x, z) = |P_{\mathbb{Z}}(x) - z| \in \mathbb{R}^{n_z}$, where $P_{\mathbb{Z}}$ is a projection operator. Let $\Omega_\omega(z) \subset \mathbb{X}$ be a set in $\mathbb{X}$ associated with the tube width ω and z:

$$\Omega_\omega(z) := \{x \in \mathbb{X} : d(x, z) \leq \omega\}. \tag{3.5}$$

where the $\leq$ is element-wise. Other tube parameterizations are possible, for example $\Omega_\omega(z) := \{x \in \mathbb{X} : \|P_{\mathbb{Z}}(x) - z\|_\omega \leq 1\}$, where $\omega \in \mathbb{R}^{n_z \times n_z}$ instead. Further discussion of this case will be provided in Section (3.4).

The coupled system (3.4) induces a sequence of sets $\{\Omega_{\omega_t}(z_t)\}_{t=0}^{T}$ that form a tube around z_t. Our goal is to learn how this tube changes over time in order to use it for planning safe trajectories.

3.3.2 Quantile Regression

Our challenge is to learn the dynamics of the tube width, f_ω. Given data collected as trajectories $\mathcal{D} = \{x_t, u_t, x_{t+1}, z_t, v_t, z_{t+1}, t\}_{t=0}^{T}$, we can formulate the learning problem for f_ω as follows.

Let f_ω be parameterized with a neural network, f_ω^θ. For a given t and data point $\{x_t, u_t, x_{t+1}, z_t, v_t, z_{t+1}, t\}$, let $\omega_t = d(x_t, z_t)$ be the input tube width to f_ω, and $\omega_{t+1} = d(x_{t+1}, z_{t+1})$ the candidate output tube width. The candidate tube width at $t+1$ must be less than the estimate of the tube width at $t+1$, i.e: $\omega_{t+1} \leq f_\omega^\theta(\omega_t, z_t, v_t, t)$. To train the network f_ω^θ to respect these bounds we can use the following *check loss* function:

$$L_\omega^\alpha(\theta, \delta) = L^\alpha(\omega_{t+1}, f_\omega^\theta(\omega_t, z_t, v_t, t)) \tag{3.6}$$

$$L^\alpha(y, r) = \begin{cases} \alpha|y - r| & y > r \\ (1-\alpha)|y - r| & y \leq r \end{cases}$$

where the loss is a function of each data sample $\delta = \{\omega_{t+1}, \omega_t, z_t, v_t, t\}$. With the assumption of i.i.d. sampled data, when $L_\omega^\alpha(\theta, \delta)$ is minimized the quantile bound will be satisfied, (see Figure 3.4 and Theorem 3.3.1). In practice we can smooth this loss function near the inflection point $y = r$ with a slight modification, by multiplying L_ω^α with a Huber loss [97, 94].

Theorem 3.3.1. *Let θ^* minimize $\mathbb{E}_\delta[L_\omega^\alpha(\theta, \delta)]$. Then with probability α, $f_\omega^{\theta^*}(\omega, z, v, t)$ is an upper bound for $f_\omega(\omega, z, v, t)$.*

Proof. With a slight abuse of notation, let x denote the input variable to the loss function, and consider the expected loss $\mathbb{E}_x[L^\alpha(y(x), r(x))]$. We find the minimum of this loss w.r.t. r by setting

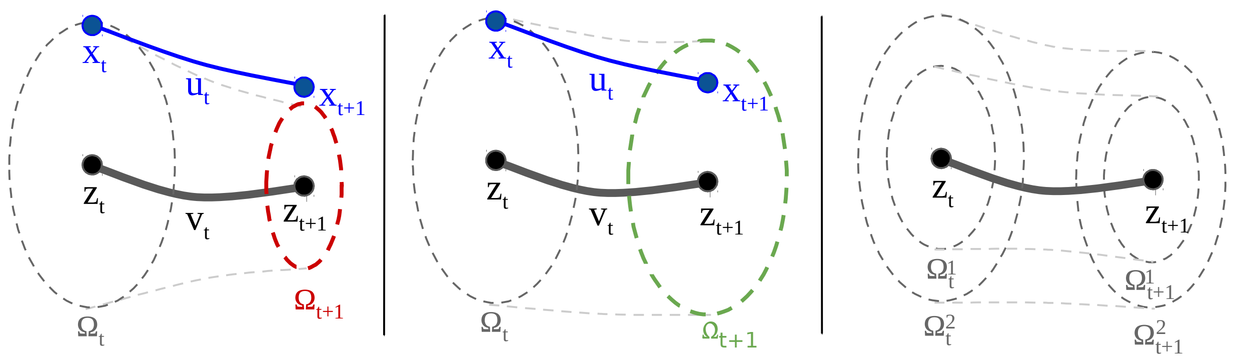


Figure 3.4: Learning tube dynamics from data. Left: The predicted tube at $t+1$ is too small. The gradient of the loss function will increase its size. Middle: The predicted tube at $t+1$ is larger than the actual trajectory in x taken, and will be shrunk. Right: The mapping $f_\omega(\omega, z_t, v_t, t)$ is monotonic with respect to ω, which results in $\Omega_t^1 \subseteq \Omega_t^2 \implies \Omega_{t+1}^1 \subseteq \Omega_{t+1}^2$.

the gradient to 0:

$$\begin{aligned}
&\frac{\partial}{\partial r^*}\mathbb{E}_x[L^\alpha(y(x), r^*(x))] \\
&= \int_{y(x)>r^*(x)} \alpha p(x)dx - \int_{y(x)\le r^*(x)} (1-\alpha)p(x)dx \\
&= \alpha p(y(x) > r^*(x)) - (1-\alpha)p(y(x) \le r^*(x)) = 0 \\
&\implies p(y(x) \le r^*(x)) = \alpha
\end{aligned} \tag{3.7}$$

Replacing $r^*(x)$ with $f_\omega^{\theta^*}(\omega, z, v, t)$ and $y(x)$ with $f_\omega(\omega, z, v, t)$ completes the proof. □

Note that quantile regression gives us tools for learning tube dynamics $f_\omega(\omega, z, v, t, \alpha)$ that are a function of the quantile probability α as well. This opens the possibility to dynamically varying the margin of safety while planning, taking into account acceptable risks or value at risk [98]. For example, in planning a trajectory, one could choose a higher α for the near-term and lower α in the later parts of the trajectory, reducing the conservativeness of the solution.

Additionally, we note that we can train the tube bounds dynamics in a recurrent fashion to improve long sequence prediction accuracy. While we present the above and following theorems in the context of one timestep, they are easily extensible to the recurrent case.

3.3.3 Enforcing Monotonicity

In addition to the quantile loss we also introduce an approach to enforce monotonicity of the tube with respect to the tube width (Figure 3.4, right). This is important for ensuring recursive feasibility of the MPC problem, as well as allowing us to shrink the tube width during MPC at each timestep if we obtain measurement updates of the current state, or, in the context of state estimation, an update to the covariance of the estimate of the current state. Enforcing monotonicity in neural networks has been studied with a variety of techniques [99, 100]. Here we adopt the approach of using a loss function that penalizes the network for having negative divergence, similar to [101]:

$$L_m(\theta, \delta) = -\min(0, \text{div}_\omega f_\omega(\omega, z, v, t)) \tag{3.8}$$

where div_ω is the divergence of f_ω with respect to ω. In practice we find that under gradient-based optimization, this loss decreases to 0 in the first epoch and does not noticeably affect the minimization of the quantile loss. Minimizing $L_m(\theta, \delta)$ allows us to make claims about the monotonicity of the learned tube:

Theorem 3.3.2. *Suppose θ^* minimizes $\mathbb{E}_\delta[L_m(\theta, \delta)]$ and $\mathbb{E}_\delta[L_m(\theta^*, \delta)] = 0$. Then for any $z_t \in \mathbb{Z}$, $v_t \in \mathbb{V}$, $t \in \mathbb{N}$ and $\omega_t^1, \omega_t^2 \in \mathbb{R}^{n_z}$, if $\Omega_{\omega_t^1}(z_t) \subseteq \Omega_{\omega_t^2}(z_t)$, then $\Omega_{\omega_{t+1}^1}(z_{t+1}) \subseteq \Omega_{\omega_{t+1}^2}(z_{t+1})$.*

Proof. Since $\forall \theta, \delta$, $L_m(\theta, \delta) > 0$ and $\mathbb{E}[L_m(\theta^*, \delta)] = 0$, then $L_m(\theta^*, \delta) = 0$. Then $\nabla_\omega f_\omega(\omega, z, v, t) > 0$ and f_ω is nondecreasing with respect to ω. Since $\Omega_{\omega_t^1} \subseteq \Omega_{\omega_t^2}$, then $\omega_t^1 \leq \omega_t^2$, so $f_\omega(\omega_t^1, z_t, v_t, t) \leq f_\omega(\omega_t^2, z_t, v_t, t)$, which implies that $\Omega_{\omega_{t+1}^1}(z_{t+1}) \subseteq \Omega_{\omega_{t+1}^2}(z_{t+1})$. □

3.3.4 Epistemic Uncertainty

Finally, in order to account for uncertainty in regions where no data is available for estimating quantile bounds, we incorporate methods for estimating epistemic uncertainty. Such methods can include Bayesian neural networks, Gaussian Processes, or other heuristic methods in deep learning [6, 94, 102]. For the experiments in this work we adopt an approach that adds an additional output layer to our quantile regression network that is linear with respect to orthonormal weights [85]. We emphasize that a wide range of methods for quantifying epistemic uncertainty are available and we are not restricted to this one approach; however, for the sake of clarity, we present in detail our method of choice. Let $g(z, v, t)$ be a neural network with either fixed weights that are either randomly chosen or pre-trained, with l dimensional output. We branch off a second output with a linear layer: $C^\intercal g(z, v, t)$, where $C \in \mathbb{R}^{l\times k}$. The estimate of epistemic uncertainty is chosen as $u_e(z, v, t) = \|C^\intercal g(z, v, t)\|^2$. Then, the parameters C are trained by minimizing the following loss:

$$L_u(C, \delta) = \|C^\intercal g(z, v, t)\|^2 + \lambda\|C^\intercal C - I_k\|. \tag{3.9}$$

where $\lambda > 0$ weights the orthonormal regularization. Minimizing this loss produces a network that has a value close to 0 when the input data is in-distribution, and increases with known rate as the input data moves farther from the training distribution (Figure 3.5, and see [85] for detailed analysis). We scale the predicted quantile bound by the epistemic uncertainty, then add a maximum bound to prevent unbounded growth as ω grows:

$$f_\omega(\omega, z, v, t) \leftarrow \min\{(1 + \beta u_e(z, v, t))f_\omega(\omega, z, v, t), W\} \tag{3.10}$$

where $\beta > 0$ is a constant parameter that scales the effect of the epistemic uncertainty, and W is a vector that provides an upper bound on the total uncertainty. Finding an optimal β analytically may require some assumptions such as a known Lipschitz constant of the underlying function, non-heteroskedastic noise, etc., which we leave for future investigation. We set β and W by hand and find this approach to be effective in practice.

We expect that as the field matures, methods for providing guarantees on well-calibrated epistemic uncertainty in deep learning will continue to improve. In the meantime, we make the assumption that we have well-calibrated epistemic uncertainty, an assumption similar to those made with other learning-based controls methods, such as choosing noise covariances, disturbance magnitudes, or kernel types and widths. The main benefit of leveraging epistemic uncertainty modeling is that it allows us to maintain guarantees of safety and recursive feasibility when we have a limited amount of data to learn from. In the case when no reliable epistemic estimate is available, we can proceed if we simply assume there is sufficient data to learn a good model offline.

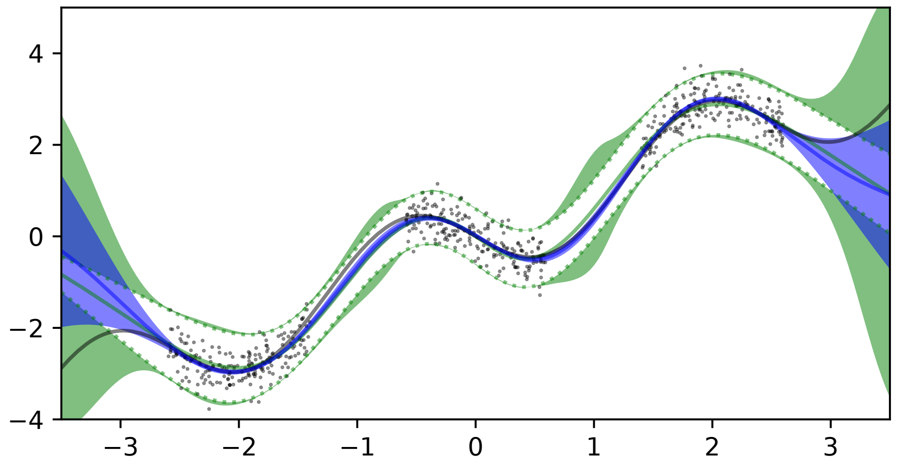


Figure 3.5: Estimating epistemic uncertainty for a 1-D function. Black dots indicate noisy data used to train the models, black line indicates the true function. Green colors indicate trained neural network models with green line indicating mean, and green dotted lines indicating learned 99% quantile bounds. Green shading indicates increased quantile bounds scaled by the learned epistemic uncertainty. Blue line and shading is GP regression with 99% bounds for comparison.

3.4 Extension to Tubes Defined by a Metric

In the previous section we considered tubes defined in the following manner (Equation 3.4):

$$\begin{aligned} x_{t+1} &= f(x_t, u_t) + w_t \\ z_{t+1} &= f_z(z_t, v_t) \\ \omega_{t+1} &= f_\omega(\omega_t, z_t, v_t, t) \\ P(d(x_t, z_t) &\leq \omega_t) \geq \alpha, \quad \forall t \in \mathbb{N} \end{aligned} \tag{3.11}$$

By this definition, the tube is defined element-wise, i.e. $d(x_t, z_t) \leq \omega_t$ is an element-wise inequality which defines a ”box”-shaped tube, where each dimension of the state must satisfy the inequality independently. As mentioned, other tube parameterizations are possible, which require slight modifications to the proofs presented in Section 3.3. Here, we consider the case where the tube is defined using a distance metric:

$$\begin{aligned} x_{t+1} &= f(x_t, u_t) + w_t \\ z_{t+1} &= f_z(z_t, v_t) \\ \omega_{t+1} &= f_\omega(\omega_t, z_t, v_t, t) \\ P(d(x_t, z_t; \omega_t) \leq 1) &\geq \alpha, \quad \forall t \in \mathbb{N} \end{aligned} \tag{3.12}$$

Here the inequality $d(x_t, z_t; \omega_t) \leq 1$ is no longer element-wise, and we define $d(\cdot, \cdot; \omega)$ as a distance metric which depends on a vector of parameters $\omega \in \mathbb{R}^{n_\omega}$. This parameter vector has its own tube dynamics f_ω. Different from the previous approach, we require d to be *non-increasing* with respect to ω, which we will define for multivariate functions. This will be useful for ensuring recursive feasibility, as we will show.

Definition 3.4.1. *Let $f(x) : \mathbb{R}^K \to \mathbb{R}$ be a multivariate function. Let $x, y \in \mathbb{R}^K$ be such that x is less than or equal to y element-wise, i.e. $x(k) \leq y(k), \forall k = 1, \cdots, K$. Then if $f(x) \leq f(y)$ we say the function f is non-decreasing, and if $f(x) \geq f(y)$, we say f is non-increasing.*

3.4.1 Quantile Metric Tube Loss

We can formulate the tube learning problem similarly as follows. We modify the check loss function (Equation 3.7) in terms of this metric tube:

$$\begin{aligned} L^\alpha_\omega(\theta, y) &= L^\alpha(d(x_{t+1}, z_{t+1}; f^\theta_\omega(\omega_t, z_t, v_t, t)), 1) \\ L^\alpha(a, b) &= \begin{cases} \alpha|a - b| & a > b \\ (1-\alpha)|a - b| & a \leq b \end{cases} \end{aligned} \tag{3.13}$$

With the assumption of i.i.d. sampled data, when $L^\alpha_\omega(\theta, y)$ is minimized the quantile bound will be satisfied (Theorem 3.4.1). In practice we can still smooth this loss function near the inflection point $a = b$ with a slight modification, by multiplying L^α_ω with a Huber loss [97, 94].

Theorem 3.4.1. *Let θ^* minimize $\mathbb{E}_y[L^\alpha_\omega(\theta, y)]$. Then with probability $\geq \alpha$, $d(x, z; \omega) \leq 1$.*

Proof. Consider the expected loss $\mathbb{E}_y[L^\alpha(a(y),1)]$. We find the minimum of this loss w.r.t. a by setting the gradient to 0 at the optimal a^*:

$$\begin{aligned}
&\frac{\partial}{\partial a^*}\mathbb{E}_x[L^\alpha(a^*(y),1)] \\
&= \int_{a^*(y)>1} \alpha p(y)dy - \int_{a^*(y)\le 1}(1-\alpha)p(y)dy \\
&= \alpha p(a^*(y)>1) - (1-\alpha)p(a^*(y)\le 1) = 0 \\
&\implies p(a^*(y)\le 1) = \alpha
\end{aligned} \tag{3.14}$$

Replacing $a^*(y)$ with $d(x,z;\omega)$ completes the proof. □

3.4.2 Enforcing Metric Monotonicity in ω

Next we must enforce monotonicity of the tube size with respect to the tube dynamics. This is important for ensuring recursive feasibility of the MPC problem. Again, we use a loss function that penalizes the network for having negative divergence, similar to [101]:

$$L_m(\theta, y) = -\min(0, \text{div}_\omega f_\omega(\omega, z, v, t)) \tag{3.15}$$

where div_ω is the divergence of f_ω with respect to ω. In practice we find that under gradient-based optimization, this loss decreases to 0 in the first epoch and does not noticeably affect the minimization of the quantile loss. Minimizing $L_m(\theta, y)$ allows us to make claims about the monotonicity of the learned tube (Theorem 3.4.3).

Definition 3.4.2. *Let $\Omega_\omega(z) \subset \mathbb{X}$ be called a tube set associated with the tube parameters ω and tube center z:*

$$\Omega_\omega(z) := \{x \in \mathbb{X} : d(x, z; \omega) \le 1\}. \tag{3.16}$$

Lemma 3.4.2. *If $\Omega_{\omega^1}(z) \subseteq \Omega_{\omega^2}(z)$, then $\omega^1 \le \omega^2$ element-wise, i.e. $\omega^1(k) \le \omega^2(k), \forall k = 1, \cdots, n_\omega$.*

Proof. (Contrapositive) Suppose $\omega^1 \not\le \omega^2$. We want to show that $\exists x \in \Omega_{\omega^1}(z)$ such that $x \notin \Omega_{\omega^2}(z)$. This can be done with a trivial example. Suppose $\omega^1 = [1,0]^\intercal$, $\omega^2 = [0,1]^\intercal$, and $d(x,z;\omega) =$

$(x-z)^\intercal \omega\omega^\intercal (x-z)$. Pick $z = [0,0]^\intercal$ and $x = [1,2]^\intercal$. Then $d(x,z,\omega^1) = 1$, so $x \in \Omega_{\omega^1}(z)$. But $d(x,z,\omega^2) = 4$, so $x \notin \Omega_{\omega^2}(z)$. □

Theorem 3.4.3. *Suppose θ^* minimizes $\mathbb{E}_y[L_m(\theta,y)]$ and $\mathbb{E}_y[L_m(\theta^*,y)] = 0$. Then for any $z_t \in \mathbb{X}$, $u_t \in \mathbb{U}$ and $\omega_t^1, \omega_t^2 \in \mathbb{R}^{n_z}$, if $\Omega_{\omega_t^1}(z_t) \subseteq \Omega_{\omega_t^2}(z_t)$, then $\Omega_{\omega_{t+1}^1}(z_{t+1}) \subseteq \Omega_{\omega_{t+1}^2}(z_{t+1})$.*

Proof. Since $\forall \theta, y$, $L_m(\theta,y) > 0$ and $\mathbb{E}[L_m(\theta^*,y)] = 0$, then $L_m(\theta^*,y) = 0$. Then $\nabla_\omega f_\omega(\omega,z,u) > 0$ and f_ω is nondecreasing with respect to ω. Since $\Omega_{\omega_t^1} \subseteq \Omega_{\omega_t^2}$, Lemma 3.4.2 implies that $\omega_t^1 \leq \omega_t^2$ element-wise. Then because f_ω is nondecreasing, $\omega_{t+1}^1 \leq \omega_{t+1}^2$. And because $d(\cdot,\cdot;\omega)$ is non-increasing with respect to ω, then $d(x,z_{t+1};\omega_{t+1}^1) \geq d(x,z_{t+1};\omega_{t+1}^2)$. So if $x \in \Omega_{\omega_{t+1}^1}(z_{t+1})$, then $d(x,z_{t+1};\omega_{t+1}^2) \leq 1$, which implies $x \in \Omega_{\omega_{t+1}^2}(z_{t+1})$. □

While we have presented the above theorems in the context of one timestep, by summing the accumulated losses over multiple timesteps, we can train both latent dynamics and tube dynamics functions in a recurrent fashion. This improves long sequence prediction accuracy and reduces compounding errors.

In the next sections we present various methods for applying this deep quantile regression approach to MPC problems. For simplicity, we assume that the tubes are defined in the manner of Equation 3.4.

3.5 Three Ways to Learn Tubes for Tube MPC

In this section we present three variations for applying our deep quantile regression approach to MPC problems, whose applicability may vary based on what components are available to the designer. By leveraging the previously described theorems for ensuring accurate quantiles, monotonicity, and uncertainty of the tube width dynamics, we can guarantee recursive feasibility of these MPC schemes, while ensuring that the trajectory of the system x_t remains within a safe set $x_t \in \mathcal{C} \subset \mathbb{R}^n$ with probability α at each timestep. The three different approaches require different elements of the system to be known or given, and are summarized as:

1. Given a tracking control law $u = \pi(x, z)$ and reference trajectory dynamics f_z, construct an invariant tube with the reference trajectory at its center (Figure 3.3).

2. Given a tracking control law π and reference trajectory dynamics f_z, construct a model of the dynamics of the error $e = x - z$, then learn an invariant tube with $z + e$ as its center (Figure 3.6a).

3. From data generated from any control law, random or otherwise, learn a reduced representation of the dynamics f_z (and optionally, a policy π to track it), along with tube bounds on the tracking error (Figure 3.6b).

3.5.1 Learning Tube Dynamics for a Given Controller

We first consider the case where we are given a fixed ancillary controller $\pi(x, z) : \mathbb{X} \times \mathbb{Z} \to \mathbb{U}$ (or potentially $\pi(x, z, v)$ with a feed-forward term v), along with nominal dynamics f_z that are used for planning and tracking in the classic tube MPC manner [103]. For now our goal is to learn f_ω alone.

We sum the three losses discussed in the previous section:

$$L(\theta, C, \delta) = L_\omega^\alpha(\theta, \delta) + L_m(\theta, \delta) + L_u(C, \delta) \tag{3.17}$$

to learn f_ω^θ, and find θ^* and C^* via stochastic gradient descent. Next, we perform planning on the coupled z and tube dynamics in the following nonlinear MPC problem. Let $T \in \mathbb{N}$ denote the planning horizon. We use the subscript notation $v_{k|t}$ to denote the variable v_k for $k = 0, \cdots, T$ within the MPC problem at time t. Let $v_{\cdot|t}$ denote the set of variables $\{v_{k|t}\}_{k=0}^T$. Then, at time t, the

MPC problem is:

$$\min_{v_{\cdot|t}\in\mathbb{V}} J_T(v_{\cdot|t}, z_{\cdot|t}, \omega_{\cdot|t}) \tag{3.18a}$$

$$s.t.\ \forall k = 0, \cdots, T:$$

$$z_{k+1|t} = f_z(z_{k|t}, v_{k|t}) \tag{3.18b}$$

$$\omega_{k+1|t} = f_\omega^\theta(\omega_t, z_t, v_t, t) \tag{3.18c}$$

$$\omega_{0|t} = d(x_t, z_{0|t}) \tag{3.18d}$$

$$z_{T|t} = f_z(z_{T|t}, v_{T|t}) \tag{3.18e}$$

$$\omega_{T|t} \geq f_\omega^\theta(\omega_{T|t}, z_{T|t}, v_{T|t}, T) \tag{3.18f}$$

$$\Omega_{\omega_{k|t}}(z_{k|t}) \subseteq \mathcal{C} \tag{3.18g}$$

Let $v^*_{\cdot|t}$, $z^*_{\cdot|t}$ denote the minimizer of the problem at time t. Note that we include $\omega_{\cdot|t}$ in the cost, which allows us to encourage larger or smaller tube widths. The tube width $\omega_{0|t}$ is updated based on a measurement x_t from the system, or can also be updated with information from a state estimator. In the absence of measurements we can also carry over the past optimized tube width, i.e. $\omega_{0|t} = \omega^*_{1|t-1}$, as long as $x_t \in \Omega_{\omega_{0|t}}(z_{0|t})$. The closed-loop control is set to $v_t = v^*_{0|t}$ and the tracking target for the underlying policy is $z_{t+1} = z^*_{1|t}$. Under these assumptions we have the following theorem establishing recursive feasibility and safety:

Theorem 3.5.1. *Suppose that the MPC problem (3.18) is feasible at $t = 0$. Then the problem is feasible for all $t > 0 \in \mathbb{N}$ and at each timestep the constraints are satisfied with probability α.*

Proof. The proof is similar to that in [79] for general set-based robust adaptive MPC. Let $z_{0|t+1} = z^*_{1|t}$ and choose any $\omega_{0|t+1}$ such that $x_{t+1} \in \Omega_{\omega_{0|t+1}}(z_{0|t+1})$ (if measurements x_{t+1} are unavailable, one can use $\omega_{0|t+1} = \omega^*_{1|t}$). With probability α, $\Omega_{\omega_{0|t+1}}(z_{0|t+1}) \subseteq \Omega_{\omega^*_{1|t}}(z^*_{1|t})$ due to Theorem 3.3.1. Let $v_{k|t+1} = v^*_{k+1|t}$ for $k = 0, \cdots, T-1$, and let $v_{T|t+1} = v^*_{T|t}$. Then $v_{\cdot|t+1}$ is a feasible solution for the MPC problem at $t = 1$, due to the terminal constraints (3.18e,3.18f) as well as the monotonicity of f_ω with respect to ω (Theorem 3.3.2). $\square$

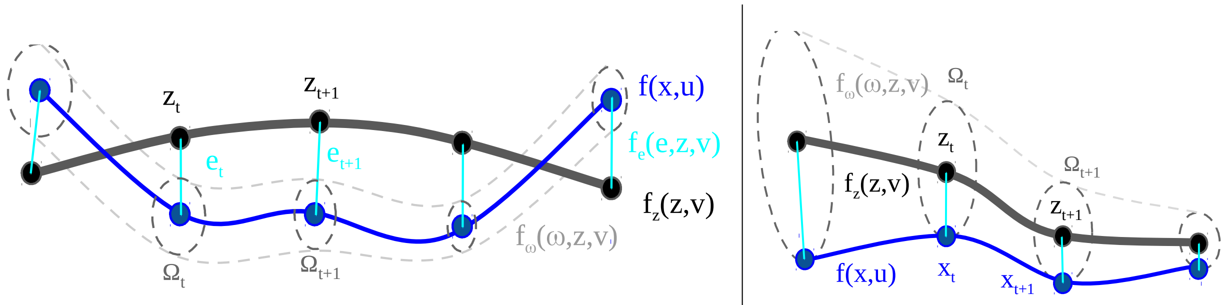
Algorithm 2: Tube Learning for Tube MPC

```
1  Require: Ancillary policy $\pi$, Latent dynamics $f_z$, Safe set $\mathcal{C}$, Quantile probability $\alpha$. MPC horizon $T$.
2  Initialize: Neural network for tube dynamics $f^\theta_\omega$. Dataset $\mathcal{D} = \{x_{t_i}, u_{t_i}, x_{t_i+1}, z_{t_i}, v_{t_i}, z_{t_i+1}, t_i\}_{i=1}^N$. Initial states $x_0$, $z_0$, Initial feasible controls $v_{\cdot|0}$.
3  for $t = 0, \cdots$ do
4      if updateModel then
5          Train $f^\theta_\omega$ on dataset $\mathcal{D}$ by minimizing tube dynamics loss (3.17).
6      if $x_t$ measured then
7          Initialize tube width $\omega_{0|t} = d(x_t, z_t)$
8      Solve MPC problem (3.18) with warm-start $v_{\cdot|t}$, obtain $v_t$, $z_{t+1}$
9      Apply control policy to system $u_t = \pi(x_t, z_{t+1})$
10     Step forward for next iteration:
       $v_{k|t+1} = v^*_{k+1|t},\ k = 0, \cdots, T-1,\ v_{T|t+1} = v^*_{T|t},\ z_{0|t+1} = z^*_{1|t},\ \omega_{0|t+1} = \omega^*_{1|t}$
11     Append data to dataset $\mathcal{D} \leftarrow \mathcal{D} \cup \{x_t, u_t, x_{t+1}, z_t, v_t, z_{t+1}, t\}$
```

Since $f^\theta_\omega(\omega_t, z_t, v_t)$ is nonlinear we find solutions to the MPC problem via iterative linear approximations, yielding an SQP MPC approach [104, 105]. Other optimization techniques are possible, including GPU-accelerated sampling-based ones [33]. We outline the entire procedure in Algorithm 2.

3.5.2 Learning Tracking Error Dynamics and Tube Dynamics

Next we show how to learn error dynamics $e_{t+1} = f_e(e_t, z_t, v_t)$ along with a tube centered along these dynamics, where $e_t = P_{\mathbb{Z}}(x) - z$ is the error between x and z, with x projected onto $\mathbb{Z}$. These error dynamics function as the mean of the distribution of dynamics $x_{t+1} = f(x_t, u_t)$ when the tracking policy is used $u_t = \pi(x_t, z_{t+1}, v_t)$. This allows the tube to take on a more accurately parameterized shape (Figure 3.6a). Setting up the learning problem in this way offers several distinct advantages. First, rather than relying on an accurate nominal model f_z and learning the bounds between this model and the true dynamics, we directly characterize the difference between the two models with f_e. This means that f_z can be chosen more arbitrarily and does not need to be a high-fidelity dynamics model. Second, using the nominal dynamics z_t as an input to f_e and learning the error "anchors" our prediction of the behavior of x_t to z_t. This allows us to predict the expected



(a) Learning Tracking Error

(b) Learning Model Error

Figure 3.6: (a) Learning error dynamics f_e along with tube dynamics f_ω. Black line is the nominal trajectory f_z, blue line is data collected from the system. Cyan indicates tracking errors, whose dynamics are learned. Grey tube denotes f_ω, which captures the error between the true dynamics and $z_t + e_t$. (b) Fitting learned dynamics to actual data. Blue inline indicates data collected from the system, black line is a learned dynamics trajectory fitted to the data.

distribution of x_t with much higher accuracy for long time horizons, in contrast to the approach of learning a model f directly and propagating it forward in time, where the error between the learned model and the true dynamics tends to increase with time.

As before, we assume we have a known π and nominal dynamics f_z. Let $\Omega^e_\omega(z, e) \subset \mathbb{X}$ be a set in $\mathbb{X}$ associated with the tube width ω,z, and e:

$$\Omega^e_\omega(z, e) := \{x \in \mathbb{X} : d(x, z + e) \leq \omega\}. \tag{3.19}$$

where the $\leq$ is element-wise. We have the following description of the error dynamics:

$$\begin{aligned} e_{t+1} &= f_e(e_t, z_t, v_t) \\ \omega_{t+1} &= f_\omega(\omega_t, z_t, v_t) \\ P(|(z_t + e_t) + x_t| \leq \omega_t) &\geq \alpha, \qquad \forall t \in \mathbb{N} \end{aligned} \tag{3.20}$$

Given a dataset $\mathcal{D} = \{x_t, u_t, x_{t+1}, z_t, v_t, z_{t+1}, t\}_{t=0}^N$, we minimize the following loss over data samples $\delta = \{x_t, x_{t+1}, z_t, z_{t+1}, v_t\}$ in order to learn $f_e(e_t, z_t, v_t)$, which we parameterize with ξ:

$$L_e(\xi, \delta) = \|f^\xi_e(P_\mathbb{Z}(x_t) - z_t, v_t) - P_\mathbb{Z}(x_{t+1}) - z_{t+1}\|_2 \tag{3.21}$$

Next, we learn f_ω by minimizing the quantile loss (3.17). However, while in the previous section $\omega_t = d(x_t, z_t)$, here we approximate the tube width with $\omega_t = d(x_t, z_t + e_t)$. We obtain e_t by propagating the learned dynamics f_e^ξ forward in time, given z_t, v_t. Then we can solve a similar tube-based robust MPC problem (3.22):

$$\min_{v_{\cdot|t}\in\mathbb{V}} J_T(v_{\cdot|t}, z_{\cdot|t}+e_{\cdot|t}, \omega_{\cdot|t}) \tag{3.22a}$$

$$s.t.\ \forall k = 0, \cdots, T:$$

$$z_{k+1|t} = f_z(z_{k|t}, v_{k|t}) \tag{3.22b}$$

$$e_{k+1|t} = f_e^\xi(e_t, z_t, v_t) \tag{3.22c}$$

$$\omega_{k+1|t} = f_\omega^\theta(e_t, z_t, v_t, t) \tag{3.22d}$$

$$\omega_{0|t} = d(x_t, z_{0|t}+e_{0|t}) \tag{3.22e}$$

$$z_{T|t}+e_{T|t} = f_z(z_{T|t}, v_{T|t}) + f_e^\xi(e_{T|t}, z_{T|t}, v_{T|t}) \tag{3.22f}$$

$$\omega_{T|t} \geq f_\omega^\theta(\omega_{T|t}, z_{T|t}, v_{T|t}, T) \tag{3.22g}$$

$$\Omega^e_{\omega_{k|t}}(z_{k|t}) \subseteq \mathcal{C} \tag{3.22h}$$

Notice that the cost and constraints are now a function of $z_t + e_t$ and do not depend on z_t only. This means that we are free to find paths z_t for the tracking controller π to track, which may violate constraints. We maintain the same guarantees of feasibility and constraint satisfaction as in Theorem 3.5.1. Since the proof is similar we omit it for brevity. See Algorithm 3.

Theorem 3.5.2. *Suppose that the MPC problem (3.22) is feasible at $t = 0$. Then the problem is feasible for all $t > 0 \in \mathbb{N}$ and at each timestep the constraints are satisfied with probability α.*

3.5.3 Learning System Dynamics and Tube Dynamics

In our third approach to learning tubes, we wish to learn the dynamics directly without a prior nominal model f_z. We restrict $\mathbb{Z} = \mathbb{X}$ and $\mathbb{V} = \mathbb{U}$, and treat z as an approximation of x. Our goal is to learn f_z to approximate f, along with f_ω that will determine a time-varying upper bound on the model error. Typically the open-loop model error will increase in time in an unbounded manner, which may make it difficult to find a feasible solution to the MPC problem. One approach is to assume the existence of a stabilizing controller and terminal set, and use a terminal condition that

Algorithm 3: Learning Tracking Error Dynamics and Tube Dynamics for Tube MPC

```
1  Require: Ancillary policy $\pi$, Latent dynamics $f_z$, Safe set $\mathcal{C}$, Quantile probability $\alpha$. MPC horizon $T$.
2  Initialize: Neural network for error dynamics $f_e^\xi$. Neural network for tube dynamics $f_\omega^\theta$. Dataset
     $\mathcal{D} = \{x_{t_i}, u_{t_i}, x_{t_i+1}, z_{t_i}, v_{t_i}, z_{t_i+1}, t_i\}_{i=1}^N$. Initial states $x_0$, $z_0$, $e_0$, Initial feasible controls $v_{\cdot|0}$.
3  for $t = 0, \cdots$ do
4      if updateModels then
5          Train $f_e^\xi$ on dataset $\mathcal{D}$ by minimizing error dynamics loss (3.21).
6          Forward propagate learned model $f_x^\xi$ on dataset $\mathcal{D}$ to obtain $\{e_{t_i}\}_{t=1}^N$. Append to $\mathcal{D}$.
7          Train $f_\omega^\theta$ on dataset $\mathcal{D}$ by minimizing tube dynamics loss (3.17), but replace
             $\omega_{t_i} = d(x_{t_i}, x_{t_i} + e_{t_i})$.
8      if $x_t$ measured then
9          Initialize tube width $\omega_{0|t} = d(x_t, z_t + e_t)$
10     Solve MPC problem (3.22) with warm-start $v_{\cdot|t}$, obtain $v_t$, $z_{t+1}$
11     Apply control policy to system $u_t = \pi(x_t, z_{t+1}, v_t)$
12     Step forward for next iteration:
         $v_{k|t+1} = v^*_{k+1|t}$, $k = 0, \cdots, T-1$, $v_{T|t+1} = v^*_{T|t}$, $z_{0|t+1} = z^*_{1|t}$, $e_{0|t+1} = e^*_{1|t}$, $\omega_{0|t+1} = \omega^*_{1|t}$
13     Append data to dataset $\mathcal{D} \leftarrow \mathcal{D} \cup \{x_t, u_t, x_{t+1}, z_t, v_t, z_{t+1}, t\}$
```

ensures the trajectory ends in this set [106, 83]. A second approach is to find a feedback control law π to ensure bounded tube widths. We describe the latter approach in more detail, but do not restrict ourselves to it.

Using a standard L2 loss function, we first learn an approximation of f, call it f_z^ϕ with parameters ϕ:

$$L_f(\phi, \delta) = \|f_z^\phi(x_t, u_t) - x_{t+1}\|_2 \tag{3.23}$$

Next, we learn a policy π^ψ with parameters ψ by inverting the dynamics:

$$L_\pi(\psi, \delta) = \|\pi^\psi(x_t, x_{t+1}) - u_t\|_2 \tag{3.24}$$

By learning a policy in this manner we decouple the potentially inaccurate model $f_z^\phi(x_t, u_t)$ from the true dynamics, in a learning inverse dynamics fashion [107]. To see this, suppose we have some z_t and v_t, and $z_{t+1} = f_z^\phi(z_t, v_t)$. If $x_t \neq z_t$ and we apply v_t to the real system, $x_{t+1} = f(x_t, v_t)$, then the error $\|x_{t+1} - z_{t+1}\|$ will grow, i.e. $\|x_t - z_t\| \leq \|x_{t+1} - z_{t+1}\|$. However, if instead we use the policy π^ψ, then $f(x_t, \pi^\psi(x_t, z_{t+1}))$ should be closer to z_{t+1}, and the error is more likely to shrink. Other approaches are available for learning π, including reinforcement learning [108],

Algorithm 4: Learning Dynamics and Model Error Bounds for Tube MPC

```
1  Require: Safe set $\mathcal{C}$, Quantile probability $\alpha$. MPC horizon $T$.
2  Initialize: Neural network for policy $\pi^{\psi}$, dynamics $f_z^{\phi}$, and tube dynamics $f_\omega^{\theta}$. Dataset
      $\mathcal{D} = \{x_{t_i}, u_{t_i}, x_{t_i+1}\}_{i=1}^{N}$. Initial state $x_0$.
3  Solve MPC problem (3.18) for initial feasible control sequence $v_{\cdot|0}$.
4  for $t = 0, \cdots$ do
5      if updateModel then
6          Train $f_z^{\phi}$ on dataset $\mathcal{D}$ by minimizing dynamics loss (3.23).
7          Train $\pi^{\psi}$ on dataset $\mathcal{D}$ by minimizing policy loss (3.24).
8          Create $\mathcal{D}_z = \bigcup_t \left[\{x_{t+k}, x_{t+k+1}, z_{k|t}, v_{k|t}, z_{k+1|t}\}_{k=0}^{T}\right]$ by solving (3.25).
9          Train $f_\omega^{\theta}$ on dataset $\mathcal{D}_z$ by minimizing tube dynamics loss (3.17).
10     if $x_t$ measured then
11         Initialize tube width $\omega_{0|t} = d(x_t, z_t)$
12     Solve MPC problem (3.18) with warm-start $v_{\cdot|t}$, obtain $v_t$, $z_{t+1}$
13     Apply control policy to system $u_t = \pi^{\psi}(x_t, z_{t+1})$
14     Step forward for next iteration:
        $v_{k|t+1} = v^*_{k+1|t},\ k = 0, \cdots, T-1,\ v_{T|t+1} = v^*_{T|t},\ z_{0|t+1} = z^*_{1|t},\ \omega_{0|t+1} = \omega^*_{1|t}$
15     Append data to dataset $\mathcal{D} \leftarrow \mathcal{D} \cup \{x_t, u_t, x_{t+1}\}$
```

imitation learning [36], etc. Finally, we learn f_ω in the same manner as before by minimizing the quantile loss in (3.17). We generate data for learning the tube dynamics by fitting trajectories of the learned model f_z^{ϕ} to closely approximate the real data x_t (Figure 3.6b). We randomly initialize $z_{0|t}$ along the trajectory x_t by letting $z_{0|t} = \mathcal{N}(x_t, \sigma I)$. We solve the following problem for each t:

$$\min_{v_{\cdot|t} \in \mathbb{V}} \sum_{k=1}^{T} \|z_{k|t} - x_{t+k}\| \tag{3.25a}$$

$$s.t. \quad z_{k+1|t} = f_z^{\phi}(z_{k|t}, v_{k|t}), \quad \forall k = 0, \cdots, T-1 \tag{3.25b}$$

From the fitted dynamics model data, we collect tube training data $\mathcal{D}_z = \bigcup_t \left[\{x_{t+k}, x_{t+k+1}, z_{k|t}, v_{k|t}, z_{k+1|t}\}_{k=0}^{T}\right]$ and proceed to train the tube model. We can now solve the same tube-based robust MPC problem (3.18), with f_z replaced with f_z^{ϕ}. This allows us to maintain the same guarantees of feasibility and safety with probability α as before. See Algorithm 4.

3.6 Experimental Details

3.6.1 Evaluation on a 6-D problem

In this section we validate each of our three approaches to learned tubes for tube MPC on a 6-state simulated triple-integrator system. We introduce two sets of dynamics for f and f_z to demonstrate our method. Consider the following 2D triple-integrator system with 6 states, where $x = [p_x, p_y, v_x, v_y, a_x, a_y]^\intercal$, along with the 4 state 2D double-integrator dynamics for the reference system: $z = [p_x^z, p_y^z, v_x^z, v_y^z]$. Let these systems have the following dynamics (we show the x-axis only for brevity sake):

$$\frac{d}{dt}\begin{bmatrix} p_x \\ v_x \\ a_x \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -k_f \end{bmatrix}\begin{bmatrix} p_x \\ v_x \\ a_x \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_x + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} w \tag{3.26}$$

$$\frac{d}{dt}\begin{bmatrix} p_x^z \\ v_x^z \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -k_f^z \end{bmatrix}\begin{bmatrix} p_x^z \\ v_x^z \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} v_x \tag{3.27}$$

where $w \sim \mathcal{N}(0, \epsilon I_{2\times 2})$, and with similar dynamics for the y-axis. We construct the following cascaded PD control law:

$$\pi_x(p_x, p_x^z) = k_d(k_p(p_x^z - p_x) - v_x + v_x^z) + k_a(-a_x) \tag{3.28}$$

We choose $k_f = 0.1, k_f^z = 1.0, k_p = 1, k_d = 10, k_a = 5$, and $\epsilon = 0.05$. We also bound $\|v_x\|, \|v_y\| \leq 1$. We simulate in discrete time with $dt = 0.1$.

We collect ∼100 episodes with randomly generated controls, with episode lengths of ∼100 steps. Following each algorithm, we then set up an MPC task to navigate through a forest of obstacles (see Figure 3.7). We found an MPC planning horizon of 20-30 steps to be effective. We ran each MPC algorithm for 100 steps, or until the system reaches the goal. We also plot 100 rollouts of the ”true” system x_t to evaluate the learned bounds. For each learned network, we use 3 layers with 256 units each. When calculating constraints for the tube, we treat the tube width ω_t as axes for an ellipse rather than a box. This alleviates the need for solving a mixed integer quadratic program, at the cost of a slightly larger tube. We use a quadratic running cost that penalizes deviation from the goal and excessively large velocities.

With Algorithm 2, we note that the tube widths are quite large. This is because this algorithm uses the reference trajectory itself as the center of the tube. While the tube encloses the trajectories, it does not create a tight bound. In Algorithm 3, we address this issue directly. We learn dynamics of the mean tracking error and use this as our tube center. The resulting tube dynamics bound the state distribution more closely. Note that when solving the MPC problem, the optimized reference trajectory z_t is free to violate the constraints, as long as the system trajectories x_t do not. This approach allows for much more aggressive behaviors. For Algorithm 3, without a good tracking controller, the tube width increases over time. However, because we replan at each timestep with a finite horizon, the planner is still able to fit through narrow passages. In the example shown we replan from the current state x_t, with the assumption that it is measured. This allows us to create aggressive trajectories with narrow tube widths.

3.6.2 Comparison with analytic bounds

We compare our learned tubes with an analytic solution for robust bounds on the system (3.27). We derive these analytic bounds by assuming worst-case noise perturbations of the closed-loop system. We find the bound W such that $P(|w_t| \leq W) \geq \alpha$ (with $\alpha = 0.95$). The worst-case error at each timestep is $w_t = \pm W$. We compare these bounds with those learned with our quantile method (Figure 3.8). Our method tends to underestimate the true bounds slightly, which is due to the training data rarely containing worst-case adversarial noise sequences.

3.6.3 Ablative Study

We perform an ablative study of our tube learning method. Using Algorithm 2, we learn error dynamics and tube dynamics. We collect randomized data (400 episodes of 40 timesteps) and train f_ω under varying values of α. We then evaluate the accuracy of f_ω by sampling 100 new episodes of 10 timesteps, and plot the frequency that f_ω overestimates the true error, along with the magnitude

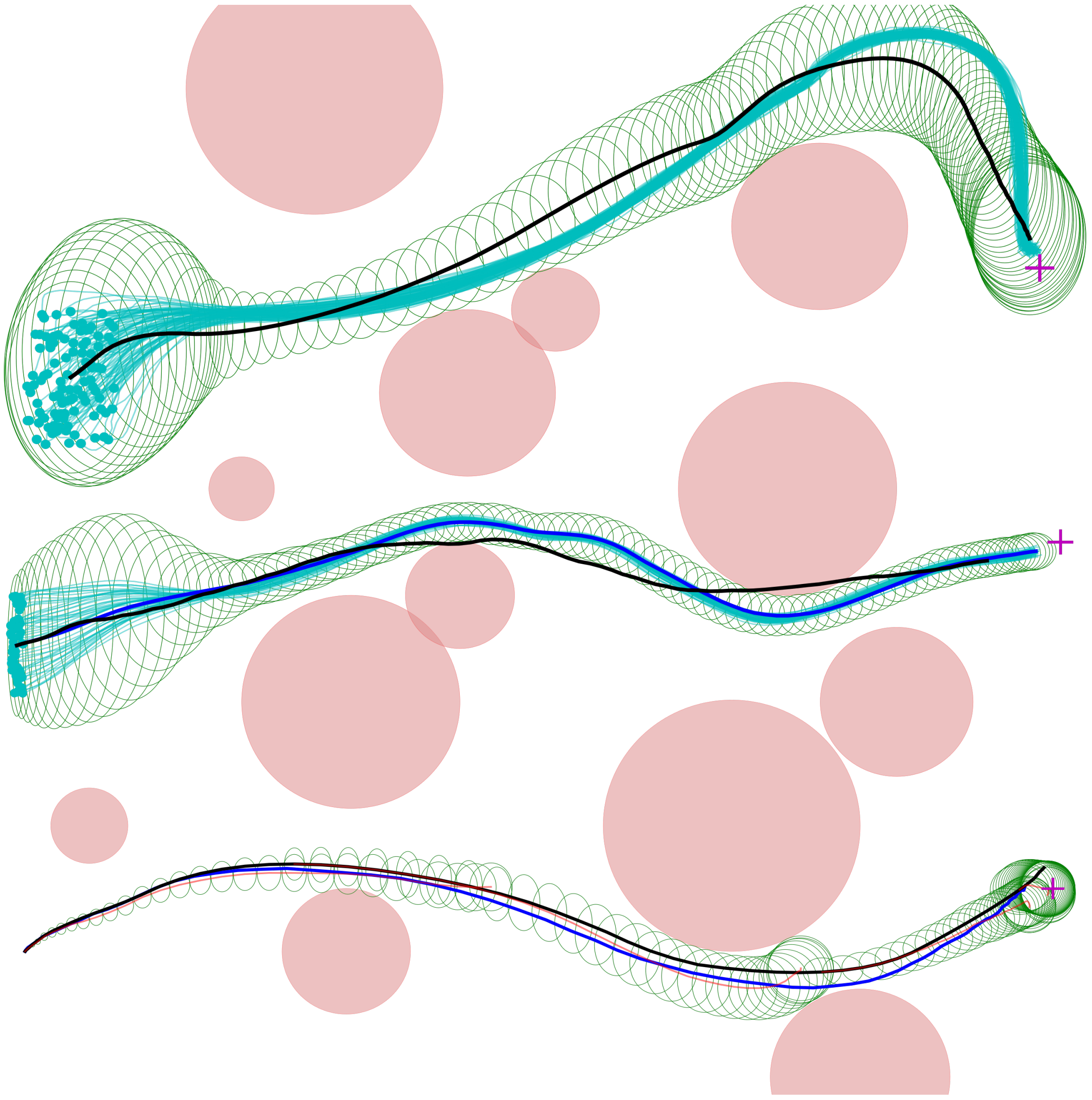


Figure 3.7: Comparison of 3 tube MPC approaches with learned tubes. Red circles denote obstacles, magenta cross denotes goal. Cyan lines indicate sampled trajectories from the system x_t with randomized initial conditions. Top: Algorithm 1, learning a tube around the reference z (black) used for tracking. Green circles indicate the tube width obtained at each timestep. Mid: Algorithm 2, learning tracking error dynamics (blue line) for the center of the tube. Bot: Algorithm 3, tube MPC problem using learned policy, dynamics, and tube dynamics. Red lines indicate planned NN dynamics trajectories at each MPC timestep, along with the forward propagated tube dynamics (green), shown every 20 timesteps. Blue line indicates actual path taken (x_t).

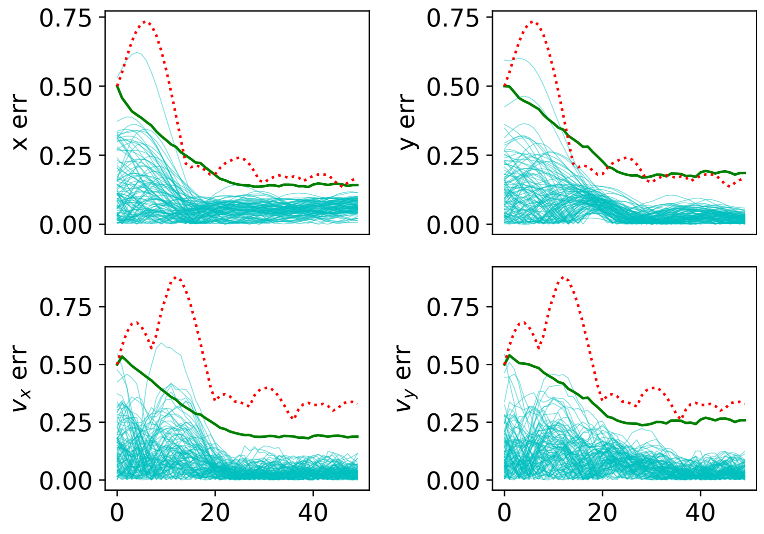


Figure 3.8: Learned 95% quantile error bounds (green) vs. 95% analytic bounds (dotted red) for the linear triple-integrator system, with 100 sampled trajectories, tracking a random reference trajectory.

of overestimation (Figure 3.9, left). We compare networks learned with the epistemic loss and without it, and find that our method produces well-calibrated uncertainties when using the epistemic loss, along with the quantile and monotonic losses (3.17). We evaluated ablation of the monotonic loss but found no noticeable differences.

We also evaluate estimation of epistemic uncertainty with varying amounts of data (from 10 to 400 episodes), with a fixed value of $\alpha = 0.95$. We find that estimating epistemic uncertainty is particularly helpful in the low-data regimes (Figure 3.9, right). As expected, the network maintains good quantile estimates by increasing the value of f_ω, which results in larger tubes. This creates more conservative behavior when the model encounters new situations.

3.6.4 Evaluation on Quadrotor Dynamics

To validate our approach scales well to high-dimensional non-linear systems, we apply Algorithm 3 to a 12 state, 4 input quadrotor model, with dynamics:

$$\dot{\mathbf{x}} = \mathbf{v} \qquad\qquad m\dot{\mathbf{v}} = mg\mathbf{e}_3 - TR\mathbf{e}_3$$

$$\dot{R} = R\hat{\Omega} \qquad\qquad J\dot{\Omega} = M + w - \Omega \times J\Omega$$

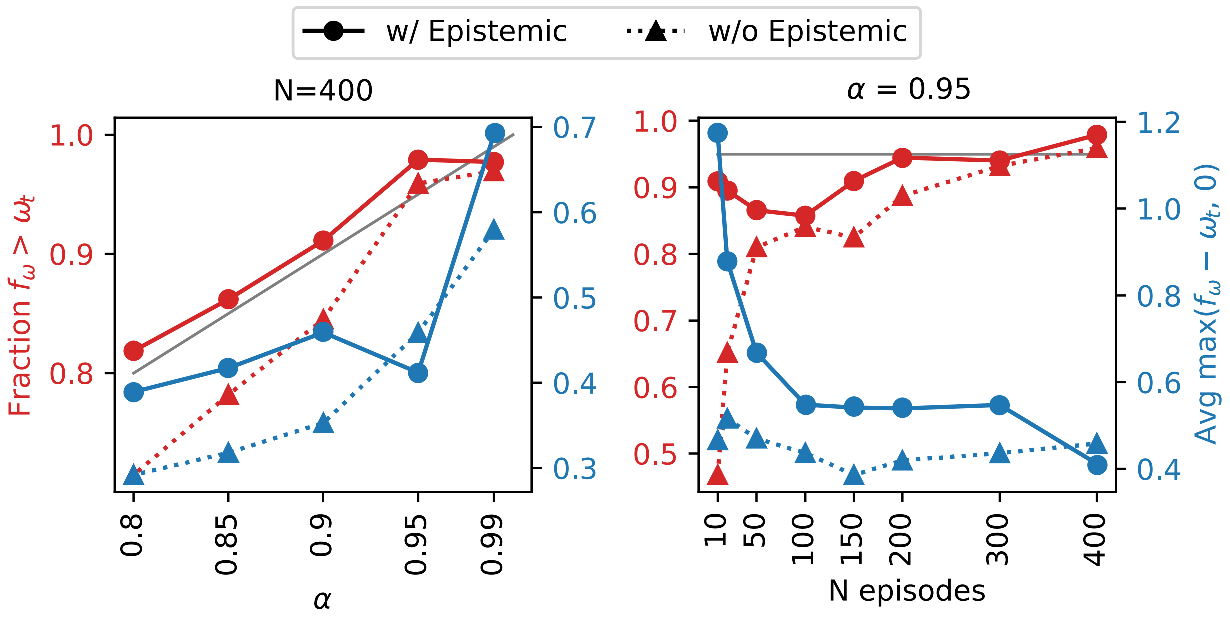


Figure 3.9: Evaluation of learned tube dynamics f_ω on triple integrator system with varying α (left) and varying number of datapoints (right). Red indicates fraction of validation samples that exceed the bound, while blue indicates average distance in excess of the bound. Models learned with the epistemic loss along with the quantile loss (circles, solid lines) perform better vs. models without epistemic uncertainty (triangles, dotted lines). Gray lines mark the best possible values.

where $\hat{\cdot} : \mathbb{R}^3 \to SO(3)$ is the hat operator. The states are the position $\mathbf{x} \in \mathbb{R}^3$, the translational velocity $\mathbf{v} \in \mathbb{R}^3$, the rotation matrix from body to inertial frame $R \in SO(3)$, and the angular velocity in the body frame $\Omega \in \mathbb{R}^3$. $m \in \mathbb{R}$ is the mass of the quadrotor, $g \in \mathbb{R}$ denotes gravitational force, and $J \in \mathbb{R}^{3\times3}$ is the inertia matrix in body frame. The inputs to the model are the total thrust $T \in \mathbb{R}$ and the total moment in the body frame $M \in \mathbb{R}^3$. Noise enters through the control channels, with $w \sim \mathcal{N}(0, \epsilon I_{3\times3})$. Our state is $x_t = \{\mathbf{x}, \mathbf{v}, R, \Omega\} \in \mathbb{R}^{18}$ and control input is $u_t = \{T, M\} \in \mathbb{R}^4$. We use a nonlinear geometric tracking controller that consists of a PD controller on position and velocity, which then cascades to an attitude controller [109]. For the nominal model f_z we use a double integrator system on each position axis. The nominal state is $z_t = \{\mathbf{x}, \mathbf{v}\} \in \mathbb{R}^6$ with acceleration control inputs $v_t = \{a_x, a_y, a_z\} \in \mathbb{R}^3$. See Figures 3.10 and 3.11.

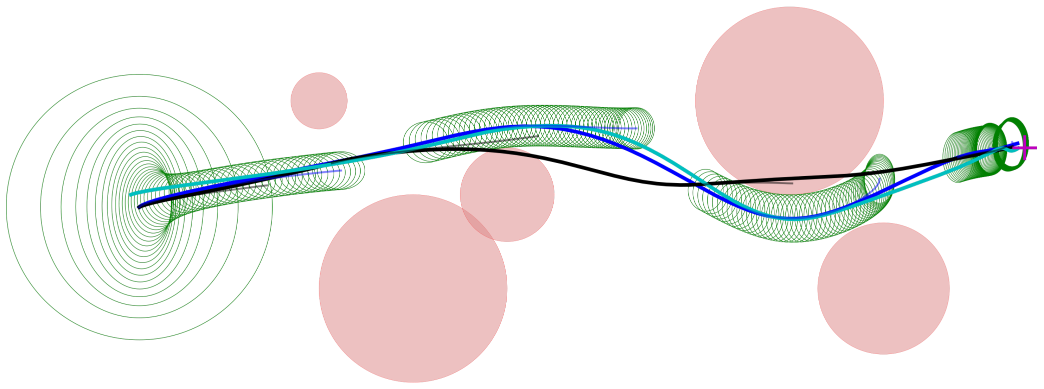


Figure 3.10: Algorithm 3 working on quadrotor dynamics, showing 5 individual MPC solutions at different times along the path taken. Thinner lines (black and blue) indicate planned future trajectories $z_{\cdot|t}$ and $e_{\cdot|t}$, respectively.

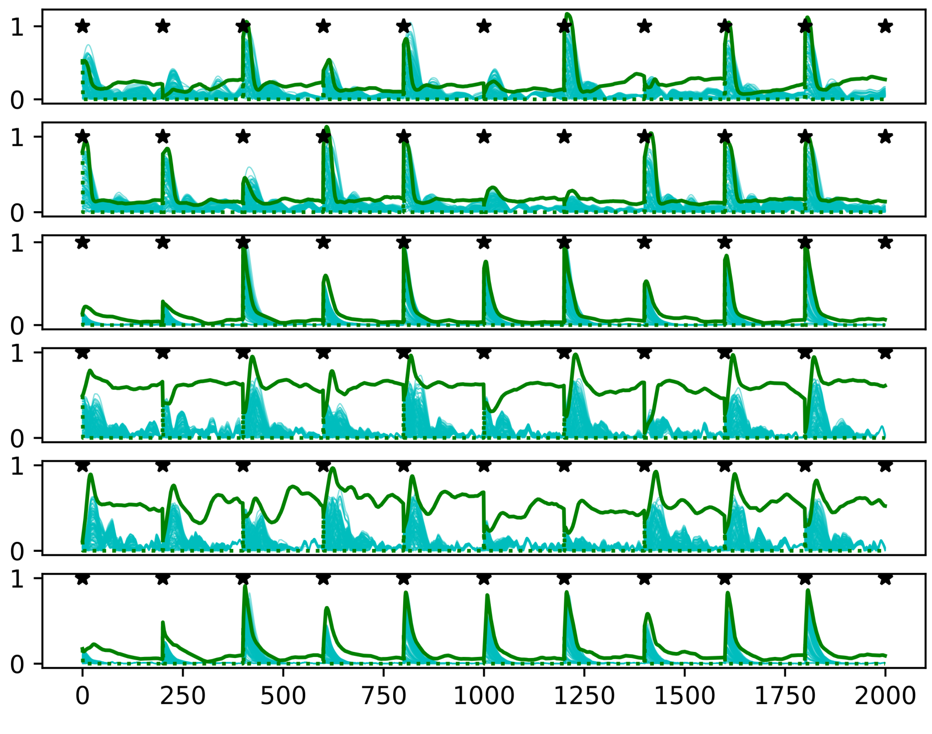


Figure 3.11: Tube widths f_ω for quadrotor dynamics, 10 episodes of 200 timesteps each, tracking random reference trajectories. From top to bottom, we plot $(p_x, p_y, p_z, v_x, v_y, v_z)$. Green lines indicate the quantile bound ω_t, with $\alpha = 0.9$, and cyan lines show 100 sampled error trajectories, $|x_t - e_t|$. Black stars indicate the start of a new episode.

3.7 Conclusion

We have introduced a deep quantile regression framework for learning bounds on controlled distributions of trajectories. For the first time we combine deep quantile regression in three robust MPC schemes with recursive feasibility and constraint satisfaction guarantees. We show that these schemes are useful for high dimensional learning-based control on quadrotor dynamics. We hope this work paves the way for more detailed investigation into a variety of topics, including deep quantile regression, learning invariant sets for control, handling epistemic uncertainty, and learning-based control for non-holonomic or non-feedback linearizable systems.

CHAPTER 4
STOCHASTIC TRAVERSABILITY EVALUATION AND PLANNING FOR RISK-AWARE OFF-ROAD NAVIGATION

4.1 Summary

Although ground robotic autonomy has gained widespread usage in structured and controlled environments, autonomy in unknown and off-road terrain remains a difficult problem. Extreme, off-road, and unstructured environments such as undeveloped wilderness, caves, and rubble pose unique and challenging problems for autonomous navigation. To tackle these problems we propose an approach for assessing traversability and planning a safe, feasible, and fast trajectory in real-time. Our approach, which we name STEP (Stochastic Traversability Evaluation and Planning), relies on: 1) rapid uncertainty-aware mapping and traversability evaluation, 2) tail risk assessment using the Conditional Value-at-Risk (CVaR), and 3) efficient risk and constraint-aware kinodynamic motion planning using sequential quadratic programming-based (SQP) model predictive control (MPC). We analyze our method in simulation and validate its efficacy on wheeled and legged robotic platforms exploring extreme terrains including an underground lava tube.

4.2 Introduction

Consider the problem of a ground robot tasked to autonomously traverse an unknown environment. In real-world scenarios, environments which are of interest to robotic operations are highly risky, containing difficult geometries (e.g. rubble, slopes) and non-forgiving hazards (e.g. large drops, sharp rocks) (See Figure 4.1) [110]. Determining where the robot may safely travel is a non-trivial problem, compounded by several issues: 1) Localization error affects how sensor measurements are accumulated to generate dense maps of the environment. 2) Sensor noise, sparsity, and occlusion induces biases and uncertainty in analysis of traversability. 3) Environments often pose a mix of

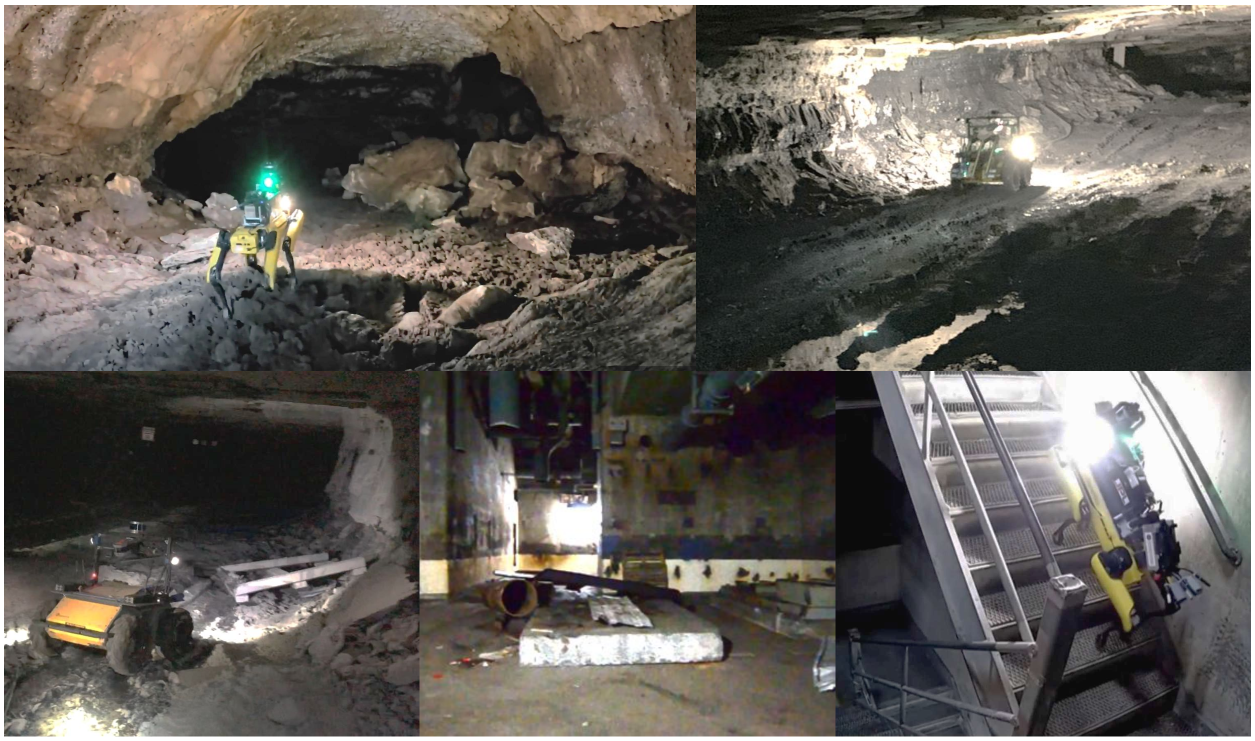


Figure 4.1: Top left: Boston Dynamics Spot quadruped robot exploring Valentine Cave at Lava Beds National Monument, CA. Top right, bottom left: Clearpath Husky robot exploring Arch Mine in Beckley, WV. Bottom middle, right: Spot exploring abandoned Satsop power plant in Elma, WA.

various sources of traversability risk, including slopes, rough terrain, low traction, narrow passages, etc. 4) These various risks create highly non-convex constraints on the motion of the robot, which are compounded by the kinodynamic constraints of the robot itself.

To address these issues we adopt an approach in which we directly quantify the traversal cost along with the uncertainties associated with that cost. We refer to this cost as *traversability*, e.g. a region of the environment in which the robot will suffer or become damaged has a high traversability cost. Building upon previous work on traversability in extreme terrains [111], we formulate the problem as a risk-aware, online nonlinear Model Predictive Control (MPC) problem, in which the uncertainty of traversability is taken into account when planning a trajectory. Our goal is to minimize the traversability cost, but directly minimizing the mean cost leads to an unfavorable result because tail events with low probability of occurrence (but high consequence) are ignored (Figure 4.2). Instead, in order to quantify the impact of uncertainty and risk on the motion of the robot, we employ a formulation in which we find a trajectory which minimizes the Conditional

Value-at-Risk (CVaR) [112]. Because CVaR captures the expected cost of the tail risk past a given probability threshold, we can dynamically adjust the level and severity of uncertainty and risk we are willing to accept (which depends on mission-level specifications, user preference, etc.). While online chance-constrained nonlinear MPC problems often suffer from a lack of feasibility, our approach allows us to relax the severity of CVaR constraints by adding a penalizing loss function.

We quantify risk via a traversability analysis pipeline (for system architecture, see Figure 4.3). At a high level, this pipeline creates an uncertainty-aware 2.5D traversability map of the environment by aggregating uncertain sensor measurements. Next, the map is used to generate both environment and robot induced costs and constraints. These constraints are convexified and used to build an online receding horizon MPC problem, which is solved in real-time. As we will demonstrate, we push the state-of-the-art in making this process highly efficient, allowing for re-planning at rates which allow for dynamic responses to changes and updates in the environment, as well as high travel speeds.

In this work, we propose STEP (Stochastic Traversability Evaluation and Planning), that pushes the boundaries of the state-of-the-art to enable safe, risk-aware, and high-speed ground traversal of unknown environments. Specifically, our contributions include:

1. Uncertainty-aware 2.5D traversability evaluation which accounts for localization error, sensor noise, and occlusion, and combines multiple sources of traversability risk.

2. An approach for combining these traversability risks into a unified risk-aware CVaR planning framework.

3. A highly efficient MPC architecture for robustly solving non-convex risk-constrained optimal control problems.

4. Real-world demonstration of real-time CVaR planning on wheeled and legged robotic platforms in unknown and risky environments.

4.3 Related Work

Our work is related to other classical approaches to traversability. Most traversability analyses are dependent on sensor type and measured through geometry-based, appearance-based, or proprioceptive methods [113]. Geometry-based methods often rely on building a 2.5D terrain map which is used to extract features such as maximum, minimum, and variance of the height and slope of the terrain [114]. Planning algorithms for such methods take into account the stability of the robot on the terrain [115]. In [116, 117], the authors estimate the probability distributions of states based on the kinematic model of the vehicle and the terrain height uncertainty. Furthermore, a method for incorporating sensor and state uncertainty to obtain a probabilistic terrain estimate in the form of a grid-based elevation map was considered in [118]. Our work builds upon these ideas by performing traversability analyses using classical geometric methods, while incorporating the uncertainty of these methods for risk-aware planning [119, 111].

Risk can be incorporated into motion planning using a variety of different methods, including chance constraints [120, 121], exponential utility functions [122], and distributional robustness [123]. Risk measures, often used in finance and operations research, provide a mapping from a random variable (usually the cost) to a real number. These risk metrics should satisfy certain axioms in order to be well-defined as well as to enable practical use in robotic applications [23]. Conditional value-at-risk (CVaR) is one such risk measure that has this desirable set of properties, and is a part of a class of risk metrics known as *coherent risk measures* [124] Coherent risk measures have been used in a variety of decision making problems, especially Markov decision processes (MDPs) [125]. In recent years, Ahmadi et al. synthesized risk averse optimal policies for partially observable MDPs and constrained MDPs [126, 127]. Coherent risk measures have been used in a MPC framework when the system model is uncertain [128] and when the uncertainty is a result of measurement noise or moving obstacles [129]. In [130, 129], the authors incorporated risk constraints in the form of distance to the randomly moving obstacles but did not include model uncertainty. Our work extends CVaR risk to a risk-based planning framework which utilizes different sources of traversability risk

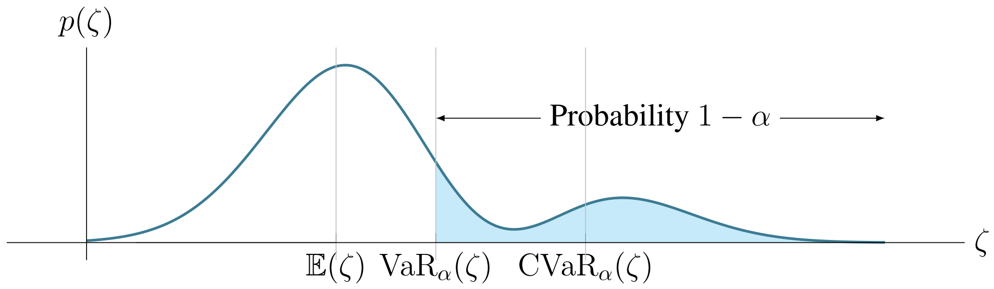


Figure 4.2: Comparison of the mean, VaR, and CVaR for a given risk level $\alpha \in (0, 1]$. The axes denote the values of the stochastic variable ζ, which in our work represents traversability cost. The shaded area denotes the $(1-\alpha)\%$ of the area under $p(\zeta)$. $\mathrm{CVaR}_\alpha(\zeta)$ is the expected value of ζ under the shaded area.

(such as collision risk, step risk, slippage risk, etc.)

Model Predictive Control has a long history in controls as a means to robustly control more complex systems, including time-varying, nonlinear, or MIMO systems [18]. While simple linear PID controllers are sufficient for simpler systems, MPC is well-suited to more complex tasks while being computationally feasible. In this work, MPC is needed to handle a) complex interactions (risk constraints) between the robot and the environment, including non-linear constraints on robot orientation and slope, b) non-linear dynamics which include non-holonomic constraints, and c) non-convex, time-varying CVaR-based constraints and cost functions. In particular, we take an MPC approach known as Sequential Quadratic Programming (SQP), which iteratively solves locally quadratic sub-problems to converge to a globally (more) optimal solution [131]. Particularly in the robotics domain, this approach is well-suited due to its reduced computational costs and flexibility for handling a wide variety of costs and constraints [132, 133]. A common criticism of SQP-based MPC (and nonlinear MPC methods in general) is that they can suffer from being susceptible to local minima. We address this problem by incorporating a trajectory library (which can be predefined and/or randomly generated, e.g. as in [134]) to use in a preliminary trajectory selection process. We use this as a means to find more globally optimal initial guesses for the SQP problem to refine locally. Another common difficulty with risk-constrained nonlinear MPC problems is ensuring recursive feasibility [19]. We bystep this problem by dynamically relaxing the severity of the risk constraints while penalizing CVaR in the cost function.

4.4 Risk-Aware Traversability and Planning

4.4.1 Problem Statement

We first give a formal definition of the problem of risk-aware traversability and motion planning. Let x_k, u_k, z_k denote the state, action, and observation at the k-th time step. A path $x_{0:N} = \{x_0, x_1, \cdots, x_N\}$ is composed of a sequence of poses. A policy is a mapping from state to control $u = \pi(x)$. A map is represented as $m = (m^{(1)}, m^{(2)}, \cdots)$ where m^i is the i-th element of the map (e.g., a cell in a grid map). The robot's dynamics model captures the physical properties of the vehicle's motion, such as inertia, mass, dimension, shape, and kinematic and control constraints:

$$x_{k+1} = f(x_k, u_k) \tag{4.1}$$

$$g(u_k) \succ 0 \tag{4.2}$$

where $g(u_k)$ is a vector-valued function which encodes control constraints/limits.

Following [113], we define *traversability* as the capability for a ground vehicle to reside over a terrain region under an admissible state. We represent traversability as a cost, i.e. a continuous value computed using a terrain model, the robotic vehicle model, and kinematic constraints, which represents the degree to which we wish the robot to avoid a given state:

$$r = \mathcal{R}(m, x, u) \tag{4.3}$$

where $r \in \mathbb{R}$, and $\mathcal{R}(\cdot)$ is a traversability assessment model. This model captures various unfavorable events such as collision, getting stuck, tipping over, high slippage, to name a few. Each mobility platform has its own assessment model to reflect its mobility capability.

Associated with the true traversability value is a distribution over possible values based on the current understanding about the environment and robot actions. In most real-world applications where perception capabilities are limited, the true value can be highly uncertain. To handle this uncertainty, consider a map belief, i.e., a probability distribution $p(m|x_{0:k}, z_{0:k})$, over a possible set $\mathcal{M}$. Then, the traversability estimate is also represented as a random variable $R : (\mathcal{M} \times \mathcal{X} \times \mathcal{U}) \longrightarrow$

$\mathbb{R}$. We call this probabilistic mapping from map belief, state, and controls to possible traversability cost values a *risk assessment model*.

A risk metric $\rho(R) : R \to \mathbb{R}$ is a mapping from a random variable to a real number which quantifies some notion of risk. In order to assess the risk of traversing along a path $x_{0:N}$ with a policy π, we wish to define the cumulative risk metric associated with the path, $J(x_0, \pi)$. To do this, we need to evaluate a sequence of random variables $R_{0:N}$. To quantify the stochastic outcome as a real number, we use the dynamic, time-consistent risk metric given by compounding the one-step risk metrics [135]:

$$J(x_0, \pi; m) = R_0 + \rho_0\big(R_1 + \rho_1\big(R_2 + \ldots + \rho_{N-1}\big(R_N\big)\big)\big) \tag{4.4}$$

where $\rho_k(\cdot)$ is a one-step coherent risk metric at time k. This one-step risk gives us the cost incurred at time-step $k+1$ from the perspective of time-step k. Any distortion risk metric compounded as given in (4.4) is time-consistent (see [23] for more information on distortion risk metrics and time-consistency). We use the Conditional Value-at-Risk (CVaR) as the one-step risk metric:

$$\rho(R) = \mathrm{CVaR}_\alpha(R) = \inf_{z \in \mathbb{R}} \mathbb{E}\left[z + \frac{(R-z)_+}{1-\alpha}\right] \tag{4.5}$$

where $(\cdot)_+ = \max(\cdot, 0)$, and $\alpha \in (0, 1]$ denotes the *risk probability level*.

We formulate the objective of the problem as follows: Given the initial robot configuration x_S and the goal configuration x_G, find an optimal control policy π^* that moves the robot from x_S to x_G while 1) minimizing time to traverse, 2) minimizing the cumulative risk metric along the path, and 3) satisfying all kinematic and dynamic constraints.

4.4.2 Hierarchical Risk-Aware Planning

We propose a hierarchical approach to address the aforementioned risk-aware motion planning problem by splitting the motion planning problem into geometric and kinodynamic domains. We consider the geometric domain over long horizons, while we solve the kinodynamic problem over a shorter horizon. This is convenient for several reasons: 1) Solving the full constrained CVaR minimization problem over long timescales/horizons becomes intractable in real-time. 2) Geometric constraints play a much larger role over long horizons, while kinodynamic constraints play a much larger role over short horizons (to ensure dynamic feasibility at each timestep). 3) A good estimate (upper bound) of risk can be obtained by considering position information only. This is done by constructing a position-based traversability model $\mathcal{R}_{\text{pos}}$ by marginalizing out non-position related variables from the risk assessment model, i.e. if the state $x = [p_x, p_y, x_{\text{other}}]^\intercal$ consists of position and non-position variables (e.g. orientation, velocity), then

$$\mathcal{R}_{\text{pos}}(m, p_x, p_y) \geq \mathcal{R}(m, x, u) \quad \forall x_{\text{other}}, u \tag{4.6}$$

Geometric Planning: The objective of geometric planning is to search for an *optimistic* risk-minimizing path, i.e. a path that minimizes an upper bound approximation of the true CVaR value. For efficiency, we limit the search space only to the geometric domain. We are searching for a sequence of poses $x_{0:N}$ which ends at x_G and minimizes the position-only risk metric in (4.4), which we define as $J_{\text{pos}}(x_{0:N})$. The optimization problem can be written as:

$$x^*_{0:N} = \underset{x_{0:N}}{\arg\min} \left[J_{\text{pos}}(x_{0:N}) + \lambda \sum_{k=0}^{N-1} \|x_k - x_{k+1}\|^2 \right] \tag{4.7}$$

$$s.t. \quad \phi(m, x_k) \succ 0 \tag{4.8}$$

where the constraints $\phi(\cdot)$ encode position-dependent traversability constraints (e.g. constraining the vehicle to prohibit lethal levels of risk) and $\lambda \in \mathbb{R}$ weighs the tradeoff between risk and path length.

Kinodynamic Planning: We then solve a kinodynamic planning problem to track the optimal geometric path, minimize the risk metric, and respect kinematic and dynamics constraints. The goal is to find a control policy π^* within a local planning horizon $T \leq N$ which tracks the path $X^*_{0:N}$. The optimal policy can be obtained by solving the following optimization problem:

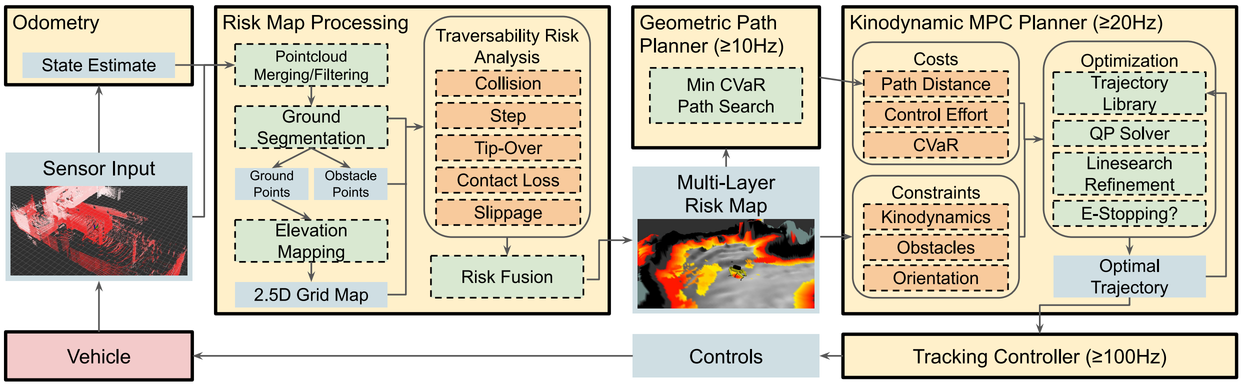


Figure 4.3: Overview of system architecture for STEP. From left to right: Odometry aggregates sensor inputs and relative poses. Next, Risk Map Processing merges these pointclouds and creates a multi-layer risk map. The map is used by the Geometric Path Planner and the Kinodynamic MPC Planner. An optimal trajectory is found and sent to the Tracking Controller, which produces control inputs to the robot.

$$\pi^* = \arg\min_{\pi \in \Pi} \left[J(x_0, \pi) + \lambda \sum_{k=0}^{T} \|x_k - x_k^*\|^2 \right] \tag{4.9}$$

$$s.t.\ \forall k \in [0, \cdots, T]: \qquad x_{k+1} = f(x_k, u_k) \tag{4.10}$$

$$g(u_k) \succ 0 \tag{4.11}$$

$$h(m, x_k) \succ 0 \tag{4.12}$$

where the constraints $g(u)$ and $h(m, x_k)$ are vector-valued functions which encode controller limits and state constraints, respectively.

4.5 STEP for Unstructured Terrain

Having outlined our approach for solving the constrained CVaR minimization problem, in this section we discuss how we compute traversability risk and efficiently solve the risk-aware trajectory optimization problem. At a high level, our approach takes the following steps (see Figure 4.3): 1) Assuming some source of localization with uncertainty, aggregate sensor measurements to create an uncertainty-aware map. 2) Perform ground segmentation to isolate the parts of the map the robot can potentially traverse. 3) Compute risk and risk uncertainty using geometric properties of the pointcloud (optionally, include other sources of risk, e.g. semantic or other sensors). 4) Aggregate these risks to compute a 2.5D CVaR risk map. 5) Solve for an optimistic CVaR minimizing path over long ranges with a geometric path planner. 7) Solve for a kinodynamically feasible trajectory which minimizes CVaR while staying close to the geometric path and satisfying all constraints.

4.5.1 Modeling Assumptions

Among many representation options for rough terrain, we use a 2.5D grid map in this paper for its efficiency in processing and data storage [136]. The map is represented as a collection of terrain properties (e.g., height, risk) over a uniform grid.

For different vehicles we use different robot dynamics models, and our general approach is applicable to any vehicle dynamics model. For example, a differential drive model, the state and controls are specified as:

$$x = [p_x, p_y, p_\theta, v_x]^\intercal \tag{4.13}$$

$$u = [a_x, v_\theta]^\intercal \tag{4.14}$$

The dynamics $x_{k+1} = f(x_k, u_k)$ for a simple differential-drive system can be written as:

$$x_{k+1} = x_k + \Delta t \begin{bmatrix} v_x \cos(p_\theta) \\ v_x \sin(p_\theta) \\ \gamma v_x + (1-\gamma) v_\theta \\ a_x \end{bmatrix} \tag{4.15}$$

where $\gamma \in [0, 1]$ is a constant which adjusts the amount of turning-in-place the vehicle is permitted.

for a system which produces longitudinal/lateral velocity and steering (e.g. legged platforms), the state and controls can be specified as:

$$x = [p_x, p_y, p_\theta, v_x, v_y, v_\theta]^\intercal \tag{4.16}$$

$$u = [a_x, a_y, a_\theta]^\intercal \tag{4.17}$$

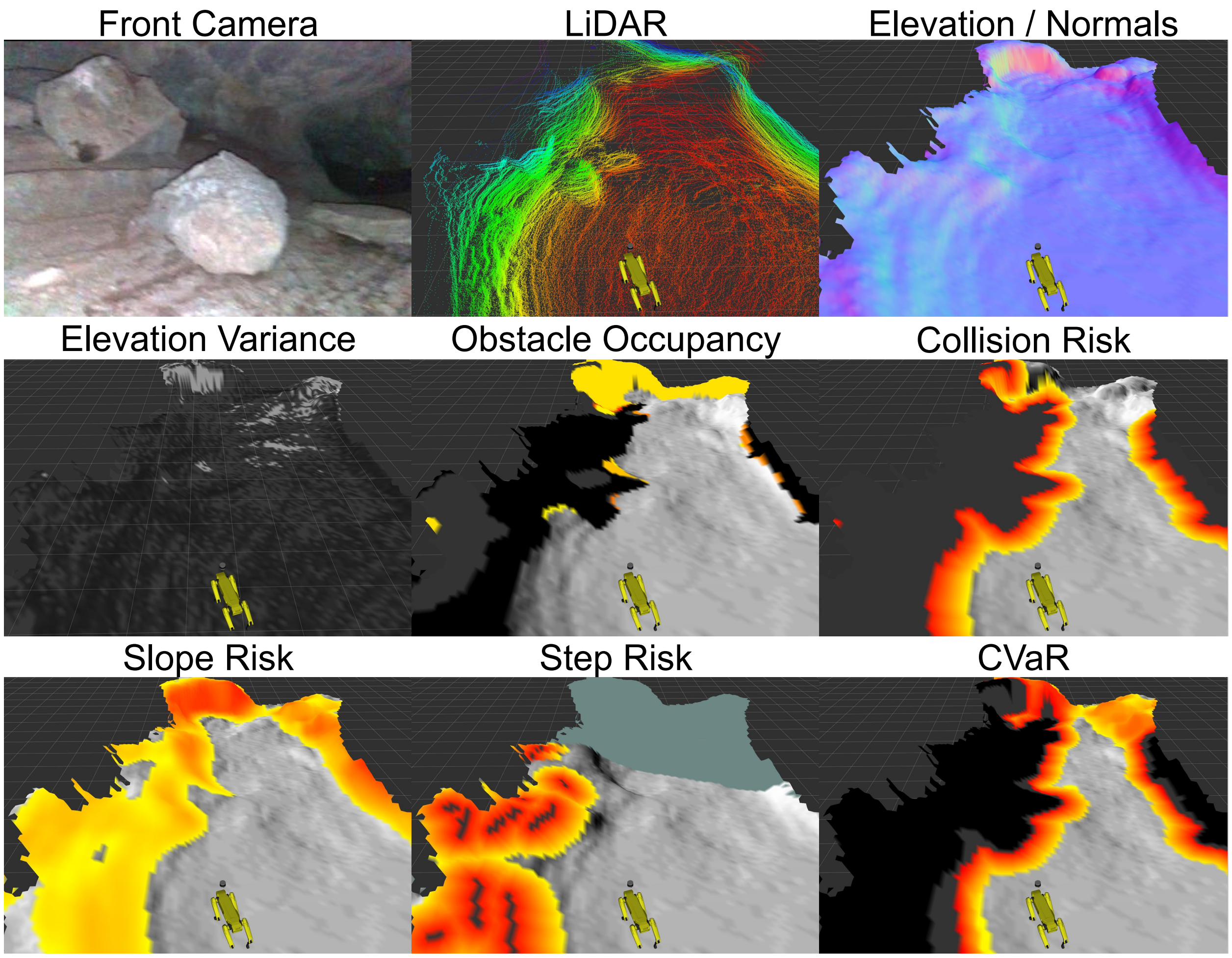


Figure 4.4: Multi-layer traversability risk analysis, which first aggregates recent pointclouds (top). Then, each type of analysis (slope, step, collision, etc.) generates a risk map along with uncertainties (middle rows). These risks are aggregated to compute the final CVaR map (bottom).

$$x_{k+1} = x_k + \Delta t \begin{bmatrix} v_x \cos(p_\theta) - v_y \sin(p_\theta) \\ v_x \sin(p_\theta) + v_y \cos(p_\theta) \\ \gamma v_x + (1-\gamma) v_\theta \\ a_x \\ a_y \\ a_\theta \end{bmatrix} \tag{4.18}$$

4.5.2 Traversability Assessment Models

The traversability cost is assessed as the combination of multiple risk factors. These factors are designed to capture potential hazards for the target robot in the specific environment (Figure 4.4). Such factors include:

- *Collision*: quantified by the distance to the closest obstacle point.
- *Step size*: the height gap between adjacent cells in the grid map. Negative obstacles can also be detected by checking the lack of measurement points in a cell.
- *Tip-over*: a function of slope angles and the robot's orientation.
- *Contact Loss*: insufficient contact with the ground, evaluated by plane-fit residuals.
- *Slippage*: quantified by geometry and the surface material of the ground.
- *Sensor Uncertainty*: sensor and localization error increase the variance of traversability estimates.

To efficiently compute the CVaR traversability cost, we assume the combined factors contribute to a random variable which follows a normal distribution $R \sim \mathcal{N}(\mu, \sigma^2)$. Let φ and $\mathbf{\Phi}$ denote the probability density function and cumulative distribution function of a standard normal distribution respectively. The corresponding CVaR is computed as:

$$\rho(R) = \mu + \sigma \frac{\varphi(\mathbf{\Phi}^{-1}(\alpha))}{1 - \alpha} \tag{4.19}$$

We construct R such that the expectation of R is positive, to keep the CVaR value positive.

4.5.3 Risk-aware Geometric Planning

In order to optimize (4.7) and (4.8), the geometric planner computes an optimal path that minimizes the position-dependent dynamic risk metric in (4.4) along the path. Substituting (4.19) into (4.4), we obtain:

$$J_{\text{pos}}(x_{0:N}) = \mu_0 + \sum_{k=1}^{N} \left[\mu_k + \sigma_k \frac{\varphi(\mathbf{\Phi}^{-1}(\alpha))}{1-\alpha} \right] \tag{4.20}$$

This is easily proved by expanding the terms:

$$\begin{aligned}
J(x_0, \pi) &= R_0 + \rho_0\big(R_1 + \rho_1\big(R_2 + \ldots + \rho_{T-1}(R_T)\big)\big) \\
&= R_0 + \rho\Bigg(R_1 + \rho\Big(R_2 + \ldots + \rho(R_{T-1} + \\
&\qquad \mu_T + \sigma_T \frac{\varphi(\mathbf{\Phi}^{-1}(\alpha))}{1-\alpha})\Big)\Bigg) \\
&= R_0 + \rho\Bigg(R_1 + \rho\Big(R_2 + \ldots + \rho(R_{T-2} + \\
&\qquad \mu_{T-1} + \mu_T + (\sigma_{T-1} + \sigma_T) \frac{\varphi(\mathbf{\Phi}^{-1}(\alpha))}{1-\alpha})\Big)\Bigg) \\
&\vdots \\
&= R_0 + \sum_{i=1}^{T} \left(\mu_i + \sigma_i \frac{\varphi(\mathbf{\Phi}^{-1}(\alpha))}{1-\alpha} \right) \\
&= \sum_{i=0}^{T} \rho(R_i)
\end{aligned}$$

We use the A* algorithm to solve (4.7) over a 2D grid. A* requires a path cost $g(n)$ and a heuristic cost $h(n)$, given by:

$$g(n) = J_{\text{pos}}(x_{0:n}) + \lambda \sum_{k=0}^{n-1} \|x_k - x_{k+1}\|^2 \tag{4.21}$$

$$h(n) = \lambda \|x_n - x_G\|^2 \tag{4.22}$$

For the heuristic cost we use the shortest Euclidean distance to the goal. The value of lambda is a relative weighting between the distance penalty and risk penalty and can be thought of as having units of (traversability cost / m). We use a relatively small value, which means we are mainly concerned with minimizing traversability costs.

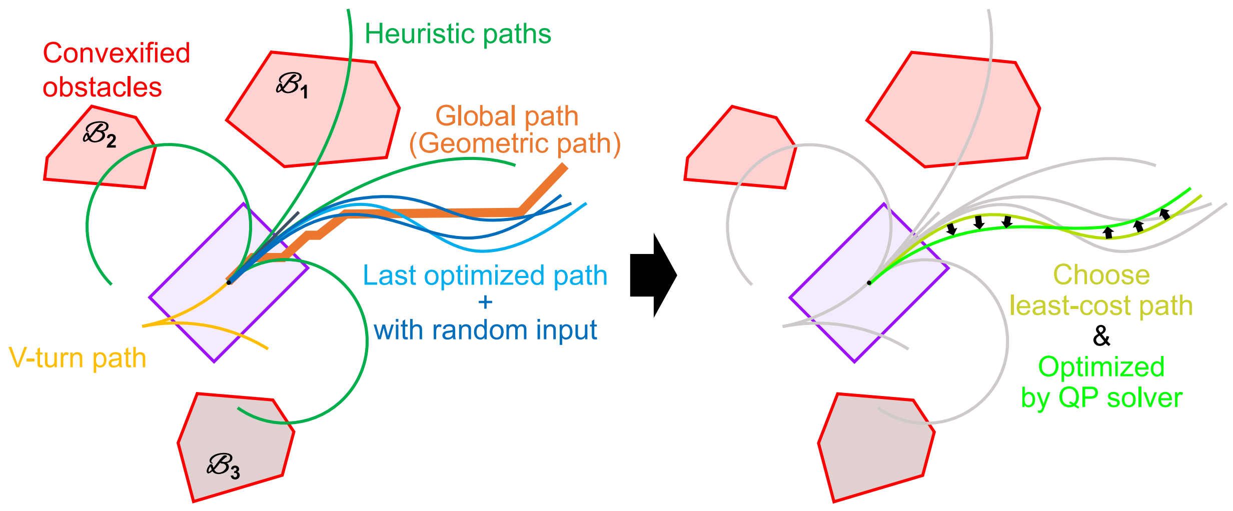

Convexified obstacles
$\mathcal{B}_1$
$\mathcal{B}_2$
$\mathcal{B}_3$
Heuristic paths
Global path (Geometric path)
Last optimized path + with random input
V-turn path
Choose least-cost path & Optimized by QP solver

Figure 4.5: Diagram of kinodynamic MPC planner, which begins with evaluating various paths within a trajectory library. The lowest cost path is chosen as a candidate and optimized by the QP solver.

4.5.4 Risk-aware Kinodynamic Planning

The geometric planner produces a path, i.e. a sequence of poses. We wish to find a kinodynamically feasible trajectory which stays near this path, while satisfying all constraints and minimizing the CVaR cost. To solve (4.9)-(4.12), we use a risk-aware kinodynamic MPC planner, whose steps we outline (Figures 4.5 and 4.6, Algorithm 5).

Trajectory library: Our kinodynamic planner begins with selecting the best candidate trajectory from a trajectory library, which stores multiple initial control and state sequences. The selected trajectory is used as initial solution for solving a full optimization problem. The trajectory library can include: 1) the trajectory accepted in the previous planning iteration, 2) a stopping (braking) trajectory, 3) a geometric plan following trajectory, 4) heuristically defined trajectories (including v-turns, u-turns, and varying curvatures), and 5) randomly perturbed control input sequences.

QP Optimization: Next, we construct a non-linear optimization problem with appropriate costs and constraints (4.9–4.12). We linearize the problem about the initial solution and solve iteratively in a sequential quadratic programming (SQP) fashion [137]. Let $\{\hat{x}_k, \hat{u}_k\}_{k=0:T}$ denote an initial solution. Let $\{\delta x_k, \delta u_k\}_{k=0:T}$ denote deviation from the initial solution. We approximate (4.9-4.12)

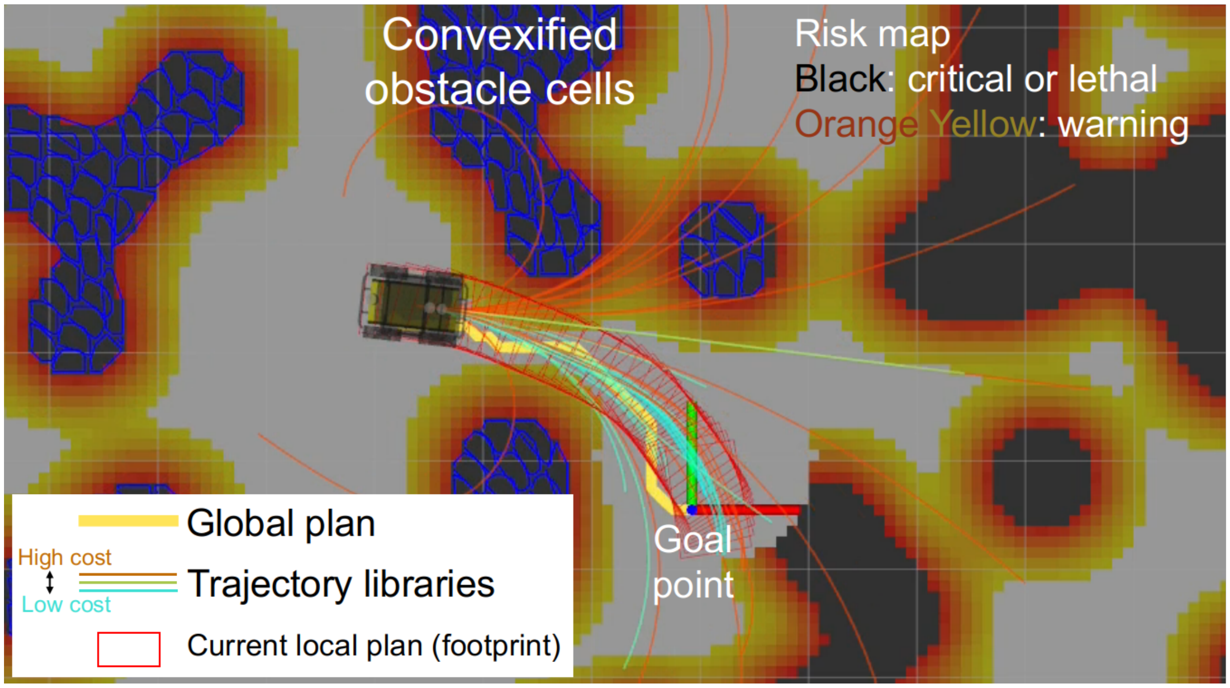


Figure 4.6: Diagram of kinodynamic MPC planner while running live, showing global plan, trajectory library, planned robot footprint, and risk map with convexified obstacle cells.

by a problem with quadratic costs and linear constraints with respect to $\{\delta x, \delta u\}$:

$$\{\delta x^*, \delta u^*\} = \arg\min_{\delta x, \delta u} \sum_{k=0}^{T} \|\hat{x}_k + \delta x_k - x_k^*\|_{Q_k} + \lambda J(\hat{x}_k + \delta x_k, \hat{u}_k + \delta u_k) \tag{4.23}$$

$$s.t. \quad \forall k \in [0, \cdots, T]:$$

$$\hat{x}_{k+1} + \delta x_{k+1} = f(\hat{x}_k, \hat{u}_k) + \nabla_x f \cdot \delta x_k + \nabla_u f \cdot \delta u_k \tag{4.24}$$

$$g(\hat{u_k}) + \nabla_u g \cdot \delta u_k \succ 0 \tag{4.25}$$

$$h(m, \hat{x}_k) + \nabla_x h \cdot \delta x_k \succ 0 \tag{4.26}$$

where $J(\hat{x}_k + \delta x_k, \hat{u}_k + \delta u_k)$ can be approximated with a second-order Taylor approximation (for now, assume no dependence on controls):

$$J(\hat{x} + \delta x) \approx J(\hat{x}) + \nabla_x J \cdot \delta x + \delta x^{\intercal} H(J) \delta x \tag{4.27}$$

and $H(\cdot)$ denotes the Hessian. The problem is now a quadratic program (QP) with quadratic costs

Algorithm 5: Kinodynamic MPC Planner (sequences $\{\cdot_k\}_{k=0:T}$ are expressed as $\{\cdot\}$ for brevity)

```
Input: current state $x_0$, current control sequence (previous solution) $\{u^*\}^{(j)}$
Output: re-planned trajectory $\{x^*\}^{(j+1)}$, re-planned control sequence $\{u^*\}^{(j+1)}$
     Initialization
 1:  $\{x^{\mathrm{r}}\}$ = updateReferenceTrajectory()
 2:  $\{u^*\}^{(j)}$ = stepControlSequenceForward($\{u^*\}^{(j)}$)
     Loop process
 3:  for $i = 0$ to qp_iterations do
 4:      $l$ = generateTrajectoryLibrary($x_0$)
 5:      $[\{x^{\mathrm{c}}\}, \{u^{\mathrm{c}}\}]$ = chooseCandidateFromLibrary($l$)
 6:      $[\{\delta x^*\}, \{\delta u^*\}]$ = solveQP($\{x^{\mathrm{c}}\}, \{u^{\mathrm{c}}\}, \{x^{\mathrm{r}}\}$)
 7:      $[\eta, solved]$ = lineSearch($\{x^{\mathrm{c}}\}, \{\delta x^*\}, \{u^{\mathrm{c}}\}, \{\delta u^*\}$)
 8:      $u^{\mathrm{c}}_k = u^{\mathrm{c}}_k + \eta \delta u^*_k, \ \forall k = 0 : T$
 9:      $\{x^{\mathrm{c}}\}$ = rollOutTrajectory($x_0, \{u^{\mathrm{c}}\}$)
10:  end for
11:  if solved then
12:      $\{x^*\}^{(j+1)}, \{u^*\}^{(j+1)} = \{x^{\mathrm{c}}\}, \{u^{\mathrm{c}}\}$
13:  else
14:      $\{x^*\}^{(j+1)}, \{u^*\}^{(j+1)}$ = getStoppingTrajectory()
15:  end if
16:  return $\{x^*\}^{(j+1)}, \{u^*\}^{(j+1)}$
```

and linear constraints. To solve Equations (4.23-4.26), we introduce the solution vector variable X:

$$X = \begin{bmatrix} \delta x_0^{\mathrm{T}} & \cdots & \delta x_T^{\mathrm{T}} & \delta u_0^{\mathrm{T}} & \cdots & \delta u_T^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \tag{4.28}$$

We can then write Equations (4.23-4.26) in the form:

$$\text{minimize} \quad \frac{1}{2} X^{\mathrm{T}} P X + q^{\mathrm{T}} X \tag{4.29}$$

$$\text{subject to} \quad l \leq AX \leq u \tag{4.30}$$

where P is a positive semi-definite weight matrix, q is a vector to define the first order term in the objective function, A defines inequality constraints and l and u provide their lower and upper limit.

Our MPC problem stated in Equations (4.9-4.12) is non-linear. In order to efficiently find a solution we linearize the problem about an initial solution, and solve iteratively, in a sequential quadratic programming (SQP) fashion [137]. Let $\{\hat{x}_k, \hat{u}_k\}_{k=0,\cdots,T}$ denote an initial solution. Let $\{\delta x_k, \delta u_k\}_{k=0,\cdots,T}$ denote deviation from the initial solution.

In the next subsection we describe these costs and constraints in detail. This is a quadratic program, which can be solved using commonly available QP solvers. In our implementation we use the OSQP solver, which is a robust and highly efficient general-purpose solver for convex QPs [66].

Linesearch: The solution to the SQP problem returns an optimized variation of the control sequence $\{\delta u_k^*\}_{k=0:T}$. We then use a linesearch procedure to determine the amount of deviation $\eta > 0$ to add to the current candidate control policy π: $u_k = u_k + \eta \delta u_k^*$, using Algorithm 6. The resulting correction coefficient is carried over into the next path-planning loop.

Algorithm 6: Linesearch Algorithm

```
Input: candidate control sequence $\{u_k^{\mathrm{c}}\}_{k=0:T}$, QP solution $\{\delta u_k^*\}_{k=0:T}$
Output: correction coefficient $\eta$
    Initialization
 1: initialize $\eta$ by default value or last-used value
 2: $[c, o]$ =getCostAndObstacles($\{u_k^{\mathrm{c}}\}_{k=0:T}$)
    Linesearch Loop
 3: for $i = 0$ to max_iteration do
 4:     for $k = 0$ to $T$ do
 5:         $u_k^{\mathrm{c}(i)} = u_k^{\mathrm{c}} + \eta \delta u_k^*$
 6:     end for
 7:     $[c^{(i)}, o^{(i)}]$ =getCostAndObstacles($\{u_k^{\mathrm{c}(i)}\}_{k=0:T}$)
 8:     if ($c^{(i)} \leq c$ and $o^{(i)} \leq o$) then
 9:         $\eta = \min(2\eta, \eta_{max})$
10:         break
11:     else
12:         $\eta = \max(\eta/2, \eta_{min})$
13:     end if
14: end for
15: return $\eta$
```

Stopping Sequence: If no good solution is found from the linesearch, we pick the lowest cost trajectory from the trajectory library with no collisions. If all trajectories are in collision, we generate an emergency stopping sequence to slow the robot as much as possible (a collision may occur, but hopefully with minimal energy).

Tracking Controller: Having found a feasible and CVaR-minimizing trajectory, we send it to a tracking controller to generate closed-loop tracking behavior at a high rate (¿100Hz), which is specific to the robot type (e.g. a simple cascaded PID, or legged locomotive controller).

4.5.5 Optimization Costs and Constraints

Costs: Note that (4.9) contains the CVaR risk. To linearize this and add it to the QP matrices, we compute the Jacobian and Hessian of ρ with respect to the state x. We efficiently approximate this via numerical differentiation.

Kinodynamic constraints: Similar to the cost, we linearize (4.10) with respect to x and u. Depending on the dynamics model, this may be done analytically.

Control limits: We construct the function $g(u)$ in (4.11) to limit the range of the control inputs. For example in the 6-state dynamics case, we limit maximum accelerations: $|a_x| < a_x^{\max}$, $|a_y| < a_y^{\max}$, and $|a_\theta| < a_\theta^{\max}$.

State limits: Within $h(m, x)$ in (4.12), we encode velocity constraints: $|v_x| < v_x^{\max}$, $|v_y| < v_y^{\max}$, and $|v_\theta| < v_\theta^{\max}$. We also constrain the velocity of the vehicle to be less than some scalar multiple of the risk in that region, along with maximum allowable velocities:

$$|v_\theta| < \kappa_\theta \, \rho(R_k) \tag{4.31}$$

$$\sqrt{v_x^2 + v_y^2} < \kappa_v \, \rho(R_k) \tag{4.32}$$

This reduces the energy of interactions the robot has with its environment in riskier situations, preventing more serious damage.

Position risk constraints: Within $h(m, x_k)$ we would like to add constraints on position and orientation to prevent the robot from hitting obstacles. The general form of this constraint is:

$$\rho(R_k) < \rho^{\max} \tag{4.33}$$

To create this constraint, we locate areas on the map where the risk ρ is greater than the maximum allowable risk. These areas are marked as obstacles, and are highly non-convex. To obtain a convex and tractable approximation of this highly non-convex constraint, we decompose obstacles into non-overlapping 2D convex polygons, and create a signed distance function which determines the minimum distance between the robot's footprint (also a convex polygon) and each obstacle [132].

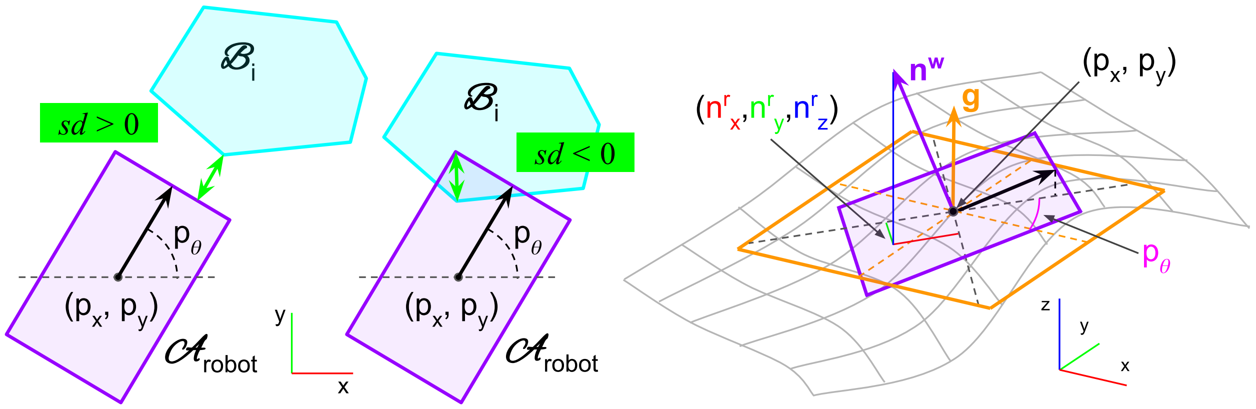


Figure 4.7: Left: Computing convex to convex signed distance function between the robot footprint and an obstacle. Signed distance is positive with no intersection and negative with intersection. Right: Robot pitch and roll are computed from the surface normal rotated by the yaw of the robot. Purple rectangle is the robot footprint with surface normal n^w. $\boldsymbol{g}$ denotes gravity vector, $n^r_{x,y,z}$ are the robot-centric surface normal components used for computing pitch and roll.

Let $\mathcal{A}, \mathcal{B} \subset \mathbb{R}^2$ be two sets, and define the distance between them as:

$$\operatorname{dist}(\mathcal{A}, \mathcal{B}) = \inf\{\|T\| \mid (T + \mathcal{A}) \cap \mathcal{B} \neq \emptyset\} \tag{4.34}$$

where T is a translation. When the two sets are overlapping, define the penetration distance as:

$$\operatorname{penetration}(\mathcal{A}, \mathcal{B}) = \inf\{\|T\| \mid (T + \mathcal{A}) \cap \mathcal{B} = \emptyset\} \tag{4.35}$$

Then we can define the signed distance between the two sets as:

$$\operatorname{sd}(\mathcal{A}, \mathcal{B}) = \operatorname{dist}(\mathcal{A}, \mathcal{B}) - \operatorname{penetration}(\mathcal{A}, \mathcal{B}) \tag{4.36}$$

We then include within $h(m, x_k)$ a constraint to enforce the following inequality:

$$\operatorname{sd}(\mathcal{A}_{\text{robot}}, \mathcal{B}_i) > 0 \quad \forall i \in \{0, \cdots, N_{\text{obstacles}}\} \tag{4.37}$$

Note that the robot footprint $\mathcal{A}_{\text{robot}}$ depends on the current robot position and orientation: $\mathcal{A}_{\text{robot}}(p_x, p_y, p_\theta)$, while each obstacle $\mathcal{B}_i(m)$ is dependent on the information in the map (See Figure 4.7).

Orientation constraints: We wish to constrain the robot's orientation on sloped terrain in such a way as to prevent the robot from rolling over or performing dangerous maneuvers. To do this, we add constraints to $h(m, x_k)$ which limit the roll and pitch of the robot as it settles on the surface of the ground. Denote the position as $p = [p_x, p_y]^\intercal$ and the position/yaw as $s = [p_x, p_y, p_\theta]^\intercal$. Let the robot's pitch be ψ and roll be ϕ in its body frame. Let $\omega = [\psi, \phi]^\intercal$. The constraint will have the form $|\omega| \prec \omega^{\max}$. At p, we compute the surface normal vector, call it $n^w = [n^w_x, n^w_y, n^w_z]^\intercal$, in the world frame. Let $n^r = [n^r_x, n^r_y, n^r_z]^\intercal$, be the surface normal in the body frame, where we rotate by the robot's yaw: $n^r = R_\theta n^w$ (see Figure 4.7), where R_θ is a basic rotation matrix by the angle θ about the world z axis.

Then, we define the robot pitch and roll as $\omega = g(n^r)$ where:

$$\omega = g(n^r) = \begin{bmatrix} \operatorname{atan2}(n^r_x, n^r_z) \\ -\operatorname{atan2}(n^r_y, n^r_z) \end{bmatrix} \tag{4.38}$$

Note that ω is a function of s. Creating a linearly-constrained problem requires a linear approximation of the constraint:

$$R_\theta = \begin{bmatrix} \cos p_\theta & \sin p_\theta & 0 \\ -\sin p_\theta & \cos p_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.39}$$

Let the robot pitch and roll vector ω be defined as $\omega = g(n^r)$, where:

$$\omega = g(n^r) = \begin{bmatrix} \operatorname{atan2}(n^r_x, n^r_z) \\ -\operatorname{atan2}(n^r_y, n^r_z) \end{bmatrix} \tag{4.40}$$

Creating a linearly-constrained problem requires a linear approximation of the constraint:

$$|\nabla_s \omega(s) \delta s + \omega(s)| <= \omega_{max} \tag{4.41}$$

Conveniently, computing $\nabla_s \omega(s)$ reduces to finding gradients w.r.t position and yaw separately. Let $\nabla_s \omega(s) = [\nabla_p \omega(s), \nabla_\theta \omega(s)]^\intercal$, then:

$$\nabla_p \omega(s) = (\nabla_{n^r} g)(R_\theta)(\nabla_p n^w) \tag{4.42}$$

$$\nabla_\theta \omega(s) = (\nabla_{n^r} g)(\frac{d}{d\theta} R_\theta)(n^w) \tag{4.43}$$

where:

$$\nabla_{n^r} g = \begin{bmatrix} \frac{n_z^r}{(n_x^r)^2+(n_z^r)^2} & 0 & \frac{-n_x^r}{(n_x^r)^2+(n_z^r)^2} \\ 0 & \frac{-n_z^r}{(n_y^r)^2+(n_z^r)^2} & \frac{n_y^r}{(n_y^r)^2+(n_z^r)^2} \end{bmatrix} \tag{4.44}$$

and

$$\nabla_p n^w = \begin{bmatrix} \frac{\partial n_x^w}{\partial p_x} & \frac{\partial n_x^w}{\partial p_y} \\ \frac{\partial n_y^w}{\partial p_x} & \frac{\partial n_y^w}{\partial p_y} \\ \frac{\partial n_z^w}{\partial p_x} & \frac{\partial n_z^w}{\partial p_y} \end{bmatrix} \tag{4.45}$$

The terms with the form $\frac{\partial n_x^w}{\partial p_x}$ amount to computing a second-order gradient of the elevation on the 2.5D map. This can be done efficiently with numerical methods [136].

Box Constraint: Note that if δx and δu are too large, linearization errors will dominate. To mitigate this we also include box constraints within (4.11) and (4.12) to maintain a bounded deviation from the initial solution: $|\delta x| < \epsilon_x$ and $|\delta u| < \epsilon_u$.

Adding Slack Variables: To further improve the feasibility of the optimization problem we introduce auxilliary slack variables for constraints on state limits, position risk, and orientation. For a given constraint $h(x) > 0$ we introduce the slack variable ϵ, and modify the constraint to be $h(x) > \epsilon$ and $\epsilon < 0$. We then penalize large slack variables with a quadratic cost: $\lambda_\epsilon \epsilon^2$. These are incorporated into the QP problem (4.29) and (4.30).

4.5.6 Dynamic Risk Adjustment

The CVaR metrics allows us to dynamically adjust the level and severity of risk we are willing to accept. Selecting low α reverts towards using the mean cost as a metric, leading to optimistic decision making while ignoring low-probability but high cost events. Conversely, selecting a high α leans towards conservatism, reducing the likelihood of fatal events while reducing the set of possible paths. We adjust α according to two criteria: 1) Mission-level states, where depending on the robot's role, or the balance of environment and robot capabilities, the risk posture for individual robots may differ. 2) Recovery Behaviors, where if the robot is trapped in an unfavorable condition, by gradually decreasing α, an escape plan can be found with minimal risk. These heuristics are especially useful in the case of risk-aware planning, because the feasibility of online nonlinear MPC

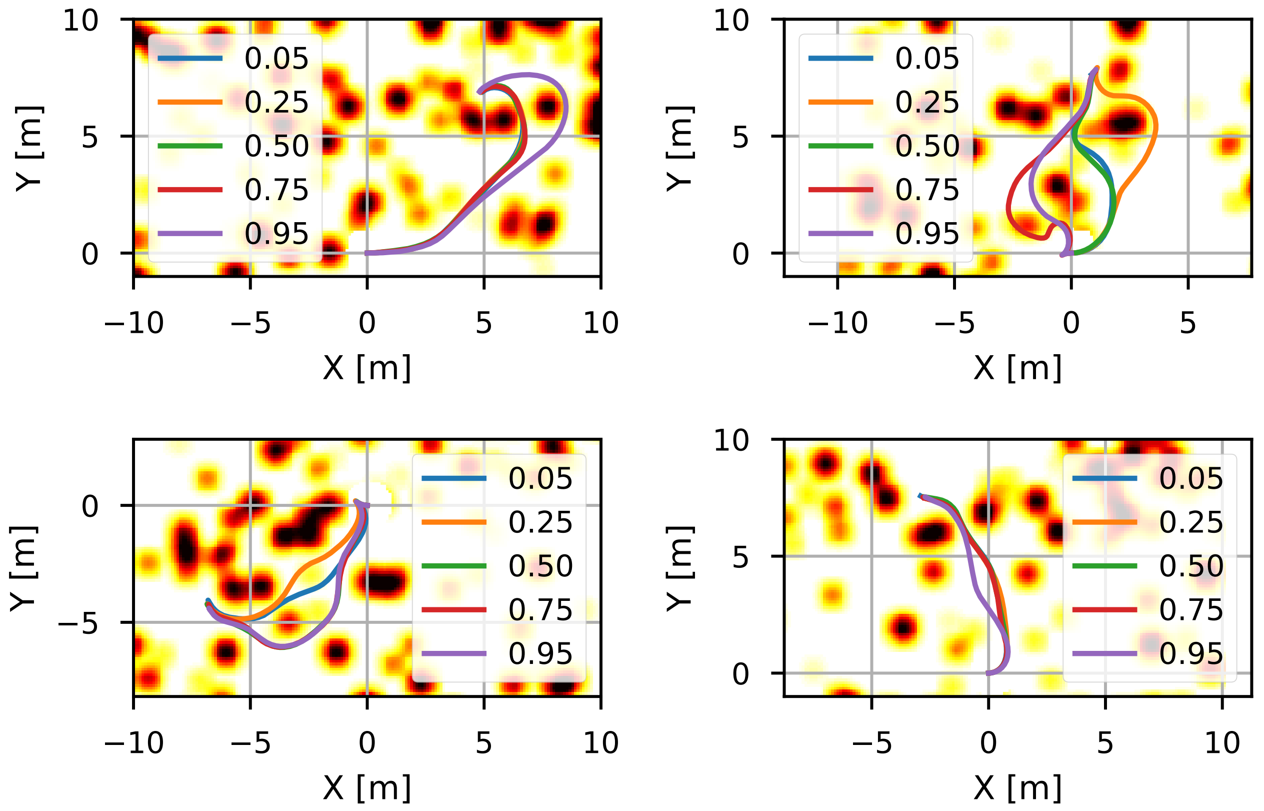


Figure 4.8: Path distributions from four simulated runs. The risk level α spans from 0.1 (close to mean-value) to 0.95 (conservative). Smaller α typically results in a shorter path, while larger α chooses statistically safe paths.

is difficult to guarantee. When no feasible solution is found for a given risk level α, a riskier but feasible solution can be quickly found and executed.

4.6 Experiments

In this section, we report the performance of STEP. We first present a comparative study between different adjustable risk thresholds in simulation on a wheeled differential drive platform. Then, we demonstrate real-world performance using a legged platform deployed at a lava tube environment.

4.6.1 Simulation Study

To assess statistical performance, we perform 50 Monte-Carlo simulations with randomly generated maps and goals. Random traversability costs are assigned to each grid cell. The following assump-

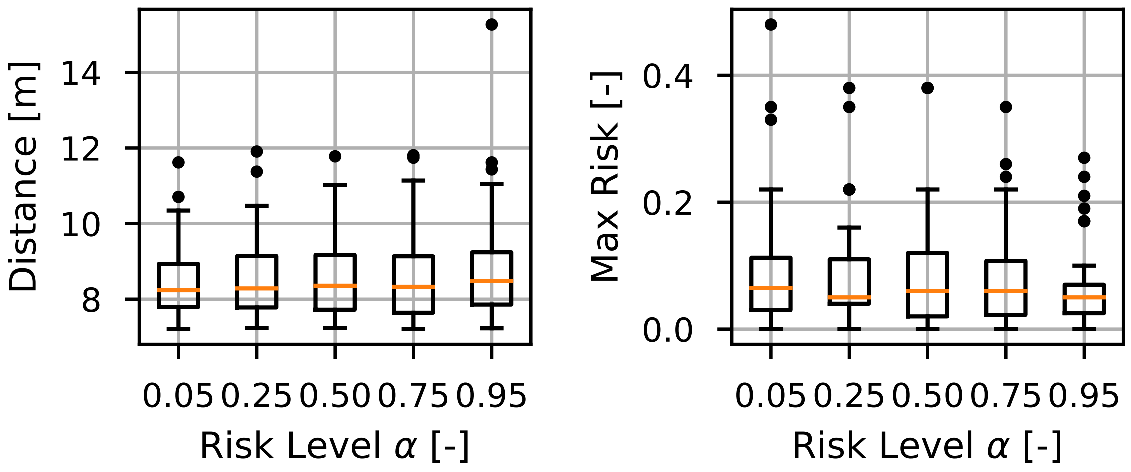


Figure 4.9: Distance vs risk trade-off from 50 Monte-Carlo simulations. Left: Distributions of path distance. Right: Distributions of max risk along the traversed paths. Box plot uses standard quartile format and dots are outliers.

tions are made: 1) no localization error, 2) no tracking error, and 3) a simplified perception model with artificial noise. We give a random goal 8 m away and evaluate the path cost and distance. We use a differential-drive dynamics model (no lateral velocity).

We compare STEP using different α levels. Figure 4.8 shows the distribution of paths for different planning configurations. The optimistic (close to mean-value) planner $\alpha = 0.05$ typically generates shorter paths, while the conservative setting $\alpha = 0.95$ makes long detours to select statistically safer paths. The other α settings show distributions between these two extremes, with larger α generating similar paths to the conservative planner and smaller α generating more time-optimal paths. Statistics are shown in Figure 4.9.

4.6.2 Hardware Results

We deployed STEP on a Boston Dynamics Spot quadruped robot at the Valentine Cave in Lava Beds National Monument, Tulelake, CA. The robot was equipped with custom sensing and computing units, and driven by JPL's NeBula autonomy software [138]. The main sensor for localization and traversability analysis is a Velodyne VLP-16, fused with Spot's internal Intel realsense data to cover blind spots. The entire autonomy stack runs on an Intel Core i7 CPU. The typical CPU usage for the traversability stack is about a single core.

Figure 4.10 shows the interior of the cave and algorithm's representations. The rough ground surface, rounded walls, ancient lava waterfalls, steep non-uniform slopes, and boulders all pose significant traversability stresses. Furthermore, there are many occluded places which affect the confidence in traversability estimates.

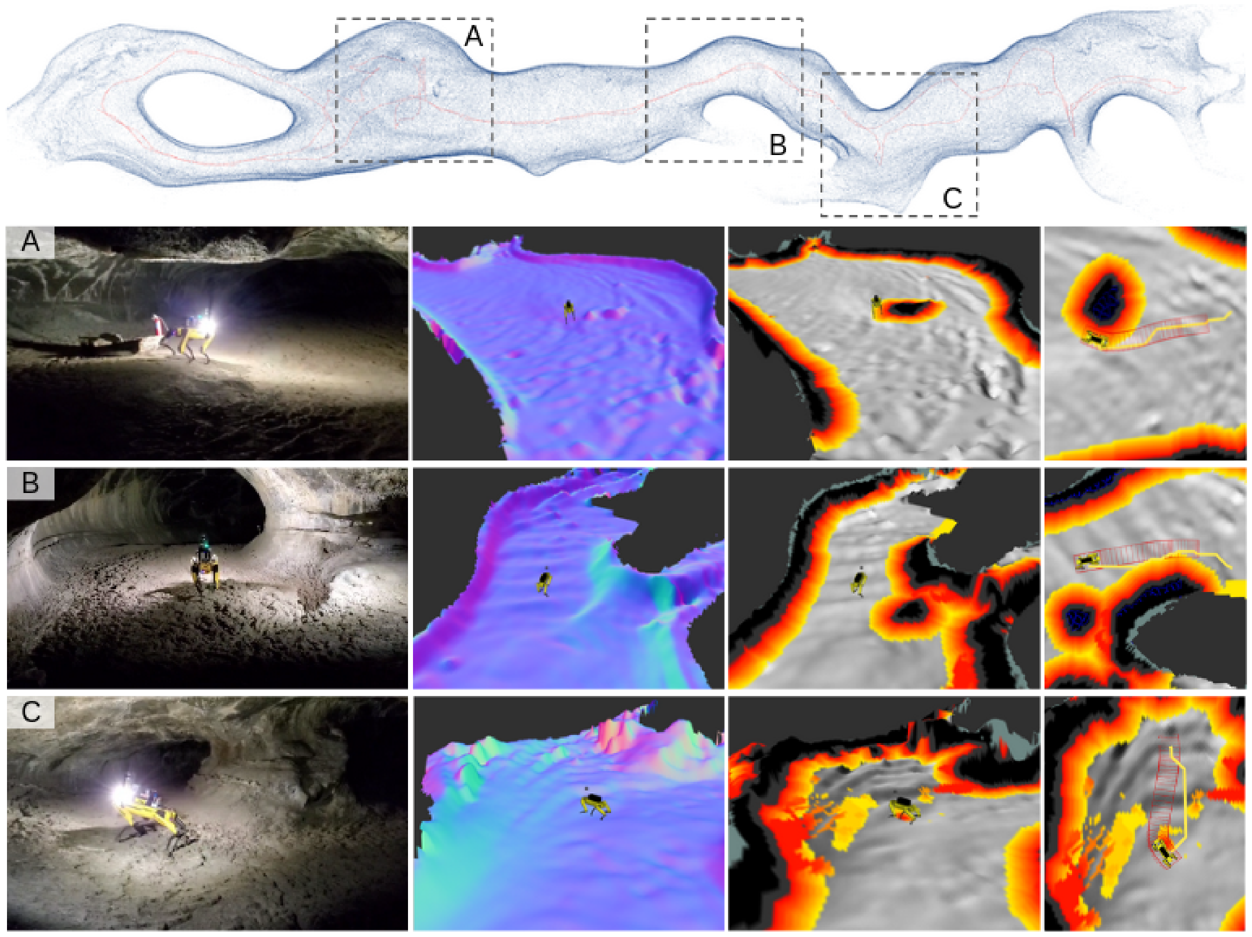


Figure 4.10: Traversability analysis results for the Valentine Cave experiment. From left to right: Third-person view, elevation map (colored by normal direction), risk map (colored by risk level. white: safe ($r <= 0.05$), yellow to red: moderate ($0.05 < r <= 0.5$), black: risky ($r > 0.5$)), and planned geometric/kinodynamic paths (yellow lines/red boxes).

We tested our risk-aware traversability software during our fully autonomous runs. The planner was able to navigate the robot safely to the every goal provided by the upper-layer coverage planner [138] despite the challenges posed by the environment. Figure 4.10 shows snapshots of elevation maps, CVaR risk maps, and planned paths. The risk map captures walls, rocks, high slopes, and ground roughness as mobility risks. STEP enables Spot to safely traverse the entire extent of the lava tube, fully exploring all regions. STEP navigates 420 meters over 24 minutes, covering 1205 square meters of rough terrain.

4.7 Conclusion

We have presented STEP (Stochastic Traversability Evaluation and Planning), our approach for autonomous robotic navigation in unsafe, unstructured, and unknown environments. We believe this approach finds a sweet-spot between computation, resiliency, performance, and flexibility when compared to other motion planning approaches in such extreme environments. Our method is generalizable and extensible to a wide range of robot types, sizes, and speeds, as well as a wide range of environments.

CHAPTER 5
COSTMAP LEARNING FOR RISK-AWARE TRAVERSABILITY IN CHALLENGING ENVIRONMENTS

5.1 Summary

One of the main challenges in autonomous robotic exploration and navigation in unknown and unstructured environments is determining where the robot can or cannot safely move. A significant source of difficulty in this determination arises from stochasticity and uncertainty, coming from localization error, sensor sparsity and noise, difficult-to-model robot-ground interactions, and disturbances to the motion of the vehicle. Classical approaches to this problem rely on geometric analysis of the surrounding terrain, which can be prone to modeling errors and can be computationally expensive. Moreover, modeling the distribution of uncertain traversability costs is a difficult task, compounded by the various error sources mentioned above. In this work, we take a principled learning approach to this problem. We introduce a neural network architecture for robustly learning the distribution of traversability costs. Because we are motivated by preserving the life of the robot, we tackle this learning problem from the perspective of learning tail-risks, i.e. the Conditional Value-at-Risk (CVaR). We show that this approach reliably learns the expected tail risk given a desired probability risk threshold between 0 and 1, producing a traversability costmap which is more robust to outliers, more accurately captures tail risks, and is more computationally efficient, when compared against baselines. We validate our method on data collected a legged robot navigating challenging, unstructured environments including an abandoned subway, limestone caves, and lava tube caves.

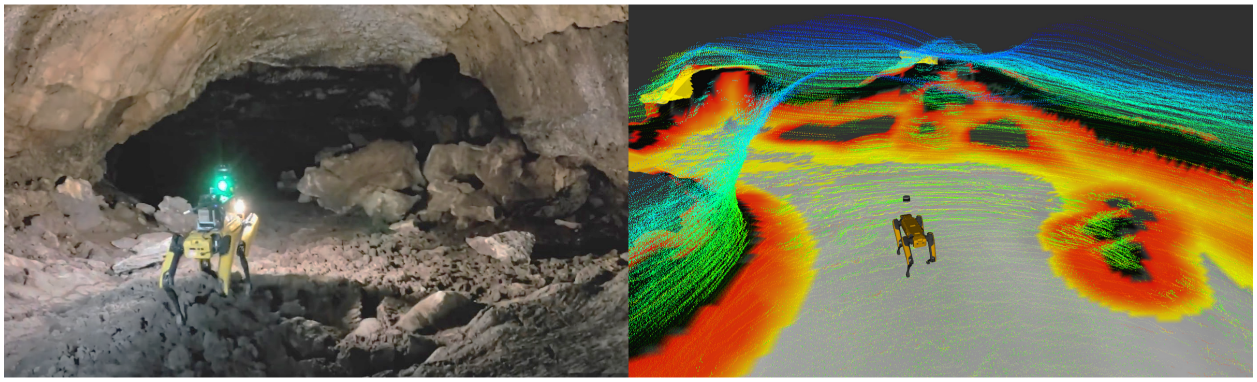


Figure 5.1: Top: Spot autonomously exploring Valentine Cave, Lava Beds National Monument, CA, USA. Bottom: LiDAR point cloud and computed costmap in the same environment. In this work, we aim to infer a CVaR costmap from the LiDAR point cloud.

5.2 Introduction

Uncertainty is ever-present in robotic sensing and navigation. Localization error can severely degrade the quality of a robot's environment map, leading to a robot over-estimating or under-estimating the safety of traversing some particular terrain [111]. Sensor noise, sparsity, and occlusion can similarly degrade localization and mapping performance, especially in perceptually degraded environments such as in dark, dust, or featureless environments [139]. Moreover, the environment itself is a constant source of mobility uncertainty, as ground-vehicle interactions are notoriously difficult to model and even more difficult to accurately compute in real time - especially in unstructured environments such as those filled with slopes, rough terrain, low traction, rubble, narrow passages, and the like (Figure 5.1) [110]. In each case, modeling these uncertainties often relies on computationally tractable but distributionally restrictive assumptions. For example, an EKF used for localization assumes a Gaussian distribution and known process and observation noise [140]. Or, often computing traversability means calculating worst-case bounds on the uncertainty of the robot's settled pose on a patch of terrain [116]. Note that these uncertainties are not additive, often compounding and interacting. Localization error can lead to a degraded map, which and result in poor traversability estimates. These poor estimates can lead the robot into dangerous and out-of-nominal actions, which in turn may affect sensor measurement quality, which then affects localization error.

To escape this vicious cycle, we would like to have a system which more accurately quantifies the uncertainty of computing traversability costs. Classically, traversability methods do not take into account this uncertainty, instead relying on building a 2.5D terrain map which is used to extract features such as maximum, minimum, and variance of the height and slope of the terrain [114]. Planning algorithms for such methods take into account the stability of the robot on the terrain [115].

Computing uncertainties allows the robot to have some measure of control over the level of risk it is willing to take. The concept of risk arises naturally when talking about uncertainty and its effect on robot safety. Always planning for the worst-case scenario is often not desirable as it can lead to too conservative behavior, which will not allow the robot to accomplish its task. On the other hand, ignoring uncertainties entirely will also lead to task failure if the robot does not survive. It is therefore advantageous to have fine-grained control over the level of risk the robot should take - this control can be used by task or mission-level planners which weigh a variety of factors to decide the best course of action [141].

Risk measures, often used in finance and operations research, provide a mapping from a random variable (usually the cost) to a real number. These risk metrics should satisfy certain axioms in order to be well-defined as well as to enable practical use in robotic applications [23]. Conditional value-at-risk (CVaR) is one such risk measure that has this desirable set of properties, and is a part of a class of risk metrics known as *coherent risk measures* [124].

Accurately constructing these risk measures poses a challenge, often relying on distributional assumptions such as Gaussianity. For example, in [116, 117], the authors estimate the probability distributions of states based on the kinematic model of the vehicle and the terrain height uncertainty. A method for incorporating sensor and state uncertainty to obtain a probabilistic terrain estimate in the form of a grid-based elevation map was considered in [118]. In our previous work [26], we model traversability risks as Gaussian random variables and use them to efficiently construct risk measures which are useful for kinodynamic planning. While these approaches have the advantage of allowing for efficient, kinodynamic, risk-aware planning, they may suffer from relying too heavily on these modeling assumptions. Such assumptions include: 1) Localization error is accurately known (and is Gaussian), 2) various component risks can be accurately determined from a map (such as slope, step size, roughness, collision), and 3) these risks are normally distributed and both mean and variances can be accurately modeled.

Learning CVaR from data is an attractive approach to bypassing these complex issues. The concept of learning CVaR itself has generated much theory and has been applied to various fields including finance and reinforcement learning [112, 142, 143]. These methods have recently begun to gain interest to the deep learning community. Since [20] showed that minimizing a *quantile* loss function results in unbiased prediction of the VaR, or quantile function, many deep learning applications of this idea have been examined [144, 94]. However, there is little work on predicting CVaR directly. Instead, it is typically computed by sampling from the risk variable of interest [145], or relies on assumptions of the underlying distribution. More recently, [146] propose a method for predicting CVaR with a deep neural network under assumptions of i.i.d. samples, and prove convergence of the scheme. In this work, we propose an approach for distribution-free CVaR learning which scales to images and large datasets. We construct loss and evaluation functions for both VaR and CVaR which enforce monotonicity with respect to the risk probability.

Learning-based costmaps have seen more recent development for robotics and autonomous driving [147]. The concept is attractive due to possibly for bypassing traditional modeling approaches, reducing computation, improving performance from data [148]. Recent work on CVaR based costmaps rely on assumptions of distribution (gaussian) or sampling to estimate CVaR [149]. [150] learns an approximating CVaR costmap from precomputed CVaR values on a collision-avoidance task. Our contribution to this space is to propose learning CVaR directly in a distribution-free manner, with a novel application to navigation in challenging terrain.

We propose an architecture in which raw or minimally processed point cloud data is transformed and fed into a CNN (Convolutional Neural Network). In contrast to other approaches, the network directly produces a CVaR costmap which encodes the traversability risk, given a desired probability of confidence. Note that in this work we treat the risks in each cell in the costmap as *static* and independent of the others, that is, we do not consider the risk of traversing multiple cells sequentially. (See recent work [26, 23] which addresses the notion of dynamic risk metrics in the context of planning and assessing risk over a path). We restrict our approach to assessing point-wise risks which can then be used to commit the robot to finding a path which uses these risks as a constraint, avoiding the riskiest regions and thereby remaining safe. (In contrast, in [26] we minimize the CVaR of the risk along an entire path.)

Our contributions are summarized as follows:

- A novel neural network architecture for transforming pointcloud data into risk-aware costmaps.
- A loss function which trains this network to produce quantile and CVaR values without distribution assumptions.
- A solution to a challenging traversability learning task in unknown environments, validated on a wide range of unstructured and difficult terrain.
- Evaluation metrics which assess the goodness-of-fit of the network and comparison between different baseline approaches.

5.3 Method

In this section we establish definitions and describe our method for learning CVaR costmaps. We first formally define our notion of ”traversability”. Then we discuss risk metrics, Value-at-Risk (VaR) and Conditional Value-at-Risk (CVaR). We discuss the losses used to train a network to produce these values, and describe how we obtain training labels.

5.3.1 Traversability as a random variable

We define *traversability* as the capability for a ground vehicle to reside over a terrain region under an admissible state [113]. We represent traversability as scalar value which indicates the magnitude of damage (or cost of repair) the robot will experience if placed in that state :

$$r = \mathcal{R}(m, x) \tag{5.1}$$

where $x \in \mathcal{X}$ is the robot position in 2d coordinates, $m \in \mathcal{M}$ represents the current belief of the local environment, $r \in \mathbb{R}$, and $\mathcal{R}(\cdot)$ is a traversability assessment model. This model captures various unfavorable events such as collision, getting stuck, tipping over, high slippage, to name a few. A mobility platform has a unique assessment model which reflects its mobility capability.

Associated with the true traversability value is a distribution over possible values based on the current understanding about the environment and robot state. In most real-world applications where perception capabilities are limited, the true value can be highly uncertain. To handle this uncertainty, consider a map belief, i.e., a probability distribution $p(m|x_{0:k}, z_{0:k})$ over a possible set $\mathcal{M}$, where $z_{0:k}$ are sensor observations. Then, the traversability estimate is also represented as a random variable $R : (\mathcal{M} \times \mathcal{X}) \longrightarrow \mathbb{R}$. We call this probabilistic mapping from map belief and state to possible traversability cost values a *risk assessment model.*

5.3.2 Risk Metrics, VaR and CVaR

A risk metric $\rho(R) : R \to \mathbb{R}$ is a mapping from a random variable to a real number which quantifies some notion of risk. Local environment information is encoded within m. This can be raw sensor data, e.g. observations, or processed map data (e.g. represented as $m = (m^{(1)}, m^{(2)}, \cdots)$ where m^i is the i-th element of the map). In the case of a 2D or 2.5D representation, i represents the cell index. Risk metrics can be coherent, which mean they satisfy six properties with respect to the random variables they act upon, namely: normalized, monotonic, sub-additive, positive homogeneity, and translation invariant. [23] argues that the consideration of risk in robotics should utilize coherent risk metrics for their intuitive and well-formulated properties. In this work we isolate the risk quantification problem from the planning problem. This avoids the need to ensure the construction of a *dynamic* risk metric, in the language of [23], which will make learning more tractable. This comes at a cost that minimizing the total risk along a path is not as simple as adding up the risks of individual states. Instead, we consider risks point-wise, and seek a path which does not exceed a tolerable level of risk at any point in time. In this work we are concerned with right-tail risk only, since the random variable R is a positive "cost" for which we seek to avoid high values.

Let $\alpha \in [0,1]$ denote the *risk probability level*. High values of α imply more risk. The Value-at-Risk of the random variable $R(m,x)$ can be defined by (we write R for brevity):

$$\mathrm{VaR}_\alpha(R) := \inf\{z \in \mathbb{R} | P(R < z) > \alpha\} \tag{5.2}$$

This is also known as the α-quantile. While there are multiple ways to define the Conditional Value-at-Risk, one common definition is as follows:

$$\mathrm{CVaR}_\alpha(R) := \mathbb{E}[R | R \geq \mathrm{VaR}_\alpha(R)]. \tag{5.3}$$

Building on the work of [112], [151] shows that CVaR can also be written as:

$$\mathrm{CVaR}_\alpha(R) = \mathrm{VaR}_\alpha(R) + \frac{1}{1-\alpha}\mathbb{E}[(R - \mathrm{VaR}_\alpha(R))_+] \tag{5.4}$$

where $(\cdot)_+ = \max(\cdot, 0)$.

5.3.3 Learning VaR and CVaR

We wish to construct a model with inputs m, x, and α, and outputs $\mathrm{CVaR}_\alpha(R(m,x))$, parameterized by network weights θ. We denote this model as $C_\theta(m,x,\alpha)$. To learn CVaR we take a joint approach where we learn both VaR and CVaR together. We construct a similar VaR model, also as a function of m and x as $\mathrm{VaR}_\alpha(R(m,x))$, which we denote $V_\theta(m,x,\alpha)$. Learning VaR can be accomplished by minimizing the Koenker-Bassett error with respect to θ [20]:

$$l_\alpha^V(\theta) = \alpha(R(m,x) - V_\theta(m,x,\alpha))_+ + (1-\alpha)(R(m,x) - V_\theta(m,x,\alpha))_- \tag{5.5}$$

where $(\cdot)_+ = \max(\cdot, 0)$ and $(\cdot)_- = \min(\cdot, 0)$. From the estimated VaR, we can compute the expected CVaR and construct an L1-loss for C_θ:

$$l_\alpha^C(\theta) = |C_\theta(m,x,\alpha) - R(m,x)| \mathbb{1}_{R(m,x) \geq V_\theta(m,x,\alpha)} \tag{5.6}$$

With the assumption of i.i.d. sampled data, when $l_\alpha^V(\theta)$ and $l_\alpha^C(\theta)$ are minimized, V_θ will approximate VaR and C_θ will approximate CVaR. If during training we uniformly randomly sample values of $\alpha \in [0,1]$, then the input α to these models should be meaningful as well.

In practice, a few modifications to these losses are needed to improve numerical stability. First, similar to [94], we smooth the quantile loss function l_α^V near the inflection point $V_\theta = R$ by using a modified Huber loss [97]:

$$l_h(e,\alpha) = \begin{cases} (1-\alpha)|e| & \text{if } e \le \frac{-h}{1-\alpha} \\ \frac{1}{2h}((1-\alpha)|e|)^2 + \frac{h}{2} & \text{if } \frac{-h}{1-\alpha} < e \le 0 \\ \frac{1}{2h}(\alpha|e|)^2 + \frac{h}{2} & \text{if } 0 < e \le \frac{h}{\alpha} \\ \alpha|e| & \text{if } \frac{h}{\alpha} < e \end{cases} \tag{5.7}$$

where $h \in \mathbb{R}^+$ controls how much smoothing is added (Figure 5.2). The new VaR loss is then:

$$\hat{l}_\alpha^V(\theta) = l_h(R(m,x) - V_\theta(m,x,\alpha), \alpha). \tag{5.8}$$

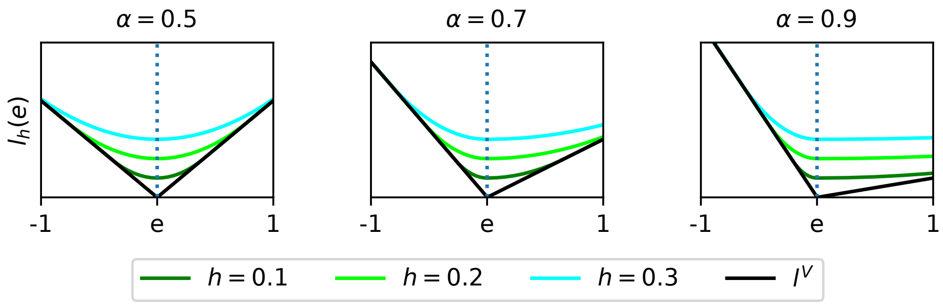


Figure 5.2: Modified quantile Huber loss $l_h(e,\alpha)$, for varying values of α and h.

Second, instead of learning C_θ directly, we can learn the residual between C_θ and V_θ. Suppose we construct a model $\hat{C}_\theta$ instead of C_θ, and let $C_\theta = V_\theta + \hat{C}_\theta$. Then the loss $l_\alpha^C(\theta)$ can be written as:

$$l_\alpha^C(\theta) = |\hat{C}_\theta - (R(m,x) - V_\theta(m,x,\alpha))| \mathbb{1}_{R \ge V_\theta} \tag{5.9}$$

This serves to separate the error signals between the two models.

Third, we introduce a monotonic loss to enforce that increasing values of α result in increasing values of $V_\theta(m,x,\alpha)$ and $C_\theta(m,x,\alpha)$. This can be done by penalizing negative divergence of the output with respect to α [101, 25]. We also introduce a smoothing function to prevent instability near the inflection point when the divergence equals 0. The total monotonic loss is:

$$d_V = (\nabla_\alpha V_\theta(m, x, \alpha))_- \tag{5.10}$$

$$d_C = (\nabla_\alpha C_\theta(m, x, \alpha))_- \tag{5.11}$$

$$s(d) = \exp(d) - d - 1.0 \tag{5.12}$$

$$l^m(\theta) = s(d_V) + s(d_C) \tag{5.13}$$

In practice we find that under gradient-based optimization, this loss decreases to near 0 in the first epoch and does not noticeably affect the minimization of the quantile and CVaR losses.

To summarize, the total loss function is the sum of the modified huber quantile loss (5.8), the L1 residual CVaR loss (5.9), and the monotonic loss (5.13):

$$L(\theta; R, m, x, \alpha) = \lambda_V \tilde{l}_\alpha^V(\theta) + \lambda_C l_\alpha^C(\theta) + \lambda_m l^m(\theta) \tag{5.14}$$

We minimize this loss over a dataset, sampled i.i.d from the distribution of R, which is a function of m and x. We also sample from a uniform distribution for α. In other words, we seek to minimize the expected loss with respect to θ:

$$\begin{aligned}\mathbb{E}_{R,\alpha}[L(\theta; R, m, x\alpha)] &= \int L(\theta; R, m, x, \alpha) p(R) p(\alpha) dR d\alpha \\ &= \int L(\theta; R, m, x, \alpha) p(R|m, x) p(m, x) p(\alpha) dm dx d\alpha \end{aligned} \tag{5.15}$$

This expectation is approximated using stochastic gradient descent [152]. Given the dataset $\mathcal{D} = \{m_n, x_n\}_n$, the distribution $p(m, x)$ is sampled from the distribution of data collected by the robot as it moves in the environment, observing samples of its position x and its environment data m. The distribution $p(R|m, x)$ is sampled by determining the traversability cost given a sampled (m, x). This determination may be stochastic or imprecise, and may possibly come from any traversability assessment method. Happily, our learning approach should capture this stochasticity. In Figure 5.3, we demonstrate this learning approach on a toy 1D problem. We use a 3 layer feed-forward MLP neural network to learn the state-varying VaR and CVaR values of a random variable. We

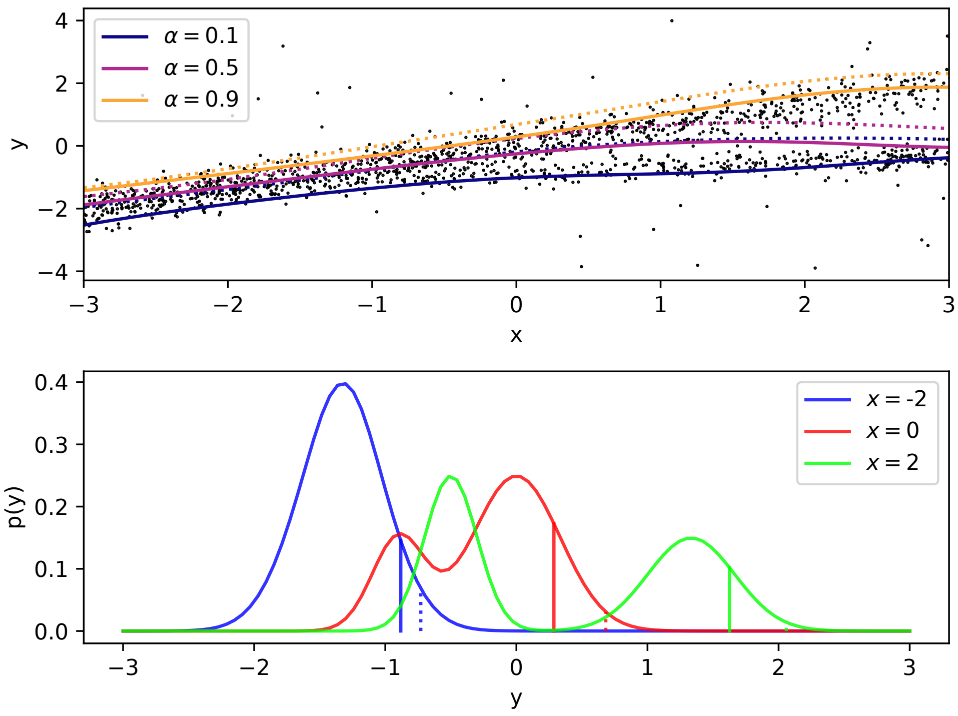


Figure 5.3: Learned VaR and CVaR on a toy 1D problem. Top: Samples drawn from a distribution which is multimodal and heteroskedastic. Solid and dotted lines show learned VaR and CVaR levels, respectively, for different values of α. Bottom: PDF of the true distribution for varying values of x. Also marked are the learned VaR (solid vertical line) and CVaR (dotted vertical line) values.

notice that our learning approach captures the risk of the known distributions accurately. Next, we describe the application of this loss function to learning unknown distributions of 2D traversability costs from 3D point clouds.

5.3.4 Obtaining Ground Truth Labels

As mentioned, computing traversability costs from sensor data is a rich field in itself. Computing these costs may include geometric analysis [114], semantic understanding and detection [153], proprioceptive sensing [154], or hand-labeling. Our approach is extensible to both dense and sparse ground truth labels in the costmap. In this work we focus on costs arising through geometric analysis. We leverage prior work ([26]), which itself follows a tradition of geometric analyses for ground vehicle traversal [116, 114]. These analyses are based on point cloud data and provide model-based costs which are computed in a dense local region around the robot, but are affected by sensor occlusion, sparsity, noise, and localization error. These sources of ambiguity and stochasticity are exactly what we aim to capture with our risk-aware model. However, instead of having to explicitly model these uncertainties, we simple compute a mean value best-guess at the traversability cost, and use learning to aggregate the variation of these costs.

5.4 Implementation Details

In this section, we discuss details of the dataset, data processing, network architecture, and training procedures.

5.4.1 Dataset

Our motivation for tackling this problem comes from extensive field testing in various underground and decaying urban environments. These environments pose significant traversability challenges for wheeled, legged, and tracked robots [111]. We collected data from autonomous exploration runs in six different environments of various types (Figure 5.4, Table 5.1). These environments are: an abandoned subway station in Los Angeles, CA, consisting of two floors - a large open atrium with pillars (Subway Atrium / SA) and a basement floor with offices and narrow corridors (Subway Office / SO); Kentucky Underground Storage in Lexington, KY, which consists of very large cavernous grids of tunnels (Limestone Mine / LM); Wells Cave in Pulaski County, KY, which is a natural limestone cave with very rough floors, narrow passages, low ceilings, and rubble (Limestone Cave / LC); Valentine Cave at Lava Beds National Park, CA, which is an naturally formed ancient lava tube, with sloping walls and rough, rocky floors (Lava Tube A / TA); and Mammoth Cave at Lava Beds National Park, CA, which is also a lava tube but with sandy, sloping floors and occasional large piles of rubble (Lava Tube B / TB). A new data sample was added to the dataset at approximately 0.5Hz, while the average top speed of the robot is 1m/s.

Data was collected using Boston Dynamics's Spot legged robot equipped with JPL's NeBula payload [14]. The payload includes onboard computing, one VLP-16 Velodyne LiDAR sensor, and a range of other cameras and sensors. Spot is equipped with 5 Intel RealSense depth cameras which

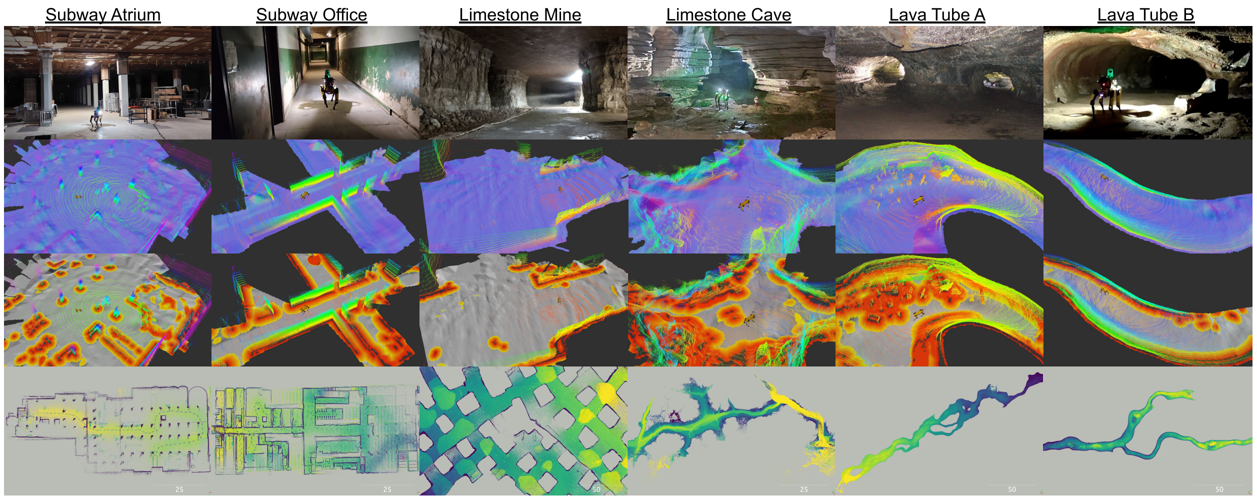


Figure 5.4: Datasets collected in 6 different environments. Top row: Photo of the environment. Second row: LiDAR pointcloud and elevation map produced after ground segmentation. Third row: Handcrafted risk map with varying risk (white: safe ($r <= 0.1$), yellow to red: moderate ($0.1 < r <= 0.9$), black: risky ($r > 0.9$)). Pointclouds are also shown. Bottom row: Map of the entire environment, generated by aggregating LiDAR pointclouds during each data collection run. Scale (in meters) is shown in the lower right corner.

are pointed at the ground.

5.4.2 Computing Traversability Cost

Traversability costs are computed online and saved along with dataset. These costs are generated via a risk-aware traversability pipeline [26], which we will briefly summarize here. First, LiDAR point clouds and odometry are fused to aggregate a more dense LiDAR pointcloud. Next, we perform ground segmentation to isolate ”ground” points, ”obstacle” points, and ”ceiling” points. Ground points are those which lie near the ground, while obstacle points are those which lie above the ground which will restrict the robot’s movement. We then perform elevation mapping, which probabilistically builds a 2.5D elevation map using the ”ground” points. Localization noise and delay are taken into account here, resulting in elevation variance information as well. We then compute costs as a function of various geometric information arising the the elevation map and the point cloud, such as slope, step size, roughness, negative obstacles, and collision obstacles. These costs are then aggregated into a combined mean cost (as well as a cost variance). The traversability cost is scaled between 0 and 1, with 1 being lethal and 0 being safe. The mean cost and variance of

Table 5.1: Details of datasets, with number of data samples, duration of the runs, approximate distance traveled, and average width of the passages in the environment. Abbreviation key: SA - Subway Atrium, SO - Subway Office, LM - Limestone Mine, LC - Limestone Cave, TA - Lava Tube A, TB - Lava Tube B.

	SA	SO	LM	LC	TA	TB
# Samples	1585	2883	942	331	852	1148
Duration (min)	53	96	31	11	28	38
Distance (m)	1000	1000	600	300	600	800
Min Width (m)	5	1	10	0.5	1	3
Max Width (m)	50	10	20	5	10	10

the cost are then used to compute CVaR, with assumption of a gaussian distribution. In our dataset we only use the mean cost which represents R. We aim to predict CVaR from the distribution of costs that arise through this analysis and the environment itself.

5.4.3 Transforming Pointclouds to Costmaps

We aim to construct a network whose inputs are the LiDAR pointcloud, and whose output is a CVaR costmap (2D). There are many different approaches for processing LiDAR data using neural networks [155, 156]. For point-cloud-to-image translation, however, there are fewer more recent approaches [157]. Engineering a pointcloud-to-image neural network architecture is beyond the scope of this work. For simplicity, we restrict ourselves to using an image-to-image autoencoder type of network (namely, U-Net with partial convolutions [158]), leaving the difficult task of transforming a LiDAR pointcloud to images for future work. To adapt our pipeline for this approach, we leverage the same point cloud processing and elevation mapping as mentioned in the above traversability cost analysis. We aggregate pointclouds in time, perform ground segmentation, and generate a 2.5D elevation map on the ground points. We then bin the pointcloud into 5 z-height bins relative to the elevation map. We count the number of points which fall into these bins and generate an image from these counts. We end up with the following input features, each feature channel is a 2D image: the elevation map (rescaled), the total number of LiDAR points in the aggregated pointcloud to be found in a location, the obstacle points (with more recent points having higher intensity), 5 z-height LiDAR point histogram bins, and a channel which encodes the distance

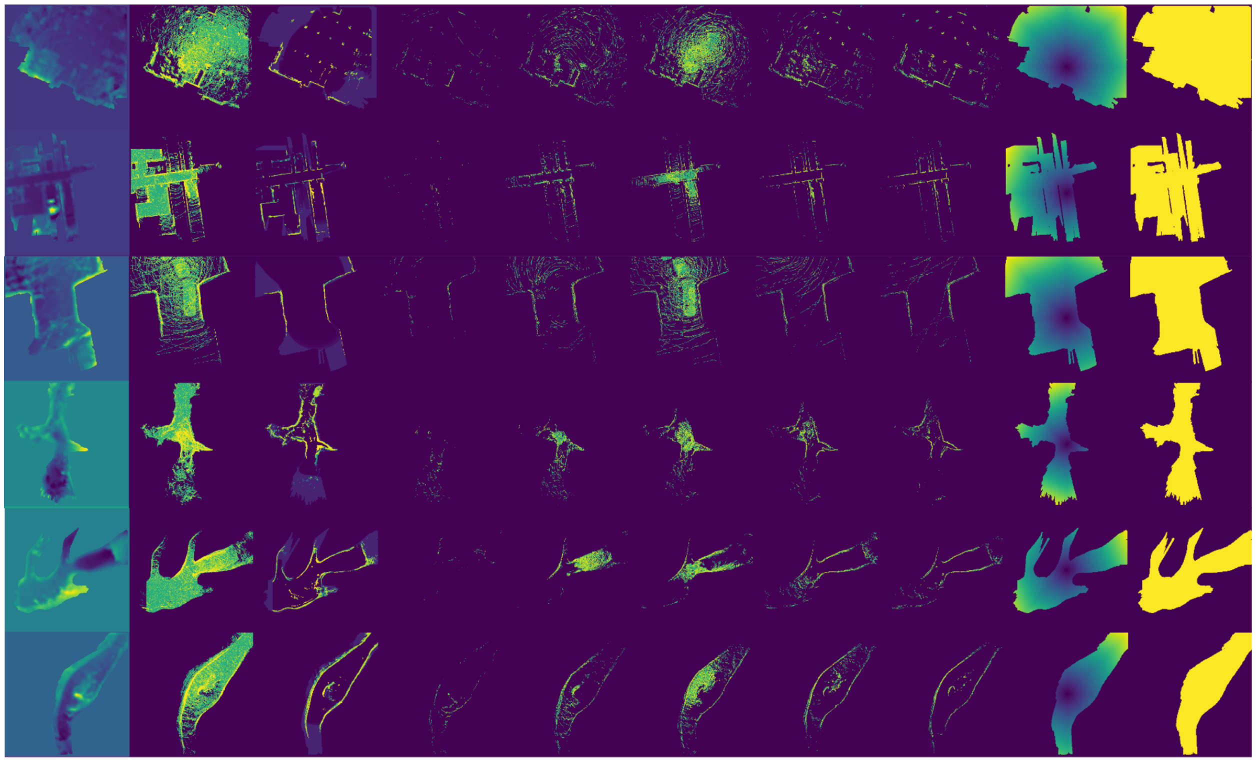


Figure 5.5: Input features converted from LiDAR pointclouds, showing different features in columns from left to right, while each row corresponds to one sample from each of the 6 datasets. Features are, from left to right: 1) elevation, 2) number of LiDAR points, 3) obstacle points (older points have a lower intensity), 4-8) number of points in each of 5 z-height bins, relative to the elevation map, with a bin height of 0.1m, 9) distance from the robot location, 10) ”known” region mask, which marks regions which have sensor coverage.

from the robot position in the map (Figure 5.5). Image sizes are 400x400 pixels, with each pixel representing 0.1m, for a total area of 40m x 40m covered.

The presence of gaps and occlusions in LiDAR data lead to known and unknown regions in the image-translated input data. These unknown regions need to be handled by the network. A naive approach would be to use zero-padding, i.e. replacing unknown pixels with 0. However, this can lead to blurring and other spurious artifacts at the edges. We take inspiration from recent work on neural network architectures use for in-painting, which naturally lend themselves to this image-to-image learning task with unstructured, unknown masks [158]. We include a mask of the known regions as input to the network.

Additionally, we provide the desired α level as an image to the input of the network. Since

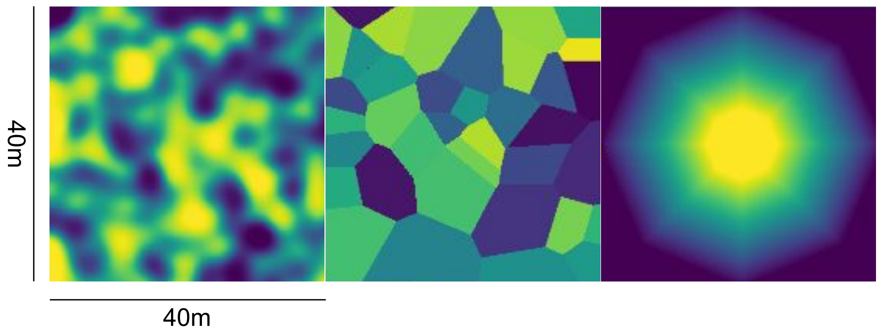


Figure 5.6: Some examples of α input channel images. Left: Randomly generated Gaussian filter, renormalized to have a uniform distribution. Middle: Randomly generated Voronoi-regions, again with uniformly distributed α values. Right: A radially decaying α input, which is an example of a custom desired variation of risk level for the resulting CVaR costmap. This type of variation may be useful for trying to ensure a greater degree of caution for obstacles near the robot. In this example, $\alpha = 1$ within 5 meters of the robot, and decays linearly down to 0 at 20m from the robot.

α is an image it can vary spatially. The loss function takes this desired α into account per-pixel, and trains the network to produce the correct risk level according to the desired input pattern. During training, we use a uniformly distributed smoothed random pattern, whose distribution spans $\alpha \in [0, 1]$.

Figure 5.7 outlines the costmap network architecture used in our approach. The pipeline consists of three components: 1) Pointcloud-to-image features transform, 2) an image-to-costmap translation network using a PartialConv U-Net, and 3) VaR ($V_\theta(m, x, \alpha)$) and CVaR ($\hat{C}_\theta(m, x, \alpha)$) outputs, which are fed into the loss function (Equation 5.14). The loss is averaged pixel-wise across the known mask regions only. This implies that each α value is independent from its neighbors. In practice, during both training and inference time, we expect to use only a certain distribution of patterns of α. A smoothly varying α pattern may emphasize correlations between neighboring pixels, while a completely random α image might encourage less correlations (See Figure 5.6).

See Table 5.2 for network architecture details. The network has approximately 1.1 million parameters. At the last layer, we clamp the output between 0 and 1 with leaky ReLU activation functions, since we expect both V_θ and $\hat{C}_\theta$ to remain within this range. This is specific to the distribution of the traversability costs used in this work and are not required in general.

Output of the network

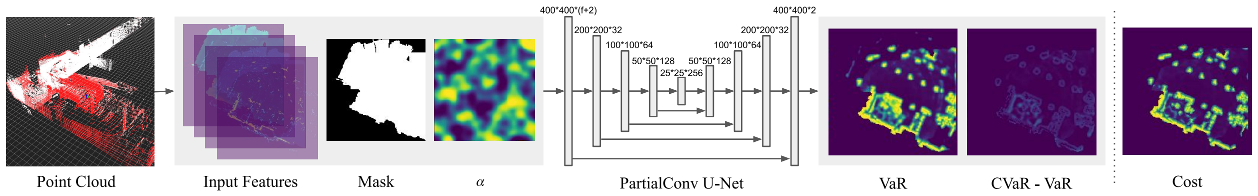


Figure 5.7: Our pointcloud-to-costmap pipeline. From left to right: Raw point clouds are aggregated and used to create 2d image-like input features and the mask. A 2d α channel also provides input to the network. The PartialConv U-Net architecture maps these input features to 2 output channels, namely VaR and $\mathrm{CVaR} - \mathrm{VaR}$. These two outputs are combined with the handcrafted cost labels to compute the loss.

Figure 5.8 shows corresponding outputs for the inputs shown in Figure 5.5. The leftmost column shows the ground truth traversability cost, while the next 3 columns show CVaR at varying α levels ($\alpha = 0.1, 0.5, 0.9$). Note that CVaR steadily increases as α increases, and generally is a greater value than the traversability cost. The next three columns show CVaR at the same varying alpha levels, minus the VaR output of the network when $\alpha = 0.1$. This provides more insight into the changes in VaR and CVaR as α is increased. The final column shows the same $\mathrm{CVaR} - \mathrm{VaR}_{0.1}$ except with a varying α input, in a radially decaying pattern, with high $\alpha = 1$ in the center, and low $\alpha = 0$ at the edges (See Figure 3.9c). This kind of pattern could be useful for a risk-aware system which wishes to be more conservative when planning motions in the regions near the robot. Such a behavior could be useful when the user believes he can rely more on data closer to the robot with greater sensor coverage, as well as wishing to allow for a relaxed commitment to avoiding traversability risks further away.

5.4.4 Training

Next we describe some details of training and hyperparameters. We used weights for the CVaR loss (5.14) of $\lambda_V = 10.0,\ \lambda_C = 1.0\ \lambda_m = 1.0e-4$. We used a Huber smoothing coefficient of $h = 1.0e-3$. The network was trained on a $90:5:5$ split of training, validation, and test data. We used Adam optimization with an initial learning rate of 0.0005, and batch size of 1. No pre-training of the network was used, weights were initialized with Xavier random weights. During training, data augmentation was used - including rotation, translation, scaling, shearing, and flipping. Figure

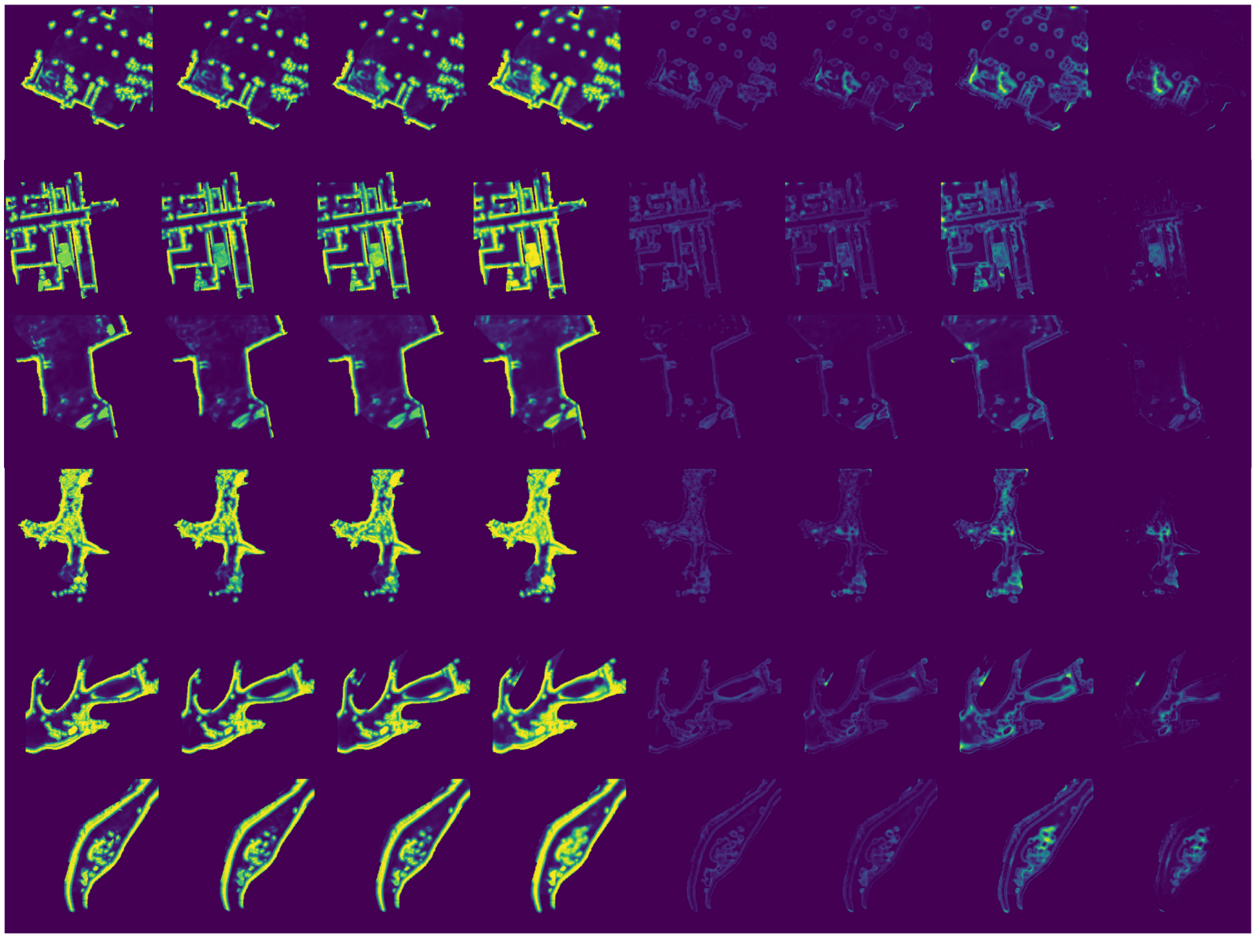


Figure 5.8: Network outputs on the same input examples shown in Figure 5.5. Columns from left to right: 1) Handcrafted cost label, 2-4) CVaR values with $\alpha = 0.1, 0.5, 0.9$ respectively, 5-7) $\mathrm{CVaR}_\alpha - \mathrm{VaR}_{0.1}$ for varying $\alpha = 0.1, 0.5, 0.9$ respectively. This enables us to more clearly see the differences between values of α. It also shows the difference between CVaR and VaR. 8) $\mathrm{CVaR}_\alpha - \mathrm{VaR}_{0.1}$ when α is a radially decaying output (See Figure 3.9c). Notice that the risk also decays radially from the center of the map.

Table 5.2: Neural network architecture details. PConv are partial convolutions, with PConv1-4 encoding and PConv5-8 decoding. BN indicates if the layer is followed by Batch Normalization. Concat$*$ indicates skip links, which concatenate the output of previous layers.

Layer Name	Filter Size	# Filters/ Channels	Stride/ UpFactor	BN	Activation
PConv1	7×7	32	2	-	ReLU
PConv2	5×5	64	2	Y	ReLU
PConv3	5×5	128	2	Y	ReLU
PConv4	3×3	256	2	Y	ReLU
UpSample1		256	2	-	-
Concat1(w/ PConv3)		256+128		-	-
PConv5	3×3	128	1	Y	LeakyReLU(0.2)
UpSample2		128	2	-	-
Concat2(w/ PConv2)		128+64		-	-
PConv6	3×3	64	1	Y	LeakyReLU(0.2)
UpSample3		64	2	-	-
Concat3(w/ PConv1)		64+32		-	-
PConv7	3×3	32	1	Y	LeakyReLU(0.2)
UpSample4		32	2	-	-
Concat4(w/ Input)		32+2		-	-
PConv8	3×3	2	1	-	-

5.9 shows loss curves over the course of training on the dataset. Note that the VaR loss steadily decreases while CVaR loss is slower to converge. This is expected since the CVaR loss is a moving target which depends on VaR. Also, we see that the monotonic loss stays relatively small and does not increase. Training took about 16 hours on a 12GB NVIDIA Tesla K80 GPU.

5.5 Evaluation and Results

In this section, we evaluate the method on data which is both in-distribution and out-of-distribution, compare against different baselines, and investigate the behavior of the network. We first present some qualitative observations about the performance of the model. Figure 5.10 shows an interesting case where the traversability cost of an obstacle is difficult to determine. In this case (taken from the Limestone Cave dataset), a row of cinderblocks (0.15-0.2m in height) blocks Spot's path. Spot is usually capable of traversing such obstacles but they do pose some risk. (In this particular instance, the robot tripped over and fell down.) We observe the network has learned the concept of the variation of this risk as α is increased from 0.1 to 0.9. While the handcrafted traversability

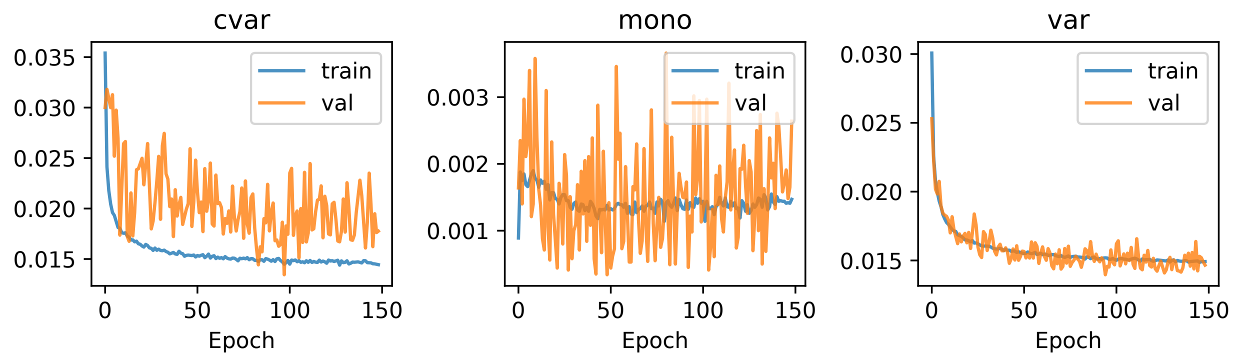


Figure 5.9: Learning curves for one training session, showing the three losses (CVaR loss, monotonic loss, and VaR loss).

cost shows a risky region on the wall for this sample, other samples or other similar situations have shown less traversability cost, so the network has learned to smoothly interpolate along this distribution. Note that this change is more than simple scaling up all costs in the output, as can be seen from the plots of $\mathrm{CVaR}_\alpha - \mathrm{VaR}_{0.1}$. As α increases, we see a much stronger increase in CVaR at the location of the low wall relative to the rest of the costmap.

We also observe the network learns to remove spurious noise and artifacts from the costmap (Figure 5.11). In this example (from Lava Tube A), several artifacts are visible in the handcrafted costmap, arising from localization noise and sparse sensor coverage. The worst offending artifact entirely blocks a passageway - this is caused by localization noise which creates artifacts in the elevation map. The network produces a clean risk map which ignores these artifacts.

5.5.1 Evaluation Metrics

We introduce three evaluation metrics for assessing the quality of VaR and CVaR learning for a given value of α. First, we compute the implied-α (I_α) value, which is a measure of how closely the predicted VaR value matches the true quantile:

$$I_\alpha = \frac{\sum_{\mathcal{D}} \mathbb{I}[R <= V_\theta(\alpha)]}{\sum_{\mathcal{D}} \mathbb{I}[mask]} \tag{5.16}$$

An accurate model should have a value of $I_\alpha \approx \alpha$. Note that we sum over all valid (non-masked) pixels and data samples. The denominator counts the number of non-masked pixels.

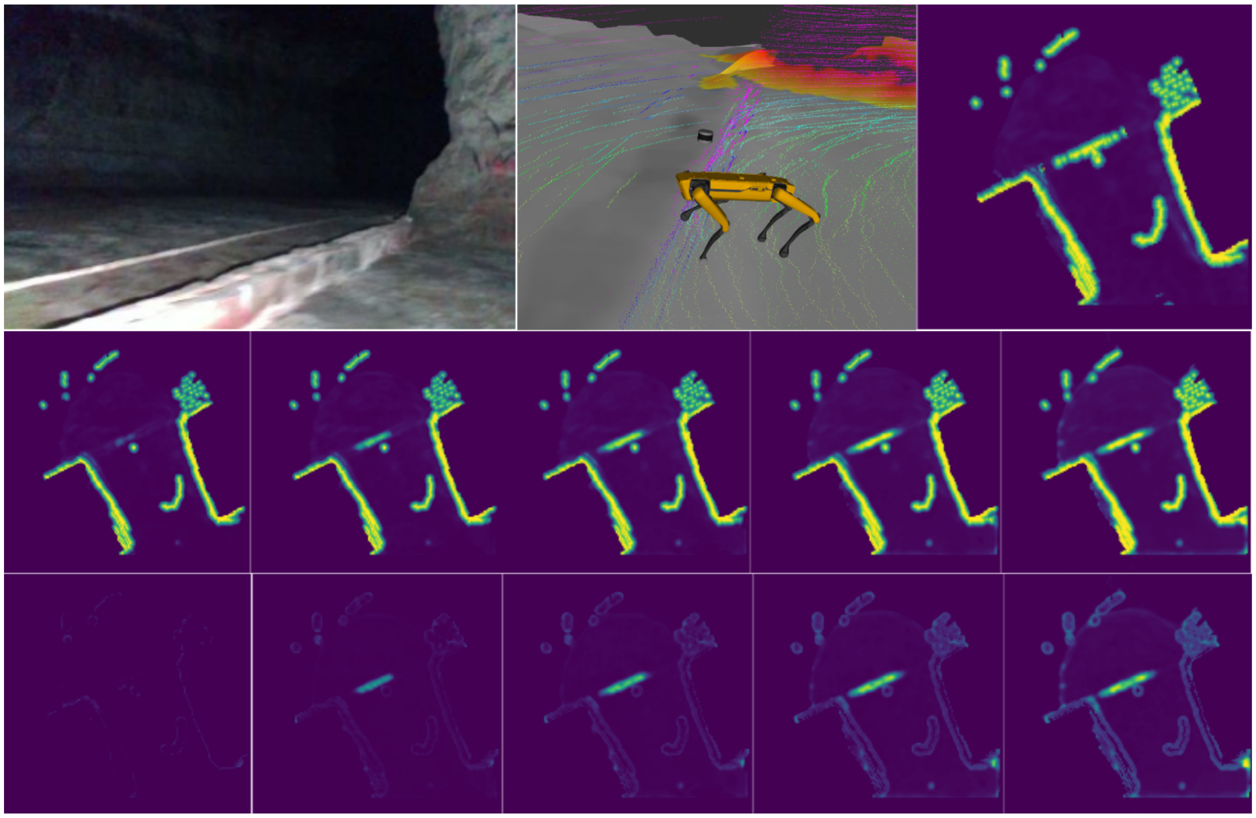


Figure 5.10: Case study for a low wall obstacle. Top left: On-board camera image of the obstacle. Top right: Point cloud and elevation map. Spot has placed one foot on top of the wall. Top right: Handcrafted traversability cost. The wall appears in the center of the map in a line. Middle row: CVaR at varying levels of $\alpha = 0.1, 0.3, 0.5, 0.7, 0.9$ respectively. Bottom row: $\mathrm{CVaR}_\alpha - \mathrm{VaR}_{0.1}$ at varying levels of alpha. The risk of the wall increases greatly as α is increased, while other regions increase in risk more gradually. Color scale of all maps range from 0 to 1.

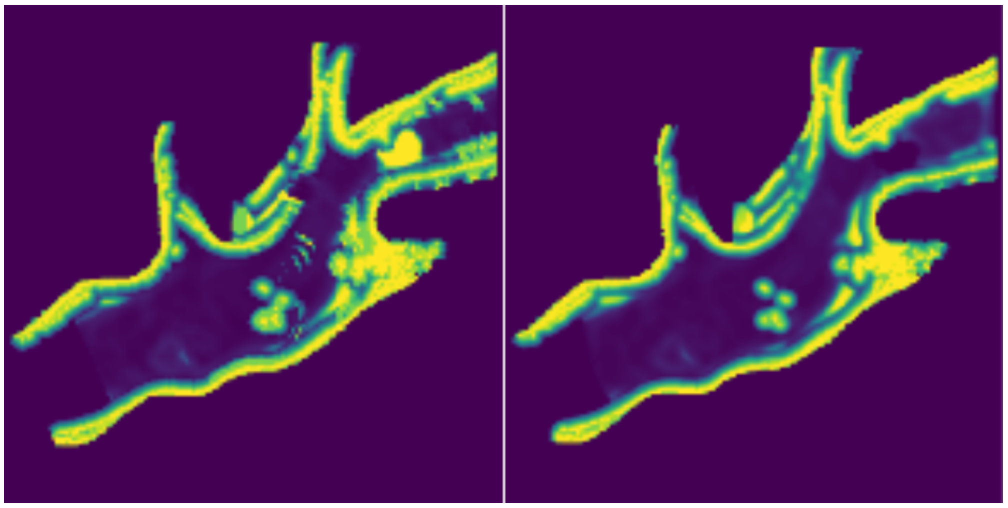


Figure 5.11: Case study of localization artifact removal. Left: Handcrafted traversability cost. Right: CVaR network output when $\alpha = 0.5$. We observe that artifacts arising from localization error and sensor sparsity are not present in the predicted risk map. In this particular case, this has significant impact on path planning through the narrow passage.

The R^2 *coefficient of determination* metric is useful for comparing the explanatory power of a model relative to the distribution of the data. To quantify the modeling capacity of the network when compared to another other baselines, we use a pseudo-R^2, which is a function of α. For assessing pseudo-R^2 for VaR, we use the following metric [159]:

$$R^2_V(\alpha) = 1 - \frac{\sum_{\mathcal{D}} \left[\alpha |R - V_\theta(\alpha)|_+ + (1-\alpha)|R - V_\theta(\alpha)|_-\right]}{\sum_{\mathcal{D}} \left[\alpha |R - \bar{V}_\alpha|_+ + (1-\alpha)|R - \bar{V}_\alpha|_-\right]} \tag{5.17}$$

where $\bar{V}_\alpha$ is the constant α-quantile computed from the training data. This metric compares the Koenker-Bassett error of a quantile regression model against the total absolute deviation of the data from a independent α quantile. A value of 1 is the best, reflecting total explanation of the dependent variable, while a value less than 0 is poor, implying worse explanatory power than the simple quantile.

Finally, to assess the CVaR modeling, we construct a similar pseudo-R^2 metric for CVaR. Instead of the Koenker-Bassett error, we use the empirical CVaR error (see [160] for a rigorous discussion of a similar but computationally less tractable approach). Our CVaR pseudo-R^2 metric compares the total absolute error between our CVaR model and an average empirical CVaR computed form the training data:

$$R_C^2(\alpha) = 1 - \frac{\sum_{\mathcal{D}} \left| C_\theta(\alpha) - \left(V_\theta(\alpha) + \frac{1}{1-\alpha}(R - V_\theta(\alpha))_+ \right) \right|}{\sum_{\mathcal{D}} \left| \bar{C}_\alpha - \left(\bar{V}_\alpha + \frac{1}{1-\alpha}(R - \bar{V}_\alpha)_+ \right) \right|} \tag{5.18}$$

where $\bar{C}_\alpha$ is the constant α-CVaR computed from the training data.

5.5.2 In-distribution (ID) Performance

We begin by evaluating the in-distribution (ID) performance of the network. Figure 5.12 shows a boxplot of the three evaluation metrics described above, testing on the held-out test data, for a network trained on all 6 datasets. The box plots show the distribution of metrics computed per-sample. Implied α holds up well, forming tracking α, while VaR R^2 and CVaR R^2 are close to 1 for most values of α. As α nears 0.9 and 0.95, the CVaR R^2 drops closer to 0, implying less of the data is being accurately explained by the model. This is likely due to the distribution of traversability costs lying between 0 and 1. When α is high, both VaR and CVaR are more frequently predicted to be 1. This results in a lower CVaR R^2 since the denominator will be small.

Next, we plot evaluation metrics on each of the 6 dataset categories individually (Figure 5.13). This allows us to evaluate the differences between each dataset. We note that the Limstone Cave dataset has the highest levels of risk, which results in more frequent distortion the CVaR R^2 metric for high α levels. Nevertheless, all datasets perform well, with pseudo-R^2 values close to 1.0.

5.5.3 Out-of-Distribution (OOD) Performance

To evaluate out-of-distribution (OOD) performance, we train the network on all data except the one dataset, and then evaluate the network on the OOD dataset (which the network has not seen during training). We do this by holding out the Subway Atrium dataset (Figure 5.14) as well as the Limestone Cave dataset (Figure 5.15). These two datasets show interesting differences, as the Subway Atrium dataset tends to have clear pillars and walls, large rooms, and moderate amounts of industrial clutter with clear, sharp edges. On the other hand the Limestone Cave dataset has the most different and challenging terrain with the highest average risks, and with many features not present

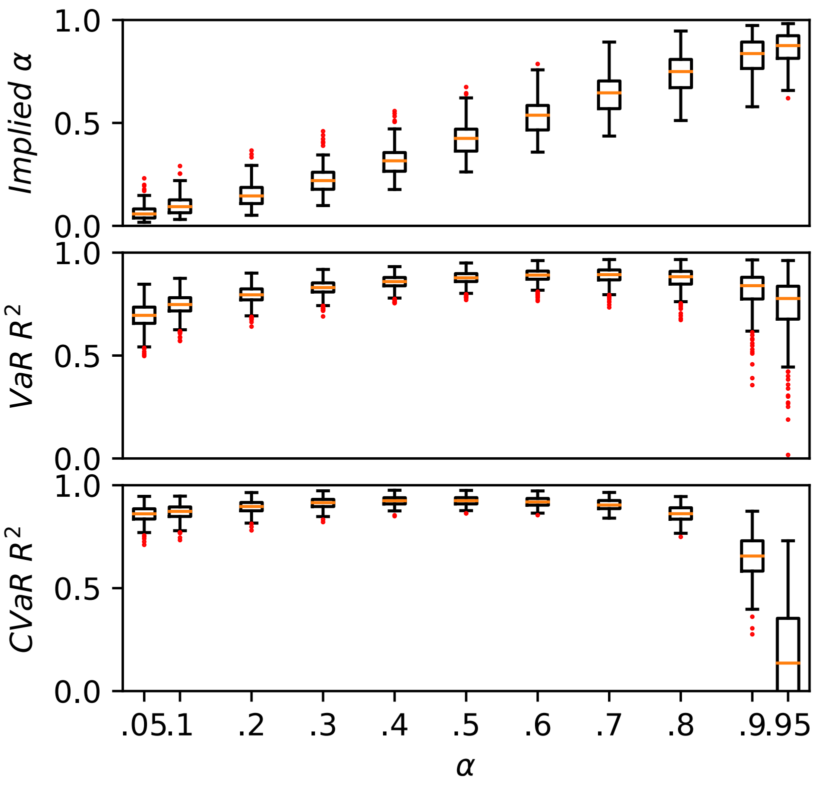


Figure 5.12: ID performance: Boxplot of evaluation metrics for model trained on all 6 datasets, evaluated on held-out test data from all 6 datasets. The evaluation metrics were computed on each sample individually, and are aggregated in ths box plot shown here. Red dots mark outliers.

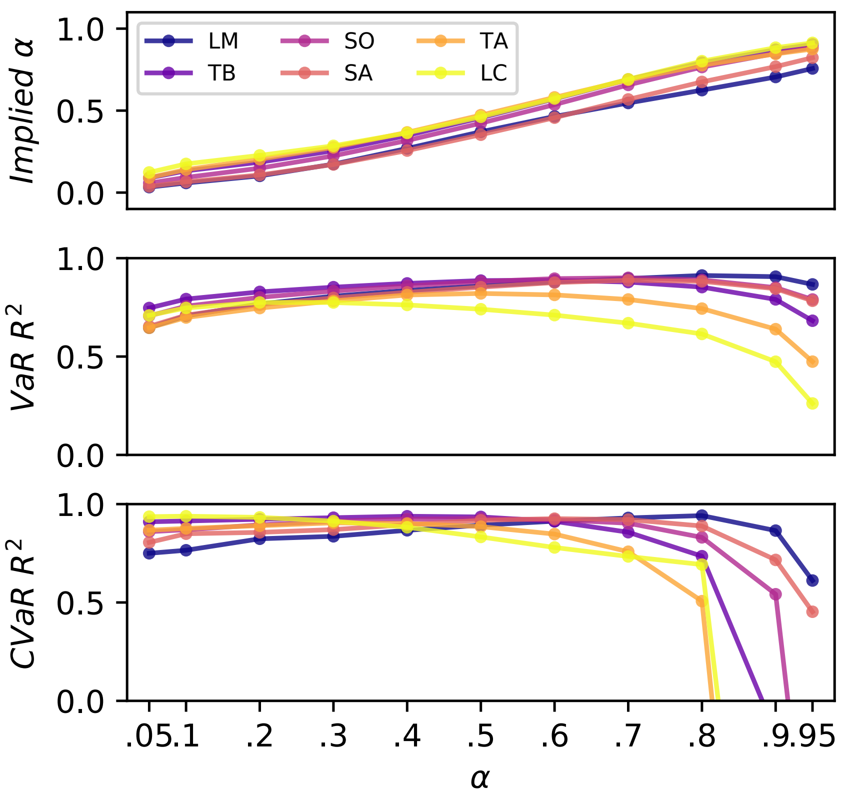


Figure 5.13: ID performance: Evaluation metrics for model trained on all 6 datasets, evaluated on held-out test data from 6 datasets individually. Some datasets have better performance than others, with the Limestone Cave having the worst performance, it being the most challenging.

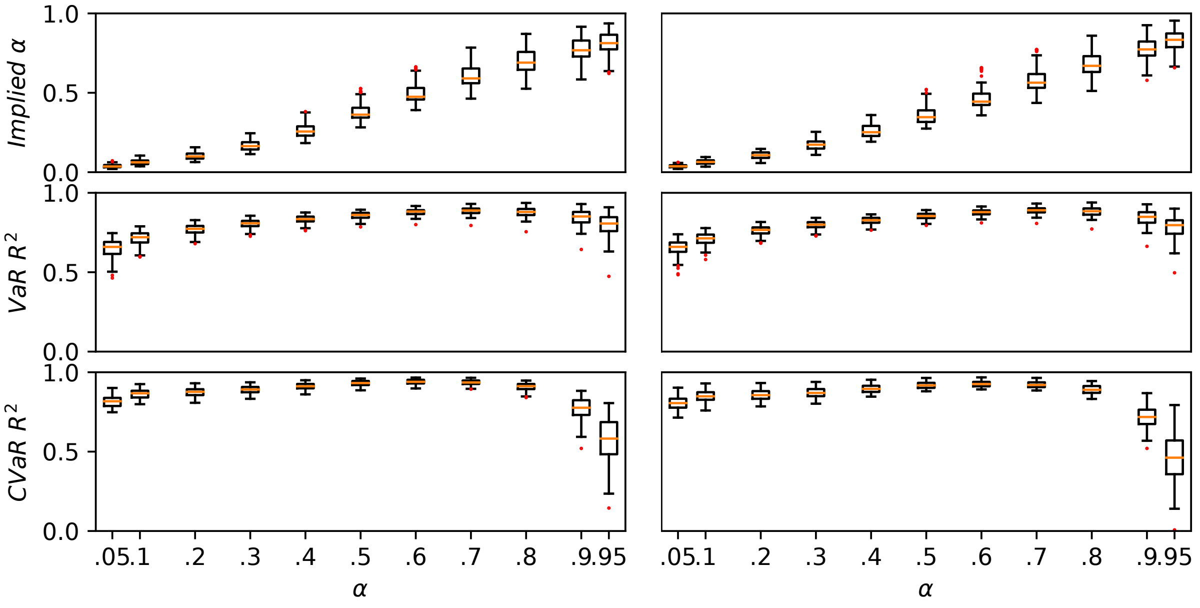


Figure 5.14: OOD performance: (Left) Boxplot of evaluation metrics for model trained on all datasets except the Subway Atrium, evaluated on held-out data from the Subway Atrium. Performance is similar to the ID performance, i.e. when the model is trained on all data, and evaluated on the Subway Atrium data (Right).

in the other datasets, such as sharp jagged rocks, narrow passages, channels of water, and rough floors. We see that while OOD performance is similar to ID performance for the Subway Atrium dataset, the Limestone Cave dataset fairs less well, where the OOD performance is noticeably worse. This shows that indeed the Limestone Cave dataset contains unique features different from the other datasets. However, even with this difference, the Limestone Cave OOD model is still able to maintain healthy pseudo-R^2 statistics (> 0) for a large range of α.

We also examine whether the network can overfit to one dataset, by training it exclusively on the Subway Atrium dataset. We then test and compute evaluation metrics on all dataset and show a comparison in Figure 5.16. All models perform well, with performance slightly worse than when the model was trained on all the data (Figures 5.12 and 5.13). However, when testing on the ID Subway Atrium dataset, the performance is very slightly better than the model trained on all the data. This may imply a small degree of overfitting.

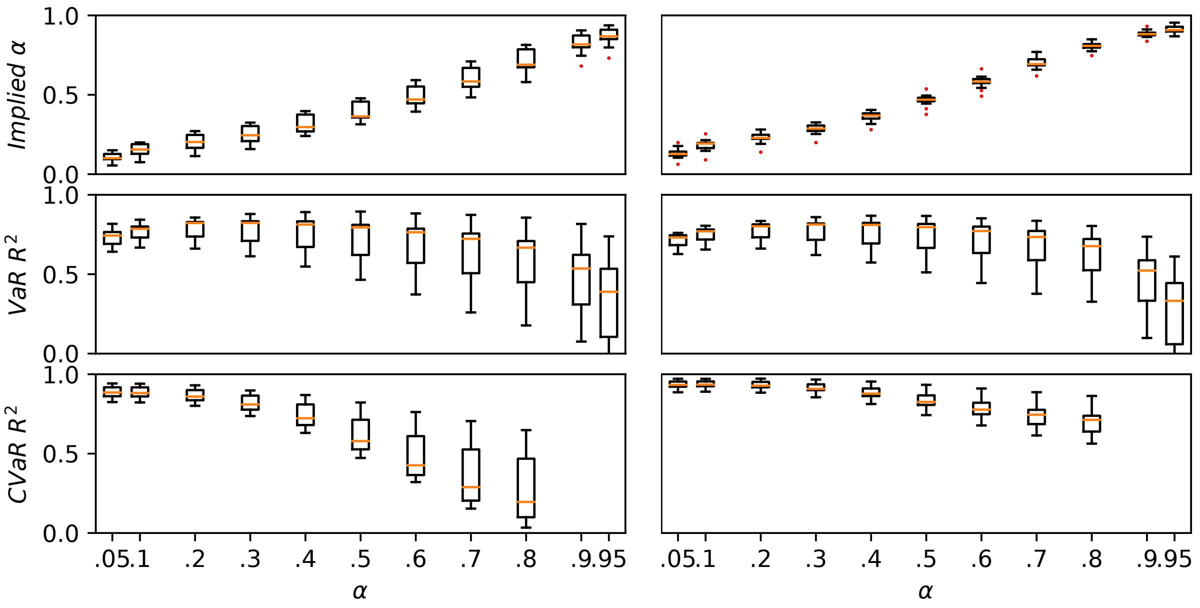


Figure 5.15: OOD performance: (Left) Boxplot of evaluation metrics for model trained on all datasets except the Limestone Cave, evaluated on held-out data from the Limestone Cave. Performance is worse than the ID performance, i.e. when the model is trained on all data, and evaluated on the Limestone Cave data (Right).

5.5.4 Comparisons against baselines

Finally, we compare our approach against three baselines: 1) the handcrafted traversability risk model which relies on assumptions of Gaussianity and accurate mean and variance, 2) a model trained to predict the traversability cost using an L1 loss function only, and 3) a model trained to predict both traversability mean and variance using a Gaussian negative log likelihood (NLL) loss [11], which is then used to compute VaR and CVaR assuming a Gaussian distribution. We computed metrics by taking the average over all test datasets, shown in Figure 5.17. The L1 Loss model does not capture any quantile information and does not depend on α. Therefore, it does not capture the VaR or CVaR as well as the trained model. The hand-crafted traversability cost model also does not perform well - it fails to accurately capture the quantile (VaR) for most values of α. It also does not provide as accurate CVaR values as the learned model. The NLL loss model fairs relatively well predicting CVaR, but less-so for VaR. However, the learned CVaR model still outperforms it in the CVaR statistic.

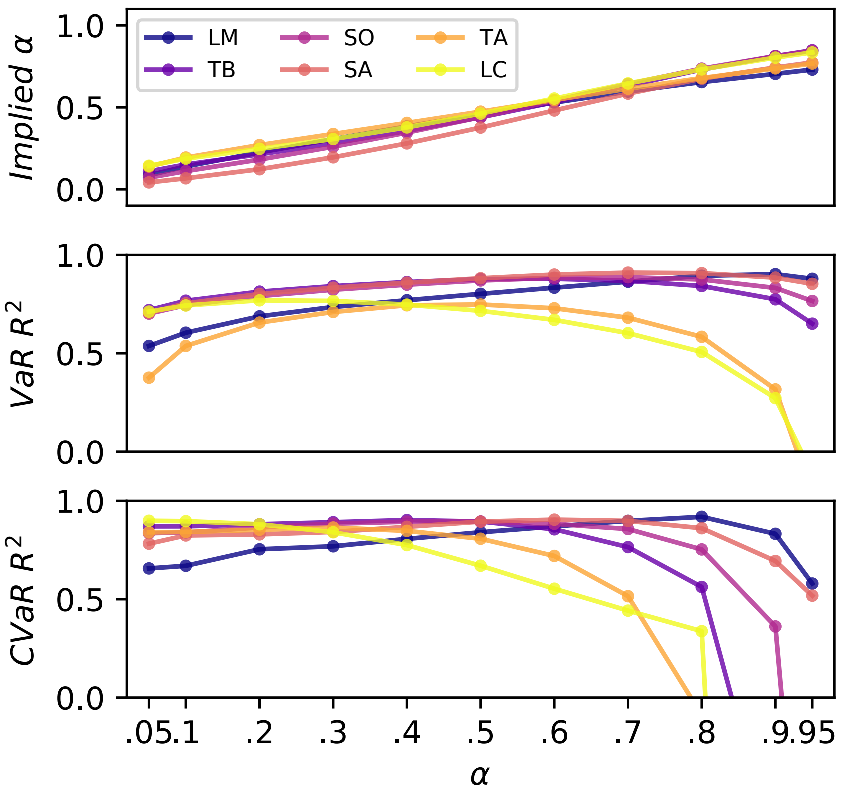


Figure 5.16: OOD performance: Evaluation metrics for model trained on Subway Atrium data only, testing on all datasets. Despite not seeing any of the other environments during training, the model is able to generalize quite well.

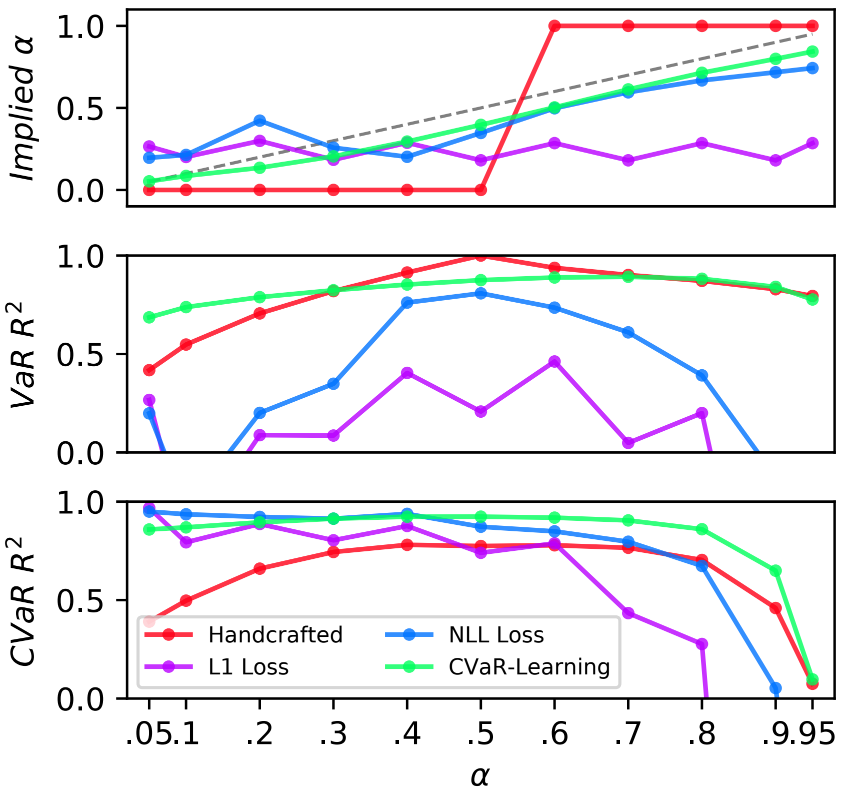


Figure 5.17: Comparing CVaR-learning model against handcrafted CVaR cost model, L1-loss model, and NLL-loss model. The learned model was trained on all 6 datasets. Shown here are evaluation metrics computed over the held-out test data for all 6 datasets. The CVaR-learning method outperforms by a clear margin. Dotted line in first plot indicates ideal implied α values.

We also consider the computation time of using the handcrafted traversability cost method vs. our CVaR learning method (Table 5.3). The CVaR learning approach tends to be almost 5x more efficient then the handcrafted approach. (Both methods currently rely on ground segmentation and elevation mapping steps.) We see that replacing the handcrafted costmap with the learned one removes the largest bottleneck in the traversability assessment pipeline.

Table 5.3: Computation time for Handcrafted costmap vs. CVaR-learning costmap, computed from N=250 samples. Also shown are times for preprocessing steps, i.e. ground segmentation and creating the 2.5 elevation map. Using a CVaR learning costmap to replace the handcrafted costmap would improve total computation time by approximately 48%.

Computation Time	μ (ms)	σ (ms)
Ground Segmentation	13.97	3.88
Elevation Mapping	140.19	65.48
Handcrafted Costmap	241.92	30.11
CVaR-Learning Costmap	**48.94**	**15.36**

5.6 Conclusion

In this work, we introduced a novel neural network architecture for transforming point cloud data into risk-aware costmaps. We also introduced a novel formulation of CVaR and VaR loss functions which train the network to accurately capture tail risk, without relying on cumbersome modeling or distribution assumptions. We demonstrated this approach on a robotic navigation task in unstructured and challenging terrain. We evaluated the method using metrics which show that this approach reliably learns the expected tail risk given a desired probability risk threshold between 0 and 1 and produces a traversability costmap which is robust to outliers, accurately captures tail risks, generalizes well to out-of-distribution data, and is more computationally efficient. Future work includes investigation of architectures which directly map LiDAR points to costs, investigating the use of importance sampling to improve risk estimates for high values of α [161], and investigation of learning dynamic risk metrics [23].

CHAPTER 6
CONCLUSION

The aim of this thesis is to explore methods for both establishing safety guarantees as well as accurately quantifying risks when using deep neural networks for robot planning and control, especially in high-risk environments. This has been demonstrated by novel contributions at multiple layers of the robotics stack, from controls, trajectory optimization, planning, and traversability. At each layer we have introduced a novel approach for uncertainty quantification using deep neural networks. Each of our contributions does not rely on a specific deep learning method; rather, we seek to show the feasibility and advantages of using black-box function approximators for safety-critical applications.

At the controls layer, we have presented an approach to guaranteeing safety while enabling adaption and learning using Bayesian neural networks. As methods for Bayesian neural network learning improve, our method will benefit. At the trajectory optimization layer, we have presented a means for tackling the difficult problem of accurately quantifying the propagation of uncertainty over many timesteps, by using deep quantile regression. Again, as methods for quantile regression in deep learning improve, our method will likewise benefit in terms of scalability, efficiency, and accuracy. At the level of planning, this thesis has presented a unified framework for risk-aware traversability and kinodynamically feasible planning by quantifying and optimizing over CVaR tail-risks in a computationally efficient manner. We then leverage this framework to enable directly learning CVaR costmaps from raw point cloud data, improving the baseline approach in accuracy, robustness, and computational efficiency. Again, this approach benefits from further advances in deep quantile regression as well as deep point-cloud to image transformation architectures.

One point that has not yet been addressed in detail is the question of why deep learning is such a useful tool, and why one should go to the effort of bringing a deep learning method into a domain which has been previously dominated by traditional model-based methods which potentially offer stronger guarantees in safety-critical applications. One answer might be that before deep learning, the intelligence of machines was bounded by human intelligence - i.e., how clever of an algorithm can a group of humans design. Now, deep learning has broken this bound, and the intelligence of machines is now bounded by the amount of data and the size of the network that is feasible to manage. In many nascent fields seeking to supplant traditional methods by neural networks, the traditional methods may at first outperform neural network prototypes and graduate student researchers with limited data and budgets. However, Moore's law implies that this is sure to change.

It is the author's hope that this thesis adds some strong datapoints towards more widespread investigation and use of deep learning, specifically for controls and planning in safety-critical systems, in order to achieve greater autonomy and intelligence for robots working on these tasks. By advancing these technologies, perhaps humans might further extend their consciousness to the stars, reduce the hazardous impacts of natural disasters or industrial accidents, and improve the lives of every individual.

REFERENCES

[1] L. Kunze, N. Hawes, T. Duckett, M. Hanheide, and T. Krajnik, “Artificial intelligence for long-term robot autonomy: A survey,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4023–4030, 2018.

[2] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, Z. D. Guo, and C. Blundell, “Agent57: Outperforming the atari human benchmark,” in *International Conference on Machine Learning*, PMLR, 2020, pp. 507–517.

[3] Q. Yu, Y. Yang, F. Liu, Y.-Z. Song, T. Xiang, and T. M. Hospedales, “Sketch-a-net: A deep neural network that beats humans,” *International journal of computer vision*, vol. 122, no. 3, pp. 411–425, 2017.

[4] Z. Kurd, T. Kelly, and J. Austin, “Developing artificial neural networks for safety critical systems,” *Neural Computing and Applications*, vol. 16, no. 1, pp. 11–19, 2007.

[5] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, *et al.*, “A review of uncertainty quantification in deep learning: Techniques, applications and challenges,” *Information Fusion*, 2021.

[6] Y. Gal and Z. Ghahramani, “Dropout as a bayesian approximation: Representing model uncertainty in deep learning,” in *international conference on machine learning*, 2016, pp. 1050–1059.

[7] D. Hafner, D. Tran, T. Lillicrap, A. Irpan, J. Davidson, T. Lillicrap, and J. Davidson, “Reliable uncertainty estimates in deep neural networks using noise contrastive priors,” *arXiv preprint arXiv:1807.09289*, 2018. arXiv: `1807.09289`.

[8] M. Segú, A. Loquercio, D. Scaramuzza, M. Segu, and D. Scaramuzza, “A general framework for uncertainty estimation in deep learning,” *arXiv preprint arXiv:1907.06890*, vol. 5, no. 2, pp. 3153–3160, 2019.

[9] N. Tagasovska and D. Lopez-Paz, “Frequentist uncertainty estimates for deep learning,” 2018. arXiv: `1811.00908`.

[10] I. Osband, “Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout,” in *NIPS workshop on bayesian deep learning*, vol. 192, 2016.

[11] A. Kendall and Y. Gal, “What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?,” 2017. arXiv: `1703.04977`.

[12] A. Liu, G. Shi, S.-J. Chung, A. Anandkumar, and Y. Yue, "Robust Regression for Safe Exploration in Control," *arXiv preprint arXiv:1906.05819*, 2019. arXiv: `1906.05819`.

[13] V. Kuleshov, N. Fenner, and S. Ermon, "Accurate uncertainties for deep learning using calibrated regression," in *International Conference on Machine Learning*, PMLR, 2018, pp. 2796–2804.

[14] A. Agha, K. Otsu, B. Morrell, D. D. Fan, R. Thakker, A. Santamaria-Navarro, S.-K. Kim, A. Bouman, X. Lei, J. Edlund, *et al.*, "Nebula: Quest for robotic autonomy in challenging environments; team costar at the darpa subterranean challenge," *arXiv preprint arXiv:2103.11470*, 2021.

[15] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*, IEEE, 2019, pp. 3420–3431.

[16] S. Singh, A. Majumdar, J.-J. Slotine, and M. Pavone, "Robust online motion planning via contraction theory and convex optimization," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 5883–5890.

[17] Y. Gal, R. T. Mcallister, and C. E. Rasmussen, "Improving PILCO with Bayesian Neural Network Dynamics Models," *Data-Efficient Machine Learning Workshop, ICML*, pp. 1–7, 2016. arXiv: `1706.08495`.

[18] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer Science & Business Media, 2013.

[19] J. Löfberg, "Oops! I cannot do it again: Testing for recursive feasibility in MPC," *Automatica*, vol. 48, no. 3, pp. 550–555, 2012.

[20] R. Koenker and K. F. Hallock, "Quantile regression," *Journal of economic perspectives*, vol. 15, no. 4, pp. 143–156, 2001.

[21] F. Riedel, "Dynamic coherent risk measures," *Stochastic processes and their applications*, vol. 112, no. 2, pp. 185–200, 2004.

[22] F. C. I. Commission *et al.*, *The financial crisis inquiry report: The final report of the National Commission on the causes of the financial and economic crisis in the United States including dissenting views*. Cosimo, Inc., 2011.

[23] A. Majumdar and M. Pavone, "How should a robot assess risk? Towards an axiomatic theory of risk in robotics," in *Robotics Research*, Springer, 2020, pp. 75–84. arXiv: `1710.11040`.

[24] D. D. Fan, J. Nguyen, R. Thakker, N. Alatur, A.-a. Agha-mohammadi, and E. A. Theodorou, "Bayesian learning-based adaptive control for safety critical systems," in *International Conference on Robotics and Automation*, 2020.

[25] D. D. Fan, A.-a. Agha-mohammadi, and E. A. Theodorou, "Deep learning tubes for tube MPC," *Robotics: Science and Systems (RSS)*, 2020.

[26] D. D. Fan, K. Otsu, Y. Kubo, A. Dixit, J. Burdick, and A.-A. Agha-Mohammadi, "STEP: Stochastic traversability evaluation and planning for safe off-road navigation," *Robotics: Science and Systems (RSS)*, 2021.

[27] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[28] NASA, *Where is Curiosity? - NASA Mars Curiosity Rover. https://mars.nasa.gov/msl/mission/whereistherovernow/*, 2018.

[29] ——, *Opportunity Updates. https://mars.nasa.gov/mer/mission/rover-status/opportunity/recent/all/*, 2018.

[30] E. Klein, E. Nilsen, A. Nicholas, C. Whetsel, J. Parrish, R. Mattingly, and L. May, "The Mobile MAV concept for Mars Sample Return," in *2014 IEEE Aerospace Conference*, 2014, pp. 1–9.

[31] A. Nelessen, C. Sackier, I. Clark, P. Brugarolas, G. Villar, A. Chen, A. Stehura, R. Otero, E. Stilley, D. Way, K. Edquist, S. Mohan, C. Giovingo, and M. Lefland, "Mars 2020 Entry, Descent, and Landing System Overview," in *2019 IEEE Aerospace Conference*, 2019, pp. 1–20.

[32] N. Wagener, C.-A. Cheng, J. Sacks, and B. Boots, "An Online Learning Approach to Model Predictive Control," *CoRR*, vol. abs/1902.0, 2019. arXiv: `1902.08967`.

[33] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Information-theoretic model predictive control: Theory and applications to autonomous driving," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1603–1622, 2018.

[34] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe Model-based Reinforcement Learning with Stability Guarantees," in *Advances in neural information processing systems*, 2017, pp. 908–918.

[35] S.-K. Kim, R. Thakker, and A.-A. Agha-Mohammadi, "Bi-Directional Value Learning for Risk-Aware Planning Under Uncertainty," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2493–2500, 2019.

[36] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 627–635.

[37] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Robust Constrained Learning-based NMPC enabling reliable mobile robot path tracking," *The International Journal of Robotics Research*, vol. 35, no. 13, pp. 1547–1563, 2016.

[38] K. Pereida and A. P. Schoellig, "Adaptive Model Predictive Control for High-Accuracy Trajectory Tracking in Changing Conditions," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 7831–7837, ISBN: 978-1-5386-8094-0.

[39] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious Model Predictive Control using Gaussian Process Regression," *arXiv*, 2017. arXiv: `1705.10702`.

[40] G. Shi, X. Shi, M. O'Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, "Neural Lander: Stable Drone Landing Control using Learned Dynamics," 2018. arXiv: `1811.08027`.

[41] G. Chowdhary, H. A. Kingravi, J. P. How, and P. A. Vela, "Bayesian Nonparametric Adaptive Control Using Gaussian Processes," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 3, pp. 537–550, 2015.

[42] Q. Nguyen and K. Sreenath, "Optimal Robust Control for Bipedal Robots through Control Lyapunov Function based Quadratic Programs.," *cmu.edu*, 2015.

[43] Q. Nguyen and K. Sreenath, "Optimal robust control for constrained nonlinear hybrid systems with application to bipedal locomotion," in *2016 American Control Conference (ACC)*, IEEE, 2016, pp. 4807–4813.

[44] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-End Safe Reinforcement Learning through Barrier Functions for Safety-Critical Continuous Control Tasks," *arXiv*, 2019. arXiv: `1903.08792`.

[45] Q. Nguyen and K. Sreenath, "L1 adaptive control for bipedal robots with control Lyapunov function based quadratic programs," in *2015 American Control Conference (ACC)*, IEEE, 2015, pp. 862–867, ISBN: 978-1-4799-8684-2.

[46] A. J. Taylor, V. D. Dorobantu, M. Krishnamoorthy, H. M. Le, Y. Yue, and A. D. Ames, "A Control Lyapunov Perspective on Episodic Learning via Projection to State Stability," *arXiv*, 2019. arXiv: `1903.07214`.

[47] T. Gurriet, A. Singletary, J. Reher, L. Ciarletta, E. Feron, and A. Ames, "Towards a framework for realizable safety critical control through active set invariance," in *Proceedings*

of the 9th ACM/IEEE International Conference on Cyber-Physical Systems, IEEE Press, 2018, pp. 98–106.

[48] V. Azimi and P. A. Vela, “Robust Adaptive Quadratic Programming and Safety Performance of Nonlinear Systems with Unstructured Uncertainties,” in *2018 IEEE Conference on Decision and Control (CDC)*, IEEE, 2018, pp. 5536–5543.

[49] ——, “Performance Reference Adaptive Control: A Joint Quadratic Programming and Adaptive Control Framework,” in *2018 Annual American Control Conference (ACC)*, IEEE, 2018, pp. 1827–1834.

[50] Q. Nguyen and K. Sreenath, “Exponential Control Barrier Functions for enforcing high relative-degree safety-critical constraints,” in *2016 American Control Conference (ACC)*, IEEE, 2016, pp. 322–328, ISBN: 978-1-4673-8682-1.

[51] A. D. Ames, K. Galloway, K. Sreenath, and J. W. Grizzle, “Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics,” *IEEE Transactions on Automatic Control*, vol. 59, no. 4, pp. 876–891, 2014.

[52] A. Look and M. Kandemir, “Differential bayesian neural nets,” *arXiv preprint arXiv:1912.00796*, 2019.

[53] X. Liu, S. Si, Q. Cao, S. Kumar, and C.-J. Hsieh, “Neural sde: Stabilizing neural ode networks with stochastic noise,” *arXiv preprint arXiv:1906.02355*, 2019.

[54] P. Hegde, M. Heinonen, H. Lähdesmäki, S. Kaski, *et al.*, “Deep learning with differential gaussian process flows,” in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2019.

[55] L. Li, M. L. Littman, T. J. Walsh, and A. L. Strehl, “Knows what it knows: a framework for self-aware learning,” *Machine learning*, vol. 82, no. 3, pp. 399–443, 2011.

[56] C. J. Roy and W. L. Oberkampf, “A comprehensive framework for verification, validation, and uncertainty quantification in scientific computing,” *Computer Methods in Applied Mechanics and Engineering*, vol. 200, no. 25-28, pp. 2131–2144, 2011.

[57] T. Lew, A. Sharma, J. Harrison, and M. Pavone, “On the Problem of Reformulating Systems with Uncertain Dynamics as a Stochastic Differential Equation. http://asl.stanford.edu/wp-content/papercite-data/pdf/dynsSDE.pdf,” *Technical Report*, 2020.

[58] J. Harrison, A. Sharma, and M. Pavone, “Meta-Learning Priors for Efficient Online Bayesian Regression,” *arXiv*, 2018. arXiv: `1807.08912`.

[59] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, “Taking the human out of the loop: A review of Bayesian optimization,” *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[60] Y. Pan, X. Yan, E. A. Theodorou, and B. Boots, “Prediction under uncertainty in sparse spectrum Gaussian processes with applications to filtering and control,” in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, pp. 2760–2768.

[61] D. Yarotsky, “Error bounds for approximations with deep ReLU networks,” *Neural Networks*, vol. 94, pp. 103–114, 2017.

[62] G. Shi, X. Shi, M. O’Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, “Neural lander: Stable drone landing control using learned dynamics,” in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 9784–9790.

[63] R. Khasminskii, *Stochastic stability of differential equations*. Springer Science & Business Media, 2011, vol. 66.

[64] A. Clark, “Control Barrier Functions for Complete and Incomplete Information Stochastic Systems,” in *2019 American Control Conference (ACC)*, 2019, pp. 2928–2935.

[65] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, “Control barrier function based quadratic programs for safety critical systems,” *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[66] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, “{OSQP}: An Operator Splitting Solver for Quadratic Programs,” *ArXiv e-prints*, pp. 1–36, 2017. arXiv: `1711.08013 [math.OC]`.

[67] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “ROS: an open-source Robot Operating System,” in *ICRA Workshop on Open Source Software*, 2009.

[68] A. Bemporad and M. Morari, “Robust model predictive control: A survey,” in *Robustness in identification and control*, Springer, 1999, pp. 207–226.

[69] L. Hewing and M. N. Zeilinger, “Stochastic Model Predictive Control for Linear Systems using Probabilistic Reachable Sets,” 2018. arXiv: `1805.07145`.

[70] K. P. Wabersich and M. N. Zeilinger, “Linear model predictive safety certification for learning-based control,” 2018. arXiv: `1803.08552`.

[71] H. Ravanbakhsh and S. Sankaranarayanan, “Learning Control Lyapunov Functions from Counterexamples and Demonstrations,” 2018. arXiv: `1804.05285`.

[72] Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli, "A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles," *Vehicle System Dynamics*, vol. 52, no. 6, pp. 802–823, 2014.

[73] D. D. Fan and E. A. Theodorou, "Differential Dynamic Programming for time-delayed systems," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, IEEE, 2016, pp. 573–579, ISBN: 978-1-5090-1837-6.

[74] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2014, pp. 4029–4036, ISBN: 978-1-4799-3685-4.

[75] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin, "Provably safe and robust learning-based model predictive control," *Automatica*, vol. 49, no. 5, pp. 1216–1226, 2013.

[76] M. Bujarbaruah, X. Zhang, M. Tanaskovic, and F. Borrelli, "Adaptive MPC under time varying uncertainty: Robust and Stochastic," *arXiv preprint arXiv:1909.13473*, 2019.

[77] B. T. Lopez, J. P. How, and J.-J. E. Slotine, "Dynamic tube MPC for nonlinear systems," in *2019 American Control Conference (ACC)*, IEEE, 2019, pp. 1655–1662.

[78] M. E. Villanueva, R. Quirynen, M. Diehl, B. Chachuat, and B. Houska, "Robust MPC via min–max differential inequalities," *Automatica*, vol. 77, pp. 311–321, 2017.

[79] J. Köhler, P. Kötting, R. Soloperto, F. Allgöwer, and M. A. Müller, "A robust adaptive model predictive control framework for nonlinear uncertain systems," *arXiv preprint arXiv:1911.02899*, 2019.

[80] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-Based Model Predictive Control: Toward Safe Learning in Control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, 2019.

[81] M. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (ICML-11)*, 2011, pp. 465–472.

[82] A. Girard, C. E. Rasmussen, J. Q. Candela, and R. Murray-Smith, "Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting," in *Advances in neural information processing systems*, 2003, pp. 545–552.

[83] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-Based Model Predictive Control for Safe Exploration," in *2018 IEEE Conference on Decision and Control (CDC)*, IEEE, 2018, pp. 6059–6066, ISBN: 978-1-5386-1395-5.

[84] E. Bradford, L. Imsland, and E. A. del Rio-Chanona, “Nonlinear model predictive control with explicit back-offs for Gaussian process state space models,” in *58th Conference on decision and control (CDC). IEEE*, 2019.

[85] N. Tagasovska and D. Lopez-Paz, “Single-Model Uncertainties for Deep Learning,” in *Advances in Neural Information Processing Systems*, 2019, pp. 6414–6425.

[86] R. Koenker and G. Bassett Jr, “Regression quantiles,” *Econometrica: journal of the Econometric Society*, pp. 33–50, 1978.

[87] J. W. Taylor, “A quantile regression approach to estimating the distribution of multiperiod returns,” *The Journal of Derivatives*, vol. 7, no. 1, pp. 64–78, 1999.

[88] F. Rodrigues and F. C. Pereira, “Beyond expectation: Deep joint mean and quantile regression for spatio-temporal problems,” *arXiv preprint arXiv:1808.08798*, 2018.

[89] F. Zhang, X. Fan, H. Xu, P. Zhou, Y. He, and J. Liu, “Regression via Arbitrary Quantile Modeling,” *arXiv preprint arXiv:1911.05441*, 2019.

[90] J. Sadeghi, M. De Angelis, and E. Patelli, “Efficient training of interval Neural Networks for imprecise training data,” *Neural Networks*, vol. 118, pp. 338–351, 2019.

[91] X. Yan, W. Zhang, L. Ma, W. Liu, and Q. Wu, “Parsimonious quantile regression of financial asset tail dynamics via sequential learning,” in *Advances in Neural Information Processing Systems*, 2018, pp. 1575–1585.

[92] H. Kozumi and G. Kobayashi, “Gibbs sampling methods for Bayesian quantile regression,” *Journal of statistical computation and simulation*, vol. 81, no. 11, pp. 1565–1578, 2011.

[93] Y. Yang, H. J. Wang, and X. He, “Posterior inference in Bayesian quantile regression with asymmetric Laplace likelihood,” *International Statistical Review*, vol. 84, no. 3, pp. 327–344, 2016.

[94] W. Dabney, M. Rowland, M. G. Bellemare, and R. Munos, “Distributional reinforcement learning with quantile regression,” in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[95] W. Langson, I. Chryssochoos, S. V. Raković, and D. Q. Mayne, “Robust model predictive control using tubes,” *Automatica*, vol. 40, no. 1, pp. 125–133, 2004.

[96] D. Q. Mayne, “Model predictive control: Recent developments and future promise,” *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.

[97] P. J. Huber, “Robust estimation of a location parameter,” in *Breakthroughs in statistics*, Springer, 1992, pp. 492–518.

[98] W. P. Gaglianone, L. R. Lima, O. Linton, and D. R. Smith, "Evaluating value-at-risk models via quantile regression," *Journal of Business & Economic Statistics*, vol. 29, no. 1, pp. 150–160, 2011.

[99] J. Sill, "Monotonic networks," in *Advances in neural information processing systems*, 1998, pp. 661–667.

[100] S. You, D. Ding, K. Canini, J. Pfeifer, and M. Gupta, "Deep lattice networks and partial monotonic functions," in *Advances in Neural Information Processing Systems*, 2017, pp. 2981–2989.

[101] A. Gupta, N. Shukla, L. Marla, and A. Kolbeinsson, "Monotonic Trends in Deep Neural Networks," *arXiv preprint arXiv:1909.10662*, 2019.

[102] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning*, Springer, 2003, pp. 63–71.

[103] D. Q. Mayne, E. C. Kerrigan, E. J. van Wyk, and P. Falugi, "Tube-based robust nonlinear model predictive control," *International Journal of Robust and Nonlinear Control*, vol. 21, no. 11, pp. 1341–1353, 2011.

[104] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear MPC and moving horizon estimation," in *Nonlinear model predictive control*, Springer, 2009, pp. 391–417.

[105] R. Camacho, R. King, and A. Srinivasan, *Inductive Logic Programming: 14th International Conference, ILP 2004, Porto, Portugal, September 6-8, 2004, Proceedings*. Springer Science & Business Media, 2004, vol. 3194.

[106] E. C. Kerrigan and J. M. Maciejowski, "Robust feasibility in model predictive control: Necessary and sufficient conditions," in *Proceedings of the 40th IEEE Conference on Decision and Control*, IEEE, vol. 1, 2001, pp. 728–733.

[107] D. Nguyen-Tuong, J. Peters, M. Seeger, and B. Schölkopf, "Learning inverse dynamics: a comparison," in *European symposium on artificial neural networks*, 2008.

[108] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[109] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE (3)," in *49th IEEE conference on decision and control (CDC)*, IEEE, 2010, pp. 5420–5425.

[110] H. Kalita, S. Morad, A. Ravindran, and J. Thangavelautham, "Path planning and navigation inside off-world lava tubes and caves," in *IEEE/ION Position, Location and Navigation Symposium*, 2018, pp. 1311–1318.

[111] R. Thakker, N. Alatur, D. D. Fan, J. Tordesillas, M. Paton, K. Otsu, O. Toupet, and A. Agha-mohammadi, "Autonomous Off-road Navigation over Extreme Terrains with Perceptually-challenging Conditions," *International Symposium on Experimental Robotics*, 2020.

[112] R. T. Rockafellar, S. Uryasev, *et al.*, "Optimization of conditional value-at-risk," *Journal of Risk*, vol. 2, no. 3, pp. 21–41, 2000.

[113] P. Papadakis, "Terrain traversability analysis methods for unmanned ground vehicles: A survey," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 4, pp. 1373–1385, 2013.

[114] S. B. Goldberg, M. W. Maimone, and L. Matthies, "Stereo vision and rover navigation software for planetary exploration," in *IEEE Aerospace Conference*, IEEE, 2002.

[115] A. Haït, T. Simeon, and M. Taïx, "Algorithms for rough terrain trajectory planning," *Advanced Robotics*, vol. 16, no. 8, pp. 673–699, 2002.

[116] K. Otsu, G. Matheron, S. Ghosh, O. Toupet, and M. Ono, "Fast approximate clearance evaluation for rovers with articulated suspension systems," *Journal of Field Robotics*, vol. 37, no. 5, pp. 768–785, 2020.

[117] S. Ghosh, K. Otsu, and M. Ono, "Probabilistic Kinematic State Estimation for Motion Planning of Planetary Rovers," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 5148–5154.

[118] P. Fankhauser, M. Bloesch, and M. Hutter, "Probabilistic Terrain Mapping for Mobile Robots With Uncertain Localization," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3019–3026, 2018.

[119] A. Agha-Mohammadi, E. Heiden, K. Hausman, and G. Sukhatme, "Confidence-rich grid mapping," *The International Journal of Robotics Research*, vol. 38, no. 12-13, pp. 1352–1374, 2017.

[120] M. Ono, M. Pavone, K. Kuwata, and J. Balaram, "Chance-constrained dynamic programming with application to risk-aware robotic space exploration," *Autonomous Robots*, vol. 39, no. 4, pp. 555–571, 2015.

[121] A. Wang, A. Jasour, and B. Williams, "Non-gaussian chance-constrained trajectory planning for autonomous vehicles under agent uncertainty," *Robotics and Automation Letters*, vol. 5, no. 4, pp. 6041–6048, 2020.

[122] S. Koenig and R. G. Simmons, “Risk-sensitive planning with probabilistic decision graphs,” in *Principles of Knowledge Representation and Reasoning*, Elsevier, 1994, pp. 363–373.

[123] H. Xu and S. Mannor, “Distributionally robust Markov decision processes,” in *Advances in Neural Information Processing Systems*, 2010, pp. 2505–2513.

[124] P. Artzner, F. Delbaen, J. Eber, and D. Heath, “Coherent measures of risk,” *Mathematical finance*, vol. 9, no. 3, pp. 203–228, 1999.

[125] Y. Chow, A. Tamar, S. Mannor, and M. Pavone, “Risk-sensitive and robust decision-making: a CVaR optimization approach,” in *Advances in Neural Information Processing Systems*, 2015, pp. 1522–1530.

[126] M. Ahmadi, M. Ono, M. D. Ingham, R. M. Murray, and A. D. Ames, “Risk-Averse Planning Under Uncertainty,” in *American Control Conference*, 2020, pp. 3305–3312.

[127] M. Ahmadi, U. Rosolia, M. D. Ingham, R. M. Murray, and A. D. Ames, “Constrained Risk-Averse Markov Decision Processes,” in *AAAI Conference on Artificial Intelligence*, 2021.

[128] S. Singh, Y. Chow, A. Majumdar, and M. Pavone, “A Framework for Time-Consistent, Risk-Sensitive Model Predictive Control: Theory and Algorithms,” *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2905–2912, 2018.

[129] A. Dixit, M. Ahmadi, and J. W. Burdick, “Risk-Sensitive Motion Planning using Entropic Value-at-Risk,” *arXiv 2011.11211*, 2020.

[130] A. Hakobyan, G. C. Kim, and I. Yang, “Risk-aware motion planning and control using CVaR-constrained optimization,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3924–3931, 2019.

[131] P. T. Boggs and J. W. Tolle, “Sequential quadratic programming,” *Acta numerica*, pp. 529–562, 1995.

[132] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, “Motion planning with sequential convex optimization and convex collision checking,” *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[133] T. Lew, R. Bonalli, and M. Pavone, “Chance-constrained sequential convex programming for robust trajectory optimization,” in *2020 European Control Conference (ECC)*, IEEE, 2020, pp. 1871–1878.

[134] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, “STOMP: Stochastic trajectory optimization for motion planning,” in *IEEE International Conference on Robotics and Automation*, 2011, pp. 4569–4574.

[135] A. Ruszczynski, “Risk-averse dynamic programming for Markov decision processes,” *Mathematical Programming*, vol. 75, no. 2, pp. 235–261, 2014.

[136] P. Fankhauser and M. Hutter, “A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation,” *Robot Operating System (ROS), The Complete Reference*, pp. 99–120, 2016.

[137] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[138] A. Bouman, M. F. Ginting, N. Alatur, M. Palieri, D. D. Fan, T. Touma, T. Pailevanian, S. Kim, K. Otsu, J. Burdick, and A. Agha-mohammadi, “Autonomous Spot: Long-range autonomous exploration of extreme environments with legged locomotion,” *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.

[139] Angel Santamaria, Rohan Thakker, David D. Fan, Benjamin Morrell, and Ali Agha, “Towards Resilient Autonomous Navigation of Drones,” *Proceedings of the International Symposium on Robotics Research*, 2019.

[140] S. J. Julier and J. K. Uhlmann, “Unscented filtering and nonlinear estimation,” *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[141] S.-K. Kim, A. Bouman, G. Salhotra, D. D. Fan, K. Otsu, J. Burdick, and A.-a. Agha-mohammadi, “PLGRIM: Hierarchical value learning for large-scale exploration in unknown environments,” in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 31, 2021, pp. 652–662.

[142] X. Warin, “Deep learning for efficient frontier calculation in finance,” *arxiv.org*, 2021. arXiv: 2101.02044.

[143] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone, “Risk-constrained reinforcement learning with percentile risk criteria,” *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6070–6120, 2017.

[144] D. Kivaranovic, K. D. Johnson, and H. Leeb, “Adaptive, distribution-free prediction intervals for deep networks,” in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2020, pp. 4346–4356.

[145] A. Tamar, Y. Glassner, and S. Mannor, “Optimizing the CVaR via sampling,” in *Proceedings of the National Conference on Artificial Intelligence*, vol. 4, 2015, pp. 2993–2999, ISBN: 9781577357025. arXiv: 1404.3862.

[146] C. Peng, S. Li, Y. Zhao, and Y. Bao, “Sample average approximation of CVaR-based hedging problem with a deep-learning solution,” *North American Journal of Economics and Finance*, vol. 56, 2021.

[147] P. Drews, G. Williams, B. Goldfain, E. A. Theodorou, and J. M. Rehg, “Aggressive deep driving: Combining convolutional neural networks and model predictive control,” in *Conference on Robot Learning*, PMLR, 2017, pp. 133–142.

[148] M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, and I. Posner, “Large-scale cost function learning for path planning using deep inverse reinforcement learning,” *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1073–1087, 2017.

[149] A. Choudhry, B. Moon, J. Patrikar, C. Samaras, and S. Scherer, “CVaR-based Flight Energy Risk Assessment for Multirotor UAVs using a Deep Energy Model,” *arXiv preprint arXiv:2105.15189*, 2021. arXiv: `2105.15189`.

[150] A. Hakobyan and I. Yang, “Distributionally robust risk map for learning-based motion planning and control: A semidefinite programming approach,” *arXiv preprint arXiv:2105.00657*, 2021. arXiv: `2105.00657`.

[151] G. C. Pflug, “Some Remarks on the Value-at-Risk and the Conditional Value-at-Risk,” in, 2000, pp. 272–281.

[152] O. Shamir and T. Zhang, “Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes,” in *International conference on machine learning*, PMLR, 2013, pp. 71–79.

[153] I. Kostavelis and A. Gasteratos, “Semantic mapping for mobile robotics tasks: A survey,” *Robotics and Autonomous Systems*, vol. 66, pp. 86–103, 2015.

[154] T. Lew, T. Emmei, D. D. Fan, T. Bartlett, A. Santamaria-Navarro, R. Thakker, and A.-a. Agha-mohammadi, “Contact Inertial Odometry: Collisions are your Friends,” *arXiv preprint arXiv:1909.00079*, 2019.

[155] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “Pointnet++: Deep hierarchical feature learning on point sets in a metric space,” *arXiv preprint arXiv:1706.02413*, 2017.

[156] Y. Zhou and O. Tuzel, “Voxelnet: End-to-end learning for point cloud based 3d object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4490–4499.

[157] S. Milz, M. Simon, K. Fischer, and M. Pöpperl, “Points2pix: 3d point-cloud to image translation using conditional generative adversarial networks,” *arXiv preprint arXiv:1901.09280*, 2019.

[158] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, “Image inpainting for irregular holes using partial convolutions,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 85–100.

[159] R. Koenker and J. A. Machado, “Goodness of Fit and Related Inference Processes for Quantile Regression,” *Journal of the American Statistical Association*, vol. 94, no. 448, pp. 1296–1310, 1999.

[160] R. T. Rockafellar, J. O. Royset, and S. I. Miranda, “Superquantile regression with applications to buffered reliability, uncertainty quantification, and conditional value-at-risk,” *European Journal of Operational Research*, vol. 234, no. 1, pp. 140–154, 2014.

[161] A. Deo and K. Murthy, “Optimizing tail risks using an importance sampling based extrapolation for heavy-tailed objectives,” in *2020 59th IEEE Conference on Decision and Control (CDC)*, IEEE, 2020, pp. 1070–1077.