Design and Analysis of Uncertainty-Aware Modularized Autonomy Stacks

by

Haotian Shen

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Claire J. Tomlin, Chair
Professor Avideh Zakhor

Spring 2024

Design and Analysis of Uncertainty-Aware Modularized Autonomy Stacks

by Haotian Shen

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

[illegible]

Professor Claire J. Tomlin
Research Advisor

May 16, 2024
(Date)

* * * * * * *

[illegible]

Professor Avideh Zakhor
Second Reader

5/16/24
(Date)

Design and Analysis of Uncertainty-Aware Modularized Autonomy Stacks

Abstract

Design and Analysis of Uncertainty-Aware Modularized Autonomy Stacks

by

Haotian Shen

Master of Science in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Claire J. Tomlin, Chair

We present the framework of uncertainty-aware modularized autonomy stack to describe modern robotic systems that utilize uncertainty quantification (UQ). In the first part of the thesis, we introduce a realization of the framework in navigation. We present a novel pipeline to obtain probabilistically safe and dynamically feasible reachable sets from a trajectory forecasting model using conformal prediction, as well as a planning method that leverages the safety guarantees of those sets. We showcase the efficacy of our pipeline in simulation with real autonomous driving data and in an experiment with Boeing vehicles. In the second part, we present an analysis of the framework through studying the system-wide impact of using UQ. We use level set estimation tools to efficiently quantify system robustness and calibration, even when the evaluation process is costly. We apply our analysis to two realistic industry-grade systems. We discover that UQ improves overall system robustness in the presence of input error, and that UQ enables a downstream module to give calibrated outputs despite erroneous outputs from upstream.

To my parents

Contents

List of Figures

List of Tables

Acknowledgments

None of this work could be done without the immense support from my advisor Professor Claire Tomlin. Thank you for the opportunity to allow me explore my research interests in the past two years and your guidance along the way. I am also grateful for the support from my second reader Professor Avideh Zakhor, who played an important role in my early interests in robotics and control during my undergraduate study.

I'd like to thank my wonderful collaborators: Professor Claire Tomlin, Anish Muthali, Sampada Deglurkar, Michael H. Lim, as well as industry partners and collaborators: Rebecca Roelofs, Aleksandra Faust from Google Research; Dragos Margineantu, James Paunicka, Blake Edwards, Jose Medina, Brandon Schwiesow, Zachary Tane and many more from Boeing; Professor Marco Pavone, Peter Karkus, Boris Ivanovic from NVIDIA Research. My work would not be possible with your support. It was a pleasure working with every one. It was an enlightening experience working with all the colleagues at Hybrid Systems Lab. I will forever cherish the memories of HSL.

Finally, I am indebted to my parents, who have made many sacrifices to support my academic and career pursuits. I would not be here without you. Thank you.

Chapter 1

Introduction

Many modern robotic systems are organized in terms of sequential modularized pipelines. For example, the perception-planning-control loop is a common architecture for vision-based robot navigation: the perception module is responsible for estimating the states of the agents; the planner proposes a high-level plan given the state estimates; and the controller tracks a trajectory based on the high-level plans. The outputs of the previous module are fed into the next one to enable downstream decision-making. Learning-based components are increasingly prevalent in the design of these pipelines. However, the learned components are often black-box models that lack interpretability and do not provide safety guarantees, especially in the presence of data distribution shifts. In the example above, the perception module may be a neural network trained on daylight image data, which can fail unexpectedly during operation at night time. Hence, it is challenging to produce assurances on the overall system's performance.

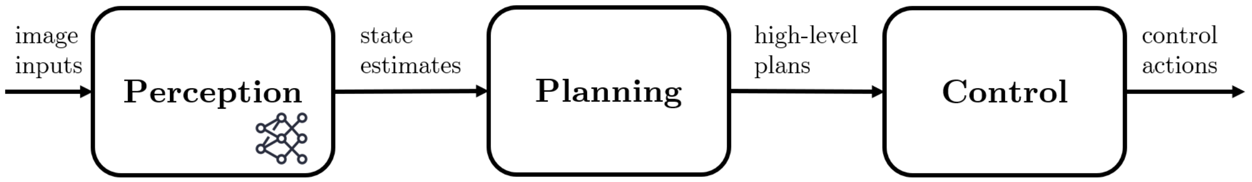


Figure 1.1: An example perception-planning-control pipeline for vision-based navigation. Neural network icon by dmitrychae.

Uncertainty quantification helps us mitigate model failures. Each upstream module augments its outputs with an uncertainty measure, representing how *uncertain* (or conversely, how confident) it is about its output. The downstream modules can thus incorporate the uncertainty measures to make safer and more robust decisions in the presence of large upstream errors. This work studies the the use of uncertainty quantification in autonomy stacks to provide system-wide safety assurances. In particular, we aim to advance our understanding of the following questions: How do we provide rigorous probabilistic guarantees for pipelines involving state-of-the-art black-box models? How do we quantify the impact of using uncertainty in the system beyond performance metrics? In Chapter 2, we present a novel pipeline to obtain probabilistically safe and dynamically feasible reachable sets from a trajectory forecasting model, as well as a planning framework that leverages the safety guarantees of those sets. In Chapter 3, we present a novel perspective on the design, use, and role of uncertainty measures for learning-based modules in an autonomous system. We use level-set estimation methods to analyze two real-world complex systems from the perspectives of robustness and calibration. In Chapter 4, we summarize our contributions and discuss several future directions.

Chapter 2

Multi-Agent Reachability Calibration with Conformal Prediction

2.1 Overview

The chapter investigates methods to provide rigorous safety assurances for autonomy stacks involving black-box models. In particular, we examine scenarios in which an autonomous agent, commonly described as "ego" agent, navigates around other, uncontrolled agents. Agents are allowed to be humans or be controlled by humans. Safety is critical in such situations, so modern systems employ forecasting models that reason about human behaviors. For example, in self-driving tasks, the autonomous car makes predictions of agents' trajectories to ensure the the safety of itself and the other vehicles it encounters. State-of-the-art systems try to achieve this through black-box behavior prediction and motion forecasting models [43, 51, 26], which lack rigorous safety assurances. While lower dimensional models can provide safety assurances, these methods rely on parametric assumptions on, for example, a human's rationality [23]. These assumptions can be inadequate for complex, multi-modal, and noisy decision-making scenarios. Uncertainty quantification methods can be applied. However, they alone are insufficient for safety assurance because they do not provide guarantees on model behavior and may be unreliable or uninterpretable [31, 58, 11].

We propose a novel pipeline to address the above issues. We first utilize conformal prediction to *calibrate* measures of uncertainty [54]. Conformal prediction is a statistical tool that uses a heuristic notion of risk to non-parametrically estimate quantiles of risk given a sequence of past observations [2]. Existing methods that leverage conformal prediction in the context of trajectory forecasting do not explicitly use interpretable metrics of prediction uncertainty [37, 17, 50, 12, 36]. We propose a method that provides rigorous confidence intervals on model error, a form of probabilistic assurance, given any interpretable heuristics on a trajectory forecasting model's prediction uncertainty. Our approach also allows us to examine the efficacy of various uncertainty quantification heuristics when attempting to predict model error. In addition, we extend our analysis to multi-agent environments to closely reflect real-world assurance cases.

By producing estimates of model error, we are able to couple statistical assurances with dynamical assurances to allow for safe downstream navigation. In particular, we turn to

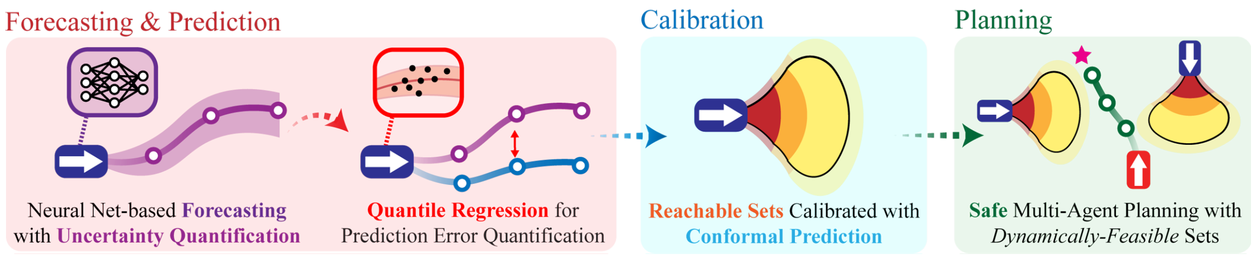

Forecasting & Prediction
Calibration
Planning
Neural Net-based Forecasting with Uncertainty Quantification
Quantile Regression for Prediction Error Quantification
Reachable Sets Calibrated with Conformal Prediction
Safe Multi-Agent Planning with Dynamically-Feasible Sets

Figure 2.1: Our method can be outlined as follows: given a trajectory forecasting model with an associated uncertainty heuristic, we design a quantile regression model that correlates uncertainty with prediction error, creating an approximate confidence interval on the model's prediction. We then apply conformal prediction to calibrate the confidence intervals and provide guarantees on miscoverage rate. We map the calibrated intervals in control action space to sets in state space through reachability analysis, and we demonstrate the utility of these confidence sets in planning tasks.

Hamilton-Jacobi (HJ) reachability analysis [5], which provides guarantees on dynamical systems by means of reachable sets and associated controllers. In HJ reachability, a Hamilton-Jacobi partial differential equation is solved to obtain an optimal value function and controller. The sub-zero level sets of this value function are the reachable sets (possible states of an agent at a given time) and tubes (possible states of an agent *up to* and including a given time).

The contributions of this chapter include:

1. A novel way to interpret trajectory forecasting models' prediction uncertainty and obtain approximate confidence intervals (Section 2.3);
2. A technique to calibrate the aforementioned intervals using conformal prediction (Section 2.3);
3. Dynamically-feasible, probabilistic reachable sets using calibrated intervals (Section 2.4);
4. A planning framework that leverages assurances developed in the previous steps (Section 2.4).

The chapter is organized as follows: Section 2.2 discusses related works in conformal prediction and assurances in trajectory forecasting models, Section 2.3 and Section 2.4 describe the contributions outlined above, and Section 2.5 showcases the safety and performance of our methods compared to baseline methods.

This chapter is primarily based on the joint work [39] with Anish Muthali, Sampada Deglurkar, Michael H. Lim, Rebecca Roelofs and Aleksandra Faust. The author would like to acknowledge and express gratitude for Anish's significant contribution in the writing of the chapter.

2.2 Related Works

Conformal Prediction

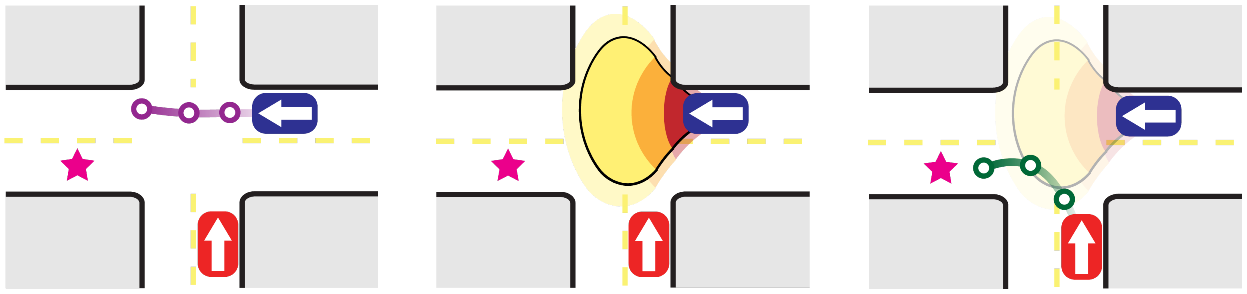
Conformal prediction [54, 2] is a class of uncertainty quantification methods for constructing prediction sets that satisfy a significance level (false negative rate) requirement. Traditionally, conformal prediction creates empirical histograms of measures of risk, called non-conformity scores, and uses these to estimate prediction intervals. Classical techniques include split conformal prediction, which creates empirical histograms from hold-out sets, and full conformal prediction, which creates empirical histograms using all available data [54]. Inductive conformal prediction, a variant of split conformal prediction, uses a non-conformity score that measures distance between train and test data [8]. These methods require that the data are identically distributed and exchangeable (any permutation of data points are identically distributed). Methods such as [49] relax the requirement for identically distributed data, and [57, 6] relax the exchangeability requirement. Adaptive Conformal Inference [25] and Rolling Risk Control (RollingRC) [20] have been proposed to relax all assumptions by further calibrating the significance level to match a desired error rate. We adapt RollingRC to provide safety assurances in any multi-agent scenario.

Probabilistic Reachability Frameworks

In this work, we introduce a method to generate probabilistic reachable sets to account for agents' dynamics. Previous work in this space typically involves randomly generating inputs and observing corresponding outputs of a dynamics model, with some associated guarantees in the sampling process [16]. Other methods, much like ours, leverage neural network uncertainty [40]. Specifically, the method of Nakamura and Bansal [40] uses Gaussian mixture models (GMMs) output by a trajectory forecasting model to generate parametric confidence intervals on control actions, which are then used as control bounds in reachability calculations. In our work, we attempt to relax assumptions that the control actions follow any parametric distribution by applying non-parametric inference techniques.

Safety Assurances in Trajectory Prediction

Various methods for incorporating uncertainty quantification have been examined for the purposes of providing safety assurances in trajectory prediction problems. Some methods provide probabilistic assurances by inferring parameters of a distribution on an agent's control actions [4, 23]. Methods such as [37] and [17] estimate confidence intervals with conformal prediction, implicitly leveraging prediction uncertainty through the non-conformity measure. Specifically, the method of Luo et al. [37] uses split conformal prediction to create a warning system, alerting drivers of "dangerous" situations. These warnings can be transformed into confidence sets, as shown in [17], which additionally eliminates exchangeability assumptions and incorporates trajectory optimization, much like our approach. Other methods emphasize the design process of the trajectory prediction neural networks, for instance by opting to use ReLU networks [12] or by opting to incorporate conformal prediction in the neural network's loss function [50]. In contrast to our approach, none of these methods consider the dynamic feasibility of confidence sets, and some methods that investigate conformal prediction, such as [12] and [37], assume exchangeability. Our method relaxes these assumptions while providing interpretability in the uncertainty quantification process and dynamic feasibility in confidence sets.



(a) Output of Trajectron++ in a simple scene with two vehicles.

(b) Approximate (opaque) and calibrated (translucent) reachable sets.

(c) Probabilistically-safe plan generated by the reachability-based planner.

Figure 2.2: Visualization of the running example. The autonomous ego vehicle is shown in red, and the human driver is shown in blue. The ego vehicle aims to navigate to the pink star while avoiding a collision with the human-driven vehicle. Confidence sets for the next three prediction steps are shown. In fig. 2.2b, the redder regions represent confidence sets for earlier prediction timesteps, and the translucent regions represent conformal prediction's calibration effect.

2.3 Assurances from Uncertainty

Our approach can be summarized in four primary steps: trajectory forecasting with uncertainty quantification (Section 2.3), leveraging uncertainty to obtain approximate prediction intervals (Section 2.3), calibrating approximate prediction intervals (Section 2.3), and obtaining dynamically feasible prediction sets (Section 2.4). We summarize our algorithm in Section 2.4, and we apply our approach to ego agent planning tasks in Section 2.4.

Running Example: *To motivate and illustrate our method, we introduce a simple running example with two vehicles at an intersection, one of them designated as the "ego" vehicle. In Figure 3.5, the autonomous ego vehicle, shown in red, aims to safely navigate to the pink-colored star while avoiding the blue vehicle.*

Trajectory Forecasting Model

We start by assuming access to a known dynamics model for each agent and a trajectory forecasting model capable of predicting an agent's control input. This trajectory forecasting model may maintain the capability to predict control actions for multiple agents at once, while considering interactions between agents. We denote the model as $f_T(\cdot) : \mathcal{X} \to \mathcal{U}$, where $\mathcal{X}$ is some arbitrary input space and $\mathcal{U}$ is a space over control actions. The network predicts $\mathbf{u}_{t:t+h} \in \mathcal{U}$, which is a collection of control action vectors indexed by timesteps t through $t+h$, for each of N total agents. Here, h is a fixed prediction horizon. We also assume the existence of an uncertainty measure on the network's outputs, denoted as $\sigma_T(\cdot) : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^d$, with d as the dimension of uncertainty representation. For example, a variance prediction or a variance estimate from inference-time dropout [31] is a valid uncertainty measure. Additionally, some neural network architectures, such as Trajectron++ [43], provide alternative uncertainty measures. This network architecture predicts a GMM over possible control actions, leading to understandings of prediction uncertainty such as the variance of the GMM's modes.

Running Example: *Given some sequence of the other vehicle's position history, Tra-*

jectron++, our trajectory forecasting model of choice, predicts a GMM in action space, and then integrates the actions to obtain states. We assume that vehicles follow the extended Dubins' car dynamics model. The state of this system is $\mathbf{x} = \begin{bmatrix} x & y & v & \theta \end{bmatrix}^\top$*, and the dynamics are given by* $\dot{\mathbf{x}} = \begin{bmatrix} v\cos(\theta) & v\sin(\theta) & u_1 & u_2 \end{bmatrix}^\top$*. The variance of the highest-probability Gaussian component, among other features of the GMM, are incorporated into the design of the uncertainty measure.*

Estimating Model Error from Uncertainty

To obtain confidence intervals on a black-box model's outputs, we estimate the neural network's confidence in an online manner, correlating its prediction uncertainty with prediction error. Quantile regression models enable us to map heuristic notions of uncertainty to an *approximate* confidence interval [34] along each action dimension. We choose a linear model since its simple parametrization allows for fast online updates and interpretability in how it perceives uncertainty. We demonstrate an example of interpretability in Section 2.5. Intuitively, our quantile regression models are approximately "calibrating" the network's uncertainty to obtain an estimate of its error.

As we observe new datapoints online, we collect $\mathbf{u}_{t-h:t}$, the last h ground truth control actions prior to timestep t. In practice, we estimate control actions by observing the state history of an agent, and then numerically computing derivatives to estimate actions from an assumed dynamics model. We contrast $\mathbf{u}_{t-h:t}$ with the network's previous prediction h timesteps ago, i.e., $\widehat{\mathbf{u}}_{t-h:t}$, and define $\mathbf{e}_{t-h:t} \coloneqq \mathbf{u}_{t-h:t} - \widehat{\mathbf{u}}_{t-h:t}$ as the prediction error.

Now, suppose we require a $1-\alpha$ approximate confidence interval on the ground truth control action. We can construct two quantile regression models $\widehat{q}_{\frac{\alpha}{2}} : \mathbb{R}^d \to \mathcal{U}$ and $\widehat{q}_{1-\frac{\alpha}{2}} : \mathbb{R}^d \to \mathcal{U}$ where $\widehat{q}_\varepsilon$ estimates the ε-quantile on the network's prediction error from timesteps t to $t+h$ for each agent, denoted $\widehat{\mathbf{e}}_\varepsilon$. For notational convenience, let us denote $\mathbb{P}_t(A)$ as the probability of event A conditioned on information until time t. We obtain an approximate $1-\alpha$ confidence interval as follows:

$$\mathbb{P}_t\Big(\widehat{\mathbf{e}}_{\frac{\alpha}{2}} \le \mathbf{e}_{t:t+h} \le \widehat{\mathbf{e}}_{1-\frac{\alpha}{2}}\Big) \tag{2.1}$$

$$= \mathbb{P}_t\Big(\widehat{\mathbf{e}}_{\frac{\alpha}{2}} \le \mathbf{u}_{t:t+h} - \widehat{\mathbf{u}}_{t:t+h} \le \widehat{\mathbf{e}}_{1-\frac{\alpha}{2}}\Big) \tag{2.2}$$

$$= \mathbb{P}_t\Big(\widehat{\mathbf{u}}_{t:t+h} + \widehat{\mathbf{e}}_{\frac{\alpha}{2}} \le \mathbf{u}_{t:t+h} \le \widehat{\mathbf{u}}_{t:t+h} + \widehat{\mathbf{e}}_{1-\frac{\alpha}{2}}\Big) \tag{2.3}$$

$$\approx 1-\alpha. \tag{2.4}$$

Thus, our approximate $1-\alpha$ confidence interval on $\mathbf{u}_{t:t+h}$ is $\widehat{\mathcal{I}}_{t:t+h} = [\widehat{\mathbf{u}}_{t:t+h} + \widehat{\mathbf{e}}_{\frac{\alpha}{2}}, \widehat{\mathbf{u}}_{t:t+h} + \widehat{\mathbf{e}}_{1-\frac{\alpha}{2}}]$.

Traditionally, quantile regression models are trained using computationally expensive linear programs [34], so instead, we opt for a faster, online gradient descent approach. We define our loss function for the quantile regression model $\widehat{q}_\varepsilon$ to be $\mathcal{L}(y, \widehat{y}) = (y - \widehat{y})\varepsilon\mathbf{1}\{y \ge \widehat{y}\} + (\widehat{y} - y)(1-\varepsilon)\mathbf{1}\{y < \widehat{y}\}$ (the "pinball loss") [46]. We set y to be the true model error, $\mathbf{e}$, and $\widehat{y} = \boldsymbol{\beta}^\top\boldsymbol{\sigma}$, where $\boldsymbol{\beta}$ represents the weights of the regression model, and $\boldsymbol{\sigma}$ is the uncertainty measure from the trajectory forecasting model. We update the weights according to $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \zeta\nabla_{\boldsymbol{\beta}}\mathcal{L}(\mathbf{e}_{t-h:t}, \boldsymbol{\beta})$, with learning rate ζ.

Calibrating Approximate Confidence Intervals

Given that the confidence intervals we obtained in the previous section are merely approximate, we aim to calibrate these intervals. To this end, we apply the RollingRC algorithm [20], which perfectly adapts to the online requirements of our method. We are motivated to use the RollingRC algorithm compared to other conformal prediction methods due to a desire to remove the data exchangeability assumption, since we allow for sequentially-dependent data and potential distribution shifts. In addition, we would like to train and calibrate the quantile regression models in a sample-efficient manner. RollingRC guarantees that the error rate deviates from α as $\mathcal{O}\big(1/T\big)$, where T is the total number of datapoints provided to the algorithm.

Following the notation from the RollingRC algorithm, we define $\theta_t \in \mathbb{R}$ as our conformal parameter, and $\varphi(\cdot) : \mathbb{R} \to \mathcal{U}$ as the algorithm's "stretching function". Now, we claim that

$$\begin{aligned}\mathbb{P}_t\Big(\widehat{\mathbf{e}}_{\frac{\alpha}{2}} - \varphi(\boldsymbol{\theta}) \le \mathbf{e}_{t:t+h} \le \widehat{\mathbf{e}}_{1-\frac{\alpha}{2}} + \varphi(\boldsymbol{\theta})\Big) \\ \ge 1 - \alpha - \mathcal{O}\big(1/t\big).\end{aligned} \tag{2.5}$$

Following similar steps as before, we obtain our newly calibrated confidence interval on $\mathbf{u}_{t:t+h}$ as $\mathcal{I}_{t:t+h} = [\widehat{\mathbf{u}}_{t:t+h} + \widehat{\mathbf{e}}_{\frac{\alpha}{2}} - \varphi(\boldsymbol{\theta}), \widehat{\mathbf{u}}_{t:t+h} + \widehat{\mathbf{e}}_{1-\frac{\alpha}{2}} + \varphi(\boldsymbol{\theta})]$.

2.4 Probabilistic Reachability and Planning

Probabilistic Reachability among Multiple Agents

In the previous sections, we have designed a method to provide confidence intervals on agents' control actions. However, for some downstream tasks, such as safe planning, confidence sets in spatial dimensions are more desirable. Hence, we use HJ reachability to obtain spatial sets, in the form of forward reachable tubes, on each agent's location given its dynamics and the probabilistic bound on control [44]. This procedure asserts that an agent's location will be contained in the produced reachable tube with probability $1 - \alpha$.

Suppose we wish to upper bound the probability that the ego vehicle collides with any agent. Let $\mathbf{x}_t^{(i)}$ be the location of non-ego agent i at timestep t and $\mathcal{S}[t]^{(i)}$ be the corresponding agent's forward reachable tube, as computed by our algorithm. We define miscoverage rate as the proportion of instances in which the ground truth position of any agent i at time t is outside $\mathcal{S}[t]^{(i)}$. We aim to obtain an upper bound on miscoverage rate, such that the ego agent can navigate in regions outside of $\mathcal{S}[t]^{(i)}$ for all $i \in \{1, \dots, N\}$ and guarantee that the probability of collision is at most γ, a pre-specified parameter. Consequently, we set the confidence interval significance level α according to our desired total miscoverage rate γ and number of agents N.

Theorem 2.4.1 (Significance Level Correction)
Suppose that we wish to have a total miscoverage rate of γ, where total miscoverage rate is an upper bound on the probability that *any* human agent is miscovered:

$$\mathbb{P}_t\left(\bigcup_{i=1}^{N}\Big\{\mathbf{x}_t^{(i)} \notin \mathcal{S}[t]^{(i)}\Big\}\right) \le \gamma. \tag{2.6}$$

Algorithm 1 Conformal Reachability Calibration.

```
1: procedure GenerateSets($\boldsymbol{\theta}$, $\widehat{\mathbf{u}}_{t:t+h}$, $\boldsymbol{\sigma}$)
2:     $\widehat{\mathbf{e}}_{\frac{\alpha}{2}} \leftarrow \widehat{q}_{\frac{\alpha}{2}}(\boldsymbol{\sigma})$ ▷ Obtain lower $\frac{\alpha}{2}$ quantile
3:     $\widehat{\mathbf{e}}_{1-\frac{\alpha}{2}} \leftarrow \widehat{q}_{1-\frac{\alpha}{2}}(\boldsymbol{\sigma})$ ▷ Obtain upper $\frac{\alpha}{2}$ quantile
4:     $\mathcal{I}_{t:t+h} \leftarrow \Big[\widehat{\mathbf{u}}_{t:t+h} + \widehat{\mathbf{e}}_{\frac{\alpha}{2}} - \varphi(\boldsymbol{\theta}),$
           $\widehat{\mathbf{u}}_{t:t+h} + \widehat{\mathbf{e}}_{1-\frac{\alpha}{2}} + \varphi(\boldsymbol{\theta})\Big]$
5:     $\mathcal{S} \leftarrow []$
6:     for $t' \in \{t, t+\Delta t, \ldots, t+h\}$ do
7:         $\mathcal{S}[t'] \leftarrow$ HJReachability($\mathcal{I}_{t'}$)
8:     return $\mathcal{S}, \mathcal{I}_{t:t+h}$
9: procedure Update($\boldsymbol{\theta}$, $\mathbf{u}_{t-h:t}$, $\mathcal{I}_{t-h:t}$)
10:    for $t' \in \{t, t+\Delta t, \ldots, t+h\}$ do
11:        $\boldsymbol{\theta}\{t'\} \leftarrow \boldsymbol{\theta}\{t'\} + \xi\big(\mathbf{1}\{\mathbf{u}_{t'-h} \notin \mathcal{I}_{t'-h}\} - \alpha\big)$
12:    GradientDescent($\widehat{q}_{\frac{\alpha}{2}}$, $\mathbf{u}_{t-h:t}$)
13:    GradientDescent($\widehat{q}_{1-\frac{\alpha}{2}}$, $\mathbf{u}_{t-h:t}$)
14:    return $\widehat{q}_{\frac{\alpha}{2}}$, $\widehat{q}_{1-\frac{\alpha}{2}}$, $\boldsymbol{\theta}$
15: procedure Main($\gamma$, $N$)
16:    $\alpha \leftarrow 1 - (1-\gamma)^{\frac{1}{N}}$
17:    $\boldsymbol{\theta}\{t, t+\Delta t, \ldots, t+h\} \leftarrow 0$
18:    $\widehat{q}_{\frac{\alpha}{2}}, \widehat{q}_{1-\frac{\alpha}{2}} \leftarrow$ InitializeRandomWeights( )
19:    $\mathbb{I} \leftarrow \{\}$
20:    $t \leftarrow 0$
21:    while true do
22:        $\widehat{\mathbf{u}}_{t:t+h}, \boldsymbol{\sigma} \leftarrow f_T(\cdot), \sigma_T(\cdot)$ ▷ Get trajectory predictions and uncertainty from model
23:        $\mathcal{S}, \mathcal{I}_{t:t+h} \leftarrow$ GenerateSets($\boldsymbol{\theta}$, $\widehat{\mathbf{u}}_{t:t+h}$, $\boldsymbol{\sigma}$)
24:        $\mathbb{I} \leftarrow \mathbb{I} \cup \mathcal{I}_{t:t+h}$
25:        if $t \geq h$ then
26:            $\mathbf{u}_{t-h:t} \leftarrow$ ObserveHistory( )
27:            $\mathcal{I}_{t-h:t} \leftarrow \mathbb{I}[t-h:t]$
28:            $\widehat{q}_{\frac{\alpha}{2}}$, $\widehat{q}_{1-\frac{\alpha}{2}}$, $\boldsymbol{\theta} \leftarrow$ Update($\boldsymbol{\theta}$, $\mathbf{u}_{t-h:t}$, $\mathcal{I}_{t-h:t}$)
29:        $t \leftarrow t + \Delta t$
```

We claim that the following α achieves an (asymptotic) total miscoverage rate of γ for N human agents:

$$\alpha = 1 - (1-\gamma)^{\frac{1}{N}}. \tag{2.7}$$

The proof of Theorem 2.4.1 is available in Appendix A, which uses the fact that the N agents act independently conditioned on past information [39]. Since α must be a fixed quantity in our algorithm, we must also fix N. Hence, we fix our algorithm to only consider the N agents closest to the ego vehicle.

Running Example: *Suppose we want a 95% probability safety assurance. Since there is only one other vehicle, we get $\alpha = 0.05$ from Theorem 2.4.1. Given the previous predictions of the blue agent's trajectory, we generate uncalibrated, time-indexed intervals on ranges of possible control actions, denoted $\widehat{\mathcal{I}}_t, \widehat{\mathcal{I}}_{t+\Delta t}, \widehat{\mathcal{I}}_{t+2\Delta t}$. We calibrate these using conformal prediction to obtain $\mathcal{I}_t, \mathcal{I}_{t+\Delta t}, \mathcal{I}_{t+2\Delta t}$. As we explain in the next subsection, HJ reachability allows us to take any sequence of intervals on control actions and generate a time-indexed* *set of states. In Figure 2.2b, we distinguish the effects of quantile regression and RollingRC's calibration.*

Full Algorithm

In Algorithm 1, we demonstrate the final algorithm to generate probabilistic reachable sets. The HJReachability function generates reachable sets given a probabilistic range of control actions, $\mathcal{I}_{t:t+\Delta t}$. Since the range can differ over time (e.g., $\mathcal{I}_t \neq \mathcal{I}_{t+\Delta t}$ necessarily), we iteratively compute time-indexed forward reachable tubes by computing the forward reachable tube over $[t, t+\Delta t]$ and using the reachable *set* at $t+\Delta t$ as the initial condition to compute the reachable tube over $[t+\Delta t, t+2\Delta t]$. We also utilize a GradientDescent function that updates the weights of the quantile regression models as described in Section 2.3. In Algorithm 1, ξ is the "learning rate" associated with the RollingRC algorithm.

Safe Planning Framework

Given the time-indexed sets $\mathcal{S}[t] \subseteq \mathcal{S}[t+\Delta t] \subseteq \cdots \subseteq \mathcal{S}[t+h]$, we desire that the autonomous agent's location at time t' is outside $\mathcal{S}[t+k\Delta t]$, where $t+(k-1)\Delta t \leq t' \leq t+k\Delta t$. We can plan by treating each agent's time-indexed forward reachable tube as a dynamic obstacle that grows with time. The obstacle-aware planning requirement motivates the application of a forward reach-avoid tube for the ego agent [21, 5]. We use this to derive an optimal control trajectory by selecting the Hamiltonian-maximizing control trajectory to a desired final state within the forward reach-avoid tube. In practice, this trajectory can involve bang-bang control, so one can track it using a tracker with a provable tracking error bound, such as a constrained iterative linear quadratic regulator. The planner's output is visualized in Appendix D.

Running Example: *From the previous section, we obtained $\mathcal{S}[t], \mathcal{S}[t+\Delta t], \mathcal{S}[t+2\Delta t]$ as a probabilistic occupancy region on the location of the other vehicle. Now, we can use the time-varying avoidance regions to plan a safe path to the goal in Figure 2.2c. Notice that the planner allows the ego agent to traverse in the yellow-colored region: it is aware that the ego vehicle would not violate the safety assurance as it can leave the yellow region by the time the other agent would enter it.*

2.5 Results

We compare the empirical safety and efficiency of our contribution to two baselines, Online Update of Safety Assurances Using Confidence-Based Predictions by Nakamura and Bansal [40] and Sample-Efficient Safety Assurances using Conformal Prediction by Luo et al. [37]. For both baselines, we perform the significance level correction described in Section 2.4.

For all benchmarking purposes, we use Trajectron++ trained on the relevant datasets. We follow the same architecture and hyperparameters as [43] by using 4 seconds (8 steps) of history to predict 3 seconds (6 steps) into the future. This is consistent with the other baselines' approaches. Set sizes are shown in square meters. We use a pre-specified total miscoverage rate of $\gamma = 0.05$, and we generate predictions for the closest $N = 3$ agents, which

Table 2.1: Coverage Rates and Set Sizes for $1-\gamma = 0.95$.

	Coverage Rates for Prediction Step					
Methods	1st (.5s)	2nd (1s)	3rd (1.5s)	4th (2s)	5th (2.5s)	6th (3s)
nuScenes Dataset						
Nakamura and Bansal	0.926 ±0.012	0.854 ±0.017	0.816 ±0.023	0.842 ±0.023	0.868 ±0.022	0.902 ±0.020
Luo et al.	1.000 ±0.000	1.000 ±0.000	1.000 ±0.000	0.998 ±0.002	0.989 ±0.006	0.968 ±0.012
Our Method	0.964 ±0.008	0.962 ± 0.011	0.968 ±0.010	0.975 ±0.008	0.981 ±0.007	0.985 ±0.007
Waymo Dataset						
Nakamura and Bansal	0.981 ±0.007	0.954 ±0.012	0.938 ±0.014	0.937 ±0.015	0.952 ±0.014	0.955 ±0.015
Luo et al.	1.00 ±0.00	1.00 ±0.00	1.00 ±0.00	0.998 ±0.002	0.985 ±0.009	0.955 ±0.015
Our Method	0.997 ±0.002	0.986 ±0.006	0.980 ±0.008	0.967 ±0.011	0.965 ±0.011	0.965 ±0.013
	Set Sizes for Prediction Step					
nuScenes Dataset						
Nakamura and Bansal	57 ±2	320 ±12	886 ±35	2060 ±85	3259 ±128	4285 ±160
Luo et al.	425 ±11	523 ±16	683 ±34	1097 ±109	1426 ±140	1814 ±186
Our Method	39 ±3	157 ± 13	462 ±39	1078 ±88	2150 ±170	3713 ±268
Waymo Dataset						
Nakamura and Bansal	64 ±7	311 ±32	951 ±96	2117 ±195	3814 ±328	5892 ±475
Luo et al.	448 ±8	736 ±43	1110 ±94	1568 ±169	2126 ±237	2687 ±301
Our Method	61 ±6	246 ±27	655 ±71	1361 ±143	2422 ±240	3885 ±365

strikes a balance between the speed of our HJ reachability calculations and the practical safety of the system.

nuScenes Dataset Results

We compare the coverage rate and efficiency of our method against the two baselines on nuScenes self-driving data [9]. We calculate average coverage rate and average set sizes individually for each forward prediction step $t, t+\Delta t, \dots, t+h$ on 100 randomly sampled scenes. For each scene, we use the first 13 seconds to calibrate each method and make predictions on the last 5.5 seconds. Table 2.1 shows step coverage and set sizes at all prediction steps. Note that an ideal algorithm maintains a coverage rate over $1-\gamma$ while providing the smallest prediction sets.

Waymo Open Motion Dataset Results

To demonstrate the planning safety and efficiency of each method, we also perform experiments on the Waymo Open Motion Dataset [18], coupled with the Nocturne simulator [53]. This allows us to apply control actions to the ego vehicle while all other agents replay their respective sequences of control actions from the dataset. We use the same planning method discussed in Section 2.4 for all three methods, since neither of the baselines have associated planners. For each scene, we calibrate using the first 7 seconds and use model-predictive control to plan for the last 3 seconds, where the goal is the final position of the ego vehicle in the ground truth data. We measure three quantities: (1) progress to goal, defined as the ratio of the distance from the final state of the ego vehicle to the goal compared to the distance from the start to the goal, subtracted from 1; (2) collision rate; (3) conservatism of each method compared to the ego vehicle's ground truth trajectory, defined as the ratio of mini-

Table 2.2: Waymo Planning Benchmarks

Method	**Progress to Goal**	**Collision Rate**	**Conservatism**
Nakamura and Bansal	0.494 ±0.029	**0.0**	**1.504** ±0.068
Luo et al.	0.305 ±0.028	0.005	1.626 ±0.072
Our Method	**0.544** ±0.028	**0.0**	1.507 ±0.068

mum distance between the ego vehicle to other agents at all times as a result of the planner, compared to that of the ground truth. The formulas and computations of these metrics are described in detail in Appendix B [39]. In Appendix C, we additionally demonstrate the impact of the aforementioned theoretical guarantees by providing safety and efficiency metrics in the absence of conformal prediction [39]. We performed the benchmarks on 200 randomly sampled scenes. Table 2.1 depicts coverage rates and set sizes for all prediction timesteps. Table 2.2 depicts average collision rate, average progress to goal, and conservatism.

Discussion of Results

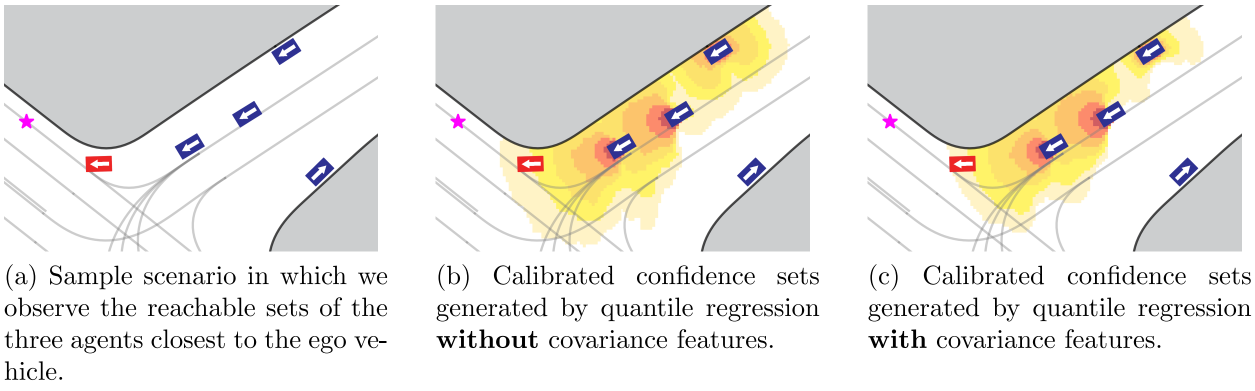
For the nuScenes dataset, we notice that our method achieves more efficient set sizes for initial prediction steps, while Luo et al. achieves more efficient set sizes for later prediction timesteps. Nevertheless, neither of these two methods violates the miscoverage requirement of $\gamma = 0.05$. The method of Nakamura and Bansal violates the miscoverage rate, however, supporting the introduction of uncertainty calibration into the algorithm. Hence, calibrating neural network uncertainty is important, not only to provide the desired coverage rate but also to generate efficient prediction sets.

For the Waymo dataset, we notice a very similar phenomenon with set sizes and coverage rates. In the planning benchmarks, our method has the best progress to goal, likely due to the initial-timestep sets being smaller. This is also reflected in the conservatism scores, with reachability-based methods performing the best. The method of Luo et al. also encountered one collision scenario in which the produced set was very large and forced the planner to take a sharp avoid action. Thus, we note the importance of initial-timestep sets being small to allow the reachability-based methods to perform better in the planning benchmarks. This allows the ego vehicle to make some progress, whereas a large initial-timestep set would inhibit any progress regardless of the relative size of later timesteps' sets.

Case Studies

Understanding Uncertainty Measures

In this case study, we demonstrate the usefulness of our interpretable quantile regression model when understanding the efficacy of uncertainty metrics. Consider the scene in Figure 2.3a. We choose the uncertainty measure based on properties of the GMM, including the distance between peaks and the (co)variance of the highest-weighted mode. The learned regression model indicates a positive correlation between prediction error and variance of the


(a) Sample scenario in which we observe the reachable sets of the three agents closest to the ego vehicle.

(b) Calibrated confidence sets generated by quantile regression **without** covariance features.

(c) Calibrated confidence sets generated by quantile regression **with** covariance features.

Figure 2.3: Case Study of Uncertainty Metrics. We demonstrate a simple example in which the choice of uncertainty measure affects the size of sets, with coverage rate held constant.

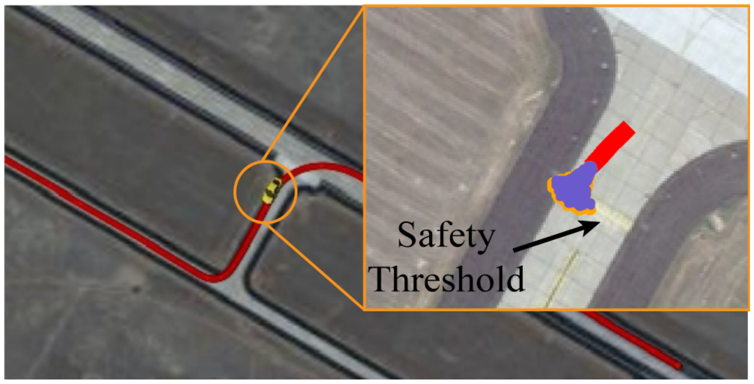


Figure 2.4: Our algorithm is applied to assure safety in potential runway incursion scenarios. Once the ground vehicle is determined to have crossed a designated safety threshold, the aircraft is cleared to land.

most-likely GMM mode. In Figure 2.3b and Figure 2.3c, we can visually discern the benefit of including these features.

Overall, this case study shows the importance of understanding the usefulness of different components of the uncertainty measure. A more useful uncertainty metric can provide more efficient sets, since a more accurate quantile regression model would require less calibration (less "stretching" from conformal prediction). Conformal prediction cannot derive confidence intervals conditional on some input, so quantile regression's accuracy is crucial for providing efficient sets.

Safety in Aerospace Applications

In this case study, we apply our algorithm to satisfy a real-world safety assurance requirement by demonstrating our algorithm on Boeing vehicles. We consider the case of an aircraft attempting to land on a runway while accounting for potential runway incursions from ground vehicles. We use our algorithm to provide assurances on the motion of a ground vehicle on the runway. Given a fixed landing plan for the plane, we adapt the sets from our algorithm to design a warning system similar to Luo et al.'s original algorithm. If a prediction set intersects the runway, a warning is issued. A visualization of this application is shown in Figure 2.4. The ground vehicle's state history is shown in red, and its uncalibrated prediction set is shown in purple. The "stretching" effect from conformal prediction is shown in orange.

2.6 Acknowledgements

This material is based upon work supported by the DARPA Assured Autonomy Program, the SRC CONIX program, Google-BAIR Commons, the NASA ULI program on Safe Aviation Autonomy, and the National Science Foundation Graduate Research Fellowship Program under Grant Nos. DGE 1752814 and DGE 2146752. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of any aforementioned organizations. The author also thanks James Paunicka, Blake Edwards, Dragos Margineantu, Douglas Stuart, and the entire team at Boeing for all their help and contributions in demonstrating our method on Boeing's runway incursion dataset. Finally, the author thanks Marius Wiggert for helping with the implementations in the HJ reachability toolbox.

Chapter 3

A System Perspective on the Uncertainty of Learning-Based Components in the Autonomy Stack

3.1 Overview

Chapter 2 describes the design of an uncertainty-aware modular pipeline centered around providing rigorous, probabilistic safety assurance. In this chapter, we take a step back and consider the general principles behind systems that use uncertainty quantification. We examine the safety properties of a system on a high level. Robustness and calibration are two important safety axes of autonomous systems. A robust system does not degrade drastically in the presence of errors. A well-calibrated system provides probability estimates representative of the true likelihood. We hypothesize that by incorporating uncertainty quantification in the modules, we can create a more robust and well-calibrated pipeline overall. We investigate the design, use and role of uncertainty measures, in an attempt to quantify the impact of using uncertainty.

Different techniques for computing uncertainty for neural networks have been proposed, including test-time dropout [24], model ensembling [35], variance propagation [42], and out-of-distribution (OoD) detection [45], to name a few. Some models are even designed to output a distribution, which can provide natural uncertainty heuristics [30, 28]. There is not a systematic way to differentiate between the plethora of techniques. As a result, it is difficult for system architects to select an uncertainty measure best suited for their use case. We argue that the efficacy of an uncertainty measure needs to be considered *in conjuction with* the decision-making under uncertainty algorithm. An uncertainty measure is only "good" if it can produce good downstream performance, which motivates viewing uncertainty from the perspective of the overall system. Our first key insight lies in the use of level set estimation tools to quantitatively and efficiently analyze the robustness of a system, even when the evaluation process is costly. We propose a metric, the sub-level set size in input error space, and argue its usefulness in judging system robustness. Our second key insight is that a downstream module is capable of giving calibrated outputs despite erroneous outputs from upstream. We show this by evaluating module i's calibration error *conditioned on* the error and uncertainty of module $i-1$.

The chapter is organized as follows: Section 3.2 discusses the related works, Section 3.3 describes our abstraction of an uncertainty-aware modularized stack, the formulation of our two key insights, and a level set estimation technique. In Section 3.4, we demonstrate a robustness analysis on a modern modular self-driving stack. In Section 3.5, we discuss the calibration perspective on an aircraft runway incursion detection system.

This chapter is primarily based on the joint work with Sampada Deglurkar, Anish Muthali, Marco Pavone, Dragos Margineantu, Peter Karkus, Boris Ivanovic. The author would like to acknowledge and express gratitude for Sampada's significant contribution in the writing of the chapter.

3.2 Related Works

There is an abundance of work in the controls community on systems theory and design. Scenario optimization considers the design of modules or entire systems under uncertainty, which takes the form of "scenarios", which are random variables from an unknown distribution. The principle is that a system design should be performant while also being robust and satisfying constraints that are functions of the uncertainty. There has also been much recent interest in system design that takes advantage of differentiability in the system components, using it to simplify optimization [32], [15].

Much of this theory also helps lay the groundwork to study the robustness of a system to external disturbances or errors, and to study system sensitivity and failure points [14]. For learning-in-the-loop systems, some of this robustness analysis often takes the form of backward or forward reachability analysis for neural networks, either as standalone modules or as part of feedback loops [19], [52]. The level sets that we will produce in this work are reminiscent of such reachable sets. Additionally, [33] shows how this kind of set generation is also a form of specification generation – that is, producing specifications on module properties given system-level specifications. In our work, we recognize the dual role of level set generation as a way to both produce specifications and assess robustness.

Calibration is an extensively studied topic in deep learning literature, including techniques of calibration [27], metrics of calibration [41, 3] and theoretical analysis of calibration measures [7]. [38] considers calibration in the context of reinforcement learning. [56] studies the effect of using calibration and conformal prediction simultaneously.

There has also been much interest in contextualizing individual modules within the larger system or otherwise better taking into account interconnections between modules [60], [39], [29], [13], [10]. There is a growing realization in the community that modules, especially learned ones, should not be designed in isolation. We take a similar philosophy in this work.

3.3 Problem Setup

Definitions and Assumptions

We consider a general robotics stack consisting of n modules. The first module $f_1 : \mathcal{X} \to \mathcal{Y}_1$ takes in the input sensor measurement $x \in \mathcal{X}$. Each subsequent module, indexed by $i =$

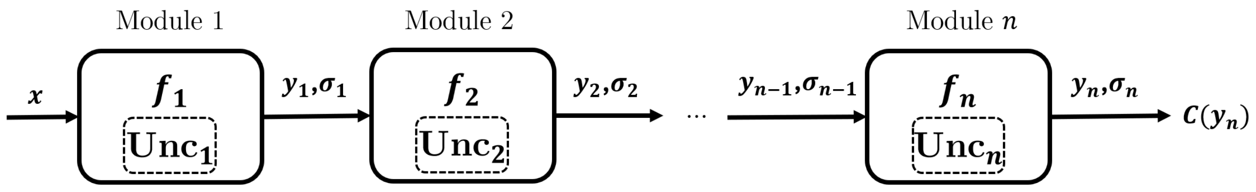


Figure 3.1: A general autonomy stack consisting of n functions arranged sequentially. The y_i's are output by the modules f_i while the σ_i's are uncertainties measures by an (optional, drawn as dotted lines) uncertainty quantification method Unc_i. x is the input of the stack. $C(y_n)$ is the cost of the final output.

$2, ..., n$, can be represented as a function f_i that maps the previous module's outputs to its own. We denote the output as $y_i \in \mathcal{Y}_i$. Each module is also optionally equipped with an uncertainty quantification function $\mathrm{Unc}_i(\cdot)$, which produces an uncertainty measure $\sigma_i \in \mathbb{R}^{d_i}$ indicating to the subsequent modules how "trustworthy" the output y_i is. In the event that we choose to not use such Unc_i, we let $\sigma_i = \mathrm{null}$. Thus, for $i = 2, ..., n$ we have

$$y_i = f_i(y_{i-1}, \sigma_{i-1}) \tag{3.1}$$

and

$$\sigma_i = \begin{cases} \mathrm{Unc}_i(f_i, y_{i-1}, \sigma_{i-1}) & \\ \mathrm{null} & \text{if } \mathrm{Unc}_i \text{ unavailable} \end{cases} \tag{3.2}$$

We keep the functional form of $\mathrm{Unc}_i(\cdot)$ generic to allow our framework to accommodate various uncertainty quantification techniques. We also have access to a cost function for evaluating the overall performance of the entire stack for a given input, written as $C : \mathcal{Y}_n \to \mathbb{R}$. Figure 3.1 depicts our general setting. Each f_i is assumed to be deterministic and time invariant. Our methods do not make a distinction between learned and non-learned (i.e. classical) modules. We slightly abuse notations for simplicity by writing

$$g_i(z_{i-1}) := \big(f_i(y_{i-1}, \sigma_{i-1}), \mathrm{Unc}_i(f_i, y_{i-1}, \sigma_{i-1})\big) \tag{3.3}$$

where $z_i := (y_i, \sigma_i)$. Hence we are able to "chain" the modules and express the end-to-end cost as

$$\tilde{C}(x) := (C \circ g_n \circ g_{n-1} ... \circ g_1)(x, \mathrm{null}) \tag{3.4}$$

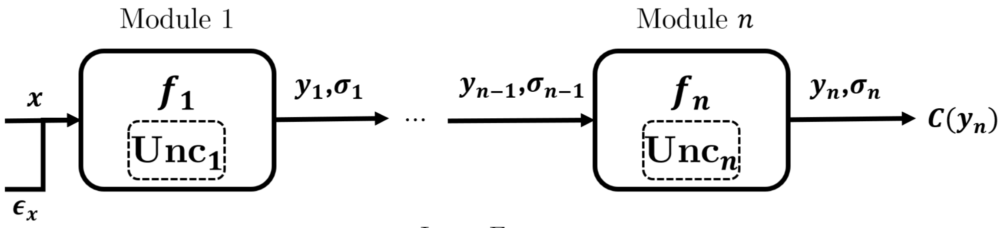
A common system specification is to require a system to overall perform "well enough" in expectation. Formally, we desire:

$$\mathbb{E}_{x \sim \mathcal{D}}[\tilde{C}(x)] < c \tag{3.5}$$

where $\mathcal{D}$ is the distribution of input data x and c is an acceptable threshold on the end-to-end cost.

Robustness

In this work, we define robustness as the ability of a system to perform at an acceptable level, even in the presence of external disturbances. We define *input error* to be any errors that may exist in x, for example due to sensor inaccuracies or data corruption, and denote it



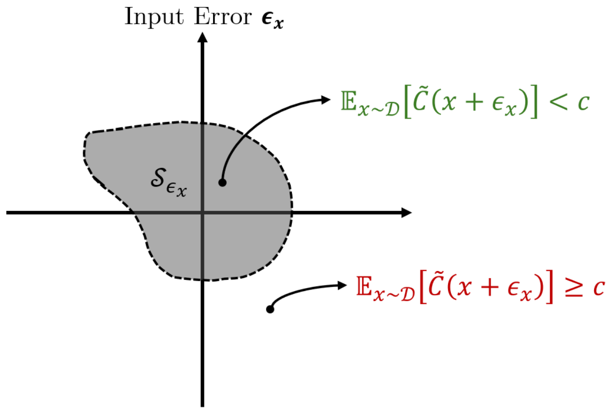

Input Error ϵ_x
$\mathbb{E}_{x\sim\mathcal{D}}[\tilde{C}(x+\epsilon_x)] < c$
$\mathcal{S}_{\epsilon_x}$
$\mathbb{E}_{x\sim\mathcal{D}}[\tilde{C}(x+\epsilon_x)] \geq c$

Figure 3.2: A visualization of the level set (shaded area) over two-dimensional input errors ϵ_x with respect to the entire stack. For points within the level set, the system specification is satisfied (green). For points outside the level set, the system specification is violated (red).

as $\epsilon_x \in \mathcal{X}$. A meaningful metric of system robustness is the size of the set of input errors that the system can "tolerate", or operate on while still satisfying the specification. Formally, this is

$$\mathcal{S}_{\epsilon_x} := \{\epsilon_x \mid \mathbb{E}_{x\sim\mathcal{D}}[\tilde{C}(x+\epsilon_x)] < c\} \tag{3.6}$$

where we approximate the expectation using samples from the dataset. Note that $\mathcal{S}_{\epsilon_x}$ is a sub-level set of the expected cost function, therefore making the computation of system robustness a level set estimation problem. We can produce $\mathcal{S}_{\epsilon_x}$ for different choices of f_is and Unc_is in order to evaluate different system designs. In this way, we can quantify the impact of using uncertainty in the system. Figure 3.2 depicts the goal of this problem statement.

Calibration

For this problem statement we zoom in on a module (as opposed to the whole stack) designed to output a probability, such as an algorithm predicting probability of vehicle collision. Suppose $y_i = f_i(y_{i-1}, \sigma_{i-1})$ and $\sigma_i \in [0,1]$, generated by Unc_i, represents its confidence on y_i. Let y_i^* be the ground truth label. If we have access to the ground truth probability p^*, perfect calibration is simply $\sigma_i = p^*$, so miscalibration is written as $\mathbb{E}_{\sigma_i}|p^* - \sigma_i|$. However, we typically do not possess the knowledge of the ground truth p^*, in which case the existing literature define perfect calibration as

$$\mathbb{P}(y_i = y_i^* \mid \sigma_i = p) = p \quad \forall p \in [0,1] \tag{3.7}$$

and miscalibration [27] as

$$\mathbb{E}_{\sigma_i}\left[\mid \mathbb{P}(y_i = y_i^* \mid \sigma_i = p) - p \mid\right] \tag{3.8}$$

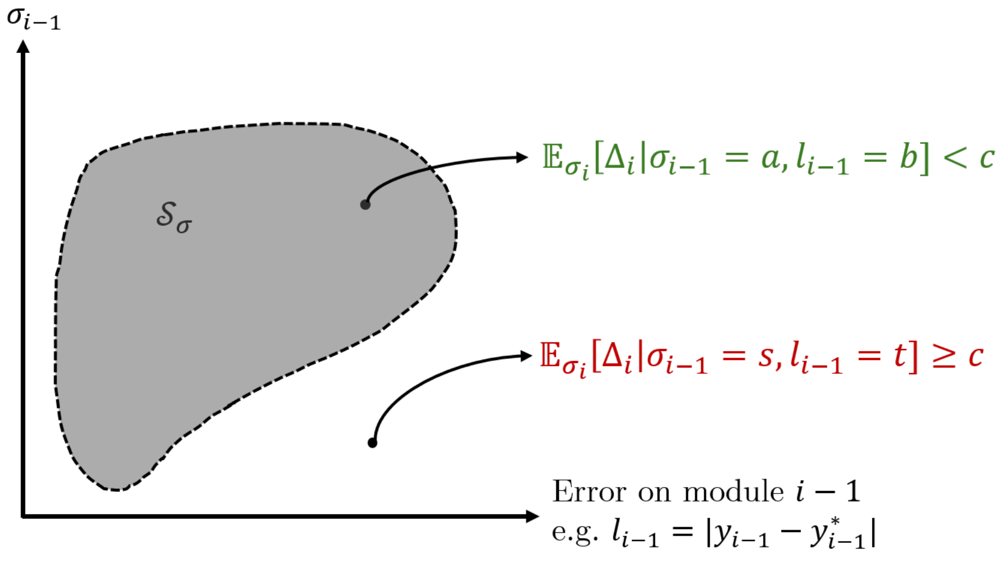

σ_{i-1}
$\mathcal{S}_\sigma$
$\mathbb{E}_{\sigma_i}[\Delta_i|\sigma_{i-1} = a, l_{i-1} = b] < c$
$\mathbb{E}_{\sigma_i}[\Delta_i|\sigma_{i-1} = s, l_{i-1} = t] \geq c$
Error on module $i-1$
e.g. $l_{i-1} = |y_{i-1} - y^*_{i-1}|$

Figure 3.3: A visualization of the level set (shaded area) over the loss and uncertainty of the previous module.

In the context of a modularized stack, we extend the notion of miscalibration to miscalibration *conditioned on* the error and uncertainty of the preceding module. Formally, let $\Delta_i = \mid \mathbb{P}(y_i = y_i^* \mid \sigma_i = p) - p \mid$, $l_i = \mathcal{L}(y_i, y_i^*)$ for some loss function $\mathcal{L}$. We compute

$$\mathbb{E}_{\sigma_i}\left[\Delta_i \mid l_{i-1} = a, \sigma_{i-1} = b\right] \tag{3.9}$$

Similar to our first problem, given a threshold c on miscalibration, we can define the set

$$\mathcal{S}_\sigma := \{(a, b) \mid \mathbb{E}_{\sigma_i}\left[\Delta_i \mid l_{i-1} = a, \sigma_{i-1} = b\right] < c\} \tag{3.10}$$

which we visualize in Figure 3.3. Note this is again a sub-level set. We desire the set to be large, which means despite high loss at module $i-1$, thanks to the design of module i, it is still possible to obtain relatively well-calibrated confidences. We will later demonstrate a comparison between two uncertainty-aware modules in which the more "intelligent" design obtains larger sets.

Sub-Level Set Estimation Using Gaussian Processes

We introduce the tool of sub-level set estimation that we use to solve the problems defined above. Note that various techniques exist for level set estimation, and any one of them can be used without losing sight of the overall system-aware uncertainty argument.

In both of the subsections above, we desire to estimate the set of inputs to a function such that the output of the function is below a threshold. A Monte-Carlo sampling-based approach seems reasonable for working with general sequences of system modules. However, in the event that the dataset size is large and the dimensionality of the input space is high, naive Monte-Carlo sampling over each grid point is potentially computationally prohibitive and motivates using Gaussian Processes (GPs) instead within a Bayesian optimization framework [22]. In this framework, we take samples of the expectations over the input space and model the expectation as a GP. We use an acquisition function to determine where in the input space to sample next. This usage of the GP to relate relevant quantities to system-level specifications is according to the guidance given by Katz et al. [33].

Let us use the placeholder $F(\eta)$ to refer to the expectations in Equations (3.6) and (3.10), where η represents the input variable. We would like η to be included in the level set if

$$\mathbb{P}(F(\eta) < c) > \lambda \tag{3.11}$$

for a confidence threshold λ. The probability is over the posterior of the GP. Similar to Katz et al., we use the MILE acquisition function [59] to produce samples of η. We then evaluate F on those samples and produce the posterior probability distribution over F, expressed in terms of the mean function $\mu_{GP}(\eta)$ and the standard deviation $s_{GP}(\eta)$. A point η_{test} is included in the level set if

$$\mu_{GP}(\eta_{test}) + \beta * s_{GP}(\eta_{test}) < c \tag{3.12}$$

where β is set according to the confidence threshold λ.

3.4 Analyzing System Robustness

In this section, we explore our first problem statement, robustness analysis, on a realistic, industry-grade autonomous vehicle (AV) system. We first describe the system before explaining how we apply the problem in Section 3.3 on the system.

Modular Autonomous Vehicle Stack

We consider a modular AV stack that is based on the DiffStack system described in [32]. This system is composed of (1) a predictor f_1 for forecasting the trajectory of uncontrolled agents in the environment, (2) a sampling-based planner f_2 and (3) a model-predictive controller (MPC) f_3. The ego vehicle controlled by the system operates in real road environments from the nuScenes dataset [9]. We focus our analysis on the uncertainty at the output of the trajectory predictor and as used by the planner.

In our AV stack, the trajectory predictor is the Trajectron++ model [43], a state-of-the-art Conditional Variational Autoencoder. Its output is a Gaussian Mixture Model (GMM) whose modes are the predicted agent future states. Due to its output structure, we can capture the prediction uncertainty of Trajectron++ via certain heuristics. For example, the variances of each Gaussian in the mixture are reasonable measures of uncertainty. The sampling-based planner is designed with an internal cost function that balances the objectives of self-driving:

$$C^{\text{int}} = C_{\text{coll}}(y_1) + C_{\text{goal}} + C_{l\perp} + C_{l\measuredangle} + C_u \tag{3.13}$$

where the terms denote the cost of collision, distance to goal, lane lateral deviation, lane heading deviation, and control effort, respectively. The planner proposes candidate plans by sampling dynamically feasible splines and selecting the plan with the lowest cost. In this work, we modify this cost function to include an uncertainty measure amenable to our analysis. Finally, the MPC controller [1] solves an iterative linear quadratic regulator (iLQR) problem to further optimize planner's best candidate plan under a quadratic approximation of the same cost function. Without loss of generality, following [32], the system produces predictions only for the uncontrolled agent closest to the ego vehicle and uses the ground truth future states of other agents.

We propose numerous designs that explicitly incorporate uncertainty into the modular AV pipeline. We use two uncertainty heuristics naturally derived from the GMM output of Trajectron++: (i) the entropy of the most-likely Gaussian, denoted as σ_{ML}, and (ii) the entropy of the categorical distribution over GMM modes, denoted as σ_K. We normalize the uncertainty to lie in $[0, 1]$. Given imperfect predictions, the planner should utilize the uncertainty measure to re-weight each of the cost terms in Equation (3.13) to ensure the ego agent's safety. In the face of high uncertainty, one such design is to be extra cautious in avoiding the predicted agent by inflating the safety margin, leading to the uncertainty-aware cost function:

$$C^{\text{int}}_{\text{avoid}} = e^{\alpha\sigma_1} C_{\text{coll}}(y_1) + C_{\text{goal}} + C_{l\perp} + C_{l\measuredangle} + C_u \tag{3.14}$$

where α is a scaling factor. An alternative design is to encourage lane keeping (often considered the "default" safe driving behavior), deprioritize goal-reaching, and give less trust to the quality of the collision cost, as shown in the following:

$$C^{\text{int}}_{\text{lane}} = \sigma_1(C_{\text{coll}}(y_1) + C_{\text{goal}}) + \frac{1}{1-\sigma_1}(C_{l\perp} + C_{l\measuredangle} + C_u) \tag{3.15}$$

Robustness Analysis Results

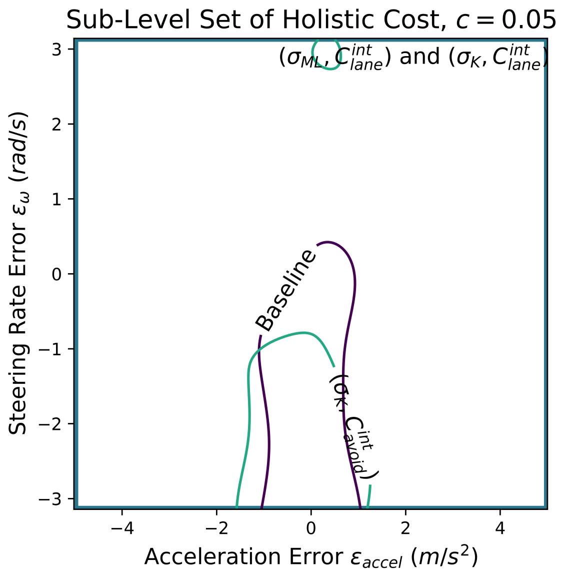
We compare the performance and robustness of five AV stacks, namely the baseline system without the use of uncertainty and the four uncertainty-aware AV stack combinations $(\sigma_{ML}, C^{\text{int}}_{\text{avoid}})$, $(\sigma_{ML}, C^{\text{int}}_{\text{lane}})$, $(\sigma_K, C^{\text{int}}_{\text{avoid}})$, $(\sigma_K, C^{\text{int}}_{\text{lane}})$.

We obtain the sub-level sets in the space of input errors. More concretely, we add input error ϵ_x by adding acceleration error $\epsilon_{\text{accel}} \in [-5, 5]$ and steering rate error $\epsilon_\omega \in [-\pi, \pi]$ to the final step of the closest agent's history states and recomputing the history trajectory to ensure dynamical feasibility of $x + \epsilon_x$. We choose two different costs to evaluate the overall system. The holistic cost described in Equation (3.13) offers a holistic assessment of quality of the output planned trajectories. The safety cost is defined as the negative of the minimum distance between the output ego plans and any other agent, across all agents and all steps in the planning horizon. It offers an interpretable and safety-oriented view on the systems.

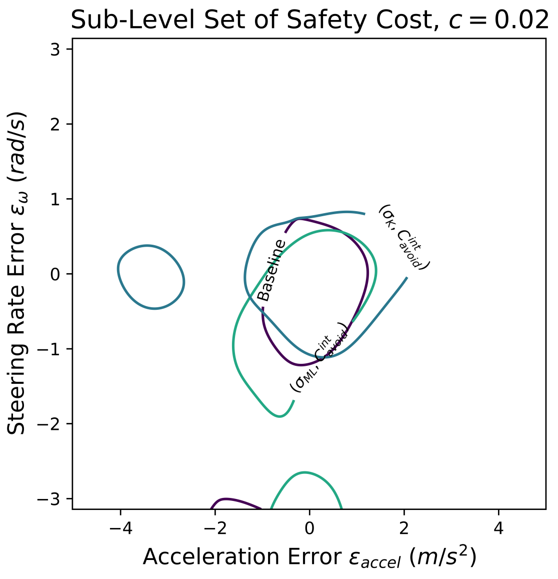
The systems are evaluated on a subset of the nuScenes dataset that is unseen by Trajectron++ at training time. Such evaluation is computationally costly due to the size of the dataset, so computing the sub-level set exactly is intractable. Hence, we employ Bayesian optimization, where we allow 15 random warm-start evaluations for initial GP fitting and 20 more evaluations sampled at locations determined by the MILE acquisition function. Cost thresholds c in Equation (3.6) are modeling choices, which are set to $c = 0.05$ for holistic cost and $c = 0.02$ for safety cost. For every cost metric, we consider the cost itself as a measurement of performance (without input error) and the size of the sub-level set as a measurement of robustness. We report cost values relative to a version of the system with access to ground truth agent future states and no input errors. (More precisely, cost of our designs subtracted by cost of the GT version). The size of the sub-level sets is the proportion of the input error space. Quantitative results are reported in Table 3.1. The sub-level sets visualizations are shown in Fig. 3.4.

Table 3.1: Sub-Level Set Sizes and Costs for AV Designs

System	Holistic Cost		Safety Cost	
	Size of Sub-Level Set ↑	Cost $\times 10^{-2}$ ↓	Size of Sub-Level Set ↑	Cost $\times 10^{-2}$ ↓
GT Prediction	—	0	—	0
Baseline	0.092	1.44	0.036	0.46
$(\sigma_{ML}, C^{\text{int}}_{\text{avoid}})$	0.0	-0.76	0.036	0.38
$(\sigma_{ML}, C^{\text{int}}_{\text{lane}})$	**1.0**	-0.49	0.0	7.77
$(\sigma_K, C^{\text{int}}_{\text{avoid}})$	0.084	2.90	**0.073**	-2.63
$(\sigma_K, C^{\text{int}}_{\text{lane}})$	**1.0**	-0.19	0.0	8.28

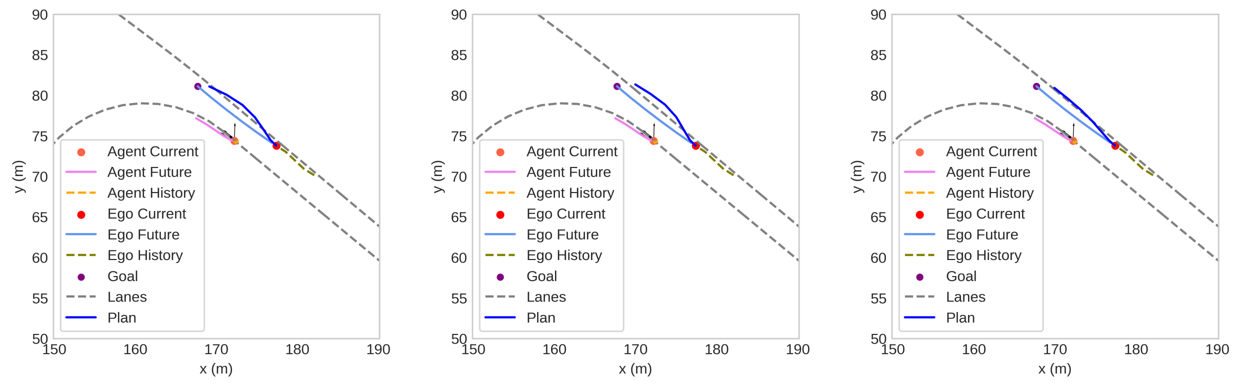


(a)



(b)

Figure 3.4: Our analysis shows that incorporating uncertainty yields larger sub-level sets and thus more robust AV stacks. Sub-level sets of size 0 are not shown. Sub-level sets of size 1 are shown as a rectangle occupying the whole space.



(a) Baseline system. (b) Using $C^{\text{int}}_{\text{avoid}}$. (c) Using $C^{\text{int}}_{\text{lane}}$.

Figure 3.5: A visualization of the qualitative differences between the three planner designs in the presence of input error. The proposed plan in 3.5b creates a larger buffer between the ego vehicle and the nearby agent than that of the baseline 3.5a. The plan in 3.5c most closely adheres to the lane center.

Discussion

For both costs, we've found one (or more) uncertainty-aware designs that outperforms the baseline system in terms of the size of the sub-level sets. The quantitative results demonstrate the efficacy of explicitly modeling and utilizing uncertainty at the interconnections between system components. Among our designs, the systems with the best sub-level set size also tend to have superior costs compared to the baseline. For example, $(\sigma_{ML}, C^{\text{int}}_{\text{lane}})$ has large sub-level set and lower cost than the baseline in terms of holistic cost, and similarly for $(\sigma_K, C^{\text{int}}_{\text{avoid}})$ in terms of safety cost. It suggests system robustness and system performance are not competing objectives. We do not necessarily sacrifice performance to gain robustness, unlike the typical findings in the field of adversarial machine learning [61]. The robustness analysis also provides quantitative evidence to the earlier claim that lane-keeping is the "default" safe driving behavior. With the excellent sub-level set size (1.0) of designs involving $C^{\text{int}}_{\text{lane}}$ on the holistic cost, it shows lane-keeping is an effective fallback controller in the presence of large input errors.

The sub-level sets also reveal interesting properties of the system and the dataset that might not be obvious or intuitive. Observe that both sub-level sets in Fig. 3.4 tend to have volume at $\epsilon_\omega < 0$, but not at $\epsilon_\omega > 0$. We attribute the tendency to two factors. First, a negative steering rate error does not degrade the prediction quality of the trajectory forecaster as much as a positive one. We found that the negative log-likelihood loss of Trajectron++ is consistently lower at a negative ϵ_ω than a positive ϵ_ω for the same $|\epsilon_\omega|$. We conjecture that such behavior in the predictor model is caused by bias in the training data distribution. Second, the corrupted predictions caused by negative steering rate errors are less detrimental to the ego agent's planner than positive errors. We empirically verify that $\epsilon_\omega < 0$ tend to increase the distance between the current (perturbed) position of the predicted agent and the goal position of the ego agent, which means the corrupted predictions are less likely to intersect with ego's planned path. Beyond measuring robustness, the sub-level sets provide insights about the system and the dataset previously unknown to the system designers, which might aid future diagnostics of the system.

3.5 The Calibration Perspective

We move onto the calibration perspective of modules that provide probability estimates. We demonstrate the problem described in Section 3.3 in the aviation domain.

Runway Incursion Detection System for Aircraft

We demonstrate our approach on a pilot warning system that detects runway intrusions during landing. Runway incursions have proven to be a non-trivial hazard in recent times [47], so designing a robust system for detecting incursions is highly important. We approach this problem using a Boeing aircraft fitted with cameras, which we utilize for learning-enabled detection.

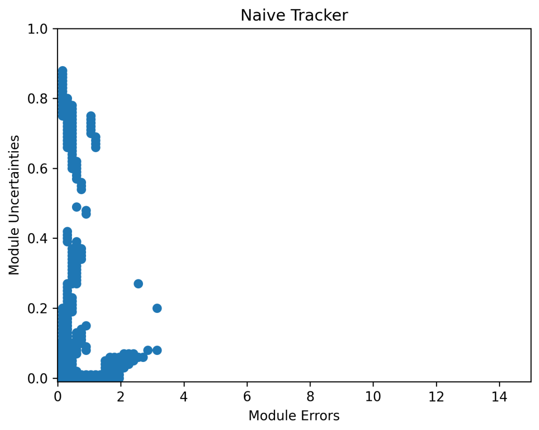
The system is comprised of three modules: a detector f_1, a tracker f_2, and an advisory component f_3 that issues a warning to the pilot. The detector is provided images of the runway, and, using a fully-convolutional one-stage detector [48], marks vehicles on the runway with a bounding box. These bounding boxes not only encode spatial information about the object, but also include a confidence score that is between 0 and 1. A higher confidence score can be perceived as a higher likelihood of an object existing. This provides a natural choice of an uncertainty metric. The tracker receives the bounding-boxed detections, along with their associated uncertainty scores, and uses a classical Kalman filtering-based approach to track objects. Additionally, we implement the tracker to constantly monitor the probability of an object's existence, using Wald's sequential likelihood ratio test [55]. Given the input confidence score, we apply a model to liken this confidence score to the probability a track exists, to fit the framework of the sequential probability ratio test (SPRT). Finally, we provide the collection of tracks to the advisory module. This module collects all tracks that appear to intrude on the runway, and it calculates the probability that at least one of these tracks exists, using the existence probabilities derived from Wald's SPRT previously. If the total probability of an intrusion exceeds a given threshold, we advise the pilot not to land.

Calibration is especially important in this safety-critical probabilistic system. Well-calibrated probability estimates provide interpretable results, improving pilots' safety assessment on the fly. It also helps to increase buy-in from aviation regulators who are especially interested in quantifying the risk of learning-based components.

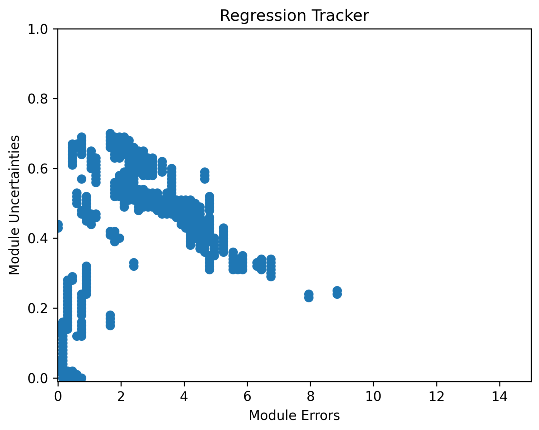
On Calibration of Uncertainty-Aware Trackers

The module of interest is the tracker f_2. We consider two uncertainty-aware tracker designs, denoted by "Naive" and "Regression". These designs are differentiated by how they map a confidence value on a detection to the existence probability of the associated track. The Naive tracker defines this ratio as simply the detection confidence divided by 1 minus the confidence. The Regression tracker, however, uses an autoregressive linear model that incorporates a rolling confidence mean and variance. Thus, both the numerator and denominator of the likelihood ratio are linear models whose weights are learned from a separate dataset.

In the context of our analysis, σ is the mean confidence score on a track; l is cross-entropy loss on the softmax distribution of the detector. Each point in Figure 3.6 represents a single track. The miscalibration threshold c is 0.4. All data are collected during real flight tests in



(a) Sub-level set size: 0.0251



(b) Sub-level set size: 0.0442

Figure 3.6: Our analysis shows that the Regression Tracker is a superior design with lower miscalibration when given large errors of the upstream detector network. Module Errors denote cross-entropy loss on the softmax distribution of the detector network. Model Uncertainties are the mean confidences on a track.

collaboration with Boeing. The structure of the data leads to a lack of coverage in parts of the $l \times \sigma$ space. Thus, we bin the space with a bin size of 0.02 and use a GP to interpolate. We only include points in the sub-level set with $\lambda = 0.9$.

Notice that the Regression Tracker obtains a larger sub-level set size, meaning it is considered relatively well-calibrated for more values of detector loss and confidences. We highlight that the Naive Tracker's level set does not have any volume when $l > 4$, while the Regression Tracker has a level set that has a non-trivial volume up to $l = 7$. Hence the Regression Tracker is more tolerant to upstream failures, which, considering its ability to maintain a sliding window of past confidences, is consistent with our expectation.

3.6 Acknowledgments

This material is based upon work supported by the DARPA Assured Autonomy Program, the NASA ULI on Safe Aviation Autonomy, and the NSF GRFP under Grant No. 2146752. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of any aforementioned organizations. Additionally, the author would like to thank James Paunicka, Blake Edwards, Jose Medina, Brandon Schwiesow, Zachary Tane, and the rest of the team at Boeing for their help and contributions. The author would also like to thank Michael H. Lim for his early contributions to this work.

Chapter 4

Conclusion and Future Directions

We presented our definition of an uncertainty-aware modularized autonomy stack, which encapsulates many modern robot system architectures involving learning-based components. We first contributed a novel self-driving pipeline design that provides rigorous safety guarantees by combining statistical tools (i.e. conformal prediction) with control theory tools (i.e. HJ reachability). We then advocate for the use of uncertainty quantification in general autonomy stacks by analyzing system robustness and calibration through level set estimation.

There are several important yet under-explored directions on the topic of probabilistic safety assurance, first of which is any-time valid guarantee. Unlike existing methods that provide miscoverage error rate between the initial and the current time, we'd like an algorithm to provide an assurance on the current prediction *regardless of* the past errors. It demands the algorithm to provide valid guarantees *any-time*, effectively disallowing it from generating trivial, degenerate sets for some time steps while still maintaining the guarantee on average. Another extension of our work is probabilistic calibration. Probabilistic calibration requires a forecaster to output distributions that are valid at *every* quantile level, not just $1 - \alpha$ level. Finally, we'd like to see practical algorithms that can operate at extreme significance levels such as 0.01% error rate. Current safety assurances of learning-based autonomous systems are typically benchmarked at the $5 - 10\%$ error level. It is still prohibitively high for safety-critical situations like aviation, where a single mistake can lead to catastrophic consequences.

We are also interested in turning our work on the analysis of general, uncertainty-aware pipelines into optimization of such pipelines. Namely, how do we devise optimization methods to maximize level sets for improved robustness and calibration? A promising direction is to develop a surrogate loss for the level set maximization problem. Another important topic is system-wide rigorous safety assurances. Current works mostly focus on providing assurances for one module in the entire pipeline. Systems composing of multiple neural networks pose a new type of safety challenge in autonomous systems.

Bibliography

[1] Brandon Amos et al. "Differentiable MPC for End-to-end Planning and Control". In: *Advances in Neural Information Processing Systems*. 2018, pp. 8299–8310.

[2] Anastasios N Angelopoulos and Stephen Bates. "A gentle introduction to conformal prediction and distribution-free uncertainty quantification". In: *arXiv preprint 2107.07511* (2021).

[3] Imanol Arrieta-Ibarra et al. "Metrics of calibration for probabilistic predictions". In: *Journal of Machine Learning Research* 23.351 (2022), pp. 1–54.

[4] Andrea Bajcsy et al. "A scalable framework for real-time multi-robot, multi-human collision avoidance". In: *2019 International Conference on Robotics and Automation*. IEEE. 2019, pp. 936–943.

[5] Somil Bansal et al. "Hamilton-jacobi reachability: A brief overview and recent advances". In: *2017 IEEE 56th Annual Conference on Decision and Control*. IEEE. 2017, pp. 2242–2253.

[6] Rina Foygel Barber et al. "Conformal prediction beyond exchangeability". In: *arXiv preprint 2202.13415* (2022).

[7] Jarosław Błasiok et al. "A unifying theory of distance from calibration". In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*. 2023, pp. 1727–1740.

[8] Dimitrios Boursinos and Xenofon Koutsoukos. "Assurance monitoring of learning-enabled cyber-physical systems using inductive conformal prediction based on distance learning". In: *AI EDAM* 35.2 (2021), pp. 251–264.

[9] Holger Caesar et al. "nuScenes: A multimodal dataset for autonomous driving". In: *Computer Vision and Pattern Recognition*. IEEE. 2020.

[10] Kaustav Chakraborty and Somil Bansal. "Discovering Closed-Loop Failures of Vision-Based Controllers Via Reachability Analysis". In: *IEEE Robotics and Automation Letters* PP (May 2023), pp. 1–8. DOI: 10.1109/LRA.2023.3258719.

[11] Bertrand Charpentier et al. "Disentangling epistemic and aleatoric uncertainty in reinforcement learning". In: *arXiv preprint 2206.01558* (2022).

[12] Yuxiao Chen et al. "Reactive motion planning with probabilistic safety guarantees". In: *Conference on Robot Learning*. PMLR. 2021, pp. 1958–1970.

[13] Anthony Corso et al. "Risk-Driven Design of Perception Systems". In: *ArXiv* abs/2205.10677 (2022).

[14] Charles Dawson and Chuchu Fan. "A Bayesian Approach to Breaking Things: Efficiently Predicting and Repairing Failure Modes via Sampling". In: *Conference on Robot Learning*. 2023.

[15] Charles Dawson and Chuchu Fan. "Certifiable Robot Design Optimization using Differentiable Programming". In: *Robotics: Science and Systems*. 2022.

[16] Alex Devonport et al. "Data-driven reachability analysis with Christoffel functions". In: *2021 60th IEEE Conference on Decision and Control*. IEEE. 2021, pp. 5067–5072.

[17] Anushri Dixit et al. "Adaptive Conformal Prediction for Motion Planning among Dynamic Agents". In: *arXiv preprint 2212.00278* (2022).

[18] Scott Ettinger et al. "Large Scale Interactive Motion Forecasting for Autonomous Driving: The Waymo Open Motion Dataset". In: *International Conference on Computer Vision*. IEEE. Oct. 2021, pp. 9710–9719.

[19] Michael Everett et al. "Reachability Analysis of Neural Feedback Loops". In: *IEEE Access* PP (2021), pp. 1–1. URL: `https://api.semanticscholar.org/CorpusID:236957330`.

[20] Shai Feldman et al. "Achieving Risk Control in Online Learning Settings". In: *arXiv preprint 2205.09095* (2023).

[21] Jaime F Fisac et al. "Reach-avoid problems with time-varying dynamics, targets and constraints". In: *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*. 2015, pp. 11–20.

[22] P. Frazier. "A Tutorial on Bayesian Optimization". In: *ArXiv* abs/1807.02811 (2018). URL: `https://api.semanticscholar.org/CorpusID:49656213`.

[23] David Fridovich-Keil et al. "Confidence-aware motion prediction for real-time collision avoidance". In: *The International Journal of Robotics Research* 39.2-3 (2020), pp. 250–265.

[24] Yarin Gal and Zoubin Ghahramani. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1050–1059.

[25] Isaac Gibbs and Emmanuel Candes. "Adaptive conformal inference under distribution shift". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 1660–1672.

[26] Junru Gu, Chen Sun, and Hang Zhao. "Densetnt: End-to-end trajectory prediction from dense goal sets". In: *International Conference on Computer Vision*. IEEE. 2021, pp. 15303–15312.

[27] Chuan Guo et al. "On calibration of modern neural networks". In: *International conference on machine learning*. PMLR. 2017, pp. 1321–1330.

[28] Dan Hendrycks and Kevin Gimpel. "A baseline for detecting misclassified and out-of-distribution examples in neural networks". In: *arXiv preprint arXiv:1610.02136* (2016).

[29] B. Ivanovic and M. Pavone. "Injecting Planning-Awareness into Prediction and Detection Evaluation". In: *IEEE Intelligent Vehicles Symposium*. 2022.

[30] Boris Ivanovic and Marco Pavone. "The Trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 2375–2384.

[31] HM Dipu Kabir et al. "Neural network-based uncertainty quantification: A survey of methodologies and applications". In: *IEEE access* 6 (2018), pp. 36218–36234.

[32] Peter Karkus et al. "DiffStack: A Differentiable and Modular Control Stack for Autonomous Vehicles". In: *6th Annual Conference on Robot Learning*. 2022.

[33] Sydney M. Katz et al. "Efficient Determination of Safety Requirements for Perception Systems". In: *Digital Avionics Systems Conference*. 2023.

[34] Roger Koenker. *Quantile regression*. Vol. 38. Cambridge university press, 2005.

[35] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. "Simple and scalable predictive uncertainty estimation using deep ensembles". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6405–6416. ISBN: 9781510860964.

[36] Lars Lindemann et al. "Safe planning in dynamic environments using conformal prediction". In: *IEEE Robotics and Automation Letters* (2023).

[37] Rachel Luo et al. "Sample-efficient safety assurances using conformal prediction". In: *Algorithmic Foundations of Robotics XV: Proceedings of the Fifteenth Workshop on the Algorithmic Foundations of Robotics*. Springer. 2022, pp. 149–169.

[38] Ali Malik et al. "Calibrated model-based deep reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 4314–4323.

[39] Anish Muthali et al. "Multi-agent reachability calibration with conformal prediction". In: *arXiv preprint arXiv:2304.00432* (2023).

[40] Kensuke Nakamura and Somil Bansal. "Online update of safety assurances using confidence-based predictions". In: *arXiv preprint 2210.01199* (2022).

[41] Jeremy Nixon et al. "Measuring calibration in deep learning." In: *CVPR workshops*. Vol. 2. 7. 2019.

[42] Janis Postels et al. "Sampling-Free Epistemic Uncertainty Estimation Using Approximated Variance Propagation". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 2931–2940.

[43] Tim Salzmann et al. "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data". In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*. Springer. 2020, pp. 683–700.

[44] Edward Schmerling. *hj_reachability*. `https://github.com/StanfordASL/hj_reachability`. 2021.

[45] Apoorva Sharma, Navid Azizan, and Marco Pavone. "Sketching Curvature for Efficient Out-of-Distribution Detection for Deep Neural Networks". In: *ArXiv* abs/2102.12567 (2021).

[46] Ingo Steinwart and Andreas Christmann. "Estimating conditional quantiles with the help of the pinball loss". In: *arXiv preprint 1102.2101* (2011).

[47] Cara Tabachnick. "What to know about the recent close calls on airport runways". In: *CBS News* (2023).

[48] Zhi Tian et al. "FCOS: Fully Convolutional One-Stage Object Detection". In: Oct. 2019, pp. 9626–9635. DOI: 10.1109/ICCV.2019.00972.

[49] Ryan J Tibshirani et al. "Conformal prediction under covariate shift". In: *Advances in Neural Information Processing Systems* 32 (2019).

[50] Renukanandan Tumu et al. "Physics Constrained Motion Prediction with Uncertainty Quantification". In: *arXiv preprint 2302.01060* (2023).

[51] Balakrishnan Varadarajan et al. "Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction". In: *International Conference on Robotics and Automation*. IEEE. 2022, pp. 7814–7821.

[52] Joseph A. Vincent and Mac Schwager. "Reachable Polyhedral Marching (RPM): An Exact Analysis Tool for Deep-Learned Control Systems". In: *ArXiv* abs/2210.08339 (2022).

[53] Eugene Vinitsky et al. "Nocturne: a scalable driving benchmark for bringing multi-agent learning one step closer to the real world". In: *arXiv preprint 2206.09889* (2022).

[54] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*. Vol. 29. Springer, 2005.

[55] A. Wald. *Sequential Analysis*. John Wiley and Sons, New York, 1947.

[56] Huajun Xi et al. "Does Confidence Calibration Help Conformal Prediction?" In: *arXiv preprint arXiv:2402.04344* (2024).

[57] Chen Xu and Yao Xie. "Conformal prediction interval for dynamic time-series". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 11559–11569.

[58] Jiayu Yao et al. "Quality of uncertainty quantification for Bayesian neural network inference". In: *arXiv preprint 1906.09686* (2019).

[59] Andrea Zanette, Junzi Zhang, and Mykel J. Kochenderfer. "Robust Super-Level Set Estimation using Gaussian Processes". In: *ECML/PKDD*. 2018.

[60] Hanli Zhang et al. "Why Change Your Controller When You Can Change Your Planner: Drag-Aware Trajectory Generation for Quadrotor Systems". In: *ArXiv* abs/2401.04960 (2024).

[61] Hongyang Zhang et al. "Theoretically principled trade-off between robustness and accuracy". In: *International conference on machine learning*. PMLR. 2019, pp. 7472–7482.