Uncertainty-Aware Planning in Bird's Eye View Representations

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Computer Science and Engineering
by Research

by

Vikrant Dewangan
2018111024
vikrant.dewangan@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
October 2023

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Uncertainty Aware Planning in Bird's Eye View Representations" by Vikrant Dewangan, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Prof. K. Madhava Krishna

To my parents and friends

Acknowledgments

I want to thank several people who played a crucial role in shaping this thesis. First and foremost, I am incredibly grateful to Dr. K. Madhava Krishna for his continuous support and guidance throughout my journey at the Robotics Research Center (RRC). I will be forever indebted to him for allowing me to work at challenging problems over the years.

I would also like to thank Dr. Arun K. Singh for his mentorship and valuable insights during my research career. It was an amazing learning experience working with his technical expertise. I am also grateful to Dr. Harikumar Kandath, Dr. Krishna Murthy Jatavallabhula, Dr. Sachin Chaudhari, and Dr. Charu Sharma, who guided me at different times in my research career. Their support and cooperation were highly crucial to shape up my career. I want to thank Dr. Kamal Karlapalem, Dr. C.V. Jawahar, Dr. Prasad Krishnan and several others who have taught useful and interesting courses during my undergraduate studies. Their courses on database systems, data visualization, machine learning, optimization techniques and linear algebra allowed me to explore these topics and obtain a stronghold. This helped me develop my research aptitude while pursuing my research on this thesis.

I would also like to use this space to thank my senior Sarthak Sharma, who motivated me, gave me guidance, and stood like a pillar of support from time to time. I would also like to thank my colleagues and juniors Aakash Aanegola and Tushar Choudhary for holding their heads high in the journey. My journey at IIIT would not have been possible without my wingmates - Pranav Kirsur, Shivaan, KV, Bhavyajeet, Tadimeti, Nomaan, Jai, and Sartak, with whom I had a great time at IIIT Hyderabad.

Finally, I would like to thank my parents and grandparents, whose blessings and constant encouragement helped me during the last several years. I would not be where I am today without their support. They have always been there to cheer me on. I am excited to continue my journey in autonomous driving.

Abstract

Autonomous driving requires accurate reasoning of the location of objects from raw sensor data. Recent end-to-end learning methods go from raw sensor data to a trajectory output via Bird's Eye View (BEV) segmentation as an interpretable intermediate representation. Motion planning over cost maps generated via Birds Eye View (BEV) segmentation has emerged as a prominent approach in autonomous driving.

However, current approaches have two critical gaps. First, the optimization process is simplistic and involves just evaluating a fixed set of trajectories over the cost map, which are not adapted based on their associated cost values. Second, the existing cost maps do not account for the uncertainty arising from noise in RGB images, BEV annotations. As a result, these approaches can struggle in challenging scenarios where there is abrupt cut-in, stopping, overtaking, and merging from neighboring vehicles.

In this thesis, we propose *UAP-BEV*, a novel approach that models the noise in Spatio-Temporal BEV predictions to create uncertainty-aware occupancy grid maps. Using queries of the distance to the closest occupied cell, we obtain a sample estimate of the collision probability of the ego-vehicle. Subsequently, our approach uses gradient-free sampling-based optimization to compute low-cost trajectories over the cost map. Notably, the sampling distribution is adapted based on the optimal cost values of the sampled trajectories. By explicitly modeling probabilistic collision avoidance in the BEV space, our approach can outperform the cost-map-based baselines in collision avoidance, route completion, time to completion, and smoothness. To further validate our method, we also show results on the real-world dataset NuScenes, where we report collision avoidance and smoothness improvements. It also outperforms other uncertainty-based methods in conservatism.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Autonomous Driving (AD) is becoming increasingly popular, and Artificial Intelligence (AI)-powered perception systems are playing a key role in making it possible. These systems use Deep Learning (DL) methods to identify objects and their properties in the environment around the vehicle, such as other vehicles, pedestrians, cyclists, and traffic signs. This information is then used to make decisions about how to safely navigate the vehicle. AI-powered perception systems are able to identify objects that are difficult to identify with traditional computer vision techniques, and they can learn from large amounts of data, which allows them to improve their performance over time.

In this chapter, we define a traditional engineering stack in the context of AD and compare with End-to-End (E2E) learning approaches. Subsequently, we expand on Bird's Eye View (BEV) representations which are popular in these E2E approaches. We then define how path planning is formulated in the BEV-based methods and the challenges associated with them. We also give an overview of Uncertainty estimation methods in DL techniques, and detail the sources uncertainty existent within BEV representations. Towards the end, we list our contributions.

1.1 Autonomous Driving Approaches

A traditional engineering stack consists of: Perception, Prediction, and Planning. These approaches picked up from the DARPA challenge, 2004. The pipeline works in a sequential fashion, with each module providing input to the next. The first module, which receives sensory information, selects relevant and physically interpretable information to pass on to the next module. This process continues until the final module, which makes a decision based on the information it has received. Each of these sub-tasks is solved sequentially. The subtasks are detailed -

- **Perception** handles detecting and tracking objects in the scene. The various objects include dynamic (vehicle, pedestrian) and static (lane, road) objects. Existing frameworks use perception modalities like LiDAR [21], Images [9, 32, 16], RADAR [19] along with high-level information such as HD-Maps, Target point. Detection involves getting an anchor head with offsets or an instance segmentation and prediction map.

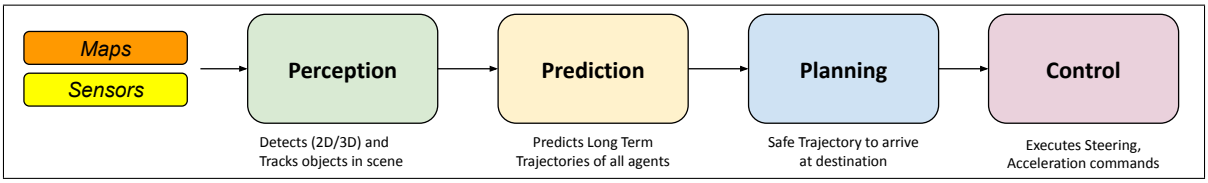


Figure 1.1 Traditional Autonomous Driving Stack: The sensors layer includes cameras, lidars, radars, and other sensors that collect data about the environment around the vehicle. The perception layer uses the data from the sensors to create a model of the environment around the vehicle. This model includes information about the location, speed, and direction of other vehicles, pedestrians, and objects. The planner layer uses the model of the environment to generate a plan for the vehicle to follow. This plan includes information about the vehicle's speed, acceleration, and steering. The control layer uses the plan from the planning layer to control the vehicle's actuators, such as the steering wheel, throttle, and brakes.

- **Prediction** involves tracking the objects/agents over time and forecasting their locations in the future. Most SOTA networks use recurrent temporal models to forecast the positions of other agents into the future.
- **Path Planning** involves designing a set of constraints/costs and sampling a set of locations feasible to output a trajectory that best satisfies those constraints. The planning section is responsible for collision avoidance. The final trajectory is fed into a PID controller, which outputs the control inputs - steering angle, brake, and throttle. Here there is a big difference in how any approach evaluates its results -
  - *Open-Loop or Offline Planning*: The controller network predicts the trajectory but does **not** move the car concerning the prediction. Instead, the Frames are recorded beforehand, and the Ground Truth is available to us, which is the human driver's trajectory.
  - *Closed-Loop or Online Planning*: The agent moves with respect to the trajectory the model provides, thus having the feedback provided back to the agent. It is a more robust test of the approach, as it involves real-time computation, performance, and adaptation to new scenarios.

  Training for most SOTA methods (apart from reinforcement learning-based approaches. [31, 7]) **always** happens in Open-loop. At test time, they are evaluated in a Closed Loop fashion by using either a simulator [10] or testing on their own real-world setup.

The modular design of this system makes it easy to understand and interpret. Each module has a specific purpose, and the output of each module is meaningful. This makes it easy for engineers to understand how the system works and to troubleshoot problems. The modular design also makes it easy to incorporate prior knowledge into the system. For example, the rules of the road, human understanding of road geometry, and agent dynamics can all be incorporated into the system. This makes the system more robust and capable of handling a wider range of situations.

1.2 End-to-End (E2E) Methods

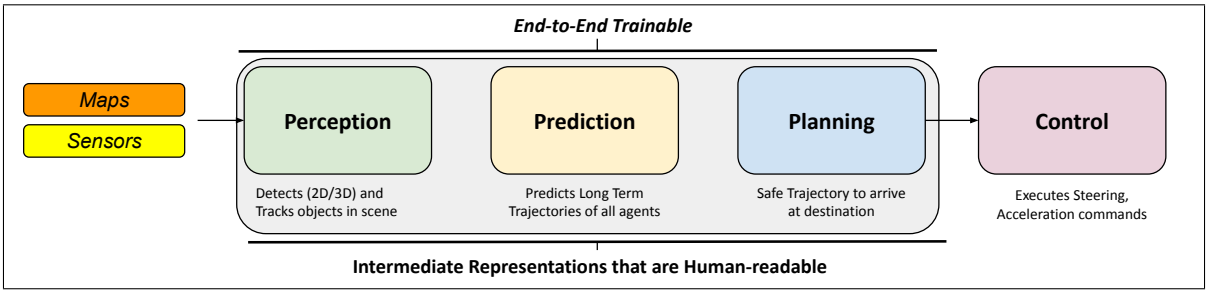


Figure 1.2 End-to-End (E2E) Learning Based Stack: The tasks perception, prediction, and planning are *jointly* learnt in an end-to-end fashion. This is generally accomplished using a Deep Neural Network (DNN) with a suitable intermediate representation. All tasks have access to raw sensor data: which lowers the cascading of errors. All tasks are aware of the end goal and have a shared computation. The intermediate representations are human interpretable which makes them easier for crucial validation. [36, 5]

The traditional approach relies on each module being able to handle a wide range of situations. However, it is difficult to design modules that can handle all possible situations, especially rare or unusual situations. This can lead to errors in one module cascading to other modules, which can cause the system to fail. The traditional approach can also lead to decreased developer productivity, as each new module or submodule requires the entire stack to be refined. This is because each module must be integrated with the other modules, and this can be a time-consuming process.

To tackle this, there have been many *data-driven* efforts that have stemmed since 2018 to solve for autonomously driving vehicles. The approach clubs various modules of the traditional autonomous driving pipeline to generate a joint probabilistic inference module which is end to end trainable, with intermediate interpretable latent representations. The end-to-end approaches learn intermediate latent representations which are trained simultaneously with the backbone encoder. This helps with verification, validation, sanity checks and testing.

In the next section, we describe the use of Bird's Eye View (BEV) representations, along with the challenges faced in using them.

1.3 Birds Eye View (BEV) Maps as Intermediate Representations

Birds Eye View (BEV) representations such as Occupancy Maps or Grids have become popular in the context of Autonomous Driving [23, 15, 36, 16]. A BEV layout is a multi-channel occupancy-grid of dimensions $H \times W \times C$, with H being the height, W being the width and C is the number of channels (semantic classes) of the BEV. These layouts are ego-centric ie. ego-vehicles are located at position $(H/2, W/2)$ facing upwards.

The task of BEV prediction lies in taking as input a sequence of past frames for P timesteps, and predicting F steps into the future. There are many existing approaches, [15, 16] which predict BEVs into the future for F timesteps. The popularity of BEV stems from its appearance-agnostic characteristics, unlike direct monocular perceptual inputs. Moreover, BEV representations can effectively tap into the vast legacy of planning frameworks tailored to leverage such representations. [27]

BEV layouts based on dense 3D LIDAR inputs are typically accurate and amenable to trajectory forecasting or layout evolution - a vital cog for motion planning in dynamic on-road scenes [6, 36]. The same, however, cannot be said for BEV estimation and evolution based on monocular perceptual inputs. Literature concerning the layout evolution of dynamic actors in a scene has been typically sparse [16, 15]. Such layouts of dynamic agents in a scene tend to be noisy and unreliable, all the more so when layout estimation and trajectory execution are interleaved in a closed-loop setting. Data-driven BEV evolution estimation typically does not consider vehicle maneuvers such as overtaking or abrupt lane changes during dataset collection. Consequently, when such maneuvers are executed, the layout estimates become unreliable and noisy and are rendered ineffective for motion planners relying on such estimates.

In the next section, we discuss the planning approach taken in these networks and describe the planning costs used along with these BEV representations.

1.4 Planning in BEV representations

1.4.1 Obtaining Cost Maps/Volumes

End-to-end networks output a *Cost Maps* or *Cost Volume*, $c_{BEV} \in \mathbb{R}^{H \times W}$ or $\mathbb{R}^{H \times W \times F}$. An additional prediction head is added to the network's final layer and thereupon, a set of template trajectories τ is used which are indexed into the cost map/volume.

Cost-map indexing:

$$c_{\tau_\mathbf{i}} = \sum_{x_j, y_j \in \tau_i} c_{BEV}(x_j, y_j) \tag{1.1}$$

Cost-Volume Indexing: Here $x_j, y_j \in \tau_i$,

$$c_{\tau_\mathbf{i}} = \sum_{t \in F} c_{BEV}(x_{j,t}, y_{j,t}, t) \tag{1.2}$$

Note that there are no annotations present for the cost map/volume. Thus the network cannot be directly trained to obtain the optimal trajectory. Here, there are 2 approaches -

- **Using a Hard Max-Margin Loss**: Adopted by Neural Motion Planner [36], this method involves predicting a cost volume $c_{BEV} \in \mathbb{R}^{H \times W \times F}$. Given the ground-truth trajectory $\bar{\tau} = \{(\bar{x}_t, \bar{y}_t)\}$ for the next F timesteps, the objective is to minimize the costs obtained by indexing the ground

truth trajectory in the cost volume while keeping the worst-performing trajectory in the template set at higher costs.

$$\mathcal{L} = \sum_{(\bar{x}_t, \bar{y}_t)} \left(\max_{\tau_i \in \tau} \sum_{t=1}^{F} \left[c_{BEV}(\tau_i) - c_{BEV}(\bar{\tau}) + d(\tau_i, \bar{\tau}) \right]_+ \right) \tag{1.3}$$

Note that $[]_+$ is a Rectified Linear Unit (ReLU) function. The term $d(\tau_i, \bar{\tau})$ calculates the distance between the ground truth trajectory so that trajectories with higher distances have higher costs.

- **Using a Soft Cross-Entropy Loss**: Used by Lift-Splat-Shoot [23], this approach predicts a distribution over the set of template trajectories τ, instead of using the hard loss.

$$p(\tau|O_t) = \frac{exp\left(- \sum_{x_k, y_k \in \tau_i} c_{BEV}(x_k, y_k) \right)}{\sum_{\tau_i \in \tau} exp\left(- \sum_{x_k, y_k \in \tau_i} c_{BEV}(x_k, y_k) \right)} \tag{1.4}$$

From here on, a cross-entropy loss is used to train the network.

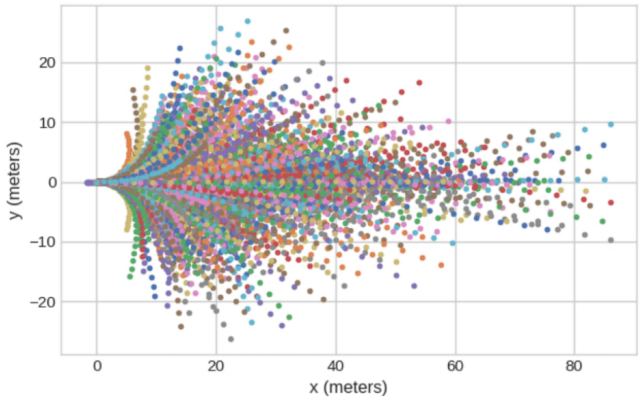


Figure 1.3 Lift-Splat-Shoot [23] Template Trajectories: These trajectories are used to "shoot" through the cost map obtained by the Neural Network. The optimal trajectory is fed into the controller.

1.4.2 Custom Costs

Instead of learning the Cost Map, many approaches design custom cost functions, to obtain a BEV cost c_{BEV}. They add the component of vehicle, road and lane geometry, along with lane rules into the cost function, and are used in conjunction with the learned costs to rank the trajectory set.

1. **Collision**: Checks if the trajectories have any spatio-temporal overlap with predicted occupancies of other agents.Is zero only if there is a safe margin around the obstacles.

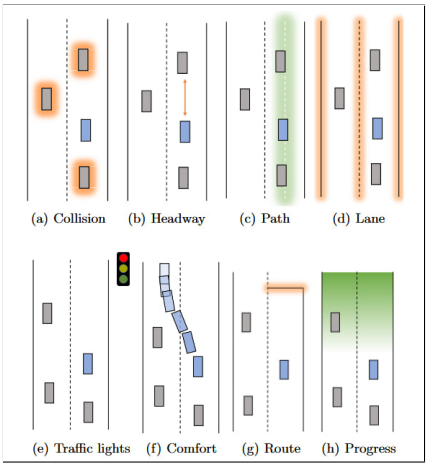


Figure 1.4 Costs in BEV-based Planning: The different types of costs associated pertaining to BEV-based planning in autonomous driving, adopted from [26]. These different costs are added together with different weights to obtain the final cost.

2. **Headway**: Checks if there is sufficient headway so that when the ego-vehicle applies the brakes, it is able to avoid collision and come to a stop. This is computed using the longitudinal distance between the agent and the leading vehicle.

3. **Path**: Checks if the ego-vehicle adheres to the path (centerline). This cost is computed as the lateral distance to the centerline of the lane.

4. **Lane**: Checks if the ego-vehicle does not violate lane boundaries. This is computed as the number of times the ego-vehicle overrides the lane boundary prediction

5. **Traffic Lights**: Checks if the ego-vehicle follows the traffic light procedures. It also checks if the ego-vehicle is within the speed limit and if any stop sign is violated.

6. **Comfort**: Checks if there is a jerk or rapid change in acceleration. Trajectories that are aggressive or perform abrupt steering are discarded with the inclusion of this cost.

7. **Route**: Checks the number of lane changes required to reach the destination. The number of lane changes that are above the optimal desired limit are penalized.

8. **Progress**: Checks if the trajectory is not moving too far from the desired route.

In the next chapter, we list some of the challenges of the existing BEV-based costs. We follow it up with a discussion on uncertainty quantification techniques, and provide a complete picture with respect to the existing approaches.

1.4.3 Core Challenges with BEV-based planning

The core challenges associated with the existing BEV-based local trajectory planning can be summarized as follows:

- The learnt c_{BEV} obtained using large neural networks with complicated architectures [16, 23, 36], makes it challenging to apply gradient-based optimization on the planning costs. Existing works like [36] follow a gradient-free approach where trajectories are sampled from a distribution, and then the cost (5.1a) is evaluated on them. The lowest cost trajectory is chosen as the optimal one. This process represents a single iteration of a full-blown sampling-based optimizer like CEM [2] or Model Predictive Path Integral [34]. Thus, existing works do not leverage the improvements achieved by adapting the sampling distribution online.

- Existing work like [16, 23, 36] rely purely on c_{BEV} to compute optimal driving behavior. However, the existing c_{BEV} is oblivious to the underlying uncertainty stemming from noisy sensors. We show in later chapters that such inadequacy reduces their performance in a closed-loop setting.

In the next section, we discuss on uncertainty – quantification techniques in deep networks. We provide an overview on the possible sources of uncertainty in E2E approaches in Autonomous Driving and the type that we are resolving.

1.5 Uncertainty: An Overview

Uncertainty quantification is essential in computer vision applications, especially in safety-critical applications like autonomous driving, aerial mobility, and indoor navigation. As more and more approaches adopt deep learning, it becomes necessary to quantify the uncertainty present in these models and tackle them. For many of these approaches, calibration becomes a crucial component alongside their performance in their field.

This section details the different uncertainty quantification techniques applied to semantic segmentation and object detection. It then reasons the sources of uncertainty that can occur in monocular BEV representations as well as causes. Then, we detail some existing approaches to deal with uncertainty in BEV representations.

1.5.1 Introduction

Uncertainty estimation in Deep Learning approaches is broadly divided into 2 types - *Regression* (Depth/Object Detection) and *Classification* (Semantic Segmentation). Most modern approaches [15, 17] use *Bayesian Neural Networks* [17] to estimate the uncertainty. These networks are fitted to predict the posterior over the output instead of a single estimate. Within this paradigm, there exist 2 main types of uncertainty -

- *Aleatoric Uncertainty*: This is also called *data/statistical uncertainty* - and refers to the true noise present already in the data samples. This is generally caused by noisy inputs - the noise can be from sensors, or tracking noise. This type of uncertainty comes with the data, and **cannot** be thrown away with more data.

- *Epistemic Uncertainty* Also called *model/systematic uncertainty*, this accounts for uncertainty in model parameters. This can be reduced by incorporating more training data into the process. This can also occur if the model weights are not optimized correctly.

The above 2 terms combined are termed as *predictive uncertainty*. Earlier works [17] have tried to reason about the aleatoric uncertainty by corrupting their logits with Monte-Carlo Sampling and attenuating their loss (which is cross entropy for classification tasks). Given a set of model weights W and the neural network predicts a set of unaries f_i per pixel i, which is corrupted with the per-pixel variance predicted by the network σ_i.

$$\hat{\mathbf{x}}_{i,t} = \mathbf{f}_i^{\mathbf{W}} + \sigma_i^{\mathbf{w}} \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, I) \tag{1.5}$$

The corrupted vector is then squished with Monte Carlo sampling. Assume T samples, the loss function becomes -

$$\mathcal{L}_x = \sum_i \log \frac{1}{T} \sum_t \exp \left(\hat{x}_{i,t,c} - \log \sum_{c'} \exp \hat{x}_{i,t,c'} \right) \tag{1.6}$$

They combine the above with epistemic uncertainty estimation by placing a distribution over the neural network weights W. The Neural Network then becomes a *Bayesian Neural Network*.

1.5.2 Confidence Calibration

Confidence calibration is defined as the ability of some model to provide an accurate probability of correctness for any of its predictions. Any deep learning model is set to be well-calibrated if the confidence associated with its prediction matches the probability of the event happening. In other words, if a neural network predicts that tomorrow there is a chance of rain with a confidence of 0.2, this prediction should have a 20% chance of being correct if the neural network is appropriately calibrated. Most modern neural networks are uncalibrated [12]. Calculating the Expected Calibration Error (ECE) is a suitable method of measuring calibration. For a set of bins B_n, it can be calculated as

$$ECE = \sum_{n=1}^{N} \frac{1}{n} \left| \sum_{i \in B_n} \left(\mathbf{I}(\hat{y}_i == y_i) - \hat{p}_i \right) \right| \tag{1.7}$$

Here $\mathbf{I}$ is the indicator function denoting accuracy.

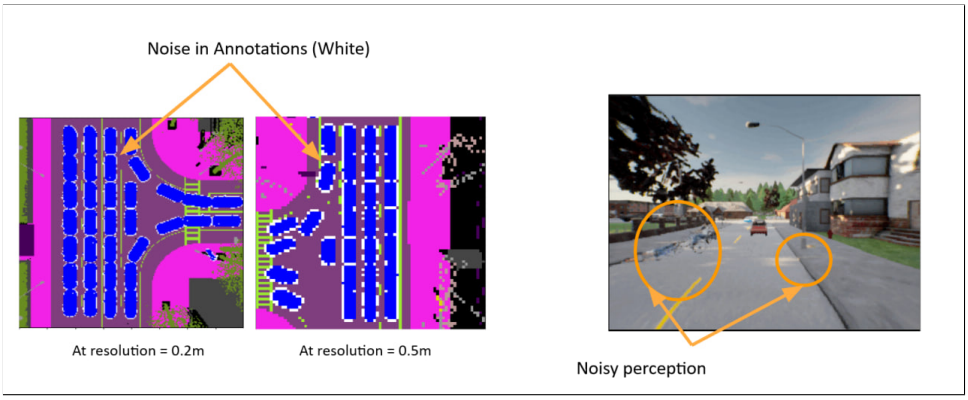


Figure 1.5 Sources of Uncertainty: A visualization of the uncertainty present due to Noisy BEV Annotations and RGB Sensor inputs. The error in BEV annotations at displayed at 2 resolutions - 0.2m and 0.5m.

1.5.3 Sources of Uncertainty in Monocular Representations

Monocular images are susceptible to noise and small changes in intensity, especially in a closed-loop setting. Figure 1.5 denotes common sources of uncertainty present. Any voxel-based compression is noisy due to the inherent loss of data. This is effectively demonstrated in the figure, where we notice more white areas of uncertainty at higher resolution (0.5m). Significant noise is present in the RGB images, which causes the performance to degrade. Due to the above causes of uncertainty, we are motivated to assume that the underlying uncertainty distribution is non-parametric.

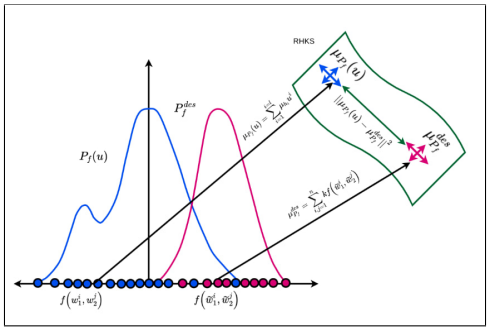


Figure 1.6 Mapping to RKHS: The similarity between any 2 distributions w_1 and w_2 can be expressed by transporting them into another space (RKHS), and computing the distance there: $||\mu_{P_f}(u) - \mu_{P_f}^{des}||$. The non-parametric assumption allows us to perform this transformation.

1.5.4 Non-parametric Assumption

We sample from the error distribution, which is non-parametric in nature and transform the samples into another space (the Reproducing Kernel Hilbert Spaces) and compute the error. Thereupon, we

compute the Maximum Mean Deviation, which represents the distance between observed and desired distributions.

1.6 Contributions

We propose **UAP-BEV**, a novel Uncertainty Aware Planning in the BEV space that can handle non-parametric noise inherent to data-driven BEV estimation of dynamic actors in a scene. Our algorithm in itself consists of two parts. In the first part, we sample the BEV representations derived from monocular cameras and obtain an uncertainty-aware estimate of the distance to the closest obstacle. The closest-distance estimate is then subsequently mapped to the probability of collision between the ego and the neighboring vehicles using the concept of distribution embedding in Reproducing Kernel Hilbert Space (RKHS). In the second part, we use a custom sampling-based optimization to compute trajectories that minimize collision avoidance probability while producing smooth trajectories. Our optimizer's novelty stems from using a projection operation that pushes the sampled trajectories toward constraint satisfaction before evaluating their costs. The constraints stem from the velocity, acceleration bounds, and barrier functions [35] accounting for lane adherence and distance-keeping with the leading vehicles based on the BEV predictions of the neighboring vehicles.

The main contributions of this thesis are as follows -

1. We propose for the first time in literature to the best of our knowledge, an Uncertainty Augmentation pipeline for noisy BEV layouts obtained through data-driven methods with monocular perceptual inputs.

2. Our approach includes a novel sampling-based optimizer that can efficiently minimize the BEV-derived costs. We show significant performance enhancements in a diverse set of metrics over SOTA prior methods [23],[16] that do not consider BEV noise in simulation experiments conducted on a number of CARLA towns over a diverse set of trajectory metrics. We also show through ablations the role of barrier functions, that when utilized along with the collision probability estimates reduce collision rates to zero despite noisy BEV estimates.

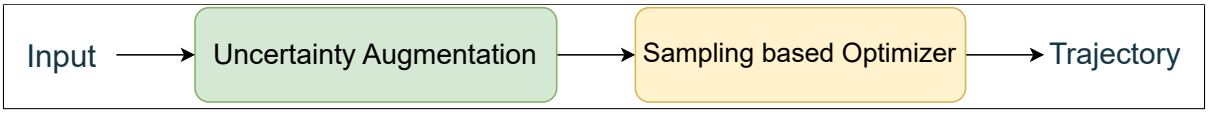


Figure 1.7 Flow Chart: The chart gives an overview of our approach. Uncertainty augmentation is detailed at chapter 4, whereas our Sampling-based Optimizer is detailed at chapter 5. Experiments and Results are discussed at Chapter 6.

1.7 Thesis Organization

The remaining parts of this thesis are organized as follows. Chapter 2 briefly describes related work for our problem setting. Chapter 3 sets up the background for using BEV representations. Chapter 4 details the process of augmenting uncertainty in BEV representations. Chapter 5 will follow with the description of the Sampling-based optimizer. Chapter 6 will contain the experiments performed and subsequent results obtained. Chapter 7 will contain the conclusive remarks and associated publications regarding this thesis.

Chapter 2

Related Work

2.1 End-to-End (E2E) Learning Based Approaches

From a modular architecture comprising of a cascade of task-specific blocks (sensor fusion, perception, planning, and control), the autonomous driving stack has evolved to an E2E system [9, 23, 16, 5, 26, 36] that learns to generate driving behaviors from sensor inputs like LiDARs and cameras.

Works like [5, 26, 36] take in voxelized LiDAR point cloud as input and use 3D object detection [36] or semantic BEV generation [26, 5] as auxiliary tasks. Learning pipelines proposed in [9, 23, 16] take in surrounding monocular images as input to reason about the semantic BEV representation. Approaches like [5, 26, 36, 23, 16] model driving behavior by using classification loss or max-margin loss on a set of template trajectories (usually from recorded driving behaviors), where learned imitation behavior closest to the ground truth trajectory is encouraged. [9] predicts an offset vector towards a target waypoint which is converted to motion commands by the lower-level controllers. In [8], the authors train an expert agent on the environment's state, which is then used in imitation learning to train for driving given vision sensor inputs. However, such E2E methods do not account explicitly for uncertainty in their BEV outputs that can result in suboptimal performance of their planner under the duress of such noise, as shown in Chapter 7.

2.2 Uncertainty In Bird's Eye View Representations

There can be multiple sources of *aleatoric uncertainty* in BEV tasks, such as the noise in BEV annotations, intrinsic, extrinsic, and input RGB. Existing methods like FIERY[15] use a Bayesian Neural Network, similar to [17], to estimate the uncertainty associated. However, these approaches suffer from calibration errors. Also, it is unclear if the uncertainties present in sensor inputs (RGB Cameras, GPS) are translated well into BEV space. This approach also suffers from the problem of noisy annotations in BEV space. Approaches like [17, 16, 15, 18] have aimed at quantifying uncertainties in perspective tasks like depth regression[18, 17], per-pixel semantic segmentation[18, 17]. In BEV space, few ([16, 15]) have addressed the issue of noise. The authors of [16, 15] quantify uncertainty by weighing each task's

loss by its *homoscedastic uncertainty*, taking inspiration from multi-task setting in [18]. However, it is unclear if the *aleatoric uncertainty* present in the RGB space can translate well into the BEV space. Further, it is unclear if [18] can manage the noisy annotations in the BEV space. The authors of [20] fit a 2D Gaussian to every object cluster. However, this approach leads to conservatism in driving behavior.

While parameterizing the future distribution of agents in BEV, in [15], the authors of Calibrated Perception Uncertainty [20] fit Gaussian Mixture Models to the entropy of the prediction, visualized in 2.1. In [16], the authors model the future distribution as either a Bernoulli or Gaussian distribution.

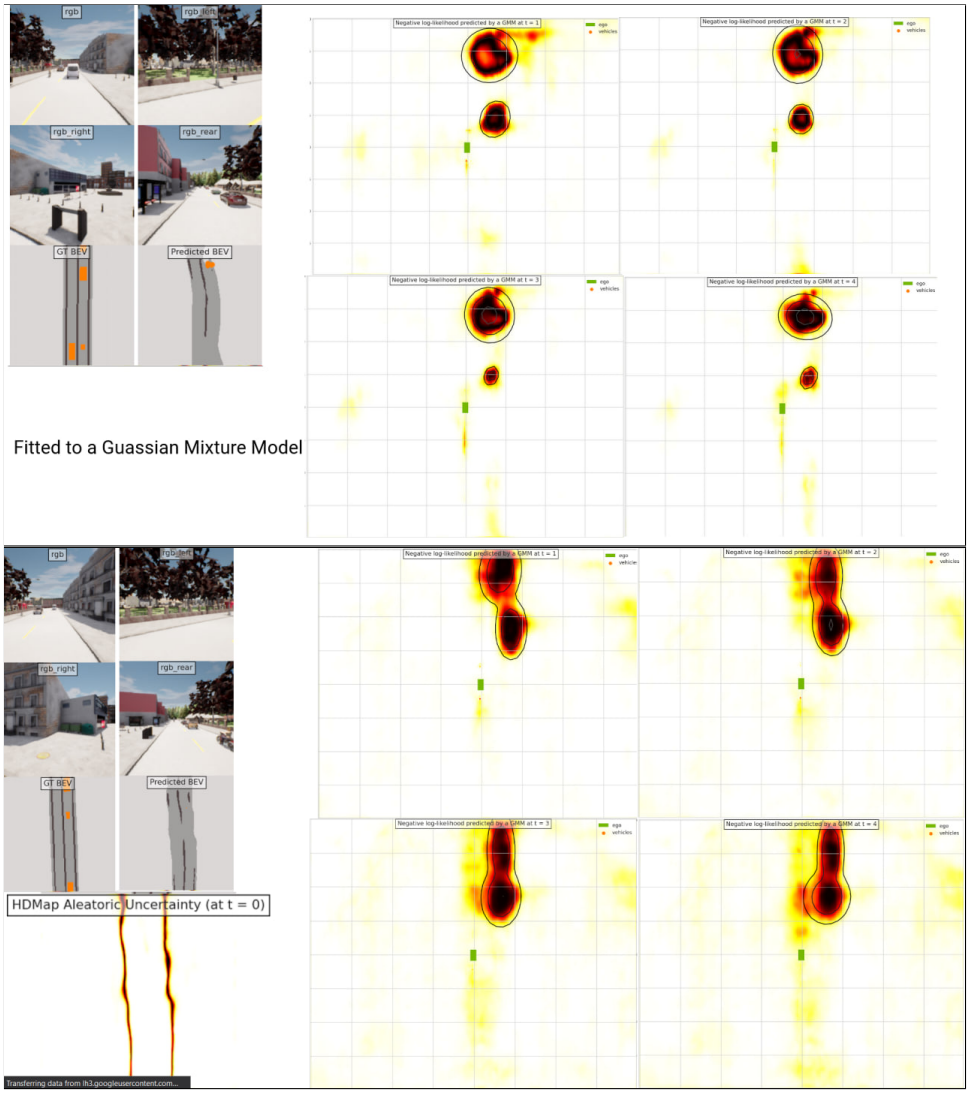


Figure 2.1 ST-P3 Gaussian: Calibrated Perception Uncertainty [20] fits a 2D Gaussian Mixture Model (GMM) to the entropy of the associated prediction. The concentric lines denote the 2D Gaussian ellipses fitted onto the entropy of the model.

Based on the detected objects, they formulate presence, location, shape, trajectory, and undetected-object-ahead uncertainty. They also calibrate their uncertainty and report the confidence calibration

curve. However, they have not associated planning with the above uncertainties, and it is unclear how an end-to-end system may utilize their approach for uncertainty-aware planning.

2.3 Non-parametric Uncertainty Estimation Methods

The above methods must perform more accurate uncertainty reasoning in the BEV and couple it to a planner. Modeling the error in observations as a non-parametric distribution is a popular way to deal with the uncertainty in the inputs and annotations and plan with it. Along this line, authors of [14] have proposed Maximum Mean Discrepancy (MMD) in Reproducing Kernel Hilbert Spaces (RKHS) to estimate collision probability conditioned on the sensor noise. Their gradient-free Cross Entropy Minimization (CEM) and reduced set allowed them to achieve collision avoidance, smoothness, and real-time performance improvements. [13] adopts the same approach on plane segmentation tasks needed for quadrotor navigation in urban settings. Our work extends [14], [13] to dynamic and uncertain autonomous driving settings where monocular RGB images are the primary sensing modality.

[14]/ [13] model uncertainty over Euclidean Signed Distance Field (ESDF)/Cuboid representations derived from RGBD cameras in static voxel grids. In contrast, our approach models uncertainty over BEV predictions using monocular RGB image inputs in challenging driving scenarios. It thus adapts to dynamic actors and eliminates the need for depth cameras. Compared to [13], we rely on a single distance-based estimate instead of complex four plane parameters to estimate the uncertainty. Moreover, compared to [14, 13], we parallelize the cost evaluation, particularly the collision probability estimates, over GPUs.

Both these approaches use a voxelized representation of the world, similar to our BEV setting, and assume noisy perception. However, they do not place dynamic actors in their scenes and model uncertainty only over static voxel grids. With our dynamic occupancy grid prediction module, we model uncertainty into the future for our use case and show results in challenging scenarios.

In the next chapter, we give a background of Frenet frame planning and trajectory sampling and the benefits of dense sampling with the existing sampling method.

Chapter 3

Background

3.1 Problem Formulation

Assume the availability of monocular images $\mathbf{X_k^n} \in \mathbb{R}^{H \times W \times 3}$ from surround N camera setup of the current timestep k_0 for the previous P steps $k \in \{k_0 - P, \dots k_0\}$ and a route $\mathcal{R}$ in the form of a reference lane centerline, the task is to predict BEV representations $\mathbf{O_k} \in \mathbb{R}^{H \times W}$, F frames into the future, $k \in \{k_0, k_1, \dots k_F\}$, to translate them into a cost map c_{BEV} and perform trajectory planning to guide the ego-vehicle without collision along the centerline.

In this chapter, we describe our Frenet-frame planning formalization as well as existing trajectory sampling techniques. We then compare them against our sampling techniques and draw conclusions.

3.2 Frenet Frame Trajectory Parametrization

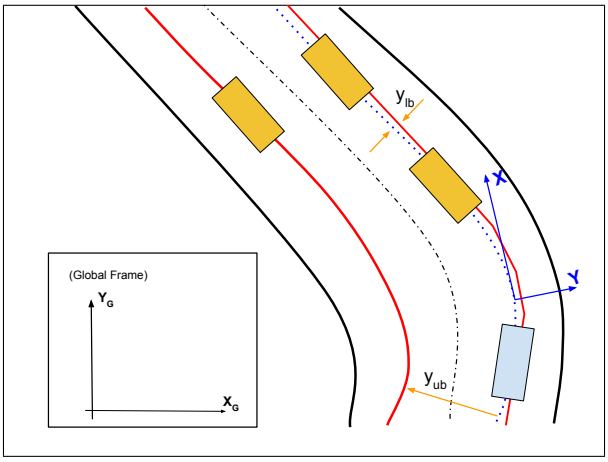


Figure 3.1 Frenet-Frame Visualization: A representation of the Frenet frame (blue axes) with respect to the road and global axes with the route (dotted blue), lane boundary (red), agent (blue), and other agents (yellow) indications.

As is typical of autonomous driving, trajectory planning happens in the road-aligned frame known as the Frenet frame. This frame allows one to treat curved roads as one with straight-line geometry. Additionally, the longitudinal and lateral motions of the vehicle will always be aligned with the X and Y axes of the Frenet frame, respectively, simplifying the optimization framework. The x and y trajectories of the ego-vehicle in the Frenet frame can be parametrized in the following form, where k_n represents the end step of the planning horizon. Note that the planning horizon need not be the same as the prediction horizon of the BEV.

$$\begin{bmatrix} x[k_0] & \dots & x[k_n] \end{bmatrix}^T = \mathbf{W}\mathbf{c}_x, \begin{bmatrix} y[k_0] & \dots & y[k_n] \end{bmatrix}^T = \mathbf{W}\mathbf{c}_y, \tag{3.1}$$

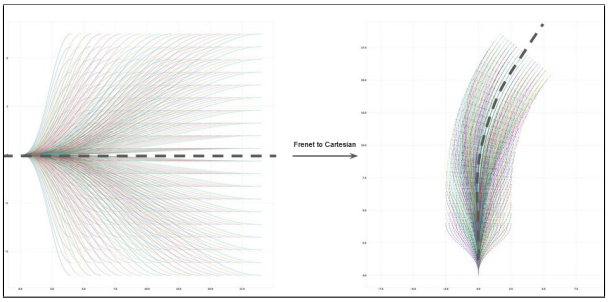
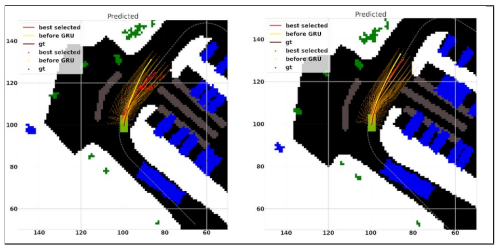


Figure 3.2 Sampled Trajectories in the Frenet Frame: The trajectories are sampled first in the Cartesian frame, after which a Cartesian-to-Frenet transformation is applied to obtain the trajectory set in the Frenet frame.

[a] 

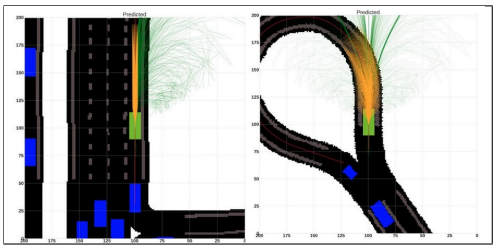
[b] 

Figure 3.3 Frenet Projected Samples in NuScenes and CARLA: Examples of projected frenet samples (orange) vs Clothoid samples (green) in (a) NuScenes and (b) CARLA
. Frenet samples are road-aligned and follow the lane geometry much better than the Clothoid samples.

3.3 Sampling Techniques

In this section, we describe the two main methods of sampling in current literature - Clothoid-based. We then detail a comparison between the methods, and reason the usage of Frenet-frame sampling over Clothoid-based Sampling.

3.3.1 Clothoid-based Sampling

Clothoid curves [28] are popular family of curves used for planning. Used in Autonomous Driving by [36, 6], the curve can be adjusted to the driving speed limit, curvature, and acceleration limits. The curvature κ of a point on this curve is proportional to its distance c, along the curve.

3.3.2 Comparison with Frenet-based Sampling

In this section, we justify the usage of frenet frame sampling along with the removal of Gated Recurrent Unit (GRU) present in ST-P3. We shall use the term "Original" to represent the Clothoid samples and "Dense" for denoting the frenet frame samples. The cost analysis of dense vs clothoidal trajectory sets has been written in 3.1 visualized in 3.5.

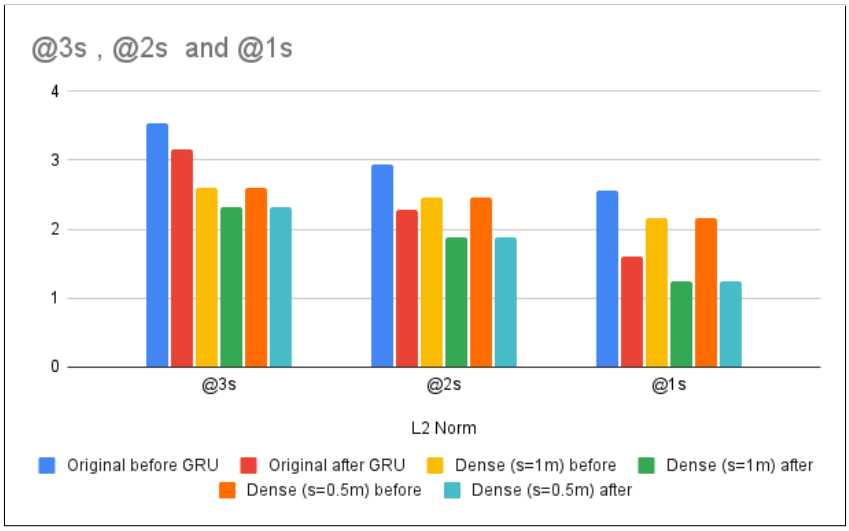


Figure 3.4 L2 Norm Visualized: Visualized is the L2 norm between the sampled trajectories and the observed ground truth at 1, 2, and 3 seconds along with the application of GRU is also tested.

No	Sampling	GRU	Resolution (m) [for Dense]	Safety	Headway	Lane Divider	Rule	Cost Volume	Comfort
1	Clothoids			**1.53**	0.91	**0**	**0.38**	0	**0.06**
2	Dense		1	1.85	**0.23**	0.84	8.08	8.21	2.64
3	Dense		0.5	3.49	0.71	0.07	5.77	4.33	0.99
4	Clothoids	✓	-	6.82	5.95	0	1.92	9.49	**53.62**
5	Dense	✓	1	9.74	5.12	0.45	11.15	7.69	55.63
6	Dense	✓	0.5	10.54	4.10	1.11	4.62	7.48	65.98
7	GT	-	-	11.32	1.55	0.52	13.85	0	9.79

Table 3.1 Cost Analysis for Clothoids and Dense Trajectories: Given is a cost breakdown for different settings of dense trajectories, along with removal and addition of GRU. The addition of GRU can be seen to improve the cost from the cost volume but degrades the comfort and safety costs, which are critical for autonomous driving.

L2 norm has also been written in Fig. 3.2 and visualized in Table. 3.4.

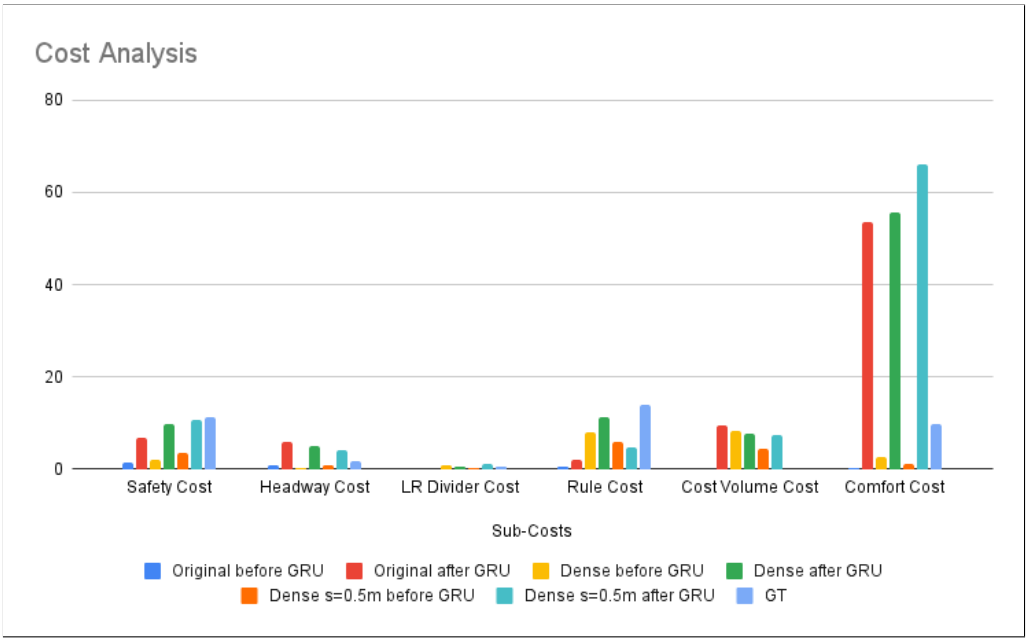


Figure 3.5 Cost Analysis Visualized for Clothoids and Dense Trajectories: Visualized is the cost breakdown for different Clothoid and Dense trajectories settings before and after applying the GRU. The addition of GRU is seen to improve the cost volume cost but degrades the comfort and safety cost, which are critical for autonomous driving.

Inferences Drawn:

- Application of GRU for refinement improves the L2 score. This is evident in 3.2. However, it creates a high jump in Safety, Headway, and especially Comfort Cost.
- This is attributed because these costs are computed for the optimal trajectory before GRU refinement.
- One can see the dense samples achieve better scores in Comfort and Safety, They also achieve very good L2 results at hyper-fine sampling.

No	Sampling	GRU	Resolution (m) [for Dense]	@3s	@2s	@1s
1	Clothoids		-	3.53	2.93	2.56
2	Dense		1	2.60	2.45	2.16
3	Dense		0.5	2.28	1.98	1.57
4	Clothoids	✓	-	3.15	2.27	1.61
5	Dense	✓	1	2.32	1.87	1.24
6	Dense	✓	0.5	**1.77**	**1.50**	**1.08**

Table 3.2 L2 Norm Quantized: A detailed quantification is done for the L2 norm between the sampled trajectories and the observed ground truth at 1, 2, and 3 seconds. The application of GRU is also tested. Dense sampling is associated to be the best-performing trajectory, with the lowest L2 norm.

In the next chapter, we will detail how to create uncertainty-aware occupancy grid maps and nominal estimates of neighbouring vehicles' trajectories from BEV. Subsequently, in the following chapter, we present our cost functions and our core algorithmic results: our sampling-based optimization for local trajectory planning. Note that we convert our trajectory $\boldsymbol{\xi}$ from the frenet to the ego-centric frame before applying the c_{BEV} as the BEVs are in the ego-centric frame.

Chapter 4

Uncertainty-Aware BEV Representations

In this chapter, we first detail the process of generating BEV representations from a set of monocular-view images. Thereupon, we elaborate on creating such uncertainty-aware BEV representations using the closest-distance query. Finally, we detail our novel optimizer which uses a sampling algorithm to return optimal trajectories with probabilistic collision avoidance.

4.1 Generating BEV Representations from Surround Monocular Images

4.1.1 Network Architecture

We adopt the backbone of ST-P3 [30] to extract BEV features from a sequence of N surround camera monocular images. Using the voxel pooling operation [23], we transform them into the Bird's Eye View space $b_i \in \mathbb{R}^{C \times H \times W}$ using intrinsic and extrinsic parameters. To generate a consistent representation of the scene, we use ego-motion warping as proposed by [15] on past Bird's Eye View features using the inverse of ego-translation from x_T to x_{T-k} for $k > 0$ a timestep in the past.

The warped features are then passed into the Spatio-temporal module [15, 16] to enhance the temporal features $S_i \in \mathbb{R}^{C \times H \times W}$. To obtain intermediate future representations, the Spatio-temporal features are processed using a Convolutional Dual-GRU [16] in a seq-to-seq fashion, followed by a decoder.

4.1.2 Ensuring Temporal Consistency

During closed-loop evaluation, spatial and temporal consistency must be consistent across inferences to achieve correct scene understanding, especially when adversaries move into locations or orientations not abundant in the training data. We extrapolate position using a constant velocity model to resolve the disappearance of predicted bounding boxes to ensure temporal consistency. To achieve this, we first obtain instance centers $C_{i,t}$, and bounding box $B_{i,t}$ of our blobs in the occupancy maps across each inference at time step t. We track the instance centers across inferences, and in the event of any center suddenly disappearing from the prediction, we use the past instance centers $C_{i-k}, k \in \{1, \dots 3\}$ information to obtain the current center $C_{i,t}$ by linearly extrapolating the previous instance centers.

Similarly, we take the average dimension of the bounding box from the previous timesteps and fill the processed center with the dimensions of the calculated box.

4.2 Uncertainty Aware BEV Representations

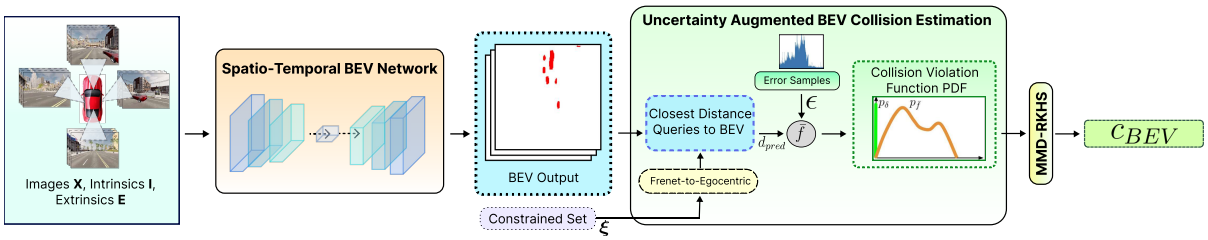


Figure 4.1 Uncertainty-Augmentation Pipeline: Our approach uses a Spatio-Temporal Network [16] to obtain a set of future BEV predictions, which we convert into an occupancy map prediction. We used the ground-truth information to learn the uncertainty in the BEV prediction. During inference time, we query the closest occupied cell to the ego-vehicle and then perturb it with samples drawn from the learned uncertainty. We then use the noisy samples of distance queries and use Reproducing Kernel Hilbert Spaces (RKHS) of Probability Density Functions (PDF) of Collision Violation Function to optimize our uncertainty-aware trajectory with the Maximum Mean discrepancy (MMD) measure as the surrogate cost for collision avoidance. We adopt a sampling-based approach and augment a projection operator into the optimization pipeline detailed in Chapter 4 for constraint satisfaction.

Given a sequence of images for N surround cameras, for the P past frames, $\mathbf{X}_k^n$, we generate BEV representations for a future horizon of F frames using ST-P3's architecture [16, 23], centered around the current ego-vehicle location. We can obtain distance queries to the closest obstacle in the BEV predictions/occupancy map $\mathbf{O}$. More precisely, let $D_O : \mathbb{R}^2 \rightarrow \mathbb{R}$ be a computationally fast function such that $d[k] = D_0((x[k], y[k]), \mathbf{O})$ represents the distance to the closest occupied cell at any time step k for any query point $(x[k], y[k])$. The distance queries over all time steps can be converted into collision costs in the form

$$\overline{f} = \prod_{k=k_0}^{k=k_n} \max(r_{safe} - d[k], 0) \tag{4.1}$$

where r_{safe} is the required minimum clearance between the ego-vehicle and its closest neighbour at time-step k.

We next present a core component of our pipeline; estimating uncertainty in $d[k]$ and formulating a probabilistic variant of (4.1) that can model probabilistic safety.

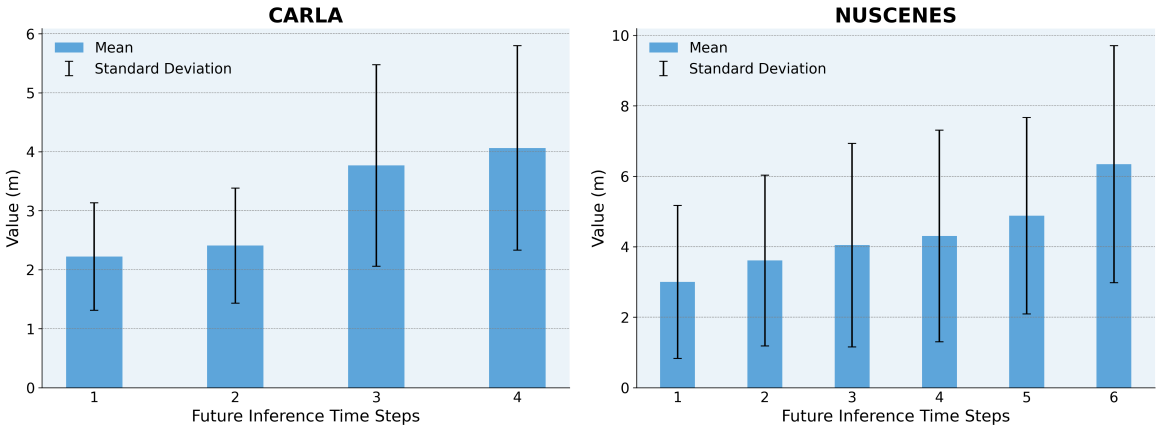


Figure 4.2 Mean and Standard Deviation of Error with Time: The mean and variance of the Error Distribution $\mathbf{\Delta_k}$ with time for CARLA and NuScenes. The mean and standard deviation increase with time, indicating that predictions further into the future are less reliable.

4.2.1 Accumulating Error in Closest-Distance Queries on BEV Predictions

During offline training, We have access to the actual ground truth BEVs, and thus the ground truth vehicles' positions in CARLA[10] and NuScenes[4]. We use this information to compare the predicted BEV with that generated BEV from the ground truth by quantifying the error in the distance queries. Fig.4.2 shows the mean and standard deviation of the error in distance to the closest occupied cell $d[k]$ observed in CARLA and NuSenes. As can be seen, due to the nature of BEV segmentation provided by ST-P3's architecture, the uncertainty in distance queries increases with time. We fit an average time-dependent distribution $\mathbf{\Delta}_k$ over several scenes in CARLA and NuScenes.

4.2.2 Probabilistic Safety Through Distance Samples

Let $\epsilon_{k,i}$ be the i^{th} sample drawn from $\mathbf{\Delta}_k$ such that $d_i[k] = d[k] + \epsilon_{k,i}$ represent the noisy samples of the distance to the closest occupied cell. We draw m such samples. We can now use these distance samples to compute the sample estimate of collision-cost (4.1) in the following manner :

$$\overline{f}_i = \prod_{k=k_0}^{k=k_n} \max(r_{safe} - d_i[k], 0) \tag{4.2}$$

Expression (4.2) represents the various possibilities of collision cost due to the uncertain distance information. We intend to use all the samples of $\overline{f}_i$ to infer the probability of collision avoidance. A simple

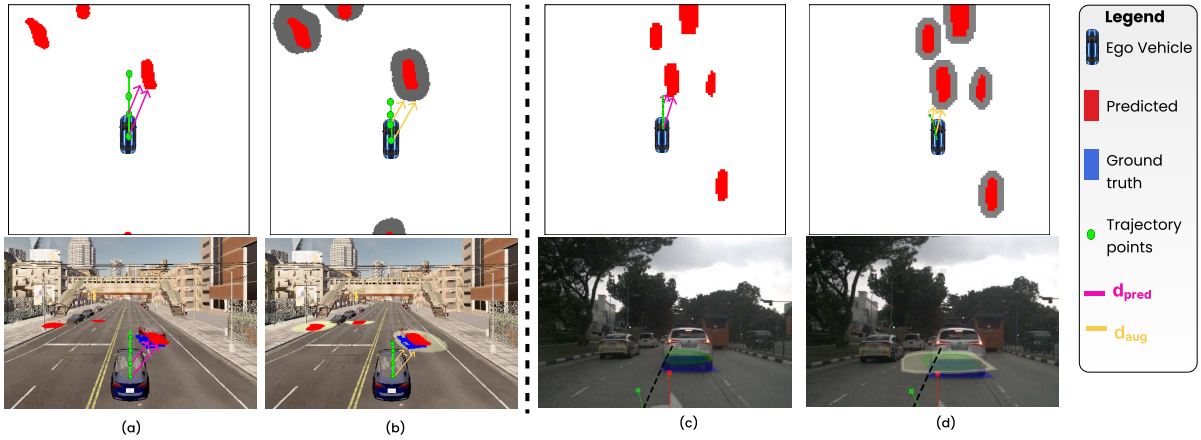


Figure 4.3 Closest-Distance Augmentation: **(a), (c)**: The scene before augmenting uncertainty in CARLA and NuScenes respectively. **(b), (d)**: The same scene after augmenting uncertainty estimates into trajectory points. The grey area is just for visualization and represents the bubble around the cluster if the uncertainty-augmented collision violation was sampled at that point.

choice is to compute the mean of all the samples. However, such an approach would not capture the true notion of risk. In our approach, we use the concept of distribution embedding in RKHS.

Let $\kappa : \mathbb{R}^2 \rightarrow \mathbb{R}$ be a positive-definite kernel function (e.g. Gaussian kernel) associated with RKHS. Then, the RKHS embedding of $\overline{f}_i$ is given by [14].

$$\mu_{\overline{f}} = \sum_{i=0}^{m} \frac{1}{m} \kappa(\bar{f}_i, \cdot), \qquad \mu_{\delta} = \sum_{i=0}^{m} \frac{1}{m} \kappa(\mathbf{0}, \cdot) \tag{4.3}$$

The first half of (4.3) uses all the samples of f_i to represent the underlying distribution as a point $\mu_{\overline{f}}$ in RKHS. As shown in [14], [13], the l_2 distance between $\mu_{\overline{f}}$ and the RKHS embedding of Dirac-Delta distribution centered at zero, μ_{δ} can be used as a measure of the probability of collision avoidance. Thus, we define our uncertainty-aware c_{BEV} as

$$c_{BEV} = \overbrace{\|\mu_{\overline{f}} - \mu_{\delta}\|_2^2}^{MMD} \tag{4.4}$$

Few important points about c_{BEV} are in order

- First, the r.h.s of (4.4) can be efficiently computed using the so-called kernel trick. Also, (4.4) explicitly depends on the ego-vehicle trajectory.
- Given a set of ego-vehicle trajectories, the one for which the c_{BEV} of (4.4) is close to zero, will have the highest probability of collision avoidance.

4.2.3 Advantage of BEVs in Error Quantification

In monocular vision setup, one can find the below representations and possible methods to accumulate the error and model the uncertainty in their predictions -

- Predicting *obstacle centers* into the future: [15] predicts instance centers, which we use to track the vehicle into the future. However, this requires precise predictions of the center of the obstacle and is extremely sensitive to errors in obstacle location at test time, especially on out-of-distribution data.

- *3D monocular object detection* predicting bounding boxes into the future: [25] and [24] predict 2D bounding boxes in monocular vision setup. The non-parametric uncertainty can be modeled over the box parameters - height, width, location, and rotation. We can use the Minkowski difference to compute the shape difference between the predicted boxes. However, this approach has been found to lead to conservative behaviours, as slight errors in rotation or location can lead to a very large difference region.

Compared to the above approaches, our method models a single observation parameter - distance to the closest obstacle to model the uncertainty. We eliminate direct supervision of the other parameters during training and are better able to capture the shape (or error in the shape) of the vehicle.
In the next chapter, we present our sampling-based optimizer to compute trajectories optimal concerning our uncertainty-aware c_{BEV}.

Chapter 5

Sampling based Optimizer

In this chapter, we first formulate a cost function, which includes the analytical and BEV-based costs. We then derive a compact matrix representation of the same. We detail the proposed optimizer along with a detailed algorithm of the same. We follow it by highlighting the batch projection operator in our approach and justifying it's needs.

5.1 Formulating Cost Function

The matrix $\mathbf{W}$ is formed by a piece-wise combination of cubic polynomial trajectories, each of which operates for a time interval δt. $\mathbf{c_x}$ and $\mathbf{c_y}$ are coefficients that are obtained through the optimization process. The higher derivatives of the position trajectory have the general form $\mathbf{W}^{(q)}\mathbf{c}_x$ (along the x-axis), where $\mathbf{W}^{(q)}$ represents the q^{th} derivative of the basis matrix. The y component of the ego-vehicle trajectory and its derivatives are obtained similarly with the same basis matrices.

We can formalize the local planner in the following form, wherein $(.)^{(q)}$ represents the q^{th} derivative of the variable and $(x[k], y[k])$ represents the position of the ego-vehicle at time-step k.

$$\sum_k c_a(x^{(q)}[k], y^{(q)}[k]) + c_{BEV}(x[k], y[k]), \tag{5.1a}$$

$$(x^{(q)}[k_0], y^{(q)}[k_0], x^{(q)}[k_f], y^{(q)}[k_f]) = \mathbf{b}, \tag{5.1b}$$

$$\mathbf{g}(x^{(q)}[k], y^{(q)}[k]) \leq 0 \tag{5.1c}$$

The first term in the cost function c_a captures costs that can be analytically modeled: smoothness, a departure from cruise speed, tracking, etc. The second term c_{BEV} is the cost map generated from the BEV representation. Typically, c_{BEV} captures drivable area, agent interactions, etc. The equality

constraints (5.1b) ensure boundary conditions on the position derivatives. Our formulation considers $q = (0, 1, 2)$. The inequality constraints (5.1c) can capture bounds on velocities, accelerations, drivable area, etc. We present a list of inequalities contained in $\mathbf{g}(.)$ in Table 5.1.

Constraint Type	Expression	Parameters
Discrete-time barrier for longitudinal separation g_1	$g_2[k] : h_2[k+1] - h_2[k] \geq -\gamma_{long} h_2[k]$ $h_2[k] = x_o[k] - x[k] \geq s_{min}$	s_{min}: minimum longitudinal separation $x_o[k]$:x-coordinate of leading vehicle at time index k γ_{long}: Longitudinal barrier constant [35]
Velocity bounds $g_2 = (g_{2,lb}, g_{2,ub})$	$g_{2,ub}[k] : \sqrt{\dot{x}[k]^2 + \dot{y}[k]^2} \leq v_{max}$ $g_{2,lb}[k] : \sqrt{\dot{x}[k]^2 + \dot{y}[k]^2} \geq v_{min}$	v_{min}, v_{max}: min/max velocity of the ego-vehicle
Acceleration bounds g_3	$g_3[k] : \sqrt{\ddot{x}[k]^2 + \ddot{y}[k]^2} \leq a_{max}$	a_{max}: max acceleration of the ego-vehicle
Discrete-time barrier for Lane boundary $g_4 = (g_{4,lb}, g_{4,ub})$	$g_{4,ub}[k] : h_{4,ub}[k+1] - h_{4,ub}[k] \geq -\gamma_{lane} h_{4,ub}[k]$, $g_{4,lb}[k] : h_{4,lb}[k+1] - h_{4,lb}[k] \geq -\gamma_{lane} h_{4,ub}[k]$ $h_{4,ub}[k] = -y[k] + y_{ub}$ $h_{4,lb}[k] = y[k] - y_{lb}$	y_{lb}, y_{ub}: Lane limits as a function of the ego-vehicle's position. γ_{lane}: Lane barrier constant [35]

Table 5.1 List of Inequality Constraints: The constraints used in the table are used in the projection optimization at step k. Each constraint is detailed with an expression along with associated parameters. Barrier constraints are also included.

5.2 Compact Matrix Representation

Using the trajectory parameterization(3.1), we can represent (5.1a)-(5.1c) in the following compact form. This representation will simplify the exposition in later sections.

$$c_a(\boldsymbol{\xi}) + c_{BEV}(\boldsymbol{\xi}) \tag{5.2}$$

$$\mathbf{A}\boldsymbol{\xi} = \mathbf{b}, \qquad \mathbf{g}(\boldsymbol{\xi}) \leq \mathbf{0} \tag{5.3}$$

The developments in the last section provide us c_{BEV} as an MMD map computed through a neural network-based BEV representation augmented with uncertainty estimates. This section develops a sampling-based approach for optimizing over c_{BEV}. The key novelty of our optimizer is that it incorporates a projection operator to push the sampled trajectories toward feasible regions before evaluating cost over them. We approximate predictions of neighbouring vehicle centres that can be obtained by averaging the predicted vehicle representation in the BEV frame. This is used to formulate discrete-time barrier constraints [35] for longitudinal separation g_1.

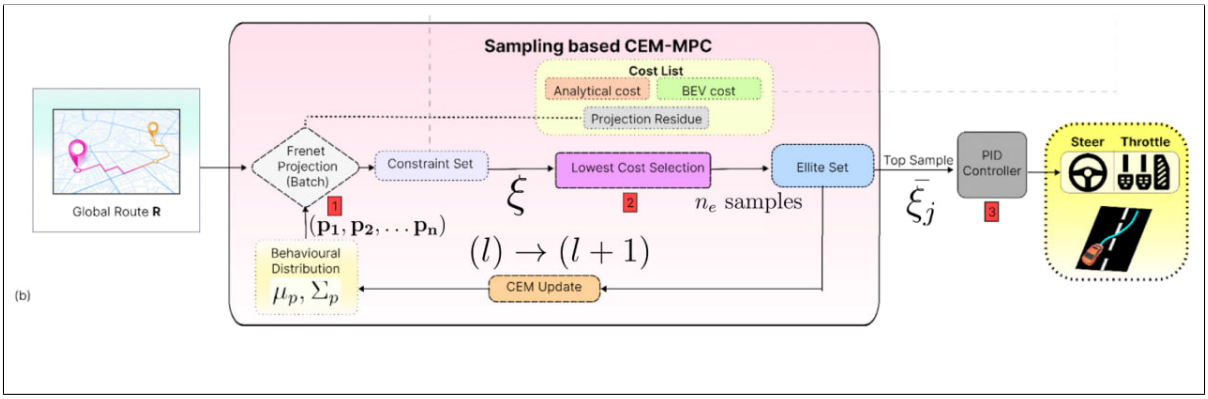


Figure 5.1 Optimizer Pipeline: The optimizer samples n_s samples from a behavioural distribution, which is used to perform the batched Frenet projection. The top constrained set ξ is ranked using custom costs, and top n_e samples are used to update the parameters of the behavioural distribution. Top sample $\bar{\xi}_j$ is given to the PID controller to be executed.

5.3 Proposed Optimizer

The overall algorithm is presented in **Alg.**5.2, wherein the left superscript l is used to track the values of the respective variable across iterations. The algorithm proceeds by sampling behavioural inputs $\mathbf{p}_j$ such as lateral offsets and desired velocity set-points. These are then fed to a Frenet space planner inspired by [33]. The trajectory coefficients that the Frenet planner returns are then fed to our projection optimizer in lines 7-8. The resulting output $\overline{\boldsymbol{\xi}_j}$ is then evaluated for constraint residuals in line 9. We rank the top n_s samples with the lowest constraint residual in the $ConstraintEliteSet$ in line 10. In line 11, we compute an augmented cost (residuals+primary cost) over the samples from $ConstraintEliteSet$. We then rank these samples based on the augmented cost value, extract the lowest n_e samples, and place them in $EliteSet$ (line 13). We update the sampling distribution based on the samples from the $EliteSet$. Specifically, we use the formula (5.4a)-(5.4b) from [1] to update the mean and covariance for sampling in the next iteration. The constant β is the so-called temperature parameter.

$$ {}^{l+1}\boldsymbol{\mu}_{\mathbf{p}} = (1-\eta)^{l}\boldsymbol{\mu}_{\mathbf{p}} + \eta \frac{\sum_{j=1}^{j=n_e} s_j \mathbf{p}_j}{\sum_{j=1}^{j=n_e} s_j}, \tag{5.4a} $$

$$ {}^{l+1}\boldsymbol{\Sigma}_{\mathbf{p}} = (1-\eta)^{l}\boldsymbol{\Sigma}_{\mathbf{p}} + \eta \frac{\sum_{j=1}^{j=n_e} s_j (\mathbf{d}_j - {}^{l+1}\boldsymbol{\mu}_{\mathbf{p}})(\mathbf{p}_j - {}^{l+1}\boldsymbol{\mu}_{\mathbf{p}})^T}{\sum_{j=1}^{j=n_e} s_j} \tag{5.4b} $$

$$ s_j = \exp \frac{-1}{\beta}(c_a(\overline{\boldsymbol{\xi}}_j) + c_{BEV}(\overline{\boldsymbol{\xi}}_j) + r_j(\overline{\boldsymbol{\xi}}_j) \tag{5.4c} $$

Algorithm 1: Sampling-Based Optimization Over Uncertainty Augmented Learned BEV-based Costs

N = **Maximum number of iterations**

Initiate mean ${}^{l}\boldsymbol{\mu}_{\mathbf{p}}, {}^{l}\boldsymbol{\Sigma}_{\mathbf{p}}$, at $l = 0$ for sampling Frenet Parameters $\mathbf{p}$

for $l = 1, l \leq N, l++$ **do**

  Draw $\overline{n}_s$ Samples $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, ..., \mathbf{p}_{n_s})$ from $\mathcal{N}({}^{l}\boldsymbol{\mu}_{\mathbf{p}}, {}^{l}\boldsymbol{\Sigma}_{\mathbf{p}})$

  Initialize $CostList$ = []

  Query Frenet-planner for $\forall \mathbf{p}_j$: $\boldsymbol{\xi}_j = \text{FrenetPlanner}(\mathbf{p}_j)$

  Project to Constrained Set

$$\overline{\boldsymbol{\xi}}_j = \arg\min_{\overline{\boldsymbol{\xi}}_j} \frac{1}{2} \left\| \overline{\boldsymbol{\xi}}_j - \boldsymbol{\xi}_j \right\|_2^2$$

$$\mathbf{A}\overline{\boldsymbol{\xi}}_j = \mathbf{b}, \qquad \mathbf{g}(\overline{\boldsymbol{\xi}}_j) \leq \mathbf{0}$$

  Define constraint residuals: $r_j(\overline{\boldsymbol{\xi}}_j)$

  $ConstraintEliteSet \leftarrow$ Select top n_s samples of $\mathbf{p}_j, \overline{\boldsymbol{\xi}}_j$ with lowest constraint residual norm.

  $cost \leftarrow c_a(\overline{\boldsymbol{\xi}}_j) + c_{BEV}(\overline{\boldsymbol{\xi}}_j) + r_j(\overline{\boldsymbol{\xi}}_j)$, over $ConstraintEliteSet$

  append $cost$ to $CostList$

  $EliteSet \leftarrow$ Select top n_e samples of $(\mathbf{p}_j, \overline{\boldsymbol{\xi}}_j)$ with lowest cost from $CostList$.

  $({}^{l+1}\boldsymbol{\mu}_{\mathbf{p}}, {}^{l+1}\boldsymbol{\Sigma}_{\mathbf{p}}) \leftarrow$ Update distribution based on $EliteSet$

end

return *Trajectory coefficients* $\overline{\boldsymbol{\xi}}_j$ *corresponding to lowest cost in the* $EliteSet$

Figure 5.2 Sampling-Based Optimization Algorithm: The steps are listed to sample and compute uncertainty augmented Learned BEV-based Costs. *ElliteSet* is chosen using the *CostList* computed, and top n_e samples are used to update the distribution for the next set of samples $(p_1, p_2, \ldots p_{n_s})$

5.4 Batch Projection

The optimization in lines 7-8 can be done in parallel. We extended the GPU accelerated projection optimizer from [22], [29] to include the discrete-time barrier constraints for longitudinal separation and lane-boundary constraints (recall Table 5.1, rows 1 and 4). Moreover, unlike [29], our projection operator does not have collision constraints as these have been rolled into the c_{BEV} costs. We present the detailed derivation in the appendix. But the core idea essentially boils downs to reducing projection in lines 7-8 to a sequence of optimizations that can all be trivially batched over GPUs.

In the next chapter, we show the results of the experiments performed, and compare with existing baselines. We report improvement in the stated metrics, and try to analyze the results.

Chapter 6

Experiments and Results

6.1 Experiment Setting

We conduct our closed-loop experiments on the self-driving simulator CARLA [11] v0.9.13, owing to its high-precision physics engine and sensors capable enough for our use case. We implemented our sampling-based optimizer in Google's JAX - a GPU accelerated internal library[3]. Our perception models were trained on Nvidia 3090, and the CARLA simulation and open-loop experiments on NuScenes were conducted on Nvidia TITAN X. We used $\gamma = 0.1$ for the RBF Kernel. We used $\gamma_{lane} = 0.9$ and $\gamma_{long} = 0.9$. The temperature parameter β was taken as 0.9. and the learning-rate η at 0.6. We chose v_{max}, v_{min} at $(10, 0)$.

We adopted the BEV configuration setting of ST-P3, with a resolution of $(H, W) = (200, 200)$ grid at 0.20m $\times$ 0.20m resolution in CARLA, and $0.50m \times 0.50m$ in NuScenes. We use $P = 3$ time frames of past context and predict $F = 4$ frames into the future for CARLA and $F = 6$ frames for NuScenes.

6.2 Generating Scenarios in CARLA

We divide our experiments on CARLA into 2 categories based on the lane boundary -

- Inlane Driving (**In**): Within Inlane, we create the scenarios - Abrupt Stopping/Slowing, and Cutin of adjacent vehicle.

- Overtaking Allowed (**Ov**): We simulate static and dynamic vehicles while overtaking. We change $y_{lb} = 3.5$ to allow overtaking.

We adopt the 14 standard routes introduced in [9] and modify it to test on 2.8 km of route length across multiple towns. A visualization of the routes along with their detailed lengths is given below.

Route	Town/Route	Original Length (m) [9]	Part chosen length (m)
1	Town01 Route01	1303	206
2	Town01 Route02	322	286
3	Town02 Route01	110	110
4	Town02 Route01	638	325
5	Town03 Route01	306	58
6	Town03 Route02	462	76
7	Town04 Route01	2811	128
8	Town04 Route02	204	38
9	Town06 Route01	686	254
10	Town06 Route02	706	187
11	Town05 Route01	337	192
12	Town05 Route01	1642	638
13	Town03	961	145
14	Town04 Route03	225	158
	Total	10713	**2801**

Table 6.1 Information of Routes: A description of the 14 routes used by [9], along with their original and chosen length. These routes will be combined with different weather settings to generate multiple scenarios.

Due to the controlled nature of the neighbouring vehicles, we can re-create the same scenario for the baselines discussed in the next section. For the two types of ego behaviors. Further, the other dynamic agents in the scene can be participants in any of the challenging scenarios listed above.

6.3 Baselines and Metrics

6.3.1 Baselines

We use camera-based methods in our approach which leverage BEV representations as intermediates for planning. We do not consider CCO-VOXEL [14] and UrbanFly [13] as they parameterize the trajectories over 3D coefficients with 6 Degrees of Freedom and violate the non-holonomic assumption, making it unsuitable for Autonomous Driving applications.

0. **ST-P3 Gaussian**: (not considered in quantitative results) We adapt the approach of Calibrated Perception Uncertainty[20], and fit a 2D Gaussian Mixture Model to the log-likelihood of probabilities. We use the fitted ellipses as obstacle representations and couple it with the cost-volume

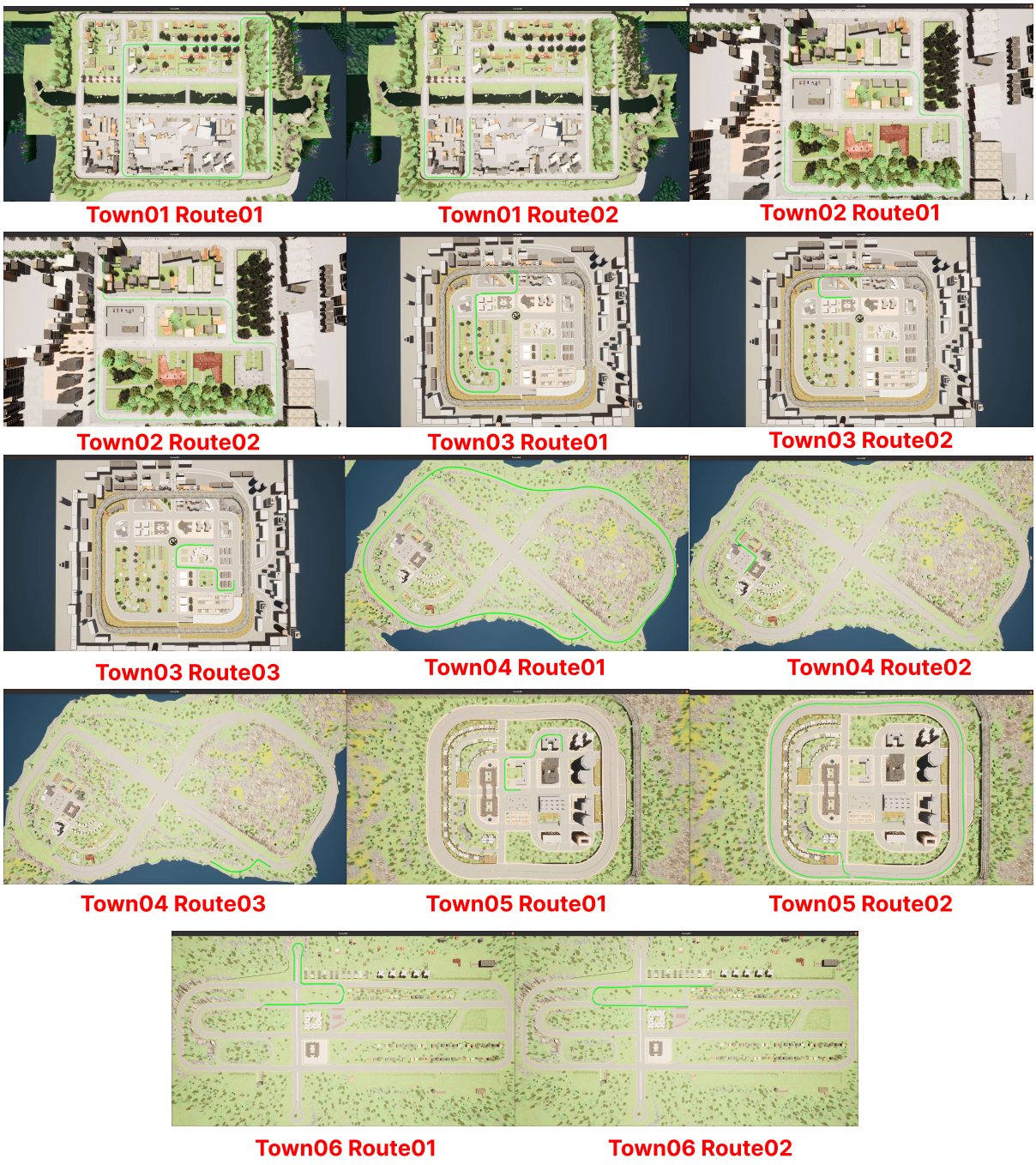


Figure 6.1 Town-wise Distribution of Routes: A visualisation of different routes in CARLA Town-wise, adapted from [9]. The start point is represented in red, the endpoint in blue, and the trajectory is visualized in green.

based planning. Note that we do not consider this approach in our quantitative estimation as we found it to be too conservative and unable to complete the route.

1. **Lift-Splat-Shoot (Static) (LSS)** [23]: Lift-Splat Shoot used to generate BEV representations of the current timestep, with a cost grid $c_{LSS} \in \mathbb{R}^{H \times W} = (200, 200)$. At test time, we sample polynomial trajectories from a fixed set and the lowest cost trajectory is passed to the PID controller to obtain the control inputs.

2. **ST-P3 Imitation Learning (IL) (ST-P3-IL)** [16]: BEV representations of the current and future timesteps are generated. It predicts a Cost-Volume over time, $c_{ST-P3} \in \mathbb{R}^{F \times H \times W} = (4, 200, 200)$. At test time, we use the same approach as stated above.

3. **ST-P3 with Sampling-based optimization (ST-P3-SO)**: Here, we use our sampling-based optimisation algorithm over the cost volume to obtain the optimal trajectory. The c_{BEV} cost is the cost-volume generated by the ST-P3. Thus, this baseline has been constructed to showcase the importance of our proposed uncertainty-aware c_{BEV} based on MMD.

4. **UAP-BEV**: Our approach, with uncertainty-augmentation and sampling-based optimiser.

6.3.2 Metrics

The metrics in closed-loop CARLA simulation are -

1. **Collisions**: The average number of vehicle collisions per km $(/km)$.

2. **Route Completion (RC)**: The percentage of routes completed, i.e.: the ego-vehicle navigates to the target, without getting stuck or stopping for a fixed time.

3. **Duration**: Total time (in s) for successfully completed routes.

4. **Smoothness**: Rate of Change of Acceleration (m/s^3).

For NuScenes, we perform open-loop experiments and only report **Collision Rate** $(\%)$ and **Smoothness**. We use the **Intersection-over-Union (IoU)** to denote the improvements in the quality of BEV representations in the later part of this chapter.

6.4 Closed-Loop Experiments on CARLA

6.4.1 Qualitative Results

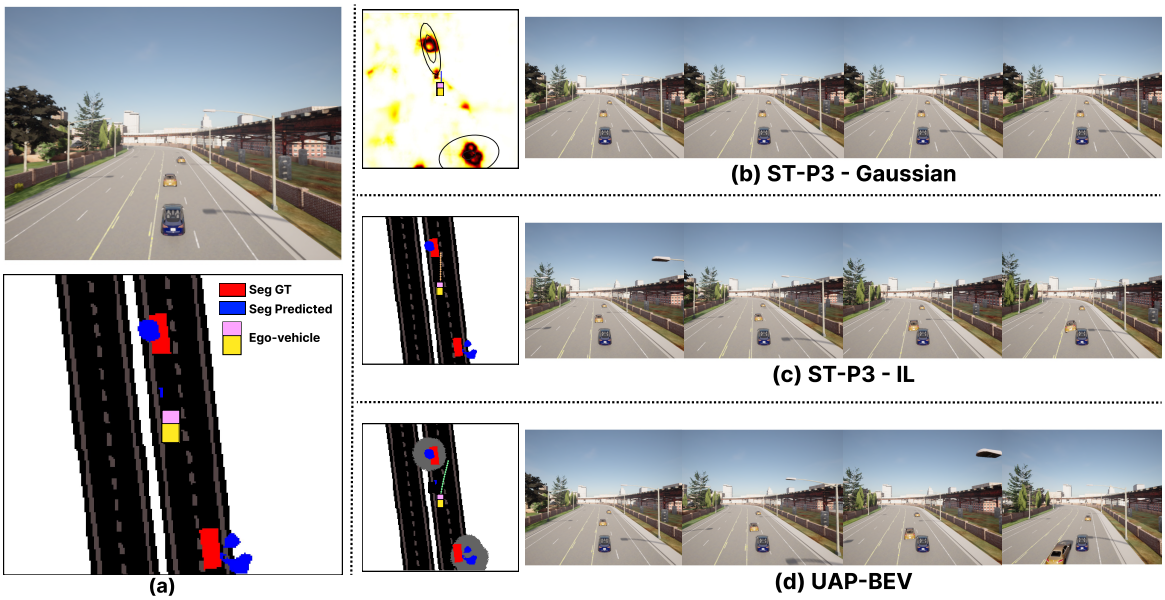


Figure 6.2 Qualitative Results in Overtaking Scenario: Given is scene simulated in CARLA where the ego vehicle (blue) is driving and has to overtake the static vehicle in front to reach destination using a BEV perception model ST-P3. Visualized are the BEV representations used for planning. Note that the grey area is only for visualization purpose, and the closest-distance augmentation results in the uncertainty estimate. Note that the RGB perspective shots are frontal view with the camera just behind the ego-vehicle, and the BEV corresponds to a the region of 20mx20m around the ego-vehicle.

We observe in 6.2 that using a Gaussian approximation of underlying uncertainty (**ST-P3 Gaussian**) proves to be conservative and ego-vehicle fails to move ahead. Traditional Imitation-Learning (IL) based approaches also cannot account for the error (**ST-P3 IL**) and result in a collision. Our uncertainty-aware planner (**UAP-BEV**) is able to overtake the leading vehicle while countering the error in perception, maintaining a safe distance compared to others.

Fig.6.3 shows a scenario, where the ego vehicle experiences a cutin from a vehicle in an adjacent lane. Due to a mismatch between the ground-truth d_{GT} and predicted d_{pred} distance to the closest obstacle, the ST-P3 trajectory collides with the leading vehicle. In contrast, our uncertainty-aware approach (UAP-BEV) that models c_{BEV} through MMD and uses an algorithm for optimization, is able to successfully navigate through the cutin.

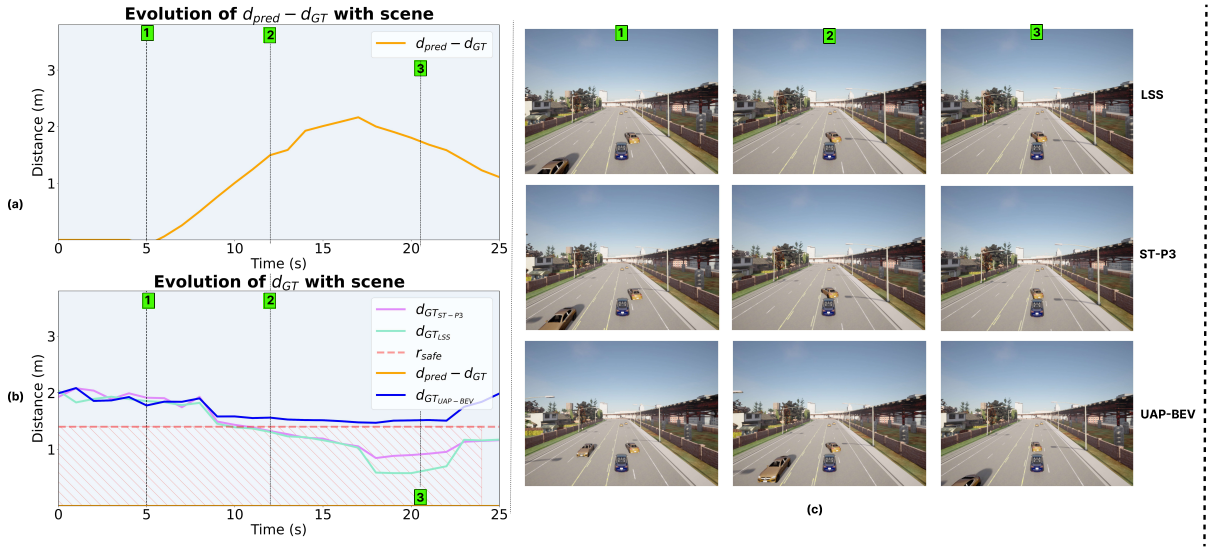


Figure 6.3 Qualitative Results in a Cutin scenario: Ego-vehicle faces a cutin from adjacent vehicle, and has to perform Inlane **(In)** driving. On the left side, we plot the error $d_{pred} - d_{GT}$ with time, as the cutin happens. We also plot below, the evolution of d_{GT} (which is collision distance) with time for all the baselines.

6.4.2 Quantitative Results

Below, we report the closed loop results for the 14 routes overall on CARLA -

Method	**Collisions** ↓ (/km)		**RC** ↑ (%)		**Duration** ↓ (s)		**Smoothness** ↓ (jerk,m/s^3)	
	In	*Ov*	*In*	*Ov*	*In*	*Ov*	*In*	*Ov*
LSS	0.013	0.014	26.57	20	39.84	31.84	4.74	4.85
ST-P3	0.011	0.013	48.44	46	28.56	26.34	4.53	4.55
ST-P3-SO	0.004	0.005	85.94	90	22.19	22.7	2.98	2.95
UAP-BEV	**0.00003**	**0.00005**	**100**	100	**18.9**	19.1	**1.88**	2.93

Table 6.2 Benchmark Comparison on CARLA On Inlane (*In*) and Overtaking Scenarios (*Ov*), the comparison is shown against the existing baselines LSS, ST-P3, ST-P3 (SO) and UAP-BEV (Ours). We report 100% route completion and improvements in all other metrics.

Table 6.2 presents the quantitative results on the CARLA simulator. It can be seen that the ST-P3-SO baseline, which couples ST-P3 perception with our sampling-based optimizer from the algorithm, already substantially improves the collision rate. Moreover, our primary method UAP-BEV which uses the optimizer algorithm with our MMD based c_{BEV} drives the collision rate to zero. Our UAP-BEV achieves an improvement of **39.8%** in smoothness metric over all the baselines. In both in-lane driving

and overtaking scenarios, our approaches achieve a route competition rate of almost double the other baselines. The ST-P3-SO and UAP-BEV demonstrate an improvement of **1.774x** and **2.06x** respectively over ST-P3-IL in route competition metric.

6.5 Open-Loop Results on NuScenes

To validate our proof-of-concept, we evaluated our method on the NuScenes dataset. We found that our method outperformed existing baselines. The qualitative and quantitative results are discussed below -

6.5.1 Qualitative Results

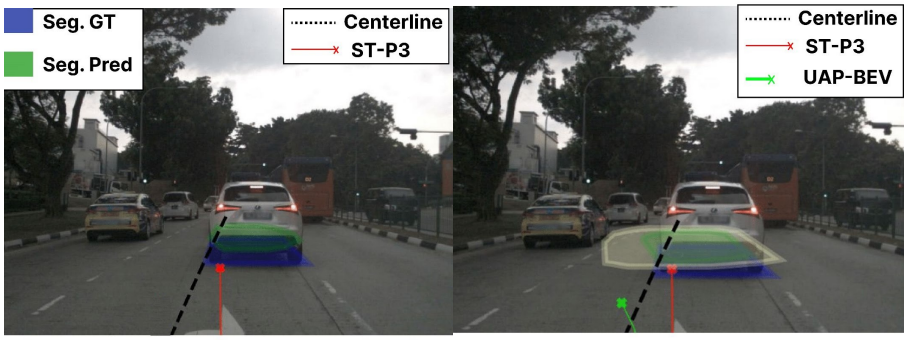


Figure 6.4 Qualitative Results in NuScenes: We demonstrate Collision Avoidance in one of the scenes in NuScenes. Upon addition of the uncertainty (yellow), to the existing prediction (dark green), the trajectory (red) becomes collision-free (light green) and is able to avoid the collision by a margin.

In figure 6.4, we compare our method against one of the baselines - **ST-P3**. This is the **ST-P3 IL** variant. In the figure, we can see that our method is able to avoid collisions more often than ST-P3. This is because our method takes into account the uncertainty in the BEV predictions.

6.5.2 Quantitative Results

Table 6.3 presents results on the NuScenes dataset. Here again, our UAP-BEV shows almost two to three times reduction in a collision over LSS and ST-P3-IL. Similar improvements can be found in the smoothness metric as well. We recall that the NuScenes dataset only allows us to perform open-loop simulation (executed trajectory of the ego-vehicle is pre-decided and fixed). Thus the improvement here

Method	Collision Rate(%) ↓	Smoothness (m/s^3) ↓
LSS	9.31	5.31
ST-P3	6.01	5.11
ST-P3-SO	4.31	2.82
UAV-BEP	**3.34**	**2.80**

Table 6.3 Benchmark Comparison on NuScenes: The table compares the existing baselines on an Open-Loop dataset. Here, only collision rate and smoothness are reported, as they are the only relevant metrics for an open-loop setting.

is less drastic compared to CARLA. We also do not report the route-competition metric as it is irrelevant here.

6.6 Ablation Studies

6.6.1 Longitudinal Barrier Constraint

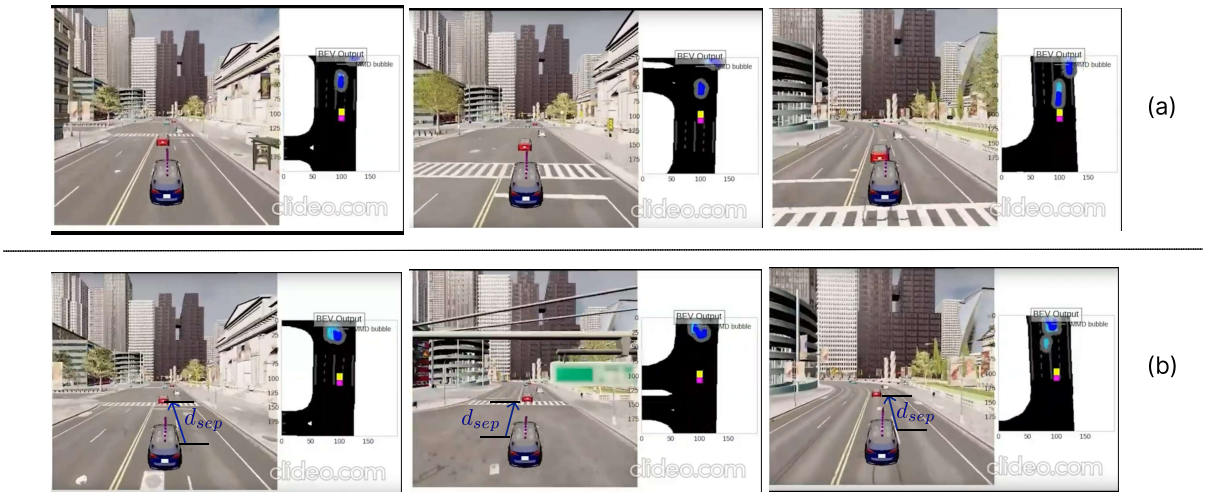


Figure 6.5 Longitudinal Barrier Ablation Given scene in CARLA with the ego-vehicle (blue) and leading vehicle (red), shown (a) Without Barrier, (b) With Barrier. The leading vehicle slows down, causing the ego-vehicle to crash, when the barrier is not applied. This is prevented in the latter case in the presence of a minimum barrier of d_{SEP}.

A qualitative result is shown in 6.5, where the leading vehicle abruptly slows down. Here, we see that in the absence of the barrier, the ego-vehicle is unable to slow down and crashes into the leading vehicle. This is primarily due to the error in future prediction. However, in the case of the longitudinal barrier,

we notice that ego-vehicle maintains a separation distance $d_{sep} = 10m$, from the leading vehicle, thus slowing down and avoiding a collision.

Table 6.4 demonstrates the effectiveness of the longitudinal barrier constraints. This constraint ensures a minimum separation distance from the leading vehicle in the scene. Note that we only utilize this constraint during the Inlane driving scenarios **(In)**. The trajectory of the leading vehicle was obtained by computing the approximate centres of the BEV predictions.

ID	UAP	Barrier	Collisions /km	Route Completion	Duration (s)
1			0.004	85.84	22.43
2		✓	0.0005	91.23	25.43
3	✓		0.0003	96.37	**19.17**
4	✓	✓	**0.00004**	**100**	21.18

Table 6.4 Barrier Ablations: The table denotes the effectiveness of our approach with and without a Longitudinal Barrier. The barrier enforces a minimum separation distance of d_{SEP}, thereby making it safer to navigate.

The longitudinal barrier constraint encourages conservative driving, by rewarding trajectories that maintain a minimum distance from the leading vehicle. It leads to safer trajectories observed through reduced collisions and increased RC. However, this comes at the expense of higher execution times.

6.6.2 Noise Modelling Methods

As discussed in the methodology section, errors are asymmetric along the X and Y axes, as well as into the future. We benchmark the new formulations **UAP-2D** and **UAP-Future** respectively. We report in Table 6.5 that modeling noise distributions separately improves the result over the standard implementation.

ID	Error Distribution	Collisions /km	Route Completion
1	Vanilla - UAP	0.0006	90.87
2	2D - UAP	0.0003	91.15
3	Future - UAP	0.0001	95.34
4	Combined - UAP	**0.00004**	**100**

Table 6.5 Asymmetric Error: The table further reports the metrics for different modifications to the existing computation techniques. The asymmetric error is encountered at different channels - along spatial (2D-UAP) and temporal (Future-UAP).

6.7 CEM Convergence Analysis

6.7.1 Convergence with Trace/Variance plots

Similar to [14] and [13], we perform Convergence Analysis on our CEM method. The cost profile of the mean trajectory is plotted and can be seen in . This signifies that the gradient update to the behavioural inputs distribution is such that the trajectories coming from it correspond to the lower-cost regions.

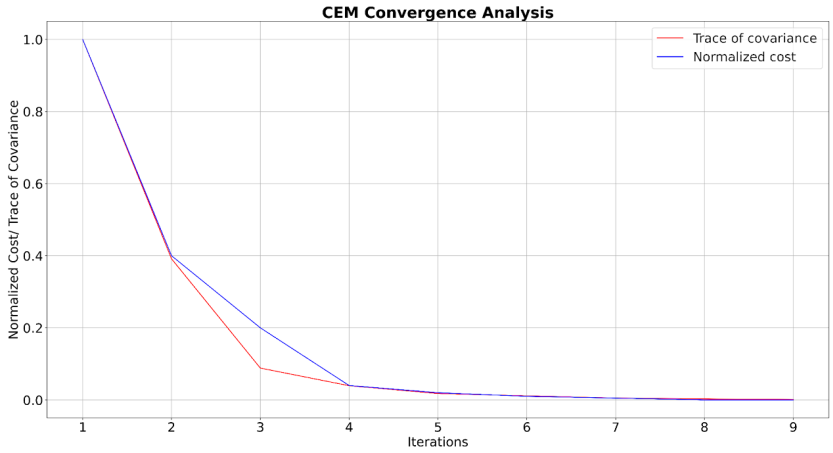


Figure 6.6 Trace vs Covariance plot: Given is a plot for normalized cost versus iterations and trace versus iterations. As the number of iterations increases, both the cost and trace decrease, indicating convergence in our approach.

6.7.2 Points within a trajectory

The histogram plot 6.7 shows that the bulk of trajectories converge towards low-cost region (around 0). This shows that our sampling algorithm gradually converges towards low-cost regions, and within 11 iterations, the trajectories sampled from new mean and variance have a value of 0 collision violation estimate along all points within them.

6.7.3 Compile Time Optimization

We use PyTorch 2.0, with JIT (Just-In-Time) compilation functionality. We also batch our MMD computations, unlike [13] and [14], which perform unbatched computation in C++/python-numpy, we are able to obtain significant speed improvements for the planning part.

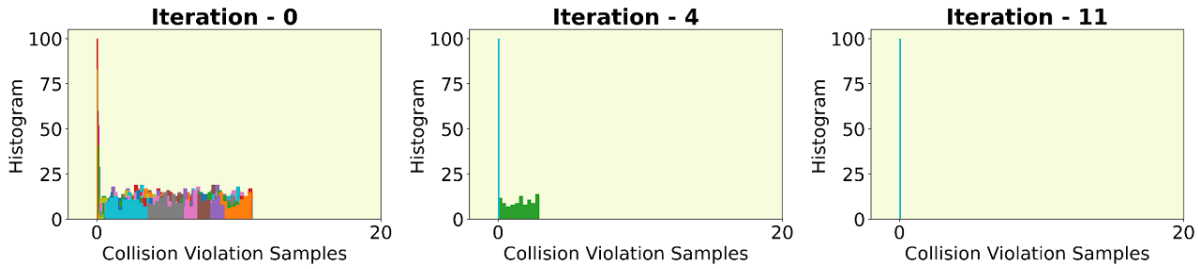


Figure 6.7 Histogram Plot for Points across trajectory 6.7 Plots the distribution of points within a trajectory with the same color. With the x-axis representing collision violation cost and the y-axis denoting the number of such points. Gradual convergence can be shown as all points across a trajectory converge to near zero.

6.8 Improvements in Quality of BEV Representations

We report the improvement with the heuristic technique of ensuring temporal consistency, in Table 6.6.

Method	IoU	Method	Future mIoU
Lift-Splat-Shoot	21.34	NEAT	7.34
ST-P3	13.24	ST-P3	2.13
ST-P3 + TC*	**34.78**	ST-P3 + TC*	**18.23**

Table 6.6 IoU Scores for Baselines and us: The table reports our Perception Improvements with Temporal Consistency. It shows both for the current timestep (left) and averaged over the future timesteps (right), quantized by Future mIoU.

Chapter 7

Conclusion

This thesis shows the characterization of uncertainty in the popular BEV layout prediction methods with monocular RGB images as inputs and how nonparametric uncertainty can be used by a trajectory planning framework to execute collision-free trajectories. A sampling-based optimal rollout in the popular CEM formulation is used as the planning framework with two novel components.

The first novelty involves the non-convex projection of the CEM samples to the nearest sample that satisfies various constraints based on curvature, smoothness, and kinematics, and the second novelty introduces an MMD cost term based on matching distributions in the Kernel Hilbert Space. The efficacy of the proposed framework is shown by significant performance gain over prior arts [23, 16] that sample the best trajectory over a cost volume learned over a BEV estimate. We show such gains both in closed-loop simulation in a number of CARLA towns and open-loop trajectories on public datasets such as NuScenes.

UAP-BEV, to the best of our knowledge, is the first method to characterize non-parametric uncertainty in the BEV frame and leverage it to compute collision-free trajectories. The proposed planner utilizes a sampling-based optimization with a novel uncertainty-aware collision cost constructed from BEV predictions. Our proposed optimizer also includes a projection operator to push the sampled trajectories towards feasible regions before computing the cost over them.

Our current work involved generating custom scenarios and evaluating them with the representations obtained from monocular images with off-the-shelf networks. Future work would involve improving the quality of BEV representations with a non-heuristic approach (network architectural modification) and submitting our method to CARLA leaderboard v2 for more robust testing.

Related Publications

1. **UAP-BEV: Uncertainty-Aware Planning in Bird's Eye View Representations using Surround Monocular Images**

   Vikrant Dewangan, Basant Sharma, Tushar Choudhary, Sarthak Sharma, Aakash Aanegola, Arun. K. Singh, K. Madhava Krishna

   Accepted to IEEE International Conference on Automation Science and Engineering (CASE) 2023

Other Publications

1. **MPC-based obstacle aware multi-UAV formation control under imperfect communication**

   Vikrant Dewangan, Harikumar Kandath

   [*In process of submission*]

Bibliography

[1] M. Bhardwaj, B. Sundaralingam, A. Mousavian, N. D. Ratliff, D. Fox, F. Ramos, and B. Boots. Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation. In *Conference on Robot Learning*, pages 750–759. PMLR, 2022.

[2] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, and P. L'Ecuyer. Chapter 3 - the cross-entropy method for optimization. In C. Rao and V. Govindaraju, editors, *Handbook of Statistics*, volume 31 of *Handbook of Statistics*, pages 35–59. Elsevier, 2013.

[3] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[4] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[5] S. Casas, A. Sadat, and R. Urtasun. Mp3: A unified model to map, perceive, predict and plan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14403–14412, 2021.

[6] S. Casas, A. Sadat, and R. Urtasun. Mp3: A unified model to map, perceive, predict and plan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14403–14412, 2021.

[7] R. Chekroun, M. Toromanoff, S. Hornauer, and F. Moutarde. Gri: General reinforced imitation and its application to vision-based autonomous driving, 2022.

[8] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, pages 66–75. PMLR, 2020.

[9] K. Chitta, A. Prakash, and A. Geiger. Neat: Neural attention fields for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15793–15803, October 2021.

[10] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[11] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator, 2017.

[12] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR, 06–11 Aug 2017.

[13] S. S. Harithas, A. S. Thatavarthy, G. Singh, A. K. Singh, and K. M. Krishna. Urbanfly: Uncertainty-aware planning for navigation amongst high-rises with monocular visual-inertial slam maps, accepted in *2023 American Control Conference(ACC)*.

[14] S. S. Harithas, R. D. Yadav, D. Singh, A. K. Singh, and K. M. Krishna. Cco-voxel: Chance constrained optimization over uncertain voxel-grid representation for safe trajectory planning, accepted in *IEEE International Conference on Robotics and Automation (ICRA),2022*.

[15] A. Hu, Z. Murez, N. Mohan, S. Dudas, J. Hawke, V. Badrinarayanan, R. Cipolla, and A. Kendall. Fiery: future instance prediction in bird's-eye view from surround monocular cameras. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15273–15282, 2021.

[16] S. Hu, L. Chen, P. Wu, H. Li, J. Yan, and D. Tao. St-p3: End-to-end vision-based autonomous driving via spatial-temporal feature learning. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXVIII*, pages 533–549. Springer, 2022.

[17] A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017.

[18] A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018.

[19] Y. Kim, S. Kim, J. Shin, J. W. Choi, and D. Kum. Crn: Camera radar net for accurate, robust, efficient 3d perception, 2023.

[20] M. Kängsepp and M. Kull. Calibrated perception uncertainty across objects and regions in bird's-eye-view, 2022.

[21] Z. Liu, A. Amini, S. Zhu, S. Karaman, S. Han, and D. Rus. Efficient and robust lidar-based end-to-end navigation, 2021.

[22] H. Masnavi, J. Shrestha, M. Mishra, P. Sujit, K. Kruusamäe, and A. K. Singh. Visibility-aware navigation with batch projection augmented cross-entropy method over a learned occlusion cost. *IEEE Robotics and Automation Letters*, 7(4):9366–9373, 2022.

[23] J. Philion and S. Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *European Conference on Computer Vision*, pages 194–210. Springer, 2020.

[24] C. Reading, A. Harakeh, J. Chae, and S. L. Waslander. Categorical depth distribution network for monocular 3d object detection, 2021.

[25] T. Roddick, A. Kendall, and R. Cipolla. Orthographic feature transform for monocular 3d object detection, 2018.

[26] A. Sadat, S. Casas, M. Ren, X. Wu, P. Dhawan, and R. Urtasun. Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*, pages 414–430. Springer, 2020.

[27] A. Sadat, M. Ren, A. Pokrovsky, Y.-C. Lin, E. Yumer, and R. Urtasun. Jointly learnable behavior and trajectory planning for self-driving vehicles, 2019.

[28] D. H. Shin and S. Singh. Path generation for robot vehicles using composite clothoid segments. Technical Report CMU-RI-TR-90-31, Carnegie Mellon University, Pittsburgh, PA, December 1990.

[29] A. K. Singh, J. Shrestha, and N. Albarella. Bi-level optimization augmented with conditional variational autoencoder for autonomous driving in dense traffic. *arXiv preprint arXiv:2212.02224*, 2022.

[30] M. Tan and Q. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[31] M. Toromanoff, E. Wirbel, and F. Moutarde. End-to-end model-free reinforcement learning for urban driving using implicit affordances, 2020.

[32] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8445–8453, 2019.

[33] J. Wei, J. M. Snider, T. Gu, J. M. Dolan, and B. Litkouhi. A behavioral planning framework for autonomous driving. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 458–464. IEEE, 2014.

[34] G. Williams, P. Drews, B. Goldfain, J. Rehg, and E. Theodorou. Aggressive driving with model predictive path integral control. pages 1433–1440, 05 2016.

[35] J. Zeng, B. Zhang, and K. Sreenath. Safety-critical model predictive control with discrete-time control barrier function. In *2021 American Control Conference (ACC)*, pages 3882–3889. IEEE, 2021.

[36] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8660–8669, 2019.