Driving the Future: Sequential Point-cloud processing and it's application in Autonomous Navigation

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

Master of Science in
Computer Science and Engineering
by Research

by

KAUSTAB PAL
2020701021
KAUSTAB.PAL@RESEARCH.IIIT.AC.IN

International Institute of Information Technology
Hyderabad - 500 032, INDIA
February 2024

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Driving the Future: Sequential Point-cloud processing and it's application in Autonomous Navigation" by KAUSTAB PAL, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Advisor: Prof. K. MADHAVA KRISHNA

To everyone who believed in me.

Acknowledgments

I want to express my heartfelt gratitude to Professor K. Madhava Krishna of the Robotics Research Center (RRC) at IIIT Hyderabad. He gave me a priceless opportunity to explore robotics at IIIT Hyderabad, and his role in my academic journey is akin to that of an academic father. He has significantly shaped my understanding and approach to research. Without his support and guidance, I wouldn't have gained the fundamental perspective needed to contribute meaningfully to the research field. I'm also immensely thankful to Professor Avinash Sharma from the 3D Vision Group at CSE@IITJ and CVIT@IIITH for his substantial contributions to enhancing my research capabilities.

I want to express my deepest gratitude to my parents for always supporting me and giving me the freedom to follow my passions. Their love and encouragement mean the world to me. I'm also grateful to my childhood friend Shougandh, who has consistently supported and motivated me to grow. Special thanks to Shwetasree for being an inspiration and to Shreya for bringing positivity into my life. I'm especially grateful to Mr. Sounak Dey from TCS Research for believing in me, mentoring me, and teaching me valuable skills for research and mentorship.

During my time at RRC, I have been fortunate to make some exceptional friends who played a pivotal role in helping me grasp and understand the intricate concepts within my field and also helped me be a better person in life. My heartfelt gratitude extends to Aditya, Swati, Jhanvi, Rishabh, Omama, Sudarshan & Vaishnavi, all of whom not only supported but also believed in me and consistently pushed me beyond my comfort zone. I also had the honor of mentoring Sarath and Himani, and my gratitude extends to them. They helped me develop my teaching skills, and I'm thankful for that experience.

Finally, I wish to express my gratitude to the institute for providing a platform for aspiring researchers to pursue their dreams and personal growth. Furthermore, I would like to acknowledge the thesis examiners for their invaluable feedback, which significantly contributed to the enhancement of this thesis.

Once again, I extend my heartfelt appreciation to everyone involved.

Abstract

Advancements in 3D sensing, driven by the adoption of LiDAR technology, have reignited innovation in autonomous navigation. LiDAR sensors offer real-time, large-scale point clouds, surpassing traditional vision solutions. Datasets like nuScenes and KITTI empower researchers in tasks such as Localization, Place Recognition, and Obstacle Trajectory Prediction. This thesis contributes by exploring the modeling and prediction of large-scale point cloud sequences. Additionally, it showcases a practical application, representing point-cloud sequences as occupancy grid maps and generating trajectories for autonomous navigation. This dual focus enhances our understanding of autonomous navigation complexities, providing valuable insights into real-world implementation challenges.

The first study introduces ATPPNet, an innovative architecture tailored to predict future point cloud sequences using Conv-LSTM, channel-wise and spatial attention, and a 3D-CNN branch. Extensive experiments conducted on publicly available datasets demonstrate the model's impressive performance, outperforming existing methods. The thesis includes a comprehensive ablative study of ATPPNet and an application study showcasing its potential for tasks such as odometry estimation.

In the second study, the thesis proposes NeuroSMPC, a novel integration of data-driven frameworks with sampling-based optimal control for real-time applications, particularly on-road autonomous driving. The 3D-CNN layers in NeuroSMPC predicts optimal mean control without iterative resampling, reducing computation time. The approach proves effective in generating diverse control samples around the predicted optimal mean, facilitating real-time trajectory rollout in the presence of dynamic obstacles. The 3D-CNN architecture implicitly learns future trajectories of dynamic agents, ensuring collision-free navigation without explicit future trajectory predictions. Performance gains are demonstrated over multiple baselines in on-road scenes through closed-loop simulations in CARLA. Real-world applicability is showcased by deploying the system on a custom Autonomous Driving Platform (AutoDP). These studies collectively advance the understanding and implementation of autonomous navigation systems in dynamic environments.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Advancements in technology have led to a growing use of robotics in research labs and industry. These robots/agents often require to autonomously navigate in their environments to reach their designated goals. This research direction, widely explored in robotics, has applications in various domains, including autonomous aerial/aquatic drones, vehicles, and mobile robots. The task of autonomous navigation has three main stages: Perception, Planning and Control. This thesis primarily focuses on the Perception and Planning stages of autonomous navigation.

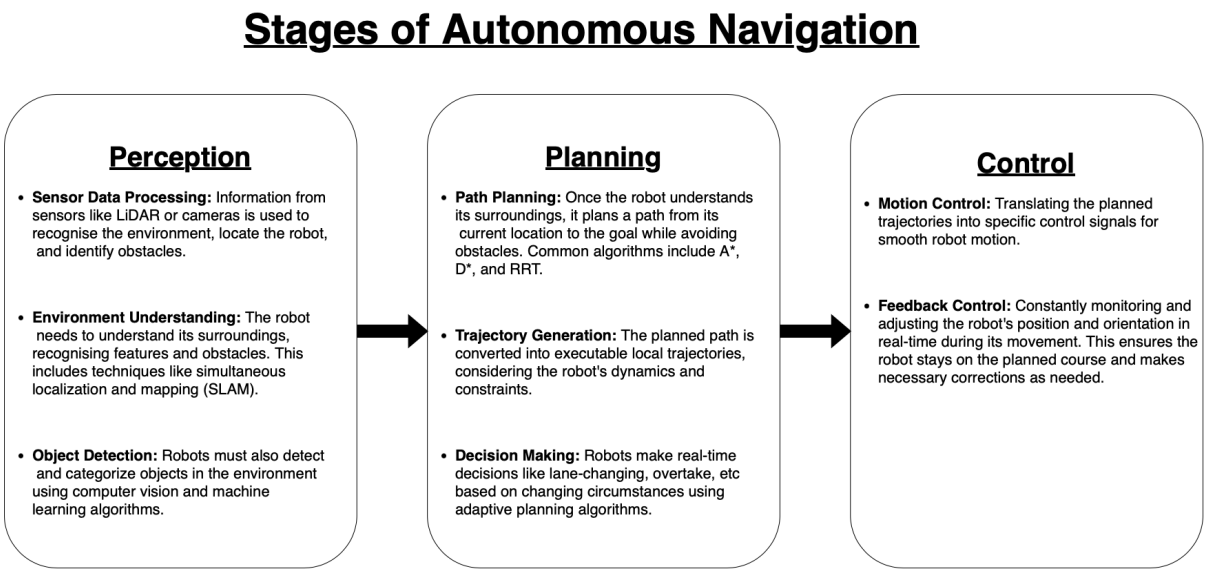


Figure 1.1: The three stages of autonomous navigation

Recent advancements in 3D perception, propelled by commercial LiDAR (Light Detection and Ranging) technology, have led to substantial progress in perception systems. These advancements enable LiDAR sensors to produce real-time, large-scale sequential point clouds,

providing a high-fidelity perception of the 3D world compared to traditional monocular/stereo-based vision solutions. The availability of such extensive data, exemplified by datasets like nuScenes [2] and KITTI [1], has empowered researchers to tackle complex tasks such as Localization, Place Recognition, Segmentation, Obstacle Trajectory Prediction and Autonomous Navigation. The majority of existing methods addressing these tasks rely on captured sequential point clouds within a specified temporal window from the recent past. The planning and decision making tasks mainly makes use of the detect then forecast framework to first detect the objects in the captured sequential point clouds and then forecast how their poses will evolve. This decoupled approach cannot be scaled as it is expensive in nature. An alternative way to address this issue is to first generate a spatio-temporal model of the observed sequential point-clouds. This model inherently encapsulates information about the scene's dynamics and can predict how the scene will evolve by generating future sequences of point-clouds.

Modeling sequences of large-scale point clouds presents unique challenges. A primary obstacle arises from the inherent unordered nature of point clouds in the spatial dimension, despite their temporal order. Additionally, the varying sampling sizes further complicate the task of capturing spatio-temporal coherence among these point clouds. This complexity renders traditional architectures for feature encoding (such as CNNs) and sequence prediction (like LSTMs) unsuitable, as they are not designed to process spatially unordered data. Another significant challenge stems from the extreme sparsity of LiDAR point clouds, making it difficult to represent spatial geometric structures accurately. The inherent noise in sensing introduces additional hurdles, especially in scenes with significant clutter. Furthermore, the memory-intensive nature of extracting features from sequences of full-scale point clouds, each containing over 100,000 points, adds to the computational demands.

Despite these challenges, once a spatio-temporal modeling approach generates a feature vector representing observed point cloud sequences, it proves versatile for various applications. One noteworthy application is autonomous navigation, where the spatio-temporal model can be used to facilitate the planning of a collision-avoiding trajectory. Different strategies exist for trajectory planning, such as generating multiple trajectory proposals [3–5] or incorporating sampling-based optimization paradigms. However, achieving real-time trajectory rollout remains a challenge in this context. Addressing these challenges is essential for advancing the capabilities of autonomous navigation systems based on spatio-temporal models.

In section 1.1, a discussion has been made about the existing methods that spatio-temporally models sequential 3D data and the gaps in the literature. These methods leverage the derived feature vector to predict future sequences of point clouds, thereby validating the effectiveness of the learned model. Subsequently, a discussion was presented on trajectory generation methods that leverage a representation of observed sequences of point-clouds and the current gaps

in the literature. Section 1.2 discusses briefly about the contributions in this thesis followed by section 1.3 discussing about the organisation of this thesis.

1.1 Related Work

Traditionally, $3D$ data is processed with deep learning encoders using volumetric [6–8], point-cloud [9, 10] and multi-view projection [11–13] methods.

In regards to *spatio-temporally modelling sequences of observed point clouds and predicting future point cloud sequences*, primarily two lines of work exist, focusing on point cloud and range image representation. The existing point cloud prediction methods either reformulate the task as scene flow estimation [14] or employ RNN kind of temporal prediction [15, 16]. The former predicts just a translation of the $3D$ points and hence does not represent the future point cloud accurately. At the same time, the latter works on down-sampled point clouds (for memory efficiency reasons) thereby limiting the resolution of $3D$ data.

On the other hand, range image based representations project the point cloud data to a $2D$ virtual image plane of the LiDAR sensor, thereby retaining only the single (closest, farthest, or average) depth of the scene for every pixel. Early work with this representation [17] used LSTMs to process the temporal sequences and predict a sequence of future range images. [18] used $3D$-CNNs with circular padding and skip-connections to predict a sequence of future range images while [19] used Conv-LSTMs on each of the features from the convolution encoder for the prediction task. However, their network is cumbersome and they use the auto-regressive approach for prediction of range images. Recent work in [20] uses the self-attention mechanism of Transformers along with a semantic-based loss function. This method compresses the $3D$ tensors into height and width dimensions and processes each of them separately using two separate transformer blocks. As a result, they are using self-attention only on the channels and since they are compressing the feature tensor into height and width they are also losing the spatial context. Additionally, their model size in terms of the number of parameters is large.

In the realm of *autonomous navigation* literature, significant attention is devoted to layout representations [21–23], agent trajectory prediction [24, 25], and end-to-end trajectory generation [4, 23]. Most of the trajectory generation frameworks involve choosing the best possible trajectory out of an elite set that is typically obtained from large dataset of driving examples or expert trajectories [4, 23]. However the expert trajectories despite the diversity can be sub-optimal and need not be the best response for a given input scenario. Most of these methods do not show their trajectory planning in closed loop simulation scenarios as they are evaluated

in pre-recorded datasets that prevent perception and planning in coupled closed loop settings. Also none of the above methods showcase their formulation on a real self driving vehicle in real world on-road scenes.

Elsewhere, in the manipulation planning community, sampling based optimal control [26, 27] have become popular essentially due to the derivative free blackbox optimization feature of such frameworks. Such sampling based trajectory optimization methods suffer from the curse of time complexity with a number of variants being proposed to overcome this disadvantage. For instance [28] resorts to efficient parallelization of controls while [29] contributes through sample efficient methods. Whereas in [30] gradient based updates of sample parameters leads to enhanced performance. Nonetheless these methods still need iterative improvement of the sample parameters to reach optimum mean control values that preclude their adaptation to on-road real time autonomous driving applications.

1.2 Contributions

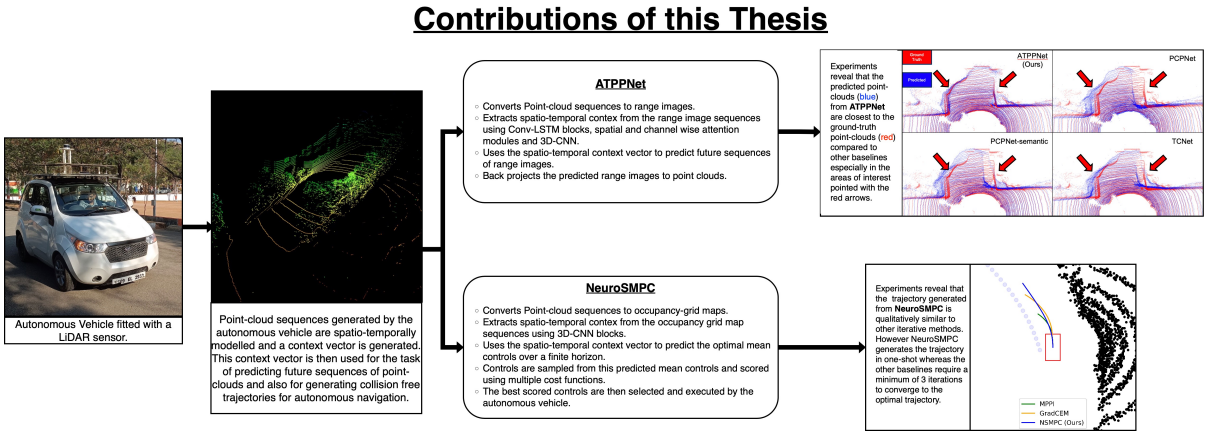


Figure 1.2: Contributions of this thesis: ATPPNet and NeuroSMPC

Based on the limitations of the existing literature in the field of *spatio-temporally modelling sequences of observed point-clouds and predicting future sequences of point-clouds*, a new framework ATPPNet (*ATPPNet: Attention based Temporal Point cloud Prediction Network*) has been proposed in chapter 2. ATPPNet leverages Conv-LSTM [31] blocks along with channel-wise and spatial attention modules for extracting an enhanced spatio-temporal context for the task of future point cloud prediction. Further, a complimentary $3D$-CNN branch is also leveraged to spatio-temporally encode the global feature embeddings of the range images. Additionally, the re-projection mask associated with the predicted range images are also predicted

to retain only the valid range values when re-projecting to the point cloud. In contrast to the approach presented in [20], the processing of range image sequences using Conv-LSTM, along with the application of spatial and channel-wise attention directly on learned spatio-temporal 3D features, demonstrates superior performance without the need for a separate semantic-based loss function. The proposed architecture achieves state-of-the-art performance on two publicly available datasets and yields real-time future point cloud prediction (faster than a typical rotating $3D$ LiDAR sensor point cloud rate i.e., 10Hz). Thorough qualitative and quantitative evaluations have been performed and a detailed ablation study has also been provided to validate the effectiveness of the proposed architecture.

In contrast to the existing literature on *trajectory generation*, a novel framework NeuroSPMC was proposed in chapter 3 for real-time collision free trajectory generation by modelling sequences of observed point-clouds represented as occupancy grid maps. The spatio-temporal context vector, derived from these sequences of observed occupancy grid maps, is then used to predict optimal mean controls over a finite horizon. The predicted mean controls are used to sample controls, which are subsequently scored using multiple cost functions. The best-scored controls are selected and executed by the autonomous vehicle. Real time closed loop simulations were shown on a number of CARLA scenes along with a closed loop implementation on the autonomous vehicle AutoDP to achieve point to point on-campus autonomous driving. Moreover the low latency control rollout facilitates navigation in dynamic scenes without collisions despite the lack of explicit trajectory prediction into the future of the dynamic actors.

1.3 Organization of the Thesis

The thesis is organized into four chapters. A brief summary of each chapter is mentioned below.

- **Chapter 1:** This is an introductory chapter detailing the motivation behind the research, the contributions of the thesis and its outline.

- **Chapter 2:** This chapter introduces ATPPNet for modelling observed sequences of point-cloud and predicting future sequences of point-clouds. The architecture of ATPPNet was explained and multiple experiments reveal that ATPPNet achieves SOTA performance on various publicly available datasets while beating the existing methods by $8 - 10\%$ margin

- **Chapter 3:** This chapter introduces NeuroSMPC that uses $3D$-CNN layers to model observed sequences of point-clouds represented as occupancy grid maps and predicts the optimal mean control over a finite horizon.The performance gain of NeuroSMPC was shown over multiple baselines in a number of on-road scenes through closed loop simulations in CARLA. The real world applicability of the proposed system was also showcased by running it on a custom Autonomous Driving Platform (AutoDP).
- **Chapter 4:** This chapter concludes the thesis by summarizing the various contributions brought out by this thesis.

Chapter 2

ATPPNet: Attention based Temporal Point cloud Prediction Network

2.1 Introduction

Predicting future point clouds is an important yet challenging task in the field of autonomous driving. The goal is to predict future point cloud sequences that not only preserves object structures but also accurately depict their temporal motion. The foresight provided by these predicted point clouds proves invaluable for subsequent tasks such as object trajectory estimation to facilitate collision avoidance or estimating locations with the least odometry drift.

However, the task of predicting future point clouds comes with its own set of challenges. One key challenge is that the point clouds are unordered in the space dimension (albeit ordered temporally) and vary in sampling size hence it is difficult to model spatio-temporal coherence among them. Traditional architectures designed for feature encoding, such as Convolutional Neural Networks (CNNs), and sequence prediction (e.g., LSTMs), cannot be directly applied due to their incapacity to process spatially unordered data. Another key challenge is that the LiDAR point clouds are extremely sparse making it difficult to capture the geometrical structures of the objects in the scene and hence predicting them in the future timesteps is extremely difficult. The noise in sensing puts additional challenges in the perception of real-world scenes where objects are largely cluttered. Additionally, the sheer volume of data in each full-scale point cloud, exceeding 100,000 points, intensifies the memory requirements for extracting features from these sequences.

Addressing these challenges, the proposed ATPPNet architecture employs a novel approach to predict future point cloud sequences given a sequence of previous time step point clouds obtained with LiDAR sensor and represented as range images. ATPPNet leverages Conv-

LSTM [31] blocks along with channel-wise and spatial attention modules for extracting an enhanced spatio-temporal context for the task of future point cloud prediction. Further, the proposed architecture leverages a complimentary $3D$-CNN branch to spatio-temporally encode the global feature embeddings of the range images. Additionally, ATPPNet also predicts the re-projection mask associated with the predicted range images to retain only the valid range values when re-projecting to the point cloud.

To summarize, the main contributions are as follows:

- A novel architecture, ATPPNet, has been proposed to predict future point clouds from a sequence of past point clouds. ATPPNet leverages Conv-LSTM along with spatial and channel-wise attention for this predictive task. ATPPNet achieves SOTA performance on various publicly available datasets while beating the existing methods by $8 - 10\%$ margin.

- The empirical results demonstrate that ATPPNet significantly enhances the performance of downstream tasks, like odometry estimation.

The rest of the chapter is organised as follows: Section 2.2 describes the architecture of the proposed network in detail; Section 2.3 details the experiments and the analyses the results of the experiments.

2.2 Methodology

The proposed ATPPNet (Attention-based Temporal Point cloud Prediction Network) leverages Conv-LSTM blocks along with channel-wise and spatial attention modules, complemented by a 3D-CNN branch, to process a past sequence of 3D point clouds and predict future point clouds.

In a temporal window, let $S_P = \{S_{t-M+1}, S_{t-M+2}, \ldots, S_t\}$ represent the input sequence of $3D$ point clouds spanning M time steps. Each $S_\tau \in \mathbb{R}^3$ corresponds to a set of $3D$ points captured at a specific time step τ. The objective of the proposed ATPPNet is to forecast the future sequence of $3D$ point clouds for a temporal window encompassing N time steps, denoted as $S_F = \{S_{t+1}, S_{t+2}, \ldots, S_{t+N}\}$.

Following the approach in [18], the range image representation is adopted by initially converting the point clouds into the spherical coordinate system. Subsequently, the resulting point cloud $S_\tau \in \mathbb{R}^3$ is projected onto the virtual image plane of the LiDAR sensor, represented as $R_\tau \in \mathbb{R}^2$.

Let $R_P = \{R_{t-M+1}, R_{t-M+2}, \ldots, R_t\}$ denote the sequence of past range images within a fixed temporal window, obtained from S_P. Similarly, $R_F = \{R_{t+1}, R_{t+2}, \ldots, R_{t+N}\}$ represents the sequence of predicted range images corresponding to S_F.

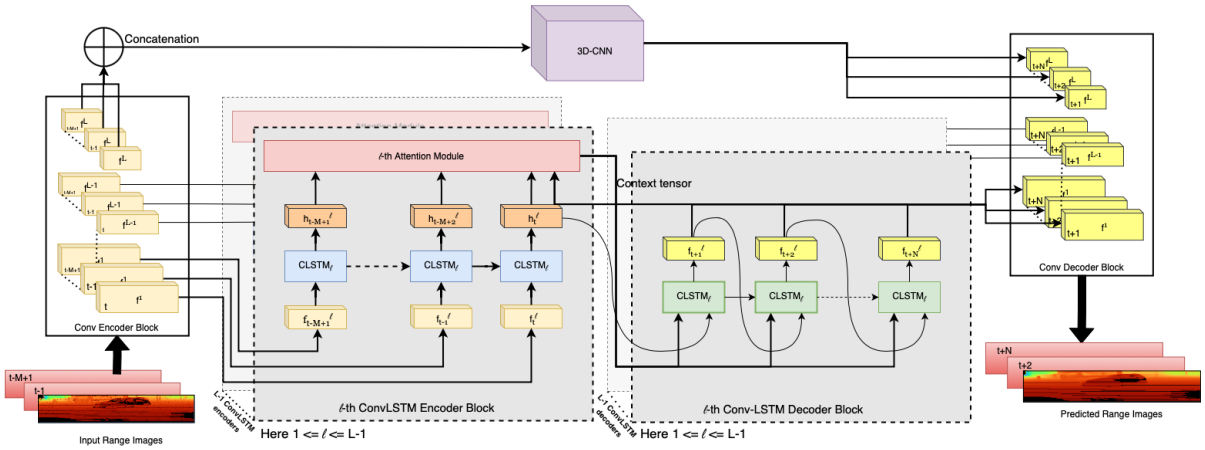


Figure 2.1: **ATPPNet Architecture.** ATPPNet leverages Conv-LSTM along with channel-wise and spatial attention dually complemented by a 3D-CNN branch for extracting an enhanced spatio-temporal context to recover high quality fidel predictions of future point clouds.

2.2.1 Overall Architecture

The overview of the ATPPNet architecture is depicted in Figure 2.1. A convolution encoder shared across the network processes the input range images R_P and produces a set of multi-scale feature tensors for each range image. Subsequently, the first $L-1$ feature tensors are input to Conv-LSTMs, enabling the modeling of spatio-temporal relationships across R_P.

Furthermore, spatial and channel-wise attention mechanisms are applied to the Conv-LSTM outputs, resulting in $L-1$ context tensors. These context tensors represent a consolidated spatio-temporal encoding of R_P. Additionally, for the final L-th feature tensor, a $3D$-CNN layer processes the spatio-temporal relationship, generating the L-th feature tensor for the subsequent N time steps.

On the decoder side, the context tensor, along with the hidden state from the last time step, is provided to $L-1$ Conv-LSTMs. These Conv-LSTMs generate feature tensors for each of the $L-1$ layers, projecting N time steps into the future. All L feature tensors on the decoder side are subsequently processed to generate the range image sequence R_F and their corresponding re-projection masks M_F for all N future time steps. Each pixel $M_\tau \in M_F$ can be interpreted as the probability for each range image pixel to be valid or invalid.

This re-projection mask is utilized during the back-projection of a range image to a point cloud, where only the range values corresponding to probabilities greater than 0.5 are retained. The specific construction details of these architectural blocks are outlined below.

2.2.2 Convolution Encoder & Decoder block

The convolution encoder block takes the range image and first performs a $2D$-convolution operation, resulting in a tensor with an increased number of channels but the same spatial resolution. This tensor is further processed using L convolutional sub-blocks.

Each sub-block takes as input a feature tensor and applies a combination of $2D$-convolution, $2D$-batch normalization, and leaky-ReLU operation while keeping the tensor dimensions the same. A strided-convolution operation is subsequently performed, resulting in a down-sampled tensor.

The convolution decoder block follows the reversed structure of the convolution encoder block. There are L sub-blocks, each of which takes an input tensor and passes it through a $2D$-transposed convolution, $2D$-batch normalization, and leaky-ReLU operation resulting in a spatially scaled-up tensor while keeping the number of channels the same.

Another $2D$-convolution operation is then performed, to decrease the channel size of the tensor. The output of the L-th layer is finally passed through another $2D$-convolutional layer resulting in the predicted range images and the associated re-projection mask.

2.2.3 Conv-LSTM encoder

The utilization of $L-1$ Conv-LSTMs is proposed to leverage the spatio-temporal context inherent in input sequences. Following the architectural paradigm of S2Net [19], multiple Conv-LSTM layers are employed for each of the feature tensors, aiding in the preservation of high-frequency details across the range image sequences. The Conv-LSTMs corresponding to each of the $L-1$ features generate a hidden state for every time step within the M sequence.

2.2.4 Attention Module

Let $\tau \in [t-M+1, \ldots, t-1, t]$ be a specific time step in the given input temporal window, and the feature tensors for the layer l at every time step be represented as f_τ^l. Similarly, let the output of l-th Conv-LSTM at time step t denoted as g_t^l where $l \in [1, \ldots L-1]$. As part of the attention module, the joint embedding $\mathcal{J}\tau^l$ of $f\tau^l$ and g_t^l is computed using the formulation from [32]:

$$\mathcal{J}_\tau^l = \sigma_1(W_f f_\tau^l \oplus W_g g_t^l),$$

where σ_1 is a non-linear activation function chosen as ReLU and $\oplus$ is the concatenation operation. W_f and W_g are implemented as $2D$-convolution operations with 1×1 kernel. The use of σ_1, W_f, and W_g allows the network to learn non-linear relationships between the features, which is especially important when the image is noisy like our range images. The resulting tensors $\mathcal{J}_\tau^l$ are passed through the spatial and channel-wise attention module [33] to find the $3D$ attention map $\mathcal{M}(\mathcal{J}_\tau^l) \in R^{C_l \times H_l \times W_l}$. The refined feature tensor for layer l at time step τ is computed as:

$$\hat{f}_\tau^l = f_\tau^l \otimes \mathcal{M}(\mathcal{J}_\tau^l).$$

Here $\otimes$ is the element-wise multiplication. To compute the $3D$ attention map $\mathcal{M}(J_\tau^l)$, the channel-wise attention and spatial attention are computed separately and then combined as

$$\mathcal{M}(\mathcal{J}_\tau^l) = \sigma(M_c(\mathcal{J}_\tau^l) \otimes M_s(\mathcal{J}_\tau^l))a,$$

where σ is the Sigmoid function and $\otimes$ is the element-wise multiplication operation. The M_c function first applies the global average pooling operation on the $3D$ tensor $\mathcal{J}_\tau^l$ to get the channel tensor which is then passed through an MLP layer to get the channel-wise attention values. The M_s function applies $2D$-convolution operation on the $3D$ tensor $\mathcal{J}_\tau^l$ and returns a single channel tensor which represents the spatial attention values.

The refined feature tensors for all the M time steps for each of the $L-1$ layers are then used to compute the context tensors, that are subsequently served as input to the decoder Conv-LSTMs.

$$context_l^t = \sum_{\tau=t}^{t-M+1} \hat{f}_\tau^l$$

2.2.5 Conv-LSTM decoder

The Conv-LSTM decoder follows a similar structure as the Conv-LSTM encoder. $L-1$ Conv-LSTM decoders are used to predict $L-1$ feature tensors for each of the N future time steps. Let $\tau \in [t+1, \ldots, N]$ be a specific time step in the predicted future temporal window. For the τ-th time step, the l-th Conv-LSTM decoder takes as input the context tensor $context_l^{\tau-1}$ where $l \in [1, \ldots, L-1]$ along with the hidden state of the Conv-LSTM for time step $\tau-1$ to compute the output feature tensor. This output feature tensor along with the hidden states of the previous time steps are used to re-compute the context tensor $context_l^\tau$ to be used

for the next time step. These output feature tensors on the Conv-LSTM decoder side are used by the convolutional decoder to generate the predicted range images R_F

2.2.6 $3D$-CNN block

The feature tensors for the L^{th} layer from the convolutional encoder block for all the past M time steps are concatenated to create a $4D$ tensor. A $3D$-CNN layer is used to process this feature tensor and generate N feature maps for the L-th convolutional decoder block. It's crucial to note that due to the spatial focus tendencies of $3D$-CNNs, which concentrate on fewer, contiguous areas within feature tensors [34], a $3D$-CNN is applied specifically to the last L-th layer of the feature tensor (obtained with $2D$-CNN), and is chosen for its ability to capture global structures in the range image [35], [36]. Thus, the $3D$-CNN block extracts only the complementary spatio-temporal context as the primary spatio-temporal context is already obtained by applying Conv-LSTM's on the initial layers of the convolutional encodings as they tend to capture the high frequency details in the range images. This also gives us the additional advantage of speeding up our inference time.

2.2.7 Loss Function

A combination of losses are used when training the network. Since the ground truth point clouds are projected onto $2D$ range images of dimension $H \times W$, $2D$ image-based losses can be used.

Firstly, the average range loss $\mathcal{L}R$ is employed to calculate the error between the predicted range values $\hat{r}_{\tau,i,j} \in \mathbb{R}^{N\times H\times W}$ and the corresponding ground-truth range values $r_{\tau,i,j} \in \mathbb{R}^{N\times H\times W}$. The formulation for the average range loss is as follows:

$$\mathcal{L}_R = \frac{1}{N \times H \times W} \sum_{\tau=t+1}^{t+N} \sum_{i=1}^{H} \sum_{j=1}^{W} || \hat{r}_{\tau,i,j} - r_{\tau,i,j} ||_1,$$

where $|| \bullet ||_1$ represents the L_1 norm. The range loss $\mathcal{L}_R$ is computed only for the valid ground truth points. To train the re-projection mask output, the Binary Cross-Entropy loss is used between the predicted mask values $\hat{m}_{\tau,i,j} \in \mathbb{R}^{N\times H\times W}$ and the ground truth mask values $m_{\tau,i,j} \in \mathbb{R}^{N\times H\times W}$. The average mask loss $\mathcal{L}_M$ is computed as:

$$\mathcal{L}_M = \frac{1}{N \times H \times W} \sum_{\tau=t+1}^{t+N} \sum_{i=1}^{H} \sum_{j=1}^{W} -m_{\tau,i,j} \log \hat{m}_{\tau,i,j} \tag{2.1}$$

$$- (1 - m_{\tau,i,j}) \log(1 - \hat{m}_{\tau,i,j}), \tag{2.2}$$

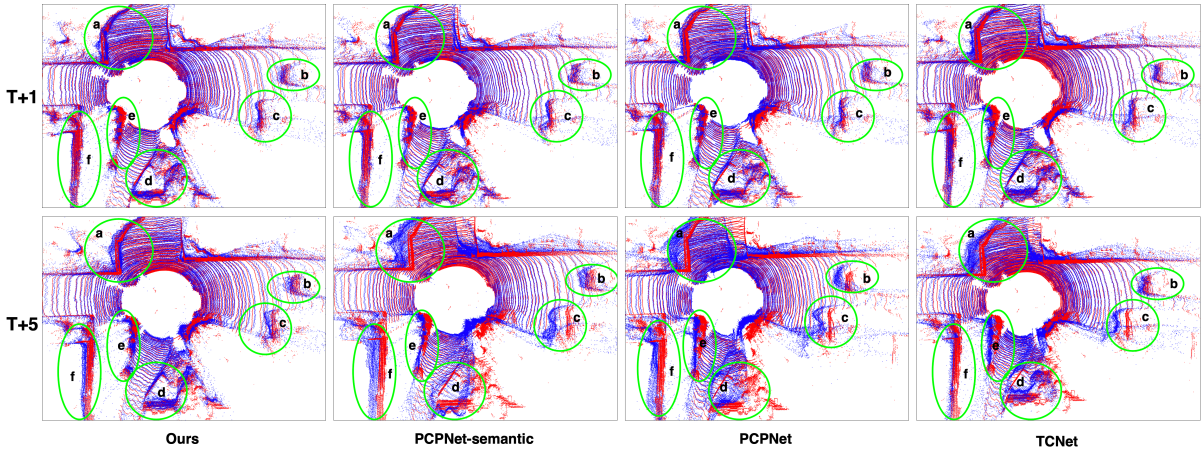


Figure 2.2: Qualitative comparison conducted on sequence 10 of the KITTI odometry dataset. The predicted points (blue) and the ground truth points (red) are combined for a better visual comparison. The top row shows the point clouds at prediction step $t+1$ and the bottom row shows the point clouds at prediction step $t+5$. The areas of interest are circled in green.

where $\hat{m}_{\tau,i,j}$ is the predicted probability of whether the range value is valid. $m_{\tau,i,j}$ is 1 if the ground-truth range value is valid and 0 otherwise. A masked range image is generated by taking only the range values from the range image whose corresponding mask values are greater than 0.5. Since the predicted masked range images are re-projected into point clouds,Chamfer distance [37] represented as $\mathcal{L}_C$ is used for evaluating fidelity of the predicted point clouds.

The combined loss function is given as

$$\mathcal{L} = \mathcal{L}_R + \mathcal{L}_M + \alpha_C \mathcal{L}_C.$$

α_C is the weight associated with the Chamfer distance.

2.3 Experiments and Results

2.3.1 Experimental Settings

ATPPNet is trained in a self-supervised manner in the sense that only the sequential point cloud data scans are used and no manually annotated labels are required. For the experiments, the temporal window size has been kept as $M = N = 5$. In the convolutional encoder block, the initial convolutional operation outputs 16 channels while retaining the spatial dimension. $L = 4$ sub-blocks were used where the channel size increases by a factor of 2 for every successive sub-block obtained by the convolutional encoder. In each of the sub-blocks, the first

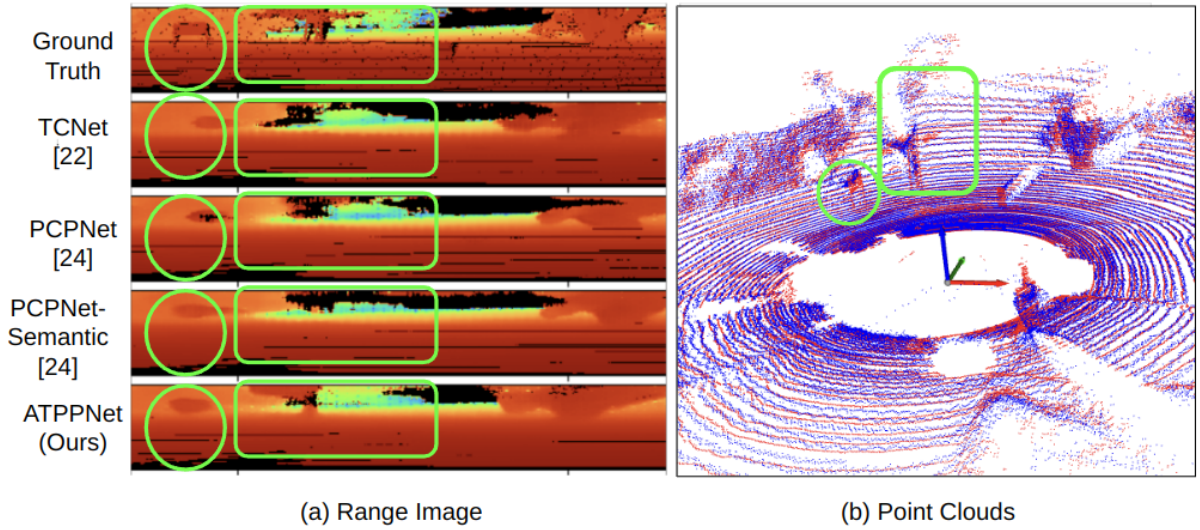


Figure 2.3: (a) Predicted range images by our ATPPNet and existing methods in comparison to ground truth and, (b) the $3D$ rendering of the predicted point cloud by ATPPNet (blue) and ground-truth (red). Green circle/rectangle highlights regions where ATPPNet's predictions are superior.

convolutional operation uses a kernel size of 3×3 with stride $(1, 1)$, and the second convolution operation uses a kernel size of 2×4 with stride $(2, 4)$. All the convolutional operations use circular padding [18]. Each Conv-LSTM block uses 3 layers.

Similar to the trend in the literature [18, 20], the proposed architecture was trained for 50 epochs with $\alpha_C = 0$ and then fine-tuned with the Chamfer distance loss for the next 10 epochs by setting $\alpha_C = 1$. The proposed model is trained on a system with an Intel Xeon E5-2640 CPU and 3 Nvidia RTX 2080 GPUs using the Distributed Data Parallel strategy. While training, the ADAM optimizer [38] was used with default parameters along with an initial learning rate of 0.0003 and the StepLR learning rate scheduler with gamma as 0.99.

2.3.1.1 KITTI Odometry dataset [1]

The training data consists of sequences $00 - 05$, the validation set comprises sequences $06 - 07$, and sequences $08 - 10$ are designated for testing. In the KITTI dataset [1], the LiDAR system encompasses 64 channels. Consequently, range images of size 64×2048 have been utilized in our approach.

Prediction Step	TCNet [18]	PCPNet [20]	PCPNet-Semantic [20]	ATPPNet (Ours)
1	0.554	0.543	0.503	**0.468**
2	0.671	0.662	0.620	**0.570**
3	0.779	0.773	0.727	**0.667**
4	0.878	0.872	0.825	**0.760**
5	0.974	0.973	0.920	**0.851**
Mean	0.771	0.765	0.719	**0.663**

Table 2.1: Range Loss results on the KITTI odometry test set verifies that ATPPNet has a performance improvement of 4.026% over SOTA on the mean range loss. Bold values correspond to the best performing model in that corresponding time step.

2.3.1.2 nuScenes dataset [2]

The proposed network was trained on this dataset employing a training strategy similar to that used on the KITTI dataset. In line with PCPNet [20], sequences $00-69$ were employed for training, $70-84$ for validation, and $85-99$ for testing. The architecture was trained on range images with dimensions 32×1024, since the LiDAR system in use possesses 32 channels.

2.3.2 Qualitative Analysis

Figure 2.2 (and Figure 2.3) shows a qualitative comparison of the predicted point clouds generated using the proposed ATPPNet and the other methods: TCNet [18], PCPNet and PCPNet-semantic [20]. The areas of interest are highlighted with numbered green circles.

In the predicted sequence $t+1$ shown in Figure 2.2, it can be observed over the circles a, b, c & d that ATPPNet outperforms the other methods by generating point clouds that are less noisy and structurally more similar to the ground truth. It can be observed in time step $t+5$ (bottom row) that the circles numbered a, b, c, d, e & f in the point cloud generated by ATPPNet is more fidel to the ground truth compared to the predicted point clouds from the other methods that have large visible deviations from the ground truth and are more noisy.

	Sampled Point Cloud						Full-Scale Point Clouds			
Prediction Step	Point-LSTM [15]	MoNet [16]	TCNet [18]	PCPNet [20]	PCPNet-Semantic [20]	ATPPNet (Ours)	TCNet [18]	PCPNet [20]	PCPNet-Semantic [20]	ATPPNet (Ours)
1	0.332	0.278	0.290	0.285	0.280	**0.258**	0.253	0.252	0.242	**0.225**
2	0.561	0.409	0.357	0.341	0.340	**0.311**	0.309	0.301	0.298	**0.270**
3	0.810	0.549	0.441	0.411	0.412	**0.375**	0.377	0.362	0.354	**0.326**
4	1.054	0.692	0.522	0.492	0.495	**0.445**	0.448	0.435	0.427	**0.391**
5	1.299	0.842	0.629	0.580	0.601	**0.523**	0.547	0.514	0.503	**0.461**
Mean	0.811	0.554	0.448	0.422	0.426	**0.382**	0.387	0.373	0.365	**0.335**

Table 2.2: Chamfer distance results on KITTI Odometry test sequence with the sampled point clouds on the left and full-scale point clouds on the right. ATPPNet has a performance improvement of 10.328% over SOTA for the sampled point clouds and 8.219% over SOTA for the full-scale point clouds. Bold values correspond to the best performing model in that corresponding time step.

2.3.3 Quantitative Analysis

In this section, a quantitative analysis of the proposed ATPPNet has been performed with two point based methods (PointLSTM [15], MoNet [16]) on the KITTI [1] test set, and three range image based methods (TCNet [18], PCPNet and PCPNet-semantic [20]) on the KITTI [1] and the nuScenes [2] test set. The point based methods [15, 16] use down-sampled point clouds to 65536 points and this is also adopted by ATPPNet and other methods i.e., [18, 20]. The range loss and Chamfer distance were used to evaluate the predicted range images and the point clouds, respectively.

Table 2.1 shows the quantitative results of the range loss for all the methods on the KITTI test set. Compared to the other methods, ATPPNet generates better range images as the prediction time step increases, which can also be verified by the improvement of 4.026% over SOTA (PCPNet-semantic [20]) in the mean range loss.

In Table 2.2, the Chamfer distance has been evaluated on the sampled point clouds (left column) and full-scale point clouds (right column) on the KITTI test set. As can be observed, the proposed method is having an improvement of 10.328% over SOTA on sampled point clouds and an improvement of 8.219% over SOTA on full-scale point clouds. It can also be observed that the margin of Chamfer distance between ATPPNet and other methods increases as the pre-

Evaluation metric	TCNet [18]	PCPNet-Semantic [20]	ATPPNet (Ours)
Mean Chamfer Distance	1.389	1.360	**0.932**
Mean Range Loss	0.719	0.704	**0.598**

Table 2.3: Mean Range Loss and Chamfer distance on the nuScenes test set. ATPPNet is making an improvement of 15.056% over SOTA on the mean Range loss and 31.470% over SOTA on the mean Chamfer distance.

	A) Attention Module			**B) Conv-LSTM (CLSTM) layers**			
Evaluation metric	No Attention	S-Attention	C-Attention	$L-1$ CLSTM	$L-1$ & $L-2$ CLSTM	all L CLSTM	ATPPNet (Ours)
Chamfer distance	0.365	0.359	0.356	0.405	0.378	0.366	**0.335**
Range loss	0.719	0.687	0.690	0.770	0.698	0.717	**0.663**

Table 2.4: Results of Ablation study on Attention Module and Conv-LSTM layers. Bold values correspond to the best performing model.

diction time step increases (i.e., farther in future). This indicates a more stable prediction of the point clouds across all the time steps as depicted in Figure 2.2.

In Table 2.3, the quantitative analysis of the proposed model trained on the nuScenes dataset has been reported. ATPPNet is improving 15.05% on the mean range loss and 31.47% on the mean Chamfer distance over SOTA. The inference time on the KITTI and nuScenes dataset is 89.5 ms and 70.7 ms respectively.

2.3.4 Ablation Study

This section presents a comprehensive analysis of the significance of various blocks within the proposed architecture, focusing on their impact on the KITTI test set. The effectiveness of the proposed method is systematically demonstrated through this investigation.

Window size	3	5	7
T+1	**0.221**	0.255	0.258
T+2	0.274	**0.270**	0.306
T+3	0.344	**0.326**	0.363
T+4	NA	**0.391**	0.426
T+5	NA	**0.461**	0.498
T+6	NA	NA	0.580
T+7	NA	NA	0.657

Table 2.5: An ablation study on how the performance changes as we vary the sequence length.

Impact of Attention Module: In Table 2.4 column A, the results of our ablation study on the attention module are shown. To conduct this study, 3 experiment were set up: (1) removing the attention module (column "No Attention"), (2) using just spatial attention (column "S-Attention") and (3) using just channel-wise attention (column "C-Attention"). It can be observed in all the 3 experiments that the range loss and Chamfer distance deteriorates as compared to our original method.

Effects of Spatio-Temporal Modelling: In Table 2.4 column B, the impact of varying the number of feature tensors from the convolutional encoder to be modelled spatio-temporally has been demonstrated. For this, three experimental setups were adopted: (1) modelling only the $L-1$-th feature tensor with Conv-LSTM and L-th tensor with $3D$ CNN. (column "$L-1$ CLSTM"), (2) modelling only the $L-1$-th and $L-2$-th feature tensor with Conv-LSTM and L-th tensor with $3D$ CNN. ("$L-1$ & $L-2$ CLSTM"), and (3) modelling all the L layers from the convolutional encoder with Conv-LSTM (column "all L CLSTM"). For the aforementioned experiments a deterioration in the performance compared to the original method was observed. The significance of the $3D$-CNN layer for the L-th layer is further emphasized by the findings from experiment (3). This verifies that $3D$ CNNs are better at modelling contiguous areas in the feature tensors [34] which tend to appear at the lower level features from the convolutional encoder [35], [36].

Impact of Sequence Length: In Table 2.5, the results are presented, showcasing the variations in performance corresponding to different temporal window sizes. It is important to note that the temporal window size is kept the same for both input and output. A decrease in the Chamfer distance as we increase the window size from 3 to 5 can be observed. However, further

Pose error	TCNet [18]	PCPNet [20]	PCPNet-Semantic [20]	ATPPNet (Ours)
$\mathcal{L}_P^{t+1}$	0.1342	0.1363	0.1280	**0.1209**
$\mathcal{L}_P^{t+2}$	0.2412	0.2282	0.2235	**0.2065**
$\mathcal{L}_P^{t+3}$	0.3670	0.3388	0.3343	**0.3038**
$\mathcal{L}_P^{t+4}$	0.5084	0.4630	0.4558	**0.4128**
$\mathcal{L}_P^{t+5}$	0.6736	0.6022	0.5878	**0.5328**
Mean	0.3849	0.35374	0.3459	**0.3154**

Table 2.6: **LOAM pose error.** We adopt LOAM [36] and evaluate the disparity between the motion estimates on ground truth and predictions.

increasing the window size from 5 to 7 leads to an increase in the Chamfer distance. A possible explanation for this is that the window size of 3 is too short to model the spatio-temporal context of the scene while, for the window size of 7, the context length and the prediction horizon is too long.

2.3.5 Results on Downstream Task

In this section, the impact of the proposed model has been analyzed on a downstream task of generating motion estimates (i.e., odometry) for the ego vehicle. LOAM [39] is utilized to evaluate the disparity between motion estimates derived from ground truth and predictions. Let $\hat{p}^\tau \in \mathbb{R}^2$ denote the trajectory pose obtained using predicted point clouds and $p^\tau \in \mathbb{R}^2$ denote the trajectory pose obtained using ground truth point clouds at time step τ, where $\tau \in [t+1, \ldots, t+N]$. The pose error $\mathcal{L}_P^\tau$ is given as:

$$\mathcal{L}_P^\tau = || \hat{p}^\tau - p^\tau ||_2^2 .$$

As reported in Table 2.6, the pose error ($\mathcal{L}_P^\tau$) for ATPPNet is the least as compared to the other methods. This verifies that the improved point cloud prediction from the proposed method translates to a tangible outcome in the form of improved localization vis-a-vis other methods. Additionally, such prediction of localization error can be effectively leveraged by active localization strategies [40] that steer the vehicle to regions where the localization is expected to be better.

Chapter 3

NeuroSMPC: A Neural Network guided Sampling Based MPC for On-Road Autonomous Driving

3.0.1 Introduction

On road autonomous driving entails persistent and consistent real-time decision making in often highly dynamic and evolving scenes. To accomplish this most trajectory planning frameworks employ a scheme of generating multiple candidate trajectory proposals and scoring them according to an appropriate cost function [3–5]. The candidate trajectories are typically obtained from large scale driving data [3] or by sampling from a parameter distribution that defines a trajectory [4].

As an alternative sampling based optimization paradigms have been popular in the high dimensional planning literature that seamlessly integrate dynamical model of the systems over which the trajectory plans are computed. The major concern however here is the disadvantage in rolling out trajectories in real-time.

The proposed novel framework interleaves neural network driven prediction of finite horizon controls with samples drawn from the variance centred around the mean predicted by the network. The deep network outputs a vector of controls that constitute the optimal mean control over a finite horizon. The network is supervised with the best controls from the elite samples of the sampling based control framework [28]. Typically this mean is expected to be close to the global optimum [29, 30]. The best rollout sample is one that optimizes a scoring or cost function detailed later.

The advantages of the original framework are maintained by abstaining from directly executing the network output controls. Instead, the optimal rollout candidate is selected from the samples around the network output controls. Simultaneously, we address the time complexity challenges inherent in such frameworks by opting for the optimal mean obtained from the

network, which is at least three orders faster. It's worth noting that most formulations of this nature typically entail numerous iterations of the following two steps to converge to the optimal control:

- Selection of a subset of the candidate samples (the elite set) based on a scoring function.
- Update of the mean in accordance with the elite samples and re-sampling from the newly updated distribution.

Additionally, the network, employing its spatio-temporal convolution (3D-CNN) architecture, implicitly grasps the temporal evolution of dynamic actions without relying on explicit trajectory prediction. Despite the absence of explicit trajectory prediction, the executed trajectory demonstrates collision-free navigation across a diverse range of on-road scenarios featuring multiple dynamic actors. This success can be attributed to the deep network's capability for extremely low-latency, high-frequency control roll-outs.

To summarize, the main contributions are as follows:

1. A neural network interleaved sampling based optimal control that computes finite horizon control rollout in real-time thereby making it suitable for real-time self driving for on-road scenes.
2. We show 3D convolutions that convolve spatial and temporal components of a time sequenced Birds Eye View (BEV) layout learns implicitly the future trajectories of the dynamic obstacles so much so the planner based on the network output avoids collision with dynamic obstacles despite lack of explicit trajectory prediction into the future.
3. A number of closed loop simulations in diverse scenarios in the CARLA simulator as well as real-time on campus navigation through our AutoDP confirm the efficacy and real-time veracity of the proposed framework.
4. Moreover the comparative analysis on compute time with other sampling based optimal control frameworks [28, 30] clearly depicts the vast performance gain of the proposed method.
5. Furthermore, *RoadSeg* is introduced as a system for real-time road segmentation on board in known environments. It can effectively generate Bird's Eye Views (BEVs) even in setups with limited resources.

The rest of the chapter is organised as follows: Section 3.1 describes the architecture of the proposed network in detail; Section 3.2 details the experiments and the analyses the results of the experiments.

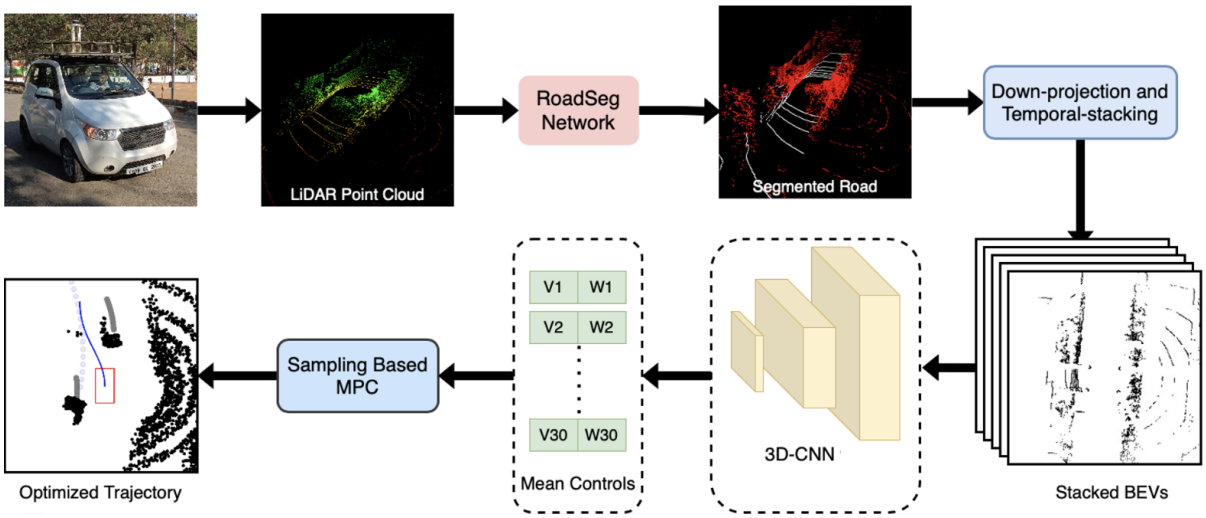


Figure 3.1: **Pipeline:** Each Point-cloud from the AutoDP is first passed through the RoadSeg Network to segment the points that belong to the road. The non-road points are considered as obstacles. A down-projection operation is performed to the obstacle points to create a BEV occupancy grid map. 5 past BEVs are then stacked together and passed through a 3D-CNN architecture to get the mean controls for a short-horizon. These mean controls are then used as the mean of a sampling based MPC to sample more trajectories. The best trajectory is chosen from amongst these trajectories to be executed by our AutoDP.

3.1 Methodology

In the context of on-road driving, given a point cloud and the global path that the AutoDP needs to follow, the objective is to produce occupancy grid maps from the point cloud data. These maps are then utilized to generate controls, directing the AutoDP towards obstacle-free regions on the road, aligning with its global path. The state of the AutoDP at timestep h is represented by the vector $\widetilde{\boldsymbol{x}}_{\boldsymbol{h}} = \begin{bmatrix} x_h, y_h, \theta_h \end{bmatrix}$ where x_h and y_h represents the x and the y coordinate of the vehicle respectively and θ_h is the orientation of the vehicle. The controls for the vehicle at timestep h are represented by the vector $\boldsymbol{u}_{\boldsymbol{h}} = \begin{bmatrix} v_h, \omega_h \end{bmatrix}$ where v_h and ω_h are the velocity and the angular-velocity of the vehicle respectively.

3.1.1 Pipeline

The framework pipeline is illustrated in Fig. 3.1. Initially, the point cloud from AutoDP undergoes segmentation using the RoadSeg Network to isolate road-related points. Points not belonging to the road are identified as obstacles. A down-projection operation is applied to these obstacle points, resulting in a Bird's Eye View (BEV) occupancy grid map. This process

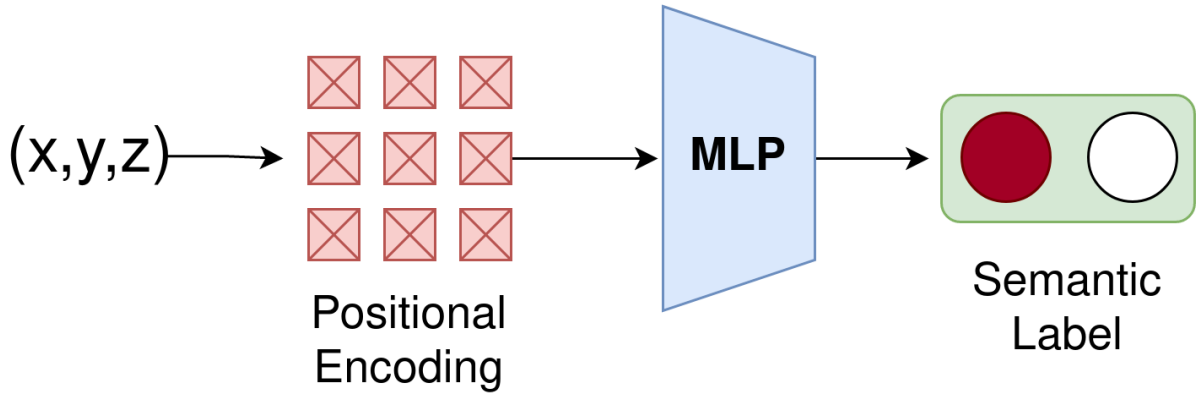


Figure 3.2: The RoadSeg network takes in the point postions as input, passes them thorough a nerf-like positional encoding followed by a fully connected layer. The result is a segmentation score (road/not-road) for each point.

is repeated five times, and the resulting BEVs are stacked together. The stacked BEVs are then fed through a 3D-CNN-based architecture to obtain mean controls. These mean controls serve as the mean for a sampling-based Model Predictive Control (MPC) to generate additional trajectories. From these trajectories, the best one is selected for execution by AutoDP.

3.1.1.1 RoadSeg Network

For the system to work in real time on a resource constrained setup, its essential that the individual building blocks have minimal GPU footprint and the highest possible FPS. The major bottleneck in the pipeline is road-segmentation, which is usually a GPU-intensive process. To address this, leverage is taken of the known map environment of AutoDP. Initial road segmentation is performed using computationally heavy neural networks, and this information is then distilled using a smaller network designed for real-time operation. The resulting network serves as an implicit representation of the operational area for the robot's navigation [41].

The process begins by creating a map of the operational area using LEGO-LOAM [42]. This map, obtained through a custom calibrated lidar-camera setup, is utilized for global localization and planning, similar to other autonomous driving platforms [43]. With 3D points and corresponding RGB values, off-the-shelf segmentation models are employed to segment the road in the 3D map offline. The segmentation model choice is arbitrary. Once road points are segmented, this information is distilled with a smaller neural network, termed the RoadSeg network.

The RoadSeg network takes x, y, z points, passing them through a NERF-like positional encoder [44], followed by a few MLP layers. The network's task is to predict if the given points belong to the road or not, trained using the offline-segmented LEGO-LOAM map as

ground truth. Despite the sparsity of the LEGO-LOAM map, the positional encoding enables the network to learn general spatial trends. This capability allows it to segment the road in entire (denser) lidar point clouds during inference. Moreover, it can identify regions not seen during the map creation process, such as those occupied by other vehicles. The RoadSeg network, illustrated in Fig 3.2, demonstrates its advantages over other baselines in Sec. 3.2.2.

3.1.1.2 3D-CNN based Neural Network

The occupancy grid map at the current timestep T along with the past 4 occupancy grid maps and the global path are concatenated into a $6 \times H \times W$ spatio-temporal tensor. This spatio-temporal tensor is then passed into the 3D-CNN based encoder architecture to extract a feature vector. The feature vector is then passed through 4 fully-connected layers to get an output vector of dimension $H \times 2$ which represents the optimal controls (velocity and angular-velocity). The feature vector from the encoder architecture encodes the spatio-temporal correlations between the occupancy grid maps.

3.1.1.3 Sampling based MPC

The goal is to generate controls for short horizons of H timesteps into the future. The 3D-CNN based Neural Network produces an output that serves as the mean for a Gaussian distribution. Subsequently, N control sequences of length H are sampled from this Gaussian distribution and rolled out using the unicycle kinematics model of the vehicle, resulting in N trajectories. Each of these trajectories is then evaluated based on two cost functions:

- *Smoothness cost*: The smoothness cost ensures that more preference is always given to a trajectory with a smooth change in it's linear and angular velocities.

$$\hat{c}_{ang}(\hat{\boldsymbol{x}}_{\boldsymbol{i}}, \boldsymbol{u}_{\boldsymbol{i}}) = \sqrt{\sum_{h=1}^{H-1} (u_{w_h} - u_{w_{h-1}})^2} \tag{3.1}$$

$$\hat{c}_{lin}(\hat{\boldsymbol{x}}_{\boldsymbol{i}}, \boldsymbol{u}_{\boldsymbol{i}}) = \sqrt{\sum_{h=1}^{H-1} (u_{v_h} - u_{w_{v-1}})^2} \tag{3.2}$$

  where u_v and u_w represents the linear and angular velocity components of the sampled controls respectively.

- *Obstacle avoidance cost*: The obstacle avoidance cost ensures that the trajectories are not colliding with any obstacles (occupied cells in the occupancy grid map).

$$d(\widetilde{\boldsymbol{x}}_{\boldsymbol{h}}, \widetilde{\boldsymbol{o}}_{\boldsymbol{h}}) = || \ \widetilde{\boldsymbol{x}}_{\boldsymbol{h}} - \widetilde{\boldsymbol{o}}_{\boldsymbol{h}} \ ||_2 \tag{3.3}$$

Here $\widetilde{\boldsymbol{x}}_{\boldsymbol{h}}$ and $\widetilde{\boldsymbol{o}}_{\boldsymbol{h}}$ are the states of the agent and the obstacle respectively at time-step h. The agent and the obstacles are represented as circles. Let r_A and r_O be the radius of the agent and the obstacle respectively. The state $\widetilde{\boldsymbol{x}}_{\boldsymbol{h}}$ is said to be in collision with the obstacle state $\widetilde{\boldsymbol{o}}_{\boldsymbol{h}}$ if the euclidean distance $d(\widetilde{\boldsymbol{x}}_{\boldsymbol{h}}, \widetilde{\boldsymbol{o}}_{\boldsymbol{h}})$ between the agent's position and the obstacle's position is less than or equal to the sum of their radius.

$$\hat{c}_{obs}(\hat{\boldsymbol{x}}_{\boldsymbol{i}}, \boldsymbol{u}_{\boldsymbol{i}}) = \begin{cases} \infty \text{ , if } d(\widetilde{\boldsymbol{x}}_{\boldsymbol{h}}, \widetilde{\boldsymbol{o}}_{\boldsymbol{h}}) \leq r_A + r_O, h \in [0, H) \\ 0 \text{ , otherwise} \end{cases} \tag{3.4}$$

where $\hat{\boldsymbol{x}}_{\boldsymbol{i}}$ and $\hat{\boldsymbol{u}}_{\boldsymbol{i}}$ are the i-th trajectory and controls out of the N sampled trajectories and controls.

The final cost $\hat{C}(\hat{\boldsymbol{x}}_{\boldsymbol{i}}, \boldsymbol{u}_{\boldsymbol{i}})$ for each trajectory is calculated as the weighted sum of the three costs

$$\begin{aligned} \hat{C}(\hat{\boldsymbol{x}}_{\boldsymbol{i}}, \boldsymbol{u}_{\boldsymbol{i}}) = w_{ang}\hat{c}_{ang}(\hat{\boldsymbol{x}}_{\boldsymbol{i}}, \boldsymbol{u}_{\boldsymbol{i}}) + w_{lin}\hat{c}_{lin}(\hat{\boldsymbol{x}}_{\boldsymbol{i}}, \boldsymbol{u}_{\boldsymbol{i}}) + \\ w_o\hat{c}_{obs}(\hat{\boldsymbol{x}}_{\boldsymbol{i}}, \boldsymbol{u}_{\boldsymbol{i}}) \end{aligned} \tag{3.5}$$

where w_{ang}, w_{lin} and w_o are the weights chosen by the user. The trajectory with the smallest cost is chosen as the best trajectory and the controls at $h = 1$ are used to drive the ego-vehicle after which a new trajectory is recomputed again. Figure 3.3 shows the output of the neural network and the best trajectory.

3.1.2 Dataset

Each sample in the dataset consists of a sequence of 5 occupancy-grid maps along with the global path as the input and the short horizon optimal control as the output. The CARLA simulator [45] was used to create a dataset. Obstacles were spawned with varying velocities randomly amongst the CARLA maps. Each occupied cell in the occupancy grid map is used as an obstacle while planning the optimal controls using the Sampling based MPC (SMPC). Since the SMPC operates in the center line reference frame (CRF), all the obstacles were transformed to the frame attached to the global path[46]. This allows the curved roads to

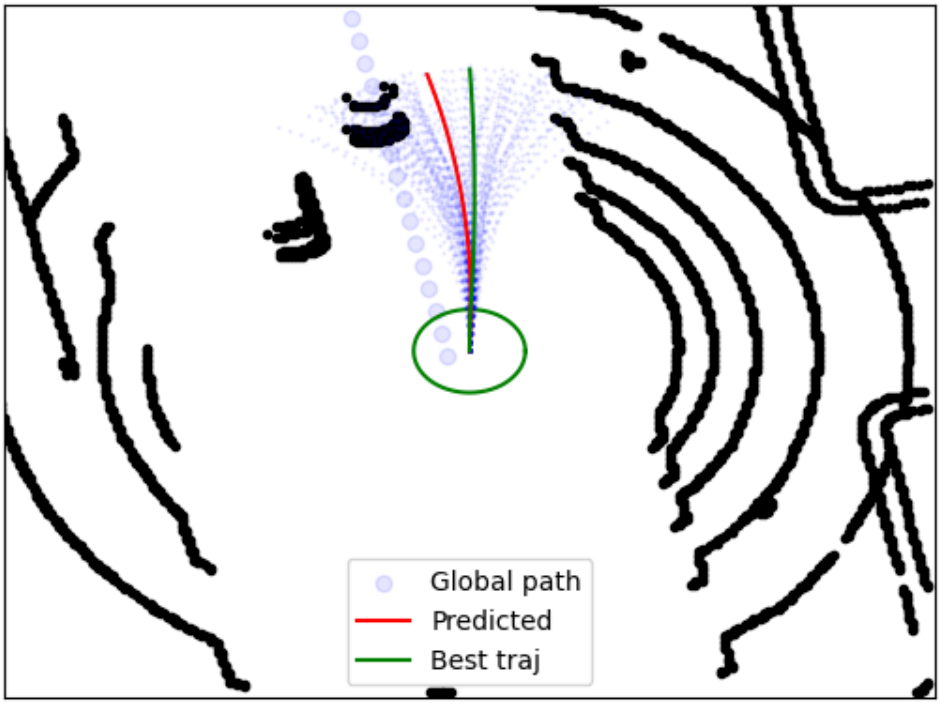


Figure 3.3: This figure demonstrates the output trajectory (red) of the neural network, the sampled trajectories (blue) from a Gaussian distribution with the neural network output as the mean, and the best trajectory (green) selected from the sampled trajectories after scoring them with the smoothness and obstacle avoidance cost functions.

be treated as straight roads and plan feasible trajectories easily without taking the curvature bound constraints. To plan the trajectories that avoids collision with dynamic obstacles whose velocities are known from the CARLA ground truth data, the SMPC was used with the MPPI update rule. The optimal trajectories are then transformed back to the global frame so that the trajectories are within the curvature bounds of the roads in the global frame.

During inference the BEV does not need to be converted from the global frame to the CRF which is a big advantage of our method. Also the 3D-CNN layers learns the dynamic nature of the scene implicitly so we also don't need a separate obstacle trajectory predictor.

3.2 Experimental Evaluation

3.2.1 Qualitative Analysis

In this section, a qualitative comparison has been shown between the trajectories generated by the proposed NeuroSMPC (NSMPC) formulation and trajectories generated by Model Predictive Path Integral (MPPI) and gradient based Cross-Entropy Method (GradCEM) approaches. All the methods generate a short horizon (30 timesteps) control (velocity and angular-velocity) sequence. These controls are rolled out using the unicycle kinematics model to generate the trajectory. The NSMPC takes the past 5 birds-eye view occupancy grid maps and the global path as input. For both the MPPI and GradCEM approach, the occupied cells in the occupancy grid map are considered as obstacles. In both the approaches the distribution is updated iteratively to get the mean controls. Fig. 3.4 shows the trajectory generated by the proposed method compared to the trajectories generate by MPPI and GradCEM. It can be observed that the trajectory generated from our NSMPC formulation is qualitatively similar to the trajectories generated from the iterative approaches in an empty straight and curved road scenario and in a road with obstacles.

3.2.2 Quantitative Analysis

RoadSeg Network: The RoadSeg Network, though only limited to the current operational area, allows for a much faster and accurate road segmentation with minimal GPU footprint. A comparison for the proposed RoadSeg Network is done with two other baselines: RANSAC based plane segmentation and LSeg[47] in terms of computation time and GPU memory utilization. From Table 3.1, it can be observed that RoadSeg outperforms other approaches by a

significant margin. The lesser GPU footprint of RoadSeg enables the execution of the entire NeuroSMPC pipeline on a single laptop within our AutoDP.

Approach	GPU Memory Utilization (MB)	Inference Time (sec)
RoadSeg	790	0.0015
RANSAC	-	0.08
LSeg	3492	0.018

Table 3.1: RoadSeg Performance compared against RANSAC and LSeg. RoadSeg results in lesser GPU footprint and faster inference.

Neural guided Sampling based MPC: A comparison is done of the proposed method with two other sampling based MPC baselines: MPPI and GradCEM, during inference with respect to the number of iterations required to reach an optimal trajectory. Table 3.2 shows that using the proposed method, an optimal trajectory can be achieved in single-shot, whereas using MPPI or GradCEM, iterative update of the mean is needed to get an optimal trajectory. For MPPI, the mean of the distribution was updated for 5 iterations whereas for GradCEM, the mean was updated for 3 iterations.

3.2.3 Sim2Real

One of the key challenges of a deep learning based system is to ensure that a model trained on a dataset collected from a simulator can also run effectively in the real world. While it is very easy to generate large volumes of data from a simulator, it is very difficult to accurately model real world physical phenomenons like friction, impact and uncertainty in a simulator. This results in the dataset from simulator not representing the real physical world accurately. This challenge was addressed by utilizing a simplified form of data. The occupancy grid map

Approach	No. of iteration
NSMPC (Ours)	0
MPPI	5
GradCEM	3

Table 3.2: Comparison of SMPC (proposed method) with MPPI and GradCEM in terms of no. of update iterations required to get an optimal trajectory.

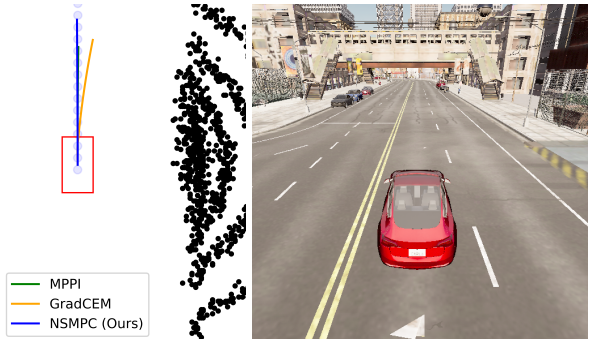
from the simulator lidar is very similar to the occupancy grip map from the real world Lidar. This is because the lidar configuration in the simulator is the same as our AutoDP. The only thing missing was the simulator's absence of sensor noise which was addressed by introducing minor random perturbations to the lidar points. After training the model on synthetic data obtained from a simulator, the model was successfully deployed on the AutoDP. Fig. 3.5 shows the execution of the model on the AutoDP while avoiding an obstacle in a controlled on-campus scene. Despite occasional instances where the model's output led to collisions with the obstacle, the sampling nature of the proposed formulation guarantees the derivation of an obstacle-avoiding trajectory from the distribution of trajectories sampled from the model's output as the mean.

3.2.4 Why Sample?

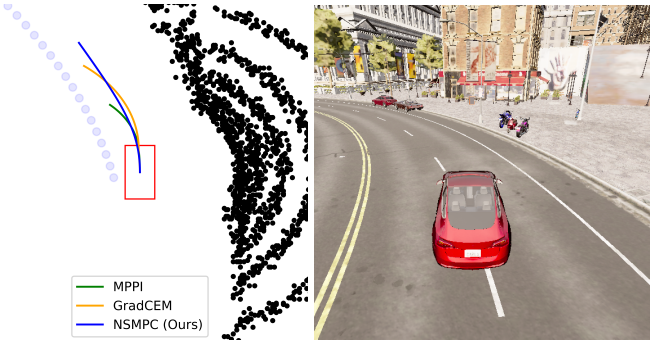
The Neural Network's output is stochastic in nature and is not always guaranteed to avoid collisions with the obstacles. Ensuring safety requires that the trajectory executed by AutoDP is consistently collision-free. To address this, a distribution of trajectories is sampled using the proposed model's output as the mean. The sampled trajectories are then scored using the cost functions mentioned in Section 3.1.1.3. The best trajectory is then selected from the sampled trajectories and executed by the AutoDP. Fig. 3.6 shows an example scenario where the Neural Network output is colliding with the obstacle in front. The best trajectory (green) chosen from sampled trajectories is then executed by the AutoDP to avoid colliding with the obstacle.

3.3 Implementation and Training

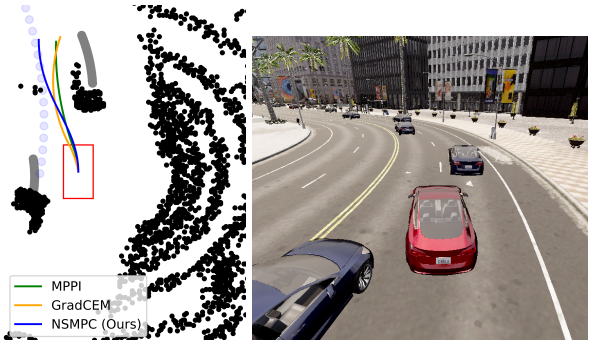
The MPPI, GradCEM and the proposed Neural guided Sampling based MPC has been implemented using the PyTorch [48] library and trained on a system with a Intel Core i7-5930K (6 Core 12 threads) CPU and one NVIDIA RTX A4000 GPU with 16 GB VRAM and 64 GB RAM. The Lion optimizer [49] was used with a learning rate of 0.001 to train the proposed network parameters.



(a) Straight road



(b) Curved road



(c) Obstacle avoidance

Figure 3.4: A qualitative comparison is shown of the trajectories generated by NSMPC (our method), MPPI and GradCEM in three different driving conditions: Straight empty road, Curved empty road and a road with Obstacles. The grey line denotes the future trajectory of the dynamic obstacles.

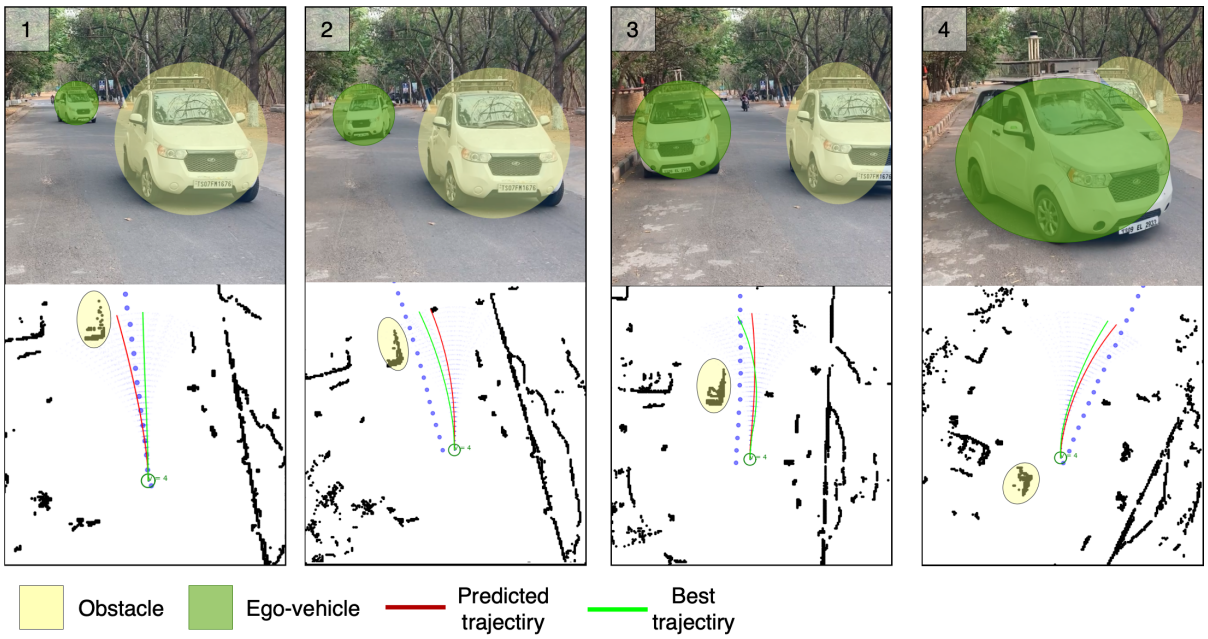


Figure 3.5: **Run on a real car:** The presented study demonstrates the successful application of the model trained on synthetic data from a simulation to operate effectively in a real-world environment. The model underwent testing on an actual self-driving car, navigating around an obstacle within a controlled on-campus scene to prioritize safety. In the visual representation, the top row of image sequences illustrates the capability of the ego-vehicle (depicted in green) to maneuver and avoid the obstacle (highlighted in yellow). Meanwhile, the bottom row of images displays corresponding occupancy grid maps, showcasing the predicted trajectory (in red) and the optimal sampled trajectory (in green) within the ego-vehicle coordinate frame.

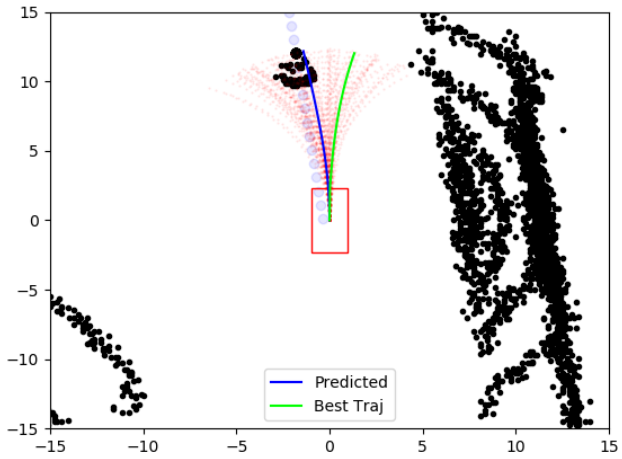


Figure 3.6: **Why sample?** Since the NN output may not always be collision free, a distribution of trajectories (red) is sampled by using the NN output (blue) as the mean of a Gaussian distribution. The best trajectory (green) is then selected to be executed by the AutoDP.

Chapter 4

Conclusion and Future Work

In this thesis, the primary focus was on developing and presenting a self-supervised approach for modeling a sequence of observed point clouds and predicting the subsequent sequences. The practical application of modelling sequences of observed point-clouds are then showcased in the domain of autonomous driving.

- **ATPPNet:** A novel self-supervised approach is introduced for predicting future point cloud sequences based on provided past point cloud sequences. The method employs spatial and channel-wise attention in conjunction with Conv-LSTMs, complemented by a 3D-CNN branch, to effectively model spatio-temporal information and predict future point cloud sequences at a frequency lower than that of LiDAR. The proposed approach, named ATPPNet, is thoroughly evaluated on various real-world datasets, demonstrating superior performance through comprehensive ablations. An application study is presented to showcase the potential of the method in predicting point clouds.

- **NeuroSMPC:** A novel neural network guided Sampling Based Optimal Controller called NeuroSMPC was proposed as an effective mechanism of interleaving single shot optimal inference with sampling based frameworks. NeuroSMPC overcomes the entailment of iterative resampling of sampling based optimal control frameworks by inferring single shot optimal trajectory rollouts and yet retains the advantage of sample diversity by sampling controls around the predicted rollouts. NeuroSMPC trajectory rollouts are similar to sampling based optimal control formulations [28, 30] yet come at a much faster clip thus making it suitable for real time settings like on-road autonmous driving. NeuroSMPC's spatio-temporal convolution architecture learns implicitly the dynamic nature of scenes without the need for explicit prediction of future states as it seamlessly avoids dynamic obstacles through an implicit understanding of their future evolution. NeuroSMPC has been tested in various CARLA scenes and a sim2real transfer on AutoDP

(AutonomousDrivingPlatform) for on-road campus autonomous driving establishes its efficacy.

4.0.1 Future Work:

- **ATPPNet:** The future work would mainly focus on retaining the shapes of the small objects in the predicted point-clouds. Another direction of the future work would be to predict the point clouds by querying a location at a particular time, to determine areas with minimal drift in motion.
- **NeuroSMPC:** The current study operates under the assumption that the vehicle traverses flat terrain, utilizing the kinematics model for trajectory planning and evaluation. Future research endeavors will extend this approach by incorporating the dynamics model of the vehicle to model uneven terrains, such as climbing slopes, and subsequently plan trajectories tailored to these challenging topographical conditions.

Related Publications

1. **NeuroSMPC: A Neural Network guided Sampling Based MPC for On-Road Autonomous Driving**
   Kaustab Pal*, Aditya Sharma, Mohd Omama, Parth N. Shah, K. Madhava Krishna
   IEEE International Conference on Automation Science and Engineering (CASE 2023)

2. **ATPPNet: Attention based Temporal Point cloud Prediction Network**
   Kaustab Pal*, Aditya Sharma, Avinash Sharma, K. Madhava Krishna
   submitted to IEEE International Conference on Robotics and Automation (ICRA2024)

Bibliography

[1] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[2] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar 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.

[3] Sergio Casas, Sadat Abbas, and Raquel Urtasun. Mp3: A unified model to map, perceive, predict and plan. In *2021 IEEE Conference on Computer Vision and Pattern Recognition*, pages 14403–14411. IEEE, 2021.

[4] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel 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.

[5] Atsushi Kawasaki and Akihito Seki. Multimodal trajectory predictions for autonomous driving without a detailed prior map. In *2021 IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3723–3732. IEEE, 2021.

[6] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 922–928. IEEE, 2015.

[7] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3577–3586, 2017.

[8] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions On Graphics (TOG)*, 36(4):1–11, 2017.

[9] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[10] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5589–5598, 2020.

[11] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proc. ICCV*, 2015.

[12] Tan Yu, Jingjing Meng, and Junsong Yuan. Multi-view harmonized bilinear network for 3d object recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 186–194, 2018.

[13] Ze Yang and Liwei Wang. Learning relationships for multi-view 3d object recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7505–7514, 2019.

[14] David Deng and Avideh Zakhor. Temporal lidar frame prediction for autonomous driving. In *2020 International Conference on 3D Vision (3DV)*, pages 829–837. IEEE, 2020.

[15] Hehe Fan and Yi Yang. Pointrnn: Point recurrent neural network for moving point cloud processing. *arXiv preprint arXiv:1910.08287*, 2019.

[16] Fan Lu, Guang Chen, Zhijun Li, Lijun Zhang, Yinlong Liu, Sanqing Qu, and Alois Knoll. Monet: Motion-based point cloud prediction network. *IEEE Transactions on Intelligent Transportation Systems*, 23(8):13794–13804, 2021.

[17] Xinshuo Weng, Jianren Wang, Sergey Levine, Kris Kitani, and Nicholas Rhinehart. Inverting the pose forecasting pipeline with spf2: Sequential pointcloud forecasting for sequential pose forecasting. In *Conference on robot learning*, pages 11–20. PMLR, 2021.

[18] Benedikt Mersch, Xieyuanli Chen, Jens Behley, and Cyrill Stachniss. Self-supervised point cloud prediction using 3d spatio-temporal convolutional networks. In *Conference on Robot Learning*, pages 1444–1454. PMLR, 2022.

[19] Xinshuo Weng, Junyu Nan, Kuan-Hui Lee, Rowan McAllister, Adrien Gaidon, Nicholas Rhinehart, and Kris M Kitani. S2net: Stochastic sequential pointcloud forecasting. In *European Conference on Computer Vision*, pages 549–564. Springer, 2022.

[20] Zhen Luo, Junyi Ma, Zijie Zhou, and Guangming Xiong. Pcpnet: An efficient and semantic-enhanced transformer network for point cloud prediction. *IEEE Robotics and Automation Letters*, 8(7):4267–4274, 2023.

[21] Kaustubh Mani, Swapnil Daga, Shubhika Garg, Sai Shankar Narasimhan, Krishna Murthi Jatavallabhula, and Madhava Krishna. Monolayout: Amodal scene layout from a single image. In *2021 IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3723–3732. IEEE, 2021.

[22] Thomas Roddick, Alex Kendall, and Robert Cippolla. Orthographic feature transform for monocular 3d object detection. In *2019 British Machine Vision Conference*. BMVA, 2019.

[23] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV*, page 194–210, Berlin, Heidelberg, 2020. Springer-Verlag.

[24] Namhon Lee, Wongun Choi, Vernazal Paul, Philip Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 336–344. IEEE, 2018.

[25] Shashank Srikanth, Ahmed Junaid Ansari, Sarthak Sharma, Krishna Murthi Jatavallabhula, and Madhava Krishna. Intermediate representations for future prediction. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765. IEEE, 2019.

[26] R.Y. Rubinstein and D.P. Kroese. The cross-entropy method: A unified approach to combinatorial optimization. *Monte-Carlo Simulation, and Machine Learning, Springer-Verlag,*, 2004.

[27] Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model predictive path integral control using covariance variable importance sampling, 2015.

[28] Mohak Bhardwaj, Balakumar Sundaralingam, Arsalan Mousavian, Nathan D. Ratliff, Dieter Fox, Fabio Ramos, and Byron Boots. Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation. In *Proceedings of the 5th Conference on Robot Learning*, pages 750–759, 2022.

[29] Cristina Pinneri, Samburaj Sawany, Sebastien Blaes, Jorh Stuckler, and George Martius. Sample-efficient cross-entropy method for real-time planning. In *2020 4th Conference on Robot and Learning (CoRL)*. CoRL, 2020.

[30] Homanga Bharadhwaj, Kevin Xie, and Florian Shkurti. Model-predictive control via cross-entropy and gradient-based optimization. In *Learning for Dynamics and Control*, pages 277–286. PMLR, 2020.

[31] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wangchun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28, 2015.

[32] Jo Schlemper, Ozan Oktay, Liang Chen, Jacqueline Matthew, Caroline Knight, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. Attention-gated networks for improving ultrasound scan plane detection. *arXiv preprint arXiv:1804.05338*, 2018.

[33] Jongchan Park, Sanghyun Woo, Joon-Young Lee, and In So Kweon. Bam: Bottleneck attention module. *arXiv preprint arXiv:1807.06514*, 2018.

[34] Joonatan Manttari, Sofia Broomé, John Folkesson, and Hedvig Kjellstrom. Interpreting video features: A comparison of 3d convolutional networks and convolutional lstm networks. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

[35] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer, 2014.

[36] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[37] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017.

[38] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[39] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019.

[40] Mohd Omama, Sripada VS Sundar, Sandeep Chinchali, Arun Kumar Singh, and K Madhava Krishna. Drift reduced navigation with deep explainable features. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6316–6323. IEEE, 2022.

[41] Krishna Murthy Jatavallabhula, Alihusein Kuwajerwala, Qiao Gu, Mohd Omama, Tao Chen, Shuang Li, Ganesh Iyer, Soroush Saryazdi, Nikhil Keetha, Ayush Tewari, et al. Conceptfusion: Open-set multimodal 3d mapping. *arXiv preprint arXiv:2302.07241*, 2023.

[42] Tixiao Shan and Brendan Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765. IEEE, 2018.

[43] Autoware Foundation. Autoware: Open-source software for urban autonomous driving, 2023.

[44] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[45] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[46] Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. Optimal trajectory generation for dynamic street scenarios in a frenet frame. In *2010 IEEE International Conference on Robotics and Automation*, pages 987–993. IEEE, 2010.

[47] Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and René Ranftl. Language-driven semantic segmentation. *arXiv preprint arXiv:2201.03546*, 2022.

[48] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani,

Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[49] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms, 2023.