Motion Forecasting of Surrounding Agents For Autonomous Vehicles

A THESIS

submitted by

JUNAID FAYAZ LONE

for the award of the degree

of

MASTER OF TECHNOLOGY

(by Research)

COMPUTER SCIENCE AND ENGINEERING

INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY DELHI

NEW DELHI - 110020

May 2025

THESIS CERTIFICATE

This is to certify that the thesis titled **Motion Forecasting of Surrounding Agents For Autonomous Vehicles**, submitted by **Junaid Fayaz Lone**, to the Indraprastha Institute of Information Technology, Delhi, for the award of the degree of **Masters of Technology by Research**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Saket Anand
Thesis Supervisor
Associate Professor
Dept. of Electronics and Communication
IIIT Delhi, 110020

Dr. Sanjit Kaul
Co-Supervisor
Professor
Dept. of Electronics and Communication
IIIT Delhi, 110020

Place: New Delhi

Date: May 2025

ACKNOWLEDGEMENTS

With profound gratitude, I acknowledge the exceptional guidance of Dr. Saket Anand, whose mentorship has been the cornerstone of my research journey. His unwavering support has not only shaped this thesis but has fundamentally transformed my understanding of computer vision and deep learning. The opportunity to work under his direction these past two years has been an honor that will influence my professional path for years to come.

I also am deeply indebted to Dr. Sanjit Kaul, my co-advisor, whose brilliant insights and thoughtful feedback consistently illuminated new directions for my research.

The remarkable people at Vision Lab deserve special recognition. Their collaborative spirit, intellectual generosity, and friendship created an environment where ideas flourished and challenges became opportunities for growth. I am very grateful for their support.

I appreciate the financial support provided by Vehant Technologies through their Research Fellowship program. Particular thanks go to Dr. Renu Rameshwaram, whose expertise, guidance, and encouragement were instrumental in bringing this project to fruition.

Above all, my deepest appreciation goes to my family, whose unconditional love and steadfast belief in my abilities have sustained me through every challenge. Their sacrifices and encouragement have been my greatest source of strength. This achievement is as much theirs as mine.

Junaid Fayaz Lone

MT23001

ABSTRACT

KEYWORDS: Motion Forecasting; Trajectory Prediction; Autonomous Systems; Multi-agent Interaction

Motion forecasting of surrounding agents is fundamental for autonomous systems navigating complex, dynamic environments. This capability enables autonomous vehicles and robots to anticipate the future trajectories of vehicles, pedestrians, and other moving entities.

Recent models leverage large-scale datasets to learn spatio-temporal patterns. Many existing models focus on single-agent prediction. This poses a problem because, as the number of agents in a scene increases, the computational time scales linearly due to the redundant re-encoding of features—such as static or HD maps—that could otherwise be shared across agents.

Moreover, some models struggle to accurately capture interactions between agents, limiting their ability to understand the influence they have on one another. They also encounter challenges with variable sequence lengths; as the sequence length increases, models using GRUs or RNNs can lose context over long sequence lengths, and improper handling of padding timesteps can diminish encoding quality.

We also explore monocular motion forecasting in the context of traffic surveillance. In this setting, the goal is to forecast the future positions of agents using monocular camera footage by leveraging monocular depth estimation. This approach typically operates without access to high-definition (HD) maps and instead relies solely on the historical motion of agents—essentially performing map-free motion forecasting.

In this work, we propose a new approach that addresses the challenges such as multi-agent forecasting, agent-agent interaction, and monocular camera based motion forecasting. We evaluate our autonomous driving model on two benchmark datasets—Argoverse 2 and nuScenes—under standard evaluation metrics: MinFDE, MinADE, and MR. For monocular motion forecasting, we evaluate our method on the BrnoComp dataset.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

MinADE	Minimum Average Displacement Error
MinFDE	Minimum Final Displacement Error
MR	Miss Rate
MAPGP	Multi Agent PGP (Our Model)

NOTATION

m	meters, m
α	Angle of thesis in degrees
β	Flight path in degrees

CHAPTER 1

INTRODUCTION

Autonomous systems operate in complex, dynamic environments where the behavior of surrounding agents is inherently unpredictable. The capability to accurately forecast the trajectories of these agents—commonly referred to as motion forecasting—is critical for safe and efficient navigation. Through motion forecasting, an autonomous system can anticipate future states of the environment, plan its maneuvers accordingly, and minimize the risk of collisions or traffic disruptions.

Over the past decades, research on trajectory prediction has expanded significantly, leveraging advancements in deep learning and the availability of large-scale autonomous driving datasets [1; 2; 3]. These datasets provide rich annotations of agent trajectories along with high-definition (HD) maps, which play a crucial role in capturing the static road structure and traffic regulations that influence agent behavior.

Early approaches to trajectory prediction, such as [4], encoded HD maps using convolutional neural networks (CNNs). These methods treated HD maps as rasterized bird's-eye view images and applied CNNs to extract features. While CNNs are effective at capturing local spatial patterns, their ability to encode the road geometry is inherently limited due to their limitations in capturing the spatial context (next lane, neighbouring lanes , traffic signals) of the road.

To address these limitations, more recent methods have shifted towards graph-based representations of HD maps. Works such as [5] and [6] propose encoding HD maps as graphs rather than as rasterized images. In these approaches, road elements (e.g., lanes) are treated as nodes and their relationships (e.g., neighboring lane, next lane) as edges, enabling the model to effectively capture both local and global spatial dependencies. By doing so, these methods offer a more structured and efficient way to incorporate map information into trajectory prediction models. Recent methods, such as [7], encode the local region around each agent directly as 2D points representing lane centerlines.

Agent history encoding plays a crucial role in enriching past motion information, which is essential for accurately predicting future trajectories. Given an agent's past T_h

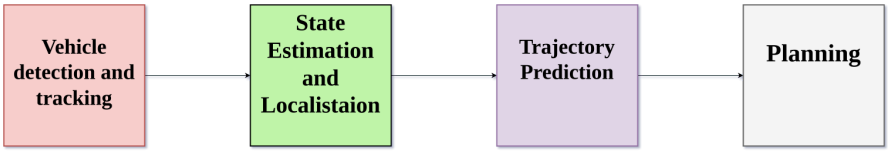


Figure 1.1: Pipeline for Autonomous Driving: From Vehicle Detection to Planning.

positional states (x_t, y_t) over a sequence of historical time steps, the trajectory is treated as a sequential input and encoded using recurrent neural networks (RNNs) such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs)[8]. More recently, Transformer-based [7] architectures have emerged as powerful alternatives due to their ability to model long-range dependencies effectively. The encoded representation of the agent's past motion provides contextual awareness of its movement patterns, aiding in robust forecasting. In our approach, we employ a Transformer-based encoder to capture rich spatiotemporal dependencies in agent trajectories, enabling a more expressive representation for downstream prediction tasks.

Motion forecasting becomes particularly challenging in scenarios where only monocular cameras are available, such as in traffic surveillance settings, due to the absence of complex sensors like LiDARs and radars. Prior work, such as [9], has addressed this limitation by leveraging sequences of images and optical flow to estimate the states of surrounding vehicles relative to an ego vehicle. In contrast, our approach employs a depth estimation-based model to infer the state of agents in the scene. By incorporating depth cues, this method enables more robust and geometrically informed estimation of vehicle positions and motions, thereby enhancing the quality of motion forecasting in monocular camera setups

In this thesis, we present three methods we implemented to improve upon our baseline and for intergration in real world applciation: (1) vehicle state estimation using a monocular camera, (2) A Transformer and Attention-based encoder for agents and lane nodes and their interactions (3) an efficient multi-agent prediction approach. These methods will be discussed in the Further sections: Vehicle Tracking and State Estimation,Trajectory Encoding in Motion Forecasting and Multiagent Prediction.

1.1 Trajectory Encoding in Motion Forecasting

Many motion forecasting models rely on sequence-based encoders to process historical trajectories. For instance, Social-GAN [10] employs a generative adversarial network with LSTMs to model human-like motion patterns and interactions. Another approach, PGP [8], utilizes a GRU's with attention mechanisms to encode agent history and capture dependencies between agents and their surroundings.

With the advent of large-scale datasets such as NuScenes and Argoverse, encoding trajectory data has become more challenging due to variations in temporal resolution. In NuScenes, data is sampled at 2Hz, leading to shorter historical sequences. In contrast, Argoverse samples at 10Hz, meaning a 5-second history results in sequences of length 50. Handling such long sequences presents challenges for traditional GRU and LSTM models due to vanishing gradient, inefficient masking and padding, because GRUs and LSTMs process inputs sequentially, making it difficult to handle variable-length sequences effectively with proper masking and padding. Transformers, on the other hand, naturally support flexible sequence lengths through self-attention mechanisms with key masking.

In our approach, we implement a Transformer-based encoder to effectively model the historical trajectories of target agents. This encoder consists of a self-attention mechanism that captures agent-agent interactions within the scene, allowing the model to learn dependencies between different agents. Additionally, a transformer encoder is used to process the historical trajectory of each agent, enabling the extraction of meaningful temporal features.

Through testing on the nuScenes dataset, our method demonstrated a significant improvement over the baseline. Specifically, we achieved a 7 cm reduction in both MinADE_5 and 8cm reduction in MinADE_{10}. Furthermore, our approach also outperformed the baseline in terms of MinFDE. The complete results are presented in Table 3.1.

1.2 Multi Agent Forecasting

The baseline model we utilized for trajectory prediction was the PGP [8] model, which was originally designed for single-agent forecasting. One major drawback of this approach is that as the number of agents in a scene increases, the computational time required for inference also increases significantly. Specifically, the computational time per prediction for a single agent was measured at 14 milliseconds. However, as the number of agents in the scene increases, the total computational time became impractically high, making real-time trajectory prediction infeasible.

Models like HiVT [7] have addressed this limitation of multi-agent forecasting by defining a local region around each agent with a 50-meter radius. Within this region, all features, including other agents and lane centerlines, are transformed into the agent's local coordinate frame. The local region is then encoded, and a global interaction graph neural network is used to combine the different local context vectors. Agents are represented with δx and δy, capturing the spatial relationships between different local regions.

More recently, state-of-the-art approaches such as QCNet [11] have pushed the limits of multi-agent forecasting. QCNet introduces a powerful agent-centric query-based mechanism combined with interaction-aware modeling and factorized attention, enabling accurate and scalable trajectory prediction across dense urban scenes. However, this performance comes at the cost of extreme resource demands: training QCNet requires up to 160 GB of VRAM, making it impractical for many researchers and deployment scenarios.

We use PGP as our baseline due to its favorable computational profile. It requires less than 6 hours to train and consumes very low GPU memory, both during training and inference. This makes it especially suitable for development in constrained settings or rapid experimentation.

To address the limitations of PGP in multi-agent settings, we propose a multi-agent graph encoder that optimizes computation by encoding the scene graph only once per scenario. This stands in contrast to prior models, where the map had to be re-encoded for each agent individually, leading to redundant computations.

Our approach still requires some overhead computation, but it significantly improves efficiency. The computation time reaches approximately 28ms ± 5ms per scenario for a variable number of agents, N. This represents only a 2× increase compared to the single-agent model, rather than scaling with a complexity of $O(N)$. The detailed approach will be discussed in further chapters.

1.3 Vehicle Tracking and State Estimation

Accurate vehicle tracking and state estimation are integral to autonomous driving systems and Traffic surveillance. Traditional approaches often leverage camera–LiDAR calibration techniques to fuse complementary sensor data, as demonstrated by benchmark datasets such as Argoverse [1; 2] and nuScenes [3]. This calibration relies on multimodal sensor fusion for robust 3D localization of surrounding agents, leveraging both LiDAR and camera data. To associate detected objects with spatial data, LiDAR points are projected onto the image plane using intrinsic and extrinsic calibration matrices. A 3D bounding box is then fitted around the clustered LiDAR points to estimate object position, orientation, and dimensions.

We explore monocular depth estimation approach that utilizes static surveillance traffic cameras for tracking and estimating positions of agents. To achieve this, we propose a monocular state estimation module which uses monocular depth estimation models, including Metric3D [12] and ZoeDepth [13]. The complete methodology will be presented in subsequent chapters. Our state estimation approach enables the prediction of agent trajectories over a 3-second horizon, achieving an average minADE of 0.43m and 0.68m on the BrnoComp dataset using two different trajectory prediction models.

1.4 Contribution of the Thesis

Our key contributions to Motion Forecasting through this work include the following:

- **Monocular Vehicle Tracking and State Estimation:** Our first contribution is a novel monocular state estimation approach that leverages static surveillance traffic cameras. Our method employs state-of-the-art monocular depth estimation

models, such as Metric3D and ZoeDepth, to accurately estimate vehicle position, orientation.

- **Transformer-based Trajectory Encoding in Motion Forecasting:** We use a Transformer-based encoder to model historical trajectories in motion forecasting. Our Transformer-based encoder effectively captures and extracts meaningful temporal features from historical data.
- **Agent Agent Interaction** We use self-attention across agent history encodings, enabling us to effectively capture agent-agent interactions. This approach enhances our ability to model interactions between agents while allowing the temporal encoder to integrate nearby agents' histories.
- **Efficient Multi-Agent Prediction:** Lastly, we propose an efficient multi-agent graph encoder that optimizes computation in scenarios with numerous agents. Traditional single-agent forecasting models incur significant redundancy by re-encoding the map for each agent individually. Our approach encodes the scene graph only once per scenario, significantly reducing computational overhead.

CHAPTER 2

Literature Review

This chapter presents a review of existing techniques for trajectory prediction of surrounding agents in autonomous driving. By analyzing various forecasting methods, we aim to identify key challenges that can be addressed in our approach. Additionally, we discuss various deep learning-based methodologies used for motion prediction, with the objective of integrating relevant strategies into our proposed framework.

2.1 Trajectory Prediction

Trajectory prediction is a crucial component in autonomous driving, as it enables the forecasting of future movements of surrounding agents to ensure safe navigation. Traditional methods, such as Kalman filters [14; 15], rely on historical trajectory data. While these approaches perform well for short-term predictions, they often struggle with modeling complex interactions and dynamic environments.

Recent advancements have introduced deep learning techniques, particularly Gated Recurrent Units (GRUs) [8] and Long Short-Term Memory (LSTM) networks [10], to capture temporal dependencies in agent trajectories. More recently, transformer-based models [16] have demonstrated superior sequence encoding capabilities. Approaches such as [17; 18] utilize transformer encoders to model agent history, yielding state-of-the-art results.

Furthermore, incorporating map information has proven to be beneficial for long-term trajectory prediction. For instance, [19] leverages map constraints along with agent histories to improve forecasting accuracy. Early works such as [4] employed Convolutional Neural Networks (CNNs) to encode map data as raster images. However, CNN-based representations suffer from limitations in capturing high-level semantic information. To address this, studies like [6; 5] propose encoding maps using lane graphs, where each node represents a discretized centerline of lane segments. This structured

representation offers richer semantic information compared to Bird's Eye View (BEV) rasterization.

Another recent advancement involves the use of learnable tokens. Studies such as [20] incorporate learnable queries to enhance the model's ability to capture fine-grained spatiotemporal interactions. Additionally, methods like [21] explore knowledge graphs to model complex relationships between static and dynamic agents, further improving scene understanding.

In our thesis, we implement a lane graph representation for the map and use transformer encoders to encode both trajectories and lane segments. The graph-based representation effectively captures spatial relationships, while transformers facilitate robust trajectory modeling and handle variable-length sequences efficiently.

2.2 Multi agent Prediction

In dynamic environments such as autonomous driving, predicting the trajectories of multiple agents is crucial for safety, collision avoidance, and efficient navigation. Traditional single-agent forecasting models predict each agent's future trajectory independently, This simplification leads to suboptimal results, especially in dense traffic scenarios.

Moreover, as the number of agents in a scene increases, single-agent models require separate forward passes for each agent, leading to a significant rise in computational cost. This approach becomes impractical for real-time applications, where fast and efficient predictions are necessary. In contrast, multi-agent forecasting models explicitly model agent interactions, capturing joint dependencies and improving both accuracy and efficiency. By processing all agents collectively, they reduce redundant computations and scale more effectively to crowded environments.

Recent advancements, such as [7], tackle multi-agent forecasting by defining a local region (with a 50-meter radius) around each agent. Within this region, the model encodes the local context, and a graph neural network propagates this context among neighboring regions. This process yields a global interaction context vector, which is subsequently leveraged for downstream prediction tasks.

Similarly, [22] normalizes each lane segment using its geometric center and learns an edge representation to describe the transformation between the lane and other static or dynamic elements in the scene.

Another approach, QCNet [11], introduces a query-centric paradigm for encoding heterogeneous traffic scenes efficiently. Instead of re-encoding the scene for each agent, QCNet encodes all scene elements—including multiple agents and map features—into a shared representation in their local spacetime reference frames. This allows for parallel multi-agent trajectory decoding while capturing rich contextual interactions using factorized attention mechanisms across time steps, agents, and map features. The decoding process employs a learnable query-based approach.

In contrast to these methods, we adopt a streamlined approach: the map is encoded globally only once and then transformed into local coordinates for each agent to extract an agent-centric local map. This eliminates the need to re-encode the map for each individual agent, enhancing computational efficiency while preserving the necessary context for accurate prediction.

CHAPTER 3

Motion Forecasting with Multisensor Perception Systems

This chapter focuses on predicting the future positions of moving vehicles using input from multiple sensors, such as cameras, LiDAR, and radar. The process begins by tracking vehicles across the different sensor streams and fusing their data to achieve a more accurate estimation of each vehicle's position and depth information. After fusing the sensor data, we estimate the current state of each vehicle, including its location, speed, and sometimes even acceleration. With this comprehensive state information, we apply trajectory prediction models to forecast where the vehicles are likely to move in the near future. These predictions can enhance planning and improve safety.

3.1 Problem Formulation

Motion forecasting involves predicting the future trajectories of agents (e.g., vehicles, pedestrians) in a dynamic scene. Formally, given the observed history of T_h timesteps for each agent, our goal is to predict their future states over the next T_f timesteps.

At each timestep t, for every agent in the scene, we have access to the following state information:

- Position: (x_t, y_t)
- Velocity: v_t
- Acceleration: a_t
- Heading change rate: $\dot{\theta}_t$

In addition to the agent's motion history, we are also provided with a local map context m that includes lane-level semantics. The map consists of a set of lane centerlines around the agent, where each lane centerline is represented as a sequence of points. Each lane segment point is described by:

- Position: (x, y)
- Heading: θ
- Semantic attributes: e.g., lane type, turn direction, traffic control information, etc.

Given this historical information and the map context, the task is to learn a model that estimates the conditional probability distribution over the future trajectories of an agent:

$$P(\mathbf{X}_{T_h+1:T_h+T_f} \mid \mathbf{X}_{1:T_h}, m)$$

where $\mathbf{X}_{1:T_h}$ denotes the observed trajectory history of an agent, and $\mathbf{X}_{T_h+1:T_h+T_f}$ denotes its future trajectory over the prediction horizon.

This formulation is foundational in applications such as autonomous driving, where accurate trajectory prediction under complex scene dynamics and map constraints is critical for safe and efficient planning.

3.2 Proposed Methodology

3.2.1 Encoder

The encoder module is responsible for extracting meaningful representations from the raw input data, which consists of agent trajectories, interactions between agents, and the surrounding map context. It is composed of three main components: the Trajectory Encoder, which captures the temporal dynamics of individual agent motions; the Agent-Agent Interaction Encoder, which models the social interactions and dependencies among agents; and the Map Encoder, which encodes the static spatial layout and semantic information of the environment.

Trajectory Encoder

A trajectory is defined as a temporal sequence of states and semantic features corresponding to an individual agent observed over a fixed historical window. Formally, each trajectory is represented as a tensor of shape $T_h \times$ input_feats, where T_h is the number of historical timesteps and *input_feats* includes both state and semantic information.

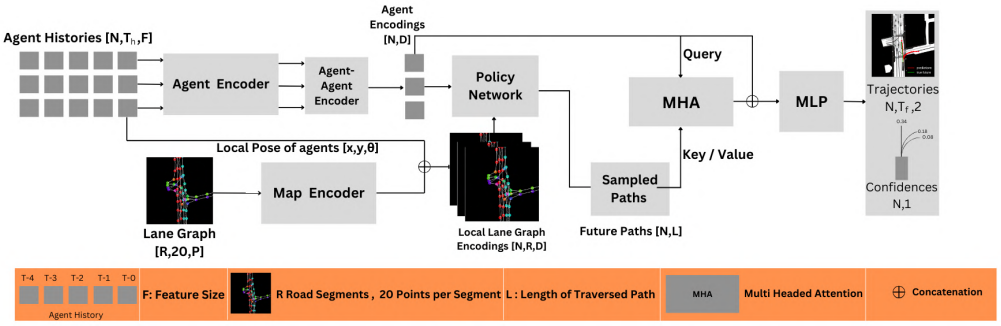


Figure 3.1: Architecture of our proposed methodology

The state features encompass physical motion characteristics of the agent, such as position (x, y) coordinates, velocity, acceleration, and heading change rate. These features are crucial for capturing the kinematic behavior and dynamics of each agent over time. To complement the state features, we also incorporate semantic features that encode the category of the agent. These include object classes such as `pedestrian`, `vehicle`, `bus`, among others.

To model the temporal evolution of these combined features, we utilize a Transformer encoder architecture. This choice offers multiple advantages: it captures long-range dependencies within the trajectory, supports variable-length inputs through causal masking, and handles missing or padded timesteps seamlessly.

Furthermore, we prepend a learnable classification token, denoted as `[CLS]`, to each trajectory sequence. This token acts as an anchor for summarizing the entire input sequence. After passing the input through the Transformer layers, the output corresponding to the `[CLS]` token is extracted and used as a fixed-size embedding that holistically represents the historical trajectory of the agent. This embedding is then forwarded to subsequent modules for interaction modeling and policy prediction.

Agent-Agent Interaction Encoder

Understanding the interactions between agents is crucial for accurately modeling complex, multi-agent environments where the behavior of one agent can be influenced by the presence and motion of others. These interactions are especially significant in crowded or dynamic scenes, such as intersections or crosswalks, where collision avoidance and social norms play a vital role in determining an agent's future trajectory.

To capture such inter-agent dependencies, we implement an agent-agent interaction encoder using a self-attention mechanism. Specifically, we apply multi-head self-attention across all agents present in the scene at each timestep. The input to this module consists of the encoded trajectory embeddings produced by the Trajectory Encoder for each agent.

To explicitly incorporate relative positional information, we compute the pairwise spatial offsets between agents. For each agent pair (i, j), we calculate the relative displacement features $\Delta x = x_j - x_i$ and $\Delta y = y_j - y_i$, where (x_i, y_i) and (x_j, y_j) are the spatial coordinates of agents i and j, respectively. These relative displacements are concatenated with the key and value vectors in the attention computation, enriching the self-attention mechanism with spatial priors that reflect the proximity and alignment between agents.

By integrating both feature and spatial information in the attention mechanism, each agent's updated embedding becomes context-aware—encoding not only its own trajectory history, but also the influence of surrounding agents weighted by their spatial relevance.

This approach enables our model to capture diverse and subtle social interactions, such as yielding, following, or overtaking behaviors, thereby improving the realism and accuracy of multi-agent trajectory prediction.

Map Encoder

We begin by converting the high-definition (HD) map into a structured representation called a **Lane Graph**. Each lane in the HD map is divided into fixed-length segments of 20 meters, with each segment represented as a *node* in the graph. We define two types

of *edges* to capture spatial and directional relationships between these segments:

- **Successor Edges**: Connect a lane segment to the next segment along the direction of travel.
- **Proximal Edges**: Connect nearby lane segments that run approximately in the same direction.

This results in a directed graph $G = (V, E)$, where V denotes the set of lane segment nodes and E includes both successor and proximal edges.

To encode the lane graph, we utilize a **2-layer Transformer encoder**. Each lane segment, treated as a sequence of points, is independently processed by the Transformer to generate a context-aware embedding. We prepend a `[CLS]` token to each input sequence to aggregate and summarize the encoded representation of each lane segment. This yields a fixed-dimensional embedding that captures both the geometry and local context of each segment.

A major computational bottleneck in multi-agent trajectory forecasting arises from the need to re-encode the map for each agent in their local (agent-centric) reference frame. For N agents, this results in $\mathcal{O}(N)$ repeated encodings, which significantly increases computational cost.

To mitigate this issue, we propose a two-stage approach:

1. **Global Map Encoding**: We encode the lane graph once in the global reference frame using the Transformer encoder. This produces a set of globally-referenced lane segment features.
2. **Agent-Centric Transformation**: For each agent, we extract its position and heading (x, y, θ) and apply a transformation to rotate and translate the globally encoded lane graph features into the agent's local frame. Specifically, we compute the center point of each lane segment, transform it, and then encode the transformed coordinates using a multi-layer perceptron (MLP). We concatenate the MLP output with the corresponding global lane segment embedding.

This approach yields N agent-centric lane graph representations, one per agent, while avoiding redundant map encoding. The resulting embeddings incorporate agent-specific spatial context. This method significantly reduces the computational complexity and improves scalability in multi-agent forecasting scenarios.

3.2.2 Aggregator

Following the generation of agent-centric map encodings, the **Aggregator** module is responsible for modeling future agent behavior by integrating spatial context with learned agent dynamics. This module comprises three key components: *Policy Learning*, *Future Sampling*, and *Fusion*. Together, these components enable the prediction of plausible future trajectories by reasoning over the lane graph in a structured and context-aware manner.

Policy Learning

With the agent-centric lane graph representations and historical encodings in place, the next step involves learning a **policy** that guides an agent's traversal through the lane graph. This policy is designed to model the likelihood of an agent transitioning from one lane segment to another, based on spatial structure and the agent's motion history.

Formally, for each possible edge (u, v) in the lane graph—where u is the source node and v is a candidate destination node—we compute a score that reflects the agent's preference for selecting that edge. These scores are obtained via a multi-layer perceptron (MLP)-based scoring function defined as follows:

$$\text{score}(u, v) = \text{MLP}(u_e, v_e, h_a, 1_{e \in \text{succ}}) \tag{3.1}$$

Here:

- u_e and v_e denote the feature encodings of the source and destination lane segments, respectively.
- h_a represents the agent's historical encoding, capturing its motion and behavior up to the current timestep.
- $1_{e \in \text{succ}}$ is a binary edge type indicator defined as:

$$\text{edge.emb} = \begin{cases} 1, & \text{if } e \text{ is a successor edge} \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

Using this scoring function, we compute a score for each of the outgoing edges from the current node u. To obtain a probabilistic policy over possible next steps, we apply a softmax over all candidate destination nodes v' connected to u:

$$P(e_{u,v} \mid h_a) = \frac{\exp(\text{score}(u, v))}{\sum_{v'} \exp(\text{score}(u, v'))} \tag{3.3}$$

This softmax distribution defines the agent-specific policy over lane graph transitions. At inference time, this policy allows the agent to select or sample likely future lane segments by navigating through the lane graph in a probabilistic manner.

Future Sampling

Once the policy described above has been learned, we leverage it to sample plausible future paths for each agent within the lane graph. To initiate this process, we first determine an *initial node* for the agent—defined as the closest lane segment that the agent is currently occupying based on its spatial coordinates. This node serves as the starting point for traversal within the graph.

Using the initial node as the root, we perform a forward sampling procedure along the lane graph guided by the learned policy. At each step of this process, the agent samples one of the outgoing edges based on the softmax probabilities produced by the policy network. This sampling is repeated iteratively for a fixed number of steps L, resulting in a sequence of lane segments that represent a high-level prediction of the path the agent is likely to follow.

To account for the possibility that an agent may come to a stop before completing the full L-step traversal, we incorporate a special **terminator edge** at each node. This edge allows the policy to choose a stopping action, effectively ending the traversal early if the agent's motion history suggests that it is likely to slow down or stop. This mechanism adds flexibility to the future sampling process and enables the model to capture varying behavior modes such as deceleration or halting at intersections.

The result of this process is a set of sampled paths for each agent, where each path corresponds to a sequence of lane segments that the agent is predicted to occupy over the planning horizon. These sampled trajectories serve as structured hypotheses about the agent's future behavior and are subsequently used in the fusion stage for downstream trajectory prediction.

Fusion

In the final stage of the Aggregator module, we integrate the sampled future paths with the agent's temporal dynamics to produce a unified representation that captures both spatial and temporal context. This is achieved through a cross-attention mechanism that fuses the high-level path information derived from the lane graph with the agent's motion history encoding.

For each agent, the sampled path consists of a sequence of L lane graph nodes representing the predicted future lane segments the agent is likely to occupy. We enhance this sequence with **positional encodings** to preserve the order and temporal alignment of the lane segments across the prediction horizon.

To perform fusion, we employ a **multi-headed cross-attention** mechanism. Specifically:

- The agent's motion encoding, which summarizes its historical trajectory and behavioral features, is used as the **query**.
- The sampled lane segment embeddings, augmented with positional encodings, are used as the **keys** and **values**.

The cross-attention module enables the agent's motion history to attend to different parts of the sampled path, allowing it to extract relevant spatial cues from the predicted future context. This fusion effectively incorporates information about where the agent is likely to go (spatial) and how it has been moving (temporal) into a unified feature representation.

The resulting fused representation serves as a rich, context-aware input to the downstream decoder, which utilizes it for fine-grained trajectory generation or action prediction. This fusion step is critical in ensuring that predictions are grounded in both the dynamic behavior of the agent and the structure of the driving environment.

3.2.3 Decoder

The final stage of the trajectory forecasting pipeline is the **Decoder**, which is responsible for generating precise future predictions based on the fused spatio-temporal representations obtained from the Aggregator module.

We employ two parallel multi-layer perceptron (MLP) heads to decode the final output:

- The first MLP head predicts a sequence of (x, y) coordinates corresponding to the agent's predicted positions over the next T_F timesteps.
- The second MLP head estimates a **confidence score** for each predicted trajectory modality, representing the model's belief in the likelihood of that particular future.

This design allows the model to produce multimodal trajectory predictions, where each modality corresponds to a distinct plausible future, and the associated confidence values indicate the relative likelihoods. The use of multiple prediction heads enables the system to capture the inherent uncertainty and multimodality present in real-world driving behavior.

Overall, the Decoder translates the high-level, fused representations into Future time-indexed spatial predictions for each agent.

3.3 Implementation Details

This chapter outlines the practical aspects of how our proposed model is implemented and trained. It provides a comprehensive overview of architectural configurations, training protocols, optimization strategies, and experimental settings used.

3.3.1 Training Setup

Our model utilizes an embedding and hidden representation dimension of 32 across all modules. Training is performed using the Adam optimizer with an initial learning rate of 0.001. To improve convergence, we apply a learning rate scheduler with a step size of 10 epochs and a decay factor $\gamma = 0.5$, progressively reducing the learning rate as training progresses.

The model is trained for 100 epochs on a single NVIDIA A100 GPU. We employ the Leaky ReLU activation function throughout the network to allow gradient flow through negative inputs, which has been shown to improve stability over standard ReLU in some settings.

Figure 3.2: samples from Nuscenes dataset

All model components, including the transformer-based map encoder, policy network, and decoder MLPs, share this consistent configuration unless otherwise specified.

3.3.2 Dataset Description

The nuScenes[3] and Argoverse 2[2] datasets are widely used benchmarks for motion forecasting in autonomous driving. nuScenes, developed by Motional (formerly known as nuTonomy), consists of 1,000 urban driving scenes from Boston and Singapore, captured with a 360° sensor suite including cameras, radars, and Lidar. It provides annotations for 23 object classes at 2 Hz, with a focus on predicting future agent trajectories over a 6-second horizon given past history of a maximum of 2 seconds. The dataset encourages multimodal prediction to account for uncertainty in future motion. Another dataset, Argoverse 2, created by Argo AI, offers over 250,000 scenarios from diverse urban, suburban, and rural environments across various U.S. cities. Each scenario includes 5 seconds of motion history and 6 seconds of future trajectory, annotated at 10 Hz. Argoverse 2 also provides high-definition maps with lane graphs and traffic elements, supporting map-based forecasting tasks. Both datasets enable research into complex, real-world motion prediction but differ in their geographic coverage and temporal resolution.

Datasets such as Argoverse 2 [2] and nuScenes [3] leverage LiDAR, camera, and radar sensors to obtain 3D coordinates of objects within a scene. In Argoverse 2, forecasting tasks are performed for vehicles, cyclists, and pedestrians, whereas the nuScenes dataset focuses solely on vehicle trajectory prediction as part of its challenge. Both datasets provide high-definition (HD) maps with centimeter-level (cm-level) resolution per pixel, facilitating fine-grained scene understanding. Additionally, they offer various

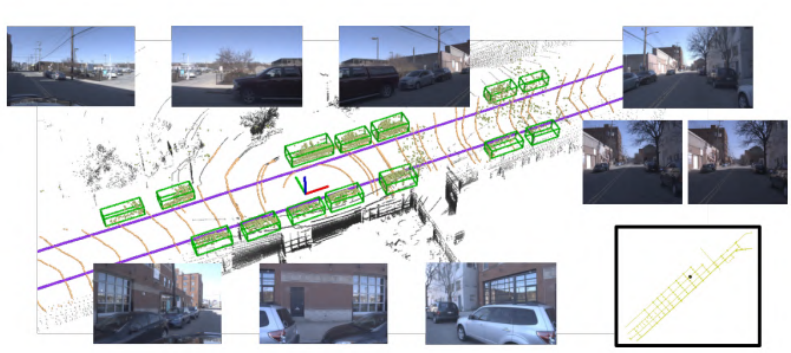


Figure 3.3: samples from Argoverse 2 dataset

classifications of trajectories based on the type of motion and the kinematic difficulty associated with predicting the trajectory. Figure 3.4 illustrates the different trajectory categories and the kinematic complexity distributions.

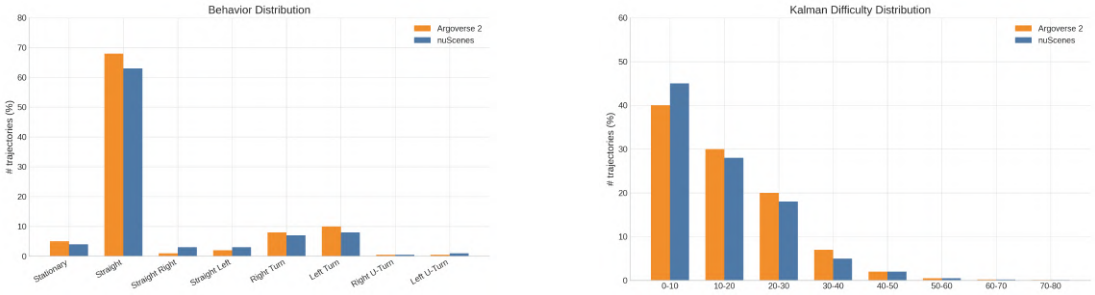


Figure 3.4: Types of Trajectories and Kalman difficulty of Trajectories in Nuscenes and Argoverse 2 Dataset

3.3.3 Loss Functions

We train our model in an end-to-end manner using two distinct loss functions: one for learning the trajectory decoding (motion prediction), and another for learning the graph traversal policy.

Trajectory Decoding Loss (MinADE). For trajectory forecasting, we use the Minimum Average Displacement Error (MinADE) as the loss function. Given M predicted trajectory modes for an agent, and the ground-truth trajectory $\tau^{\text{gt}} = \{(x_t^{\text{gt}}, y_t^{\text{gt}})\}_{t=1}^{T_F}$, we compute the Average Displacement Error (ADE) for each predicted trajectory τ^m as:

$$\text{ADE}^m = \frac{1}{T_F} \sum_{t=1}^{T_F} \left\| (x_t^m, y_t^m) - (x_t^{\text{gt}}, y_t^{\text{gt}}) \right\|_2 \tag{3.4}$$

We then select the mode with the minimum ADE across all M modes:

$$\mathcal{L}_{\text{ADE}} = \min_{m \in \{1,\dots,M\}} \text{ADE}^m \tag{3.5}$$

This "winner-takes-all" strategy encourages the model to produce at least one trajectory that closely matches the ground truth.

Confidence Head Loss. Alongside the trajectory coordinates, the model also predicts a confidence score $c^m \in (0, 1)$ for each mode m, indicating the model's belief in that mode being the most accurate. We train this confidence head such that the mode with the lowest ADE^m is assigned the highest confidence. Let $m^* = \arg\min_m \text{ADE}^m$ be the index of the best mode. The confidence loss is then defined as a cross-entropy loss over the mode index:

$$\mathcal{L}_{\text{conf}} = -\log \frac{\exp(c^{m^*})}{\sum_{j=1}^{M} \exp(c^j)} \tag{3.6}$$

This formulation encourages the predicted confidence distribution to place maximal probability on the most accurate trajectory.

Total Loss. The total loss is a weighted sum of the trajectory decoding loss and the confidence loss:

$$\mathcal{L} = \mathcal{L}_{\text{ADE}} + \lambda_{\text{conf}} \mathcal{L}_{\text{conf}} \tag{3.7}$$

where λ_{conf} controls the relative importance of the confidence head during training.

Policy Learning Loss (Behavior Cloning). For learning the lane graph traversal policy, we adopt a behavior cloning strategy. Specifically, we use the ground truth future trajectory to determine the set of edges E_{gt} that were traversed by the agent in the lane graph. To address ambiguity in intersections, only those lane segments whose direction aligns with the agent's yaw within a certain threshold are considered.

Given the model's predicted edge distribution $\pi_{\text{route}}(v|u)$ at node u, we define the policy loss as the negative log-likelihood over the ground-truth edge transitions:

$$\mathcal{L}_{\text{BC}} = \sum_{(u,v) \in E_{gt}} -\log(\pi_{\text{route}}(v|u)) \tag{3.8}$$

This corresponds to a cross-entropy loss where edges actually traveled are assigned a probability of 1, and all others 0. This guides the policy network to match the routing behavior observed in the data.

Total Loss. The total training loss combines both components:

$$\mathcal{L}_{\text{total}} = \mathcal{L} + \lambda \cdot \mathcal{L}_{\text{BC}} \tag{3.9}$$

where λ is a hyperparameter used to balance the importance of the trajectory and policy losses.

3.4 Results

Evaluation Metrics We evaluate the performance of our trajectory prediction model using standard multimodal metrics: **MinADE@K**, **MinFDE@K**, and **MissRate@K**, where $K \in \{3, 6, 5, 10\}$ based on dataset. These metrics assess the accuracy and diversity of the predicted trajectories by comparing them with the ground truth over T_f time steps.

- **Average Displacement Error (ADE)** is defined as the average L2 distance between predicted and ground truth coordinates over all time steps:

$$\text{ADE}(\hat{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{T} \sum_{t=1}^{T} \|\hat{\mathbf{y}}_t - \mathbf{y}_t\|_2 \tag{3.10}$$

- **Final Displacement Error (FDE)** measures the L2 distance between the predicted final position and the ground truth final position:

$$\text{FDE}(\hat{\mathbf{Y}}, \mathbf{Y}) = \|\hat{\mathbf{y}}_T - \mathbf{y}_T\|_2 \tag{3.11}$$

- **MinADE@K** is the minimum ADE over K predicted trajectories:

$$\text{MinADE@K} = \min_{k \in \{1,\ldots,K\}} \text{ADE}(\hat{\mathbf{Y}}^{(k)}, \mathbf{Y}) \tag{3.12}$$

- **MinFDE@K** is the minimum FDE over K predicted trajectories:

$$\text{MinFDE@K} = \min_{k \in \{1,\ldots,K\}} \text{FDE}(\hat{\mathbf{Y}}^{(k)}, \mathbf{Y}) \tag{3.13}$$

- **MissRate@K** is the proportion of samples for which all K predicted final positions are farther than a threshold δ (typically 2 meters) from the ground truth final position:

$$\text{MissRate@K} = \frac{1}{N}\sum_{i=1}^{N} 1\left[\min_{k\in\{1,\ldots,K\}} \left\|\hat{\mathbf{y}}_T^{(i,k)} - \mathbf{y}_T^{(i)}\right\|_2 > \delta\right] \tag{3.14}$$

Here, $\hat{\mathbf{Y}}^{(k)} = (\hat{\mathbf{y}}_1^{(k)}, \ldots, \hat{\mathbf{y}}_T^{(k)})$ denotes the k-th predicted trajectory, $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_T)$ is the ground truth trajectory, and N is the number of samples in the dataset.

Quantitative Evaluation. Table 3.1 presents the comparison of our method against existing state-of-the-art approaches on the Argoverse2 and NuScenes datasets. On Argoverse2, our model achieves competitive performance with a MinADE6 of 0.801 and MinFDE6 of 1.586, closely matching the performance of strong baselines such as EMP-Net and QCNet. On the more challenging NuScenes dataset, our method shows strong generalization capabilities, outperforming several recent methods including GOHOME, FDNET, and PGP. Specifically, our approach achieves a MinADE5 of 1.23 and MinADE10 of 0.92, and yields the lowest miss rates (0.50 for top-5 and 0.32 for top-10 trajectories).

Table 3.1: Comparison of methods on Argoverse2 validation set and NuScenes datasets. (- indicates data not reported)

Argoverse 2 Dataset		
Method	**MinADE6 ↓**	**MinFDE6 ↓**
QCNet	0.72	1.25
EP-F	0.786	1.526
EMPNet	0.71	1.39
Ours	**0.801**	**1.586**

NuScenes Dataset

Method	**MinADE5 ↓**	**MinADE10 ↓**	**MissRate5 ↓**	**MissRate10 ↓**	**MinFDE5 ↓**	**MinFDE10 ↓**
GOHOME	1.41	1.15	0.57	0.47	–	–
FDNET	1.41	1.00	0.42	–	2.5152	–
Autobot	1.37	1.03	0.62	0.44	–	–
THOMAS	1.33	1.04	0.55	–	–	–
PGP	1.30	1.00	0.52	0.34	–	–
SemanticFormerR	**1.14**	**0.87**	0.50	0.32	2.20	–
Ours	1.23	0.92	**0.50**	**0.32**	**2.34**	**1.47**

Ablation Study. As shown in Table 3.3, we perform an ablation study on the encoder architecture to understand the impact of sequence modeling choices. When using a GRU-based encoder, the model achieves a MinADE5 of 1.28 and MinADE10 of 0.95 on nuscenes [3]. Replacing the GRU with a transformer-based encoder yields a noticeable improvement, reducing MinADE5 to 1.23 and MinADE10 to 0.92 on nuscenes[3] dataset. The same trend is amplified in the Argoverse 2 Dataset due to increase in temporal Length. This suggests that transformer encoders are more effective in capturing long-range dependencies and masking padding to allow for dynamic history length.

Table 3.2: Ablation study for encoder architecture on Nuscenes Dataset.

Encoder Type	**MinADE5** ↓	**MinADE10** ↓
GRU	1.28	0.95
Transformers	**1.23**	**0.92**

Table 3.3: Ablation study for encoder architecture on Argoverse 2 Dataset.

Encoder Type	**MinADE6** ↓	**MinFDE6** ↓
GRU	0.984	1.889
Transformers	**0.801**	**1.586**

Efficiency. Table 3.4 reports the training and inference times for our method on two different GPU configurations. On an NVIDIA A100 GPU, training takes approximately 5.5 hours with an inference latency of 25 milliseconds per scenario. On a lower-end NVIDIA 2050 GPU, training requires 23.3 hours with a scenario-level inference time of 28 milliseconds. Unlike baseline approaches that report inference time on a per-agent basis—typically around 13 milliseconds per agent—our method reports inference time at the scenario level, encompassing all agents simultaneously. As a result, our method exhibits more stable performance as the number of agents increases. While the compute time for agent-wise models scales linearly with the number of agents, our scenario-level inference time remains largely consistent, increasing only marginally by a few milliseconds even when the number of agents exceeds 15 per scene. This makes our approach well-suited for deployment in dense urban scenarios where multiple agents must be jointly forecasted in real time.

Table 3.4: Training and inference times.

GPU	**Training Time** ↓	**Inference Time** ↓
NVIDIA A100 (20GB MIGs)	5.5 Hours	25ms
NVIDIA 2050	23.3 Hours	28ms

Comparison of Error Distributions Between Our Model and PGP. Figure 3.5 presents a comparative analysis of our model against the PGP baseline using histograms of minimum Final Displacement Error (MinFDE) and Average Displacement Error (MinADE) at prediction horizons of 5 and 10 time steps. Across all four plots—Min FDE 5, Min FDE 10, Min ADE 5, and Min ADE 10—our model (shown in orange) consistently demonstrates a left-shifted distribution relative to PGP (in blue), indicating lower prediction errors over a majority of samples.

Notably, for both short-term (5-step) and long-term (10-step) horizons, our model yields a higher concentration of predictions within low-error bins, especially in the 0–2 meter range. This pattern suggests improved trajectory precision and robustness. In contrast, the PGP model exhibits a longer tail with more frequent occurrences of higher error values, particularly evident in the Min FDE 5 and Min ADE 5 plots. The tighter spread of errors and reduced outliers in our model's distribution highlight its capability to generate more reliable predictions, even under uncertainty.

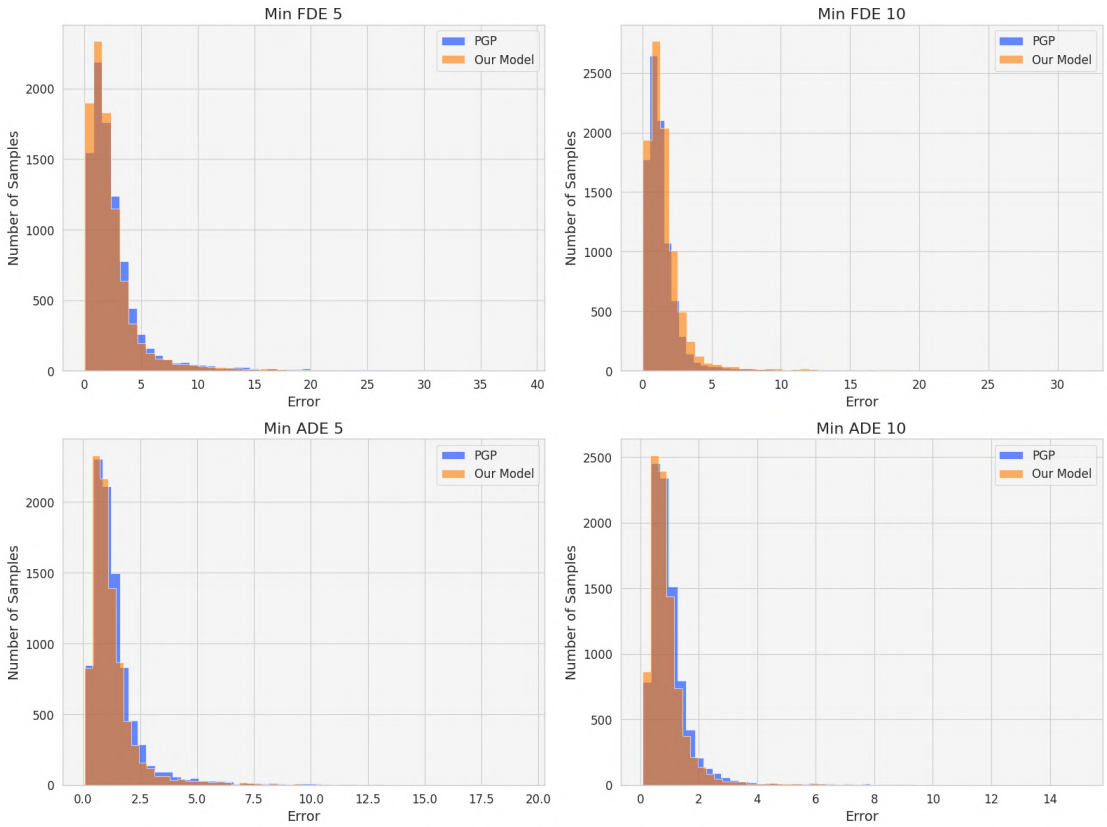


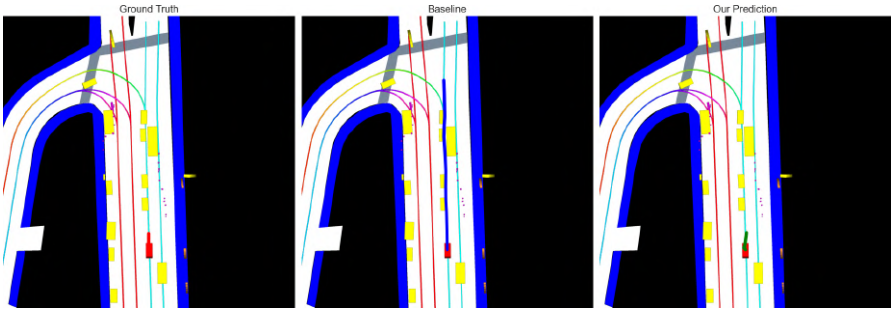
Figure 3.5: Histogram comparison of minADE and minFDE at 5s and 10s prediction horizons between PGP and our model. Our model shows a clear shift towards lower errors in all metrics, indicating better trajectory prediction accuracy.

3.4.1 Qualitative Results

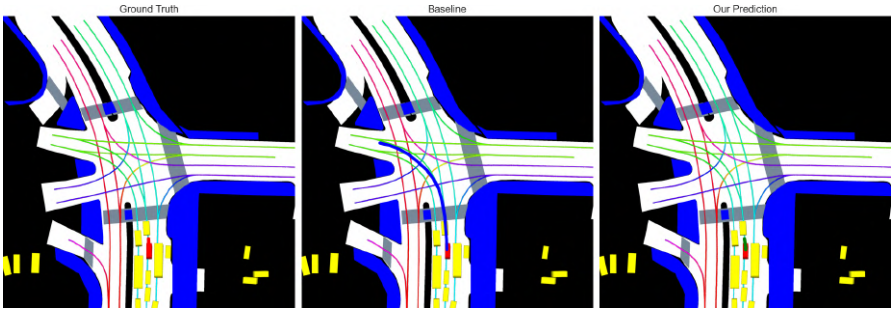
Traffic Understanding Figure 3.6 presents a qualitative analysis of vehicle behavior at a traffic signal. In the absence of agent-agent interaction modeling, the baseline model incorrectly predicts the target vehicle moving forward, failing to account for the surrounding vehicles waiting at the red light. In contrast, our model—which explicitly incorporates agent-agent interactions—correctly captures the contextual behavior of the scene, predicting that the vehicle remains stationary as all agents are waiting at the signal.

Lane Occupancy Figure 3.7 illustrates a scenario where lane occupancy plays a critical role in prediction. The turning lanes are occupied by moving agents, which mislead the baseline model into making inaccurate predictions—assuming that the lane is unavailable or incorrectly modeling the agent's trajectory. In contrast, our model, which leverages agent-to-agent interactions, produces more accurate predictions by understanding that the agent is following another vehicle and adjusting its future trajectory accordingly.

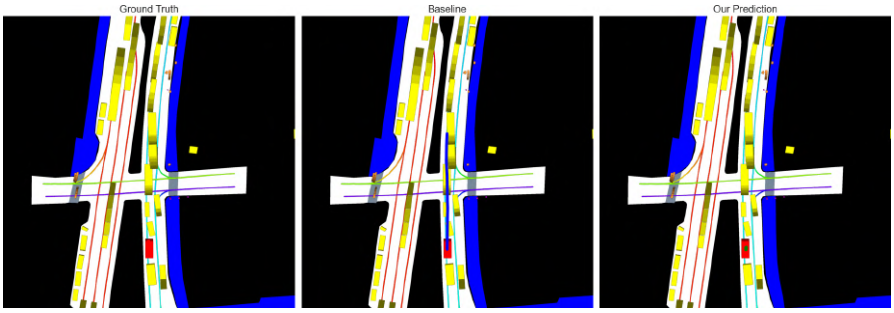
Prediction with various history lengths Figure 3.8 illustrates the impact of varying the number of observed historical timesteps on trajectory prediction performance. In the visualizations, different hue intensities of the agent boxes represent the number of historical timesteps, with lighter shades indicating more recent positions. Despite the variation in input history length—from a single timestep up to four timesteps—our model demonstrates robust and consistent prediction quality. The predicted trajectories remain well-aligned with the ground truth across all settings, highlighting the model's ability to reason effectively about agent behavior even with variable historical context.



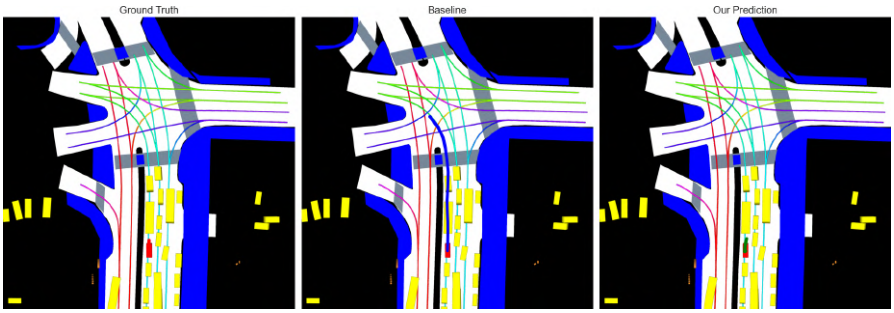
(a) Scenario 1



(b) Scenario 2

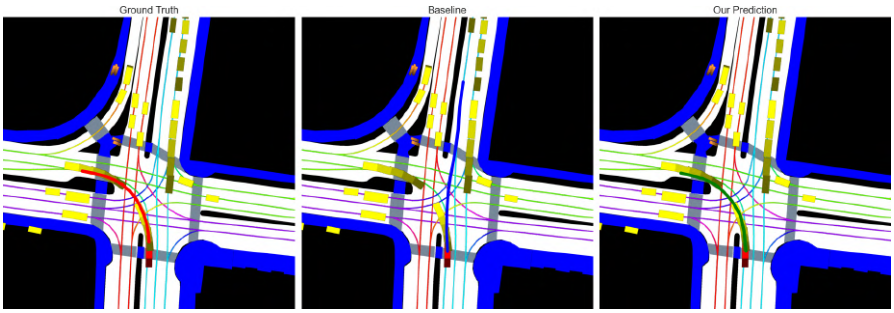


(c) Scenario 3

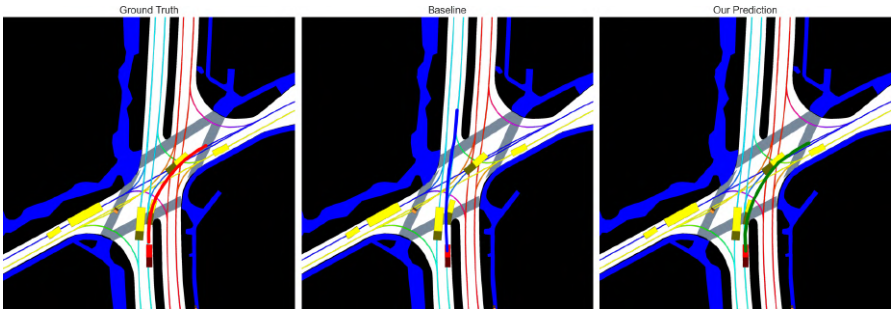


(d) Scenario 4

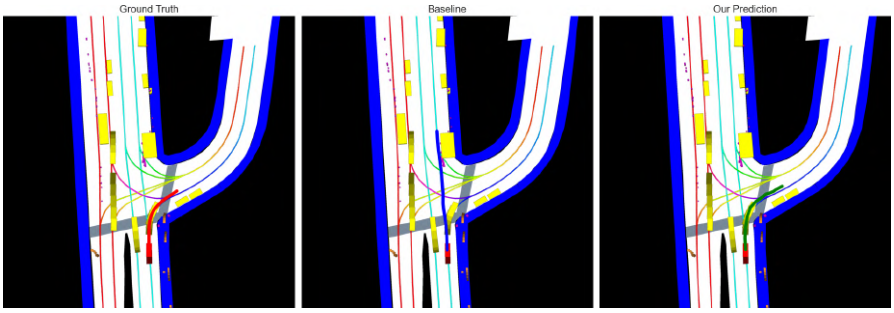
Figure 3.6: Qualitative results across four different scenarios. In each case, our model shows improved prediction accuracy over the baseline by incorporating agent-agent communication. For example, in traffic signal situations, our model correctly infers that nearby vehicles are stopped, avoiding incorrect forward trajectory predictions. In contrast, the baseline model, lacking social context, fails to make this distinction. These examples highlight the benefit of social awareness in trajectory prediction.



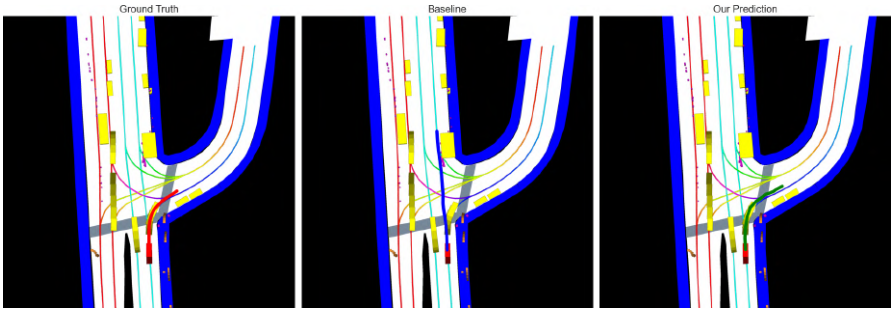
(a) Scenario 1



(b) Scenario 2

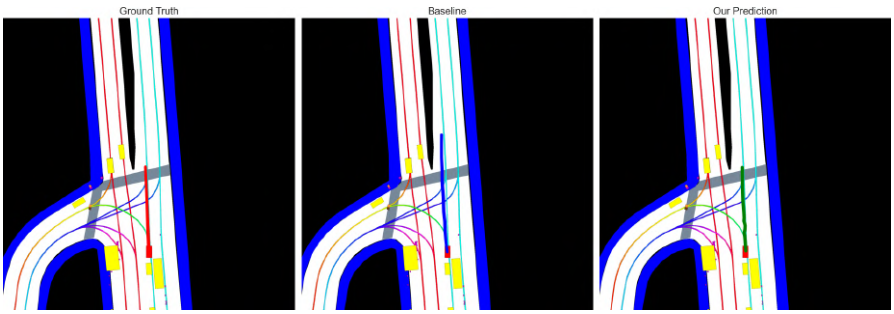


(c) Scenario 3

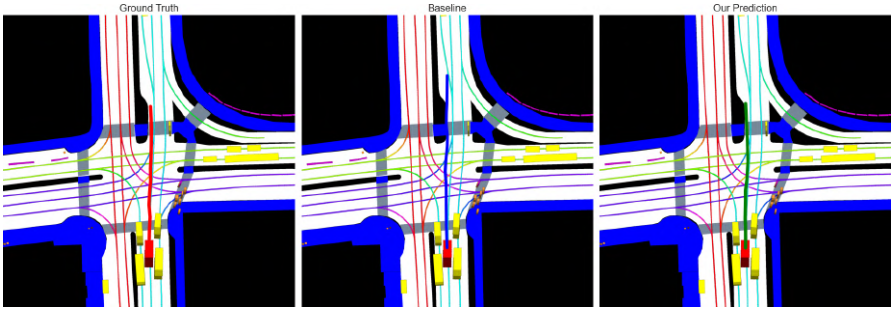


(d) Scenario 4

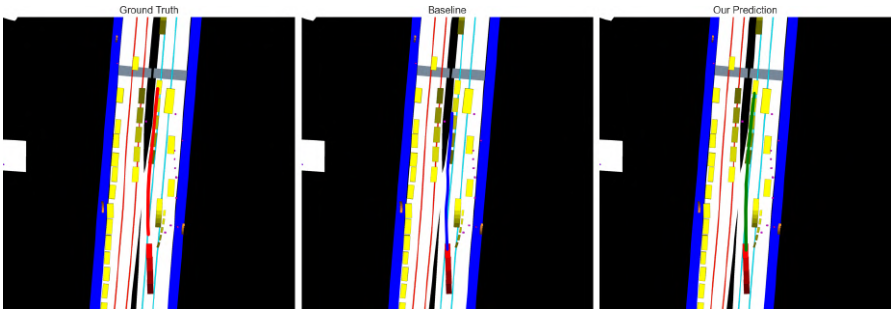
Figure 3.7: Qualitative results across different scenarios illustrating the limitations of the baseline model where agents are simply embedded into the map to update their positions. In these cases, the predicted trajectories often go awry due to the lack of direct agent-agent interaction. In contrast, our model, which explicitly incorporates agent-agent communication, is able to learn complex behaviors such as car-following and the dynamic evolution of surrounding agents, resulting in more accurate and socially aware trajectory predictions.



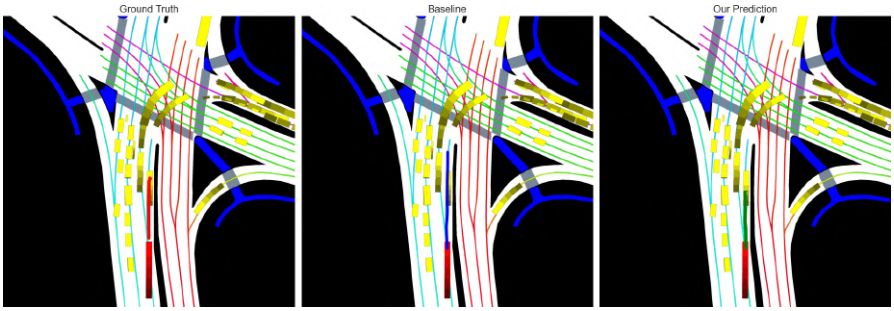
(a) Scenario 1



(b) Scenario 2



(c) Scenario 3



(d) Scenario 4

Figure 3.8: In the visualizations, different hue intensities of the agent boxes represent the number of historical timesteps, with lighter shades indicating more recent positions. As shown in the examples, our model maintains consistent prediction quality even when the number of observed historical timesteps varies from a single timestep up to four timesteps, closely aligning with the ground truth in each case.

CHAPTER 4

Motion Forecasting Using Monocular Camera

This chapter focuses on predicting the future positions of moving vehicles using input from a single (monocular) camera. The process starts by tracking vehicles in the video frames and estimating the depth of the scene to understand how far objects are from the camera. After this, we estimate the current state of each vehicle, including its position and speed. With this state information, we use trajectory prediction models to forecast where the vehicles are likely to go in the near future. These predictions can be used for planning, safety, and a better understanding of traffic behavior.

4.1 Problem Formulation

We address the problem of motion forecasting for road agents using a monocular RGB camera with a static viewpoint. The goal is to predict future trajectories of vehicles observed in the scene by transforming image-space observations to a bird's-eye view (BEV) representation using monocular depth estimation. Importantly, the system operates without access to high-definition (HD) maps or external localization signals.

Input

- A sequence of RGB frames from a static monocular camera:

$$\mathcal{I} = \{I_t\}_{t=1}^{T}, \quad I_t \in R^{H \times W \times 3}$$

  where H and W are the height and width of each frame, and T is the number of observed frames.

Output

The objective is to predict the future positions of each agent $a \in \mathcal{A}$ over a prediction horizon of H_f frames:

$$\hat{\mathcal{F}}^a = \{\hat{\mathbf{p}}_t^a\}_{t=T+1}^{T+H_f}, \quad \hat{\mathbf{p}}_t^a \in R^2$$

Assumptions and Constraints

- The camera is static.
- No HD map or prior knowledge of the scene geometry is available.
- Depth estimation is performed on monocular images without auxiliary sensors (e.g., LiDAR).
- All processing occurs in the BEV coordinate frame, which is assumed to approximate the local road plane.

4.2 Challenges with Monocular Motion Forecasting

Motion forecasting in autonomous driving systems heavily relies on accurate perception of the surrounding environment. Multi-sensor setups, incorporating LiDAR and radar, have become the standard in many state-of-the-art approaches due to their ability to provide rich and precise 3D information. These sensors are typically cross-calibrated to a shared coordinate frame, which allows for accurate spatial localization of dynamic agents (e.g., vehicles, pedestrians) in the environment. This precise localization enables forecasting models to reason about trajectories in metric 3D space, improving both the feasibility and accuracy of predicted motion paths.

Furthermore, multi-sensor systems facilitate the construction of High-Definition (HD) maps, which provide detailed structural and semantic information about the driving environment. These maps include lane centerlines, road boundaries, traffic signals, crosswalks, and other contextual cues. Conditioning motion forecasting models on HD map information enables them to generate socially and physically plausible trajectories that align with road rules and infrastructure geometry. Predictions can thus be better constrained and more robust, especially in complex urban scenarios.

In contrast, monocular motion forecasting relies solely on single RGB images, which introduces significant challenges. A single image lacks depth information, making it difficult to infer the accurate 3D positions and velocities of agents. While learning-based methods can approximate depth and motion cues, they are inherently less reliable than direct sensor measurements and also lack real-time applications.

Additionally, monocular setups do not inherently provide the environmental geometry needed to construct an HD map. As a result, predictions made without access to

lane centerlines or road boundaries often lack the necessary spatial constraints. This limitation significantly impacts model performance in structured environments where road semantics play a critical role in driving behavior. Figure 4.1 illustrates the impact of introducing noise to the lane features used in our model. As shown, the addition of significant noise consistently degrades prediction quality across all evaluation metrics. This sensitivity highlights the model's reliance on accurate lane geometry. For this reason, we do not attempt to reconstruct or estimate the map from RGB images. Instead, we explore a map-free trajectory prediction paradigm that relies solely on agent dynamics and interactions, making our approach better suited for real-world deployment where map information may be unavailable.

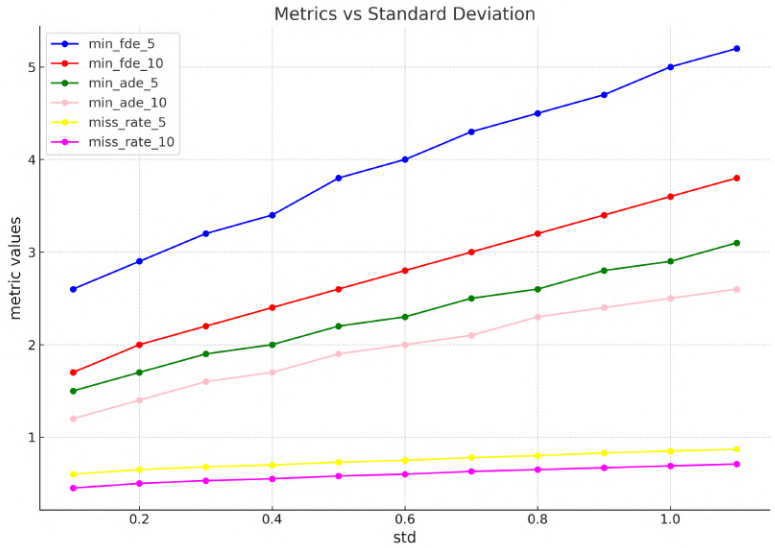


Figure 4.1: Effect of increasing standard deviation of zero-mean Gaussian noise added to map features. The metrics shown (minFDE, minADE, and miss rate for 5 and 10 samples) consistently degrade as noise increases, highlighting the sensitivity of trajectory prediction models to map uncertainty. This experiment demonstrates that even small amounts of noise can lead to significant performance deterioration, making precise HD map estimation not only impractical but also potentially unnecessary.

4.3 Proposed Methodology

This section outlines the key components of our motion forecasting framework using a monocular camera. The objective is to detect and track dynamic agents in the scene, estimate their states over time, and ultimately forecast their future trajectories based on

past observations. Our pipeline is designed to work in real-world urban scenarios where only a monocular RGB camera is available, making the task more challenging due to the lack of direct depth information.

4.3.1 Object Detection and Tracking

In this stage of our pipeline, we employ YOLOv8 for real-time object detection and tracking. YOLOv8, a state-of-the-art one-stage detector, is used to identify vehicles within each frame captured by the monocular camera. The model is capable of detecting multiple object classes; however, in our case, we focus on detecting vehicles as dynamic agents in the scene.

After detecting vehicles in a given frame, we apply an integrated tracking algorithm(ByteTrack) that assigns a unique identifier (ID) to each detected agent. This allows us to consistently track the same vehicle across successive frames. For every tracked agent, we maintain a record of its historical positions by storing its bounding box coordinates and estimated center point over time. This temporal information is essential for downstream tasks such as state estimation and motion forecasting.

he detection and tracking system is robust to moderate occlusions and maintains identity persistence even in complex urban environments, outputting a structured stream of per-frame vehicle tracks—each represented as a time series of positions with a persistent agent ID.

4.3.2 State Estimation

For each incoming frame from the monocular camera, we pass the image through a monocular depth estimation module that predicts a dense depth map for the scene. This depth map provides per-pixel distance estimates from the camera to points in the environment. Although monocular depth estimation is inherently scale-ambiguous, we mitigate this by calibrating the system with a known pixel-to-meter scale factor during setup.

Using the predicted depth map and the known camera intrinsic matrix, we project the 2D image into a 3D point cloud. This transformation allows us to generate a Bird's-

Eye View (BEV) representation of the scene, which provides a top-down spatial layout of the environment. In this BEV space, we locate the center positions of detected agents (vehicles) by projecting their bounding boxes into the ground plane.

As the scene progresses frame by frame, we monitor how each agent's center position shifts in the BEV map. By measuring the number of pixels an agent has moved along the X and Y axes and applying the pixel-to-meter scale factor, we estimate the agent's translation in real-world coordinates. This translation, along with temporal information, forms the agent's state data, including position and velocity.

The details regarding the implementation are discussed in section 4.5.2 .This state data becomes the only input for our motion forecasting module. Notably, we do not rely on an HD semantic map of the environment, as generating such maps using only monocular input is extremely challenging and often infeasible. Instead, we modify our trajectory prediction architecture to operate without explicit map information. Our system learns to infer motion patterns and interaction dynamics directly from agent state histories and their spatial relationships, allowing robust motion forecasting even in map-sparse environments.

4.3.3 Motion Prediction

In this stage, we utilize our Multi-Agent Motion Prediction Model, as described in the previous chapter, to forecast the future trajectories of vehicles in the scene. The input to this model is the state data generated in the previous stage, which includes each agent's temporal history of positions and velocities estimated from the BEV representation.

To adapt the model to our monocular-camera-only setup, we modify the original architecture by removing the map encoder and all related map encoding components. This modification is necessary because generating a high-definition (HD) semantic map using only monocular input is extremely difficult and unreliable in practical scenarios. Instead, our model relies solely on the temporal dynamics of each agent and their interactions with other agents.

The modified architecture learns motion patterns and behavior through the temporal history of agents without needing explicit knowledge of the surrounding road layout or traffic semantics. Despite the absence of map information, the model captures motion

cues, enabling it to generate accurate future trajectory predictions.

This approach ensures that our system remains deployable in environments where only a monocular camera is available.

4.4 Implementation Details

This section provides key details regarding the implementation of various components used in our system. We describe the models and tools employed for monocular depth estimation and motion prediction, along with relevant architectural modifications.

4.5 Dataset Description

To evaluate the performance of our proposed framework, we employed both publicly available and proprietary datasets. Specifically, we utilized the BrnoCompSpeed[23] dataset for benchmarking purposes and proprietary datasets provided by Vehant Technologies for testing and validation.

The BrnoCompSpeed[23] dataset consists of 21 video sequences captured under various traffic and environmental conditions, each around 1 hr long, captured at six different locations. Vehicles in the videos (20 865 instances in total) are annotated with the precise position and speed measurements from optical gates using LiDAR and verified with several reference GPS tracks.This dataset is widely recognized in the research community for its diverse scenarios and reliable ground truth, making it well-suited for benchmarking vehicle detection and speed estimation frameworks.

In addition to BrnoCompSpeed, we used video sequences from Vehant Technologies' proprietary dataset. These videos encompass a range of challenging real-world conditions, including variations in road geometry and traffic density, thereby providing a robust platform for testing the generalization capabilities of our framework.

Representative still frames from the BrnoCompSpeed dataset and the Vehant Technologies dataset are illustrated in Figure 4.2 and Figure 4.3, respectively. These figures highlight the diversity and complexity of the scenarios considered in our evaluation.

Figure 4.2: Sample frames from the BrnoCompSpeed dataset.

Figure 4.3: Sample frames from the Vehant Technologies proprietary dataset.

4.5.1 Monocular Depth Estimation

For monocular depth estimation, we evaluate three state-of-the-art models: Metric3D, DepthAnything V2, and ZoeDepth. These models are tested to determine their suitability for real-time depth estimation in a motion forecasting pipeline using a monocular camera.

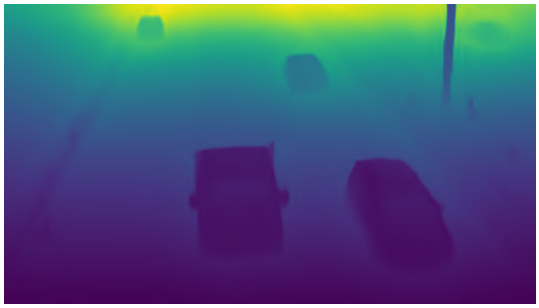
Metric3D is capable of producing metric-scale depth estimates, which are essential for accurate 3D localization. However, during our evaluation, we found that Metric3D suffers from significantly high latency, rendering it impractical for real-time applications where rapid inference is critical.

DepthAnything V2, on the other hand, provides fast inference and is designed for general-purpose depth prediction. Despite its efficiency, our testing revealed that DepthAnything V2 produced unreliable and noisy depth maps in various driving scenes, leading to inconsistent state estimation.

ZoeDepth strikes a balance between inference speed and prediction quality. It generates dense and stable depth maps with reasonably accurate scale estimation, making it a suitable choice for our monocular camera-based system. As a result, we adopt ZoeDepth as our default depth estimation module in the pipeline.

An example scenario is shown in Figure 4.5, where depth maps generated using ZoeDepth are visualized alongside their corresponding RGB images.

RGB Image 1



Depth Map 1

Figure 4.4: RGB images and corresponding monocular depth predictions using our depth estimation module.

4.5.2 BEV Map Generation and State Estimation

To generate a spatially consistent top-down view of the scene, we transform each monocular image and its corresponding depth map into a Bird's-Eye View (BEV) representation. This BEV map allows for accurate estimation of agent positions and movements over time in the ground plane.

We first estimate the depth of each RGB image using the monocular depth estimation module. This depth map is then used in conjunction with the camera intrinsic matrix, which models the pinhole camera's focal lengths and principal point. These intrinsics are calibrated beforehand and remain fixed throughout deployment.

Each pixel in the image is back-projected into 3D space using its depth value and the intrinsic parameters, resulting in a dense point cloud in the camera coordinate frame. We then apply a geometric transformation to align this point cloud with the ground plane by incorporating fixed pitch, yaw, and roll angles, along with a translation accounting for the camera's height above the ground.

The transformed 3D points are projected into a 2D grid to form the BEV image. A fixed scale (e.g., pixels per meter) is applied to convert metric distances to pixel coordinates, and irrelevant or out-of-range depth values are clipped to reduce noise. The RGB values from the original image are then used to color the BEV map.

This BEV map forms the spatial foundation for state estimation, enabling us to track how far an agent has moved between frames in real-world coordinates. This displacement information is used to build the agent's motion for trajectory prediction.

Mathematically, let $I_t \in R^{H \times W \times 3}$ be the input RGB image at time t, where H and W represent the height and width of the image. The monocular depth estimation function f_{depth} produces a depth map $D_t \in R^{H \times W}$:

$$D_t = f_{\text{depth}}(I_t)$$

Using the predicted depth map and the known camera intrinsic matrix $K \in R^{3 \times 3}$:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

We project each pixel (u, v) with depth $d = D_t(u, v)$ to a 3D point $P = [X, Y, Z]^T$ in the camera coordinate system:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = d \cdot K^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

The resulting 3D point cloud is then transformed into a Bird's-Eye View (BEV) representation. Let $\mathcal{P}_t$ denote the set of all 3D points derived from I_t.

We then project the detected agents onto this ground plane. For each agent a_i with 2D bounding box $b_i = [u_{\min}, v_{\min}, u_{\max}, v_{\max}]$, we compute its center position in the image plane:

$$c_i^{\text{img}} = \left(\frac{u_{\min} + u_{\max}}{2}, \frac{v_{\min} + v_{\max}}{2} \right)$$

Using the depth map and camera parameters, we project this center point to the ground plane to obtain the agent's position in the BEV space:

$$c_i^{\text{BEV}} = \Psi(c_i^{\text{img}}, D_t, K, \pi_{\text{ground}})$$

where Ψ maps image coordinates to BEV coordinates in pixel units, resulting in

$c_i^{\text{BEV}} = (x_i^{\text{pix}}, y_i^{\text{pix}})$.

At this stage, we apply the calibrated pixel-to-meter scale factor λ_{scale} to convert BEV pixel coordinates to real-world metric coordinates:

$$c_i^t = (x_i^t, y_i^t) = \lambda_{\text{scale}} \cdot c_i^{\text{BEV}} = \lambda_{\text{scale}} \cdot (x_i^{\text{pix}}, y_i^{\text{pix}})$$

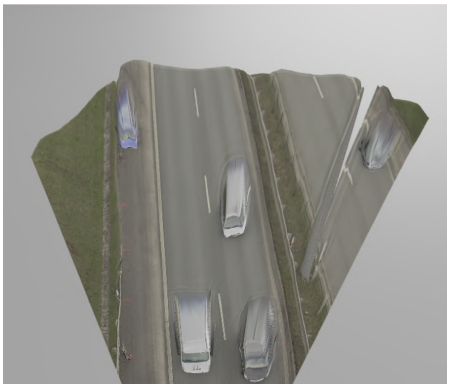
As the scene progresses frame by frame, we track each agent's position across consecutive time frames. For an agent a_i observed at times t and $t + \Delta t$, the displacement vector in metric coordinates is:

$$\Delta c_i^{t,t+\Delta t} = c_i^{t+\Delta t} - c_i^t = (x_i^{t+\Delta t} - x_i^t, y_i^{t+\Delta t} - y_i^t)$$

The velocity vector $v_i^t = (v_{x,i}^t, v_{y,i}^t)$ is computed as:

$$v_i^t = \frac{\Delta c_i^{t,t+\Delta t}}{\Delta t} = \left(\frac{x_i^{t+\Delta t} - x_i^t}{\Delta t}, \frac{y_i^{t+\Delta t} - y_i^t}{\Delta t} \right)$$

RGB Image 1



Depth Map 1

Figure 4.5: RGB image and corresponding BEV map.

4.5.3 Motion Prediction Models

For motion prediction, we utilize two distinct models: our multi-agent prediction model trained on the standard nuScenes dataset, and HiVT (Hierarchical Vector Transformer), a motion forecasting model trained on the Argoverse 2 dataset. Both models are designed to forecast future trajectories of dynamic agents in urban driving environments.

In their original form, these models rely on map-based inputs, including lane graphs and HD semantic maps, to inform predictions. However, in our monocular camera-based setup, such map information is unavailable or unreliable. To address this, we modify the architecture of both models by removing components responsible for encoding and fusing lane and map features.

Instead, we restructure the models to rely exclusively on the temporal history of agent states, such as position and velocity over time, derived from our monocular-based state estimation pipeline. This adjustment allows the models to function without map priors, making them compatible with monocular-only systems.

We retrain both the modified models on their respective datasets—nuScenes and Argoverse 2—using only agent past trajectories as input. This training enables the models to learn motion patterns and interactions purely from historical behavior, without the aid of explicit environmental context.

To enable map-free trajectory prediction, we modify our original architecture by removing the lane graph encoding and policy traversal components. Instead, we leverage a multi-headed attention mechanism to directly model the spatial and temporal relationships between agents. This allows the model to reason about agent interactions without relying on explicit map information. Figure 4.6 shows our modified architecture.

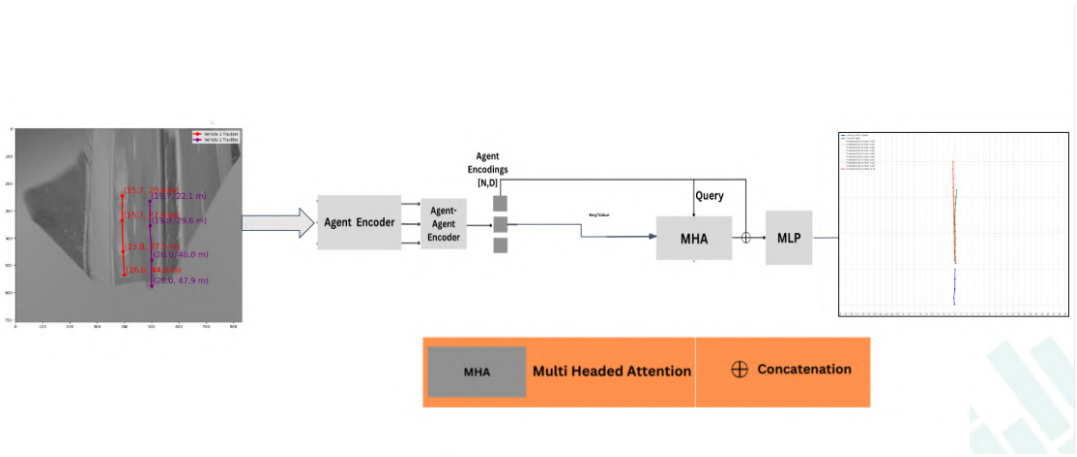


Figure 4.6: Architecture of the modified version of our model, which enables map-free trajectory prediction by removing the dependency on explicit lane features and relying solely on agent dynamics and interactions.

4.6 Results

Quantitative Results Table 4.1 presents a comparison of the minimum Average Displacement Error (minADE) for our model and HiVT across varying future prediction horizons ranging from 1 to 6 seconds. The results are reported on the BrnoComp dataset. It is important to note that our model was trained on the nuScenes dataset, while HiVT was trained on the Argoverse 2 dataset.

Future Horizon (sec)	HiVT (minADE in meters)	MAPGP (minADE in meters)
1	0.11	0.38
2	0.31	0.52
3	0.43	0.68
4	0.62	1.03
5	0.84	1.48
6	0.96	1.68

Table 4.1: Comparison of minADE with our model and HiVT when used for motion prediction across different future lengths when evaluated on BrnoComp Dataset

Latency Table 4.2 summarizes the latency breakdown for the prediction pipeline, including both depth estimation and motion prediction stages. Both our model and HiVT use the same monocular depth estimation module, which takes approximately 90 milliseconds per frame. However, our motion prediction model is significantly more efficient, requiring only 26 milliseconds compared to HiVT's 39 milliseconds. This results in a lower total latency of 116 milliseconds for our model versus 129 milliseconds for HiVT, demonstrating the runtime efficiency of our approach.

Component	HiVT (ms)	Ours (ms)
Depth Estimation	90	90
Motion Prediction	39	26
Total Latency	**129**	**116**

Table 4.2: Comparison of prediction latency for HiVT and our model. Both use the same depth estimation module; our model demonstrates lower motion prediction latency.

State Estimation Error State estimation using monocular depth prediction inherently introduces error due to the lack of direct depth measurements. This uncertainty tends to increase with the distance of the object from the camera. As shown in Table 4.3,

position estimation remains relatively reliable (error < 0.6 meters) for agents within 20 meters of the camera. However, beyond this range, the depth estimation becomes increasingly unreliable, leading to a sharp rise in positional uncertainty. This limitation must be accounted for during downstream tasks such as trajectory forecasting or collision risk assessment.

Distance from Camera (m)	**Average Error in Position (m)**
5	0.12
10	0.23
15	0.41
20	0.58
25	1.03
30	1.93
35	2.97

Table 4.3: State estimation error as a function of distance from the camera. Uncertainty grows significantly beyond 20 meters.

4.6.1 Qualitative results

BEV Transformation Figure 4.7 illustrates the process of transforming images from a monocular camera into a Bird's Eye View (BEV) map. The raw images are first processed to extract relevant spatial features, which are then projected into the BEV space. By applying appropriate scaling factors in x and y directions, the positions in the image plane are accurately converted to real-world coordinates.

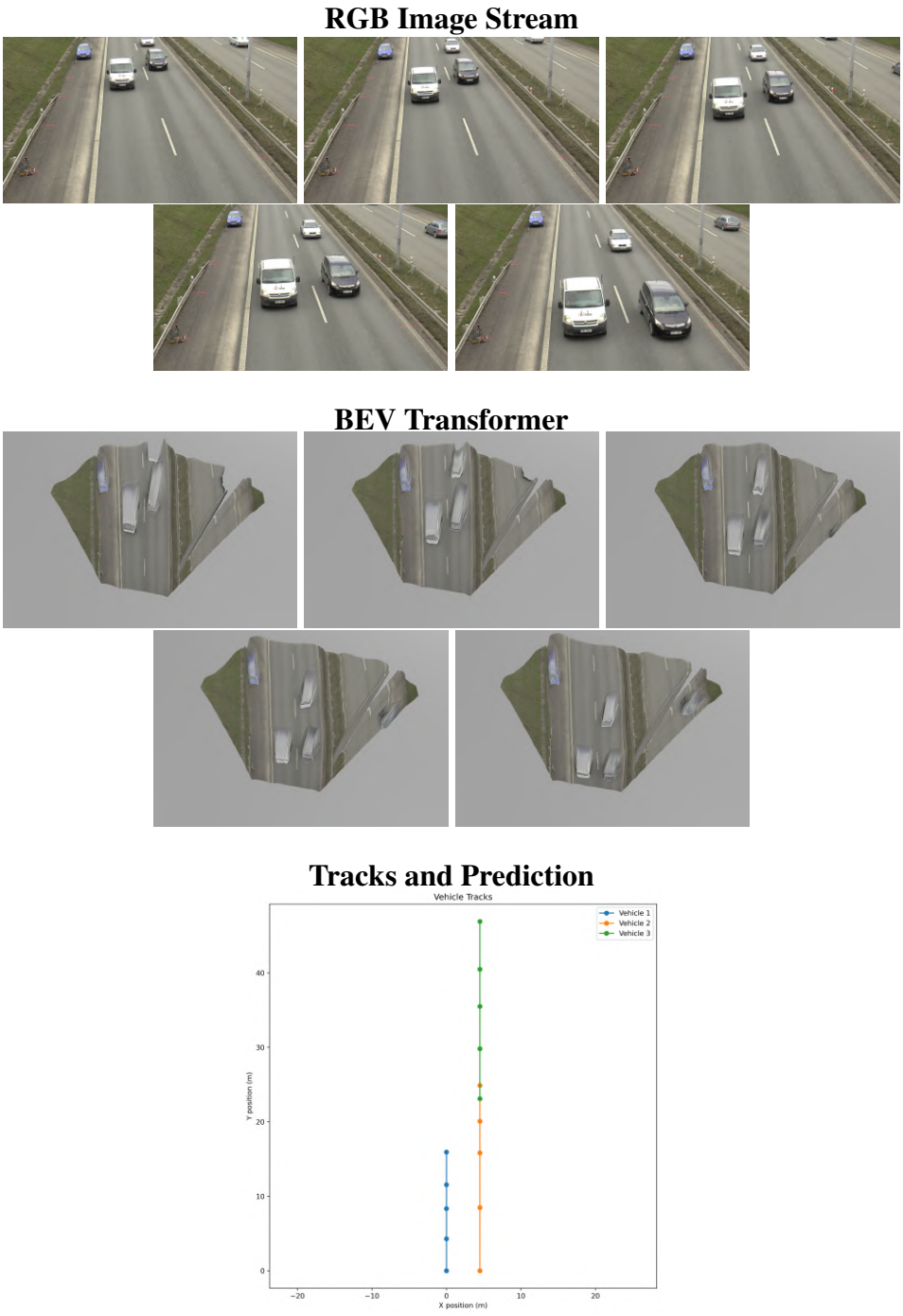
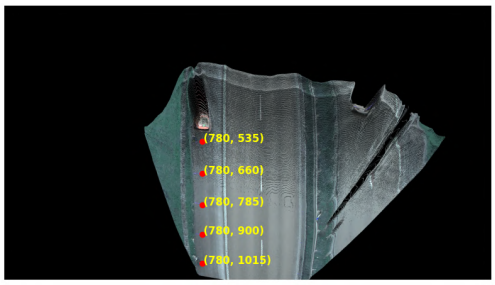


Figure 4.7: Visualization of RGB image stream (top row), BEV transformer output (middle row), and vehicle position predictions (bottom row).

BEV map ablation

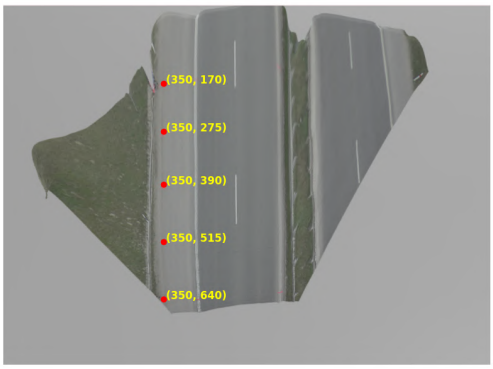
BEV Map Ablation Figure 4.8 illustrates the pixel values corresponding to points placed at equal physical intervals (marked with red line in original images) in the real world, as projected onto the BEV (Bird's Eye View) transformed image. As observed, the pixel distances between these points remain relatively consistent and stable up to approximately 20 meters. Beyond this range, however, the BEV projection begins to introduce noticeable noise and distortion, leading to reduced reliability in spatial representation. This indicates that the BEV transformation maintains geometric fidelity only within a limited distance from the origin.

(a) Monocular camera image



(b) BEV map with coordinates

(c) Monocular camera image



(d) BEV map with coordinates

Figure 4.8: Each scenario shows the monocular camera image (top) and its corresponding Bird's Eye View (BEV) representation with pixel coordinates (bottom) from different equidistant portions of the image. These visualizations help understand the robustness of the depth map estimation.

CHAPTER 5

Conclusion and Future Work

5.1 Conclusion

In this work, we propose a comprehensive approach to motion forecasting for autonomous driving scenarios, combining multiple novel components into a unified pipeline. First, we introduce a Transformer-based encoder architecture tailored for capturing temporal dependencies in agent motion. We also develop a multi-agent forecasting model that accounts for interactions among multiple dynamic entities in a scene. Additionally, we present a specialized agent-agent interaction module that models relational behavior explicitly.

Our approach leads to significant improvements over the baseline models across all key metrics, including minimum Final Displacement Error (minFDE), minimum Average Displacement Error (minADE), and Miss Rate. This demonstrates the effectiveness of our design choices in accurately capturing the complexity of agent motion and interaction in real-world driving environments.

Moreover, we extend the scope of motion forecasting by proposing a monocular camera-based solution. This approach leverages monocular depth estimation to infer spatial scene geometry, enabling state estimation and motion forecasting without relying on LiDAR or HD maps. Our method offers a scalable solution for scenarios where access to dense map priors or expensive sensors is limited.

5.2 Future Work

While our current approach effectively models the past and present interactions among agents to predict their future trajectories, it treats these interactions as static during prediction. As part of future work, we aim to extend our model to reason about **future interactions** dynamically.

Specifically, we plan to generate initial future trajectories for all agents and use these predicted paths to refine each agent's trajectory by considering how they will interact with one another in the future. This iterative refinement based on **anticipated future influence** will allow our system to better capture complex social behaviors, such as merging, yielding, or overtaking, thereby producing more realistic and coherent trajectory forecasts. This direction opens up a promising avenue for socially-aware and contextually adaptive motion prediction.

REFERENCES

[1] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, "Argoverse: 3D Tracking and Forecasting with Rich Maps," Nov. 2019, arXiv:1911.02620 [cs]. [Online]. Available: http://arxiv.org/abs/1911.02620

[2] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, D. Ramanan, P. Carr, and J. Hays, "Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting," Jan. 2023, arXiv:2301.00493 [cs]. [Online]. Available: http://arxiv.org/abs/2301.00493

[3] 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," May 2020, arXiv:1903.11027 [cs]. [Online]. Available: http://arxiv.org/abs/1903.11027

[4] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric, "Multimodal Trajectory Predictions for Autonomous Driving using Deep Convolutional Networks," Mar. 2019, arXiv:1809.10732 [cs]. [Online]. Available: http://arxiv.org/abs/1809.10732

[5] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun, "Learning lane graph representations for motion forecasting," in *ECCV*, 2020.

[6] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "Vectornet: Encoding hd maps and agent dynamics from vectorized representation," 2020. [Online]. Available: https://arxiv.org/abs/2005.04259

[7] Z. Zhou, L. Ye, J. Wang, K. Wu, and K. Lu, "Hivt: Hierarchical vector transformer for multi-agent motion prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[8] N. Deo, E. M. Wolff, and O. Beijbom, "Multimodal Trajectory Prediction Conditioned on Lane-Graph Traversals," Sep. 2021, arXiv:2106.15004 [cs]. [Online]. Available: http://arxiv.org/abs/2106.15004

[9] J. Hayakawa and B. Dariush, "Ego-motion and surrounding vehicle state estimation using a monocular camera," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, Jun. 2019, p. 2550–2556. [Online]. Available: http://dx.doi.org/10.1109/IVS.2019.8814037

[10] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," 2018. [Online]. Available: https://arxiv.org/abs/1803.10892

[11] Z. Zhou, J. Wang, Y.-H. Li, and Y.-K. Huang, "Query-centric trajectory prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[12] M. Hu, W. Yin, C. Zhang, Z. Cai, X. Long, H. Chen, K. Wang, G. Yu, C. Shen, and S. Shen, "Metric3d v2: A versatile monocular geometric foundation model for zero-shot metric depth and surface normal estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[13] S. F. Bhat, R. Birkl, D. Wofk, P. Wonka, and M. Müller, "Zoedepth: Zero-shot transfer by combining relative and metric depth," 2023. [Online]. Available: https://arxiv.org/abs/2302.12288

[14] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 03 1960. [Online]. Available: https://doi.org/10.1115/1.3662552

[15] A. Cosgun, L. Ma, J. Chiu, J. Huang, M. Demir, A. M. Añon, T. Lian, H. Tafish, and S. Al-Stouhi, "Towards full automated drive in urban environments: A demonstration in gomentum station, california," *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1811–1818, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:6194432

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023. [Online]. Available: https://arxiv.org/abs/1706.03762

[17] Z. Lan, Y. Jiang, Y. Mu, C. Chen, and S. E. Li, "Sept: Towards efficient scene representation learning for motion prediction," 2023. [Online]. Available: https://arxiv.org/abs/2309.15289

[18] Y. Liang, K. Ouyang, Y. Wang, X. Liu, H. Chen, J. Zhang, Y. Zheng, and R. Zimmermann, "Trajformer: Efficient trajectory classification with transformers," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, ser. CIKM '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1229–1237. [Online]. Available: https://doi.org/10.1145/3511808.3557481

[19] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, E. Kaus, R. G. Herrtwich, C. Rabe, D. Pfeiffer, F. Lindner, F. Stein, F. Erbs, M. Enzweiler, C. Knöppel, J. Hipp, M. Haueis, M. Trepte, C. Brenk, A. Tamke, M. Ghanaat, M. Braun, A. Joos, H. Fritz, H. Mock, M. Hein, and E. Zeeb, "Making bertha drive—an autonomous journey on a historic route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, 2014.

[20] X. Tang, M. Kan, S. Shan, Z. Ji, J. Bai, and X. Chen, "Hpnet: Dynamic trajectory forecasting with historical prediction attention," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 15 261–15 270.

[21] Z. Sun, Z. Wang, L. Halilaj, and J. Luettin, "Semanticformer: Holistic and semantic traffic scene representation for trajectory prediction using knowledge graphs," 2024. [Online]. Available: https://arxiv.org/abs/2404.19379

[22] J. Cheng, X. Mei, and M. Liu, "Forecast-MAE: Self-supervised pre-training for motion forecasting with masked autoencoders," *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[23] J. Sochor, R. Juranek, J. Spanhel, L. Marsik, A. Siroky, A. Herout, and P. Zemcik, “Comprehensive data set for automatic single camera visual speed measurement,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 5, p. 1633–1643, May 2019. [Online]. Available: http://dx.doi.org/10.1109/TITS.2018.2825609

LIST OF PAPERS BASED ON THESIS

1. Authors.... Title... *Journal*, Volume, Page, (year).