Decision making in self-driving cars using Graph Neural Networks

Jilan Samiuddin

Department of Electrical and Computer Engineering

McGill University, Montreal

A thesis submitted to McGill University

in partial fulfillment of the requirements of the degree of

Doctor of Philosophy

December 2024

Abstract

The autonomous driving industry's rapid growth highlights the necessity for advanced technologies to guarantee safety, comfort, and efficiency. This thesis focuses on three fundamental aspects of autonomous driving systems: trajectory prediction, trajectory planning, and control adaptation. The first contribution of this study is the introduction of a new technique for trajectory prediction that utilizes spatial-temporal graphs to capture historical traffic interactions. The use of a depthwise graph encoder network and sequential Gated Recurrent Unit decoder improves vehicle trajectory prediction compared to other deep learning methods. Next, an innovative online graph planner is introduced for generating feasible and comfortable trajectories. The planner creates a spatial-temporal graph that integrates the autonomous vehicle, nearby vehicles, and virtual road nodes. The graph is then processed using a sequential network with a behavioral layer for kinematic constraint compliance. Testing the planner on complex driving tasks demonstrates its effectiveness, surpassing existing state-of-the-art approaches. Finally, a novel approach for online learning in vehicle modeling and lateral control is introduced, using heterogeneous graphs and Graph Neural Networks. This technique enables the vehicle model and lateral controller to adapt to dynamic conditions, enhancing performance under perturbations. The self-learning model-based lateral controller is evaluated on the CARLA simulator, showing promising results. These contributions improve trajectory prediction, planning, and control adaptability, advancing autonomous driving technology and enhancing safety and efficiency of autonomous vehicles.

Sommaire

La croissance rapide du secteur de la conduite autonome met en évidence la nécessité de technologies avancées pour garantir la sécurité, le confort et l'efficacité. Cette thèse se concentre sur trois aspects fondamentaux des systèmes de conduite autonome : la prédiction de trajectoire, la planification de trajectoire et l'adaptation des commandes. La première contribution de cette étude est l'introduction d'une nouvelle technique de prédiction de trajectoire qui utilise des graphes spatio-temporels pour prendre en compte les interactions passées du trafic. L'utilisation d'un réseau d'encodeurs de graphes profond et d'un décodeur d'unités récurrentes à portes séquentielles améliore la prédiction de la trajectoire du véhicule par rapport à d'autres méthodes d'apprentissage profond. Ensuite, un planificateur en ligne innovant est introduit pour générer des trajectoires réalisables et confortables. Le planificateur crée un graphe spatio-temporel qui intègre le véhicule autonome, les véhicules à proximité et des nœuds routiers virtuels. Le graphe est ensuite traité à l'aide d'un réseau séquentiel avec une couche comportementale pour le respect des contraintes cinématiques. Le test du planificateur sur des tâches de conduite complexes démontre son efficacité, dépassant les approches traditionnelles. Enfin, une nouvelle approche d'apprentissage en ligne dans la modélisation de véhicules et le contrôle latéral est introduite, utilisant des graphes hétérogènes et des réseaux de neurones en graphes. Cette technique permet au modèle de véhicule et au contrôleur latéral de s'adapter aux conditions dynamiques, améliorant ainsi les performances en cas d'incertitude. Le contrôleur latéral basé sur un modèle d'auto-apprentissage est évalué sur le simulateur CARLA, montrant des résultats prometteurs. Ces contributions améliorent la prédiction de trajectoire, la planification et l'adaptabilité du contrôle, faisant progresser la technologie de conduite autonome et améliorant la sécurité et l'efficacité des véhicules autonomes.

Acknowledgements

I begin by expressing my utmost gratitude and thanks to The Almighty Allah for blessing me with the opportunity and empowering me to advance successfully in my journey of life. Also, I want to extend my sincere appreciation to all those who stood by me and provided guidance during my journey. I apologize in advance for any oversight.

I want to sincerely thank Prof. Benoit Boulet, my supervisor, for his remarkable patience, unwavering guidance, constant support, and invaluable advice throughout my Ph.D. journey. Above all, he built a work atmosphere that is both conducive and friendly, something that I view as essential for my personal success. I am deeply grateful to Dr. Di Wu for his consistent support and valuable feedback during the course of this research. Furthermore, I want to acknowledge and thank Quebec's Fonds de recherche Nature et technologies for its crucial financial support during my Ph.D.

My family, especially my beloved wife, Atoshi, has been an invaluable source of support and care, without whom the pursuit of a Ph.D. would have been exceedingly challenging. The immense love and support from my family has continuously motivated me throughout my thesis journey. Whenever I returned home from work, the presence of my two children, Umar and Aasiyah, never failed to provide me with a sense of relief from stress. I also want to acknowledge and appreciate my mother for always having faith in me and being there to support me in all my endeavors. Lastly, I want to extend my deepest appreciation to all my friends and colleagues who have always been ready to lend a helping hand whenever I required their support.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

Abbreviation	Definition	Abbreviation	Definition
SDG	Sustainable Development Goals	RHC	Receding Horizon Control
AC	Autonomous car	MLP	Multilayer perceptron
AI	Artificial intelligence	ADE	Average displacement error
MoD	Mobility on demand	FDE	Final displacement error
ADS	Automated driving system	RMSE	Root mean square error
GNN	Graph neural network	PF	Potential function
GLC	Graph-based lateral controller	STG	Spatial-Temporal Graphs
AiGem	Agent-Interaction Graph Embedding	MFP	Frenet path planner from MATLAB
GCN	Graph convolutional network	G2O	Graph-to-output
GAT	Graph attention network	CW	Center of wheelbase
GRU	Gated Recurrent Unit	GVM	Graph vehicle model
LSTM	Long Short-Term Memory	MLE	Maximum lateral error
MPC	Model Predictive Control		

Nomenclature

Symbol/Variable		Symbol/Variable	
s, d	Longitudinal, lateral positions	N_A	Number of actors
x, y	Cartesian coordinates	W, B	Weights, biases
θ	Heading of vehicle	α	Attention coefficient in GAT network
$a, \ddot{s}, \ddot{d}$	Acceleration and/or deceleration	$\mathcal{N}$	Neighborhood of a node
$v, \dot{s}, \dot{d}$	Velocity	E	Edge connection array of a graph
J	Jerk	R	Reference path coordinates array
δ	Steering command		
T	Throttle command	**Subscript/Supercript**	
X	States of the ego	acc, dec	Acceleration , Deceleration
y/Y	Output	rec, safe	Recommended, safety
N_V	Number of virtual nodes	upper, lower	Upper bound, lower bound
V_s, V_d	Longitudinal, lateral virtual nodes	long, lat	Longitudinal, Lateral
U_o, U_v	Obstacle potential, velocity potential	e, a	Ego, actor
$\mathcal{G}, G$	Graph	r, l	Right and left side of ego
h, Z	Feature or node embedding	w	Wheel of ego
p, q	Target node, neighboring node	r, f	Rear end and front end of ego
t_s	Sampling period	m	Vehicle model
N	Planning Horizon	c	Vehicle controller

Chapter 1

Introduction

Vehicle driving has been a common but somewhat demanding task humans have been doing for more than a century. Since the advent of computers and control technology, it has been possible for systems to assist human drivers up to the point of automating entirely the driving task. The levels of automation in the driving task were defined by the Society of Automotive Engineers on a scale of 0 to 5 where Level 0 is unassisted human driving and Level 5 is fully autonomous driving unsupervised by a human. The forefront of revolutionizing the transportation industry lies in autonomous driving technology, aiming to achieve Level 5 autonomy. The promising level of autonomy being introduced is expected to improve road safety, enhance traffic efficiency, and provide mobility solutions for diverse populations, including individuals who are incapable of driving. However, the pursuit of Level 5 autonomy is burdened with complex challenges, particularly in the domains of perception, decision-making, and control. The purpose of this thesis is to tackle some of these challenges by devising advanced algorithms and models that allow vehicles to autonomously navigate complex environments, thereby making a significant contribution to the overall objective of fully autonomous driving.

1.1 Motivation

The hype around self-driving cars has been growing over the past years and has sparked much research. This is primarily because of the need and demand for safe and eco-driving which is a segment of the many Sustainable Development Goals (SDG) set by the United Nations [2]. The integration of sustainable transportation methods can contribute to both economic growth and increased accessibility. By promoting sustainable transportation, we can achieve greater economic integration, environmental conservation, social equity, stronger urban-rural connections, and increased productivity in rural regions [3]. Autonomous cars (ACs) have significant potential to tackle these issues because of the potential to achieve eco-driving, in contrast to human-driven vehicles [4, 5], through the advancement of artificial intelligence (AI). Hence, it has become inevitable for tech companies to secure a place in the race to integrate autonomy in the transportation system.

The sole contribution of the global transport system amounts to around one-quarter of energy-related global greenhouse gas emissions [6]. Driving pattern strongly influences emissions [7, 8]. The driving pattern is commonly determined by the speed profile [7], which has an impact on emissions. Nevertheless, frequent braking or constant shifts in speed result in elevated energy consumption [7, 8]. The selection of the route plays a role in the emissions as well [7]. Traffic congestion is another aspect that contributes to pollution [9, 10].

Moreover, Mobility on Demand (MoD) is transforming the future transportation services due to an increase in urban population. At its core, MoD is based on the principle of treating transportation as a commodity, where different modes are assigned specific economic values [11]. These values include a range of factors, such as cost, travel time, waiting time, number of connections, convenience, and other pertinent attributes. With autonomous MoD systems, the current taxi demand in New York City can be accomplished by using ACs numbering at approximately 70% of the current taxi fleet [12]. In Singapore, the entire population's personal mobility can be replaced by one-third autonomous vehicles of the current number of passenger vehicles [12]. ACs can significantly improve mobility and accessibility for people with disabilities [13], unable to drive

traditional vehicles, making them more independent and confident in their daily activities. The United Nations, therefore, suggests to allocate funding for development of assistive technologies to improve the quality of life and autonomy of persons with disabilities [14].

There are several factors restricting the standardization of a Level 5 AC to fit the industrially realistic prototype. When human drivers make decision to navigate through roadways, they try to ensure safety without violating traffic laws. ACs deployed on roadways will also face similar situations to humans, and thus, must demonstrate similar navigational skills like humans, if not better. However, a huge concern is the ability of ACs to achieve the desired goals safely without violating traffic laws and comfortably in situations characterized by uncertainty. General road-users will always be skeptical of accepting ACs on roads unless manufacturing companies can effectively demonstrate that ACs are as safe at least as a average human driver [15].

1.2 Problem Statement

The potential impact of autonomous driving technology on transportation is immense, as it can bring about significant advantages such as reduced traffic accidents, enhanced mobility for various populations, and increased efficiency in transportation systems. Nevertheless, the ability of ACs to perform well depends on the robustness and reliability of their AI systems. The presence of a robust AI system is crucial in effectively managing a vast range of unpredictable real-world scenarios, enabling autonomous vehicles to safely navigate intricate environments and promptly respond to dynamic changes. Moreover, the inclusion of online learning capabilities is of utmost importance for ACs to enhance their performance by effectively responding in real-time to emerging situations and new data. The continuous learning process improves the system's capacity to address new challenges and maintain high safety standards. Moreover, it is of great importance to prioritize responsible AI practices to ensure that self-driving vehicles comply with road regulations and provide passengers with a comfortable and enjoyable ride. Thorough attention to these aspects is critical for the establishment of public confidence, the attainment of regulatory endorsement, and

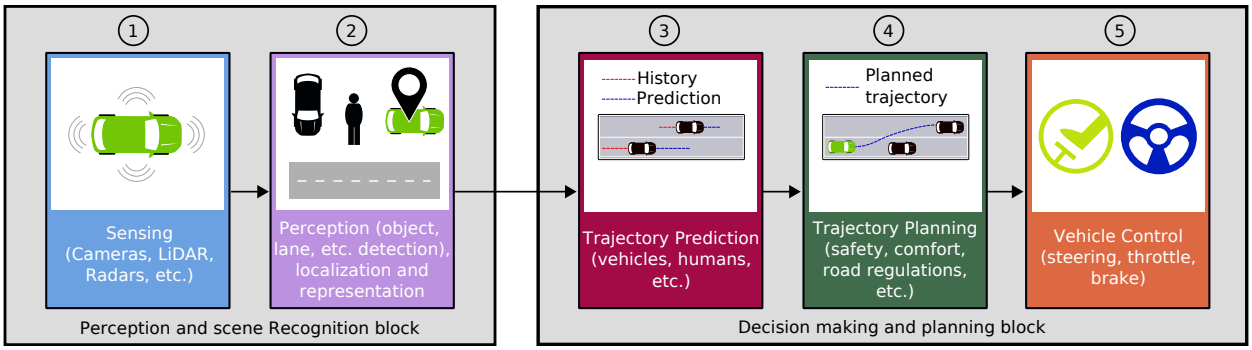


Figure 1.1: The high-level framework of the traditional autonomous driving system

the facilitation of the widespread integration of autonomous driving technology.

Despite significant advancements in autonomous driving technology, there are several crucial gaps that have yet to be addressed. Presently, artificial intelligence systems frequently show a deficiency in their ability to effectively navigate a wide range of real-life driving situations, especially those that involve rare or unforeseen events. The limitations of many existing models stem from their reliance on static datasets, which impedes their ability to adapt to new conditions or learn from ongoing experiences. This inadequacy highlights the need for improved integration of online learning methods that support ongoing improvement. In addition, there is an urgent necessity to construct frameworks that ensure the responsible operation of ACs, which includes adherence to road regulations, ethical decision-making, and passenger comfort. Most existing research has centred on isolated elements of autonomous driving, such as perception or control, with inadequate attention given to the holistic incorporation of robustness, online learning, and responsibility. The aim of this research is to address these disparities through the development and evaluation of comprehensive AI systems that integrate these essential aspects, thus advancing the state-of-the-art in autonomous driving technology. Broadly speaking, the traditional automated driving system (ADS) has two main components: (1) the perception and scene recognition block, and (2) the decision making and planning block, as shown in Figure 1.1. The former is responsible for sensing the environment and providing an accurate representation of the scene [16]. The task of the latter is to plan future motion and generate control commands [17]. In this thesis, the focus is on the decision making and planning block, more particularly, the following sub-modules: (i) the trajectory pre-

diction module, (ii) the trajectory planning module, and (iii) the vehicle control module (subblocks 3, 4 and 5 in Figure 1.1). This research aimed to develop AI techniques for autonomous driving with the following goals:

1. **Robust and learning AI** – Robust and learning AI equips ACs with the capability to smoothly maneuver through complex environments, efficiently respond to dynamic changes, and prioritize safety in decision-making. Robustness is vital in autonomous driving for effectively handling unpredictable real-world situations, enhancing safety and reliability. The integration of online learning allows AI to continuously update and improve its models based on new data, enhancing its ability to adapt to new scenarios and challenges in real-time.

2. **Responsible AI** – Ensuring the inclusion of responsible AI in autonomous driving is vital to guarantee that vehicles operate in accordance with legal and ethical boundaries, demonstrate respect for road regulations, and prioritize passenger comfort and safety. It involves developing systems that adhere to traffic laws and make decisions in all situations. Furthermore, responsible AI places emphasis on delivering a seamless and comfortable travel experience through the consideration of factors such as speed limits, road conditions, and passenger preferences. Responsible AI ensures trust and seamless integration of autonomous vehicles by prioritizing safety, legality, and comfort.

It is important to acknowledge that there are modules that combine both blocks and utilize an end-to-end learning framework for the final decision-making [17, 18, 19]. An end-to-end learning framework refers to a system that directly maps raw sensor data to control outputs through a single learning process, without the need for task-specific modules. A comparative analysis of end-to-end and traditional modular approaches in autonomous driving reveals distinct advantages and challenges inherent to each technique. The traditional modular approach divides the driving pipeline into discrete, task-specific components such as perception, planning, and control. Thus, this design facilitates our interpretation, inspection and debugging of each module's output [18, 19]. The modular nature of these systems permits the integration of task-specific algorithms,

enhancing their reliability within controlled contexts [18]. Nevertheless, this method is susceptible to the propagation of errors, whereby inaccuracies in a single module, such as misclassifications in perception, can cascade in subsequent stages, including planning and control [18, 19]. In addition, modular systems often exhibit redundant calculations in the absence of a unified optimization goal, which can contribute to inefficiencies in both computational resources and system performance as a whole.

Alternatively, the end-to-end approach optimizes the system by directly mapping raw sensor data to control outputs through a single learning process, employing techniques such as imitation and reinforcement learning [18, 19]. This strategy minimizes error propagation by concurrently optimizing the entire system for the primary driving task, thus aligning all components towards a unified objective [19]. End-to-end systems contribute to simplified development by eliminating the need for task-specific module design and enabling direct learning from extensive datasets [18, 19]. The interpretability of these systems has been improved by recent advancements, which involve the integration of auxiliary outputs such as attention maps and cost functions. However, there is still a long road to cover for these advancements to be acceptable in practice. For example, in a very recent work [20], Araluce et al. used frontal camera images to make decisions for the self-driving car, and their results show that even though the decision results are better than the ones in the dataset, the explanation performance decreased from that obtained in the dataset. Atakishiyev, in his thesis [21] related to autonomous vehicles, used a Video-Language Transformer to determine if the model can provide correct explanation when prompted with a question from the user. However, in several instances, incorrect explanations were given, which could negatively impact user trust in a self-driving vehicle's explanations and decision-making, thereby limiting the suitability of these models as a trustworthy automotive user interface. Therefore, despite these advances in end-to-end models for interpretability, significant barriers to broader adoption remain, including a lack of transparency in decision-making, difficulties in debugging, and the computationally demanding nature of the extensive training required.

Datasets pose significant challenges to end-to-end autonomous driving systems due to limited

diversity, imbalances, and inadequate annotations. The tendency of such systems to overfit specific, non-diverse datasets limits their ability to generalize to conditions they have not encountered previously [19]. Because the model is trained on too many examples of common driving maneuvers like driving straight, it may struggle with uncommon but critical situations, leading to performance issues. Moreover, datasets typically provide action labels (e.g., "stop") but lack explanation-level annotations (e.g., "red light ahead"), essential for training explainable AI systems [22]. Many also omit finer-grained labels like object proximity data, limiting the model's ability to capture evolving dynamics in driving scenes [23]. To address these limitations, we require datasets that are diverse, balanced, and richly annotated, to improve the robustness and explainability of the model.

The modular approach continues to be favoured in specific contexts due to its transparency, ease of debugging, and robustness in scenarios requiring strict adherence to safety and regulatory standards [18, 19]. The modular design ensures that critical safety checks can be performed at each stage, and failures can be identified and addressed without compromising the entire system. This justifies using the modular approach in environments where reliability, explainability, and compliance are paramount, such as urban driving or public transportation. The modular approach benefits from reduced dependency on large, diverse datasets, as each module can be trained independently using task-specific data. This flexibility allows leveraging smaller, specialized datasets tailored to individual components, simplifying data collection and improving robustness in specific tasks. Even though the end-to-end approach shows promise for future scalability and efficiency, the functionality of such a model is like a black box, and thus lacks explainability because of its complex architecture [24]. This presents an issue and impedes the progress of ACs being socially accepted on the roads – the key factor driving technological advancements is social acceptance [25]. The modular approach remains indispensable for building trust and ensuring safety in the current landscape of autonomous driving [18, 19].

As the perception module is beyond the scope of this thesis, it is assumed that the perception module provides the vehicle states, road structure, and regulations. An example of a highly accurate perception module is the NVAutoNet [26], developed by NVIDIA. This module is designed

to be computationally efficient, compatible with onboard chips, and adaptable to a wide range of vehicle types.

Sensor noise is an issue that leads to increased uncertainty. However, it is common to introduce safety margins to help counteract errors from sensor noise and enhance the system reliability. For example, Duan et al. improved the robustness of automatic emergency braking systems by adjusting safety distances [27]. Though sensor noise was not a factor in experimentation conducted in this thesis, a safety gap is incorporated in the trajectory planner module as a proactive measure to mitigate potential noise-related risks.

1.3 Methodology

Driving scenarios indicate graph-like structures, as seen in Figure 1.2 [1]. Graphs have been proven to be effective in a multitude of applications, e.g., social networks [28], medicine [29], etc. The graph neural network (GNN) [30] is an excellent solution for ACs because of its effectiveness in capturing complex relational data, which plays a crucial role in solving driving problems. Classical neural networks, limited by their reliance on fixed-size feature spaces, encounter substantial difficulty in handling the inherent complexity and variability present in road geometries and traffic scenarios [31]. GNNs enable the integration of heterogeneous data sources, including sensor data and map information, thereby improving decision-making for autonomous vehicles [31, 32]. This improves the system's ability to manage varied driving situations and increases its overall reliability. Also, the graph-based architecture of GNNs enables scalable and adaptable representations of traffic scenes, accommodating varying number of agents and diverse environmental factors [33]. Scalability and adaptability are critical to the effective implementation of autonomous systems across different settings. Furthermore, intuition plays a significant role in strategic decision making [34] which is an indispensable trait for driving. The prediction, planning and control problems are formulated by combining intuition with spatial-temporal (space-time) graphs and solved using GNNs.

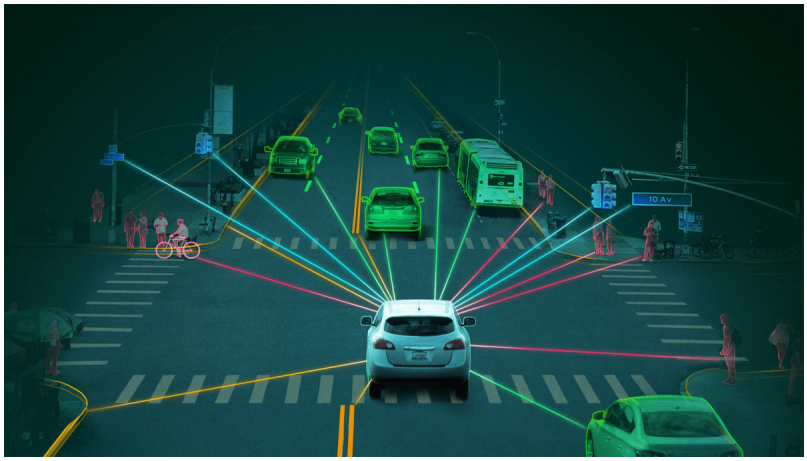


Figure 1.2: A graph-like structure depiction of vehicles on road (figure used under a Creative Commons 4.0 license) [1]

1.3.1 Trajectory Prediction

This module estimates the future positions of all the objects around the autonomous car in the current timeframe based on their histories [35]. However, in real-time, it can pose significant challenges given the dynamic and uncertain nature of roadways and the objects themselves. For the thesis, we propose a trajectory prediction model that uses a graph depicting the interactions between vehicles. While embedding interactions into graphs is common in the literature, we have also integrated the temporal (time) aspect into the graph that makes it distinct from other works.

1.3.2 Trajectory Planner

This module plans a future feasible trajectory of the AC given information on its surroundings [36]. The future trajectory must be feasible, safe, and respect constraints. In this work, a novel online trajectory planner is developed that projects the trajectory planning problem as a graph comprising road and kinematic constraints and processes it through a graph neural network architecture to find the desired trajectory. The novelty of the work lies in the intuitive approach to integrate road regulations, vehicle constraints and comfort aspect into the graph.

1.3.3 Vehicle Control

This module generates the steering command (lateral control) and the throttle command (longitudinal control) for the AC to follow the desired trajectory [37]. For this thesis, a novel online graph-based lateral controller (GLC) is developed for ACs – the first of its kind. The core novelty, besides the network architecture, is in the formulation of the graph – an intuitive representation of a vehicle transitioning from its current state to its future desired state.

1.4 Thesis Structure

Chapter 2 presents an exposition of the fundamental theoretical concepts that serve as the framework for this thesis. In this chapter, concepts such as the GNN are discussed, to provide a solid theoretical framework that is crucial for understanding the subsequent, more specific applications in the later chapters. In Chapter 3, the first problem under scrutiny, specifically trajectory prediction, is addressed. This chapter focuses on the methods and algorithms used to predict the future paths of surrounding vehicles. Chapter 4 focuses on an online trajectory planner, specifically discussing the process of formulating it. Chapter 5 is dedicated to the development of a lateral controller utilizing graph neural networks. This chapter examines the novel implementation of graph neural networks in the online lateral control of ACs, ensuring robustness in the face of disturbances. Finally, Chapter 6 concludes the thesis.

1.5 Claims of originality

Chapter 3 introduces a novel approach called AiGem (Agent-Interaction Graph Embedding) that formulates historical traffic interactions as spatial-temporal graphs to predict trajectories around autonomous vehicles. Additionally, AiGem combines a graph encoder network with a Gated Recurrent Unit decoder and Multilayer Perceptron output network. The technique has received acceptance for publication in [38].

Chapter 4 brings originality by introducing a new online spatial-temporal graph trajectory planner. It generates safe and comfortable trajectories for autonomous vehicles by considering surrounding vehicles, road constraints, and kinematic constraints within a comprehensive graph structure. Moreover, this approach distinguishes itself by incorporating a behavioral layer to determine kinematic constraints and by developing a unique potential function for training the network. The publication of this work can be found in [39]. A provisional patent application for the planner has been filed with the US Patent and Trademark Office.

The uniqueness of Chapter 5 is highlighted by the creation of an innovative online learning approach for vehicle modeling and lateral control, employing heterogeneous graphs and Graph Neural Networks. Compared to traditional controllers, this innovative method enhances performance by adapting dynamically to changing conditions and environments. Currently, the work is undergoing review for publication [40].

1.6 Contributions of authors

Three papers [38, 39, 40] provide support for this thesis, with Mr. Jilan Samiuddin listed as the first author, and Prof. Benoit Boulet and Dr. Di Wu mentioned as the second and last authors, respectively. Mr. Samiuddin deserves credit for the research planning and execution in this thesis, while Prof. Boulet and Dr. Wu provided guidance and suggestions to enhance the research work.

Chapter 2

Theoretical Background

In this chapter, theoretical concepts that have been used in developing and formulating the works for this thesis are introduced. The graph neural network (GNN) and some of its variants are at the core. Thus, theories of GNNs are presented in details in this chapter. Furthermore, in the thesis, the Frenet coordinate frame is used in several instances replacing the traditional global coordinate frame for the formulation of the problem. Model predictive control is also briefly reviewed.

2.1 Graph Neural Network

Neural networks that operate on graphs are called graph neural networks [30].

2.1.1 Graphs

Graphs are used as means of illustrating real-world problems. More particularly, graphs can capture the interactions between agents and how these agents evolve because of their involvement in a neighborhood. The agents in a graph are called the nodes and the interactions between them are presented using edges and this is shown in Figure 2.1. Formally, a graph $G = (V, E)$ is a collection of node features V and edge connections E in which the edges may or may not have features. These features are often known physical or meaningful attributes of the nodes and edges. The edges can

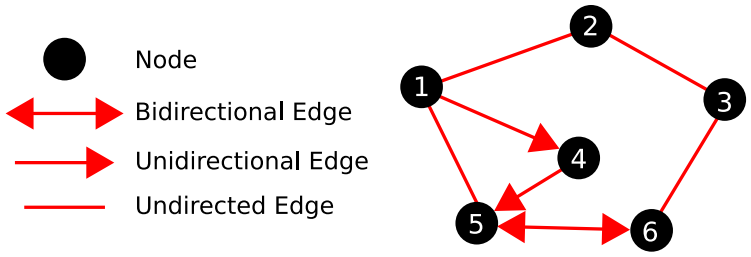


Figure 2.1: A graph containing six nodes (in black) with edges (in red) showing the connections between the nodes

be of three types:

1. *Undirected edge*: This defines a connection between two nodes whose relationship is mutual, i.e., the edge can traverse in both directions equally. For example, in a social network graph, an undirected edge may represent a mutual friendship between two people (the two nodes).

2. *Unidirectional edge*: This defines a one-way relationship from one node to the other, i.e., the edge can traverse in the direction defined. For example, in a web graph, an unidirectional edge may represent a hyperlink from one webpage to another (the two nodes).

3. *Bidirectional edge*: This is similar to an undirected edge defining a two-way relationship between two nodes with the direction of the edges explicitly defined, i.e., edge traversal is possible in both the directions. For example, in a transportation network graph, a bidirectional edge may represent a two-way street between two locations (the two nodes).

2.1.2 Graph Classifications

Based on the types or labels associated with nodes and/or edges which indicate their different roles or semantics, graphs can be broadly divided into two categories:

1. *Homogeneous graphs*: The nodes are of same type and the edges are of same type in a homogeneous graph. This implies that the feature array of every single node and edge have the same physical interpretation. For example, in a simple social network graph, all nodes represent individuals and all edges represent friendship (connection) between them.

2. *Heterogeneous graphs*: Either the nodes or the edges or both are of different types in a heterogeneous graph. Thus, the feature array will also have different physical interpretation. For example, in a publication network graph, nodes can represent the authors and the papers, while the edges can represent relations such as “authored” and “cited”.

In this thesis, heterogeneous graphs are used to represent all the autonomous driving tasks.

Based on the graph structure over time, graphs can also be classified into two categories:

1. *Static graphs*: Over time, the graph structure does not change, i.e., the nodes and the edges do not change. The graph of a map is a simple example of a static graph in which the destinations (i.e., nodes) connected by roads (i.e., edges) do not change over time.

2. *Dynamic graphs*: The graphs structure changes over time representing the evolution of relations between the nodes. This can either be in the form of nodes and/or edges being added and/or removed. For example, in a complex social network graph, where nodes represent users and edges represent their interactions (e.g., friendships, follows, or messages), the structure changes continuously as the users join or leave the network and as they modify their interactions with other users.

Figure 2.2 shows examples of a static graph and a dynamic graph. Autonomous driving problems are generally dynamic in nature where the problem structure changes over time. However, in this thesis, both static and dynamic graphs have been used for problem representation. Particularly, Chapter 3 and Chapter 4 use dynamic graphs, and, Chapter 5 uses static graphs.

2.1.3 Applications of GNNs

A GNN can be used for several types of tasks. On a general level, the tasks can be classified into four categories as follows:

1. *Graph-level tasks*: This involves making predictions or inferences about graphs as a whole (e.g., classifying graphs into categories, etc.), and also extends to generating new graphs.

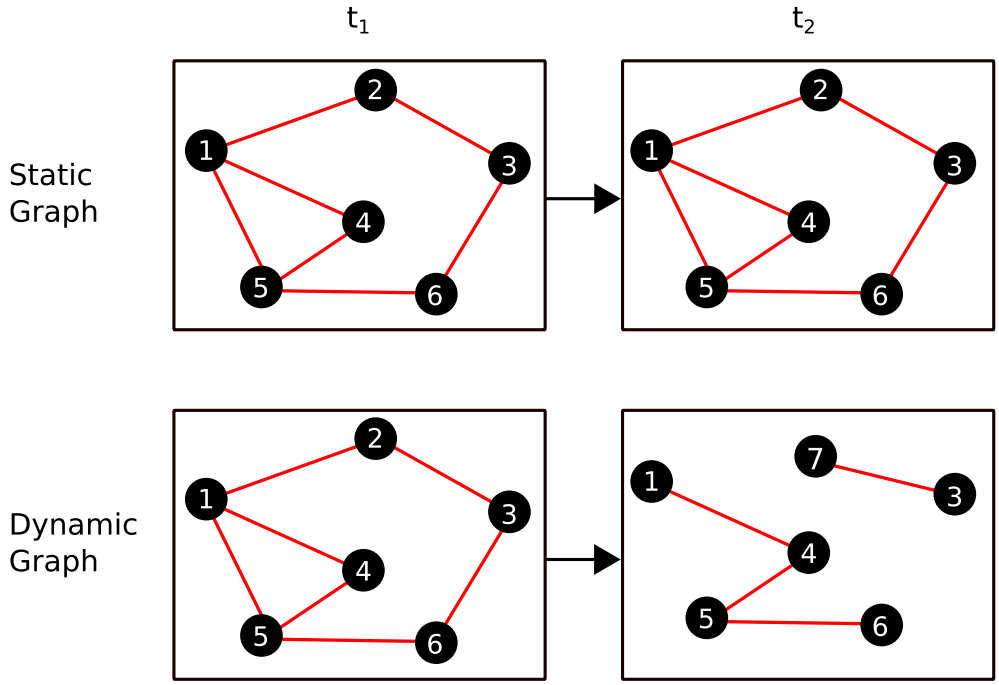


Figure 2.2: Examples of a static graph and a dynamic graph

For example, a GNN can be trained to predicting properties of molecules, represented by a graph.

2. *Subgraph-level tasks*: This involves making predictions or inferences about a specific subgraph within a larger graph. Identifying a community or a cluster within a social network is an example of such tasks.

3. *Node-level tasks*: This involves making predictions or inferences about individual nodes in a graph (e.g., node classification, node regression, etc.). For example, in a social network, users can be categorized based on their roles and attributes.

4. *Edge-level tasks*: This involves making predictions or inferences about edges between the nodes in a graph (e.g., link prediction, edge classification, etc.). Predicting future friendship between two users in a social network is an example of such tasks.

The tasks are illustrated in Figure 2.3. In this thesis, only node-level tasks have been performed using GNNs in all the works since the nature of the formulation of the problems demands so. Thus,

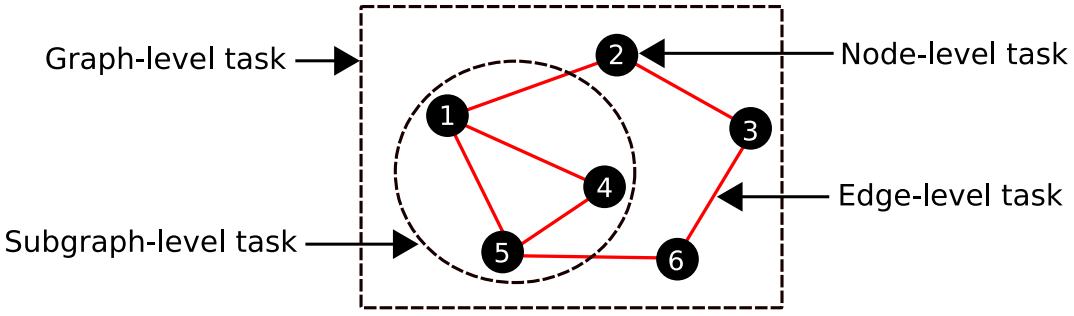


Figure 2.3: Different types of tasks of GNNs

in the rest of the chapter, the operation of a GNN will be explained for node-level inferences.

2.1.4 Graph Convolutional Network

A Graph Convolutional Network (GCN) is a GNN that extends the concept of a convolutional neural network (CNN). While a CNN can operate only on structured data, a GCN can operate on non-Euclidean structures like graphs. GCN is fundamental in understanding the basic functioning of a GNN. For the sake of simplicity, homogeneous graphs will be used for explanation.

For node representation, the GCN maps nodes of a homogeneous graph to a matrix $\mathbb{R}^{n\times m}$, where, n is the number of nodes and m is the output dimensionality of the features of the nodes. For the target node (i.e., the node of interest) p, the GCN aggregates information from its neighbors (e.g. averages the messages from its neighbors) and then applies a neural network in several layers [41]:

$$h_p^{(i+1)} = \sigma\left(W_i \underbrace{\sum_{q\in\mathcal{N}(p)} \frac{h_q^{(i)}}{|\mathcal{N}(p)|}}_{\text{message aggregation}} + B_i h_p^{(i)}\right), \tag{2.1}$$

$\forall i \in \{0,1,\ldots,L-1\}$, where, h is the embedding of a node, σ is a nonlinear activation function, W_i and B_i are the weights and biases of the i^{th} layer, respectively, $\mathcal{N}(p)$ is the neighborhood of the target node p, and L is the total number of layers. Equation (2.1) is illustrated in Figure 2.4. Note that the input embedding of a node when $i = 0$, is its original input features.

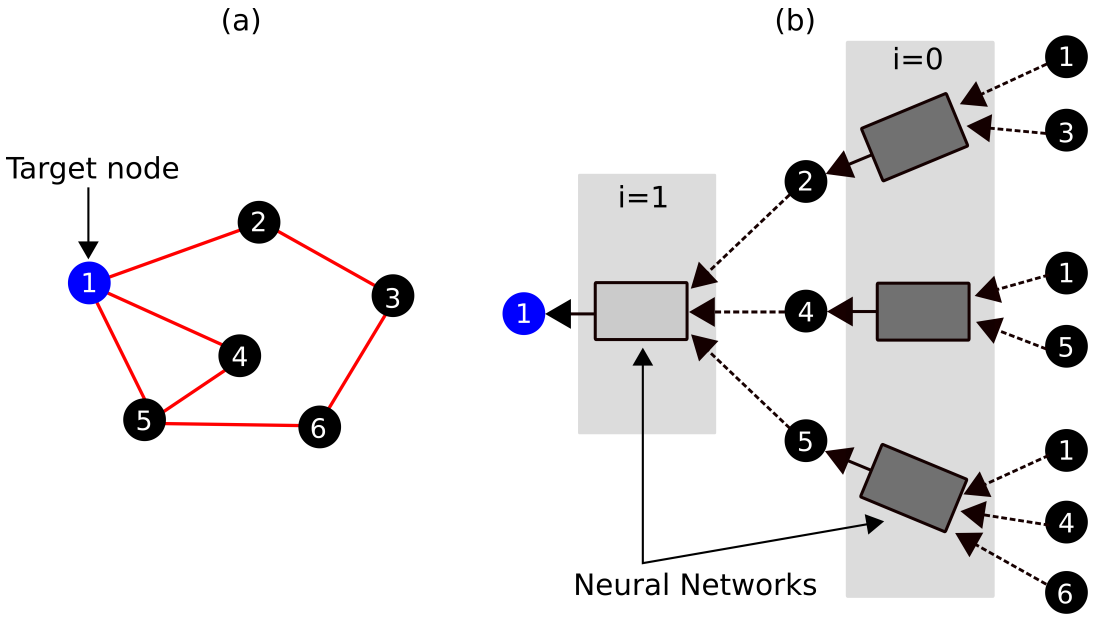


Figure 2.4: (a) The input graph with node 1 being the target node. (b) The target node aggregates information from its neighbors with $L = 2$

2.1.5 Graph Attention Network

Graph Attention Network (GAT) [34] is a GNN that integrates attention so that the learning is focused on more relevant segments of the input. The network learns the importance (also called the attention coefficient e_{pq}) of the neighbors of a node as it aggregates information from them. The attention coefficient between the target node p and its neighbor q is computed by applying a common linear transformation W to the features (h) of both p and q, followed by a shared attentional mechanism (att) as follows:

$$e_{pq} = \text{att}\left(Wh_p, Wh_q\right) \tag{2.2}$$

A single-layered feed-forward neural network is used for att [34] as shown in Figure 2.5. To compare the neighboring nodes properly, e_{pq} is normalized using the softmax function:

$$\alpha_{pq} = \frac{\exp\left(e_{pq}\right)}{\sum_{k \in \mathcal{N}(p)} \exp\left(e_{pk}\right)} \tag{2.3}$$

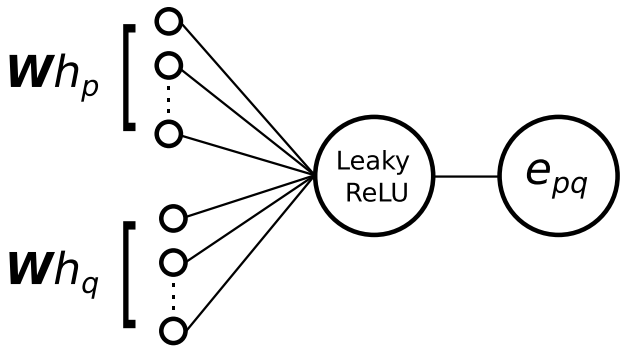


Figure 2.5: Attentional mechanism applied in a Graph Attention Network

Using the attention coefficients to weigh the importance of the neighboring nodes, the final output features of every node is obtained as follows:

$$\bar{h}_p = \sigma\left(\sum_{q\in\mathcal{N}(p)} \alpha_{pq} W h_q\right) \tag{2.4}$$

To ensure stability of the learning process of self-attention, [34] applies multi-head attention, i.e., K independent attention mechanisms of equation (2.4) are executed and then their features are either concatenated or averaged. If concatenation is applied, then the equation for the final output features of every node is as follows:

$$\bar{h}_p = \Big\|_{k=1}^{K} \sigma\left(\sum_{q\in\mathcal{N}(p)} \alpha_{pq}^k W^k h_q\right) \tag{2.5}$$

where, $||$ represents concatenation, α_{pq}^k are normalized attention coefficients computed by the k^{th} attention mechanism (att^k), and W^k is the corresponding input linear transformation's weight matrix. If averaging is applied, then the equation for the final output features of every node is as follows:

$$\bar{h}_p = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{q\in\mathcal{N}(p)} \alpha_{pq}^k W^k h_q\right) \tag{2.6}$$

As discussed earlier, because of the differences in type and dimensionality in a heterogeneous graph, a single node or edge feature tensor is unable to accommodate all the node or edge features

of the graph. PyTorch libraries are available to process heterogeneous graphs [42]. The computation of messages and update functions is conditioned on node or edge type. Thus, standard message-passing GNNs cannot be applied to heterogeneous graphs. PyTorch offers three different approaches to create models on heterogeneous graph data. For this thesis work, the automatic conversion approach of a homogeneous GNN to a heterogeneous GNN model was used, the details of which can be found in [42].

2.2 Gated Recurrent Unit

Gated Recurrent Unit (GRU) [43, 44] aims to solve the vanishing gradient problem typically found in a recurrent neural network (RNN). The GRU architecture shares similarities with the LSTM (Long Short-Term Memory [44, 45]) and is specifically designed to capture sequential data by facilitating selective retention and erasure of information as time progresses. Despite this, the GRU exhibits a simpler design in contrast to the LSTM, featuring a lower parameter count. This attribute simplifies the training process and enhances computational efficiency.

In terms of functionality, the key difference between a GRU and an LSTM lies in how they handle the memory cell state. The LSTM model incorporates a mechanism in which the memory cell state is maintained separately from the hidden state, and it is updated through the input gate, output gate, and forget gate. The GRU model employs a candidate activation vector $\hat{h}_t$ to replace the conventional memory cell state, and this vector is updated through the reset gate and the update gate. The update gate u_t and the reset gate r_t are responsible for determining the flow of information to the output.

To elaborate further, the reset gate is responsible for deciding how much of the previous hidden state should be ignored. The previous hidden state and the current input are used as input, which then produces a vector of numerical values between 0 and 1. These values serve to govern the level of "reset" applied to the previous hidden state at the current time step. The update gate determines how much of the candidate activation vector should be blended with the new hidden state. It

accepts the previous hidden state and the current input as its input and generates a vector of values ranging from 0 to 1. These values determine the extent to which the candidate activation vector is integrated into the new hidden state. Figure 2.6 shows the architecture of GRU that takes the current state x_t and the previous hidden state h_{t-1} as inputs. Following are the steps to obtain the current output state y_t and the hidden state h_t:

1. The reset gate is computed using x_t and h_{t-1}

$$r_t = \sigma\left(W_{rx}x_t + W_{rh}h_{t-1}\right), \tag{2.7}$$

2. The update gate is also computed using x_t and h_{t-1}

$$u_t = \sigma\left(W_{ux}x_t + W_{uh}h_{t-1}\right), \tag{2.8}$$

3. By utilizing the current input x_t and a modified version of the previous hidden state which is reset by the reset gate, the candidate activation vector is computed

$$\hat{h}_t = \tanh\left(W_{\hat{h}x}x_t + W_{\hat{h}h}\left(r_t * h_{t-1}\right)\right), \tag{2.9}$$

4. The update gate determines the weighting of the candidate activation vector and the previous hidden state, which are then combined to compute the new hidden state

$$h_t = u_t * h_{t-1} + (1 - u_t) * \hat{h}_t, \tag{2.10}$$

5. Finally, the output of GRU is calculated

$$y_t = \sigma\left(W_y h_t\right), \tag{2.11}$$

where, W_{rx}, W_{rh}, W_{ux}, W_{uh}, $W_{\hat{h}x}$, $W_{\hat{h}h}$, W_y are weights, and the operator $*$ is the Hadamard product.

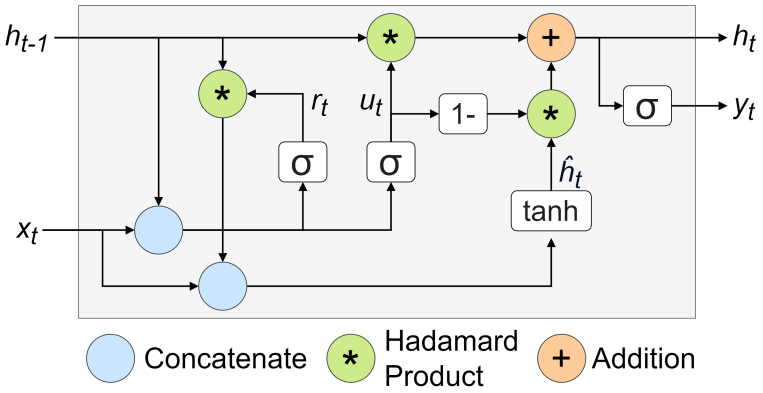
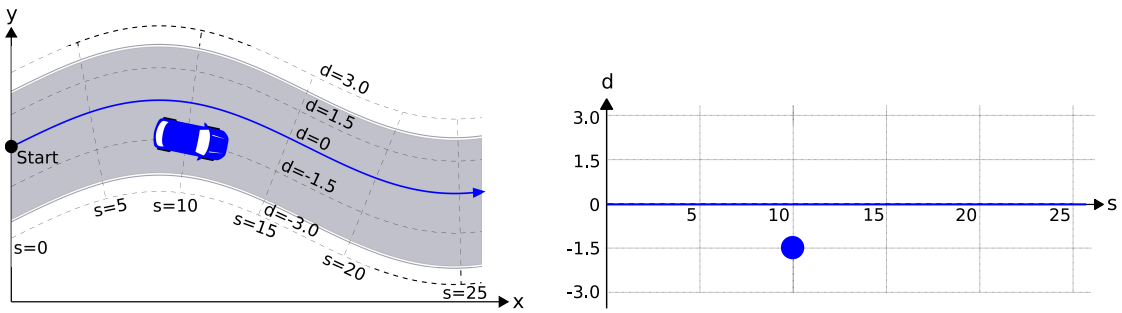


Figure 2.6: GRU architecture

For further information on the GRU and its functional distinctions from an LSTM, readers are encouraged to refer to [44].

2.3 Frenet Coordinates

Frenet coordinate system [46] is used to describe the location of a point relative to a reference curve. More particularly, the Frenet coordinates s and d represent the longitudinal and the lateral distances of the point of interest with respect to the curve from a starting point. Figure 2.7a shows the Frenet coordinates inside a Cartesian coordinate frame with respect to a reference curve.



(a) (b)

Figure 2.7: (a) A snapshot of road presented in the Cartesian coordinate frame showing the Frenet framework formulation with respect to the reference curve (in blue), and, (b) Frenet coordinate frame of the snapshot with the ego (blue circle).

It is important to note that any road structure (curved, straight, etc.) in the Cartesian coordinate

framework can be presented as a straight line in the Frenet coordinate framework as shown in Figure 2.7b. Thus, presenting an autonomous driving task in the latter domain is much simpler.

2.3.1 Frenet Planner

In Frenet path planning algorithms, first the current position and the velocity of the ego are transformed from the Cartesian coordinates to Frenet coordinates. Next, the algorithm generates candidate trajectories defined by a set of polynomial equations. These equations define the Frenet coordinates as a function of time t. Different polynomial functions can be utilized. Fourth-order (quartic) polynomial

$$s = b_5t^4 + b_4t^3 + b_3t^2 + b_2t + b_1 \tag{2.12}$$

or fifth-order (quintic) polynomial

$$d = c_6t^5 + c_5t^4 + c_4t^3 + c_3t^2 + c_2t + c_1 \tag{2.13}$$

are used for trajectory planning. The parameters in equations (2.12) and (2.13) are optimized for each trajectory for given objectives, e.g., longitudinal and lateral distances to travel, final velocity, planning horizon.

The candidate trajectories are assessed against the future positions of the surrounding vehicles for safety and the unsafe ones are eliminated. The cost of each candidate of the remaining safe candidates is then calculated using a suitable cost function. The trajectory with the lowest cost is finally selected.

2.4 Model Predictive Control

Model Predictive Control (MPC) [47, 48], alternatively referred to as Receding Horizon Control (RHC), is a widely employed control strategy in various domains including process control, robotics, and automotive systems. It involves optimizing a cost function (J), such as penalties for

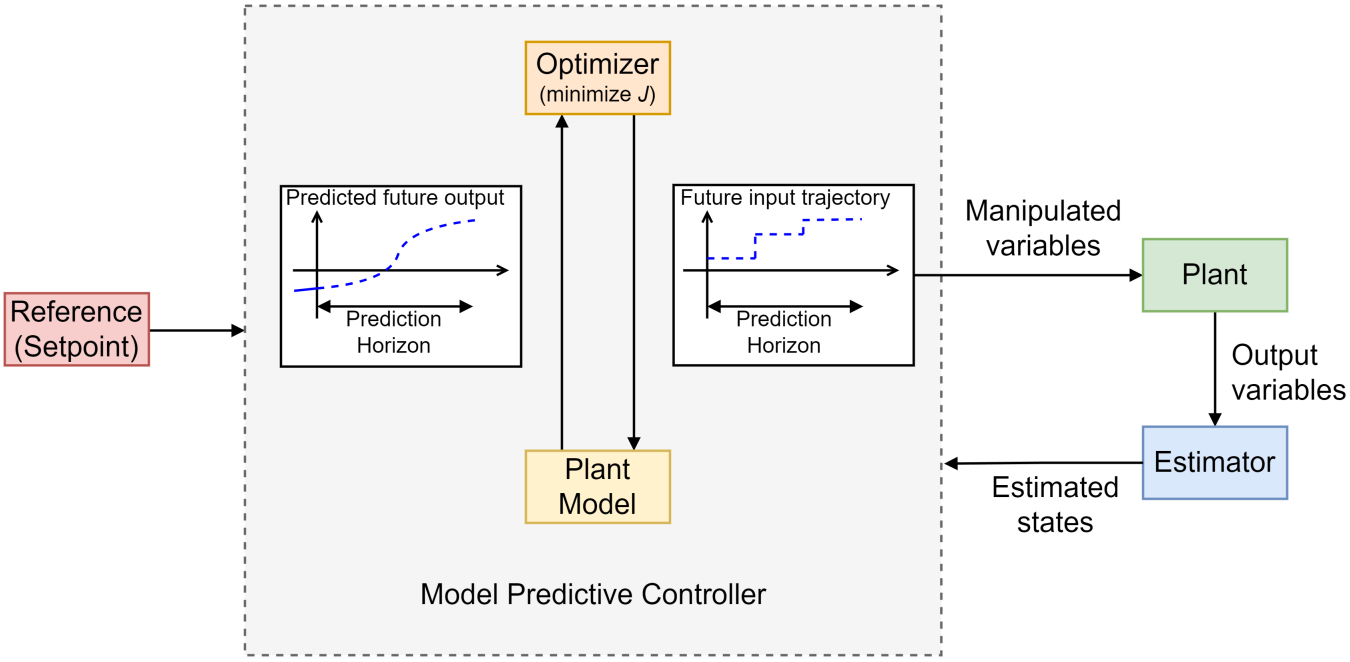


Figure 2.8: Block diagram of an MPC

deviation from desired trajectories or states and for control effort by solving an open-loop problem at each time step. In this optimization problem, the dynamics of the system is taken into account, constraints are imposed on states and controls, and predictions are made about future states based on a system model. The system model, in the case of a linear MPC, is a linear discrete-time system model:

$$X_{k+1} = AX_k + BU_k \tag{2.14}$$

with time-step k, states X, control input U, and matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ with n being the number of states and m being the number of inputs. For the states and the inputs, the following constraints hold

$$X_k \in \mathscr{X}, U_k \in \mathscr{U} \qquad \forall k \in \mathbb{N}_0 \tag{2.15}$$

where, $\mathbb{N}_0$ denotes the set of all non-negative integers. Once the optimization problem is solved, MPC only uses the initial control input from the computed sequence to control the system. During the next time step, the system's current state is measured or estimated. Then, the optimization problem is solved again over a new prediction horizon, with the updated state as the starting point.

The utilization of the receding horizon approach in MPC guarantees the constant adaptation of control actions based on the most recent information while enabling explicit constraint handling. Figure 2.8 shows the block diagram of an MPC.

Chapter 3

Trajectory Prediction for Autonomous Driving

In this chapter, we consider the problem of predicting the trajectories of the detected vehicles, referred to as actors, around the car of interest driving itself autonomously, referred to as the ego (block 3 in the autonomous driving framework of Figure 1.1). These actor trajectory predictions are required as inputs to the next module in the autonomous driving system, i.e., the trajectory planner discussed in Chapter 4, whose role is to compute an appropriate trajectory for the ego that can handle traffic around it.

3.1 Introduction

Human drivers continuously predict the maneuvers of other vehicles on the road to plan a safe and efficient future motion. Similarly, to ensure its safety and the safety of other agents on the road, the Autonomous Driving System (ADS) must predict the motion of the surrounding agents in the future with high accuracy. High accuracy in prediction will also improve the capability of the autonomous car to infer future situations [49, 50] and consequently enhance decision-making to enrich ride quality and efficiency [51, 52].

Trajectory prediction in real-time can pose significant challenges given the dynamic and un-

certain nature of roadways [53]. Although traditional prediction models such as the Kalman filter [54], car-following models [55], and kinematic and dynamic models [56, 57] are hardware efficient, they become unreliable in long-term predictions when the spatial-temporal interdependence is ignored [58]. To mitigate that, Ju et al. [59] combined Kalman filter (KF), kinematic models and neural networks and obtained better performance than the traditional ones. Predicting multiple possible trajectories and ranking them based on probability distribution of the prediction model are also found in the literature [60, 52]. However, these approaches are inherently less pragmatic in real-time scenarios [53].

With significant developments in deep learning, particularly successful implementations of LSTM in capturing temporal dependencies, several works have been conducted for trajectory prediction. In [61] and [62], LSTM is used to extract the vehicle features and then utilized to predict driver intentions. To predict multi-modal trajectories, [63] uses an encoder-decoder LSTM to generate parameters for a weighted Gaussian Mixture Model. Using the modal with the highest probability, the predicted trajectories are then clustered. Kawasaki et al. [64] also predict multi-modal trajectories by combining LSTM with KF using a novel lane feature representation technique. They also incorporate a vehicle motion model constraint that improves the prediction accuracy. For multi-modal trajectory prediction, Deo and Trivedi [65] utilize an LSTM encoder to encode the history, six different LSTM decoders for six different maneuvers, and a separate LSTM decoder to predict the probability of each of the six maneuvers.

Hyeon et al. [66] encodes the historical trajectory using an LSTM encoder and predicts the K most likely future trajectories using an LSTM decoder. Xin et al. [67] use one LSTM to predict the target lane of the vehicle and another LSTM to predict its trajectory based on its current state and the earlier predicted lane. A similar approach is proposed by Zhang et al. [68] that uses multiple LSTM-based framework for predicting intention and trajectory at intersections.

A convolutional neural network (CNN) is believed to be a better choice for trajectory prediction than recurrent neural network (RNN) models since trajectories have strong spatio-temporal continuity [69]. However, most of the literature utilize the bird-eye-view image as an input to the CNN

framework. Using the vehicle state and the raster image as inputs, [52] generates multiple possible future trajectories and determines the trajectory with the highest probability from semantic features. Strohbeck et al. [70] apply a CNN to the rasterized image and a temporal convolutional network (TCN) to the historical trajectories, and the extracted features are concatenated with the current state to determine the future trajectory. In [71], Gilles et al. apply CNN to generate a heatmap representing the agent's possible future.

Transformers, a neural network that incorporates attention mechanism, although initially utilized for machine translation in natural language processing (NLP) [72], have been proved effective in trajectory prediction. Quintanar et al. [73] apply a modified transformer to the past trajectories as its input to predict the future trajectory. Liu et al. [74] apply stacked transformers and use features from past trajectories, social interaction and road data as inputs for multi-modal trajectory prediction. Spatio-temporal transformer networks (S2TNet) is used in [75] to capture both spatio-temporal interactions and temporal sequences. Huan et al. [76] propose trajectory prediction using a multi-head attention transformer layer to model the interaction between agents. In [77], Gao et al. show the relation between the intention and the trajectory of the target vehicle using a dual transformer model.

This chapter proposes a trajectory prediction method for autonomous driving using agent-interaction graph embedding (AiGem). AiGem first constructs a heterogeneous graph from the historical data that encapsulates the interactions between the agents required for trajectory prediction of these agents. A graph encoder network then processes the graph to find embedding of the target vehicles at the current time stamp which is fed through a sequential GRU encoder network for trajectory prediction with the aid of an multilayer perceptron (MLP)-based output network. The results show that AiGem has higher accuracy in longer prediction horizons in contrast to the existing state-of-the-art methods with comparable accuracy for lower predictions horizons. The size of the model is much lighter than that of the compared methods except for [78].

3.2 Related Works

Despite obtaining high accuracy, techniques discussed earlier for trajectory prediction do not include the interactions between the vehicles. Deo et al. [60] addresses this shortcoming by modeling the spatial interactions between the vehicles with a social tensor and then further extracting features using a convolutional social pooling techniques. However, the social tensor only preserve the spatial interaction of the last time stamp of the history, and thus, the spatial-temporal relation is not captured.

Li et al. proposes GRIP++ [79] that capture the spatial-temporal relation using fixed and dynamic homogeneous graphs. They offer two different types of edges – spatial and temporal – to capture the environmental dynamics. This is similar to what is proposed in this chapter, except that in this work, a single heterogeneous graph is used to feed through the graph convolutional model. Furthermore, an encoder GRU network is used in [79], whereas, in the proposed scheme, the graph convolutional model acts as the encoder network. As a result, the model size of GRIP++ is significantly larger than the proposed model. Sheng et al. [78] also proposes a similar graph-based technique in which they stack spatial graphs from each time step in the past to form a spatial-temporal homogeneous graph. They use two different modules for extracting the spatial dependency and then extract the temporal dependency. However, since in this work, both the spatial and temporal aspects are included in the construction of the graph itself, the graph convolutional model alone is sufficient.

Lin et al. [51] uses spatial-temporal attention mechanisms with LSTM for trajectory prediction with primary focus on the explainability of their models. Katariya et al. [53] focuses on model complexity and size for faster performance and lower memory requirement. They use depthwise TCN-based (Temporal Convolutional Networks) encoder instead of LSTMs reducing the overall model size. The performances of both [51] and [53] are comparable, even though not the best, to other existing state-of-the-art methods. In this work, attention is integrated in the depthwise graph convolutional network by graph attention networks (GAT) to capture the importance of neighboring vehicles. Pishgu is proposed in [80] for trajectory prediction that utilizes graph isomorphism

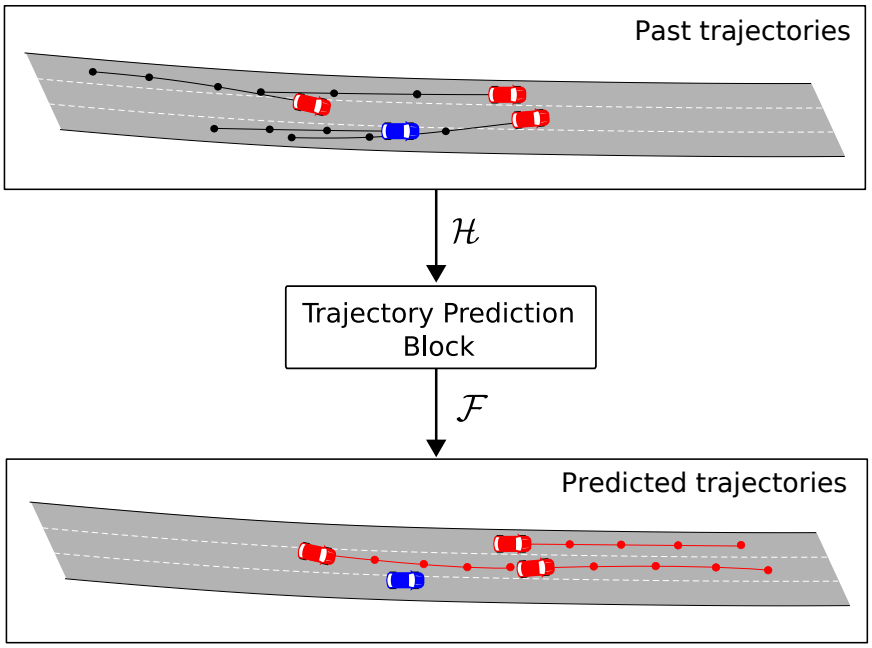


Figure 3.1: The trajectory prediction problem – given the past trajectories (solid black line) of the ego (in blue) and the actors (in red), the trajectory prediction block estimates the future trajectories (solid red line) of the actors. In this example, $K_H = 4$, $K_F = 4$ and $N_{K_H} = 3$

network (GIN) to extract interdependencies of all agents in a scene and further applies attention-based convolutions to underline the important interdependencies.

3.3 Problem Formulation

The trajectory prediction module is responsible to estimate the future positions of all the actors in the current frame based on their trajectory histories given by their global coordinates x and y with respect to the autonomous agent (also known as the ego) at the current time step K_H, their heading θ, and the velocity v. Formally, the trajectory histories over τ_H seconds, with a sampling time t_s (i.e., over $K_H = \frac{\tau_H}{t_s} + 1$ time steps), of all the observed actors can be shown as follows

$$\mathscr{H} = [X_1, X_2, \ldots, X_{K_H}] \tag{3.1}$$

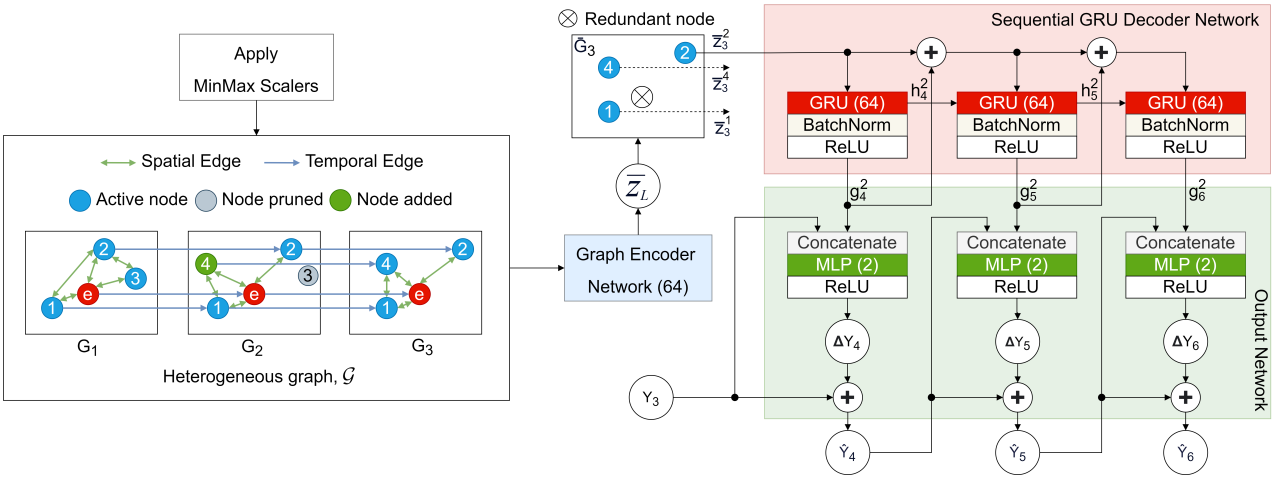


Figure 3.2: Proposed network architecture AiGem with $K_H = 3$, $K_F = 3$. Modules (GRUs and MLPs) inside the networks with the same color share the same weights. The number within parenthesis represents the output dimension of the module.

where, $X_k = \left[x_k^1, y_k^1, \theta_k^1, v_k^1, \ldots, x_k^{N_k}, y_k^{N_k}, \theta_k^{N_k}, v_k^{N_k}\right]$ with N_k being the number of actors observed at the k^{th} step. The task of the prediction module is to predict the future positions over a prediction horizon τ_F (i.e., over $K_F = \frac{\tau_F}{t_s}$ time steps) of all the N_{K_H} observed actors:

$$\mathscr{F} = [Y_{K_H+1}, Y_{K_H+2}, \ldots, Y_{K_H+K_F}] \tag{3.2}$$

where, $Y_k = \left[x_k^1, y_k^1, x_k^2, y_k^2, \ldots, x_k^{N_{K_H}}, y_k^{N_{K_H}}\right]$. Note that position with respect to the ego at the current time step K_H infers that the position of the ego $\left(x_{K_H}^e, y_{K_H}^e\right)$ at present is set to $(0,0)$ and the coordinate frame is shifted as such. This, to some extent, prevents outliers for positional features to occur during training that could lead to poor accuracy [81]. The trajectory prediction problem is illustrated in Figure 3.1.

3.4 Proposed Methodology

In this section, a novel method is proposed to articulate the trajectory prediction problem as a sequence of connected graphs based on the historical data. Essentially, the proposed method uses agent-interaction graph embedding (AiGem) – a heterogeneous graph from the historical data en-

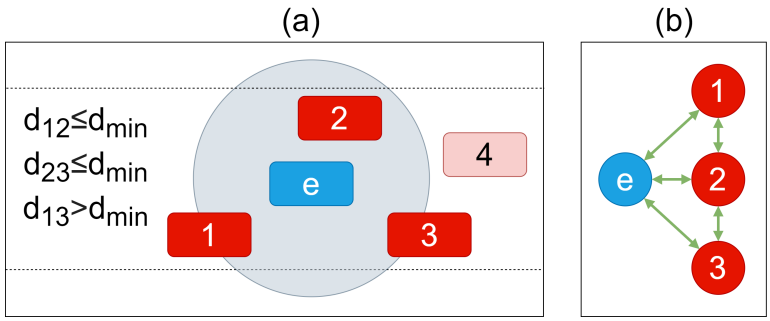


Figure 3.3: (a) An example scenario of an ego (in blue) surrounded by actors (red) of which actors 1, 2, and 3 are in its sensing area (gray circle), and, (b) Graph formulation with ego connected to the sensed actors via bidirectional spatial edges.

capsulates the interactions between the agents. A graph encoder network then processes the graph to find embedding of the target vehicles at the current time stamp which is fed through a sequential GRU encoder network for trajectory prediction with the aid of an MLP-based output network. The proposed network architecture, as shown in Figure 3.2, illustrates the three network components forming a coherent structure for solving the trajectory prediction. The three components are (1) graph encoder network, (2) sequential GRU decoder network, and (3) output network.

3.4.1 Graph Formulation

First, the spatial graph G_k for each time step in the past, i.e., $k = 1, 2, \ldots, K_H$, is generated. Figure 3.3(a) shows an example scenario at the k^{th} time step where the ego is surrounded by four actors of which three actors (1, 2 and 3) are in its sensing area, while the fourth actor is not in that range. Note that the sensing area in this work is defined to be a circle of radius around the ego. In the formulation of the graph as shown in Figure 3.3(b), alongside the ego (referred to as node e), an actor i is considered as a node of graph G if the inequality on the right-hand side of the following equation is met:

$$\mathbb{I}_{\text{node}}(i_k) = \begin{cases} 1, & \text{if } d_{ei} \leq 50 \\ 0, & \text{otherwise} \end{cases}, \tag{3.3}$$

where, $\mathbb{I}$ is an indicator function and d_{ei} is the euclidean distance between the ego and the i^{th} actor. Each node $e_k, i_k \in \mathscr{N}_k$, where $\mathscr{N}_k$ is the set of all nodes present at the k^{th} time step, has the following feature vector:

$$z_k^{e/i} = \begin{bmatrix} x_k & y_k & \theta_k & v_k \end{bmatrix} \in \mathbb{R}^4 \tag{3.4}$$

The ego e always shares a bidirectional edge with a detected actor i.

$$\mathbb{I}_{\text{ea}}\left(E^s_{e_k \longleftrightarrow i_k}\right) = \begin{cases} 1, & \text{if } \mathbb{I}_{\text{node}}(i_k) = 1 \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

In the example shown in Figure 3.3, the ego and the first three actors are part of the graph with bidirectional spatial edges (i.e., d_{ei} is set as the edge feature) between the ego and the actors. Furthermore, two of the detected actors i_k and j_k share bidirectional edges between themselves if the euclidean distance d_{ij} between them is less than or equal to a predefined threshold $d_{\min} = 25\,\text{m}$.

$$\mathbb{I}_{\text{aa}}\left(E^s_{i_k \longleftrightarrow j_k}\right) = \begin{cases} 1, & \text{if } \mathbb{I}_{\text{node}}(i_k) = 1 \\ & \text{and } \mathbb{I}_{\text{node}}(j_k) = 1 \\ & \text{and } d_{ij} \leq d_{\min} \\ 0, & \text{otherwise} \end{cases} \tag{3.6}$$

In the example shown in Figure 3.3, while actors 1 and 2, and, 2 and 3 share an edge between themselves, actors 1 and 3 do not have an edge between them since $d_{13} > d_{\min}$. Because of these spatial edges, G_k is referred to as a spatial graph.

Next, the temporal aspect is incorporated to generate the heterogeneous graph $\mathscr{G}$. The temporal unidirectional edge is defined as follows:

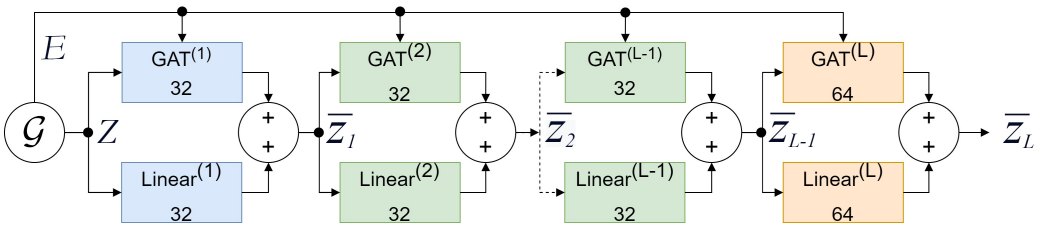


Figure 3.4: Graph encoder network with L number of layers. GAT modules with the same color share same weights and linear modules with the same color share same weights. The integers within parenthesis represent the layer number and the integers without the parenthesis represent the output dimension of the particular block.

$$\mathbb{I}_{\text{temp}}\left(E^t_{i_k \to i_{k+1}}\right) = \begin{cases} 1, & \text{if } \mathbb{I}_{\text{node}}(i_k) = 1 \\ & \text{and } \mathbb{I}_{\text{node}}(i_{k+1}) = 1\,, \\ 0, & \text{otherwise} \end{cases} \tag{3.7}$$

i.e., if actor i is within the sensing area during both the $k^{\text{th}}$and $(k+1)^{\text{th}}$ time frames, they are connected by a temporal unidirectional edge. Intuitively, the temporal edges encapsulate the sequential aspect of the historical data $\mathscr{H}$. The edge attribute is set to be the sampling time t_s. Needless to say, the temporal edge for the ego always exists between the k^{th} and $(k+1)^{\text{th}}$ time steps.

For example, as shown in the heterogeneous graph $\mathscr{G}$ of Figure 3.2, from G_1 to G_2, there exist temporal edges for actors 1 and 2 since both these actors were within the sensing area in both frames. However, actors 3 and 4 do not have the temporal edges since at $k = 2$, the former is not in the sensing area anymore while the latter materialized into the sensing area for the first time. From G_2 to G_3, nothing changed in the sensing area, and thus, all the three actors (1, 2 and 4) have temporal edge connections. Note that the two different types of edges (spatial and temporal) in the formulation of $\mathscr{G}$ makes it a heterogeneous graph.

3.4.2 Graph Encoder Network

The idea of the graph encoder network is to capture the context from the history $\mathscr{H}$ presented as a spatial-temporal heterogeneous graph $\mathscr{G}$. The graph encoder network consists of a series of

GAT convolutional networks in parallel with a series of linear layers as shown in Figure 3.4. The heterogeneous graph $\mathscr{G} = (Z, E)$ is forwarded to get the final encoded embedding $\bar{Z}_L$ of all the nodes in the graph, where Z is the feature array of all nodes in $\mathscr{G}$ and E is the edge connection array between these nodes and the corresponding edge attributes. If the total number of nodes in $\mathscr{G}$ is $N_{\mathscr{G}}$, then $Z \in \mathbb{R}^{N_{\mathscr{G}} \times 4}$ and thus, $\bar{Z}_L \in \mathbb{R}^{N_{\mathscr{G}} \times 64}$ as indicated by Figure 3.4 Formally, the forward pass can be defined using the following equations:

$$\begin{aligned}
\bar{Z}_1 =& \text{GAT}^{(1)}(Z, E) + \text{Linear}^{(1)}(Z) \\
\bar{Z}_l =& \text{GAT}^{(l)}(\bar{Z}_{l-1}, E) + \text{Linear}^{(l)}(\bar{Z}_{l-1}), \\
& \forall l = 2, \ldots, L-1 \\
\bar{Z}_L =& \text{GAT}^{(L)}(\bar{Z}_{L-1}, E) + \text{Linear}^{(L)}(\bar{Z}_{L-1}) \\
=& \begin{bmatrix} \bar{G}_1 & \bar{G}_2 & \ldots & \bar{G}_{K_H} \end{bmatrix}
\end{aligned} \tag{3.8}$$

where, $\bar{G}_k$ contains the node embedding of the nodes of the spatial graph G_k. In the above discussed example, $\bar{Z}_L \in \mathbb{R}^{12 \times 64}$ and $\bar{G}_1, \bar{G}_2, \bar{G}_3 \in \mathbb{R}^{4 \times 64}$. However, only the embedding $\bar{G}_{K_H} \in \bar{Z}_L$ if of interest – this fixed-size representation contains meaningful features about the entire input sequence due to the unidirectional temporal edges.

3.4.3 Sequential GRU Decoder Network

The task is to predict the trajectory of actor $i_{K_H} \in \mathscr{N}_{K_H}$. Without any loss of generality, actor i_{K_H} will be referred as i for simplicity. The task of the decoder is to generate a decoded state based on the current hidden state and the previously generated decoded state. Needless to say, at the beginning, the sequential GRU decoder network takes as inputs the initial hidden states $h^i_{K_H}$ (initialized as zeros) and

$$\bar{z}^i_{K_H} = f\left(\bar{G}_{K_H}, i\right) \in \bar{G}_{K_H} \tag{3.9}$$

where, $f\left(\bar{G}_{K_H}, i\right)$ is a function that extracts the corresponding embedding of actor i from $\bar{G}_{K_H}$. A residual connection is also applied similar to [82] to the output g of the GRU as it allows to leverage previously learned representations. Thus, the forward pass in the network can be defined as follows:

$$\begin{aligned}\left(g^i_{K_H+1}, h^i_{K_H+1}\right) =& \text{GRU}\left(\bar{z}^i_{K_H}, h^i_{K_H}\right)\\ \left(g^i_{K_H+2}, h^i_{K_H+2}\right) =& \text{GRU}\left(\bar{z}^i_{K_H} + g^i_{K_H+1}, h^i_{K_H+1}\right)\\ \left(g^i_k, h^i_k\right) =& \text{GRU}\left(g^i_{k-1} + g^i_{k-2}, h^i_{k-1}\right),\\ & \text{for } k = K_H+3, \ldots, K_H+K_F\end{aligned} \tag{3.10}$$

3.4.4 Output Network

For the prediction horizon $k = K_H+1, \ldots, K_H+K_F$, the output network consists of a MLP that takes g^i_k as input to predict the trajectory of actor i. The output of the MLP $\triangle Y^i_k$ is the change in position in both the x and y coordinates with respect to the previous actual or the estimated position. Thus, the final predicted position $\hat{Y}^i_k$ for the prediction horizon is obtained as follows:

$$\begin{aligned}\hat{Y}^i_{K_H+1} =& \text{MLP}\left(C\left(g^i_{K_H+1}, Y^i_{K_H}\right)\right) + Y^i_{K_H}\\ \hat{Y}^i_k =& \text{MLP}\left(C\left(g^i_k, \hat{Y}^i_{k-1}\right)\right) + \hat{Y}^i_{k-1},\\ & \text{for } k = K_H+2, \ldots, K_H+K_F\end{aligned} \tag{3.11}$$

where, $Y^i_{K_H}$ is the current position of actor i and C is a concatenation function. Note that AiGem does not simultaneously predict trajectories of the N_{K_H} actors, rather predicts trajectories of each actor at a time.

3.5 Experiment Setup and Results

3.5.1 Datasets

NGSIM datasets [83, 84], collected by the Federal Highway Administration of the U.S. Department of Transportation, are used to evaluate the performance of the proposed model. In this work, the datasets for highways US-101 [84] and I-80 [83] are used. The NGSIM datasets consist of 45 minutes of vehicle trajectories transcribed from videos – these videos are obtained through synchronized cameras mounted on top of adjacent buildings of the highway of interest. Many deep-learning techniques [60, 79, 51, 53, 78, 80] use these two datasets for performance assessment.

3.5.2 Data Processing

The data in NGSIM were recorded at 10 frames per second, i.e., the sampling time is 0.1 second. However, to make a fair comparison with other techniques [60, 79, 51, 53, 78] which downsampled the data to 0.2 second, the same is done in this work. The trajectories are then segmented into 8 seconds blocks so that the first 3 seconds can be used as historical observation, and the remaining 5 seconds can be used as prediction ground truth. Similar to [51, 53], the dataset is split into 70% training data, 10% validation data, and 20% test data. Note that in the NGSIM datasets, the heading values θ of the vehicles are not provided. However, given the history of a vehicle's trajectory, θ can be easily calculated for this past trajectory using basic trigonometry.

MinMax scalers are applied on the features of the graphs. The range for scaling the positions and the heading is $(-1, 1)$ and the range for scaling the velocity and the distance between the vehicles (spatial edge) is $(0, 1)$. However, scaling is not applied on the temporal edges.

3.5.3 Baselines

Recently, several deep-learning techniques have conducted the task of trajectory prediction on the NGSIM data. The following models for comparing the performance of the proposed AiGem:

(1) CS-LSTM [60]**:** It utilizes an LSTM encoder-decoder model with a social pooling layer for feature extraction.

(2) GRIP++ [79]**:** It utilizes a LSTM encoder-decoder model with fixed and dynamic graphs to capture the environmental dynamics.

(3) STA-LSTM [51]**:** It combines spatial-temporal attention with LSTM and increases the interpretability of the predictions.

(4) DeepTrack [53]**:** It provides a light-weight prediction model by introducing temporal and depthwise convolutions for capturing vehicle dynamics.

(5) GSTCN [78]**:** It utilizes a graph-based spatial-temporal convolutional network to first learn the spatial features and then extract the temporal features. Finally, GRU is used for prediction using the extracted features.

(6) BAT [85]**:** It comprises behavior-aware, interaction-aware, priority-aware, and position-aware modules and is capable of both perceiving and understanding the underlying interactions and managing the uncertainty and variability in predictive modeling.

Pingshu [80] is a more recent work. However, it excludes trajectories of vehicles going for exits and merging lanes, and thus, ignoring predictions of vehicles which are more dynamic in general. Since such constraints are not applied for data processing in this work, it is fair to exclude [80] from the comparison.

3.5.4 Evaluation Metrics

The proposed model, AiGem, is compared against existing deep-learning techniques using different metrics. The followings are commonly used as a measure of prediction accuracy and performance of the system:

Average displacement error (ADE): It is the average euclidean distance between the predicted positions $\hat{Y}_k$ and the ground truth Y_k for $k = K_H + 1, \ldots, K_H + K_F$ and for N_{K_H} actors:

$$\text{ADE} = \frac{\sum_{n=1}^{N_{K_H}} \sum_{k=K_H+1}^{K_H+K_F} \left\| \hat{Y}_k^i - Y_k^i \right\|_2}{K_F N_{K_H}} \tag{3.12}$$

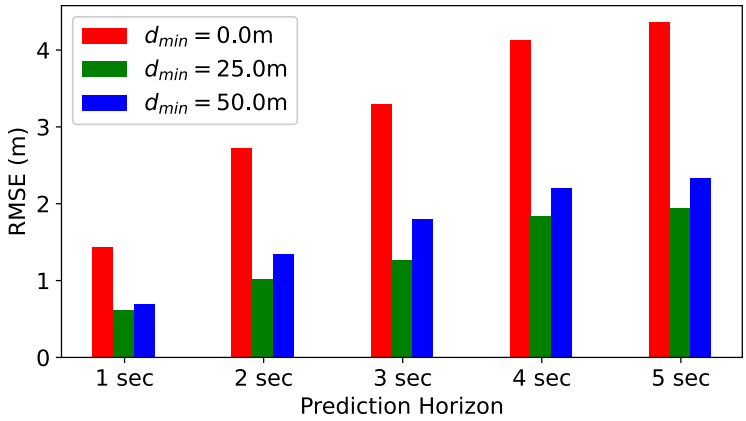


Figure 3.5: Comparison of performance of AiGem for different values of d_{min}

Final displacement error (FDE): It is the average euclidean distance between $\hat{Y}_k$ and Y_k for the last predicted step $k = K_H + K_F$ and for N_{K_H} actors:

$$\text{FDE} = \frac{\sum_{n=1}^{N_{K_H}} \left\| \hat{Y}^i_{K_H+K_F} - Y^i_{K_H+K_F} \right\|_2}{N_{K_H}} \tag{3.13}$$

Root mean square error (RMSE): It is the square root of the mean squared error between $\hat{Y}_k$ and Y_k at k^{th} step for N_{K_H} actors:

$$\text{RMSE} = \sqrt{\frac{1}{N_{K_H}} \sum_{k=1}^{N_{K_H}} \left(\hat{Y}^i_k - Y^i_k\right)^2} \tag{3.14}$$

The model size (i.e., number of parameters in the model) of AiGem is also compared with the baselines.

3.5.5 Ablation Study

Two ablation studies are conducted, particularly, (1) the effect of d_{min} as described in equation (3.6), and, (2) the effect of concatenation at the MLP input defined in equation (3.11) on the performance of the AiGem.

(1) For the formulation of the graph in section 3.4.1, equation (3.6) defines that there exists

a bidirectional edge between two actors if the distance between them is less than a predefined threshold d_{min}. The values of d_{min} are set to different values and the performance of AiGem for these different values are compared. Figure 3.5 shows how the proposed model performs for different values of d_{min}. It is clear that when $d_{\text{min}} = 0$ m (i.e., none of the detected actors are connected with each other using an edge in the spatial graph), the error is maximum for all the prediction horizons. Therefore, connecting actors using edges in the spatial graph indeed help minimizing the loss. Next, it can be observed that the error also increases when d_{min} is increased from 25 m to 50 m, i.e., the number of bidirectional edges between actors increases due to increase in d_{min}. This phenomenon makes intuitive sense since in real-life scenarios, a human driver is more likely to make decisions based on nearer actors than actors that are further. Since $d_{\text{min}} = 25$ m results in the minimum RMSE value, it is used in the formulation of the spatial graphs.

(2) In the proposed architecture, as defined in equation (3.11), the output is concatenated with the decoded embedding at the input of the MLP of the output network. To see the performance without concatenation, i.e., equation (3.11) is modified to:

$$\begin{aligned} \hat{Y}^i_{K_H+1} =& \text{MLP}\left(g^i_{K_H+1}\right) + Y^i_{K_H} \\ \hat{Y}^i_k =& \text{MLP}\left(g^i_k\right) + \hat{Y}^i_{k-1}, \\ & \text{for } k = K_H+2, \ldots, K_H+K_F \end{aligned} \tag{3.15}$$

Figure 3.6 shows the differences between the performances with and without concatenation for different prediction horizons. It is clear that for lower prediction horizons (1, 2 and 3 seconds), concatenation has clear advantage. However, for longer predictions (4 and 5 seconds), concatenation degrades the performance.

3.5.6 AiGem Training

In this work, the Adam optimizer is used to optimize AiGem with a default learning rate of 0.001. Figure 3.7 shows the evolution of the RMSE error during training. Note that it took approximately 100 training epochs to get the best model for all prediction horizons – the best model refers to the model with the lowest validation loss. As the prediction horizon increases, overfitting becomes

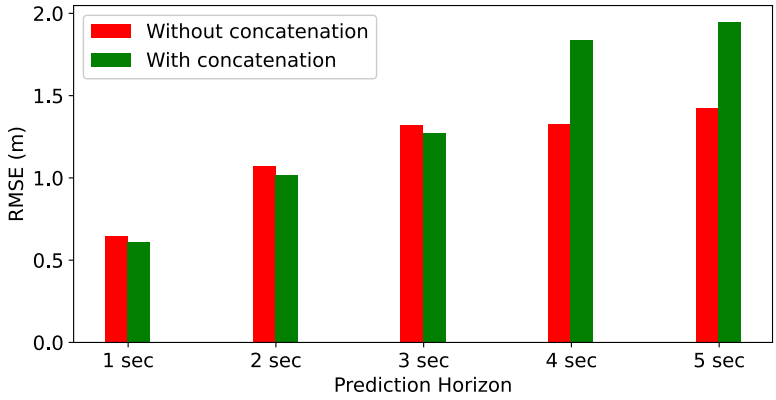


Figure 3.6: Comparison of performance of AiGem with and without concatenation of the output

evident in the training of AiGem.

It is important to note that an actor $i \in \mathcal{N}_{\mathcal{K}_{\mathcal{H}}}$ whose trajectory is to be predicted may not have a complete history. Thus, the impact of the history available of an actor on the accuracy of its trajectory prediction by AiGem is analyzed. Figure 3.8 shows the performance of the proposed network for different actors with varying number of historical data points (shown as a percentage of 3 seconds history). Although there is not a consistent overall pattern, it is noticeable that AiGem tends to forecast trajectories more accurately for actors with shorter available history, specifically those below or equal to 25%. More particularly, the prediction accuracy is better for vehicles with history below or equal to 25% than for vehicles with history above 75%. This may seem counter-intuitive since one would expect longer available history leads to higher accuracy instead. Table 3.1 displays the training errors for AiGem when trained to predict trajectories of actors with particular length of history. It can be seen that the intuition stated earlier matches with the training error - the error tends to decrease as the available history increases. However, from Table 3.2, it can also be observed that there is a trend of increasing overfitting of the network, denoted by $(\text{RMSE}_{\text{valid}} - \text{RMSE}_{\text{train}})$, with increasing history availability. Thus, it can be speculated that a smaller amount of historical data points makes AiGem less prone to overfitting and improves its performance in predicting trajectories of actors with lesser history.

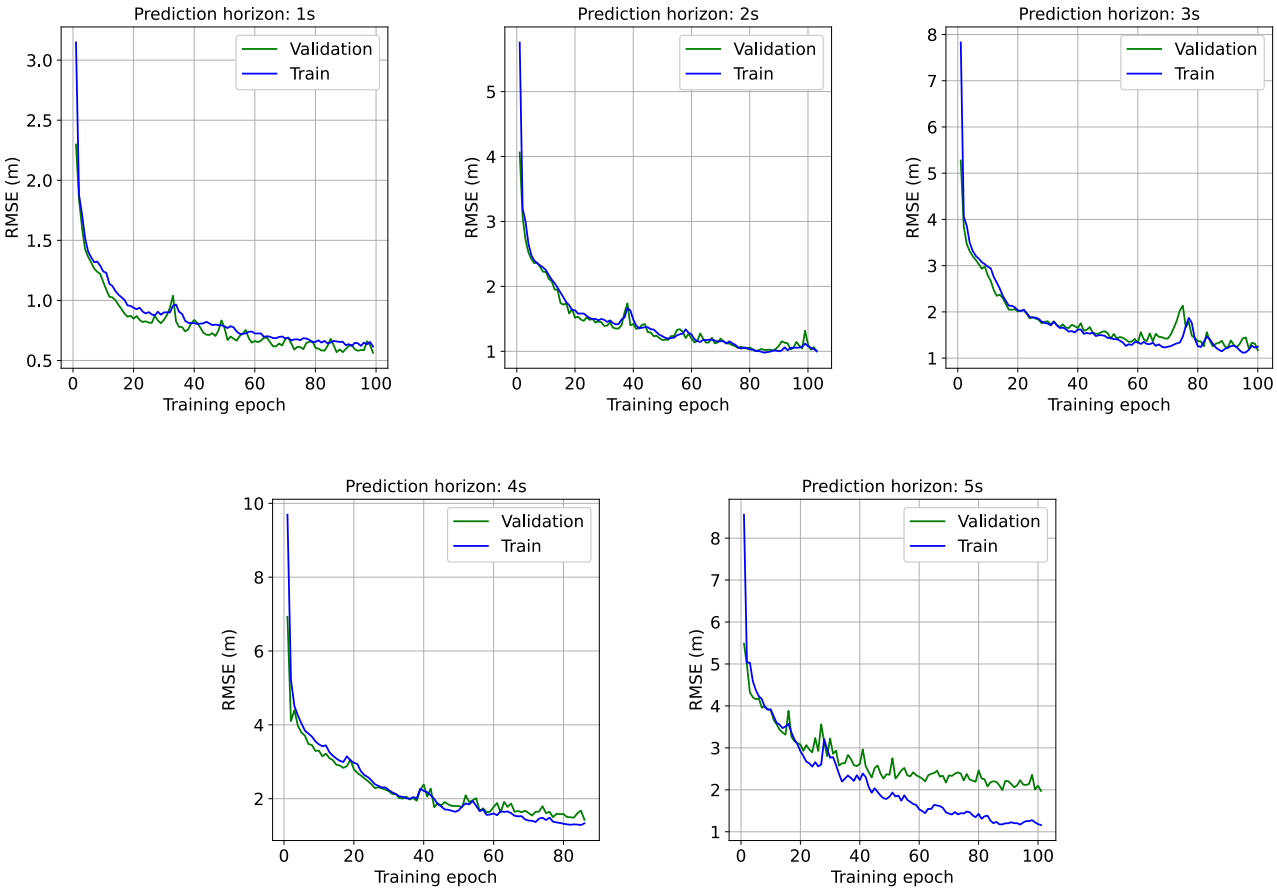


Figure 3.7: Training of AiGem showing RMSE evolution for training and validation datasets

3.5.7 Results

Table 3.3 shows the performance comparison between AiGem and other baselines. In RMSE comparison, AiGem (with and without concatenation) achieves comparable results in short predictions, i.e., 1 and 2 seconds. However, in the longer prediction horizons, i.e., 3, 4 and 5 seconds, it is at the forefront of all the models. For 3 seconds prediction, AiGem with concatenation has the best performance while AiGem without concatenation has the second best performance and leads the third best by 4.5% and 0.7%, respectively. For prediction horizons of 4 and 5 seconds, AiGem without concatenation and AiGem with concatenation have the best and the second best RMSE values, respectively. AiGem without concatenation has improved the results by 33.8% and 51.8% for 4 and 5 seconds prediction horizons, respectively. These results show that AiGem is more proficient in extracting useful features for longer predictions.

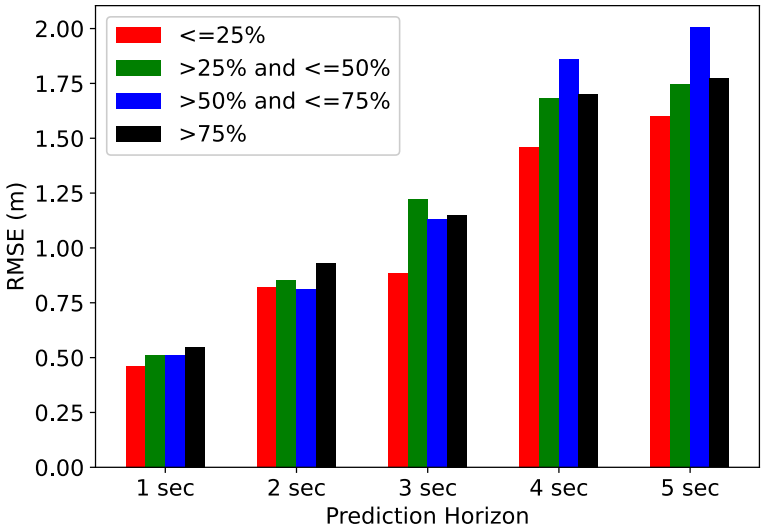


Figure 3.8: Comparison of RMSE when partial or complete history is available for different prediction horizons

Table 3.1: RMSE values (training) obtained for the best models of AiGem trained for actors having different lengths of history

Historical data available	**$RMSE_{train}$ (m)**				
	1s	**2s**	**3s**	**4s**	**5s**
$\leq$25%	0.62	1.01	1.20	1.11	1.51
$>$25% and $\leq$50%	0.63	1.02	1.19	1.10	1.17
$>$50% and $\leq$75%	0.64	0.94	1.19	1.20	1.02
$>$75%	0.51	0.77	0.92	1.11	0.89

As shown in Table 3.3, AiGem has the best performance in terms of ADE, and compared to CS-LSTM, GRIP++, STA-LSTM, DeepTrack and GSTCN, it reduces ADE by 33.2%, 5.0%, 19.1% and 23.9%, respectively. Althoguh GSTCN has the best score in ADE, AiGem scores only 0.7% lower. For the metric FDE, AiGem has the best score among all the models. It can outperform the baselines by 18.3%, 13.6% and 16.0% compared to CS-LSTM, STA-LSTM and DeepTrack, respectively.

As shown in Table 3.3, compared to CS-LSTM, GRIP++, STA-LSTM and DeepTrack, AiGem reduces ADE by 46.7%, 24.2%, 35.5%, 39.3% and 24.6%, respectively. For the metric FDE,

Table 3.2: $\mathbf{RMSE_{valid} - RMSE_{train}}$ (difference between validation and training errors to demonstrate overfitting) obtained for the best models of AiGem trained for actors having different lengths of history

Historical data available	$\mathbf{RMSE_{valid} - RMSE_{train}}$ **(m)**				
	1s	**2s**	**3s**	**4s**	**5s**
$\leq$**25%**	– 0.06	0.05	0.01	0.27	0.43
$>$**25% and** $\leq$**50%**	– 0.1	0.05	0.02	0.41	0.80
$>$**50% and** $\leq$**75%**	– 0.06	0.0	0.08	0.56	0.76
$>$**75%**	0.01	0.09	0.1	0.30	0.65

AiGem without concatenation also has the best score among all the models. It can outperform the baselines by 36.2%, 32.6% and 34.4% compared to CS-LSTM, STA-LSTM and DeepTrack, respectively.

The AiGem model without concatenation is also the second best in terms of model size as shown in Table 3.3. Compared to heaviest model, it has 85.5% less number of parameters. The lightest model is the GSTCN with 33.1% less parameters than the proposed model (without concatenation). The smaller size of GSTCN is likely due to restricting the graph to two laterally adjacent lanes and $\pm$100 meters of the roadways [80] in contrast to AiGem. As explained earlier in section 3.3, the shifting of the coordinate frame with respect to the current position of the ego allows the proposed model to adjust easily to any section of the road.

In summary, the proposed model AiGem, with and without concatenation, is lightweight, yet excels in forecasting trajectories over longer prediction horizons outperforming baseline models. The ADE and the FDE scores also show that the performance of AiGem is superior in contrast to the baselines.

Furthermore, the performance of AiGem on predicting trajectories of actors based on their positions around the ego is analyzed. To do that, broadly speaking, three positions are defined:

1. **Front position**: If an actor is more than 15 m ahead longitudinally from the ego, it is considered in the front position.

Table 3.3: Prediction metrics and model size comparison of AiGem with the baselines. The best results are in bold and the second best are underlined, with some not specifying (–).

Model	ADE (m)	FDE (m)	RMSE (m)					Params (K)
			1s	2s	3s	4s	5s	
CS-LSTM	2.29	3.34	0.61	1.27	2.09	3.10	4.37	192
GRIP++	1.61	–	0.38	0.89	1.45	2.14	2.94	496*
STA-LSTM	1.89	3.16	0.37	0.98	1.71	2.63	3.78	125
DeepTrack	2.01	3.25	0.47	1.08	1.83	2.75	3.89	109
GSTCN	1.52	–	0.44	0.83	1.33	2.01	2.98	**49.8**
BAT	–	–	**0.23**	**0.81**	1.54	2.52	3.62	299*
AiGem (with concatenation)	1.42	2.57	0.61	1.02	**1.27**	1.84	1.95	74.5
AiGem (without concatenation)	**1.22**	**2.13**	0.64	1.07	1.32	**1.33**	**1.42**	74.4

*[79] and [85] do not report their number of parameters but these numbers were extracted from their shared codes in [86] and [87], respectively.

2. **Rear position**: If an actor is more than 15 m behind longitudinally from the ego, it is considered in the rear position.

3. **Mid position**: If an actor is between $\pm$15 m longitudinally of the ego, it is considered in the mid position.

Figure 3.9 shows the performance of AiGem in predicting trajectories of actors based on the positions described above. It can be clearly observed that the proposed model is able to predict trajectories of actors more accurately that are in the rear with respect to the ego, in contrast to the mid and front positions, for all prediction horizons. The model achieves the least accuracy when the actors are in front of the ego. The trajectory prediction problem is essentially to predict the plan of surrounding actors. As shown in the example in Figure 3.10(a), when the task is to predict the plan of actor 2, which is at the rear position with respect to the ego, the formulation of the

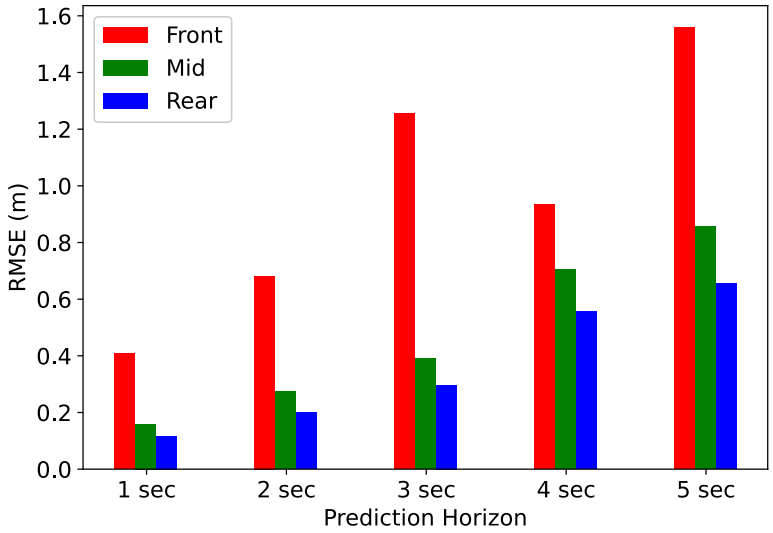


Figure 3.9: Performance assessment of AiGem for predicting trajectories of actors based on the positions around the ego

spatial-temporal graph by AiGem considers the front actors relative to actor 2. On the other hand, when the task is to predict the plan of actor 4 as shown in Figure 3.10(b), which is at the front position with respect to the ego, the formulation of the spatial-temporal graph by AiGem considers the rear actors relative to actor 4. Since it is well-known that drivers focus more on the frontal region while driving to make decisions [88], it makes intuitive sense that AiGem is able to predict better for the former scenario since it has information on the actors in front of actor 2, which is not the case for the latter scenario. In other words, when AiGem predicts the trajectory of an actor i, it is capable of capturing the intuition that front actors relative to i to have more impact on its future decision-making while rear actors relative to i have lesser impact on its future decision-making.

3.6 Conclusion

In this chapter, a deep learning model called the AiGem has been proposed that constructs a heterogeneous graph from the historical data using spatial and temporal edges to capture interactions between the agents. A graph encoder network generates embedding for the target actors in the current timestamp which is fed through a sequential GRU decoder network. The decoded states

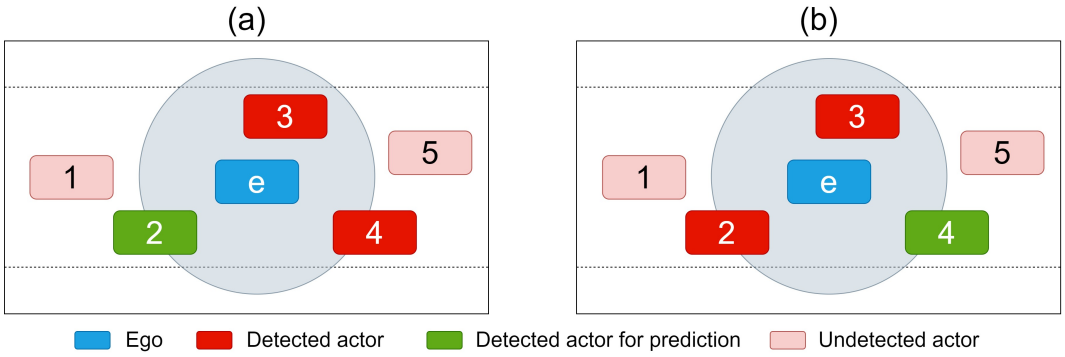


Figure 3.10: Example scenarios – (a) Predict trajectory of actor 2, and, (b) Predict trajectory of actor 4

from the decoder network are then utilized to predict future trajectories using a MLP in the output network. NGSIM datasets have been used for performance assessment. The results show that AiGem achieves comparable accuracy to state-of-the-art prediction algorithms for shorter prediction horizons. For longer prediction horizons, particularly 4 and 5 seconds, it outperforms all the baselines used. The size of the model is better than most of them and comparable to the lightest.

Separate models have been used for different predictions horizons. While this practice is common, a single-model approach for multi-horizon forecasting is more desirable as it will minimize the maintenance and the cost of implementation. Therefore, for future work, we would like to explore and modify the model architecture in order to obtain a high accuracy single-model for multi-horizon predictions.

Chapter 4

An Online Spatial-Temporal Graph Trajectory Planner

In this chapter, we consider the problem of generating a safe and reasonable trajectory for the ego car avoiding the actors around it over the next short time horizon (block 4 in the autonomous driving framework of Figure 1.1). The predicted actors' trajectories of Chapter 3 are used in the computation of the planned ego trajectory. This planned trajectory for the ego is required as input to the next module in the autonomous driving system, i.e., the lateral vehicle controller discussed in Chapter 5, whose role is to steer the ego so that it can adequately track its planned trajectory.

4.1 Introduction

Autonomous vehicles have the potential to improve the overall transportation mobility in terms of safety and efficiency. The module which is primary responsible for planning safely the motion of the vehicle through a traffic is the trajectory planner. The trajectory planner is a vast and long-researched area using a wide variety of methods such as different optimization techniques, artificial intelligence and machine learning [89]. In this work, a novel online trajectory planner is proposed by structuring the motion planning problem as sequences of spatial-temporal graphs passing through a sequential graph neural network architecture.

Different approaches have been undertaken by researchers for generating feasible trajectories. Sampling based planners like the Rapidly-exploring Random Tree (RRT) have also been extensively tested on automated vehicles for online path planning [90] due to to ease of incorporating user-defined objectives. Another sampling based technique, state lattice [91, 92], discretizes the state space in a deterministic manner. Although the spatial-temporal version of the state lattice allows to plan with dynamic obstacles, its performance depends on sampling density making it time consuming [93]. However, the resulting paths from graph search based planners and sampling based planners are usually not continuous and thus jerky [94]. Interpolating curve planners are also popular choices, e.g., clothoid curves [95], polynomial curves [96], Bezier curves [97], etc. However, these planners require global waypoints defined and can be time consuming when managing obstacles in real-time [94]. Frenet trajectory planner are also in the family of interpolating curve planners to generate optimal trajectory but utilizes the Frenet coordinate frame instead of the Cartesian coordinate frame [46]. Geisslinger et al. [98] use the Frenet planner to generate candidate trajectories for an ego and then select the best candidate based on five different ethical principles in line with the European Commission (EU). Interpolating curve planners generally generate several candidates of trajectories and the best one is chosen. However, it is not uncommon for such planners to fail to find feasible trajectories.

Graph-based approaches are also commonly found in the literature. These approaches encompass spatial or spatial-temporal representations, whether one dimensional or hierarchical, like a tree across the feasible driving area [99]. Each node of the graph has an associated cost and the graph-based algorithms seek to identify the path minimizing the cost between the adjacent nodes. Graph search based planners A* [100], hybrid A* [101], and variations of these techniques have been widely used. The 2007 DARPA Urban Challenge was won by the vehicle called Boss that utilized the Anytime D* algorithm [102]. In a more recent work, Han et al. [103] used hybrid A* to find collision free paths and further optimized the path using kinematic constraints. A typical characteristic of these algorithms is their reliance on static or semi-static environmental conditions. In reality, road conditions and traffic are dynamic, and reliance on static maps can result in out-

dated or unsafe trajectories. Although the D* algorithm is designed for dynamic environments, its performance is compromised in highly dynamic conditions [104]. Many use one-layered graph in the spatial dimensions with only lateral targets along the road [105, 106]. Gu et al. [91] applies multiple layered graphs with linear edges along the road. The adjacent nodes resulting in the least cost are then used for path optimization. McNaughton et al. [92] explore both spatial and temporal dimensions in real-time in the search of minimal cost path in the graph. Multilayered graph-based trajectory planner is also proposed by [99] where the planning task for a racing environment is divided into offline and online components. The offline part creates multiple drivable trajectories by connecting nodes in the graph, and then in real-time the online part picks the least expensive global state in the scene. However, in multilayered graph-based trajectory planners, constructing and evaluating large-scale node networks can present significant computational challenges, potentially compromising real-time performance.

Aritificial potential field techniques allow to define potential fields using potential functions (PFs) for obstacles, road structures and goals and then plan paths by moving in the descent direction of the field [107]. In [108], Noto et al. generate the reference path to satisfy the dynamic constraints and to move the vehicle in the descent direction of the PFs. Rasekhipour et al. [107] combine the power of model predictive controller to address dynamic constraints, and, the power of PFs to address obstacles and road structures, to generate trajectories for the ego. An online motion planner for vehicle-like robots proposed by Chen et al. [109] use PFs to generate the initial path meeting road constraints. The path is then optimized as an unconstrained weighted objective function for curvature maintenance, obstacle avoidance, and speed profile. However, aritificial potential field techniques have the drawback of getting trapped in the local minima and also can generate oscillatory paths in the presence of obstacles [110]. In this work, personalized PFs are proposed for obstacle avoidance and maximum velocity keeping with priority given to the former.

The usage of machine learning for motion planning also has its fair share in the literature. Sung et al. [111] use neural networks to learn a planner online from data created using off-the-shelf path planning algorithms. The results show that the paths generated by the neural networks

were smoother in contrast to the original paths. However, their network primarily prioritizes the closest obstacle during planning, potentially resulting in path conflicts with other nearby obstacles. Prediction and planning are combined in [112] which deploys a neural network to predict future states of the surrounding vehicles (actors) and an initial strategy. These are then forwarded into a optimization-based differentiable motion planner to determine the final plan by learning the weights of the cost function online. Nevertheless, their method relies significantly on a proper initialization of the control variables for the network to converge. Yang et al. [113] use a hybrid approach where the behavioral learning is done using deep reinforcement learning (DRL) and the planning is done using a Frenet planner. But in their work, only the longitudinal interaction of human-driven vehicles is considered. In [114], a Deep Deterministic Policy Gradient (DDPG) planner is presented that determines the optimal trajectory based on pre-defined initial and final states, and dynamic constraints. However, no obstacle is considered in the environment and it required 40,000 iterations before achieving a good quality trajectory. Hoel et al. [115] extend the AlphaGo Zero algorithm to a continuous state space domain without self-play and combined planning and learning for tactical decision making in autonomous driving. Their technique's applicability is restricted to situations previously encountered during the training process, i.e., the planner needs to be trained on every possible driving scenario in real-world for practical implementations. [116] proposes a safety layer in its reinforcement learning (RL) framework that pilots the exploration process by limiting actions to the safe subspace of the whole action space. These safe actions are determined by taking into account all the possible trajectories of the traffic participants. But because it generates all possible trajectories, and compares to the predicted occupancies of traffic participants, the training time is significantly longer.

In this chapter, spatial-temporal graphs are proposed that incorporate virtual nodes positioned along the road. These virtual nodes are designed to meet the road boundary conditions as well as the kinematic constraints of the ego. The ego and its surrounding actors also are nodes in the graph. The graphs are processed through a network architecture containing sub-networks in series that generate the future plan for the ego. In contrast to most prior methods, this future plan

can include one or more of the following driving behaviors: lane-keeping, lane-changing, car-following, and speed-keeping. Furthermore, a simple behavioral layer is introduced to command the kinematic constraints of the ego for different driving tasks. The trajectory planner is tested for different driving tasks and the results obtained demonstrate better performance trade-off compared to two baselines used in this work.

The main contributions of this work are summarized as follows:

1. A novel spatial-temporal graph that depicts the trajectory planning problem and incorporates road constraints and kinematic constraints.

2. A neural network architecture that can process the above-mentioned graph to generate future plan for the ego.

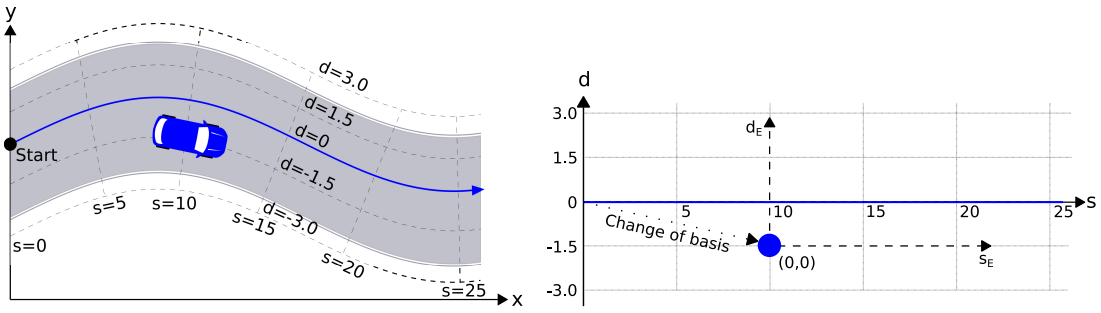
3. Personalized PFs for the architecture to learn on.

4. A simple behavioral layer for defining kinematic constraints of the ego.

The rest of this chapter is organized as follows. Section 4.2 elaborates the proposed Spatial-Temporal Graph (STG) Trajectory Planner. In Section 4.3, the experimental setup and the results are discussed. Finally, Section 4.4 concludes the chapter.

4.2 Spatial-Temporal Graph (STG) Trajectory Planner

In this section, the GNN-based trajectory planner utilizing a sequence of spatial-temporal graphs for trajectory planning is presented. For the formation of the planner, first a simple behavioral layer that determines the kinematic constraints for the ego is discussed. Next, the formulation of the spatial graphs considering the road constraints, the kinematic constraints of the ego and the actors surrounding it are elaborated. Then, the network architecture that processes these graphs to obtain the future trajectory is presented. Lastly, the cost function to be minimized by the network for safety is defined. Even though this work is on the design of the trajectory planner module, there are other vital modules in an autonomous vehicle system. Since the design of those other



(a) (b)

Figure 4.1: (a) A snapshot of road presented in the Cartesian coordinate frame showing the Frenet framework formulation with respect to the reference curve (in blue), and, (b) Frenet coordinate frame of the snapshot with the ego (blue circle). The new Frenet coordinate frame (black solid dashes) is with respect to the ego position.

modules is out of scope of this work, it is assumed that the states of the actors, road structure and the regulations are received from the perception module and the future states of the actors are received from the prediction module.

4.2.1 Frame Conversion

Presenting an autonomous driving task in the Frenet coordinate domain rather than the Cartesian coordinate domain is much simpler. In this work, the Frenet coordinates are used as features. Furthermore, after the problem is translated in the Frenet domain, the basis of the new frame is shifted ($\{s,d\} \rightarrow \{s_e,d_e\}$) with respect to the ego as shown in Figure 4.1b. By applying this shift, it can be ensured that the magnitudes of the virtual nodes, as will be discussed in section 4.2.3, always remain in a constrained range. Lastly, the Frenet coordinate frame also allows us to define the cost of a path very effectively based on the longitudinal and the lateral displacements of the ego and the other actors. Thus, on a high-level, the following steps are applied to solve the trajectory planning problem:

1. A scenario given in the Cartesian frame is first transformed to the Frenet frame.

2. The proposed planner, STG planner, solves the trajectory problem in the Frenet domain.

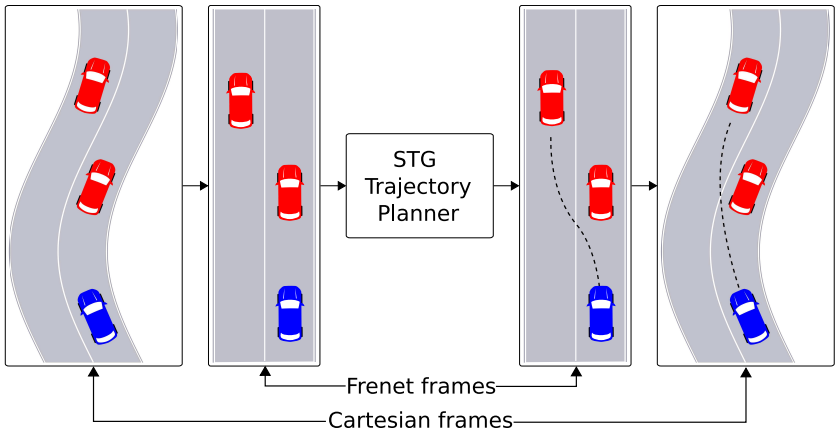


Figure 4.2: Transformations between the Cartesian and the Frenet frames to obtain trajectory (in dashed line)

3. The trajectory obtained in the Frenet frame is then transformed back to the Cartesian frame to get the actual trajectory for the ego.

These steps are illustrated in Figure 4.2.

4.2.2 Behavioral Layer

In this section, a simple behavioral layer is presented for the proposed framework – the design of a more sophisticated behavioral layer is out of scope of this work. The behavioral layer determines kinematic constraints for the ego for different driving tasks considering safety with respect to actors, comfort, and, road regulations. A kinematic constraint refers to a limitation or condition imposed on the vehicle's motion based on its physical characteristics and the surrounding environment. The behavioral layer, presented as a flow diagram in Figure 4.3, is designed to address two particular driving tasks: driving through traffic (DTT), and, following a specific path and speed (FSPS).

At first, parameters prioritizing safety (safety gap s_{safe}), comfort (the maximum longitudinal acceleration/deceleration $a_{\text{max,long}}$, the maximum lateral acceleration/deceleration $a_{\text{max,lat}}$) and road regulations (recommended speed v_{rec} if any, maximum road speed v_{max}, minimum road speed v_{min}) are defined. Next, the longitudinal gap between the rear (s_{rear}) and/or lead (s_{lead}) actor(s), and their

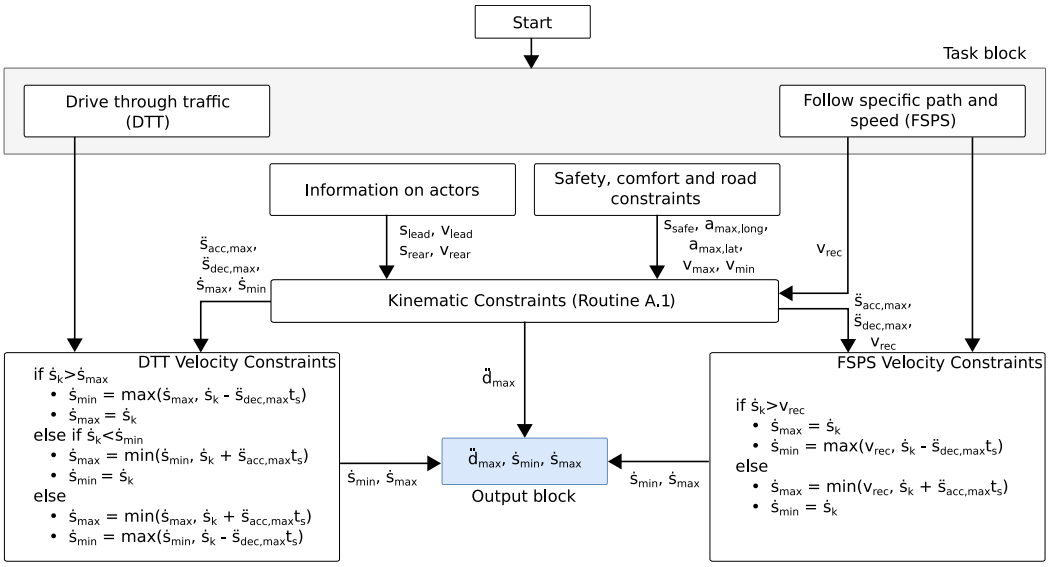


Figure 4.3: Flow diagram for the behavioral layer (see Appendix for Routine A.1)

associated velocities (v_{rear} and/or v_{lead}), are identified. Note that a lead or a rear actor refers to actors which are closest to and are in the same lane as the ego. Using the parameters, the following kinematic constraints are initially applied as:

$$\begin{aligned} \ddot{s}_{\text{dec,max}} &= a_{\text{max,long}} \\ \ddot{s}_{\text{acc,max}} &= a_{\text{max,long}} \\ \dot{s}_{\text{max}} &= v_{\text{max}} \\ \dot{s}_{\text{min}} &= v_{\text{min}} \\ \ddot{d}_{\text{max}} &= a_{\text{max,lat}} \end{aligned} \quad , \tag{4.1}$$

where, $\ddot{s}_{\text{dec,max}}$ is the maximum longitudinal deceleration, $\ddot{s}_{\text{acc,max}}$ is the maximum longitudinal acceleration, $\dot{s}_{\text{max}}$ is the maximum longitudinal velocity, $\dot{s}_{\text{min}}$ is the minimum longitudinal velocity, and $\ddot{d}_{\text{max}}$ is the maximum lateral acceleration. In the event where a lead actor occupies the safety gap during a DTT operation, the following kinematic constraints are substituted for their

equivalents in equation (4.1):

$$\begin{matrix} \ddot{s}_{\text{dec,max}} = 2a_{\text{max,long}} \\ \dot{s}_{\text{max}} = v_{\text{lead}} \end{matrix}. \tag{4.2}$$

If the operation is FSPS, the recommended speed v_{rec} is set to v_{lead}. In the event where a rear actor occupies the safety gap during a DTT operation, the following kinematic constraints are substituted for their equivalents in equation (4.1):

$$\begin{matrix} \ddot{s}_{\text{acc,max}} = 2a_{\text{max,long}} \\ \dot{s}_{\text{min}} = v_{\text{rear}} \end{matrix}. \tag{4.3}$$

However, if the operation is FSPS and the recommended speed is less than $\dot{s}_{\text{min}}$ defined in equation (4.3), then v_{rec} is set to $\dot{s}_{\text{min}}$. In the event when both a rear and a lead actor occupy the safety gap (for both DTT and FSPS operations), the following kinematic constraint is substituted for its equivalent in equation (4.1) while equations (4.2) and (4.3) also apply:

$$\ddot{d}_{\text{max}} = 2a_{\text{max,lat}}. \tag{4.4}$$

Further, if the determined speeds $\dot{s}_{\text{max}}$ and $\dot{s}_{\text{min}}$ from equations (4.2) and (4.3) , respectively, are such that $\dot{s}_{\text{min}} > \dot{s}_{\text{max}}$, then the following is also applied along with equation (4.4):

$$\dot{s}_{\text{max}} = \dot{s}_{\text{min}}. \tag{4.5}$$

Routine A.1 (see Appendix) shows the pseudo code to obtain the kinematic constraints in equations (4.1)-(4.5). Using the constraints, the behavioral layer generates the longitudinal velocity constraints ($\dot{s}_{\text{max}}$, $\dot{s}_{\text{min}}$) and the lateral acceleration constraint ($\ddot{d}_{\text{max}}$) required in the formulation of a spatial-temporal graph as will be discussed in the next subsection.

4.2.3 Spatial-Temporal Graphs

A self-driving vehicle must respect the road constraints, e.g., road/lane boundaries, speed limit, etc., to ensure safety. In addition, it cannot violate the kinematic constraints, e.g., steering limits, acceleration/deceleration limits. In the proposed formulation of the graph, the aim is to incorporate some of these constraints. Note that the formation of the graphs requires knowledge of the future trajectories of the surrounding actors.

An example snapshot is depicted in Figure 4.4(a). In this snapshot, the ego is surrounded by three other actors. The ego is presented as the node e_k in the graph with its longitudinal and lateral positions and velocities as features at the k^{th} step with $k = 0, \ldots, N-1$, where N is the planning horizon and $k = 0$ denotes the initial timestamp. All of the N_A number of actors in a scenario are represented by nodes $A_{i,k+1}$ with $i = 1, \ldots, N_A$. The future lateral and longitudinal positions of the actors at the $(k+1)^{\text{th}}$ step are the features of their corresponding nodes. Next, the idea of virtual nodes V are introduced – these are imaginary points on the road spaced laterally or longitudinally – using kinematic constraints received from the behavioral layer as shown in Figure 4.4(b) – with respect to the ego as shown in Figure 4.4(c) and Figure 4.4(d), respectively. Thus, these virtualized nodes have only one feature – the lateral or the longitudinal spacing from the ego at k^{th} step. Note that the lateral and the longitudinal positions discussed are with respect to a reference path. In the remainder of the chapter, Frenet coordinates d and s will be used for lateral displacement and longitudinal distances, respectively.

The lateral nodes are constrained by the road/lane boundaries and the lateral kinematic constraint $\ddot{d}_{\max}$ obtained from the behavioral layer. At the $(k+1)^{\text{th}}$ step, the maximum lateral velocity of the ego is given by

$$\dot{d}_{\max}^{k+1} = \ddot{d}_{\max} t_s \tag{4.6}$$

where, t_s is the sampling period. Therefore, equation (4.6) imposes the lateral kinematic constraints. The road constraint, i.e., $d_{\text{lower}} \leq d^{k+1} \leq d_{\text{upper}}$, where the boundary conditions d_{lower} and

d_{upper} constrain the ego to stay on the road, is imposed using the following:

$$\begin{aligned} d_{\max}^{k+1} &= \min\left(d_{\text{upper}}, d^k + \dot{d}_{\max}^{k+1} t_s\right) \\ d_{\min}^{k+1} &= \max\left(d_{\text{lower}}, d^k - \dot{d}_{\max}^{k+1} t_s\right) \end{aligned} \tag{4.7}$$

where, $d_{\max}^{k+1}$ and $d_{\min}^{k+1}$ are the maximum and the minimum lateral distances that the ego can move. Given the number of lateral virtual nodes N_V, the nodes are layered laterally with equal spacing as follows:

$$\begin{aligned} V_{d,k+1} = \Big[& \; d_{\min}^{k+1} \quad d_{\min}^{k+1} + \delta_d \quad d_{\min}^{k+1} + 2\delta_d \quad \cdots \\ & \; d_{\max}^{k+1} - \delta_d \quad d_{\max}^{k+1} \Big]^T \in \mathbb{R}^{N_V} \end{aligned} \tag{4.8}$$

where, δ_d is the spacing defined by

$$\delta_d := \frac{\left(d_{\max}^{k+1} - d_{\min}^{k+1}\right)}{(N_V - 1)}. \tag{4.9}$$

For the longitudinal nodes, the kinematic constraints $\dot{s}_{\max}$ and $\dot{s}_{\min}$ obtained from the behavioral layer are applied. Thus, the maximum and the minimum longitudinal distances the ego can traverse are defined by

$$\begin{aligned} s_{\max}^{k+1} &= s^k + \dot{s}_{\max} t_s \\ s_{\min}^{k+1} &= s^k + \dot{s}_{\min} t_s \end{aligned}, \tag{4.10}$$

respectively. For N_V number of virtual longitudinal nodes, the longitudinal layering of these nodes are defined similarly as above:

$$\begin{aligned} V_{s,k+1} := \Big[& \; s_{\min}^{k+1} \quad s_{\min}^{k+1} + \delta_s \quad s_{\min}^{k+1} + 2\delta_s \quad \cdots \\ & \; s_{\max}^{k+1} - \delta_s \quad s_{\max}^{k+1} \Big]^T \in \mathbb{R}^{N_V} \end{aligned} \tag{4.11}$$

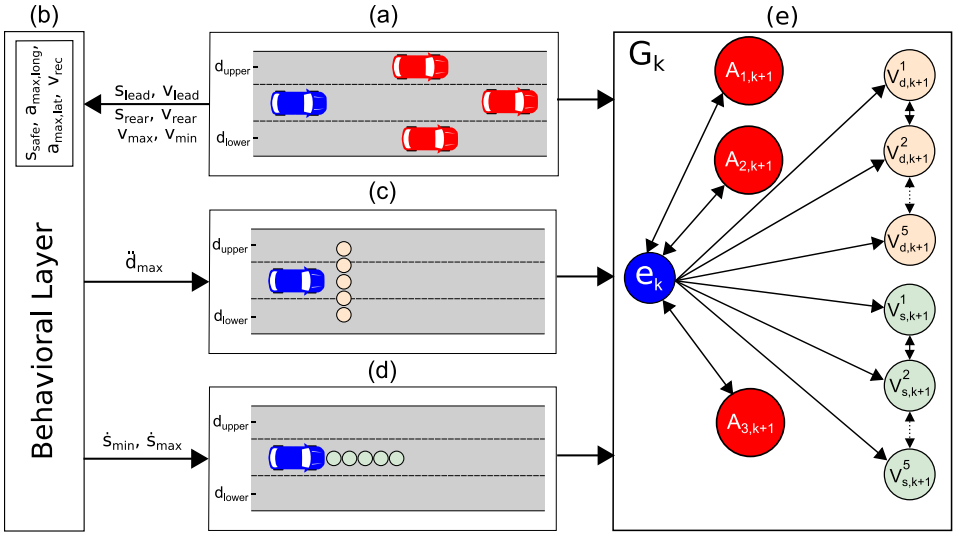


Figure 4.4: (a) The ego (in blue) is surrounded by actors (in red). (b) The behavioral layer generates kinematic constraints for the current scenario. (c) The lateral virtual nodes (in light peach) are spread out laterally along the road respecting road boundaries. (d) The longitudinal virtual nodes (in light green) are spread out longitudinally ahead of the ego along the road. (e) Formation of graph G_k for the given snapshot with $N_V = 5$

where, δ_s is the equal spacing between the adjacent nodes defined by

$$\delta_s := \frac{\left(s^{k+1}_{\max} - s^{k+1}_{\min}\right)}{(N_V - 1)} \tag{4.12}$$

After all the nodes (e_k, $A_{i,k+1}$, $V_{s,k+1}$ and $V_{d,k+1}$) have been defined, the graph G_k is formed as shown in Figure 4.4(e). The edges between e_k and $A_{i,k+1}$ are bidirectional signifying the "*interactions*" between the ego and the other actors – the edge features are the euclidean distances between the ego and the corresponding actors. The edges between the ego and the virtual nodes are unidirectional (e to V_s and V_d) signifying the transition from k^{th} to $(k+1)^{\text{th}}$ step (the temporal aspect of the graph itself). Thus, the edge attribute for the edges between e and, V_s and V_d, is set to be the sampling period t_s. The lateral virtual nodes are connected by bidirectional edges to their adjacent nodes with each edge attribute being the corresponding lateral distance between the corresponding nodes. The interactions between the longitudinal nodes are formulated in the same manner.

4.2.4 STG Network Architecture

The task of the STG network architecture is to output the future longitudinal and lateral positions of the ego. To do that, a GAT network [34] is utilized as an encoder to generate a node embedding of the spatial-temporal graph G_k (equation (4.13)) and create the encoded feature vector Φ_{GAT} (equations (4.14) and (4.15)). An MLP decoder with a fixed-size input further processes Φ_{GAT} to generate the decoded features Φ_{MLP} (equation (4.16)). A softmax function is then applied on Φ_{MLP} in two layers to compute two sets of weights Φ^s_{Softmax} and Φ^d_{Softmax} for the longitudinal and lateral virtual nodes, respectively (equation (4.17)). These weights determine the relative importance of the virtual nodes. Finally, inner products between the virtual nodes and the weights are computed to obtain the future position y_{k+1} of the ego at the $(k+1)^{\text{th}}$ step (equation (4.18)). Each of the mentioned sequential steps are further detailed below and shown in Figure 4.5.

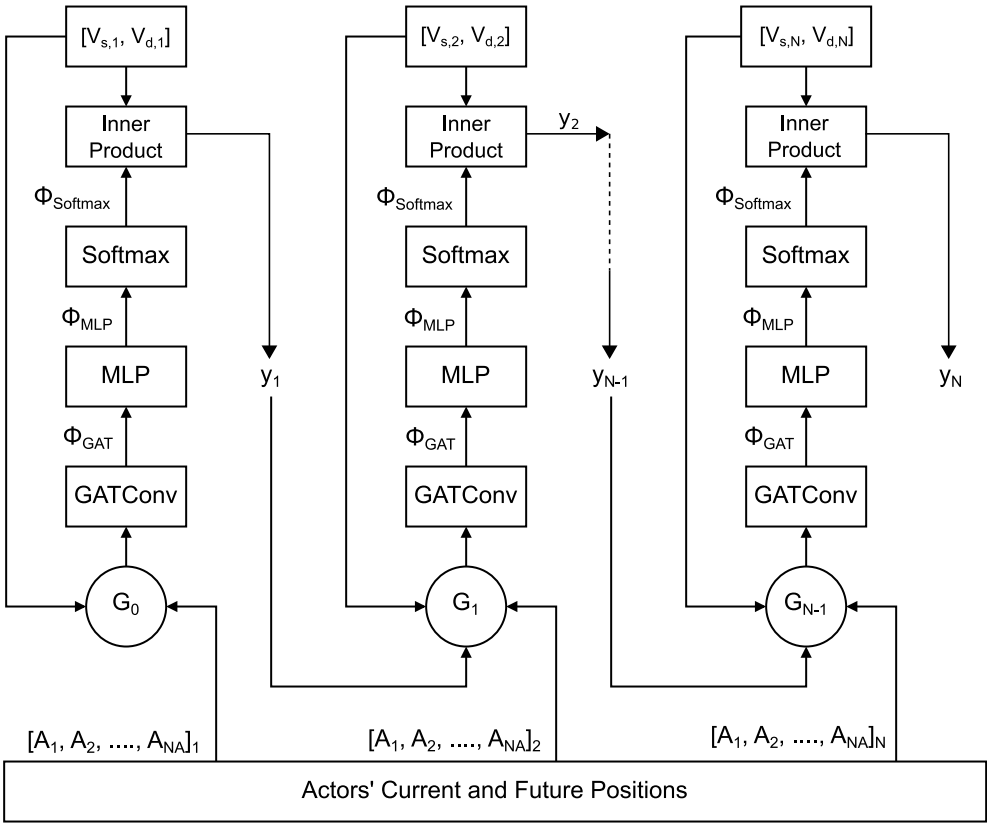


Figure 4.5: Network architecture to learn online the future trajectory of the ego

The graph G_k that represents the snapshot of the ego's surrounding at the k^{th} step needs to be

processed to identify the position of the ego at the $(k+1)^{\text{th}}$ step. In order to do so, the framework shown in Figure 4.5 is proposed. At first, G_k is passed through a GAT network that generates an embedding of size r for each of the nodes, i.e.,

$$\left[\Phi^{e}_{\text{GAT}}, \Phi^{V_s}_{\text{GAT}}, \Phi^{V_d}_{\text{GAT}}, \Phi^{A_1}_{\text{GAT}}, \ldots, \Phi^{A_{N_A}}_{\text{GAT}}\right] = \text{GATConv}\left(G_k\right) \tag{4.13}$$

where, $\Phi^{e}_{\text{GAT}} \in \mathbb{R}^r$ for the ego, $\Phi^{V_s}_{\text{GAT}} \in \mathbb{R}^{N_V \times r}$ for the N_V longitudinal virtual nodes, $\Phi^{V_d}_{\text{GAT}} \in \mathbb{R}^{N_V \times r}$ for the N_V lateral virtual nodes, and $\Phi^{A_1}_{\text{GAT}}, \Phi^{A_2}_{\text{GAT}}, \ldots, \Phi^{A_{N_A}}_{\text{GAT}} \in \mathbb{R}^r$ for the N_A actors. Note that the varying number of actors N_A surrounding the ego cause G_k to be a dynamic graph, i.e., addition and deletion of actor nodes from G_k.

Since MLPs expect a fixed-size input and the GAT network applied to G_k may have different sizes depending on N_A, this structural problem is addressed by applying graph pooling [117] on the actor nodes only:

$$\Phi^{A}_{\text{GAT}} = \Phi^{A_1}_{\text{GAT}} + \Phi^{A_2}_{\text{GAT}} + \ldots + \Phi^{A_{N_A}}_{\text{GAT}} \tag{4.14}$$

such that $\Phi^{A}_{\text{GAT}} \in \mathbb{R}^r$. Graph pooling based on the sum is often adequate for applications involving small graphs as used in this work [117]. Finally, the output of the GAT network is generated by concatenation as follows:

$$\Phi_{\text{GAT}} = \text{concat}\left(\Phi^{e}_{\text{GAT}}, \Phi^{A}_{\text{GAT}}, \left(\Phi^{V_s}_{\text{GAT}}\right)^f, \left(\Phi^{V_d}_{\text{GAT}}\right)^f\right) \tag{4.15}$$

where, $\Phi_{\text{GAT}} \in \mathbb{R}^{2r+2rN_V}$ and the $(\cdot)^f$ operator is the flattening function [118] that returns an array collapsed into one dimension. With Φ_{GAT} as the input, the MLP decoder generates the decoded states as follows:

$$\Phi_{\text{MLP}} = \text{MLP}\left(\Phi_{\text{GAT}}\right) \in \mathbb{R}^{2N_V} \tag{4.16}$$

Note that the output size for the MLP decoder is set to $2N_V$ since the idea is to create weights for the N_V longitudinal nodes and the N_V lateral nodes by applying softmax function S. Then S is

used to normalize Φ_{MLP} in two layers (first N_V elements of Φ_{MLP} for longitudinal weights and the remaining N_V elements for lateral weights) as follows:

$$\begin{aligned}\Phi^{s}_{\text{Softmax}} &= S\left(\Phi^{1:N_V}_{\text{MLP}}\right) \in \mathbb{R}^{N_V} \\ \Phi^{d}_{\text{Softmax}} &= S\left(\Phi^{N_V+1:2N_V}_{\text{MLP}}\right) \in \mathbb{R}^{N_V}\end{aligned} \tag{4.17}$$

Finally, the virtual lateral and longitudinal nodes are weighted using $\Phi^{d}_{\text{Softmax}}$ and $\Phi^{s}_{\text{Softmax}}$ to get the output

$$y_{k+1} = \begin{bmatrix} \left\langle \Phi^{s}_{\text{Softmax}}, V_{s,k+1} \right\rangle & \left\langle \Phi^{d}_{\text{Softmax}}, V_{d,k+1} \right\rangle \end{bmatrix}^{T} \tag{4.18}$$

where, $y_{k+1} = \begin{bmatrix} s^{k+1} & d^{k+1} \end{bmatrix}^{T} \in \mathbb{R}^2$ is a vector of the future longitudinal and lateral positions at the $(k+1)^{\text{th}}$ step, and, the operator $\langle\rangle$ represents the inner product. y_{k+1} is then utilized to obtain G_{k+1} and the same routine is followed to obtain y_{k+2} and so on as shown in Figure 4.5.

4.2.5 Potential Functions

The network needs to backpropagate some "*loss*" to learn. However, in an online trajectory planner, there is no labeled trajectory, and thus, no loss can be calculated. Therefore, potential functions (PFs) of the obstacles are proposed that determines the safety of the ego for a given trajectory. The obstacle PF has maximum value at its position to repel the ego.

For an actor longitudinally and laterally distanced from the ego by $\triangle s$ and $\triangle d$, respectively, is assigned the longitudinal and the lateral potential functions as follows:

$$U^{\text{long}} := \frac{b_1}{\left(b_2 \triangle s + \varepsilon_1\right)^2} \tag{4.19}$$

$$U^{\text{lat}} := \frac{b_3 U^{\text{long}}}{\left(b_4 \triangle d + \varepsilon_1\right)^2} \tag{4.20}$$

respectively, where b_1, b_2, b_3, b_4, and ε_1 are constants. Equation (4.19) – a slight variation of the repulsive potential proposed by [119] – simply indicates that maintaining a larger longitudinal

distance alone is safer. However, safety with respect to lateral distance alone is not sufficient. An actor maintaining a particular lateral distance from the ego at a very close proximity in term of the longitudinal distance possess higher risk than an actor maintaining the same lateral distance but further away longitudinally. Thus, equation (4.20) is proposed that accounts for this phenomenon and Figure 4.6(a) illustrates it. For N_A number of actors surrounding the ego, the total potential function of the obstacles U_o is determined as follows:

$$U_o := \sum_{i=1}^{N_A} U_i^{\text{lat}} \tag{4.21}$$

Next, it is important to realize that when the ego is driving through a traffic, the planner network may slow the ego so that the surrounding actors can pass by suggesting a lower potential U_o. However, this is an ineffective way of solving for the trajectory plan. Thus, to address the issue, a potential function of velocity U_v is also introduced in this work:

$$U_v := c_1 \left(\frac{c_2}{U_o + \varepsilon_2} \right)^{\frac{\dot{s}_{\max}}{\dot{s}}} \tag{4.22}$$

where c_1, c_2, and ε_2 are constants. Equation (4.22) implies that lower velocities should have higher potential when the risk is lower, i.e., U_o is lower. Thus, the velocity PF has its maximal value when the risk is the lowest and the ego is traveling at the minimum velocity. On the other hand, when U_o is higher, the velocity PF is minimum since safety is a higher priority. Figure 4.6(b) illustrates U_v. Finally, the total potential U_{total} (i.e., the "loss") backpropagated into the network is given by

$$U_{\text{total}} = \sum_{k=0}^{N-1} \left(U_o^{k+1} + U_v^{k+1} \right) \tag{4.23}$$

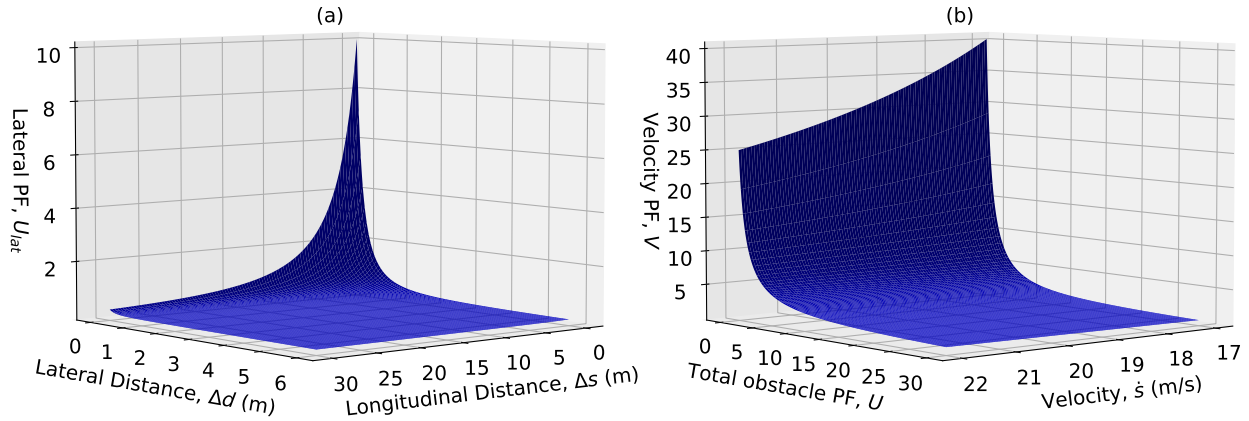


Figure 4.6: Potential functions – (a) Lateral potential function, and, (b) Velocity potential function

4.3 Experimental Results and Analysis

4.3.1 Driving Tasks

In this section, some autonomous driving tasks are defined to evaluate the performance of the proposed planner. These tasks vary in complexity, e.g., vehicle following and lane keeping are on the simpler side of the spectrum, whereas, merging, driving through traffic are at the other end [120]. Often, simpler tasks are sub-tasks of more complex tasks. Particularly, three complex problem statements are defined – vehicle following and lane keeping are sub-tasks of the defined problems.

4.3.1.1 Driving Through Traffic

Aradi [120] suggests that driving through traffic is the most complicated setup to test a planner. Although this task is scalable, the task is limited to highway driving. Three different densities – low, medium, and high – of traffic scenarios (100 for each) were generated using the SL2015 model of SUMO (Simulation of Urban MObility) simulator [121]. The low density traffic scenario consist of 1-5 actors, the medium density traffic scenarios consist of 10-14 actors surrounding the ego, and the high ones consist of 15-20 actors surrounding the ego. The actors are set to respect the speed limits of the road. Once the traffic is generated, an agent from the scene is randomly

selected to play the role of the ego while the trajectories of other agents remain unchanged. In this task, the ego has to plan a safe trajectory for a 5 seconds horizon with 0.1 second sampling time.

4.3.1.2 Merging

In this task, the ego (driving at 50 km/h) has to merge to a highway and its lane ends in approximately 100 m. The recommended speed to enter the merging lane is set at $v_{\text{rec}} = 60$ km/h. There is an actor approaching with a speed of 70 km/h in the merging lane of the highway. There is another actor in the adjacent lane driving at 75 km/h.

4.3.1.3 Taking an Exit

The ego currently driving at 80 km/h has to take an exit with the recommended speed of $v_{\text{rec}} = 50$ km/h. There is an actor approximately 25 m ahead of it driving at 55 km/h. In the course of time, the actor also takes the exit reducing its speed to the recommended speed. Another actor is driving in the adjacent lane at 75 km/h which does not take the exit.

4.3.2 Baselines

Two baselines are used in this work to assess and compare the proposed technique. The kinematic constraints used in both the baselines are the same used in the proposed method.

SL2015 from SUMO is used as the first baseline. The SL2015 model is a lane-changing behavior model used to replicate real-world lane changing maneuvers in a traffic simulation. The model allows to define kinematic constraints such as the maximum lateral and longitudinal speeds. The model makes lane changing decisions based on several factors such as the difference between the current and the desired speeds, cooperation between the driving agents, and the safety gap in the current lane. The data generated using the model already defines the trajectory of the randomly chosen agent as the ego. Thus, the trajectory not only respects the kinematic constraints, but also mimics cooperative behavior with other actors on the road to maintain safety.

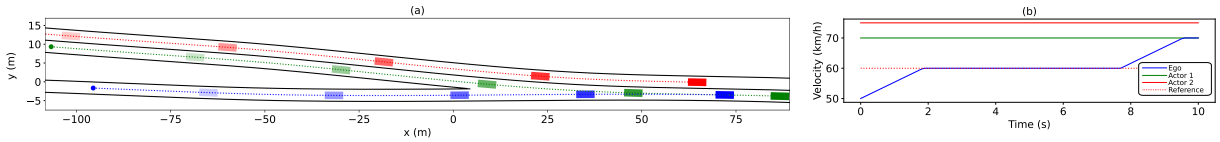


Figure 4.7: (a) The ego (blue rectangle) merging into highway from the start (dot) and the rectangles represent the progression of the vehicles over time spaced by 2 seconds. (b) The velocity profiles of the ego and the actors for the merging task

Next, the Frenet path planner from MATLAB (MFP) [122] is used. It generates multiple candidate trajectories using fourth or fifth-order polynomials relative to the reference path. The candidates are then pruned by checking for kinematic constraints. Next, the trajectories are assessed for collision against the predicted motions of the surrounding actors and the colliding ones are eliminated. Finally, the cost of the remaining trajectories are measured and the least expensive trajectory is selected. Equation (4.23) is used for measuring the cost.

4.3.3 Evaluation Metrics

The trajectory qualities are evaluated in terms of safety, comfort and the longitudinal distance traveled. Safety is assessed by the proposed risk score shown in equation (4.21) as follows:

$$\text{Risk} = \frac{1}{T}\int_0^T U_o(t)\text{dt}, \tag{4.24}$$

where, T is the planning horizon in seconds. Minimum jerk – the time rate of change in acceleration – reflects maximal smoothness , and thus, discomfort score can be defined using integrated absolute jerk [123]:

$$\text{Discomfort} = \frac{1}{T}\int_0^T |J(t)|\,\text{dt}, \tag{4.25}$$

where, $J = \sqrt{J_{\text{long}}^2 + J_{\text{lat}}^2}$ is the total jerk experienced in the trajectory in both longitudinal and lateral directions. The longitudinal distance is extracted from the trajectory itself.

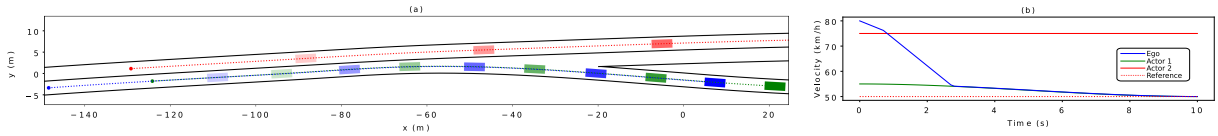


Figure 4.8: (a) The ego (in blue) taking exit and the rectangles represent the progression of the vehicles over time spaced by 2 seconds. (b) The velocity profiles of the ego and the actors for the exit task

Table 4.1: Median score comparisons (best score in bold) between the proposed and the baselines (100 random scenarios for each type of traffic)

Traffic Type	**Planner**	**Discomfort** [m/s^3]	**Risk**	**Longitudinal Distance** [m]
Low density traffic	**STG**	0.52	5.76	108.24
	MFP*	**0.2**	**5.19**	104.22
	SL2015	2.41	5.81	**109.58**
Medium density traffic	**STG**	1.04	**6.30**	**108.55**
	MFP**	**0.22**	7.12	96.85
	SL2015	2.26	10.24	96.98
High density traffic	**STG**	1.12	**7.08**	**109.39**
	MFP***	**0.26**	7.42	97.92
	SL2015	2.88	10.84	98.26

*73 feasible trajectories, **71, ***67

4.3.4 Experimental Results

For the task of driving through medium and high density traffics, the STG planner achieved 100% success rate in planning feasible trajectories, i.e., trajectories that do not cause collision and do not violate the kinematic constraints. However, MFP failed to find feasible trajectories in 27%, 29%, and 33% of the scenarios with low, medium, and high traffics, respectively. The SUMO2015 model being the basis to generate the trajectories, does not account for any failure. Table 4.1 shows the performance comparisons between the three planners for the driving through traffic task. The median score is presented for each of the metrics as it defines the central tendency. Note that the results for MFP shown in the table include only the scenarios where it succeeded.

For all types of traffics, MFP generates the most comfortable trajectories and the proposed planner stands second in the same aspect. However, the primary objective of a planner is the ability to find a feasible path and the proposed STG planner demonstrates 100% success rate in

planning in contrast to MFP. In low density traffic scenarios, MFP performs the best in terms of risk, while SL2015 performs the best in terms of longitudinal distance traveled. However, STG achieves better risk score than SL2015 and better longitudinal distance traveled than MFP for the same traffic. The scores for medium and high density traffic shows that the planner not only achieves more longitudinal distance, but also safer trajectories. MFP generates trajectories that are the shortest in terms of longitudinal distance while SL2015 generates trajectories that are the least safe.Figure 4.7 shows the performance of STG in the merging task. The velocity profile shows that it accelerates to reach the recommended speed (60 km/h) to merge. It continues with the speed until it gets behinds a lead actor (in green) and starts following it effectively.In the task of taking an exit, STG generates a safe trajectory as well as shown in Figure 4.8. The ego starts to slow down but finds a lead actor (in green also taking the exit) and then follows it quite efficiently to take the exit. In both the above two tasks, STG demonstrates its efficacy in the following three sub-tasks: speed-keeping, car-following and lane-keeping.

4.3.5 Interpretability

The trained GAT network encapsulates the interactions between the nodes with an attention mechanism as explained in section 2.1. Figure 4.9 shows the attention in equation (2.3) evolving over time between the ego and the actors in the tasks defined in sections 4.3.1.2 and 4.3.1.3. In the merging task, it can be seen that since both the actors are closing in and remain close, their attention values tend to converge towards each other. In the other task of taking an exit, the attention of the actor taking the exit ahead of the ego increases while the attention of the actor not taking the exit decreases. While it is difficult to explain the magnitude of these attention values, it is rather intuitive that the ego should have increasing focus on the actor taking exit along with it than the one going in another direction.

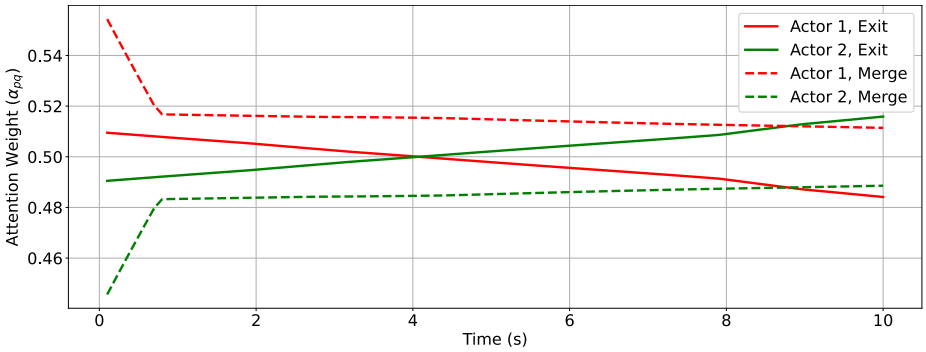


Figure 4.9: α_{pq} between ego p and actor q

4.3.6 Comparative Analysis of Numerical Performance

Wang et al. proposed a trajectory planner by applying Deep Reinforcement Learning (DRL) in the Frenet coordinate system by segmenting the driving tasks into behavior decision making and trajectory planning [124]. The experiments were conducted in CARLA environment for four types of scenarios: straightforward linear road, intricate linear road, uncomplicated curved road and intricate curved road. The traffic type used in [124] is low density traffic as per the definition provided earlier in this chapter. Their model proved to be 6.8% and 13.1% more efficient than the Expectation Maximization Planner (EMP) [125] and a simple RL method, respectively, for lane changing tasks. However, the model was unable to generate feasible trajectories in several instances with a 0.7% failure in the straightforward linear road (the best performance) and a 3.5% failure in the intricate curved road (the worst performance). In contrast, the STG trajectory planner proposed in this chapter successfully generated 100% feasible trajectories under all scenarios, including high, medium and low density traffics.

In [126], a hierarchical framework based on RL is used to generate a trajectory in the Frenet frame. The model was compared to a Monte Carlo Q-learning and a soft actor-critic algorithm (SAC) and all of these techniques achieved 100% success rate in generating feasible trajectories. However, a very simplified two-lane highway scenario including an off-ramp and two vehicles per experiment, served as the basis for the simulations in [126], in contrast to the more complex scenarios used to test the STG trajectory planner.

Naveed et al. propose a Robust-Hierarchical Reinforcement Learning (HRL) framework in which the high-level layer decides from options such as lane follow/wait and lane change, and the low level layer generates the waypoints to follow [127]. A PID controller is further deployed to generate the longitudinal and the lateral control signals. The technique was tested in CARLA such that the vehicles react to each other making the environment interactive – this is unlike the test bed for STG trajectory planner where the ego must plan its path around its actors without relying on cooperative behaviors. The scenario in which the model in [127] is tested is a straight road with high, moderate, and low traffic flows. They compared their models with several techniques including state-of-the-art RL techniques such as the vanilla DQN and the hierarchical double deep Q-Learning and achieved the highest success rate for feasible trajectories. However, they reported 4.5% and 2.5% collision rates with and without Gaussian noise, respectively. On the other hand, STG trajectory planner achieved 0% collision rate under non-cooperative behavior.

A dual variable is used to model the non-convex collision-free constraint in [128] and then further optimized in an MPC framework. With additional constraints for safe and comfortable driving, the final formulated optimization problem is convex which is solved using quadratic programming solvers. The technique is tested under five different scenarios but, unlike the test bed for STG trajectory planner, they are limited to a maximum number of two actors in addition to the ego. The model in [128] was able to generate collision-free trajectories for all the five scenarios.

Cheng et al. developed a path planning algorithm utilizing Gaussian processes to generate continuous paths parameterized by arc length in the Frenet frame, subject to constraints on curvature and avoidance of static obstacles [129]. To address dynamic environmental factors, an efficient spatial-temporal graph search is used to calculate a speed profile along the determined path. Lastly, kinodynamic feasibility is ensured through an iterative, incremental optimization of both path and speed. Unlike the planner presented in this chapter, their methodology was empirically validated in CARLA using interactive agents. The results show that the performance of their method is quite superior to other baselines. However, their success rate in generating feasible trajectories is not perfect, with a failure rate of 1.1%.

4.4 Conclusion

In this work, a novel online spatial-temporal graph trajectory planner is introduced. Heterogeneous graphs are used to formulate the problem by taking into account road boundaries and kinematic constaints. Then, these graphs are processed in a sequential neural network architecture to get the desired future trajectory that can depict common driving behaviors such as lane-keeping, lane-changing, car-following, and speed-keeping. For the network to learn, potential functions addressing safety and maximum velocity keeping are presented as well. In addition, a simple behavioral layer is presented to provide kinematic constraints for the planner. The results show that the proposed planner succeeded in generating feasible trajectories in all the driving tasks. In contrast to the baselines, the metrics point to better trade-off performance.

In the future, the efficacy of the proposed planner in more diverse driving tasks using more complex behavioral layers will be tested. We also plan to extend the application of this method to cooperative multi-agent connected autonomous driving.

Chapter 5

An Online Self-learning Lateral Controller

In this chapter, we consider the problem of lateral control of the ego car (block 5 in the autonomous driving framework of Figure 1.1) so it can track its planned trajectory computed by the trajectory planner of Chapter 4. Longitudinal control of the ego, a simpler control problem, is assumed to be handled by a separate controller and therefore we only treat the ego's lateral control problem here.

5.1 Introduction

The Automated Driving Systems (ADS) industry has experienced a surge in investment and research – the focus of the researchers are primarily, but not limited to, safety, efficiency and convenience [130]. A study has estimated that 90% of fatalities in motor vehicle accidents are driven by human errors in contrast to only 2% resulting from a vehicle malfunction [131]. Promising solutions in self-driving cars are emerging to elevate safety and reshape the transportation industry [132].

One of the significant tasks of the self-driving car, also referred to as the ego, is to drive along a predefined trajectory. This requires solving the control problems for longitudinal and lateral dynamics of the vehicle [133]. This chapter primarily focuses on solving the control problem of the lateral dynamics only. Lateral controllers using fuzzy control [134], H_∞ control [135], sliding mode control [136] have been studied. Another approach is the geometric control, such as the

Pure Pursuit controller [137] and the Stanley controller [138] that utilizes the geometry of the vehicle kinematics and the reference trajectory to compute the steering commands. Performance comparison of these controllers is presented in [139]. However, these methods rely on linear models of the lateral dynamics. In real-world driving scenarios, there are lots of uncertainties in the environments and one particular challenge in ADS is how to deal with varying environments (e.g., road networks, weather conditions, etc.) [140]. For classical solutions, for example, the control of these cars was mostly rule-based and required painstaking manual tuning of the control parameters. Yet, such control laws were challenging to generalize to individual scenarios [141]. Thus, it is necessary that autonomous cars can learn on the fly, i.e., they adopt online learning strategies.

Deep learning allows control agents to adapt to dynamic environments and generalize to new settings using self-optimization through time. Control of the ego car, in general, can be divided into two categories [142]: (1) end-to-end, and (2) the perception and control separation methods. The end-to-end learning controllers directly process sensor readings to control desired output variables. In an early end-to-end method, [143] used a simple feedforward network to train with camera images to generate steering commands. Convolutional Neural Networks (CNN) have also been a popular choice to process camera images for lateral control of the vehicle [144, 145]. Since end-to-end learning methods typically predict a control input for a particular observation and the predicted action affects the following observation, the error can accumulate and lead the network to a completely different observation [146]. Another drawback of end-to-end learning is that it requires a large dataset for training [147]. Kwon et al. [148] proposed an incremental end-to-end learning method where the training is initiated with data collected by a human driver and later replaced by the neural network for further learning. In contrast, the perception and control separation technique has a separate perception module that extracts the features (e.g., position, velocity, etc.) and the features are then used to make decisions by the controller module. Khalil et al. [149] introduce an adaptive neural lateral controller for end-to-end learning in which they utilize a base model and a predictive model inspired by human's near/far gaze distribution. Despite promising

results achieved by end-to-end learning techniques, these suffer from poor interpretability and generalization to new scenes [150]. In this work, the perception and control separation technique is used; however, perfect state measurements are assumed, since the perception module is out of the scope of this work.

Reinforcement Learning (RL) is another frequent choice for lateral control. Ma et al. [151] propose a game-theoretic receding horizon reinforcement learning lateral controller by formulating the uncertainties on the ego as a player by zero-sum differential games. In addition, an actor-critic algorithm combined with a critic neural network and two actor neural networks define their control strategy. Brasch et al. [152] used the soft-actor-critic algorithm of RL to determine steering values to reduce path tracking error. Wasala et al. [153] proposed a novel reward function to minimize tracking errors and improve comfort and safety for their RL algorithm. For lane changing maneuvers, Wang et al. [154] proposed a quadratic function as the Q-function for their RL algorithm and use neural networks to approximate the coefficients of the function. While the RL has the advantage of not requiring a vehicle model, it essentially trains in a trial-and-error mode, making it dangerous to train an agent on an actual vehicle. The proposed methods allow safer training of the actual vehicle in real time.

Another potential method for lateral control is the MPC that has demonstrated reasonable control performance [155][156]. Costa et al. [157] implemented a learning-based MPC algorithm to improve modeling accuracy and perform online tuning of MPC parameters for control. While the approach in [157] is similar to the proposed method, MPC is computationally expensive and onboard computers of autonomous cars may not be powerful enough to solve real-time optimization problems [158]. In [158], Zhou et al. mitigated the computational load problem using event-triggered MPC in contrast to the generally used time-triggered MPC; however, there is a slight performance loss. To achieve the task of platooning, Kazemi et al. [159] used a Laguerre-based MPC and a robust MPC for longitudinal and lateral control, respectively. In [160], RL is used to dynamically update the objective weights of a nonlinear MPC to perform real-time lateral control of the ego.

The main contributions of this chapter are as follows:

- Vehicle modeling: The vehicle is modeled using existing knowledge of vehicle dynamics in unification with GNN, particularly Graph Attention Network, that processes a heterogeneous graph depicting the vehicle. The strength of the model lies in the fact that it can be trained online anytime the performance degrades.
- Lateral controller: A heterogeneous graph representing the state transition of the vehicle – from current to desired – is processed through a GNN-based network to generate steering commands. The controller is competent in online learning, allowing it to adjust its parameters and respond accordingly to unseen environments.
- The methods are validated in CARLA, an open-source high-fidelity ADS platform [161].

To the best of my knowledge, no literature exists that utilizes graphs to solve lateral control problems in autonomous driving.

The rest of this chapter is organized as follows. Section 5.2 introduces the proposed vehicle modeling and the lateral controller. In Section 5.3, the procedures for training the vehicle model and the controller are discussed. Section 5.4 presents the main simulation results and a comparison with the baseline. Finally, Section 5.5 concludes the chapter.

5.2 Vehicle Model and Lateral Controller

In this section, a novel technique is proposed for the lateral controller and an enhancement for the modeling of the vehicle by taking advantage of the expressive power of graphs to represent real-world problems (e.g. [162, 163]).

5.2.1 Network Architecture for Graph

In this work, the network architecture shown in Figure 5.1 is implemented to process a graph. The framework consists of multiple layers, and each layer comprises a GAT and a linear layer in

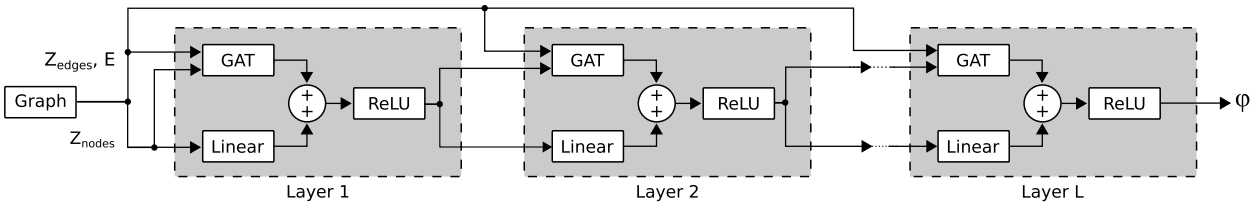


Figure 5.1: Network architecture (G2O) used in this work to map a graph – defined by its features Z and edge connections E – to output φ

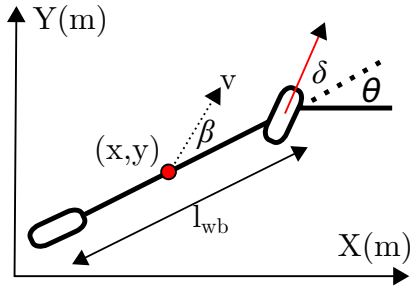


Figure 5.2: Dynamic vehicle bicycle model where the two axles are represented as single wheels

parallel, the output of which is passed through a ReLU function. For convenience, this framework will be referred as G2O (graph-to-output) in the rest of the chapter.

For Layer 1, the input is the original graph $G(Z,E)$, where Z is a collection of both node features Z_{nodes} and edge features Z_{edges}. The output of the network φ is given by the following:

$$\begin{aligned}
\bar{Z}_1 =& \text{ReLU}\left(\text{GAT}^{(1)}\left(Z_{\text{nodes}}, Z_{\text{edges}}, E\right) + \right. \\
& \left. \text{Linear}^{(1)}\left(\left(Z_{\text{nodes}}\right)^f\right)\right) \\
\bar{Z}_l =& \text{ReLU}\left(\text{GAT}^{(l)}\left(\bar{Z}_{l-1}, Z_{\text{edges}}, E\right) + \right. \\
& \left. \text{Linear}^{(l)}\left(\left(\bar{Z}_{l-1}\right)^f\right)\right), \forall l = 2, \ldots, L \\
\varphi =& \bar{Z}_L \in \mathbb{R}^{N \times M}
\end{aligned} \tag{5.1}$$

where, $(\cdot)^f$ operator is the flattening function [118] that returns an array collapsed into one dimension, N is the number of nodes and M is the output feature dimension of G2O.

5.2.2 Proposed Vehicle Model

Since the proposed controller outputs the steering command and not the actual states based on which the network needs to be optimized, an intermediate differentiable and a high accuracy representation of the actual vehicle is needed. A commonly used model for ego cars, which is also differentiable, is the dynamic bicycle model as shown in Figure 5.2, in which the front and rear wheel pairs are each lumped into a single wheel since roll dynamics are ignored. Assuming the center of mass is at the center of the wheelbase (CW), the local frame of the vehicle is considered at CW. The dynamic differential equations of the model are as follows:

$$\dot{\hat{X}}_b = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v\cos(\theta+\beta) \\ v\sin(\theta+\beta) \\ \frac{v\tan(\delta)\cos(\beta)}{l_{\text{wb}}}, \\ a_{\text{net}} \end{bmatrix} \tag{5.2}$$

where, x and y are the global coordinates of the CW, θ is the heading, v is the velocity of the vehicle, δ is the steering angle, β is the slip angle defined as

$$\beta = \arctan\left(\frac{\tan(\delta)}{2}\right), \tag{5.3}$$

l_{wb} is the length of the wheelbase, and a_{net} is the net acceleration. Since, in CARLA, the vehicle takes throttle (T) as input instead of acceleration (a) and also experiences drag force (F_{drag}), the net acceleration can be defined by

$$a_{\text{net}} = a - \frac{F_{\text{drag}}}{m_v}, \tag{5.4}$$

with,

$$F_{\text{drag}} = \frac{1}{2}\zeta A_f \rho_{\text{air}} v^2 \tag{5.5}$$

and,

$$a = 6.5T^2 + 0.6T + 0.08, \tag{5.6}$$

where, m_v and ζ are the mass and the drag coefficient respectively obtained directly from CARLA, A_f is the frontal area that has been approximated based on the bounding box of the vehicle provided in CARLA, and ρ_{air} is the air density. Equation (5.6) is obtained by a second-order polynomial curve fitting of throttle versus acceleration data obtained through simulation in CARLA. It is important to note that the equation (5.6) is not an accurate relation between the acceleration and the throttle. However, the idea is to combine the power of physics with the learning of neural network to improve the accuracy in estimating the states. Thus, while the bicycle model utilizes equation (5.6), it is empowered by the neural network to achieve higher accuracy, which is discussed in the remaining part of this section and further demonstrated in the results section (section 5.4.3).

The bicycle model described by equation (5.2) is limited by assumptions. Given a vehicle is a nonlinear dynamical system running under uncertainties, the bicycle model cannot determine the states accurately which is crucial for the controller performance. Therefore, the following Graph Vehicle Model (GVM) is proposed to estimate the states:

$$\dot{\chi} = \Gamma \odot \dot{\hat{X}}_b, \tag{5.7}$$

where $\Gamma \in \mathbb{R}^{4\times 1}$ allows dynamic model adjustment and the $\odot$ operator represents element-wise multiplication. The mapping Γ for vehicle modeling is learned using G2O$_m$ as follows:

$$\Gamma = \text{MLP}_m\left((\varphi_m)^f\right) \in \mathbb{R}^{4\times 1} \tag{5.8}$$

where, the subscript m refers to modeling, $\varphi_m \in \mathbb{R}^{6\times M_m}$ (note that there are six nodes in G_m), MLP refers to a deep multilayer perceptron with input dimension $6M_m$, and, G2O and φ have already been defined in section 5.2.1. Note that an MLP can be defined using the standard equation:

$$z^{(l)} = \sigma\left(W^{(l)} z^{(l-1)} + b^{(l)}\right), \text{ for } 1 \leq l \leq L \tag{5.9}$$

where, $z^{(0)}$ is the input, and $z^{(L)}$ is the output, L is the number of layers including the input and the

output layers, and, $W^{(l)}$ and $b^{(l)}$ are its learnable parameters in the l^{th} layer. The GVM architecture is shown in Figure 5.3.

The network is fed with a heterogeneous graph G_m, depicting the four wheels and the actuators (steering and throttle) as nodes. The features of the actuator nodes are their corresponding steering and throttle values at k^{th} time $\left(Z_k^{\delta}, Z_k^{T} \in \mathbb{R}\right)$. The front wheel and the rear wheel nodes have the following features

$$Z_{k,l/r}^{w,f} = \begin{bmatrix} s_{k,l/r}^{w,f} - s_k \\ d_{k,l/r}^{w,f} - d_k \\ v_k \\ \theta_k + \delta_{k-1} \end{bmatrix} \tag{5.10}$$

and,

$$Z_{k,l/r}^{w,r} = \begin{bmatrix} s_{k,l/r}^{w,r} - s_k \\ d_{k,l/r}^{w,r} - d_k \\ v_k \\ \theta_k \end{bmatrix}, \tag{5.11}$$

respectively, where s and d are the corresponding longitudinal and lateral coordinates of x and y respectively in the Frenet coordinate frame with respect to the reference path, the superscripts w, f and r represent wheel, front and rear respectively, and the subscripts r and l represent right and left respectively. Thus, the first two elements of the feature vectors of the wheels are in reference to CW. At the k^{th} step, the front wheels are further angled by the steering angle at the $(k-1)^{\text{th}}$ step. The wheels positions can be easily found using basic trigonometry and their corresponding Frenet coordinates are computed. The edge features between the wheels are the distances between them. Between the actuator and the wheel nodes, the sampling period t_s is assigned as the edge feature. However, no edge feature is assigned to edges between the actuator nodes themselves. The assigned edge features are constant throughout time. Figure 5.3 also shows the directions of the edges of G_m. It is not apparent how the wheels would "communicate" with each other, and

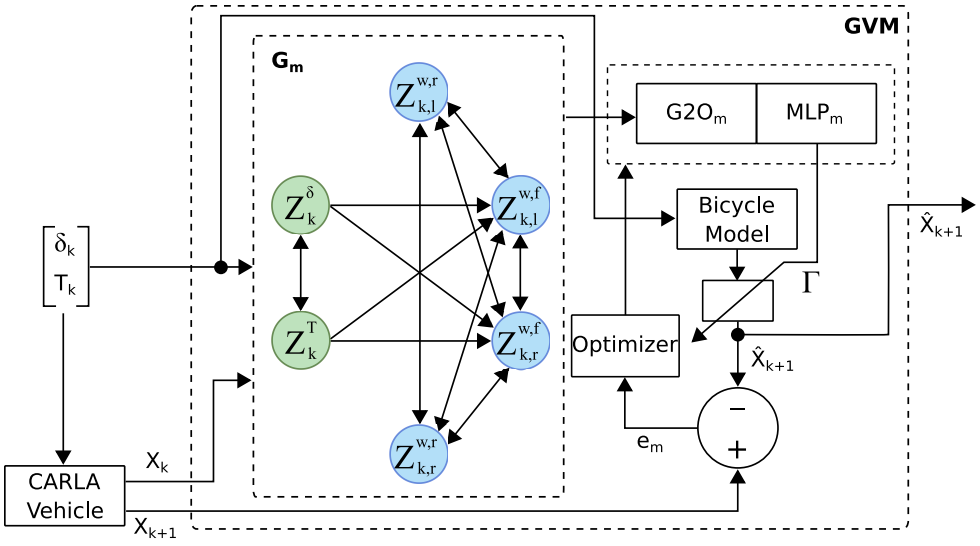


Figure 5.3: Architecture of Graph Vehicle Model to estimate $\hat{X}_{k+1}$ online based on the error e_m at each time step

thus, the edges between them are bidirectional. The same holds for the actuators. Nonetheless, it is evident that the actuators drive the wheels – both for steering and rotation. During implementation, the dynamics of the model is discretized as follows:

$$\hat{X}_{k+1} = X_k + \dot{\chi}_k t_s \tag{5.12}$$

where, X is the ground truth of the states obtained from vehicle sensors. The modeling error

$$e_m = X_{k+1} - \hat{X}_{k+1} \tag{5.13}$$

which is the difference between the estimated states and the ground truth at the $(k+1)^{\text{th}}$ time step is backpropagated into G2O_m and MLP_m to update their weights.

5.2.3 Graph-based Lateral Controller

The Graph-based Lateral Controller (GLC) proposed in this section primarily frames the "desired" transition from the current state to the desired state into a heterogeneous graph G_c as shown in

Figure 5.4. The features of the wheels at k^{th} step are the same used in equations (5.10) and (5.11). However, for the $(k+1)^{\text{th}}$ step, velocity is removed from the feature array. The removal of velocity from the desired nodes is intuitive since the task of the lateral controller is not to track velocity profile. The unidirectional edges from current to desired mimics the transition to the future, and thus, the sampling period is assigned as their edge features. The bidirectional edges between the wheels have their corresponding physical distances as the edge feature. As shown in Figure 5.4, G_c is fed into G2O_c to generate φ_c. Finally, the steering command is produced as follows:

$$\delta_k = \text{HardTanh}\left(\text{MLP}_c\left(\left(\varphi_c^{\text{desired}}\right)^f - \left(\varphi_c^{\text{current}}\right)^f\right)\right) \tag{5.14}$$

where, the subscript c refers to controller, $\varphi_c \in \mathbb{R}^{8\times M_c}$ (note that there are eight nodes in G_m), MLP has an input dimension $4M_c$, $\varphi_c^{\text{desired}}$ and $\varphi_c^{\text{current}}$ are the output node embedding of the desired and current nodes of G_c respectively, and,

$$\text{HardTanh}(\omega) = \begin{cases} 1.0, & \text{if } \sigma > 1.0 \\ -1.0, & \text{if } \sigma < -1.0 \\ 2.0\omega, & \text{otherwise} \end{cases}$$

oEquation (5.12) is used to estimate the states of the vehicle at $(k+1)$ using δ_k and compared with the desired states. The error is then backpropagated into G2O_c and MLP_c to update their weights. To calculate the error, only the position coordinates are used, i.e.,

$$e_c = Y_d - \hat{Y}_{k+1} = Y_d - D\hat{X}_{k+1} \tag{5.15}$$

where, Y_d is the desired position and $\hat{Y}$ is the estimated position with $D = \begin{bmatrix} I_{2\times 2}, & 0_{2\times 2} \end{bmatrix}$ and I being the identity matrix. Algorithm 5.1 outlines the pseudo code to obtain Y_d. $C_d(R)$ is an array of the cumulative distance of the reference path R given by positional coordinates, and the function "Frenet Coordinate" outputs the longitudinal Frenet coordinate with respect to the reference path.

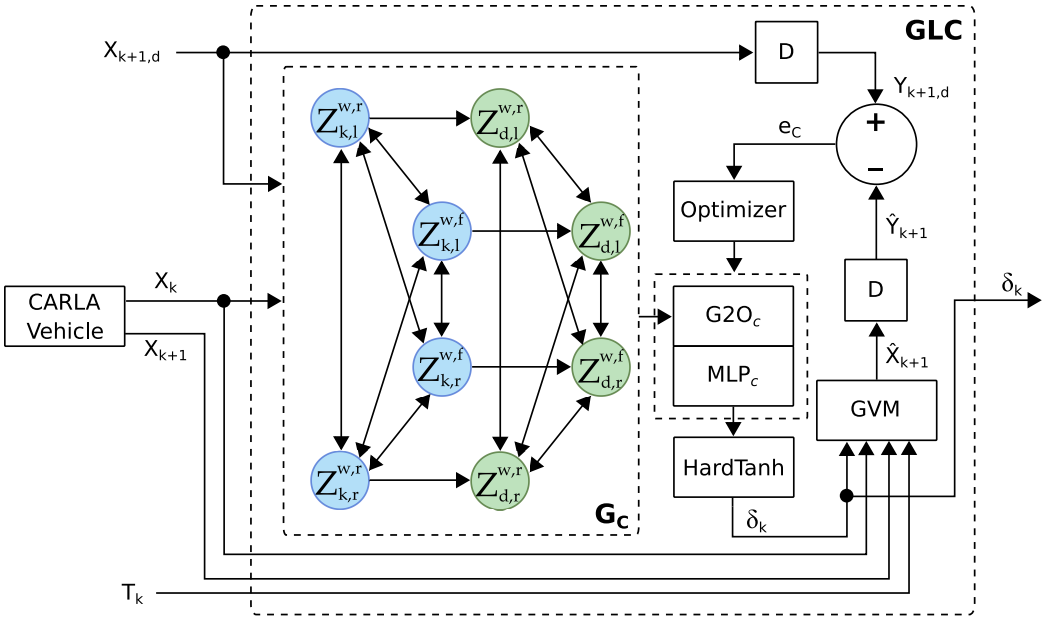


Figure 5.4: Graph-based Lateral Controller framework for online lateral control based on the error e_c at each time step

Algorithm 5.1 Pseudo code to obtain Y_d

```
Inputs: R, C_d(R), X_k
Do:
```

1 $s_k = \text{Frenet Coordinate}(R, X_k)$
2 $\text{indices} = \text{find}(C_d(R) > s_k)$
3 $Y_d = R[\text{indices}[1], :]$

It is important to note that this work is centered on the design of a lateral controller, and therefore, throttle is treated as a measured disturbance in the vehicle dynamics. A similar approach was used in the design of a lateral controller in [158].

5.3 Training Process

Prior to deploying the online learning of the lateral controller, GVM is pre-trained in the CARLA environment as shown in Figure 5.3. By passing pre-determined throttle and steering values, the output from the CARLA vehicle is utilized to train GVM on the fly. In practice, a driver would just

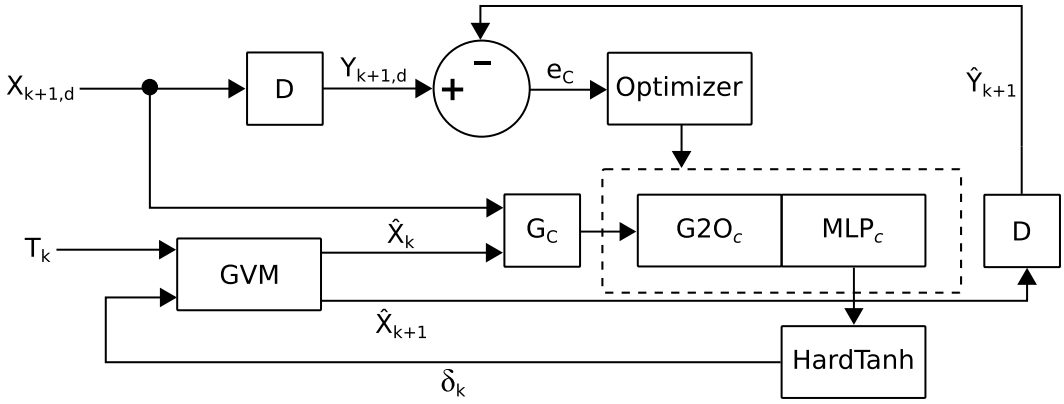


Figure 5.5: Pre-training process of the controller

have to drive the car making as much diverse maneuvers as possible and the network would learn in real time.

Furthermore, GLC is also pre-trained on the already trained GVM before the actual deployment in the CARLA simulator. The training process is similar to Figure 5.4, except that X_k is replaced by $\hat{X}_k$, directly obtained from GVM, to generate G_c as shown in Figure 5.5. The usefulness of this pre-training is evident in Figure 5.6 – with a pre-trained model on the CARLA simulator, the vehicle starts with impressive path tracking at the first learning epoch, unlike when an untrained model is used. Again, in practicality, this would mean that the car using a pre-trained GLC would be much safer at the very beginning of its learning curve. It should be noted that during the online learning, GVM can potentially be updated only when the estimation error is above a predefined threshold ε. Similarly, the online training of the GLC can be halted as long the performance is satisfactory. However, in this work, the online learning is always activated.

The loss function related to $e \in \mathbb{R}$ used to train both the GVM and the GLC is SmoothL1Loss:

$$\text{smooth}_{\text{L1}}(e) := \begin{cases} 0.5e^2, & \text{if} \quad |e| < 1 \\ |e| - 0.5, & \text{otherwise} \end{cases} \tag{5.16}$$

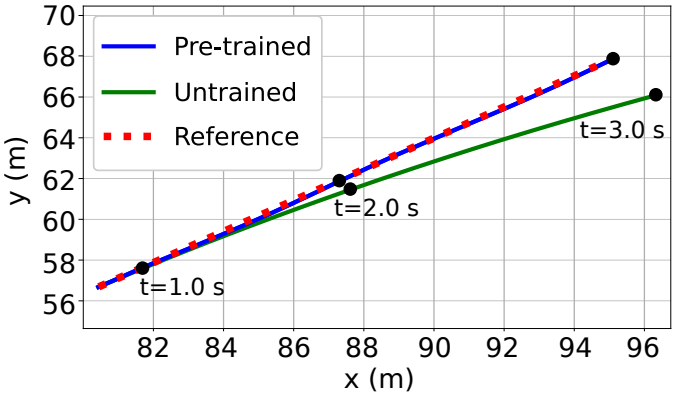


Figure 5.6: Performance comparison of the proposed controller in CARLA environment at the first learning epoch with a pre-trained model and an untrained model of the controller

5.4 Simulation Results

5.4.1 CARLA Environment Setup

CARLA [161] is an open-source autonomous car simulator that can simulate driving scenarios and can support multitude of ADS tasks [164]. Its high-fidelity environment allows to develop, test and validate ADS systems including controllers. In recent works [165, 166, 167, 168], CARLA has been widely used for different tasks. The platform has the OPENDrive [169] standalone mode that allows one to load an OpenDRIVE file and create a temporal 3D mesh that defines the road in a minimalistic manner.

In this work, CARLA version 0.9.14 has been used. At initialization, the road is loaded using OpenDRIVE format along with the ego. The center of the road along the path is set as the reference path. To obtain the reference trajectory, the CARLA auto-pilot mode was activated such that it follows the lane center at a maximum speed of $10\,\mathrm{m/s}$. The reference trajectory contains position, heading and velocity at each time-step, which is set at 0.1 second. CARLA APIs allow accessing all the states (e.g., position, heading, velocity) directly at each step. A longitudinal controller (PID) to follow the reference velocity, and a lateral controller (the proposed controller) to minimize lateral positional error, were implemented.

The first autonomous racing leagues took place in Abu Dhabi (circuit called Yas Marina, also

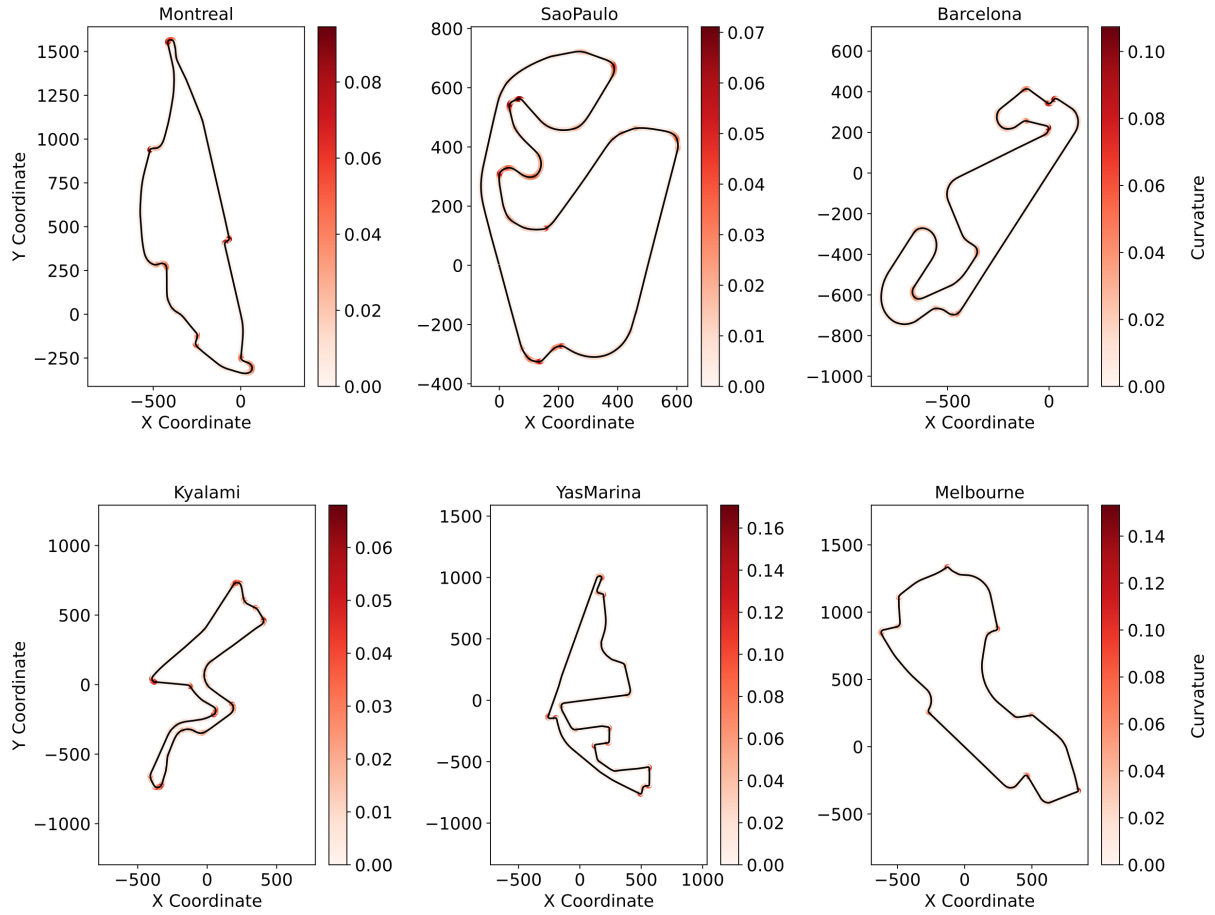


Figure 5.7: Six race tracks from six different continents along with curvature values along the race tracks

used in this work) in April 2024 with seven participating nations [170]. At the trials of the event during which no opponent was present on the track, the self-driving cars randomly curled and even turned into the walls [170] showing poor performances of the lateral component of the controllers. Therefore, for training and testing purposes in this work, six different Formula 1 race tracks from six different continents, as shown in Figure 5.7, were used. These tracks have several sharp turns, bends and straights making them practical choices to test any control algorithm for autonomous cars.

Table 5.1: MPC parameters and steering constraints used

Parameter	Value	Description
p	4	Prediction horizon
$\delta_{\min}$	-1.0 rad	Minimum steering
$\delta_{\max}$	1.0 rad	Maximum steering
$\Delta\delta_{\max}$	0.5 rad/s	Maximum steering velocity
t_s	0.1 s	Sampling period
R_d	1.0	Weight on input
Q	diag$\{$ 2.5, 2.5 $\}$	Weight on tracking error
Q_f	diag$\{$ 3.5, 3.5 $\}$	Weight on terminal condition

5.4.2 Baselines

5.4.2.1 Model Predictive Controller

A Linear Time-Varying MPC is used as the baseline lateral controller to compare the proposed controller against. Since the prediction model (see equation (5.2)) is non-linear, it is linearized at different operating points, i.e., it is updated recursively as the operating conditions change. The cost function to be minimized in the finite-horizon MPC problem is defined using weighted 2-norms as follows:

$$\begin{aligned} J = &\sum_{k=1}^{p-1} \|Y_{t+k} - R_{t+k}\|_Q^2 + \quad \left\|Y_{t+p} - R_{t+p}\right\|_{Q_f}^2 \\ &+ \sum_{k=1}^{p-1} \|\delta_{t+k}\|_{R_d}^2 + \sum_{k=1}^{p-1} \|\delta_{t+k} - \delta_{t+k-1}\|_{R_d}^2 \end{aligned} \tag{5.17}$$

where, the first term represents the tracking error of the position, the second term penalizes the error of the terminal condition, the third term penalizes high steering inputs, and the last term penalizes steering velocity. The MPC is programmed to generate only the steering command and a separate PID is used as the longitudinal controller in the baseline settings. The MPC parameters and constraints on the inputs used are listed in Table 5.1.

5.4.2.2 Stanley Lateral Controller

The Stanley lateral controller [138] uses position of the front axle to calculate the heading error (ψ) as well as the cross-track error (e_{ct}, the closest distance between the front axle and the reference trajectory). The steering angle (control law) is given as follows:

$$\delta = \psi + \arctan\frac{ke_{\text{ct}}}{v}, \tag{5.18}$$

where, k is a gain that determines the rate of error convergence and v is the velocity. The second term adjusts the steering in nonlinear proportion to e_{ct}, i.e., the steering response toward the trajectory is stronger if e_{ct} is larger. Detailed explanation on the Stanley lateral controller can be found in [138]. In this work, $k = 1.0$ has been used.

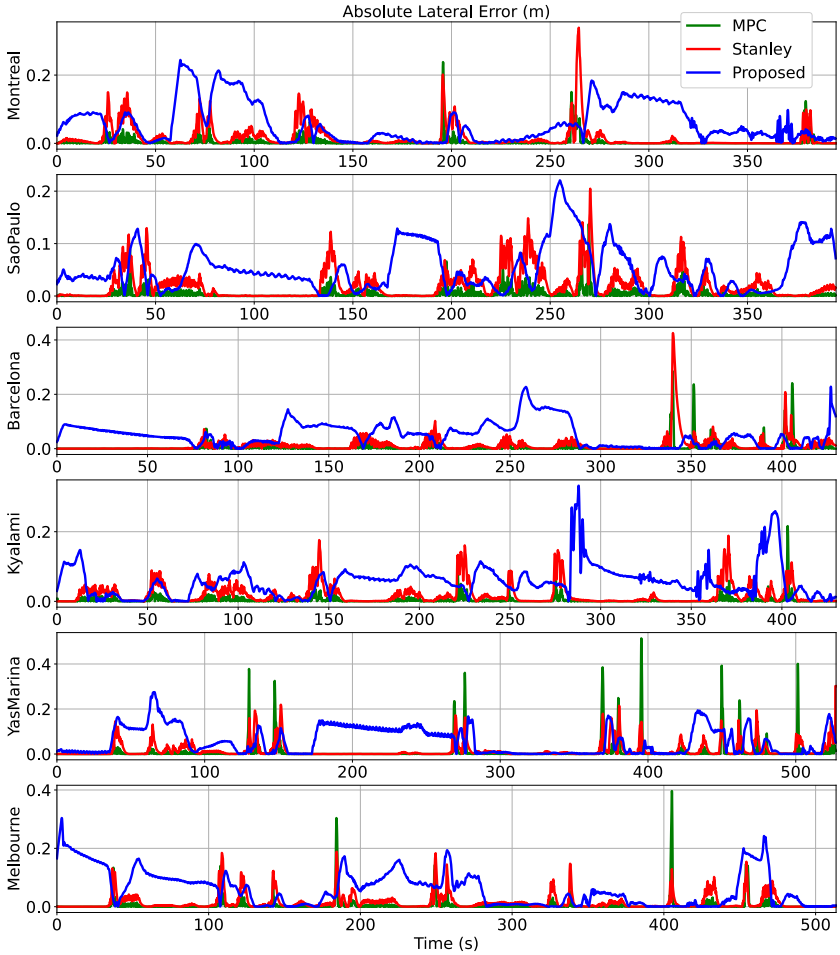


Figure 5.8: Absolute lateral error experienced by the controllers without perturbation

5.4.3 Result Analysis

The proposed controller GLC and the MPC are both simulated in the CARLA environment as described earlier with the goal to drive along the center lane. Maximum lateral error

$$\text{MLE} = \max_{i=1,\ldots,n} \left(|e_i|\right), \tag{5.19}$$

and the root mean square error

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2} \tag{5.20}$$

of the lateral position error are used to measure the performance of each controller, where e is the error signal for time $t_1, \ldots, t_n$. The tracks are segmented into two categories for better insight into performance measurement: straight section when the curvature is less than $0.02\,\text{m}^{-1}$, otherwise, turning section (see Figure 5.7). Furthermore, with the pre-trained controller discussed in section 5.3, two different tests – with and without perturbation – are carried out. In the latter test, a constant perturbation of ± 2.5 degrees is added to the steering command – such a disturbance mimics the misalignment of the wheels with the steering in the real world.

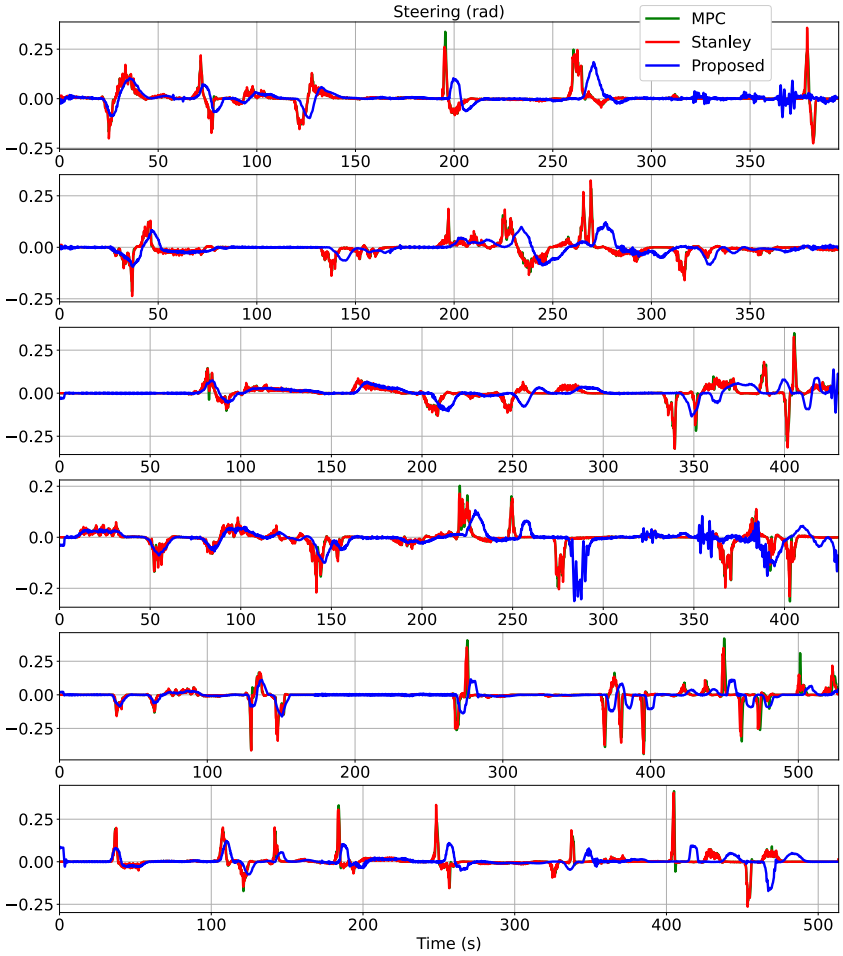


Figure 5.9: Steering commands generated by the controllers without perturbation

Table 5.2: Performance comparison for reference path tracking between GLC, MPC and Stanley for the six tracks without perturbation. The best result for each of the subcategories is in bold.

MLE (m)	**Straight**			**Turn**			**Overall**		
	GLC	**MPC**	**Stanley**	**GLC**	**MPC**	**Stanley**	**GLC**	**MPC**	**Stanley**
Montreal	0.24	**0.09**	0.33	**0.20**	0.24	0.34	**0.24**	**0.24**	0.34
Sao Paulo	0.22	**0.07**	0.21	0.21	**0.14**	0.18	0.22	**0.14**	0.21
Barcelona	0.23	**0.18**	0.42	**0.17**	0.29	0.43	**0.23**	0.29	0.43
Kyalami	0.27	**0.15**	0.18	0.33	0.22	**0.19**	0.33	0.22	**0.19**
Yas Marina	**0.28**	0.31	0.30	0.28	0.51	**0.22**	**0.28**	0.51	0.30
Melbourne	0.30	0.27	**0.18**	0.26	0.40	**0.19**	0.30	0.40	**0.19**
RMSE (m)	**Straight**			**Turn**			**Overall**		
	GLC	**MPC**	**Stanley**	**GLC**	**MPC**	**Stanley**	**GLC**	**MPC**	**Stanley**
Montreal	0.08	**0.01**	0.03	0.08	**0.07**	0.12	0.08	**0.02**	0.04
Sao Paulo	0.07	**0.01**	0.03	0.08	**0.06**	0.08	0.07	**0.01**	0.03
Barcelona	0.07	**0.01**	0.03	**0.07**	0.13	0.12	0.07	**0.02**	0.03
Kyalami	0.07	**0.01**	0.03	0.13	**0.06**	0.10	0.07	**0.01**	0.03
Yas Marina	0.08	**0.01**	0.03	**0.10**	0.18	**0.10**	0.08	**0.04**	**0.04**
Melbourne	0.08	**0.01**	0.03	0.12	0.13	**0.11**	0.09	**0.02**	0.03

Firstly, the controllers are tested without perturbation. Figure 5.8 shows the performance of the proposed controller in contrast to the baselines and the summary of the results are listed in Table 5.2. On straight paths, MPC has outperformed both GLC and Stanley in terms of both MLE and RMSE. At turns, GLC achieves better MLE in two of the six tracks, while performance of MPC and Stanley are better in one and three of six tracks, respectively. Overall, the RMSE values show that the performance of the baselines are better than the proposed controller without perturbation. However, this performance comes at the cost of comfort. Although the steering commands generated by the controllers, as shown in Figure 5.9, are within the steering limit of CARLA environment (i.e., 1 radian), Table 5.3 shows that the maximum steering velocity for all the six tracks generated by both MPC and Stanley is always above GLC. Thus, the proposed controller provides a more comfortable ride.

Table 5.3: Comparison of maximum steering velocity for reference path tracking between GLC, MPC and Stanley for the six tracks. The best result for each of the subcategories is in bold.

Maximum steering velocity (rad/s)	**No Perturbation**			**With Perturbation**		
	GLC	**MPC**	**Stanley**	**GLC**	**MPC**	**Stanley**
Montreal	**0.25**	0.61	1.08	**0.17**	1.01	1.41
Sao Paulo	**0.21**	0.71	1.34	**0.23**	0.68	1.35
Barcelona	**0.33**	0.62	0.66	**0.18**	1.34	1.34
Kyalami	**0.26**	0.61	0.80	**0.28**	0.73	1.03
Yas Marina	**0.12**	2.20	1.16	**0.26**	1.66	1.04
Melbourne	**0.22**	0.79	1.29	**0.22**	0.91	1.51

Table 5.4: RMSE for position estimation by GVM for the six tracks

	RMSE (m)
Montreal	0.021
Sao Paulo	0.020
Barcelona	0.018
Kyalami	0.018
Yas Marina	0.026
Melbourne	0.019

The performance of the proposed GVM plays a crucial role in the performance of GLC. Table 5.4 shows the accuracy of the proposed model in estimating the position of the ego. The RMSE values suggest high accuracy of the proposed vehicle model.

Next, the controllers are tested with perturbation. It can be seen from Figure 5.10 that the proposed controller GLC learned to attenuate the perturbation while MPC and Stanley failed to do so – a lateral error offset can be observed for both. In terms of both MLE and RMSE as can be seen in Table 5.5, the overall performance of GLC is superior to that of MPC and Stanley. Note that GLC still maintained a more comfortable ride even under perturbation (see Figure 5.11 and Table 5.3).

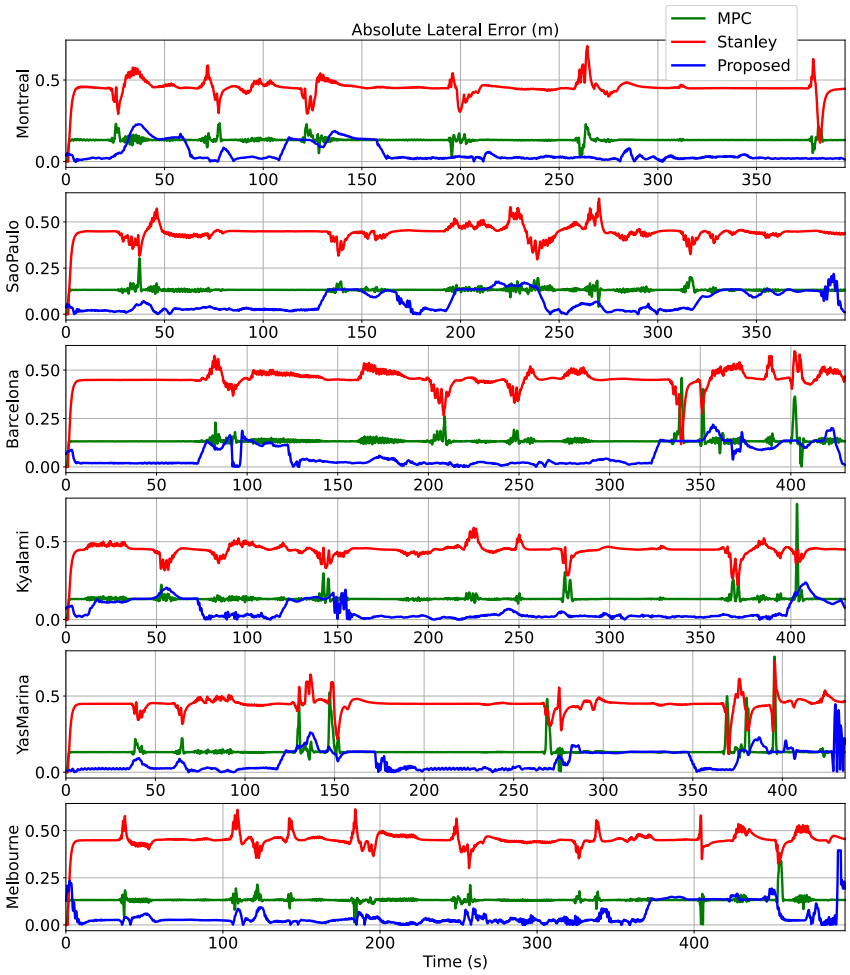


Figure 5.10: Absolute lateral error experienced by the controllers under perturbation

5.4.4 Comparative Analysis of Numerical Performance

Zhou et al. proposed an event-triggered MPC (eMPC) for autonomous path tracking [158]. Unlike the traditional time-triggered MPC (tMPC), eMPC solves the optimization problem only when an event is triggered. They also tested their proposed controller on CARLA but on a much simpler track (Figure 5 in [158]) and without any perturbation. The maximum speed was set at 30 km/h which is slightly less than the maximum speed used in our case (i.e., 10 m/s = 36 km/h). Their results show that tMPC has better performance but this performance comes at the cost of frequent triggering of the MPC – eMPC is triggered significantly less than tMPC for larger values of the event-trigger threshold, particularly, $\sigma = 0.03$. Overall, RMSE values obtained by tMPC are

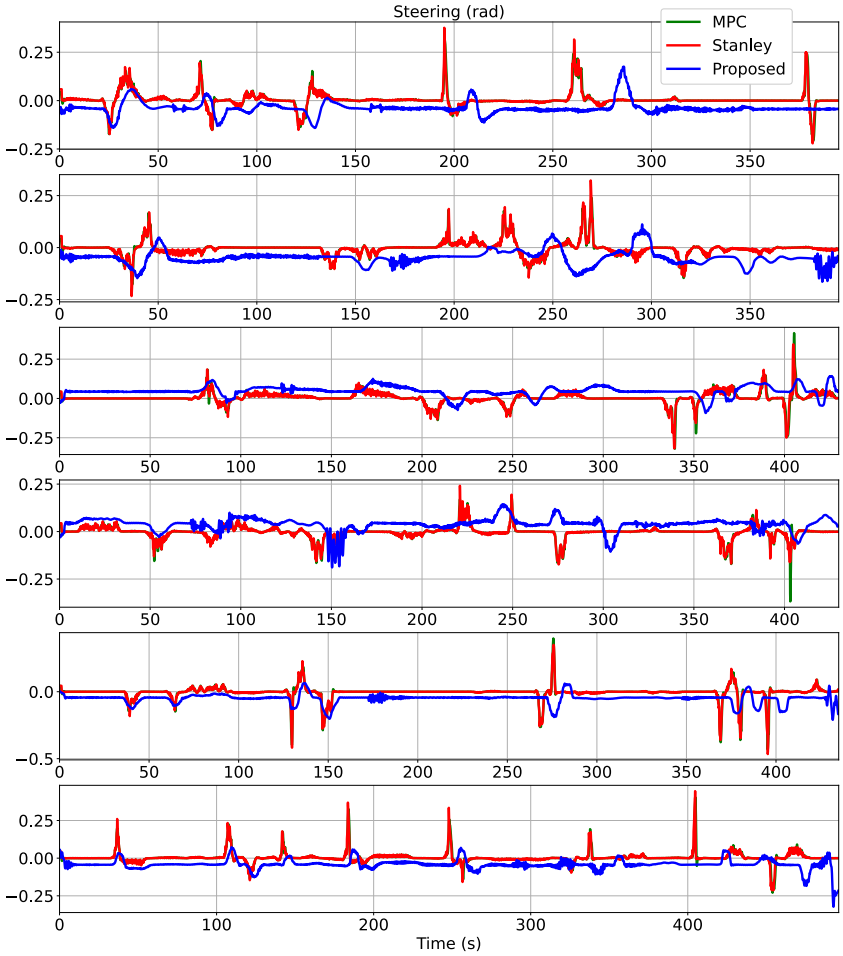


Figure 5.11: Steering commands generated by the controllers under perturbation

0.052 m at turns and 0.018 m at straights, which are comparable to the (worst) performance obtained by our proposed GLC, i.e., 0.13 m at turns (in Kyalami) and 0.08 m at straights (in Montreal, Yas Marina and Melbourne).

In [171], Artuñedo et al. implemented several classical control approaches for the lateral control problem in autonomous driving, including Linear Quadratic Regulators (LQR), PID and nonlinear MPC (NMPC). They also implemented more recent approaches – Model-Free Control (MFC) [172] developed in 2013, and Speed-Adaptive Model-Free Control (SAMFC) [173] developed in 2022. They have tested these controllers on several tracks with different maximum speeds under no perturbation. For fair comparison, only the results obtained from trajectory T1, which has much simpler turns (Figure 3(a) in [171]) in contrast to the racing tracks used to test GLC (5.7), is

Table 5.5: Performance comparison for reference path tracking between GLC, MPC and Stanley for the six tracks with perturbation. The best result for each of the subcategories is in bold.

MLE (m)	**Straight**			**Turn**			**Overall**		
	GLC	**MPC**	**Stanley**	**GLC**	**MPC**	**Stanley**	**GLC**	**MPC**	**Stanley**
Montreal	**0.23**	0.24	0.70	**0.23**	**0.23**	0.71	**0.23**	0.24	0.71
Sao Paulo	**0.18**	0.30	0.62	**0.22**	0.30	0.63	**0.22**	0.30	0.63
Barcelona	**0.21**	0.37	0.59	**0.22**	0.46	0.60	**0.22**	0.46	0.60
Kyalami	**0.24**	0.56	0.58	**0.24**	0.74	0.59	**0.24**	0.74	0.59
Yas Marina	0.45	**0.43**	0.64	**0.41**	0.76	0.73	**0.45**	0.76	0.73
Melbourne	0.40	**0.29**	0.61	0.40	**0.38**	0.61	0.40	**0.38**	0.61
RMSE (m)	**Straight**			**Turn**			**Overall**		
	GLC	**MPC**	**Stanley**	**GLC**	**MPC**	**Stanley**	**GLC**	**MPC**	**Stanley**
Montreal	**0.07**	0.13	0.45	**0.06**	0.15	0.49	**0.07**	0.13	0.45
Sao Paulo	**0.09**	0.13	0.44	**0.07**	0.15	0.51	**0.09**	0.13	0.45
Barcelona	**0.08**	0.14	0.46	**0.09**	0.25	0.45	**0.08**	0.14	0.46
Kyalami	**0.08**	0.14	0.45	**0.07**	0.27	0.40	**0.08**	0.14	0.45
Yas Marina	**0.09**	0.13	0.45	**0.14**	0.33	0.46	**0.10**	0.15	0.45
Melbourne	**0.0**7	0.13	0.45	**0.11**	0.18	0.50	**0.07**	0.13	0.45

mentioned here since the maximum speed used in this trajectory is 35 km/h. The mean values of MLE suggest that SAMFC has the best performance among the five tested controllers achieving 0.33 m. On the other hand, in terms of MLE, the best and the worst performances of GLC were 0.23 m (in Barcelona) and 0.33 m (in Kyalami), respectively.

A hierarchical dynamic drifting controller (HDDC) is proposed in [174] which includes a path tracking layer (for generating the desired states), a vehicle motion control layer (to integrate drifting and typical cornering control), and an actuator regulating layer. HDDC is integrated with both MPC and LQR separately and the performance of these two controllers is assessed on a single track. It should be noted that both the MPC-HDDC and LQR-HDDC controllers are coupled controllers, i.e., they generate both the longitudinal and the lateral commands. Overall, in terms of

MLE, MPC-HDDC has the best performance with a 0.34 m value, and in terms of RMSE, both the controllers achieve the same performance, i.e., 0.11 m. Both these values are higher compared to GLC's corresponding achieved (worst) values, i.e., 0.33 m for MLE (in Kyalami), and 0.09 m for RMSE (in Melbourne). Even though the maximum speed used in [174] is lower than the maximum speed used in this work, it appears that the absence of coupling between the lateral and the longitudinal aspects in GLC gives it an edge over HDDC.

Wasala et al. used separate reinforcement learning (RL) algorithms for longitudinal and lateral controls [153]. They tested their method with high speeds (80 km/h and 200 km/h) in simulation, which is higher than the speed in this work, and with low speeds (between 10 km/h and 40 km/h) on a real vehicle. They also used race tracks to assess the performance of their proposed controllers. For the simulations on unseen tracks with a maximum speed of 80 km/h, with Jaguar-X (car model), the lateral controller was able to achieve approximately 0.20 m for MLE and an average value of 0.32 m for RMSE. The MLE value clearly suggests better performance of their proposed controller in contrast to the proposed GLC in this work. However, the RMSE value obtained in [153] is more than three times of that obtained by GLC – this is understandable given the high speed profile of their test bed compared to the one in this work.

Combining the Pure Pursuit controller with a PID and then using a customized RL model to trade off between the two, Shane et al. were able to successfully track different paths with high speeds (up to 80 km/h) [175]. However, they also tested their technique using a lower speed of 35 km/h, which is similar to this work, and therefore experimental results related to that particular speed will be mentioned here. The experiments were carried out on four types of tracks (S-shape for training, and, T-shape, U-shape and O-shape for testing). In the training, the value of MLE obtained was 0.37 m, while in training the values were 0.25 m, 0.37 m and 0.38 m for T-shape, U-shape and O-shape, respectively. Although the average of these values is higher than that obtained by GLC, these are comparable performance results.

5.5 Conclusion

This chapter introduces an innovative approach of learning a vehicle model followed by a novel online lateral controller for autonomous driving. Heterogeneous graphs are used to represent both the components and then fed through their corresponding G2O. The vehicle learning model GVM utilizes existing knowledge of a vehicle, whereas a neural network would have to learn from scratch. Conversely, unlike a physics-based model, GVM can capture unidentified dynamics. Combining these strengths, GVM efficiently learned the vehicle dynamics online. The lateral controller GLC also learns online to effectively maintain the ego on its reference path; MPC and Stanley controller, on the other hand, cannot learn online. Without perturbation, MPC and Stanley does better tracking than GLC. However, the proposed controller gives a more comfortable ride. When the ego is perturbed, GLC outperforms both MPC and Stanley by mitigating the perturbation while maintaining satisfactory tracking performance as well as comfort. The efficacy and practicality of the proposed techniques have been validated through comprehensive simulations in CARLA. In this work, the proposed lateral controller was tested for a low velocity and a PID controller was used as a longitudinal controller for tracking that velocity. However, at higher velocities, the coupling between the two controllers become stronger, and thus, a unified controller is more suitable. For future work, we want to investigate a graph-based controller that can generate both the steering and the throttle commands.

Chapter 6

Conclusion

The fast progress of self-driving technology requires the creation of reliable, flexible, and efficient systems to guarantee safe, comfortable, and road regulation-compliant autonomous vehicles. The main focus of this thesis is on three crucial elements of autonomous driving systems: predicting trajectories, planning trajectories, and controlling the vehicle.

6.1 Research Summary

The research work presented in this thesis addresses some of the core problems in autonomous driving. Namely, it proposes novel techniques for the prediction of the actors' trajectories, the computation of a planned trajectory for the ego, and the lateral control of the ego car to track its planned trajectory. This work contributes significantly to the current state of autonomous driving technology by introducing innovative methodologies using graph learning techniques.

6.1.1 Trajectory Prediction with AiGem

Trajectory prediction plays a fundamental role in the decision-making and safety of autonomous vehicles. In this context, a deep learning model called AiGem (Agent-Interaction Graph Embedding) is proposed. AiGem employs a creative approach to build a heterogeneous graph using

historical data, effectively capturing the interactions between agents through spatial and temporal edges. The use of this graph representation allows for a thorough modeling of agent interactions as time progresses. Within the AiGem framework, there is a graph encoder network that generates embeddings for target actors at the current timestamp. Next, the embeddings are inputted into a sequential GRU decoder network. Utilizing the decoded states from the GRU, the output network employs an MLP to predict future trajectories. The NGSIM datasets are used to evaluate the performance of the model, showing that AiGem achieves similar accuracy to the most advanced prediction algorithms for shorter prediction timeframes. AiGem stands out by outperforming all baseline models in predicting events with longer horizons of four and five seconds.

The trajectory prediction module of AiGem presents notable advancements in autonomous driving, enhancing safety, improving decision-making, and decreasing collisions. The long-term trajectory prediction capability of AiGem empowers autonomous vehicles to anticipate potential hazards in advance and make well-informed decisions. By employing a heterogeneous graph, the model successfully captures the complex dynamics of agent interactions, leading to more accurate predictions of future movements. The comprehensive understanding of interactions, although not readily apparent, plays a pivotal role in averting hazardous situations by facilitating appropriate responses from the vehicle. As a result, AiGem's predictive capabilities play a crucial role in minimizing collisions by enabling autonomous systems to anticipate and respond to potential threats, thus ensuring safer and more dependable autonomous driving.

The frequency at which autonomous cars make trajectory predictions is very important, as it is greatly affected by the prediction horizon and the necessary level of accuracy. In order to navigate through immediate obstacles, predictions for shorter horizon scenarios must be made more frequently to enable swift responses to dynamic environmental changes. Implementing high-frequency updates enables the system to maintain more control and adjust promptly to abrupt changes in the surrounding conditions, such as the sudden emergence of a pedestrian or an unexpected maneuver by another vehicle. In comparison, predictions that extend several seconds into the future can be generated less frequently. This is because the focus shifts from immedi-

ate reactions to strategic planning. Predictions with longer horizons prioritize broader trends and can accommodate minor inaccuracies, as subsequent short-term predictions allow for immediate corrective actions.

6.1.2 Online Spatial-Temporal Graph Trajectory Planner

Autonomous driving also involves careful trajectory planning, which guarantees safe and comfortable vehicle navigation while considering kinematic and road constraints. An innovative online STG trajectory planner is developed that combines autonomous vehicles, surrounding vehicles, and virtual road nodes to build heterogeneous graphs. By using a graph-based representation, it becomes simpler to model the driving environment. To generate the desired future trajectory, the planner employs a sequential neural network architecture to analyze these graphs, effectively capturing familiar driving behaviors like lane-keeping, lane-changing, car-following, and speed keeping. To guarantee effective learning of the network, potential functions that consider safety and velocity constraints are integrated. Moreover, a basic behavioral layer is introduced to incorporate kinematic constraints into the planner. Testing on three complex driving tasks validates the effectiveness of the proposed planner, with its performance compared with two frequently used techniques. The results show that the planner effectively creates viable paths for all tasks, striking a better balance between safety and efficiency when compared to the baselines.

The STG trajectory planner holds several noteworthy implications for autonomous driving in real-life scenarios. First, the incorporation of a spatial-temporal graph planner that represents the environment using a heterogeneous graph facilitates more precise and contextually aware trajectory planning. This allows autonomous vehicles to navigate complex driving situations, such as changing lanes, following other cars, and maintaining speed, with enhanced accuracy. By incorporating safety and velocity constraints, the planner guarantees the generation of paths that are not only efficient but also prioritize safety, thereby reducing the chances of accidents. Moreover, the incorporation of a basic behavioral layer to consider kinematic limitations allows the planner to generate trajectories that are not only feasible but also comfortable, thereby enhancing the passenger's ex-

perience. Overall, the STG trajectory planner improves the balance between safety, efficiency, and comfort in autonomous driving, leading to more reliable and user-friendly autonomous vehicles.

6.1.3 Adaptive Vehicle Modeling and Online Lateral Control

Autonomous driving controllers must be adaptable to reliably perform in different conditions. In this thesis, a novel technique for online learning in vehicle modeling and lateral control is introduced. This approach involves the use of heterogeneous graphs and GNNs. The development includes two key components: the GVM and the GLC. By leveraging existing vehicle knowledge and established physical constraints, the GVM can effectively learn vehicle dynamics in real-time and capture previously unidentified dynamics that traditional physics-based models may overlook. With this capability for online learning, the model maintains accuracy regardless of changing conditions. The GLC, which is designed to keep the vehicle centered on its intended path, also has the ability to learn in real-time. While a traditional MPC may excel in terms of tracking accuracy under normal conditions, the GLC prioritizes a comfortable ride quality. In addition, the GLC stands out as the better choice compared to the MPC and the Stanley, as it excels in effectively managing disturbances and maintaining a high level of tracking and comfort when the vehicle encounters perturbations. Thorough simulations on the CARLA platform have verified the practicality and effectiveness of the GLC.

The integration of an online learning technique into the vehicle lateral control module brings about notable progress in autonomous driving. Utilizing this approach, the vehicle can adapt in real-time to changes in its dynamics, resulting in consistent performance even in challenging conditions that conventional models may have difficulty with. In addition to maintaining the vehicle's intended course, the GLC enhances ride comfort, making it well-suited for practical driving scenarios. The ability to manage perturbations underscores its potential to enhance safety and passenger comfort when compared to conventional controllers like MPC and Stanley. This technique indicates that autonomous vehicles that incorporate such a control system will show improved resilience, comfort, and reliability, equipping them to successfully navigate the complexities of

real-world driving scenarios.

Collectively, these contributions provide a comprehensive solution to some of the most pressing challenges in autonomous driving, paving the way for safer and more efficient transportation systems.

6.2 Future works

The findings of this thesis suggest several areas that would benefit from further investigation.

6.2.1 A Single-model for Trajectory Prediction

Separate models were used for different prediction horizons in Chapter 3. While this practice is common, opting for a single-model approach for multi-horizon forecasting [176] is highly desirable as it can significantly improve efficient usage of computational resources [177] (including memory) and can be beneficial when scaling up [178], consequently, reduce maintenance complexity. Moving forward, I plan to explore and improve the model architecture, aiming to attain a single-model approach that yields high accuracy in multi-horizon predictions.

6.2.2 Trajectory Planner for Connected Vehicles

As a result of the significant enhancements in data transfer speeds and connectivity made possible by 5G/6G technology [179, 180, 181], inevitably, autonomous vehicles will engage in real-time communication and collaborative planning [182, 183, 184]. In the future, my strategy involves expanding the implementation of the proposed method to encompass planning for V2V (Vehicle-to-Vehicle) communication architectures. The concept was evaluated in an initial simulation and showed promising outcomes for vehicle platooning (i.e., group of vehicles driving together) via V2V communication. However, the architectural design needs to be enhanced to handle complex scenarios beyond platooning.

6.2.3 Coupled Lateral and Longitudinal Controller

The coupling between the lateral and the longitudinal controllers increases with increasing velocity [185]. Moreover, the absence of coupled controllers disregards the fundamental interdependence of tire forces between longitudinal and lateral motion from a vehicle dynamics point of view [186]. Despite previous research on coupled controllers [186, 187, 188], the exploration of online learning for coupled controllers has been largely overlooked, suggesting the possibility of substantial progress. For future, my plan is to extend the capabilities of the existing GLC by constructing a unified lateral and longitudinal controller capable of generating both the steering and the throttle commands through online learning.

6.2.4 Interpretability of Models

For future, I plan to offer a framework for comprehending the information processing and action-taking of autonomous agents in pursuit of their objectives. Models of this nature play a vital role in understanding the mechanisms that drive the planning and decision-making processes within self-driving cars. For example, attention mechanisms allow to investigate the priorities put on different objects by the model. This will result in the possibility of parameterizing the model in a manner that accommodates human-like choices. For example, prioritizing pedestrians over cars given their increased vulnerability. Similarly, by quantifying the interactions among the vehicles, justifications for the trajectory choices made by the vehicles can be showcased. Such explanations will enhance social acceptability of self-driving cars.

Appendix A

Routine for STG planner

Routine A.1 shows the steps to obtain the kinematic constraints. Lines 1-6 imply that in the absence of a lead actor or a rear actor within the safety gap, the kinematic constraints should remain the same, which were determined prioritizing comfort. However, if there is a lead actor within the safety gap (line 7), then the maximum deceleration rate $\ddot{s}_{\text{dec,max}}$ is doubled (line 8) and the maximum velocity $\dot{s}_{\text{max}}$ is constrained to lead actor's velocity v_{lead} (line 9). If the task is FSPS, the recommended speed v_{rec} is set to v_{lead} (line 11). Similarly, if there is a rear actor within the safety gap (line 12), the maximum acceleration rate $\ddot{s}_{\text{acc,max}}$ is doubled (line 13) and the minimum velocity $\dot{s}_{\text{min}}$ is constrained to rear actor's velocity v_{rear} (line 14). With FSPS, the recommended velocity v_{ref} is set to $\dot{s}_{\text{min}}$ if $v_{\text{ref}} < \dot{s}_{\text{min}}$ (line 16). In the event that there is both a lead and a rear actor within the safety gap (line 17), the lateral acceleration $\ddot{d}_{\text{max}}$ is doubled (line 18). Furthermore, the maximum velocity is adjusted in the event that the rear actor is moving faster than the lead actor (lines 19-20).

It is important to note that doubling both the lateral and longitudinal accelerations from the already given acceleration constraints is a heuristic for safety – the ego should be able to speed up its way out of danger in the event a lead or a rear actor breaches the safety gap without losing control. For example, with a higher lateral acceleration in effect, the ego can swerve away faster from the near-colliding vehicle, and this behavior is common in human drivers [189].

Algorithm A.1 Pseudo code to obtain kinematic constraints

```
function Kinematic Constraints
Inputs:
    s_lead, s_rear, v_lead, v_rear, a_max,long, v_max,
    v_min, a_max,lat, v_rec, s_safe, DTT, FSPS
Do:
1       s̈_dec,max = a_max,long
2       s̈_acc,max = a_max,long
3       ṡ_max = v_max
4       ṡ_min = v_min
5       d̈_max = a_max,lat

6   if  s_lead < s_safe
7       s̈_dec,max = 2a_max,long
8       ṡ_max = v_lead
9       if FSPS
10          v_rec = v_lead

11  if  s_rear < s_safe
12      s̈_acc,max = 2a_max,long
13      ṡ_min = v_rear
14      if FSPS and v_rec < ṡ_min
15          v_rec = ṡ_min

16  if  s_lead < s_safe and s_rear < s_safe
17      d̈_max = 2a_max,lat
18      if ṡ_min > ṡ_max
19          ṡ_max = ṡ_min
Return:
    if DTT: s̈_dec,max, s̈_acc,max, ṡ_max, ṡ_min, d̈_max
    if FSPS: s̈_dec,max, s̈_acc,max, d̈_max, v_rec
```

References

[1] International Journal of Innovation and Economic Development, "Research in autonomous driving - a historic bibliometric view of the research development in autonomous driving," *[Online]. Available: https://researchleap.com/research-in-autonomous-driving-a-historic-bibliometric-view-of-the-research-development-in-autonomous-driving*, Last Accessed: July 2024.

[2] United Nations - Department of Economic and Social Affairs, "The 17 goals," *[Online]. Available: https://sdgs.un.org/goals*.

[3] ——, "Sustainable transport," *[Online]. Available: https://sdgs.un.org/topics/sustainable-transport*.

[4] K. Vincent, F. Bouali, O. Haas, J. Christensen, C. Bastien, H. Davies, H. Taghavifar, A. Diederich, and A. Harrison, "How autonomous vehicles can contribute to emission reductions, fuel economy improvements and safety?" in *Alternative Fuels and Advanced Vehicle Technologies for Improved Environmental Performance*. Elsevier, 2022, pp. 711–741.

[5] M. Igini, "Environmental pros and cons of self-driving cars," *[Online]. Available: https://earth.org/pros-and-cons-of-self-driving-cars*, Last Accessed: July 2024. [Online]. Available: https://earth.org/pros-and-cons-of-self-driving-cars/

[6] United Nations, "Fact sheet: Climate change," *Sustainable Transport Conference*, 2021.

[7] E. Ericsson, "Independent driving pattern factors and their influence on fuel-use and exhaust emission factors," *Transportation Research Part D: Transport and Environment*, vol. 6, no. 5, pp. 325–345, 2001.

[8] G. H. Shahariar, T. A. Bodisco, A. Zare, M. Sajjad, M. I. Jahirul, T. C. Van, H. Bartlett, Z. Ristovski, and R. J. Brown, "Impact of driving style and traffic condition on emissions and fuel consumption during real-world transient operation," *Fuel*, vol. 319, p. 123874, 2022.

[9] F. A. Armah, D. O. Yawson, and A. A. Pappoe, "A systems dynamics approach to explore traffic congestion and air pollution link in the city of accra, ghana," *Sustainability*, vol. 2, no. 1, pp. 252–265, 2010.

[10] M. J. Thornbush and M. J. Thornbush, "Reduced traffic congestion and air pollution," *Vehicular Air Pollution and Urban Sustainability: An Assessment from Central Oxford, UK*, pp. 11–23, 2015.

[11] S. Shaheen, A. Cohen, B. Yelchuru, S. Sarkhili, B. A. Hamilton *et al.*, "Mobility on demand operational concept report," 2017.

[12] M. Pavone, "Autonomous mobility-on-demand systems for future urban mobility," in *Autonomes Fahren*. Springer, 2015, pp. 399–416.

[13] B. E. Dicianno, S. Sivakanthan, S. A. Sundaram, S. Satpute, H. Kulich, E. Powers, N. Deepak, R. Russell, R. Cooper, and R. A. Cooper, "Systematic review: Automated vehicles and services for people with disabilities," *Neuroscience Letters*, vol. 761, p. 136103, 2021.

[14] United Nations - Department of Economic and Social Affairs, "United to achieve the sdgs for, with, and by persons with disabilities," *[Online]. Available: https://www.un.org/en/desa/united-achieve-sdgs-and-persons-disabilities*.

[15] J. Hudson, M. Orviska, and J. Hunady, "Peopleâs attitudes to autonomous vehicles," *Transportation research part A: policy and practice*, vol. 121, pp. 164–176, 2019.

[16] F. Rosique, P. J. Navarro, C. Fernández, and A. Padilla, "A systematic review of perception system and simulators for autonomous vehicles research," *Sensors*, vol. 19, no. 3, p. 648, 2019.

[17] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 187–210, 2018.

[18] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li, "End-to-end autonomous driving: Challenges and frontiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[19] P. S. Chib and P. Singh, "Recent advancements in end-to-end autonomous driving using deep learning: A survey," *IEEE Transactions on Intelligent Vehicles*, 2023.

[20] J. Araluce, L. M. Bergasa, M. Ocaña, Á. Llamazares, and E. López-Guillén, "Leveraging driver attention for an end-to-end explainable decision-making from frontal images," *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[21] S. Atakishiyev, "Development of explainable artificial intelligence approaches for autonomous vehicles," PhD thesis, University of Alberta, 2024.

[22] H. Arai, K. Miwa, K. Sasaki, Y. Yamaguchi, K. Watanabe, S. Aoki, and I. Yamamoto, "Covla: Comprehensive vision-language-action dataset for autonomous driving," *arXiv preprint arXiv:2408.10845*, 2024.

[23] J. Dong, S. Chen, M. Miralinaghi, T. Chen, P. Li, and S. Labi, "Why did the ai make that decision? towards an explainable artificial intelligence (xai) for autonomous driving systems," *Transportation research part C: emerging technologies*, vol. 156, p. 104358, 2023.

[24] L. Cultrera, L. Seidenari, F. Becattini, P. Pala, and A. Del Bimbo, "Explaining autonomous driving by learning end-to-end visual attention," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 340–341.

[25] K. Othman, "Public acceptance and perception of autonomous vehicles: a comprehensive review," *AI and Ethics*, vol. 1, no. 3, pp. 355–387, 2021.

[26] T. Pham, M. Maghoumi, W. Jiang, B. S. S. Jujjavarapu, M. Sajjadi, X. Liu, H.-C. Lin, B.-J. Chen, G. Truong, C. Fang *et al.*, "Nvautonet: Fast and accurate 360deg 3d visual perception for self driving," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 7376–7385.

[27] S. Duan, X. Bai, Q. Shi, W. Li, and A. Zhu, "Uncertainty evaluation for autonomous vehicles: A case study of aeb system," *Automotive Innovation*, pp. 1–14, 2024.

[28] Y. Wu, D. Lian, Y. Xu, L. Wu, and E. Chen, "Graph convolutional networks with markov random field reasoning for social spammer detection," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 01, 2020, pp. 1054–1061.

[29] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur, "Protein interface prediction using graph convolutional networks," *Advances in neural information processing systems*, vol. 30, 2017.

[30] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[31] E. Meyer, M. Brenner, B. Zhang, M. Schickert, B. Musani, and M. Althoff, "Geometric deep learning for autonomous driving: Unlocking the power of graph neural networks with commonroad-geometric," in *2023 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2023, pp. 1–8.

[32] Y. Zhang, D. Qian, D. Li, Y. Pan, Y. Chen, Z. Liang, Z. Zhang, S. Zhang, H. Li, M. Fu *et al.*, "Graphad: Interaction scene graph for end-to-end autonomous driving," *arXiv preprint arXiv:2403.19098*, 2024.

[33] P. Cai, H. Wang, Y. Sun, and M. Liu, "Dignet: Learning scalable self-driving policies for generic traffic scenarios with graph neural networks," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8979–8984.

[34] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[35] Y. Huang, J. Du, Z. Yang, Z. Zhou, L. Zhang, and H. Chen, "A survey on trajectory-prediction methods for autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 652–674, 2022.

[36] O. Sharma, N. C. Sahoo, and N. B. Puhan, "Recent advances in motion and behavior planning techniques for software architecture of autonomous vehicles: A state-of-the-art survey," *Engineering applications of artificial intelligence*, vol. 101, p. 104211, 2021.

[37] Q. Yao, Y. Tian, Q. Wang, and S. Wang, "Control strategies on path tracking for autonomous vehicle: State of the art and future challenges," *IEEE Access*, vol. 8, pp. 161 211–161 222, 2020.

[38] J. Samiuddin, B. Boulet, and D. Wu, "Trajectory prediction for autonomous driving using agent-interaction graph embedding," in *[Accepted] 2024 IEEE 27th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2024.

[39] ——, "An online spatial-temporal graph trajectory planner for autonomous vehicles," *IEEE Transactions on Intelligent Vehicles*, 2024.

[40] ——, "An online self-learning graph-based lateral controller for self-driving cars," *IEEE Transactions on Intelligent Vehicles*, 2024.

[41] J. Leskovec, "CS224W: Machine learning with Graphs, Stanford University," *[Online]. Available: http://web.stanford.edu/class/cs224w*, Last Accessed: July 2023.

[42] PyTorch Geometric, "Heterogeneous graph learning," *[Online]. Available: https://pytorch-geometric.readthedocs.io/en/latest/notes/heterogeneous.html*, Last Accessed: April 2024.

[43] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[44] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[45] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[46] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *2010 IEEE international conference on robotics and automation*. IEEE, 2010, pp. 987–993.

[47] E. S. Meadows and J. B. Rawlings, *Nonlinear Process Control, Chapter 5*. Prentice Hall, 1997.

[48] M. Fink, "Implementation of linear model predictive control–tutorial," *arXiv preprint arXiv:2109.11986*, 2021.

[49] X. Li, Z. Sun, D. Cao, Z. He, and Q. Zhu, "Real-time trajectory planning for autonomous urban driving: Framework, algorithms, and verifications," *IEEE/ASME Transactions on mechatronics*, vol. 21, no. 2, pp. 740–753, 2015.

[50] S. Bae, D. Saxena, A. Nakhaei, C. Choi, K. Fujimura, and S. Moura, "Cooperation-aware lane change maneuver in dense traffic based on model predictive control with recurrent

neural network," in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 1209–1216.

[51] L. Lin, W. Li, H. Bi, and L. Qin, "Vehicle trajectory prediction using lstms with spatial–temporal attention mechanisms," *IEEE Intelligent Transportation Systems Magazine*, vol. 14, no. 2, pp. 197–208, 2021.

[52] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff, "Covernet: Multimodal behavior prediction using trajectory sets," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 14 074–14 083.

[53] V. Katariya, M. Baharani, N. Morris, O. Shoghli, and H. Tabkhi, "Deeptrack: Lightweight deep learning for vehicle trajectory prediction in highways," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 18 927–18 936, 2022.

[54] B. Mourllion, D. Gruyer, A. Lambert, and S. Glaser, "Kalman filters predictive steps comparison for vehicle localization," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 565–571.

[55] Z. Sheng, S. Xue, Y. Xu, and D. Li, "Real-time queue length estimation with trajectory reconstruction using surveillance data," in *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, 2020, pp. 124–129.

[56] A. Polychronopoulos, M. Tsogas, A. J. Amditis, and L. Andreone, "Sensor fusion for predicting vehicles' path for collision avoidance systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 3, pp. 549–562, 2007.

[57] M. Brännström, E. Coelingh, and J. Sjöberg, "Model-based threat assessment for avoiding arbitrary vehicle collisions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, pp. 658–669, 2010.

[58] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *ROBOMECH journal*, vol. 1, pp. 1–14, 2014.

[59] C. Ju, Z. Wang, C. Long, X. Zhang, and D. E. Chang, "Interaction-aware kalman neural networks for trajectory prediction," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1793–1800.

[60] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 1468–1476.

[61] A. Zyner, S. Worrall, J. Ward, and E. Nebot, "Long short term memory for driver intent prediction," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 1484–1489.

[62] A. Zyner, S. Worrall, and E. Nebot, "A recurrent neural network solution for predicting driver intention at unsignalized intersections," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1759–1764, 2018.

[63] ——, "Naturalistic driver intention and path prediction using recurrent neural networks," *IEEE transactions on intelligent transportation systems*, vol. 21, no. 4, pp. 1584–1594, 2019.

[64] A. Kawasaki and A. Seki, "Multimodal trajectory predictions for urban environments using geometric relationships between a vehicle and lanes," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9203–9209.

[65] N. Deo and M. M. Trivedi, "Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms," in *2018 IEEE intelligent vehicles symposium (IV)*. IEEE, 2018, pp. 1179–1184.

[66] S. H. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi, "Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture," in *2018 IEEE intelligent vehicles symposium (IV)*. IEEE, 2018, pp. 1672–1678.

[67] L. Xin, P. Wang, C.-Y. Chan, J. Chen, S. E. Li, and B. Cheng, "Intention-aware long horizon trajectory prediction of surrounding vehicles using dual lstm networks," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 1441–1446.

[68] T. Zhang, W. Song, M. Fu, Y. Yang, and M. Wang, "Vehicle motion prediction at intersections based on the turning intention and prior trajectories model," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 10, pp. 1657–1666, 2021.

[69] N. Nikhil and B. Tran Morris, "Convolutional neural network for trajectory prediction," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 0–0.

[70] J. Strohbeck, V. Belagiannis, J. Müller, M. Schreiber, M. Herrmann, D. Wolf, and M. Buchholz, "Multiple trajectory prediction with deep temporal and spatial convolutional neural networks," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 1992–1998.

[71] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "Home: Heatmap output for future motion estimation," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2021, pp. 500–507.

[72] A. M. Braşoveanu and R. Andonie, "Visualizing transformers for nlp: a brief survey," in *2020 24th International Conference Information Visualisation (IV)*. IEEE, 2020, pp. 270–279.

[73] A. Quintanar, D. Fernández-Llorca, I. Parra, R. Izquierdo, and M. Sotelo, "Predicting vehicles trajectories in urban scenarios with transformer networks and augmented information," in *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2021, pp. 1051–1056.

[74] Y. Liu, J. Zhang, L. Fang, Q. Jiang, and B. Zhou, "Multimodal motion prediction with stacked transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7577–7586.

[75] W. Chen, F. Wang, and H. Sun, "S2tnet: Spatio-temporal transformer networks for trajectory prediction in autonomous driving," in *Asian conference on machine learning*. PMLR, 2021, pp. 454–469.

[76] Z. Huang, X. Mo, and C. Lv, "Multi-modal motion prediction with transformer-based neural network for autonomous driving," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2605–2611.

[77] K. Gao, X. Li, B. Chen, L. Hu, J. Liu, R. Du, and Y. Li, "Dual transformer based prediction for lane change intentions and trajectories in mixed traffic environment," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[78] Z. Sheng, Y. Xu, S. Xue, and D. Li, "Graph-based spatial-temporal convolutional network for vehicle trajectory prediction in autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 17 654–17 665, 2022.

[79] X. Li, X. Ying, and M. C. Chuah, "Grip++: Enhanced graph-based interaction-aware trajectory prediction for autonomous driving," *arXiv preprint arXiv:1907.07792*, 2019.

[80] G. Alinezhad Noghre, V. Katariya, A. Danesh Pazho, C. Neff, and H. Tabkhi, "Pishgu: Universal path prediction network architecture for real-time cyber-physical edge systems," in *Proceedings of the ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2023)*, 2023, pp. 88–97.

[81] A. Khamis, Z. Ismail, K. Haron, and A. T. Mohamm, "The effects of outliers data on neural network performance," *Journal of Applied Sciences*, vol. 5, no. 8, pp. 1394–1398, 2005.

[82] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[83] J. Colyar and J. Halkias, "Next generation simulation (ngsim), interstate 80 freeway dataset," *Federal Highway Administration, Tech. Rep*, 2006.

[84] ——, "Next generation simulation (ngsim), us highway-101 dataset," *Federal Highway Administration, Tech. Rep*, 2007.

[85] H. Liao, Z. Li, H. Shen, W. Zeng, D. Liao, G. Li, and C. Xu, "Bat: Behavior-aware human-like trajectory prediction for autonomous driving," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 9, 2024, pp. 10 332–10 340.

[86] X. Li, X. Ying, and M. C. Chuah, "Grip," https://github.com/xincoder/GRIP, 2019.

[87] H. Liao, Z. Li, H. Shen, W. Zeng, D. Liao, G. Li, and C. Xu, "Bat: Behavior-aware human-like trajectory prediction for autonomous driving," https://github.com/Petrichor625/BATraj-Behavior-aware-Model, 2024.

[88] Y. Ma, S. Qi, Y. Zhang, G. Lian, W. Lu, and C.-Y. Chan, "Drivers' visual attention characteristics under different cognitive workloads: An on-road driving behavior study," *International journal of environmental research and public health*, vol. 17, no. 15, p. 5366, 2020.

[89] L. Claussmann, M. Revilloud, D. Gruyer, and S. Glaser, "A review of motion planning for highway autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 5, pp. 1826–1848, 2019.

[90] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[91] T. Gu, J. Snider, J. M. Dolan, and J.-w. Lee, "Focused trajectory planning for autonomous on-road driving," in *2013 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2013, pp. 547–552.

[92] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 4889–4895.

[93] Y. Huang, H. Ding, Y. Zhang, H. Wang, D. Cao, N. Xu, and C. Hu, "A motion planning and tracking framework for autonomous vehicles based on artificial potential field elaborated resistance network approach," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 2, pp. 1376–1386, 2019.

[94] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on intelligent transportation systems*, vol. 17, no. 4, pp. 1135–1145, 2015.

[95] A. Broggi, P. Medici, P. Zani, A. Coati, and M. Panciroli, "Autonomous vehicles control in the vislab intercontinental autonomous challenge," *Annual Reviews in Control*, vol. 36, no. 1, pp. 161–171, 2012.

[96] P. Petrov and F. Nashashibi, "Modeling and nonlinear adaptive control for autonomous vehicle overtaking," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 4, pp. 1643–1656, 2014.

[97] L. Han, H. Yashiro, H. T. N. Nejad, Q. H. Do, and S. Mita, "Bezier curve based path planning for autonomous vehicle in urban environment," in *2010 IEEE intelligent vehicles symposium*. IEEE, 2010, pp. 1036–1042.

[98] M. Geisslinger, F. Poszler, and M. Lienkamp, "An ethical trajectory planning algorithm for autonomous vehicles," *Nature Machine Intelligence*, vol. 5, no. 2, pp. 137–144, 2023.

[99] T. Stahl, A. Wischnewski, J. Betz, and M. Lienkamp, "Multilayer graph-based trajectory planning for race vehicles in dynamic scenarios," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 3149–3154.

[100] S. Kammel, J. Ziegler, B. Pitzer, M. Werling, T. Gindele, D. Jagzent, J. Schröder, M. Thuy, M. Goebl, F. v. Hundelshausen *et al.*, "Team annieway's autonomous system for the 2007 darpa urban challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 615–639, 2008.

[101] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke *et al.*, "Junior: The stanford entry in the urban challenge," *Journal of field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[102] D. Ferguson, T. M. Howard, and M. Likhachev, "Motion planning in urban environments," *Journal of Field Robotics*, vol. 25, no. 11-12, pp. 939–960, 2008.

[103] Z. Han, Y. Wu, T. Li, L. Zhang, L. Pei, L. Xu, C. Li, C. Ma, C. Xu, S. Shen *et al.*, "An efficient spatial-temporal trajectory planner for autonomous vehicles in unstructured environments," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[104] K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, "A survey of path planning algorithms for mobile robots," *Vehicles*, vol. 3, no. 3, pp. 448–468, 2021.

[105] X. Li, Z. Sun, D. Cao, D. Liu, and H. He, "Development of a new integrated local trajectory planning and tracking control framework for autonomous ground vehicles," *Mechanical Systems and Signal Processing*, vol. 87, pp. 118–137, 2017.

[106] X. Hu, L. Chen, B. Tang, D. Cao, and H. He, "Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles," *Mechanical systems and signal processing*, vol. 100, pp. 482–500, 2018.

[107] Y. Rasekhipour, A. Khajepour, S.-K. Chen, and B. Litkouhi, "A potential field-based model predictive path-planning controller for autonomous road vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1255–1267, 2016.

[108] N. Noto, H. Okuda, Y. Tazaki, and T. Suzuki, "Steering assisting system for obstacle avoidance based on personalized potential field," in *2012 15th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2012, pp. 1702–1707.

[109] X. Chen, Z. Huang, Y. Sun, Y. Zhong, R. Gu, and L. Bai, "Online on-road motion planning based on hybrid potential field model for car-like robot," *Journal of Intelligent & Robotic Systems*, vol. 105, no. 1, p. 7, 2022.

[110] R. Omar, E. Sabudin, C. K. M. CK *et al.*, "Potential field methods and their inherent approaches for path planning," *ARPN Journal of Engineering and Applied Sciences*, vol. 11, no. 18, pp. 10 801–10 805, 2016.

[111] I. Sung, B. Choi, and P. Nielsen, "On the training of a neural network for online path planning with offline path planning algorithms," *International Journal of Information Management*, vol. 57, p. 102142, 2021.

[112] Z. Huang, H. Liu, J. Wu, and C. Lv, "Differentiable integrated motion prediction and planning with learnable cost function for autonomous driving," *IEEE transactions on neural networks and learning systems*, 2023.

[113] L. Yang, C. Lu, G. Xiong, Y. Xing, and J. Gong, "A hybrid motion planning framework for autonomous driving in mixed traffic flow," *Green Energy and Intelligent Transportation*, vol. 1, no. 3, p. 100022, 2022.

[114] Á. Fehér, S. Aradi, F. Hegedüs, T. Bécsi, and P. Gáspár, "Hybrid ddpg approach for vehicle motion planning," 2019.

[115] C.-J. Hoel, K. Driggs-Campbell, K. Wolff, L. Laine, and M. J. Kochenderfer, "Combining planning and deep reinforcement learning in tactical decision making for autonomous driving," *IEEE transactions on intelligent vehicles*, vol. 5, no. 2, pp. 294–305, 2019.

[116] H. Krasowski, X. Wang, and M. Althoff, "Safe reinforcement learning for autonomous lane changing using set-based prediction," in *2020 IEEE 23rd international conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–7.

[117] W. L. Hamilton, *Graph representation learning*. Morgan & Claypool Publishers, 2020.

[118] PyTorch, "TORCH.FLATTEN," *[Online]. Available: https://pytorch.org/docs/stable/generated/torch.flatten.html*, Last Accessed: February 2024.

[119] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings. 1985 IEEE international conference on robotics and automation*, vol. 2. IEEE, 1985, pp. 500–505.

[120] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 740–759, 2020.

[121] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *2018 21st international conference on intelligent transportation systems (ITSC)*. IEEE, 2018, pp. 2575–2582.

[122] The MathWorks Inc., "Navigation toolbox version: 2.3 (r2022b)," 2022. [Online]. Available: https://www.mathworks.com/help/stats/index.html

[123] N. Hogan and D. Sternad, "Sensitivity of smoothness measures to movement duration, amplitude, and arrests," *Journal of motor behavior*, vol. 41, no. 6, pp. 529–534, 2009.

[124] J. Wang, L. Chu, Y. Zhang, Y. Mao, and C. Guo, "Intelligent vehicle decision-making and trajectory planning method based on deep reinforcement learning in the frenet space," *Sensors*, vol. 23, no. 24, p. 9819, 2023.

[125] H. Fan, F. Zhu, C. Liu, L. Zhang, L. Zhuang, D. Li, W. Zhu, J. Hu, H. Li, and Q. Kong, "Baidu apollo em motion planner," *arXiv preprint arXiv:1807.08048*, 2018.

[126] Z. Wang, J. Tu, and C. Chen, "Reinforcement learning based trajectory planning for autonomous vehicles," in *2021 China Automation Congress (CAC)*. IEEE, 2021, pp. 7995–8000.

[127] K. B. Naveed, Z. Qiao, and J. M. Dolan, "Trajectory planning for autonomous vehicles using hierarchical reinforcement learning," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2021, pp. 601–606.

[128] G. Li, X. Zhang, H. Guo, B. Lenzo, and N. Guo, "Real-time optimal trajectory planning for autonomous driving with collision avoidance using convex optimization," *Automotive Innovation*, pp. 1–11, 2023.

[129] J. Cheng, Y. Chen, Q. Zhang, L. Gan, C. Liu, and M. Liu, "Real-time trajectory planning for autonomous driving with gaussian process and incremental refinement," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 8999–9005.

[130] D. Cao, X. Wang, L. Li, C. Lv, X. Na, Y. Xing, X. Li, Y. Li, Y. Chen, and F.-Y. Wang, "Future directions of intelligent vehicles: Potentials, possibilities, and perspectives," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 1, pp. 7–10, 2022.

[131] S. Singh, "Critical reasons for crashes investigated in the national motor vehicle crash causation survey," Tech. Rep., 2015.

[132] M. Joerger and M. Spenko, "Towards navigation safety for autonomous cars," *Inside GNSS*, 2017.

[133] J. Jiang and A. Astolfi, "Lateral control of an autonomous vehicle," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 2, pp. 228–237, 2018.

[134] J. Yang and N. Zheng, "An expert fuzzy controller for vehicle lateral control," in *IECON 2007-33rd Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2007, pp. 880–885.

[135] X. Huang, H. Zhang, and J. Wang, "Robust weighted gain-scheduling hâ vehicle lateral dynamics control in the presence of steering system backlash-type hysteresis," in *2013 American Control Conference*. IEEE, 2013, pp. 2827–2832.

[136] S.-H. Lee and C. C. Chung, "Predictive control with sliding mode for autonomous driving vehicle lateral maneuvering," in *2017 American Control Conference (ACC)*. IEEE, 2017, pp. 2998–3003.

[137] R. C. Coulter *et al.*, *Implementation of the pure pursuit path tracking algorithm*. Carnegie Mellon University, The Robotics Institute, 1992.

[138] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[139] S. Dominguez, A. Ali, G. Garcia, and P. Martinet, "Comparison of lateral controllers for autonomous vehicle: Experimental results," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 1418–1423.

[140] K. Muhammad, A. Ullah, J. Lloret, J. Del Ser, and V. H. C. de Albuquerque, "Deep learning for safe autonomous driving: Current challenges and future directions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4316–4336, 2020.

[141] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, "A survey of deep learning applications to autonomous vehicle control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2, pp. 712–733, 2020.

[142] D. Li, D. Zhao, Q. Zhang, and Y. Chen, "Reinforcement learning and deep learning based lateral control for autonomous driving [application notes]," *IEEE Computational Intelligence Magazine*, vol. 14, no. 2, pp. 83–98, 2019.

[143] D. Pomerleau, "Neural network vision for robot driving," in *Intelligent Unmanned Ground Vehicles: Autonomous Navigation Research at Carnegie Mellon*. Springer, 1997, pp. 53–72.

[144] Z. Chen and X. Huang, "End-to-end learning for lane keeping of self-driving cars," in *2017 IEEE intelligent vehicles symposium (IV)*. IEEE, 2017, pp. 1856–1860.

[145] V. Rausch, A. Hansen, E. Solowjow, C. Liu, E. Kreuzer, and J. K. Hedrick, "Learning a deep neural net policy for end-to-end control of autonomous vehicles," in *2017 American Control Conference (ACC)*. IEEE, 2017, pp. 4914–4919.

[146] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 661–668.

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

[148] J. Kwon, A. Khalil, D. Kim, and H. Nam, "Incremental end-to-end learning for lateral control in autonomous driving," *IEEE Access*, vol. 10, pp. 33 771–33 786, 2022.

[149] A. Khalil and J. Kwon, "Anec: Adaptive neural ensemble controller for mitigating latency problems in vision-based autonomous driving," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 9340–9346.

[150] Z. Zhu and H. Zhao, "Learning autonomous control policy for intersection navigation with pedestrian interaction," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 5, pp. 3270–3284, 2023.

[151] Q. Ma, X. Yin, X. Zhang, X. Xu, and X. Yao, "Game-theoretic receding-horizon reinforcement learning for lateral control of autonomous vehicles," *IEEE Transactions on Vehicular Technology*, 2024.

[152] M. Brasch, I. S. Heinz, and A. Bayer, "Lateral control of a vehicle using reinforcement learning," in *2022 8th International Conference on Control, Decision and Information Technologies (CoDIT)*, vol. 1. IEEE, 2022, pp. 451–456.

[153] A. Wasala, D. Byrne, P. Miesbauer, J. OâHanlon, P. Heraty, and P. Barry, "Trajectory based lateral control: A reinforcement learning case study," *Engineering Applications of Artificial Intelligence*, vol. 94, p. 103799, 2020.

[154] P. Wang, C.-Y. Chan, and A. de La Fortelle, "A reinforcement learning based approach for automated lane change maneuvers," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1379–1384.

[155] C. Shen, H. Guo, F. Liu, and H. Chen, "Mpc-based path tracking controller design for autonomous ground vehicles," in *2017 36th Chinese Control Conference (CCC)*. IEEE, 2017, pp. 9584–9589.

[156] P. Falcone, M. Tufo, F. Borrelli, J. Asgari, and H. E. Tseng, "A linear time varying model predictive control approach to the integrated vehicle dynamics control problem in autonomous systems," in *2007 46th IEEE Conference on Decision and Control*. IEEE, 2007, pp. 2980–2985.

[157] G. Costa, J. Pinho, M. A. Botto, and P. U. Lima, "Online learning of mpc for autonomous racing," *Robotics and Autonomous Systems*, p. 104469, 2023.

[158] Z. Zhou, C. Rother, and J. Chen, "Event-triggered model predictive control for autonomous vehicle path tracking: Validation using CARLA simulator," *IEEE Transactions on Intelligent Vehicles*, pp. 1–9, 2023.

[159] A. Kazemi, I. Sharifi, and H. Talebi, "Longitudinal and lateral control of vehicle platoons using laguerre-based and robust mpc with merge and exit maneuvers," *Control Engineering Practice*, vol. 142, p. 105737, 2024.

[160] M. Jia, M. Tao, M. Xu, P. Zhang, J. Qiu, G. Bergsieker, and J. Chen, "Rl-mpc: Reinforcement learning aided model predictive controller for autonomous vehicle lateral control," SAE Technical Paper, Tech. Rep., 2024.

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

[162] Z. Zheng, C. Wang, T. Xu, D. Shen, P. Qin, X. Zhao, B. Huai, X. Wu, and E. Chen, "Interaction-aware drug package recommendation via policy gradient," *ACM Transactions on Information Systems*, vol. 41, no. 1, pp. 1–32, 2023.

[163] X. Li, X. Wang, X. He, L. Chen, J. Xiao, and T.-S. Chua, "Hierarchical fashion graph network for personalized outfit recommendation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 159–168.

[164] Y. Li, W. Yuan, S. Zhang, W. Yan, Q. Shen, C. Wang, and M. Yang, "Choose your simulator wisely: A review on open-source simulators for autonomous driving," *IEEE Transactions on Intelligent Vehicles*, 2024.

[165] H. Yang, J. Wu, Z. Hu, and C. Lv, "Real-time driver cognitive workload recognition: Attention-enabled learning with multimodal information fusion," *IEEE Transactions on Industrial Electronics*, vol. 71, no. 5, pp. 4999–5009, 2023.

[166] D. Li and O. Okhrin, "Modified ddpg car-following model with a real-world human driving experience with carla simulator," *Transportation research part C: emerging technologies*, vol. 147, p. 103987, 2023.

[167] J. Pahk, S. Park, J. Shim, S. Son, J. Lee, J. An, Y. Lim, and G. Choi, "Lane segmentation data augmentation for heavy rain sensor blockage using realistically translated raindrop images and carla simulator," *IEEE Robotics and Automation Letters*, 2024.

[168] F. Khan, "Simulation-based testing of automated driving systems," in *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*, 2024, pp. 217–219.

[169] Association for Standardization of Automation and Measuring Systems, "ASAM OpenDRIVE," *[Online]. Available: https://www.asam.net/standards/detail/opendrive/*, Last Accessed: July 2023.

[170] Wes Davis, "In the first autonomous racing league race, the struggle was real," *The Verge. [Online]. Available: https://www.theverge.com/2024/4/27/24142989/a2rl-autonomous-race-cars-f1-abu-dhabi*, Last Accessed: July 2024.

[171] A. Artuñedo, M. Moreno-Gonzalez, and J. Villagra, "Lateral control for autonomous vehicles: A comparative evaluation," *Annual Reviews in Control*, vol. 57, p. 100910, 2024.

[172] M. Fliess and C. Join, "Model-free control," *International journal of control*, vol. 86, no. 12, pp. 2228–2252, 2013.

[173] M. Moreno-Gonzalez, A. Artuñedo, J. Villagra, C. Join, and M. Fliess, "Speed-adaptive model-free lateral control for automated cars," *IFAC-PapersOnLine*, vol. 55, no. 34, pp. 84–89, 2022.

[174] G. Chen, X. Zhao, Z. Gao, and M. Hua, "Dynamic drifting control for general path tracking of autonomous vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 3, pp. 2527–2537, 2023.

[175] Y. Shan, B. Zheng, L. Chen, L. Chen, and D. Chen, "A reinforcement learning-based adaptive path tracking approach for autonomous driving," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 10 581–10 595, 2020.

[176] Moid Hassan, "Scalable time series forecasting," *Data Science at Microsoft. Medium. [Online]. Available: https://medium.com/data-science-at-microsoft/scalable-time-series-forecasting-fee61da75923*, Last Accessed: July 2024.

[177] R. Wen, K. Torkkola, B. Narayanaswamy, and D. Madeka, "A multi-horizon quantile recurrent forecaster," *arXiv preprint arXiv:1711.11053*, 2017.

[178] C. Valenzuela, H. Allende, and C. Valle, "Multi-horizon scalable wind power forecast system," in *Progress in Artificial Intelligence and Pattern Recognition: 6th International Workshop, IWAIPR 2018, Havana, Cuba, September 24–26, 2018, Proceedings 6*. Springer, 2018, pp. 317–325.

[179] M. Adil, H. Song, M. K. Khan, A. Farouk, and Z. Jin, "5g/6g-enabled metaverse technologies: Taxonomy, applications, and open security challenges with future research directions," *Journal of Network and Computer Applications*, p. 103828, 2024.

[180] R. Giuliano, "From 5g-advanced to 6g in 2030: New services, 3gpp advances and enabling technologies," *IEEE Access*, 2024.

[181] R. Chataut, M. Nankya, and R. Akl, "6g networks and the ai revolutionâexploring technologies, applications, and emerging challenges," *Sensors*, vol. 24, no. 6, p. 1888, 2024.

[182] S. Sharma, R. Tyagi, and S. Mohan, "Artificial intelligence and green 6g network-enabled architectures, scenarios, and applications for autonomous connected vehicles," in *6G Connectivity-Systems, Technologies, and Applications*. River Publishers, pp. 187–212.

[183] G. Kakkavas, M. Diamanti, V. Karyotis, K. N. Nyarko, M. Gabriel, A. Zafeiropoulos, S. Papavassiliou, and K. Moessner, "5g perspective of connected autonomous vehicles: Current landscape and challenges toward 6g," *IEEE Wireless Communications*, 2024.

[184] A. A. Anvigh, Y. Khavan, and B. Pourghebleh, "Transforming vehicular networks: How 6g can revolutionize intelligent transportation?" *Science, Engineering and Technology*, vol. 4, no. 1, pp. 80–93, 2024.

[185] R. N. Jazar, *Vehicle Dynamics: Theory and Application*. Springer, 2017.

[186] Y.-M. Choi and J.-H. Park, "Game-based lateral and longitudinal coupling control for autonomous vehicle trajectory tracking," *Ieee Access*, vol. 10, pp. 31 723–31 731, 2021.

[187] A. Chaibet, L. Nouveliere, S. Mammar, M. Netto, and R. Labayrade, "Backstepping control synthesis for both longitudinal and lateral automated vehicle," in *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.* IEEE, 2005, pp. 42–47.

[188] E. H. Lim and J. K. Hedrick, "Lateral and longitudinal vehicle control coupling for automated vehicle operation," in *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*, vol. 5. IEEE, 1999, pp. 3676–3680.

[189] X. Li, A. Rakotonirainy, and X. Yan, "How do drivers avoid collisions? a driving simulator-based study," *Journal of safety research*, vol. 70, pp. 89–96, 2019.