UNIVERSITÄT BONN

Masterarbeit
zur Erlangung des Grades
Master of Science (M. Sc.)
der Landwirtschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn
Institut für Geodäsie und Geoinformation

Evaluation of Traffic Scene Evolutions Using Scene-Consistent Prediction

von

Dhagash Desai

aus
Ahmedabad, India

Bonn 2024

Supervisor:
Prof. Dr. Cyrill Stachniss, University of Bonn, Germany

Second Supervisor:
Julian Schmidt, Mercedes-Benz AG, Germany
Benedikt Mersch, University of Bonn, Germany

Statement of Authorship

I hereby certify that this master thesis has been composed by myself. I have not made use of the work of others or presented it here unless it is otherwise acknowledged in the text. All references and verbatim extracts have been quoted, and all sources of information have been specifically acknowledged.

Place, Date	(Signature)

Acknowledgments

I would like to thank all the people who directly or indirectly helped in the last three years; without their support, this long academic journey to a master's and this thesis would have been impossible. Firstly, I would like to thank Prof. Dr. Cyrill Stachniss for allowing me to work on this topic under his supervision. I would also like to thank Dominik at Mercedes-Benz AG for presenting me with this topic, providing me with working facilities, and for helpful discussion throughout the thesis.

I am incredibly grateful to my mentors, Julian Schmidt and Benedikt Mersch. They have always motivated me, patiently listened to my endless doubts, and, more importantly, have always been ready to discuss the subject matter whenever needed. I also acknowledge Julian Jordan's valuable input during the thesis. This thesis wouldn't have been possible without their constant support. Special thanks to Ignacio Vizzo, Tiziano Guadagnino, Luca Lobefaro, and Meher Malladi for all the fun discussions about robotics and whatnot, but more importantly, for the endless disturbance that helped create a stress-free environment during the thesis. I also thank Elias Marks, Federico Magistri, Lasse Peters, Matteo Sodano, Thomas Laebe, and Jens Behley for their guidance throughout the master's. I would also like to thank the janitor and the hausmeister for maintaining the working spaces in the lab.

In the past three years, I have had the pleasure of meeting numerous people I now call friends. They always stood by my side during tough times and are the reason I call Germany my second home. So, special thanks to Amit, Sumanth, Richa, Preetham, Alonso, Amanda, Niveditha (recent friend), Sourav, and Kenneth. I am also thankful for the support from friends back in India. Also, big thanks to my friends in Stuttgart, Sayooj, John, and Marco. I want to dedicate a special thanks to Saurabh, who became my close friend, for all the fun times we had throughout the master's, be it projects, coursework, or intense discussions about life. This section would be incomplete without expressing my deepest gratitude to Sai for being my biggest cheerleader. I cannot thank her enough for all her love, support, and understanding and for being my comfort and peace in these challenging times.

Finally, I would like to thank my family for providing me with unwavering

support and belief in every decision I make. I want to express my profound appreciation to my mother, who has been my inspiration for hard work and work ethic. Words cannot express my gratitude to her for always supporting and giving us freedom in the most challenging times.

Abstract

Recent years have seen a significant increase in interest in autonomous driving because it provides safer, more comfortable, and more efficient forms of transportation. When autonomous vehicles (AV) are used in urban settings, they inevitably will interact with human-driven cars and vulnerable users of the road, such as pedestrians and cyclists. This not only requires AVs to detect but also to understand and estimate the intentions of surrounding agents to make safe, reliable, and socially compliant decisions. This task is referred to as motion prediction. The fact that each agent has a different unobservable intent and a variety of complex interaction possibilities makes this task difficult.

Learning-based models have gained prominence in the field of motion prediction due to the availability of large-scale datasets and their ability to process complex information, hence achieving long-term prediction. However, most of the earlier methods generate inconsistent agent-independent predictions that result in conflicting forecasts. As a result, the research on motion prediction has shifted focus to generate sets of scene-level or joint future predictions of surrounding agents for each driving scenario.

In this thesis, we leverage recent advances in scene-consistent prediction to provide an initialization for the downstream planning task. The scene-consistent predictor outputs multiple possible evolutions of traffic scenes, where each predicted scene includes the future trajectory of all agents present in the scenario, including the autonomous vehicle (AV). To this end, we employ an explicit decision cost module comprising various pre-defined cost metrics to evaluate/rank each predicted scene adhering to the downstream planning task. Furthermore, we introduce two approaches to integrate goal information of the AV during the prediction process. One approach introduces this information as part of the decision-cost module, and the other approach provides a way to integrate this information as input to the model directly, thereby aligning predictions to high-level goal information available to AV. We extensively evaluate our approach on the Argoverse2 dataset and show the effectiveness of our decision-cost module and goal-integration approaches.

Zusammenfassung

In den letzten Jahren hat das Interesse am autonomen Fahren stark zugenommen, da es sicherere, komfortablere und effizientere Transportmöglichkeiten bietet. Wenn AVs in städtischen Umgebungen eingesetzt werden, müssen sie unweigerlich mit von Menschen gesteuerten Autos und ungeschützten Verkehrsteilnehmern wie Fußgängern und Radfahrern interagieren. Diese müssen AVs nicht nur erkennen, sondern auch ihre Absichten verstehen und einschätzen, um sichere und zuverlässige Entscheidungen zu treffen. Diese Aufgabe wird als Bewegungsvorhersage bezeichnet. Die Tatsache, dass jeder Agent eigene, unbekannte Ziele hat und eine Vielzahl komplexer Interaktionsmöglichkeiten besteht, macht diese Aufgabe schwierig.

Lernbasierte Modelle haben auf dem Gebiet der Bewegungsvorhersage an Bedeutung gewonnen, da große Datensätze zur Verfügung stehen und. Diese Modelle sind in der Lage, komplexe Informationen zu verarbeiten und somit langfristige Vorhersagen zu treffen. Die meisten bestehenden Ansätze erzeugen jedoch inkonsistente, von anderen Verkehrsteilnehmern unabhängige Vorhersagen, die zu widersprüchlichen Prognosen führen. Infolgedessen hat sich der Schwerpunkt der Forschung im Bereich der Bewegungsvorhersage verlagert, um für jede Fahrsituation eine komplette, konsistente Szene beziehungsweise gemeinsame Vorhersagen für alle Akteure zu treffen.

In dieser Arbeit nutzen wir die jüngsten Fortschritte in der szenenkonsistenten Vorhersage, um eine Initialisierung für die nachgelagerte Planungsaufgabe bereitzustellen. Der szenenkonsistente Prädiktor gibt mehrere mögliche Entwicklungen von Verkehrsszenen aus, wobei jede vorhergesagte Szene die zukünftige Trajektorie aller im Szenario anwesenden Agenten, einschließlich des autonomen Fahrzeugs (AV), enthält. Zu diesem Zweck verwenden wir ein explizites Entscheidungskostenmodul, das verschiedene vordefinierte Kostenmetriken umfasst, um jede vorhergesagte Szene unter Berücksichtigung der nachgelagerten Planungsaufgabe zu bewerten. Darüber hinaus führen wir zwei Ansätze ein, um Zielinformationen des AV während des Vorhersageprozesses zu integrieren. Ein Ansatz führt diese Informationen als Teil des Entscheidungskostenmoduls ein. Der andere Ansatz bietet eine Möglichkeit, diese Informationen direkt als Eingabe in das Modell zu integrieren, wodurch die Vorhersagen an den für AV verfügbaren groben

Zielinformationen ausgerichtet werden. Wir evaluieren unseren Ansatz auf dem Argoverse2 Datensatz und zeigen die Effektivität unseres Entscheidungskostenmodul und der zwei Integrationsansätze für die Zielinformation.

Task Description

Rheinische
Friedrich-Wilhelms-
Universität Bonn

Uni Bonn · Stachniss · Nussallee 15 · 53115 Bonn

Prof. Dr. Cyrill Stachniss
Professor

Institute of Geodesy
and Geoinformation

Photogrammetry
& Robotics Lab

Nussallee 15
53115 Bonn

Tel.: +49-228-73-2714
Fax: +49-228-73-2712
cyrill.stachniss@igg.uni-bonn.de

Secretary:
Birgit Klein
Tel.: +49-228-73-2713
Fax: +49-228-73-2712

Bonn, June 27, 2023

To whom it may concern

Task Description for M.Sc. thesis: Evaluation of Traffic Scene Evolutions using Scene-Consistent Prediction

Motivation: The ability to predict the future behavior of other agents in a scene is crucial for autonomous vehicles (AV), as it enables the generation of safe and reliable trajectories. While traditional physics-based prediction models are effective for short-term predictions, they struggle to accurately forecast beyond a horizon of 1-2 seconds. In contrast, learning-based prediction models perform better for longer prediction horizons due to the availability of extensive datasets.

However, a limitation of these learned prediction models is their agent-independent nature, treating each agent's trajectory prediction as an independent problem. This approach fails to capture the complex dependencies and interactions among agents in real-world scenarios, potentially leading to conflicting forecasts. To address this limitation, scene-consistent motion prediction models have emerged, enabling joint reasoning about the future states of all agents within the scene. Attention-based transformer models, in particular, have shown promising results in this domain.

State-of-the-art scene-consistent prediction models generate multiple potential outcomes of a traffic scene simultaneously. Unlike traditional agent-independent methods, these models consider the joint prediction of agents, establishing correspondences between the predicted trajectories of different agents. This joint prediction approach allows for a more comprehensive assessment of the potential future evolution of each scene by considering the interactions and dependencies between agents. By capturing the feasibility and coherence of the entire scene, this holistic approach provides a deeper understanding of how the scene may unfold in the future.

Task Description: The main objective of the thesis is to develop a prediction approach that produces multiple possible outcomes of a future scene and selects the most likely one. To do this, an existing scene-consistent prediction model should be extended by a decision cost module to assess the evolution of predicted traffic scenes with respect to pre-defined cost metrics. Based on such an evaluation, the least cost scene can be chosen and forwarded to a planner for ego-motion planning. The assessment should assign a cost to each scene based on predicted collisions with other traffic participants, adherence to traffic rules, and other factors that shall be considered. Since existing approaches typically only use previously observed states of the scene and a high-definition map, the student should explore how to integrate goal information of AV to achieve a more

informed scene prediction.

In summary, the core tasks of the thesis are:

- Literature review to gain a comprehensive understanding of the current state-of-the-art scene-consistent prediction models and select an approach
- Reproducing the results of the selected approach and testing it on Argoverse 2
- Modifying and extend the scene-consistent model with a decision cost module to assess multiple future evolution of a traffic scene
- Extending the pipeline to to achieve more informed predictions by integrating goal information of the AV

Point Of Departure: The point of departure for the student should be:

- Literature review of existing scene-consistent models and their metrics [5] [2] [3]
- Getting familiar with Pytorch [1]
- Getting familiar with Argoverse 2 [6]
- Comprehensive study about planning cost functions [4]

Kind regards,

Cyrill Stachniss

[1] P. Adam, G. Sam, M. Francisco, L. Adam, B. James, C. Gregory, K. Trevor, L. Zeming, G. Natalia, A. Luca, D. Alban, K. Andreas, Y. Edward, D. Zachary, R. Martin, T. Alykhan, C. Sasank, S. Benoit, F. Lu, B. Junjie, and C. Soumith. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[2] R. Girgis, F. Golemo, F. Codevilla, M. Weiss, J.A. D'Souza, S.E. Kahou, F. Heide, and C. Pal. Latent Variable Sequential Set Transformers for Joint Multi-Agent Motion Prediction. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2022.

[3] B. He and Y. Le. Multi-future Transformer: Learning diverse interaction modes for behaviour prediction in autonomous driving. *Intelligent Transport Systems*, 16(9):1249–1267, 2022.

[4] Z. Huang, H. Liu, J. Wu, W. Huang, and C. Lv. Learning interaction-aware motion prediction model for decision-making in autonomous driving, 2023.

[5] J. Ngiam, B. Caine, V. Vasudevan, Z. Zhang, H.L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal, D. Weiss, B. Sapp, Z. Chen, and J. Shlens. Scene Transformer: A unified multi-task model for behavior prediction and planning. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2022.

[6] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J.K. Pontes, D. Ramanan, P. Carr, and J. Hays. Argoverse 2: Next Generation Datasets for Self-driving Perception and Forecasting. In *Proc. of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

Contents

Chapter 1

Introduction

Autonomous vehicles (AV), commonly referred to as self-driving vehicles, are systems that enable navigating in complex urban scenarios without human intervention [42]. Thus, the technology has the potential to revolutionize the transportation system in the future, providing a safer, more convenient, and more enjoyable driving experience. Furthermore, it can provide independent mobility for elderly and disabled people. The levels for autonomous driving range from Level 0 (no automation) up to Level 5 (full vehicle autonomy) as defined by SAE (Society of Automotive Engineers) [42]. In the past few decades, there has been significant interest in AVs from both industry and academia, leading to tremendous progress in the technology. However, despite substantial improvement in the field, developing Level 5 i.e., fully autonomous vehicles, remains an unsolved challenge.

To understand the dynamic environment around the AVs, they are equipped with a range of sensors such as LiDARs, RADAR, cameras, Inertial Measurement Units (IMUs), and many more, enabling them to sense, act and plan in complex urban environments. A typical AV equipped with sensors is shown in Figure 1.1. The task of the AVs is to take information from various sensor streams and output control commands to navigate a given scenario. Many approaches have been proposed to tackle the problem. The most prevalent approach in both industry and academia is a modular pipeline architecture. Modular pipelines break the task of autonomous driving into manageable sub-tasks i.e., perception, prediction, planning, and control. The perception module is responsible for localizing, detecting, and tracking the objects surrounding the AV using raw sensor data as input.

To navigate safely and reliably, it is essential to estimate the future driving behavior of surrounding traffic participants. The prediction module computes the expected driving behavior for each traffic participant over a desired time in the future, often called the prediction horizon, by taking tracking inputs from

Figure 1.1: An autonomous vehicle equipped with various sensors such as LiDARs, cameras, and RADARs. Courtesy: *https://www.wired.com/story/gm-cruise-generation-3-self-driving-car/.*

the upstream perception module. The predicted trajectories can then be used in the planning module to check whether the generated trajectory for AV is feasible. This, in turn, allows for a safe and reliable plan, which the control module can then execute.

In this thesis, we focus on enhancing current state-of-the-art motion prediction approaches, which, as discussed later, allows for a more comprehensive assessment of the potential future evolution of each scene. Specifically, we utilize the planning-based constraints to assess the predicted scenes, which, in turn, leads to more informed predictions.

1.1 Motivation

Learning-based models have gained prominence in the field of motion prediction, leveraging map-relevant information along with the agent's past states from upstream modules to make reliable long-term forecasts. The increasing availability of open-source datasets and the increase in available computing power contribute to the rise of these models. In the context of multi-agent motion prediction, most existing approaches predict independent trajectories for each agent without considering their interactions in the future [8, 25, 11, 1, 33, 27]. Such predictions may lead to inconsistent forecasts, highlighting the need for scene-consistent predictions for downstream tasks like planning.

Recent approaches inspired by advancements in the natural language pro-

cessing (NLP) domain address this limitation by focusing on predicting multiple scenes or joint future trajectories of agents within a scene [30, 32, 12, 15]. In these methods, each scene consists of distinct future trajectory predictions for each agent, including predictions for the ego vehicle (AV), that are consistent with one another. This implies that the predictions for other agents are implicitly conditioned on one another.

In this thesis, we focus on assessing the predicted evolution of each scene from scene-consistent predictor, leading to a deeper understanding of how the scene may unfold in the future and selecting the most likely scene based on a detailed assessment. Some common methods to select the scene from multiple predictions involve using a score-decoder that gives a likelihood score for each scene [12, 15]. Another commonly used approach is to rank the trajectories based on Euclidean distance and select the one with the least Euclidean distance [32]. However, the main drawback of the following evaluation/scoring criterion is that they lack interpretability/reasoning about why this scene is more likely than others and has a sub-optimal performance at intersections due to the use of the Euclidean distance metric as ranking criteria.

Furthermore, while many state-of-the-art approaches primarily use an agent's past trajectory and road network information to predict future states, they often overlook the goal information for the AV. Specifically, the goal information which indicates the intended destination or planned route for the AV as provided by the downstream planner is often neglected in existing methods. Certain approaches address this limitation by incorporating goal information. For instance, some methods condition predictions of other agents based on specific end-states of the AV [31, 30]. Another common approach involves generating a set of trajectories for the AV and obtaining conditioned predictions for each trajectory [35]. However, these methods have drawbacks, as they often rely on a specific end-state for the AV and may not consider multiple ways to achieve the goal. Additionally, generating trajectories and conditioning predictions can significantly slow the planning process.

1.2 Goal and Main Contributions

This thesis's main contribution is to develop a comprehensive evaluation approach to assess the evolution of predicted scenes to achieve more informed scene prediction. We select an existing scene-consistent approach and extend it by a modular decision cost module that evaluates the predicted scenes based on pre-defined cost metrics. The cost metrics for assessment consider adherence to traffic, collision with other traffic participants, and other factors.

We also consider high-level goal information in the decision cost module, par-

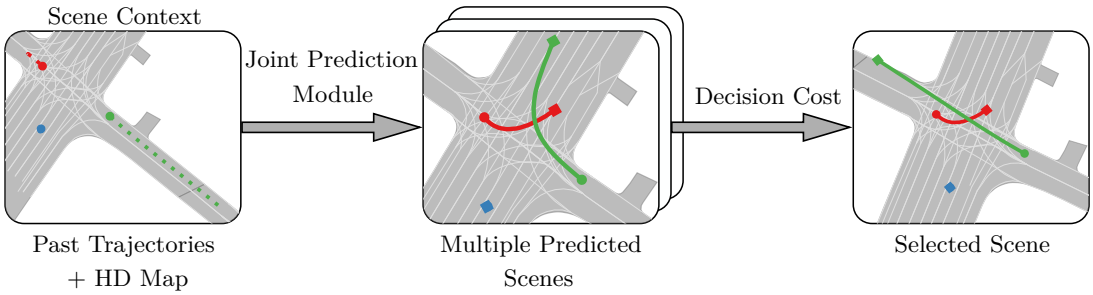


Figure 1.2: The past trajectories of agents and HD-Map information are used to predict multiple future scenes. Each scene is then evaluated and ranked using the decision cost module to select the most likely scene among all the predicted scenes.

ticularly for AV. This, in turn, allows us to evaluate the predictions of other agents conditioned on potential AV's trajectory. Subsequently, the AV's trajectory corresponding to the least scene cost can be forwarded as an initialization to ego-motion planning. We also provide an approach to integrate high-level goals in the model training to produce a goal-informed consistent prediction. An overview of our approach is presented in the Figure 1.2. In summary

- We propose an explicit decision cost module comprising various pre-defined cost metrics to comprehensively evaluate/rank the evolutions of predicted scenes generated from a scene-consistent motion predictor
- We propose methods to integrate goal information of AV. First, we include it in the decision cost and then modify the network to include the goal information of AV to get conditioned prediction on the high-level goal of AV
- We thoroughly assess our proposed method using the open-source Argoverse2 dataset, demonstrating the effectiveness of various cost components. Additionally, we compare different approaches of goal integration and our method with unimodal baselines, focusing on prediction and open-loop planning metrics

1.3 Overview of the Thesis

This thesis aims to provide an approach for a comprehensive evaluation to assess the evolution of predicted scenes to achieve more informed scene prediction. In Chapter 2, we examine key components of autonomous driving and provide a comprehensive summary of existing techniques and state-of-the-art methods for motion prediction related to autonomous driving. We provide a background on fundamentals frequently used in the thesis in Chapter 3. In Chapter 4, we provide a detailed explanation of components and the implementation details of

our approach. Chapter 5 provides quantitative and qualitative analysis of our pipeline on various baselines and models.

Furthermore, we open-source our pipeline, which has been efficiently implemented in Python. We also provide the weights for the trained model and the data loaders associated with the dataset used in this thesis, allowing further improvement on the proposed idea.

Chapter 2

Related Work

Autonomous driving as technology is gaining popularity in academia and industry, paving the way for safe, efficient, and reliable transportation in the future. In this chapter, we first provide an overview of key components of autonomous driving technology and describe the popular system design choices to tackle the autonomous driving problem in Section 2.1. Then, in Section 2.2, we provide a brief overview of existing techniques present, particularly for motion-prediction modules related to autonomous driving.

2.1 Autonomous Driving Systems

Many studies have been proposed on how to tackle the task of autonomous driving effectively. Yurtsever et al. [41] provides a comprehensive survey on autonomous driving systems that classify the autonomous driving architecture based on algorithm design into two broad categories: modular and end-to-end driving systems. End-to-end systems use neural networks as driving policies. These policies generate actions or trajectories from sensory input or perception results [18], as seen in Figure 2.1b. The approaches for end-to-end driving systems are deep imitation learning [19, 7], where the model aims to imitate human behavior (ground truth), or reinforcement learning [23, 3], where the goal of the network is to learn a set of actions that yields the maximum cumulative future rewards [41]. However, the end-to-end driving systems suffer from poor interpretability and generalization capabilities and lack safety and stability guarantees [18].

Modular systems, on the other hand, consist of separate components. These components connect sensory inputs and actuator outputs, as shown in Figure 2.1a. Although end-to-end systems have become increasingly popular in academia in recent years, modular systems are more prevalent in the industry. They are used by companies such as Motional, Tesla, and Waymo. Several studies on

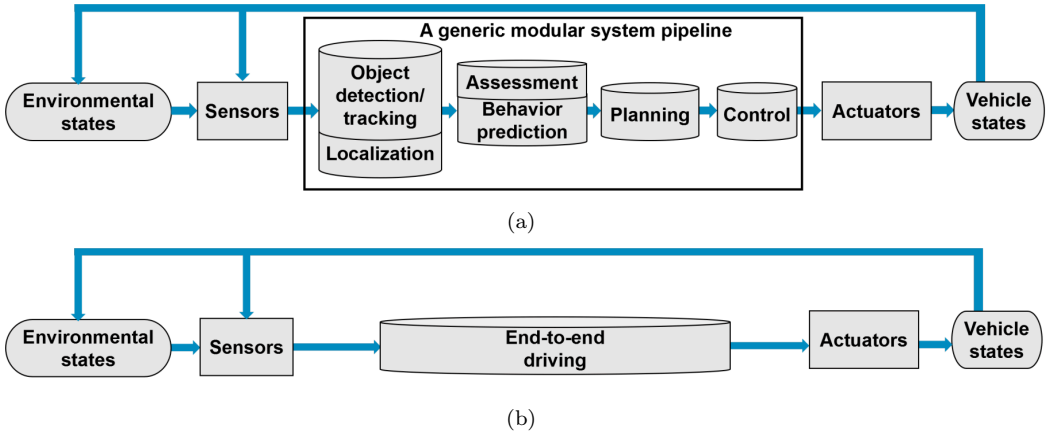


Figure 2.1: An overview of information flow in common autonomous driving systems (a) modular pipeline, information passes through different components to produce vehicle control (b) end-to-end pipeline, takes input from sensors and passes it to neural-network generating driving policies. Courtesy: *Yurtsever* et al. [41].

modular system-based approaches have been conducted since the first successful demonstration of modular systems in the 2005 DARPA challenge [37]. The core components of modular systems are perception, prediction, planning, and control. The modular systems [26, 38] use data obtained by an upstream perception component and provide them to the prediction module, which then predicts the movement of surrounding agents present in a scenario. These predictions are then used in the planning module to plan a safe and collision-free trajectory for AV. The advantages of separate components are that they enable specialized and parallelized development of individual components. The output of each module is well-defined and human-interpretable. Thus, the intermediate representation from the modular systems provide interpretability [34]. However, the drawback of such systems is the independence assumption between prediction and planning components, which are not truly independent, as the AV's plan might significantly impact the behaviors of other agents in the scene [30]. In this thesis, we build upon a modular approach.

2.2 Motion Prediction

Motion prediction is crucial to planning safe and reliable trajectories for AVs. Motion prediction aims to predict future vehicle trajectories, considering observations of past driving trajectories and the driving environment. There has been a growing amount of literature on motion prediction for AV driven by the increasing interest in self-driving technology. Earlier motion prediction approaches have utilized physics-based models [2, 22] (e.g. constant velocity and constant acceleration) or hand-crafted rules [16] to address the problem. However, due

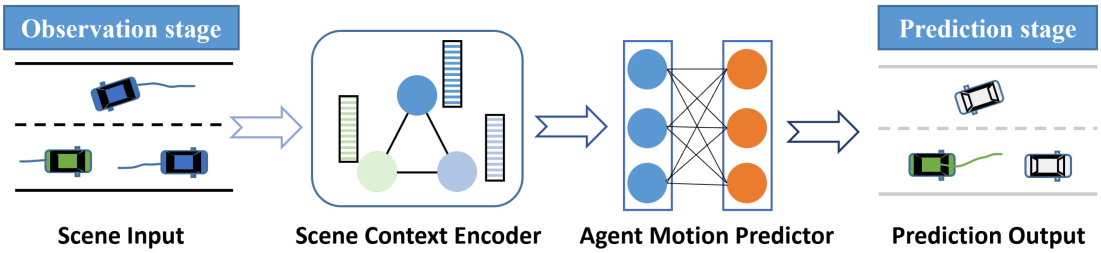


Figure 2.2: The diagram depicts the common learning-based motion prediction paradigm. The target vehicle is represented in green. The input to the system includes the historical trajectories of various vehicles and the road structure. The model's output is the predicted future trajectory of the target vehicle. Courtesy: *Huang* et al. [17].

to their low complexity, these models fail to accurately describe complex vehicle interactions in real-world environments [14]. As a result, they are not effective for long-term prediction. With the increase in amount of large-scale publicly available datasets and benchmarks such as Argoverse [6], Argoverse2 [39], nuScenes [4], and Waymo [10] as well increase in available computation, learning-based motion-prediction approaches are becoming more prominent in recent years. Hence, in this thesis, we will focus mainly on learning-based motion-prediction models.

Learning-based motion prediction models draw inspiration from advancements in Natural Language Processing (NLP), as both domains involve data structures containing sequences. Mozaffari et al. [29] categorizes data-based models according to their representation of the input, output type, and prediction methods. The input representation of models has changed drastically since its inception. The more primitive methods rely only on the past trajectory of the target vehicles [44], and the surrounding vehicles [9]. However, it is not enough to rely on the track history of the target and the surrounding vehicle to predict behavior, as other factors, such as environmental conditions and traffic rules, can also alter vehicle behavior [29]. Therefore, the more recent models consider the agent's movement history and elements of the road map, and more importantly, the interactions between the agents and the scene [30]. Hence, a state-of-the-art motion prediction pipeline contains the components described in Figure 2.2. In this thesis, we assume that information about the agent's past states and HD-Map of the scene is available.

The most popular approach to modeling the scene input is grid- or graph-based. Grid-based approaches are inspired by computer vision literature and model inputs such as rasterized images, and then extract their features using convolutional neural networks (CNN) [8, 25, 20, 33, 5]. However, due to the local structure of the receptive field, it is challenging to capture spatially distant interactions in a lane [30]. Liang et al. [27] proposed the first method of modeling a scene input (HD-Map) as a graph and extracting its properties with the help of a graph convolutional networks (GCN). The graph-based approach prevents loss

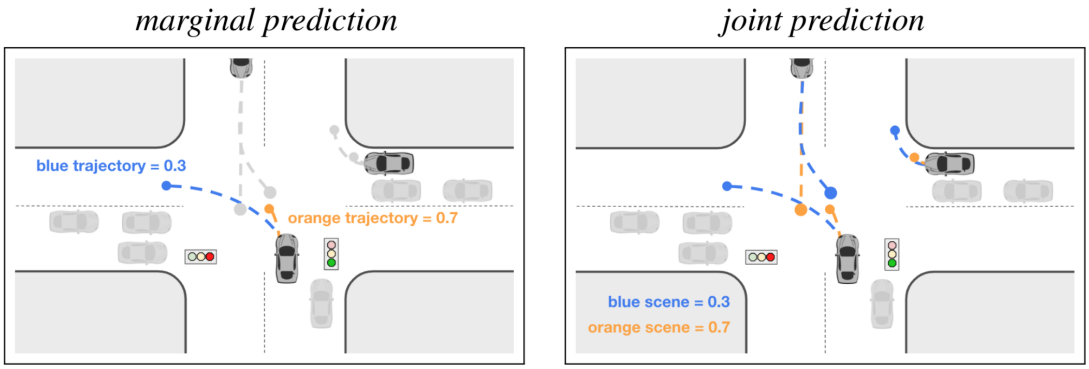


Figure 2.3: The figure depicts differences between marginal and joint motion prediction. Left: Marginal prediction of bottom center vehicle; score indicates likelihood. The prediction is done independently without considering the other agents, resulting in inconsistencies. Right: Joint prediction of all the agents present in the scene. The score indicates the likelihood of the entire scene, i.e., the future of all agents present, resulting in a consistent future. Courtesy: *Ngiam* et al. [30].

of information at input compared to rasterization, and a GCN effectively captures the long-distance dependency of lane graphs compared to CNN [27]. Hence, a similar approach is now widely used in modern models to encode the context of the scene [11, 32].

The other important aspect of motion models is agent-context encoding. Cui et al. [8] models the agent's history as a multichannel top-down rasterized image, and relationships between scene-context and agents are captured with the help of CNN. Another set of methods draw inspiration from the NLP domain, i.e., to model the agent trajectory as a sequence of states and encode information about it using recurrent neural networks (RNNs) [25]. Gao et al. [11] proposed an approach to model agent trajectories as graphs because trajectories are nothing more than directed splines that can be efficiently represented as graphs, and the characteristics are then extracted using graph neural networks (GNN). Alahi et al. [1] presented the first approach to modeling agent relationships, using the aggregation layer to capture interaction between entities, and many studies have adopted similar strategies to model interaction [11, 25]. More recent methods rely on techniques such as soft-attention [33] or GNN [27]. Due to the uncertainty of the task, prediction models often generate multiple plausible future trajectories for nearby actors in the environment. Consequently, for the thesis, we select a multi-modal approach. However, most existing models are formulated to predict each agent's future trajectory independently, termed marginal predictions. This is computationally inefficient and might produce inconsistent results [18].

Recent approaches address this limitation by focusing on multi-agent joint prediction for consistent future trajectory generation [18]. This method, known as joint prediction or scene-consistent prediction, assumes that well-learned in-

teractions during the encoding stage implicitly contain information about future states [17], thereby aiding in the planning task [18]. Figure 2.3 depicts the differences between joint and marginal prediction and highlights inconsistencies arising from marginal prediction. Ngiam et al. [30] use attention mechanisms to extract scene context, facilitating smooth trajectory learning and interactions among agents. They decode future predictions of other agents and AV plans simultaneously using an multilayer perceptron (MLP).

Similarly, He et al. [15] extract agent and scene encoding through different encoders, passing the information to parallel interaction modules for diverse modes. MLP-based decoders then generate future predictions for surrounding agents simultaneously. Girgis et al. [12] employs latent-variable formulation to capture multi-modality, i.e., they use seed parameters to encode each mode and outputs the multi-agent joint prediction for each mode simultaneously. In contrast, some approaches explicitly model future interactions among agents to formulate the problem of multi-agent joint prediction. For instance, Rowe et al. [32] addresses the joint motion prediction problem by formulating it with a sparse-directed acyclic graph interaction, encoding future interactions in this sparse interaction graph. In this thesis, we extend upon the approach of Rowe et al. [32], leveraging its superior performance on competitive benchmarks, quicker inference, and the capability to generate predictions aware of future interactions.

In multi-modal motion prediction models, generating multiple future trajectories and selecting the most likely trajectory is crucial. It is essential for evaluating the prediction model's performance, particularly when assessing the best mode among all the modes it produces according to benchmarks and for downstream tasks such as planning. To address this challenge, researchers have proposed various approaches. One approach involves forwarding the encoded context to an MLP layer, which produces a logit value. When used with softmax, this logit value yields the likelihood of the scene. The likelihood can then be used to select the most likely trajectory [30, 15]. Some methods directly rank the trajectories based on L2 distance to the ground-truth data [32]. Nevertheless, the main drawbacks of the current evaluation/scoring criteria include their lack of interpretability, the suboptimal performance of L2 distance metric at intersections as highlighted by [8], the unavailability of ground-truth data in real-world scenarios, and more importantly, the evaluation/scoring criteria being poorly aligned with the downstream task of planning. On the other hand, we use an explicit decision cost module on the prediction module to evaluate the several modes. This module considers the essential planning cost functions that align well with subsequent tasks and provide interpretability.

As mentioned earlier, effective motion prediction models use past trajectories of agents and environmental information to foresee their future states. However,

for AV, there's a vital piece of information often overlooked by many approaches—the goal information, indicating where the AV intends to be in the future. Some methods exist that use this information. Rheinhart et al. [31] introduces one of the pioneering approaches, conditioning the prediction of other agents in the scene based on the robot's state. Ngiam et al. [30] employs a similar method by masking the future goal state of the AV while encoding the agent context. In contrast, Song et al. [35] takes a different approach, initially generating explicit trajectories for the AV, considering the goal. Subsequently, these trajectories serve as input to the prediction model, resulting in a conditioned output incorporating the AV plan. However, the mentioned approaches condition predictions based on a specific end-state for the AV without considering the possibility of multiple ways to achieve a high-level goal, such as going straight or turning left. Additionally, relying on a fixed end-state may not align with practical scenarios. Hence, in this thesis, we acknowledge that we cannot obtain information about the exact location where AVs may end up. Therefore, our framework focuses on goal lanes rather than a specific end-state or initial plan.

In summary, this thesis extends the existing scene-consistent model by incorporating a decision cost module to assess the evolution of predicted traffic scenes using predefined cost metrics. Importantly, we assign a cost to each predicted scene based on the anticipated collision with other participants in the scene, adherence to traffic rules, and integration of goal information into the decision cost module. This approach, which ranks scenes based on predefined costs, offers interpretability and explainability regarding how a scene might unfold. Finally, AVs trajectory corresponding to the lowest scene cost can be forwarded to a planner for ego-motion planning. Additionally, we hypothesize multiple possible ways to integrate high-level goal information without relying on an exact specific state.

Chapter 3

Fundamentals

This chapter introduces the basic techniques and fundamental concepts used in the thesis. In Section 3.1, we cover the basics related to the chosen approach for motion prediction. Following that, Section 3.2 describes the loss functions that were used to train the motion predictor in this thesis. Lastly, Section 3.3 introduces graph-search techniques and provides a brief overview of how we use HD-Map as a road-network graph in our thesis.

3.1 Model Fundamentals

In this thesis, we adopt the approach presented by Rowe et al. [32] as the scene-consistent model predictor. The scene-consistent predictor aims to take input from the upstream tasks and forecast the consistent motion of all agents in the given scenario. In this section, we provide a brief introduction to the fundamental concepts crucial to the predictor model. We also highlight the context in which these concepts are applied in the approach by Rowe et al. [32].

3.1.1 Direct Acyclic Graph

A directed acyclic graph (DAG) is a directed graph that does not have closed loops or cycles within the graph. This acyclic nature ensures a clear and distinct ordering among the nodes, making it particularly useful for representing structures with a definite flow or hierarchy. Each node represents an entity in a DAG, and directed edges between nodes denote relationships or dependencies. The acyclicity prevents the possibility of entering into an infinite loop when traversing the graph. This property is crucial for applications where a well-defined order or precedence among elements is necessary.

DAGs find widespread use in various domains, including but not limited to

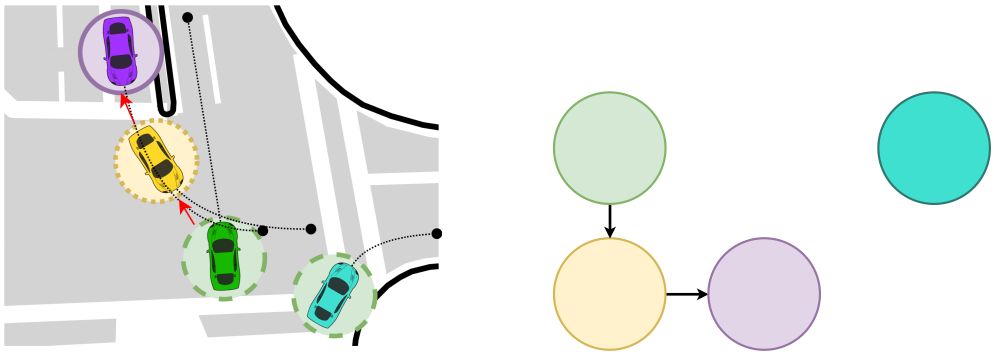


Figure 3.1: Overview of influencer-reactor relationships modeled through directed acyclic graph (DAG). Left: Future trajectories of agents in ground truth; Right: DAG representing influencer-reactor relationships based on ground truth. Here, nodes are agents, and edges represent the influence of one agent on another. Courtesy: *Rowe* et al. [32].

computer science. They are employed to model dependencies between tasks, represent data flow in computational systems, and visualize the relationships within complex structures. The acyclic nature of DAGs simplifies reasoning about cause-and-effect relationships and facilitates efficient algorithm sorting and traversal. In this thesis, the selected motion-prediction architecture uses DAGs to model influencer and reactor relationships among agents as shown in Figure 3.1.

3.1.2 Graph Neural Networks

GNNs are a class of machine learning models designed for data that can be represented as graphs. GNNs leverage the graph structure to learn and generalize patterns in data, making them especially effective for tasks involving relational information. GNNs operate by aggregating information from neighboring nodes, allowing them to capture local structures and propagate information across the entire graph. This ability to consider the relationships between nodes distinguishes GNNs from traditional neural networks.

In motion prediction, approaches employing GNNs often construct a graph representation where vehicles are represented as nodes. The interactions between vehicles are then modeled through directed edges, specifically connecting vehicles that interact based on a predefined distance threshold. This graph-based representation allows the GNNs to capture the spatial relationships and dependencies among vehicles, enabling effective modeling of their dynamic interactions in given scenarios.

In recent years, a new approach for directed acyclic graph neural network (DAGNN) [36] has been developed to leverage hierarchical and relational data presented in the form of a DAG. In the aforementioned approach, information is processed layer by layer, following the natural flow of the graph. Specifically, the approach involves aggregating and combining data from predecessors

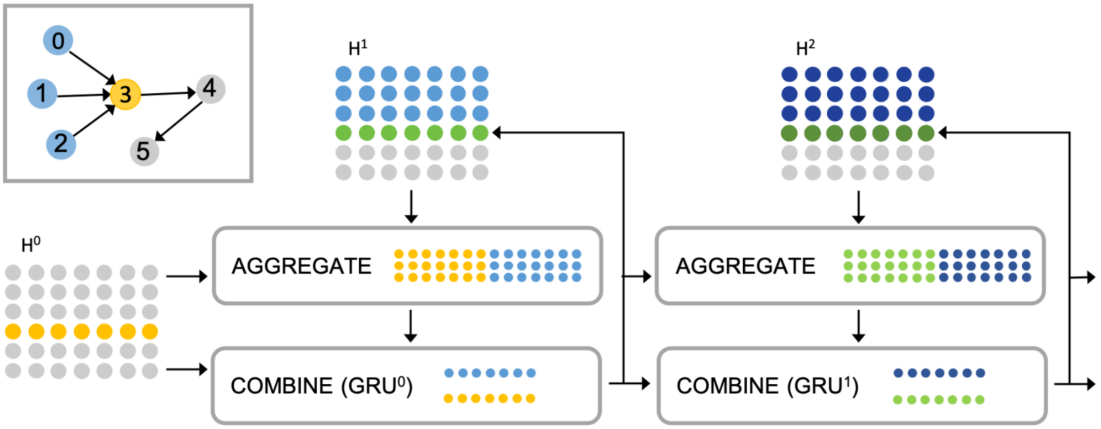


Figure 3.2: The figure illustrates the processing of node 3 in directed acyclic graph (DAG). The initial hidden representation is $\mathcal{H}^0$. In the first layer, representations are already computed for 0, 1, and 2 nodes (blue) and are aggregated first with the initial representation (orange). The GRU treats this aggregated information as the hidden state and initial representation as input and outputs the updated representation (green), which is input to immediate predecessors four and subsequent layers. Courtesy: *Thost* et al. [36].

to update the representation of the current layer as shown in Figure 3.2. This layer-wise processing ensures that information is propagated in a consistent and ordered manner, making DAGNNs effective for capturing dependencies in structured data represented by directed acyclic graphs. In the context of the selected motion predictor, the authors use this network to condition the reactor's prediction on influencer actions, aligning with the structure of DAGNN with some minor modifications.

3.2 Loss Functions

In learning-based models, the goal is to learn a function f_θ that maps inputs to outputs in the output space. In this subsection, we define input samples as $X = \{x_1, x_2, \ldots, x_N\} \in \mathbb{R}^{D_i}$ and the target outputs are denoted as $Y = \{y_1, y_2, \ldots, y_N\} \in \mathbb{R}^{D_o}$, where D_i and D_o represents dimensions of input space and output space respectively.

Given the inputs X, outputs Y, and the parametrized function f_θ, the loss function is defined to measure dissimilarity between the learned mapping and given outputs Y. The loss function over all the input samples is given by

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{i=N} \ell\left(f_\theta(x_i), y_i\right), \tag{3.1}$$

where, y_i and $f_\theta(x_i)$ are true and predicted values for given sample i respectively. Hence, the goal of learning-based models is to solve optimization problem defined

as

$$\theta^* = \min_{\theta} \mathcal{L}(\theta) \tag{3.2}$$

where, θ^* are optimal parameters of parametrized function f_θ that minimize the loss function given by $\mathcal{L}(\theta)$. Thus, the design of the loss function is crucial to learning the appropriate mapping function f_θ. The primary tasks involved in our selected model architecture are regression and classification. In this section, we will describe loss functions related to both tasks and explain the design choices by Rowe et al. [32] to train the model.

3.2.1 Regression Loss

The problem of regression can be defined as finding function f_θ that explains our target outputs Y for given input space X. From a motion prediction perspective we can think of it as given past states find mapping function that predicts the future state of a given agent. The typical function used for such tasks is L2 regression loss for a given sample i, which is formally defined as,

$$\ell\left(f_\theta(x_i), y_i\right) = \left(y_i - f_\theta(x_i)\right)^2. \tag{3.3}$$

In simplest terms, it measures how close the predicted values $f_\theta(x_i)$ are to target values y_i. However, when the regression targets are unbounded, training with L2 loss requires careful tuning of the learning rate. Otherwise, it might result in exploding gradients and unstable training. To overcome this sensitivity, Girshick [13] proposed smooth_{L1} loss as define in Equation (3.5).

$$\ell\left(f_\theta(x_i), y_i\right) = smooth_{L1}\left(y_i - f_\theta(x_i)\right) \tag{3.4}$$

where,

$$smooth_{L1}(x) \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ |x| - 0.5, & \text{otherwise} \end{cases}. \tag{3.5}$$

Thus, smooth_{L1} combines the benefits of L1 and L2 loss, and its more robust to outliers. Given the unbounded nature of the target in motion prediction problem and to prevent exploding gradients, Rowe et al. [32] uses the loss defined in Equation (3.4).

3.2.2 Classification Loss

The classification problem for learning-based models can be defined as given an input $X \in \mathbb{R}^{D_i}$, determine class/label $Y \in \{0, 1, \ldots, K-1\}$, where K is the number of different classes/labels available in the target set and D_i are dimensions of input space. Typically, we formulate f_θ as a conditional probability distribution,

and then the problem can be seen as finding $p\left(y|x;\theta\right)$ that determines unseen class/label given $X \in \mathbb{R}^{D_i}$. Assuming $\forall x_i \in \mathcal{X}$ are independently and identically distributed (IID), conditional likelihood over training samples is formulated as,

$$P(Y|X;\theta) = \prod_{i=1}^{i=N} p\left(y_i|x_i;\theta\right) \tag{3.6}$$

Furthermore, assuming $p\left(y|x;\theta\right)$ follows Bernoulli distribution, the Equation (3.6) can be rewritten as Equation (3.7)

$$P(Y|X;\theta) = \prod_{i=1}^{i=N} \left(f_\theta\left(x_i\right)\right)^{y_i} \dot{\left(}1 - \left(f_\theta(x_i)\right)\right)^{1-y_i} \tag{3.7}$$

where, $f_\theta(x_i)$ gives predicted probability for $y_i = 1$. Then, the maximum-likelihood estimation problem can be formulated as

$$\begin{aligned} \theta_{MLE} &= \underset{\theta}{\operatorname{argmax}}\, P(Y|X;\theta), \\ &= \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{i=N} \left(f_\theta\left(x_i\right)\right)^{y_i} \dot{\left(}1 - \left(f_\theta(x_i)\right)\right)^{1-y_i}. \end{aligned} \tag{3.8}$$

We take the log-likelihood of Equation (3.8) to account for numerical instabilities. The objective can then be maximizing log-likelihood, which can also be seen as minimizing the negative of log-likelihood as given in Equation (3.9). This negative-loss likelihood is known in the literature as cross-entropy loss; more precisely, the loss in Equation (3.9) is known as binary cross-entropy loss.

$$\ell_{BCE} = -\sum_{i=1}^{i=N} \left(y_i \log(f_\theta(x_i))\right) + \left((1-y_i) \log(1 - f_\theta(x_i))\right). \tag{3.9}$$

We can use the softmax function over K (number of classes) to extend binary cross-entropy loss to multi-class classification. The softmax function converts the scores to a probability distribution over the K classes. Thus, multi-class classification loss is defined as,

$$\ell_{cross-entropy} = -\sum_{i=1}^{i=N} \sum_{j=1}^{j=K} (y_i \log(f_\theta^j(x_i))) \tag{3.10}$$

in which,

$$f_\theta^j(x_i) = \frac{e^{s_i^j}}{\sum_{k=1}^{k=K} e^{s_i^k}} \tag{3.11}$$

where, s_i^j is the predicted score for j^{th} class for the i^{th} sample, $y_i^j = 1$ if i^{th} sample belongs to target class j. However, the loss defined in Equation (3.10) and Equation (3.9) gives equal weights to all the classes and does not consider

the class imbalance problem present in the data; the given data is skewed. These loss functions might not work well. To tackle this problem, many different weights were introduced, out of which one method proposed by Lin et al. [28] is considered as the state-of-the-art method to tackle extreme class-imbalance problems which is defined as,

$$\ell_{focalloss} = -\sum_{i=1}^{i=N}\sum_{j=1}^{j=K} \alpha\ (1 - f_\theta^j(x_i))^\gamma (y_i\ \log(f_\theta^j(x_i))) \tag{3.12}$$

where, α and γ are modulating parameters. By adding those modulating parameters, focal loss down-weights the contribution to loss from easily classified samples. Rowe et al. [32] uses the focal loss function to train their interaction classifier. They chose this method because, in many scenarios, most agents don't interact; only one or two agents do. This creates an imbalanced dataset, where one class is much more common than the other. The focal loss helps to handle this imbalance and improves the training of the interaction classifier.

3.3 Graph Fundamentals

In this thesis, we assume we have HD-Map data associated with each local scenario available. This map is an important prior for the motion-prediction module of our framework. Additionally, we use map information in modeling one of our decision-cost modules. This section introduces how we process and convert HD-Map information into a road-network graph. Then, we discuss the graph traversal algorithm, a fundamental algorithm for one of our decision-cost modules.

Processing HD-Map: The HD-Map associated with each scenario contains 3D lane-level details, such as lane boundaries, lane-marking type, traffic direction, cross-walk, drivable area polygons, and intersection annotations [39], highlighted in Figure 3.3. The open-source API associated with the dataset provides essential methods to extract information from the following HD-Map. Since we are only interested in the lanes that the vehicle can end up in, we iterate only through the lane segments in a 50 m search radius from the last-observed ego-vehicle coordinates. This design choice makes road-graph creation for subsequent processes computationally efficient. We convert this information to road-graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where the vertices $\mathcal{V}$ corresponds to N_{lanes} lane-segments present in the scenario, and the directed edge $\mathcal{E}$ contains information about path connecting two nodes, and neighborhood information of the lane-segment as shown in Figure 3.4.

Graph Search Algorithms: In the domain of graph theory, a graph-search problem involves navigating through a graph structure to find a specific target or to explore the graph's connectivity. In this thesis, we employ a graph search

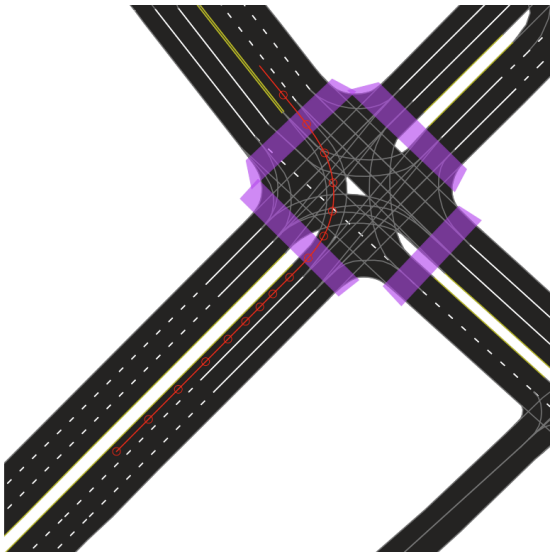


Figure 3.3: The figure shows different components of HD-Map. Lane, lane boundaries (dashed white, double yellow, etc), cross-walk (purple). Lane boundaries that vehicles are likely to follow are shown in gray. The path of the ego vehicle is shown in red. Courtesy: *Wilson* et al. [39].

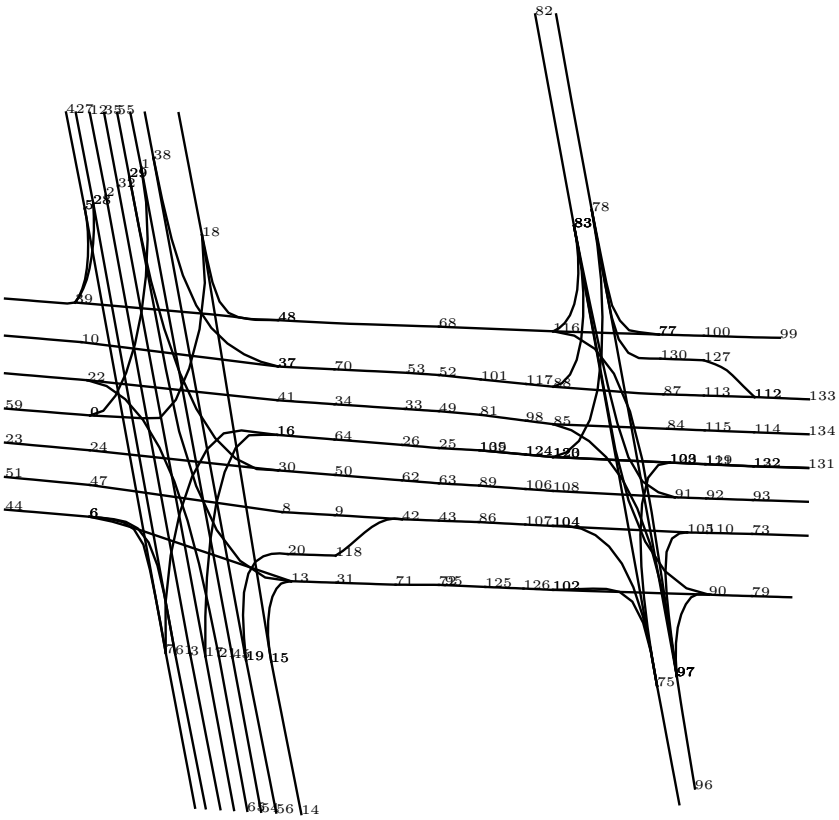


Figure 3.4: The figure shows road-network graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. Here, the numbers represent nodes $\mathcal{V}$, and a line connecting two nodes presents edge $\mathcal{E}$; the information about points connecting two nodes is also stored as edge information.

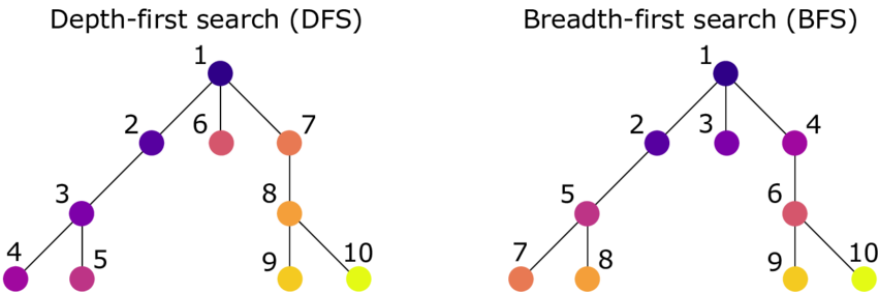


Figure 3.5: Overview of BFS and DFS. The number represents the order of exploration starting from the source nodes. Courtesy: *Wéber* [40].

technique to find and explore all possible successor lanes in the provided road-network graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ that correspond to potential lane segments where our AV can end-up in, considering the high-level goal. Several traversal algorithms exist, each with its characteristics and applications. Noteworthy methods include Depth-First Search (DFS) and Breadth-First Search (BFS).

DFS is a fundamental graph traversal algorithm that explores a graph as deeply as possible along each branch before backtracking. This method involves systematically visiting vertices, marking them as visited, and then exploring as deeply as possible along each branch before backtracking when necessary. The exploration continues recursively until all reachable vertices have been visited.

In contrast to DFS, BFS is another fundamental graph traversal algorithm with distinct characteristics. While DFS explores as deeply as possible before backtracking, BFS is designed to systematically visit all nodes within the same level before moving on to the next level. BFS uses a queue to manage the order in which vertices are visited. Beginning from a source vertex, BFS systematically expands its exploration to adjacent vertices, marking them as visited and enqueuing them for further exploration. This process continues until all reachable vertices have been visited. The pseudocode for BFS is given in Algorithm 1. The

Algorithm 1 Breadth-First Search (BFS)

```
1:  procedure BFS(G, s)                    ▷ G is the graph, s is the source vertex
2:      visited ← {s}                      ▷ Set to keep track of visited vertices
3:      queue ← [s]    ▷ Queue for BFS traversal starting with the source vertex
4:      while queue is not empty do
5:          v ← dequeue(queue)  ▷ Dequeue a vertex from the front of the queue
6:          for u in neighbors of v do
7:              if u is not in visited then
8:                  Mark u as visited
9:                  Enqueue u into queue
10:         process(v)                     ▷ Perform any desired operation on the vertex
```

time complexity of BFS is O(V + E), where V is the number of vertices and E is the number of edges. The space complexity is O(V), accounting for the storage of the queue and the set of visited vertices.

In this thesis, we opt for BFS due to its systematic exploration, aligning with our goal of identifying all potential lane segments where the AV can end up. Unlike DFS, BFS ensures that all vertices at the current level are visited before moving on to the next level, offering an organized and comprehensive exploration of the graph's structure.

Chapter 4

Approach

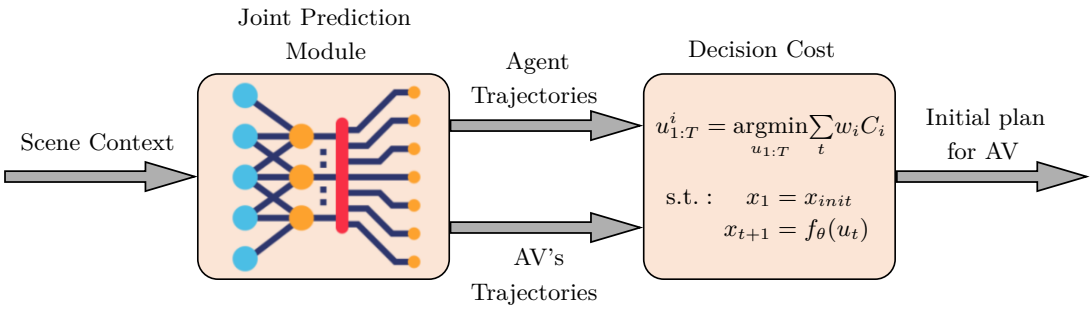


Figure 4.1: An overview of our approach. We first generate consistent scenes using a scene-consistent motion-predictor. These scenes are then systematically ranked based on predefined cost metrics. The trajectory of an autonomous vehicle (AV) associated with the scene with the least cost is subsequently chosen as the initial plan for downstream modules.

In this chapter, we introduce our approach that evaluates the consistent scenes resulting from the scene-consistent motion predictor and selects the optimal scene according to predefined decision cost, thus aiding the planning task. Figure 4.1 depicts a block diagram of essential sub-tasks in our pipeline. Section 4.1 formulates the problem of motion-prediction mathematically. In our approach, we first generate the scene-consistent prediction from a scene-consistent motion predictor; the selected predictor model is explained briefly in Section 4.2. Then, in Section 4.3, we describe components of our decision cost module, in which we evaluate the predicted trajectory obtained from the predictor. Finally, Section 4.4 explains briefly the overall scene cost which is then used to rank predicted scenes.

4.1 Problem Definition

The objective of multi-agent joint motion-predictor frameworks is to predict the future states of all agents in a given scene using previously observed states.

In this section, we express the joint-motion prediction task mathematically. In a given scenario S, the set of past states for multiple agents is denoted as $X = \{X_0, X_1, \ldots, X_N\}$, where N is the number of agents in the scene, and additional HD-Map context information is denoted as C. The objective of motion prediction is to forecast the future states of these N agents, represented by $Y = \{Y_0, Y_1, \ldots, Y_N\}$.

The past trajectory of an agent i is defined as $X_i = \{x_{i,t-\tau+1}, x_{i,t-\tau+2}, \ldots, x_{i,t}\}$, and the future trajectory is defined as $Y_i = \{y_{i,t}, y_{i,t+1}, \ldots, y_{i,t+\delta}\}$. In these definitions, each element $y_{i,t} \in \mathbb{R}^2$ in the future trajectory represents the coordinates of agent i at time step t. Each element $x_{i,t} = [p_{i,t}, v_{i,t}, \psi_{i,t}]$ in the past trajectory includes coordinates $p_{i,t} \in \mathbb{R}^2$, velocities $v_{i,t} \in \mathbb{R}^2$, and yaw angle $\psi_{i,t} \in \mathbb{R}$ for agent i at time step t. Here, δ and τ refer to the maximum length of time steps for the future and past, respectively.

Considering the uncertainty in the future, the task is typically defined to predict a set of diverse K modes, where each mode consists of the predicted future of each agent in S, which can be represented as $Y_{i,k} = \{Y_{i,0}, Y_{i,1}, \ldots, Y_{i,K}\}$. Most learning-based approaches establish an optimization process to learn the posterior distribution $P(Y|X, C)$ to accomplish this task.

4.2 Joint-Motion Prediction

This thesis primarily deals with ranking consistent scenes resulting from the joint-motion predictor. In this section, we briefly discuss the factorized joint motion prediction (FJMP) approach by Rowe et al. [32], the prediction approach we use to generate consistent evolution of traffic scenes. We selected the following approach because of its competitive results on joint-prediction benchmarks on the INTERACTION dataset challenge [43] and an easy-to-use open-source codebase, making it suitable for our approach. The key contribution of this approach is to model the future interaction of the agents, represented through a DAG. This, in turn, allows the decomposition of the joint prediction task into a sequence of conditional and marginal predictions based on the order of the DAG. The underlying hypothesis is that this learning process structure can simplify the complexity of the task at hand. The approach of FJMP [32] can be broadly divided into four parts: (a) feature encoding, (b) auxiliary proposal decoder, (c) directed acyclic graph predictor, (d) factorized joint decoder. In this section, we will briefly discuss the main components of our selected motion-prediction approach.

Feature Encoding: The inputs for the feature encoder are the past trajectories of all agents in the scenario and the associated HD-Map. The initial processing of raw input follows the established approach of LaneGCN [27]. Subsequently, the

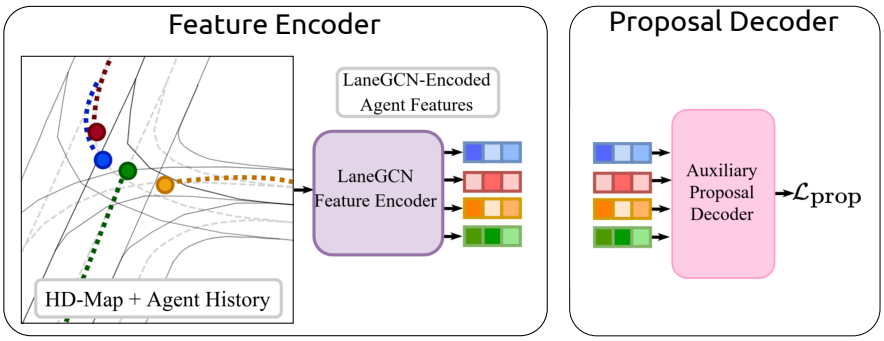


Figure 4.2: Agent past states and HD-Map are taken as input to the feature encoder, which outputs feature representation for each agent. These representations are then used by the proposal decoder to generate predictions during the training stage. Courtesy: *Rowe* et al. [32].

past trajectories of all agents are centered and rotated around the AV before being fed into the feature encoder. The feature encoder is inspired by LaneGCN [27] with minor modifications that reduce the parameters, which leads to faster inference. The output from the feature encoder is a *context and spatially aware* feature representation of D dimensions for all the agents in the scene. This principle is illustrated in Figure 4.2.

During training, the representation from the feature encoder is forwarded to the proposal decoder for predicting the joint future trajectories, explicitly supervised with a $smooth_{L1}$ loss denotes as $\mathcal{L}_{\text{prop}}$ in Figure 4.2. This pretext task helps generate *future-aware* feature representation. Notably, the proposal decoder is not utilized during inference, as its purpose is only to enrich the features extracted from the feature encoder.

Direct Acyclic Graph Predicted (Interaction Graph Predictor): To construct the directed interaction graph $\mathcal{G}$, a fully connected undirected interaction graph $\mathcal{G}_U = \{\mathcal{V}_U, \mathcal{E}_U\}$ is initially created, where each agent corresponds to a node in $\mathcal{V}_U$. These nodes are initialized using the representations from the feature encoder. A classifier is trained to label each edge $e_{m,n} \in \mathcal{E}_U$ as either `no-interaction`, `m-influences-n` or `n-influences-m`, supervised by focal-loss. The directed interaction graph $\mathcal{G}$ is formed based on the predicted interaction labels, incorporating directed edges according to the predicted influencer and reactor, with no edge added otherwise. The directed interaction graph $\mathcal{G}$ is converted to a DAG to achieve a factorized joint prediction. This is obtained by iterative cycling through the graph $\mathcal{G}$ and removing edges with the lowest predicted probability, leveraging the efficient Johnson's algorithm [21]. Figure 4.3 gives an overview of the interaction graph predicted as used in FJMP.

Factorized Joint Decoder: The unique partial ordering of $\mathcal{G}$ and the future-aware representations from the feature encoder are inputs to the decoder that produces joint predictions using a modified DAGNN. The process involves the

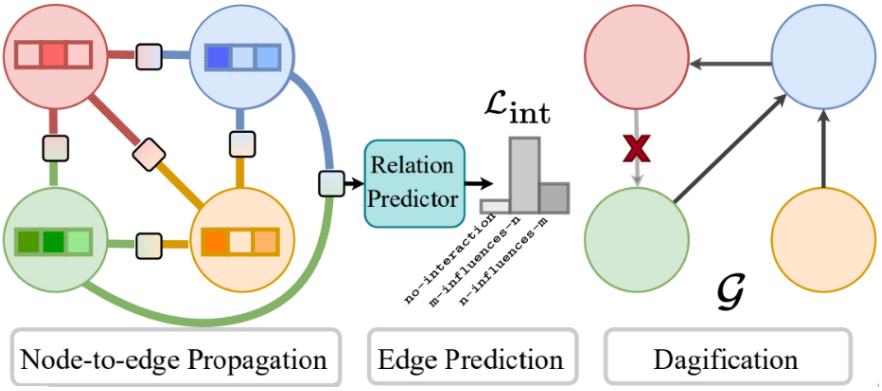


Figure 4.3: The figure illustrates the key steps in the directed acyclic graph (DAG) predictor. Nodes in the graph represent agents in the scene with their respective feature representations denoted by different colors. The resulting DAG $\mathcal{G}$ captures the hierarchical influence relationships among agents, guiding the conditional future predictions for each agent in the scene. Courtesy: *Rowe* et al. [32].

parallel decoding of all source nodes $\mathcal{S}_G$ to obtain marginal predictions. The marginal predictions are then encoded using an MLP layer. These encoded features of the source node $\mathcal{S}_G$ are used to update the features of nodes in the subsequent partial ordering of $\mathcal{G}$. The combined feature of each node contains context about the predicted future of its parent node(s). This updated feature is then passed onto the DAGNN to generate conditional future predictions for the respective agent. This iterative process continues until all the graph $\mathcal{G}$ nodes have factorized joint predictions. For generating K joint future predictions, K copies of the feature representation for each agent are created and processed in parallel. A one-hot encoding representing each mode is added to the feature embedding for each agent to account for mode collapse and maintain diverse modes. For this thesis, $K = 6$ i.e., output from the predictor model will be 6 consistent scenes. The following steps are shown using DAG of size three in Figure 4.4.

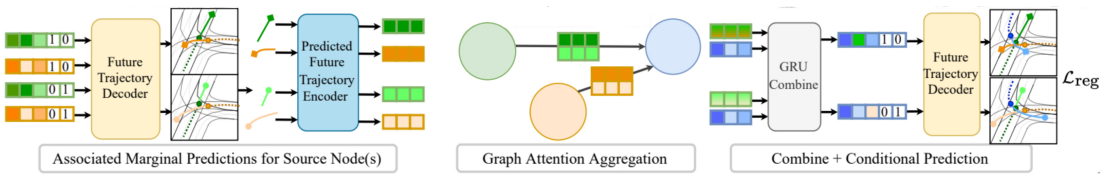


Figure 4.4: Illustration of the decoder in FJMP. Green and orange agents represent influencers for the blue agent. The process involves generating future predictions for influencers (source), combining their representations using Graph Attention Aggregation, and conditioning the feature representation of the blue agent with this information to produce conditional future predictions (Combine + Conditional Prediction). Courtesy: *Rowe* et al. [32].

4.3 Decision Costs

Since the model outputs scenes consistently, the predicted trajectories of all agents in the scene, including AV, are implicitly conditioned on one another. This implies that the predicted trajectories of surrounding agents are conditioned on the potential future plan of the AV. We introduce an explicit decision cost module inspired by classical planning approaches to evaluate and rank these predicted traffic scenes. We hypothesize that selecting the AV 's trajectory corresponding to the least-cost scene will provide a more advantageous starting point for downstream tasks, such as planning. This decision-cost module comprises various cost terms that encode different aspects of driving primarily for AV, including ride comfort costs, collision costs, and, most importantly, goal-following costs. In this section, we will briefly discuss each of the following cost functions in detail.

4.3.1 Ride Comfort Costs

Ensuring ride comfort is a primary concern for any vehicle, as riders typically prefer smooth and comfortable maneuvers [18]. To quantify this comfort aspect, we introduce a cost metric associated with the predicted trajectory's longitudinal acceleration $\hat{v}$. This metric allows us to assess the predicted trajectories in terms of their impact on ride comfort, aligning with the passenger's preference for a comfortable driving experience. The longitudinal acceleration limits are $[-5, 5]\, m/s^2$. The effect of the cost function on a scenario from the dataset is shown in Figure 4.5.

$$c_{acc} = \frac{1}{\delta} \sum_{i=t}^{i=t+\delta} acc_{cost}(\hat{v}_i) \tag{4.1}$$

in which,

$$acc_{cost} = \begin{cases} 0, & \text{if } -5 \leq \hat{v} \leq 5 \\ (|\hat{v}| - 5)^2, & \text{otherwise.} \end{cases} \tag{4.2}$$

4.3.2 Collision Costs

To prevent collisions, ensuring a safe distance from other traffic participants is a fundamental requirement for AVs. We explicitly model a collision cost to guarantee that the selected trajectory avoids collisions with other agents in the scenario. In this thesis, we modify the approach from the INTERACTION dataset [43]. A list of circles defines each agent according to radius r_i and r_j, and the collision threshold ϵ_C is determined as per Equation (4.3). Unlike the INTERACTION approach, we define a non-binary cost function. The cost is computed by calculating the Euclidean distance between agent i and j at each time step. The

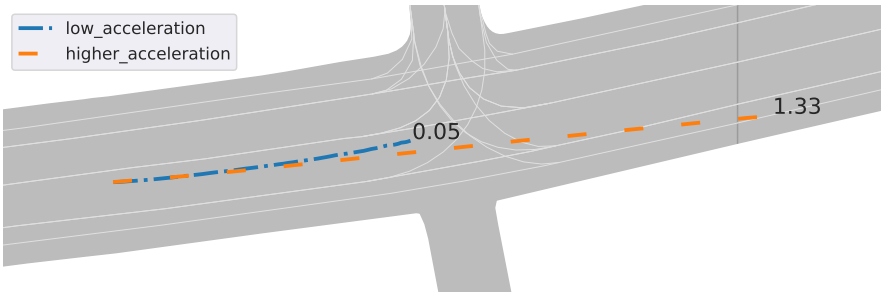


Figure 4.5: A figure representing different predicted modes and their corresponding acceleration cost as per Equation (4.1).

minimum distance, d_{min}, is then selected from all time steps. Given, d_{min} and ϵ_C, the cost is calculated as per Equation (4.4)

$$\epsilon_C = \frac{r_i + r_j}{\sqrt{3.8}} \tag{4.3}$$

where, r_i and r_j are defined according to the agent dimensions as given in the dataset.

$$c_{collision} = \begin{cases} \left(1 - \frac{d_{min}}{\epsilon_C}\right)^3, & \text{if } d_{min} \leq \epsilon_C \\ 0, & \text{otherwise.} \end{cases} \tag{4.4}$$

Figure 4.6 illustrates detection of collision as per Equation (4.4). In scenario involving multiple agents, after calculating the collision cost concerning a specific agent i with all the other agents in the scene, the maximum of these cost is selected. This choice is made to encode the highest possible cost that a predicted trajectory may incur and mitigate its impact on downstream tasks.

4.3.3 Goal-following Costs

We have another critical information for the AV - the planned route or the high-level goal it aims to reach. This information outlines the specific path or objective the AV intends to follow. This high-level goal is essential as it provides context for the vehicle's planned trajectory. We aim to capture this information by including

Figure 4.6: A figure showing the collision of two agents at a certain time step of predicted trajectories. The specific time step is highlighted by circles.

a goal-following cost component in our decision-cost framework. This integration allows us to understand better and evaluate the AV's predicted trajectories in relation to its goal, contributing to more informed and goal-aware decision-making processes.

As discussed in Section 2.2, one option that can be used is to select trajectories based on their Euclidean distance from the goal point (assuming it is known, which may not always be the case). However, this method may lead to inaccurate rankings, particularly in intersection scenarios, as illustrated in Figure 4.7. Alternatively, our approach integrates goal information by assessing the overlap between the forward reachability from the predicted end-point and the goal point. This approach acknowledges the possibility of multiple ways to achieve a given high-level goal. The underlying assumption is that if the predicted position and the goal location share the same reachable lanes, the trajectory is well-aligned with the goal direction. Given the size and information about the road network graph, we believe this measure effectively encodes goal-related information.

In our decision-cost module, we define the goal-following cost as a binary cost, either 1 or 0, based on the alignment of the predicted trajectory with the specified high-level goal. The goal-following cost aims to identify all sets of lanes (denoted as L) that are reachable from the high-level goal. The cost is defined such that if the sets of forward-reachable lane segments from the predicted trajectory and the high-level goal overlap, the cost is 0; otherwise, it is 1. To calculate the goal-following cost, we assume the availability of a road-network graph $\mathcal{G}_{map} = \{\mathcal{V}, \mathcal{E}\}$,

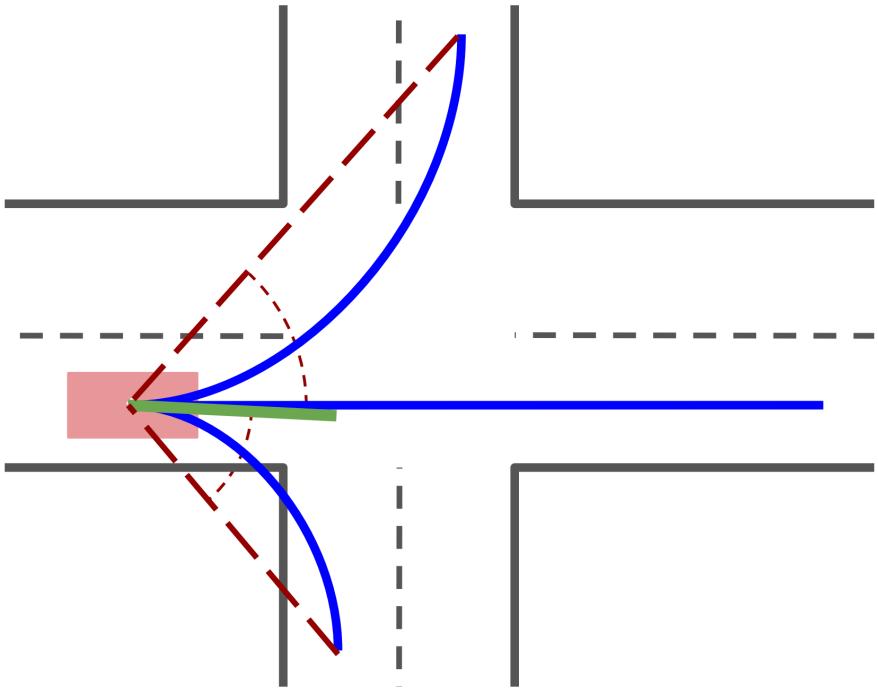


Figure 4.7: Erroneous mode selection (modes shown in blue): according to L2 distance, the right-turn mode is closest to ground truth (green). Courtesy: *Cui* et al. [8].

where nodes $\mathcal{V}$ represent lane-segments in the scenario, and edges $\mathcal{E}$ contain information about neighbors and the path of the lane-segments. Additionally, we have predicted and ground-truth end-points denoted as $\hat{y}_{i,t}$, y_{goal} along with their respective orientations $\hat{\psi}_{i,t}$, ψ_{goal}. This thesis assumes the goal to be $y_{goal} = y_{i,t}$ (the end-point of the ground-truth trajectory for AV in the respective scenario) and associated orientation $\psi_{goal} = \psi_{i,t}$. We assume this information is available, as a navigation planner typically provides a set of lanes or at least knows the high-level goal that AV is trying to reach. The goal-following cost calculation involves two main steps: first, finding the closest assigned lane segments $\forall p \in [\hat{y}_{i,t+\delta}, y_{goal}]$ in the road-network graph, and second, identifying all the successor lanes corresponding to the assigned lane segments. Then, based on the overlap between two sets of lanes, we assign a cost to the given predicted trajectory. The details of each step are explained using the point $p \in [\hat{y}_{i,t+\delta}, y_{goal}]$ and its associated orientation $\psi \in \hat{\psi}_{i,t+\delta}, \psi_{i,t}$ as an example. The overall process of goal-following cost is outlined in Algorithm 2.

1. **Lane assignment:** To determine the reachable set of lane segments, we first identify the closest lane $n_i \in \mathcal{V}$ in the given map for the point p. This involves obtaining all lane segments within a $\pm 45°$ range from the associated orientation ψ to that point. Subsequently, we select the lane closest to the given point using L2 distance. This step of selecting lanes with similar orientation is crucial, as relying solely on L2 distance for the nearest

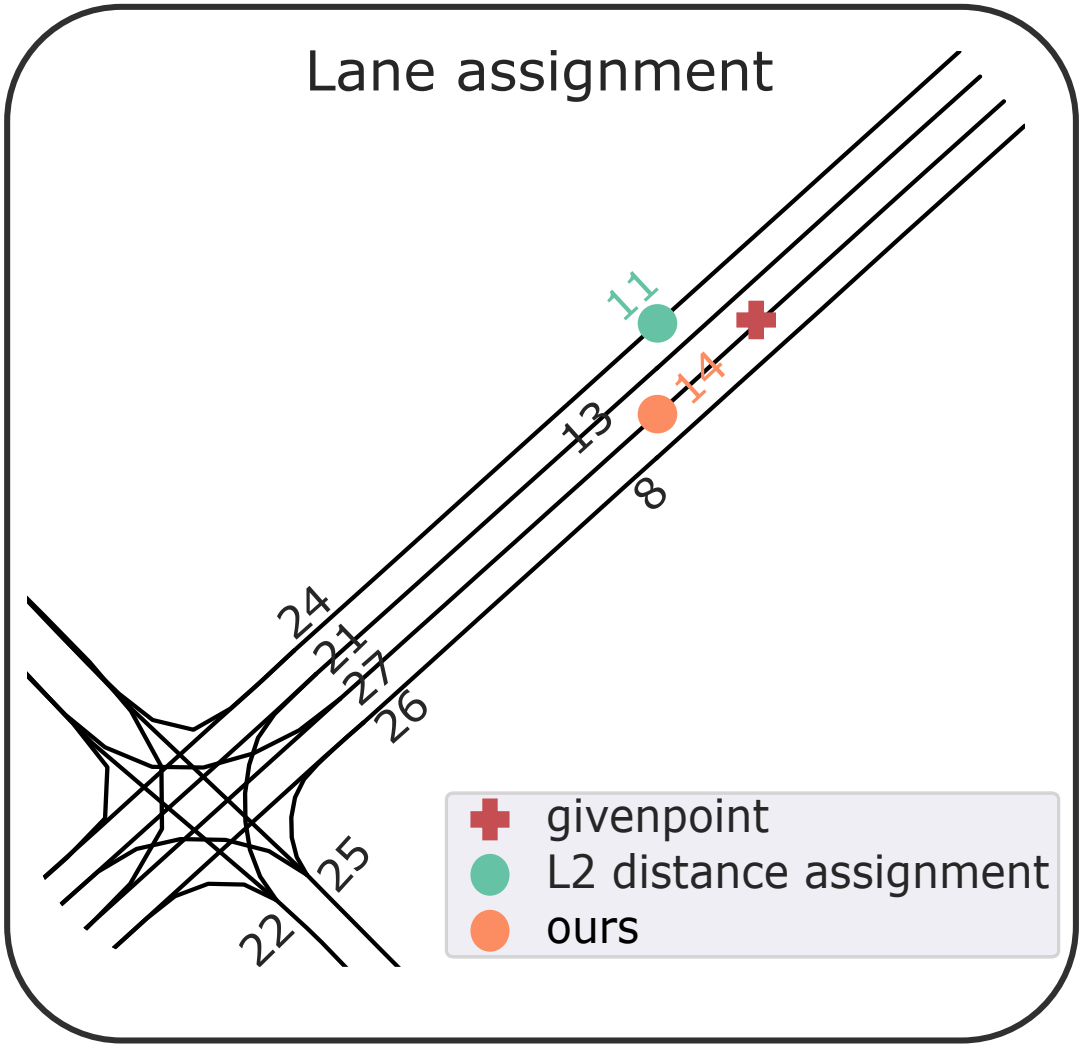


Figure 4.8: The figure shows for given point p lane assignment using different strategies.

lane segments could result in erroneous assignments as seen in Figure 4.8, particularly in intersection scenarios.

2. **Finding successor lanes:** After identifying the closest lane segment n_i, we proceed to find all the successor lane segments from v_i using the breath first search (BFS) algorithm, as explained in Section 3.3 . We also keep track of the neighbors of the lane segment n_i. Consequently, the reachable lane set L comprises successor lanes associated with the closest lane segment and their neighbors. We decide to accommodate lane changes while following the goal and consider the possibility of multiple lanes heading in the same direction toward the high-level goal. Figure 4.9 highlights successor lanes given the assigned node.

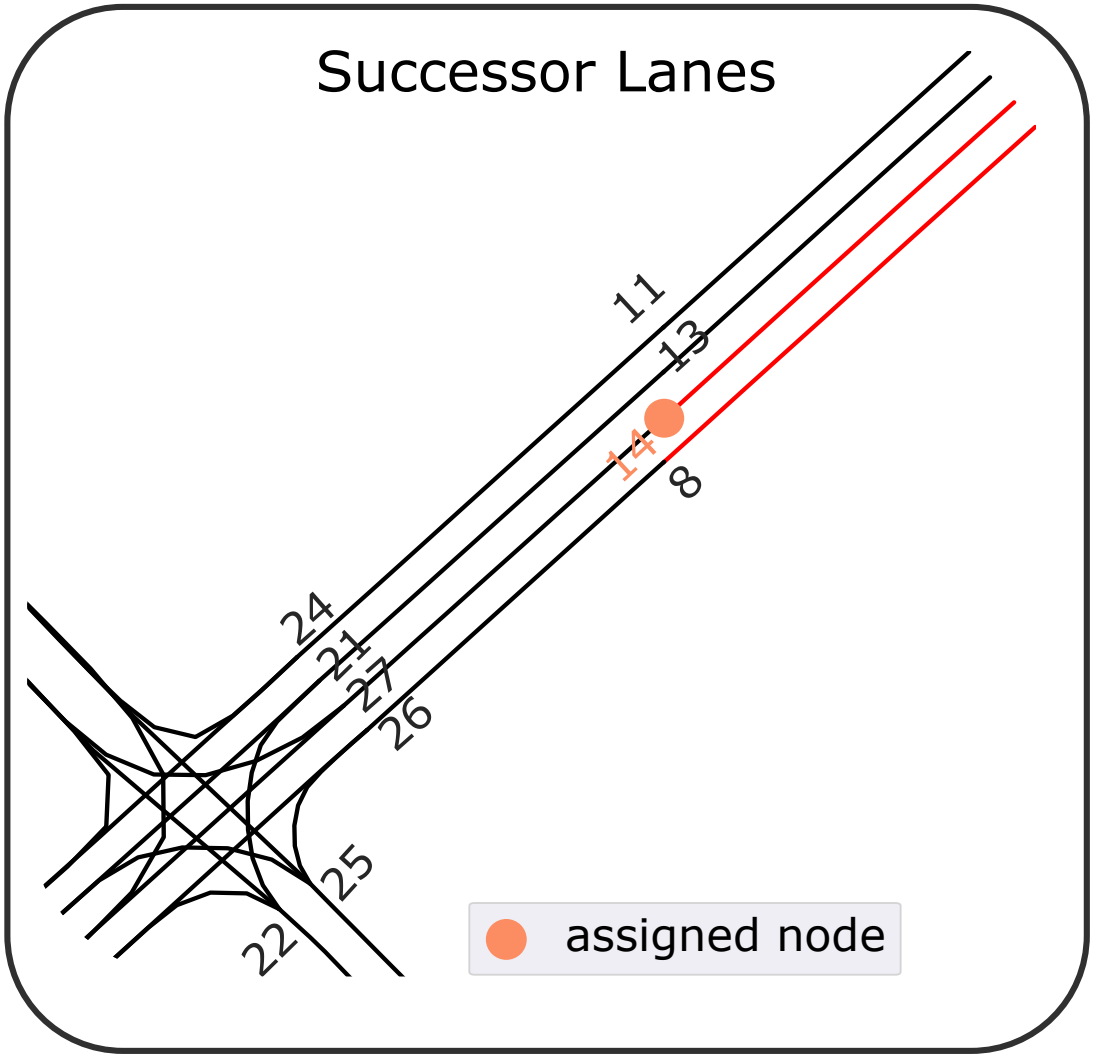


Figure 4.9: In this diagram, we use red color to highlight the successor lanes of the assigned node, which is marked with an orange circle, using our approach.

4.4 Scene Cost

The overall scene cost consists of two components: the cost related to the ego-vehicle C_{ego}, i.e., the AV, and the cost associated with surrounding agents C_{agents}. Including agent cost allows us to capture the impact of the ego-vehicle on the agents in the scene, aligning with game-theoretic approaches where the optimal scenario involves every agent behaving optimally. Each cost component is the weighted sum of various components, as explained in Section 4.3. It's important to note that the agent cost C_{agents} does not include the goal-following cost for obvious reasons – we may not be aware of the plans or high-level goals of the surrounding agents. Hence, the overall scene cost of each predicted scene is as per Equation (4.5).

$$C_{scene} = C_{ego} + C_{agent} \tag{4.5}$$

Algorithm 2 Goal-following cost as proposed in this thesis

Require: Road-network graph $\mathcal{G}$, Points $p \in [\hat{y}_{i,t}, y_{i,t}]$, Orientation $\theta = [\hat{\psi}_{i,t}, \psi_{i,t}]$

$lane\_assigned \leftarrow \emptyset$
for $\forall p \in [\hat{y}_{i,t}, y_{i,t}]$ and $\forall \psi \in [\hat{\psi}_{i,t}, \psi_{i,t}]$ **do**
    $v \leftarrow \textsc{AssignLane}(p, \psi)$
    $lane\_assigned \leftarrow lane\_assigned \cup v$
$\hat{\mathcal{L}} \leftarrow \textsc{FindSuccessorLanes}(lane\_assigned[0])$
$\mathcal{L} \leftarrow \textsc{FindSuccessorLanes}(lane\_assigned[1])$
if $\hat{\mathcal{L}} \cap \mathcal{L} \neq \emptyset$ **then**
    $c_{goal} \leftarrow 0$
else
    $c_{goal} \leftarrow 1$
return c_{goal}

where,

$$C_{ego} = w_a \cdot c_{acc} + w_c \cdot c_{collision} + w_g \cdot c_{goal}$$

$$C_{agent} = w_a \cdot c_{acc} + w_c \cdot c_{collision}$$

Here, w_a, w_c, and w_g are weights for ride comfort, collision, and goal cost, respectively. Thus, we have scene cost corresponding to each predicted scene, i.e., to each mode $k \in K$. The scene/mode with the least cost is then selected as the optimal scene, and the trajectory corresponding to AV is then passed on to downstream tasks.

Chapter 5

Experiments

In this chapter, we present our experiments to highlight the advantages and limitations of our proposed framework. This chapter is structured in the following manner: Section 5.1 provides a detailed overview of the dataset used for validating our hypothesis. In Section 5.2, we briefly outline the architecture details employed in training the model. Subsequently, Section 5.3 offers an overview of the metrics used for evaluation and their significance. Section 5.4 briefly explains the various baselines used to assess our approach. Finally, Section 5.5 and Section 5.6 present results from detailed evaluation of our framework against different baselines both quantitatively and qualitatively.

5.1 Dataset

We train and validate our framework on Argoverse2 [39], a large-scale open-source urban driving dataset. It contains a curated collection of real-world scenarios with highly complex and diverse road structures and traffic interactions. Importantly, it covers a large geographic area spanning six geographically diverse cities. The complexity of different scenes and diverse interactions, especially relevant to the ego-vehicle decision process, makes this dataset stand out in evaluating our decision cost and joint prediction module. The dataset contains 250,000 non-overlapping scenarios. Each scenario has its associated HD-Map, 11 seconds of trajectory data (2D position, velocity, and orientation) of all the tracks observed by ego-vehicle recorded at 10 Hz, and an additional classification label for each track belonging to one of ten unique object category types. For each scenario, the 11 seconds time window is split as a 5 seconds observation horizon and 6 seconds as the prediction horizon. The total number of scenarios used for training is 199,908, while 24,988 are used for validation. Since we do not have access to the testing split from official Argoverse2. We further split the validation set, keeping 16,765 frames for testing and 8203 frames for validation.

5.2 Implementation

We train the motion-prediction module as per details presented by Rowe et al. [32] to reproduce competitive results on the prediction benchmark. Specifically, we set $K = 6$ for the factorized decoder. The hidden dimension of the entire model architecture is 128, with the only exception being the history encoder, which has 256 hidden dimensions. We train our model on NVIDIA GeForce 2080 Super using Adam optimizer [24]. We set the batch size to 32 and train for 36 epochs with a learning rate 1e-3, step-decayed by a factor of 1/10 at epoch 32. The model training on a single GPU takes approximately 10 - 12 hours.

5.3 Metrics

Conducting a thorough evaluation is crucial in scientific research. This section outlines the metrics used for comparing and validating our proposed approach against other baselines. For this thesis, we emphasize upon open-loop evaluation and select metrics that can be computed in an open-loop fashion. These metrics involve comparing the output of our approach or baselines with the trajectory of the human expert driver, which serves as the ground-truth data available from the dataset. The open-loop evaluation is carried out across all scenes in the dataset, assuming the number of scenes to be N_{scene}.

In this thesis, we focus on choosing the best mode from the multiple modes provided by the scene-consistent motion predictor. Therefore, the metrics below can be broadly classified into prediction metrics, which access the performance of the predictor model as we are extending upon a prediction model and planning metrics. This targeted evaluation allows us to assess the performance of our approach.

5.3.1 Prediciton Metrics

We evaluate the predictors on widely used metrics in the domain of motion prediction to assess the performance and the accuracy of the motion prediction models.

Final displacement error (FDE) is defined as per Equation (5.1), where $\hat{y}^k_{i,t}$ and $y_{i,t}$ are the predicted and groundtruth positions respectively of agent i at time t, N_{scenes} represents number of frames, N is number of agents present in the scene, k is mode corresponding to selected trajectory. It quantifies the accuracy of the predicted trajectory by measuring the Euclidean distance between the endpoint of the predicted trajectory and the ground truth for all agents. Here, the predicted trajectory corresponds to the mode selected by our decision cost module referred to in Equation (5.1) as k. In other words, FDE assesses the proximity of the

expected position to the actual position at the end of the predicted time horizon. A lower FDE indicates higher alignment to the ground-truth data.

$$FDE = \frac{1}{N_{scenes}} \sum_{n=1}^{N_{scenes}} \frac{1}{N} \sum_{i=1}^{i=N} \|\hat{y}_{i,t}^{k} - y_{i,t}\|_2. \tag{5.1}$$

Average displacement error (ADE) is defined as per Equation (5.2). It is defined similarly to FDE but instead calculates the average Euclidean distance between the predicted trajectory and ground truth for all the agents in the scene. ADE emphasizes the overall precision of the motion-prediction model.

$$ADE = \frac{1}{N_{scenes}} \sum_{n=1}^{N_{scenes}} \frac{1}{N} \sum_{i=1}^{i=N} \left(\frac{1}{t} \sum_{t=0}^{T} \|\hat{y}_{i,t}^{k} - y_{i,t}\|_2 \right), \tag{5.2}$$

where, N_{scenes} represents number of frames, N is number of agents present in the scene, k is mode corresponding to selected trajectory and T are time-steps corresponding to predicted trajectory.

5.3.2 Planning Metrics

We take inspiration from the state-of-the-art approach by Huang et al. [18] for selecting planning metrics. In this section, we are particularly interested in evaluating selected AV's trajectory corresponding to the best-predicted scene. We assess the following to evaluate suitability for initializing downstream modules such as planning.

Planning Metrics is a metric to quantify human likeliness. This is calculated by comparing the L2 distance between the selected trajectory and the ground-truth trajectory at particular time steps (i.e., 2 s, 4 s, 6 s) as per Equation (5.3). Lower planning errors indicate that the trajectory closely resembles the expert driving trajectory.

$$HDS_{@t} = \frac{1}{N_{scenes}} \sum_{i=1}^{N_{scenes}} \|\hat{y}_{t}^{k} - y_{t}\|_2, \tag{5.3}$$

where t represents the discrete time steps for the comparison. In our thesis, $t = [2, 4, 6]$.

Collision rate is a metric that gauges the safety performance of our approach, which is particularly relevant for downstream tasks. The collision is calculated based on AV's selected trajectory and the ground-truth future trajectories of other agents present in the scenario. The metric is binary, taking values of either 0 or 1, where 0 indicates no collisions detected throughout the trajectory, and 1 signifies a collision at some point. In this thesis, we adopt the approach of INTERACTION [43] dataset, which defines each agent by a list of circles, and

two agents are defined as colliding if the L2 distance between any two circles is lower than the given threshold. The threshold is defined as per Equation (4.3).

Comfort metrics quantifies the rider's comfort and the feasibility of the selected trajectory. Three key quantities are introduced to capture and quantify these comfort metrics: longitudinal acceleration, jerk, and lateral acceleration. These values are computed as averages over time within a given overtime and ardently compared against the expert driving trajectory. By examining and comparing these quantities, we understand how the selected trajectory corresponds to professional driving behavior and provides a comprehensive assessment of the passenger's comfort and the practical feasibility of the trajectory in realistic scenarios.

Final heading errors (FHE) is defined as per Equation (5.4), where $\hat{\psi}_t^k$ and ψ_t are orientation corresponding to predicted and ground-truth trajectories respectively for AV at time t, k is mode corresponding to selected trajectory and N_{scenes} represents total number of frames used for evaluation. It is computed by calculating the absolute error between the heading of the ground-truth endpoint and the selected trajectory endpoint. This metric shows how closely the selected trajectory matches the expert's driving direction.

$$FHE = \frac{1}{N_{scenes}} \sum_{i=1}^{N_{scenes}} \|\hat{\psi}_t^k - \psi_t\| \tag{5.4}$$

Goal-check metric assesses whether the selected trajectory aligns with the high-level goal of the AV, which is crucial for planning purposes. It is a binary metric with values 0 or 1, indicating whether the selected trajectory is not heading in the goal direction (0) or is aligned with the goal (1). The calculation of this metric is based on the approach outlined Section 4.3.3. It measures the reachability of the selected trajectory to the high-level goal. These metrics are calculated for each scene, and the final result is presented as the overall percentage of scenes that adhere to the goal.

Progress metric measures the advancement of the selected trajectory for the AV. It is determined by computing the Euclidean distance between the starting and ending points of the chosen trajectory. A higher value of this metric signifies greater progress in the given time. This metric is computed for each scene, and the final result is presented as the average of all scenes.

5.4 Baselines

In this section, we mention details about the chosen baselines to compare our proposed framework. We compare our framework with the following baselines to address the limitations and advantages of our approach.

FJMP (Goal): In this baseline, we modify the existing FJMP architecture to encode high-level goal information directly into the model architecture. Specifically, we make the following modifications to the network to integrate the goal information. First, during preprocessing, we identify all the successor lanes given the high-level goal for each scenario, taking it as the ground-truth future endpoint of the AV's trajectory. The identification process is similar to what was mentioned in Section 4.3.3. Additionally, we find the route connecting the high-level goal and the last observed endpoint in the ground-truth of the AV's trajectory, using BFS on the road-network graph $\mathcal{G}$ as mentioned in Section 3.3. We aim to encode this information in training, as typically, this information is available through upstream tasks like planning. Thus, the route lanes include the route from the observed ground-truth endpoint to the high-level goal, and all the lanes are then reachable from the high-level goal. We encode the information about route lanes in the given HD-Map through a binary mask.

Additionally, we introduce an additional class representing the AV for each scenario and pass this information as input. We hypothesize that adding this information as input can drive the predictions to align with the high-level goal. The model is trained using the exact specifications and hyperparameters mentioned in Section 5.2, ensuring that results in comparison are fair.

Imitation Learning + Prediction (IL) is based on FJMP, keeping all the specifications and hyperparameters the same as mentioned in Section 5.2. However, instead of predicting multiple modes for agents and AV, we predict only a single mode. This differs from the traditional IL pipeline, which predicts the AV's future trajectory, as it performs prediction as an additional subtask.

IL (Goal) is similar to the IL-based approach, as it also predicts unimodally. However, in this baseline, we modify the model and train FJMP (Goal) to predict unimodal output for agents and AV trajectory.

FJMP + RandomSelection: In this baseline, the explicit decision cost module is replaced by random selection. In other words, the mode is selected based on a discrete uniform distribution, introducing randomness into the mode selection process.

5.5 Quantitative Evaluation

In this section, we list the quantitative evaluations of our method and other baselines mentioned in Section 5.4 based on open-loop metrics defined in Section 5.3. These evaluations are performed on the testing split that consists of 16,785 frames. In Section 5.5.1, we evaluate different components within our decision cost module combined with multi-modal baselines, namely FJMP and

Experiments	Collision ↓ (%)	Goal-Check ↑ (%)	Progress ↑ (m)	FHE ↓ (rad)	Acc. ↓ (m/s^2)	Jerk ↓ (m/s^3)	Lat. Acc. ↓ (m/s^2)	Planning Metrics (m) ↓ @2s	@4s	@6s
FJMP + RS	1.50	88.82	30.932	0.62	1.130	7.473	0.147	0.361	0.868	3.705
FJMP + C	**0.70**	89.12	**31.447**	0.58	1.130	7.365	0.148	0.353	0.843	3.568
FJMP + C + G	**0.70**	**94.54**	31.436	**0.57**	1.130	7.374	0.148	**0.352**	**0.835**	**3.485**
FJMP + C + G + A	1.32	**94.54**	29.820	0.62	**1.097**	**7.294**	**0.143**	0.370	0.895	3.883
Human-like	0.03	-	-	-	1.067	2.777	0.082	-	-	-

(a) Planning Metrics

Experiments	ADE ↓ (m)	FDE (m) ↓ (m)
FJMP + C	1.19	3.11
FJMP + C + G	**1.18**	**3.09**
FJMP + C + G + A	1.23	3.55

(b) Prediction Metrics

Table 5.1: Evaluation of different cost components. RS: Random Selection, C, G, and A represent Collision, Goal-following, and Ride Comfort Costs, respectively.

FJMP (G). We further assess the significance of multi-modalities in Section 5.5.2. Finally, in Section 5.5.3, we present ablation on different strategies to integrate high-level goals in our framework.

5.5.1 Analysis of cost components

The initial row in Table 5.1a highlights the necessity of incorporating the decision cost module into the network. Without it, randomly selecting consistently predicted scenes increases collisions and AV trajectories not aligning with the goal. The assessment, considering collision, goal-check, and planning terms, reveals that FJMP, including collision and goal components, performs better than others. Interestingly, adding the ride-comfort component leads to a notable increase in collision metrics. We observed a pattern in the network's predictions: scenes predicted by the network tend to represent specific behaviors. When we choose the mode with low acceleration cost, we favor trajectories where the AV and other agents make less progress compared to the ground truth, as seen by the progress metric. We also observe a similar pattern in prediction metrics, as noted in Table 5.1b. The collision and goal-check costs result in comparable prediction metrics. However, adding the acceleration component tends to favor trajectories that make low progress, leading to increased prediction metrics.

We introduced another metric called Collision (Pred) to analyze this effect further. This metric is similar to the collision metric mentioned in Section 5.3, but instead of comparing with the ground-truth trajectories of surrounding agents, it involves a comparison with the predicted trajectories of surrounding agents

Experiments	Collision (%) ↓ GT	Collision (%) ↓ Pred
FJMP + RS	1.50	1.15
FJMP + C	**0.70**	**0.20**
FJMP + C + G	**0.70**	**0.20**
FJMP + C + G + A	1.32	0.21

Table 5.2: Collision cost analysis.

belonging to the same mode as the AV to check collisions. Since the scenes are consistent, there isn't a significant change in metrics when comparing Collision (Pred) as seen in Table 5.2. However, a notable difference appears when comparing the AV's predicted trajectory to the ground truth trajectories of other agents, seen while comparing Collision (GT). In these cases, the AV stops according to the prediction while other agents continue moving in the ground truth data. This also shows the limitation of open-loop evaluation, as in real-world scenarios, the predictions would be generated in a receding manner.

Similarly, we evaluate the influence of different components on another baseline FJMP (G) shown in Table 5.3a. Here, considering all the cost components, it performs best in planning metrics because we have limited diversity in modes predicted and don't follow the earlier observed pattern of network prediction, as seen in qualitative evaluation Figure 5.1. Since the modes are more or less similar, we observe quite similar behavior of different cost components on prediction metrics in Table 5.3b

Experiments	Collision ↓ (%)	Goal-Check ↑ (%)	Progress ↑ (m)	FHE ↓ (rad)	Acc. ↓ (m/s^2)	Jerk ↓ (m/s^3)	Lat. Acc. ↓ (m/s^2)	Planning Metrics (m) ↓ @2s	@4s	@6s
FJMP(G) + C	0.91	86.42	29.488	0.82	1.231	10.624	**0.156**	0.358	0.787	**3.075**
FJMP(G) + C + G	0.86	**92.65**	29.624	0.77	1.243	10.766	0.157	0.357	0.787	3.085
FJMP(G) + C + G + A	**0.72**	**92.65**	**29.708**	**0.75**	**1.227**	**10.441**	0.158	**0.348**	**0.772**	3.086
Human-like	0.03	-	-	-	1.067	2.777	0.082	-	-	-

(a) Planning Metrics

Experiments	ADE ↓ (m)	FDE (m) ↓ (m)
FJMP(G) + C	**1.23**	**3.16**
FJMP(G) + C + G	**1.23**	3.18
FJMP(G) + C + G + A	1.28	3.33

(b) Prediction Metrics

Table 5.3: Evaluation of different cost components with FJMP (G) baseline. G: Goal.

Experiments	Collision ↓ (%)	Goal-Check ↑ (%)	Progress ↑ (m)	FHE ↓ (rad)	Acc. ↓ (m/s^2)	Jerk ↓ (m/s^3)	Lat. Acc. ↓ (m/s^2)	Planning Metrics (m) ↓ @2s	@4s	@6s
IL	0.89	88.51	30.970	0.69	1.184	8.885	**0.148**	**0.289**	**0.687**	**2.903**
FJMP + C + G	**0.70**	**94.54**	**31.436**	**0.57**	**1.130**	**7.374**	**0.148**	0.352	0.835	3.485

(a) Planning Metrics

Experiments	ADE ↓ (m)	FDE (m) ↓ (m)
IL	1.10	**2.81**
FJMP + C + G	**1.09**	3.09

(b) Prediction Metrics

Table 5.4: Comparison between unimodal and multimodal baseline. IL: Imitation Learning.

5.5.2 Effectiveness multi-modal predictions

To evaluate the effectiveness of multi-modal predictions, we compare our best-performing approach from Table 5.1 with the imitation learning (IL) baseline. As shown in Table 5.4a, our approach performs better than the IL baseline in all metrics, except for planning errors and in prediction metrics as well, IL is performing better by quite a margin in FDE seen in Table 5.4b. Since the IL baseline is trained solely for single-mode predictions through direct regression, it tends to overfit the ground-truth data more easily. This overfitting results from a frequently observed drawback with unimodal distributions, meaning they tend to converge to the mean of different behavior modes in the training dataset, as mentioned in Section 2.2. This, in turn, results in similar output for specific scenarios in the dataset.

5.5.3 Ablation on strategies to include goal

As shown in Table 5.5, both approaches demonstrate comparable results in collision metrics but show a marginal difference in ride-comfort and planning metrics. The performance improvement in FJMP (G) can be attributed to utilizing information about where the AV will end up in the ground truth during training, serving as a strong prior for our network.

Experiments	Collision ↓ (%)	Goal-Check ↑ (%)	Progress ↑ (m)	FHE ↓ (rad)	Acc. ↓ (m/s^2)	Jerk ↓ (m/s^3)	Lat. Acc. ↓ (m/s^2)	Planning Metrics (m) ↓ @2s	@4s	@6s
FJMP + C + G	**0.70**	**94.54**	**31.436**	**0.57**	**1.130**	**7.374**	**0.148**	0.352	0.835	3.485
FJMP(G) + C + G+ A	0.72	92.65	29.708	0.75	1.227	10.441	0.158	**0.348**	**0.772**	**3.086**
Human-like	0.03	-	-	-	1.067	2.777	0.082	-	-	-

Table 5.5: Comparison between different strategies for goal integration.

Experiments	Collision ↓ (%)	Goal-Check ↑ (%)	Progress ↑ (m)	FHE ↓ (rad)	Acc. ↓ (m/s^2)	Jerk ↓ (m/s^3)	Lat. Acc. ↓ (m/s^2)	Planning Metrics (m) ↓ @2s	@4s	@6s
FJMP + C + G	0.70	**94.54**	**31.436**	**0.57**	**1.130**	**7.374**	**0.148**	0.352	0.835	3.485
FJMP(G) + C + G + A	0.72	92.65	29.708	0.75	1.227	10.441	0.158	0.348	0.772	3.086
IL(G)	**0.50**	85.99	30.820	0.84	1.382	13.466	0.169	**0.313**	**0.688**	**2.676**

(a) Planning Metrics

Experiments	ADE ↓ (m)	FDE (m) ↓ (m)
FJMP + C + G	1.18	3.09
FJMP(G) + C + G	1.28	3.33
IL(G)	**1.09**	**2.76**

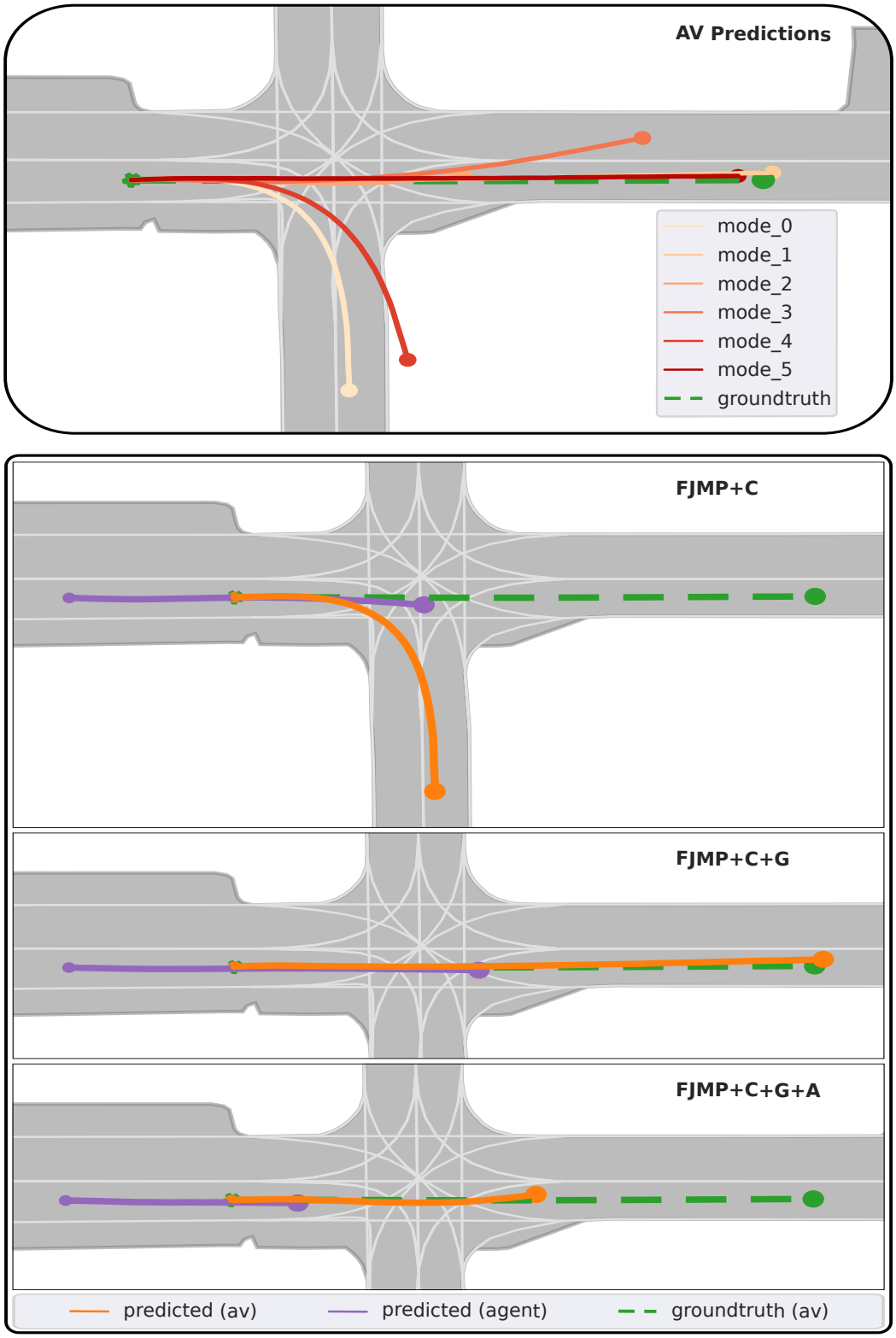
(b) Prediction Metrics

Table 5.6: Comparision between baselines with goal integration in training and our approach.

This effect becomes more apparent when incorporating similar information into the imitation-learning baseline, as evident in Table 5.6a and Table 5.6b. The improved performance in collision metrics can also be attributed to the strong prior about the reachable set from the ground truth, allowing the model to overfit and generate trajectories closely resembling the ground truth data.

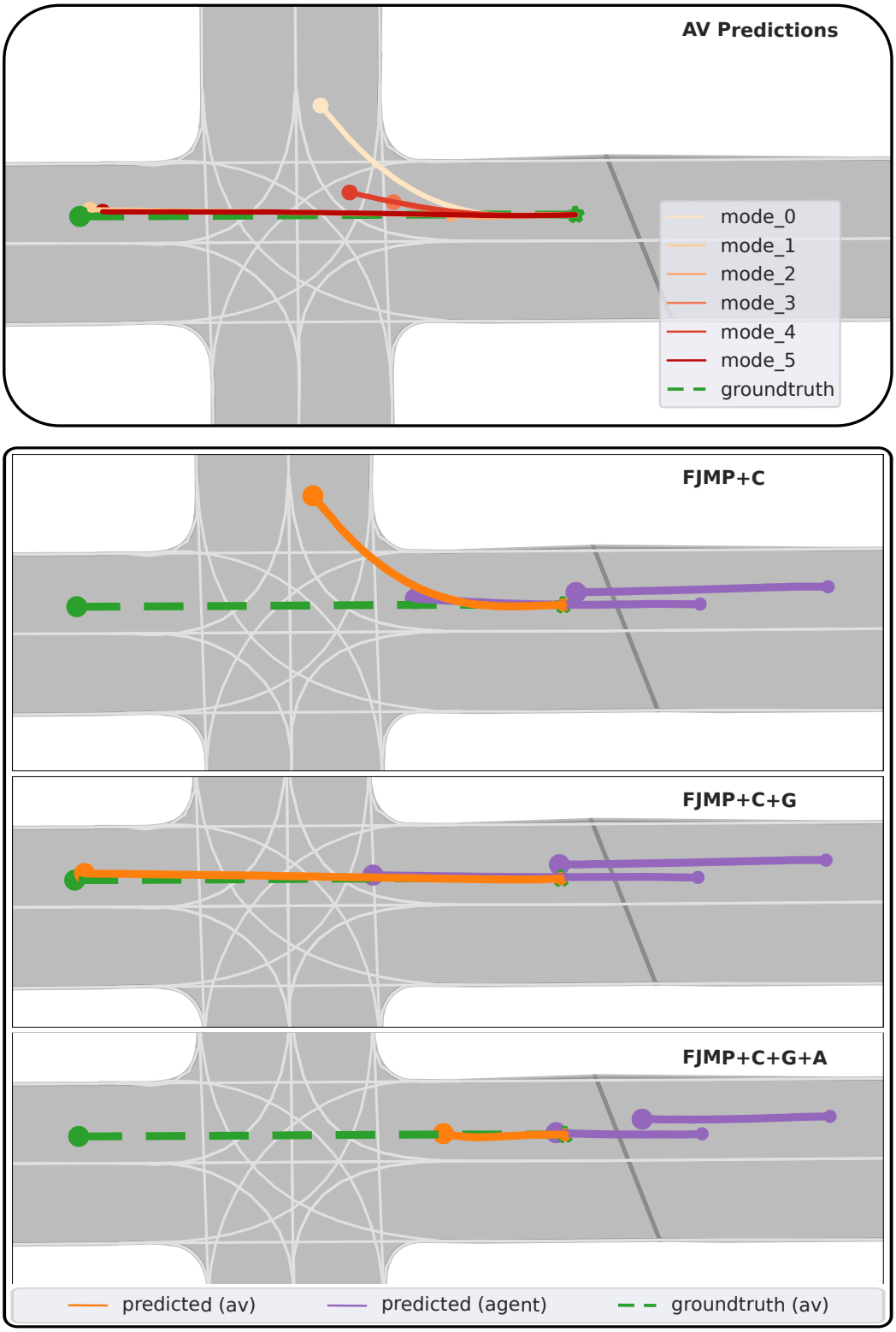
5.6 Qualitative Evaluation

In this section, we present qualitative results to validate further and compare the baselines and support the claims presented in Section 5.5. Figure 5.1 displays the modes selected by the addition of different components of our decision cost module in three different scenarios respectively. The importance of the goal-following cost module introduced in our approach is evident, especially for the AV as seen in Figure 5.1. While the collision cost module effectively deals with collisions, it might lead to a predicted forecast of the AV that doesn't align with the high-level goal. Since we know the high-level goal of the AV, we are more interested in finding the prediction of surrounding agents conditioned on the AV plan, which the addition of goal-following cost as a post-processing step can effectively achieve. Additionally, the importance of joint-prediction or scene-consistent prediction can be seen in last row of Figure 5.1a and Figure 5.1b, as the leading vehicle i.e., AV brakes the follower agents are also seen to be slowing down.



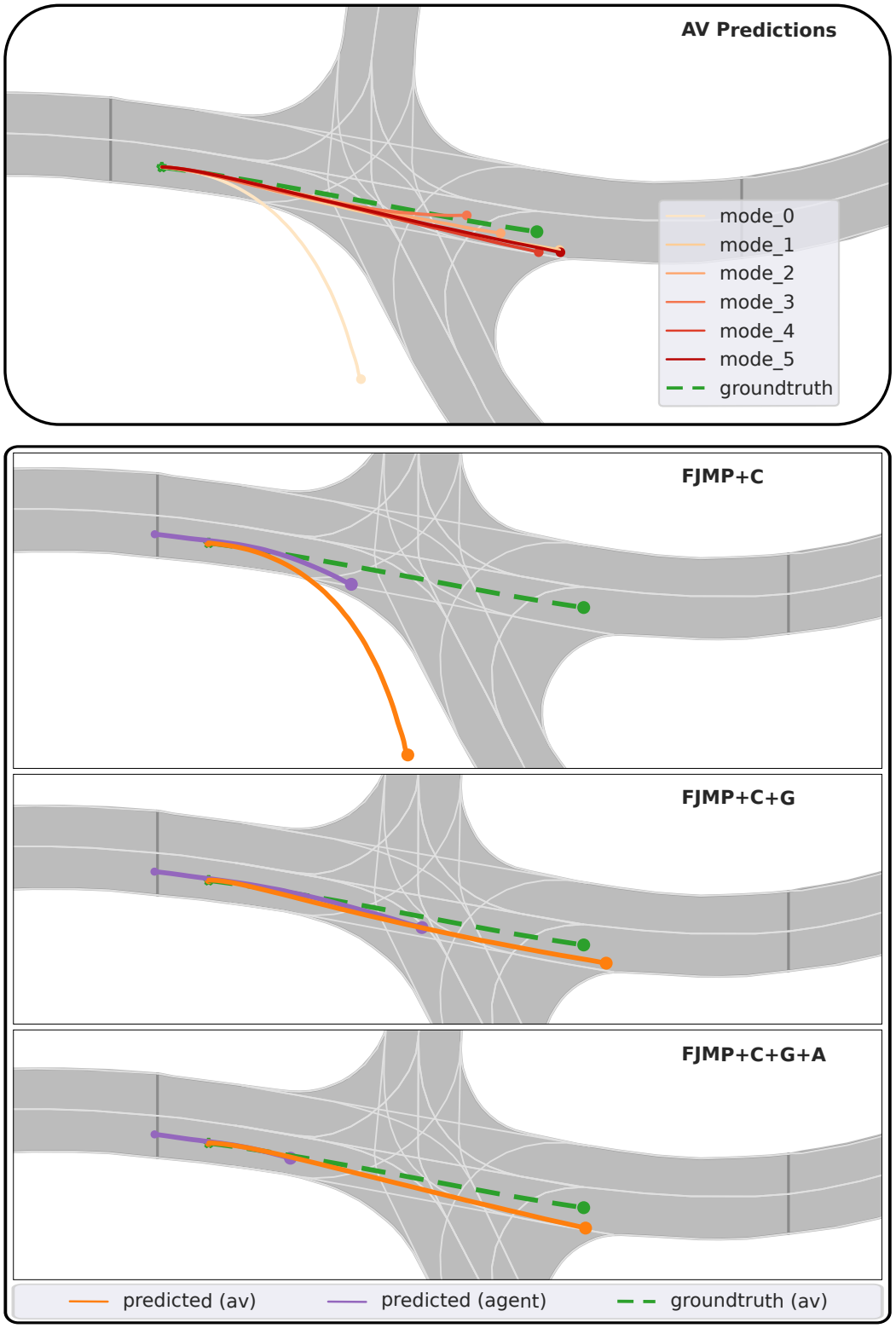
(a) Scenario 1.

Figure 5.1: The figure shows a comparison of different cost components on the selection of AV trajectory. The first row represents predicted modes of AV; for simplicity, modes corresponding to other agents are not shown.



(b) Scenario 2.

Figure 5.1: The figure shows a comparison of different cost components on the selection of AV trajectory. The first row represents predicted modes of AV; for simplicity, modes corresponding to other agents are not shown. (contd.)



(c) Scenario 3

Figure 5.1: The figure shows a comparison of different cost components on the selection of AV trajectory. The first row represents predicted modes of AV; for simplicity, modes corresponding to other agents are not shown. (contd.)

The patterns observed in the prediction module, reflecting specific behaviors, are depicted in Figure 5.2. Analyzing the distance traveled by the AV in different modes reveals a distinct pattern in the box plot distribution and median values. For instance, mode 1 demonstrates high acceleration, while mode 2 exhibits low acceleration. The remaining modes show similarities with slight variations.

Figure 5.3 shows a comparison between predicted modes of AV by the FJMP (Goal) and FJMP on three different scenarios. When comparing the modes, it is evident that the modes predicted from FJMP (Goal) align with the high-level goal information of the AV, and this also explains why FJMP (Goal) performs better in planning metrics as seen in Table 5.5. However, this also highlights the limitation that adding strong prior i.e., lane information from the HD-Map, we may loose diversity in predictions of AV. Similarly, we can observe this pattern when adding the goal information in the IL baseline as seen in Figure 5.4 and explains higher performance in both collision and planning metrics in Table 5.6.

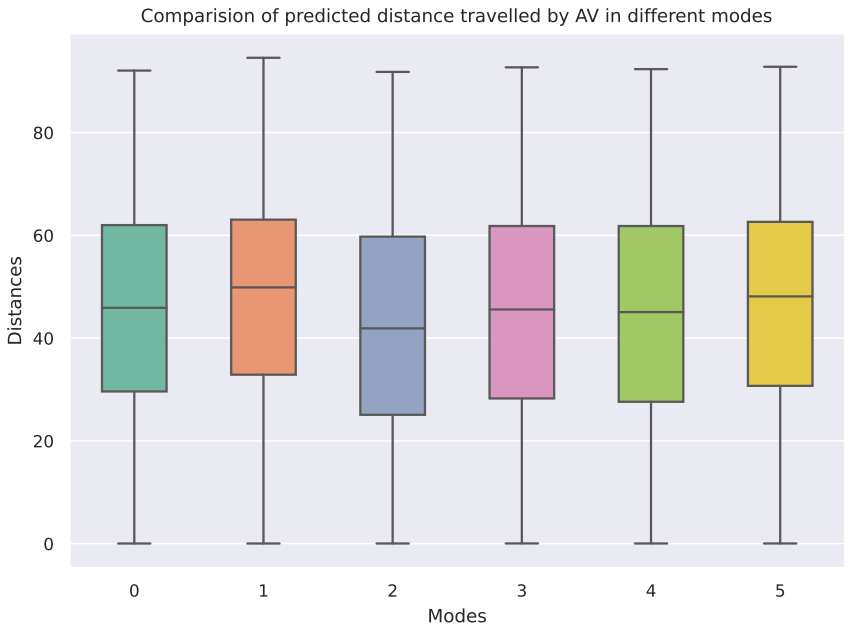


Figure 5.2: Box plot representing the distribution of progress associated with each mode. Mode 2 is the least progressing, whereas mode 1 is the most progressing.

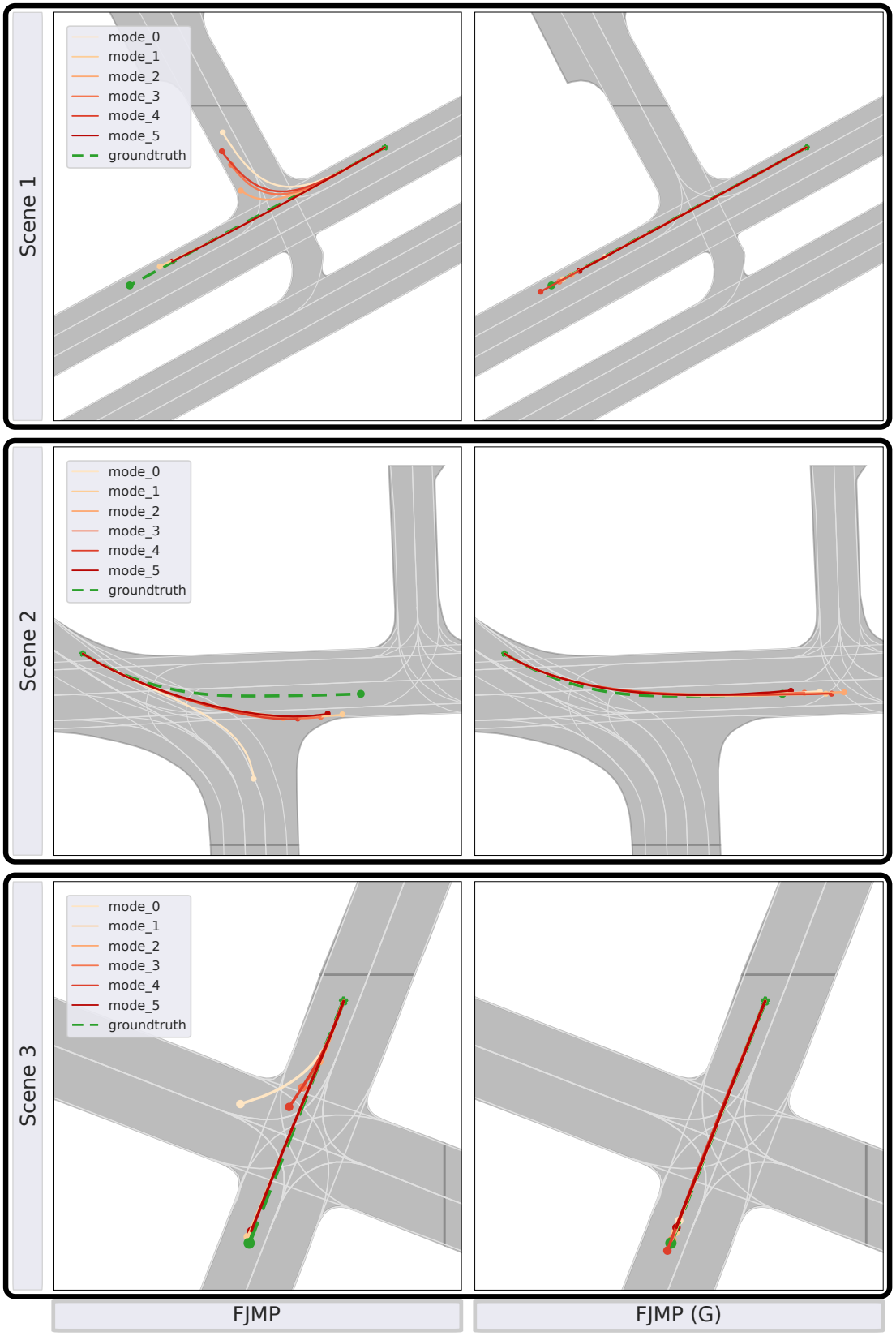


Figure 5.3: The figures show the effects of goal integration in the FJMP model.

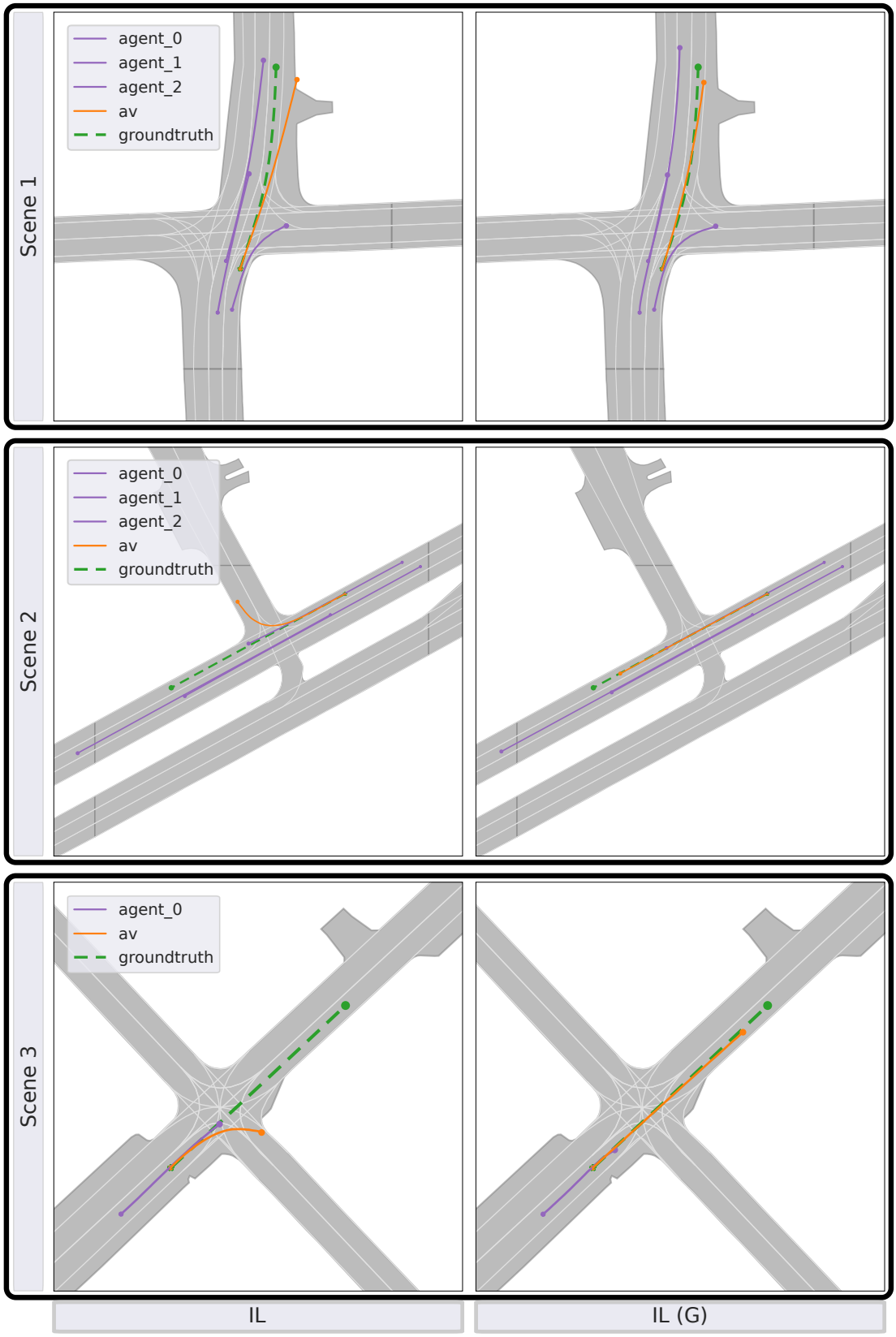


Figure 5.4: The figure shows goal integration in imitation learning (IL) baseline.

Chapter 6

Conclusion

In this thesis, we present an approach to extend existing scene-consistent motion predictor with an explicit decision-cost module that allows for a comprehensive assessment of the predicted scenes abiding by the planning task. We further present methods to integrate goal information available from the AV, leading to more informed predictions. We provide a thorough experimental evaluation backing the claims made within this thesis.

Through our experiments, we conclude that a comprehensive assessment of the decision-cost module yields more informed predictions and offers interpretability, explaining why a particular scene is the most likely. Moreover, adding goal information to the decision-cost module presents an efficient strategy for incorporating the high-level goal of the AV into prediction models without explicitly relying on a predefined route or endpoint. Through quantitative experiments on open-loop planning metrics, we hypothesize that the selected trajectory of AV can be utilized as a valid initialization for the planning module, as the selected trajectory adheres to the downstream planning task. Furthermore, our experiments with goal integration during training highlight the impact of incorporating such information, revealing that the predicted trajectories of the AV align with the specified high-level goal.

Our approach has a few limitations that we need to mention. While the decision-cost module offers interpretability, it might struggle to tell apart scenarios that look very similar. We may need to add further complexity to our decision-cost module to address this.

Another limitation is our heavy reliance on detailed HD-Map information and the need for precise data during training, especially when incorporating high-level goal information. In real-world situations, getting such exact data might be challenging, which could limit the effectiveness of our model.

Additionally, our experiments and recent studies indicate that the way we currently evaluate our model might not be the best for measuring planning per-

formance. Real-world scenarios involve dynamic interactions where other vehicles respond to what our vehicle is doing, indicating that closed-loop evaluation is better suited to measure planning performance. So, exploring ways to directly include goal information in how we predict scenes, trying evaluations in closed-loop settings, and making our decision-cost module more complex could be promising directions for future improvements. These steps could help overcome current limitations and make our framework more robust and applicable.

6.1 Short summary of key contributions

In conclusion, the key contributions of our presented approach to comprehensively evaluate/rank the evolutions of predicted scenes generated from scene-consistent motion predictor using an explicit decision cost module comprising of various predefined cost metrics are listed below:

- It can select a scene that adheres to the downstream planning tasks, i.e., choose a collision-free, goal-aligning, and traffic-following predicted trajectory from all possible outcomes, thereby providing more informed prediction
- The model with goal integration demonstrates the capability of aligning the predicted outcomes with the high-level goal information of the AV, producing goal-conditioned predictions without relying on a predefined route or endpoint
- We thoroughly assess our proposed approach using the Argoverse2 dataset. Our evaluation focuses on understanding different aspects of the decision cost and goal integration modules and comparing the effectiveness of unimodal and multimodal predictions

6.2 Open source contributions

We provide an open-source repository of our proposed pipeline efficiently written in Python and a data loader and model weights for models presented in this thesis. We have also modified the existing scene-consistent model architecture to test the ease of use of our modular decision cost module. We provide hyperlinks below to all these repositories for evaluation and use by the community.

- `https://github.com/Dhagash4/FJMP`
- `https://github.com/Dhagash4/Autobot`

Bibliography

[1] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social LSTM: Human Trajectory Prediction in Crowded Spaces. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, 2016.

[2] Samer Ammoun and Fawzi Nashashibi. Real time trajectory prediction for collision risk estimation between vehicles. In *Proc. of the IEEE Intl. Conf. on Intelligent Computer Communication and Processing (ICCP)*, 2009.

[3] Szilárd Aradi. Survey of deep reinforcement learning for motion planning of autonomous vehicles. *IEEE Trans. on Intelligent Transportation Systems (T-ITS)*, 23(2):740–759, 2020.

[4] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint*, arXiv:1903.11027, 2019.

[5] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *Proc. of the Conf. on Robot Learning (CoRL)*, 2020.

[6] Ming-Fang Chang, John W Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[7] Long Chen, Lukas Platinsky, Stefanie Speichert, Błażej Osiński, Oliver Scheel, Yawei Ye, Hugo Grimmett, Luca Del Pero, and Peter Ondruska. What data do we need for training an av motion planner? In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[8] H. Cui, V. Radosavljevic, F.C. Chou, T.H. Lin, T. Nguyen, T.K. Huang, J. Schneider, and N. Djuric. Multimodal Trajectory Predictions for Au-

tonomous Driving using Deep Convolutional Networks. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 2090–2096, 2019.

[9] Shengzhe Dai, Li Li, and Zhiheng Li. Modeling vehicle interactions via modified lstm models for trajectory prediction. *IEEE Access*, 7:38287–38296, 2019.

[10] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R Qi, Yin Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2021.

[11] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. VectorNet: Encoding HD Maps and Agent Dynamics from Vectorized Representation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[12] Roger Girgis, Florian Golemo, Felipe Codevilla, Martin Weiss, Jim Aldon D'Souza, Samira Ebrahimi Kahou, Felix Heide, and Christopher Pal. Latent Variable Sequential Set Transformers For Joint Multi-Agent Motion Prediction. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2022.

[13] Ross Girshick. Fast r-cnn. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2015.

[14] Mahir Gulzar, Yar Muhammad, and Naveed Muhammad. A Survey on Motion Prediction of Pedestrians and Vehicles for Autonomous Driving. *IEEE Access*, 9:137957–137969, 2021.

[15] Baotian He and Yibing Li. Multi-future Transformer: Learning diverse interaction modesfor behaviour prediction in autonomous driving. *IET Intelligent Transport Systems*, 16(9):1249–1267, 2022.

[16] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.

[17] Renbo Huang, Zhuo Guirong, Xiong Lu, Lu Xiong, Shouyi Lu, and Wei Tian. A Review of Deep Learning-Based Vehicle Motion Prediction for Autonomous Driving. *Sustainabilty*, 15(20):14716, 2023.

[18] Zhiyu Huang, Haochen Liu, Jingda Wu, and Chen Lv. Differentiable integrated motion prediction and planning with learnable cost function for autonomous driving. *IEEE transactions on neural networks and learning systems*, 2023.

[19] Zhiyu Huang, Chen Lv, Yang Xing, and Jingda Wu. Multi-modal sensor fusion-based deep neural network for end-to-end autonomous driving with scene understanding. *IEEE Sensors Journal*, 21(10):11781–11790, 2020.

[20] Boris Ivanovic and Marco Pavone. The Trajectron: Probabilistic Multi-Agent Trajectory Modeling With Dynamic Spatiotemporal Graphs. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2018.

[21] Donald B. Johnson. Finding all the elementary circuits of a directed graph. *SIAM J. Comput.*, 4(1):77–84, 1975.

[22] Nico Kaempchen, Kristian Weiss, Michael Schaefer, and Klaus CJ Deitmyer. IMM object tracking for high dynamic driving maneuvers. In *Proc. of the IEEE Vehicles Symposium (IV)*, 2004.

[23] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[24] D.P. Kingma and J.Ba. Adam: A method for stochastic optimization. *arXiv preprint*, abs/1412.6980, 2014.

[25] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[26] Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Soeren Kammel, J Zico Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, et al. Towards fully autonomous driving: Systems and algorithms. In *Proc. of the IEEE Vehicles Symposium (IV)*, 2011.

[27] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning Lane Graph Representations for Motion Forecasting. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020.

[28] T.Y. Lin, P. Goyal, R.B. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *arXiv preprint*, abs/1708.02002, 2017.

[29] Sajjad Mozaffari, Omar Y. Al-Jarrah, Mehrdad Dianati, Paul Jennings, and Alexandros Mouzakitis. Deep Learning-Based Vehicle Behavior Prediction for Autonomous Driving Applications: A Review. *IEEE Trans. on Intelligent Transportation Systems (T-ITS)*, 23(1):1558–0016, 2022.

[30] Jiquan Ngiam, Benjamin Caine, Vijay Vasudevan, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, David Weiss, Ben Sapp, Zhifeng Chen, and Jonathon Shlens. Scene Transformer: A unified architecture for predicting multiple agent trajectories. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2022.

[31] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2019.

[32] Luke Rowe, Martin Ethier, Eli-Henry Dykhne, and Krzysztof Czarnecki. FJMP: Factorized Joint Multi-Agent Motion Prediction over Learned Directed Acyclic Interaction Graphs. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[33] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020.

[34] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *arXiv preprint*, arXiv:1708.06374, 2017.

[35] Haoran Song, Wenchao Ding, Yuxuan Chen, Shaojie Shen, Michael Yu Wang, and Qifeng Chen. Pip: Planning-informed trajectory prediction for autonomous driving. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020.

[36] Veronika Thost and Jie Chen. Directed acyclic graph neural networks. *arXiv preprint*, arXiv:2101.07965, 2021.

[37] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian, and Pamela Mahoney. Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics (JFR)*, 23(9):661–692, 2006.

[38] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, M Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas Howard, Sascha Kolski, Alonzo Kelly, Maxim Likhachev, Matthew Mcnaughton, Nick Miller, and Dave Ferguson. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics (JFR)*, 25:425–466, 2008.

[39] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2021.

[40] Richárd Wéber. *Vulnerability and sensor placement analysis of water distribution networks.* PhD thesis, Budapest University of Technology and Economics, 04 2021.

[41] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020.

[42] ZF. Autonomous driving: An overview. Accessed on January 17th, 2023.

[43] Wei Zhan, Liting Sun, Di Wang, Haojie Shi, Aubrey Clausse, Maximilian Naumann, Julius Kummerle, Hendrik Konigshof, Christoph Stiller, Arnaud de La Fortelle, et al. Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps. *arXiv preprint*, arXiv:1910.03088, 2019.

[44] Alex Zyner, Stewart Worrall, and Eduardo Nebot. A recurrent neural network solution for predicting driver intention at unsignalized intersections. *IEEE Robotics and Automation Letters (RA-L)*, 3(3):1759–1764, 2018.

List of Figures

List of Tables

List of Algorithms

Appendix A

Paper

Evaluation of Traffic Scene Evolutions Using Scene-Consistent Prediction

Dhagash Desai Benedikt Mersch Julian Schmidt Cyrill Stachniss

***Abstract*— In this paper, we leverage recent advances in scene-consistent prediction to provide an initialization for the downstream planning task. The scene-consistent predictor outputs multiple possible evolutions of traffic scenes, where each predicted scene includes the future trajectory of all agents present in the scenario, including autonomous vehicles (AV). To this end, we employ an explicit decision cost module comprising various pre-defined cost metrics to evaluate/rank each predicted scene adhering to the downstream planning task. Furthermore, we introduce two approaches to integrate the goal information of the AV during the prediction process. One approach presents this information as part of the decision-cost module, and the other approach provides a way to directly integrate this information as input to the model directly, thereby aligning predictions to high-level goal information available to AV. We extensively evaluate our approach on the Argoverse2 dataset and show the effectiveness of our decision-cost module and goal-integration approaches.**

I. Introduction

Autonomous vehicles (AV), commonly called self-driving vehicles, enable navigating in complex urban scenarios without human intervention [21]. To navigate safely and reliably in urban scenarios, predicting the future driving behavior of participants in the surrounding traffic is essential for autonomous vehicles (AV). Most existing approaches in learning-based motion prediction predict independent trajectories for each agent without considering their interactions in the future [1], [4], [6], [13], [14], [18]. Such predictions may lead to inconsistent forecasts, highlighting the need for scene-consistent predictions for downstream tasks like planning. Recent approaches address this limitation by predicting multiple scenes or joint future trajectories of agents within a scene [7], [8], [15], [17]. In these methods, each scene consists of distinct future trajectory predictions for each agent, including predictions for the ego vehicle (AV), that are consistent with one another. This implies that the predictions for other agents are implicitly conditioned on one another.

In this paper, we focus on assessing the predicted evolution of each scene from scene-consistent predictor, leading to a deeper understanding of how the scene may unfold in the future and selecting the most likely scene based on a detailed assessment. Some common methods to select the scene from multiple predictions involve using a score-decoder that gives a likelihood score for each scene [7], [8]. Another commonly used approach is to rank the trajectories based on Euclidean distance and select the one with the least Euclidean distance [17]. However, the main drawback of the following evaluation/scoring criterion is that they lack interpretability/reasoning about why this scene is more likely than others and has a sub-optimal performance at intersections due to the use of the Euclidean distance metric as ranking criteria. Furthermore, while many state-of-the-art approaches primarily use an agent's past trajectory and road network information to predict future states, they often overlook the goal information for the AV. Specifically, the goal information, which indicates the intended destination or planned route for the AV as provided by the downstream planner, is often neglected in existing methods. Certain approaches address this limitation by incorporating goal information. For instance, some methods condition predictions of other agents based on specific end-states of the AV [15], [16]. Another common approach involves generating a set of trajectories for the AV and obtaining conditioned predictions for each trajectory [19]. However, these methods have drawbacks, as they often rely on a specific end-state for the AV and may not consider multiple ways to achieve the goal. Additionally, generating trajectories and conditioning predictions can significantly slow the planning process.

The main contribution of this paper is to develop a comprehensive evaluation approach to assess the evolution of predicted scenes to achieve more informed scene prediction. We select an existing scene-consistent approach and extend it by a modular decision cost module that evaluates the predicted scenes based on pre-defined cost metrics, including factors like traffic adherence and collision avoidance.

We also consider high-level goal information in the decision cost module, particularly for AV. This, in turn, allows us to evaluate the predictions of other agents conditioned on potential AV's trajectory. Subsequently, the AV's trajectory corresponding to the least scene cost can be forwarded as an initialization to ego-motion planning. We also provide an approach to integrate high-level goals in the model training to produce a goal-informed consistent prediction. An overview of our approach is presented in the Fig. 1.

In sum, we make three key claims: (i) We propose an explicit decision cost module comprising various pre-defined cost metrics to comprehensively evaluate/rank the evolutions of predicted scenes generated from a scene-consistent motion predictor. (ii) We propose methods to integrate goal information of AV. First, we include it in the decision cost and then modify the network to include the goal information of AV to get conditioned prediction on the high-level goal of AV. (iii)

Dhagash Desai, Benedikt Mersch, and Cyrill Stachniss are with the University of Bonn, Germany. Cyrill Stachniss is additionally with the Department of Engineering Science at the University of Oxford, UK, and with the Lamarr Institute for Machine Learning and Artificial Intelligence, Germany. Dhagash Desai and Julian Schmidt are additionally with Mercedes-Benz AG

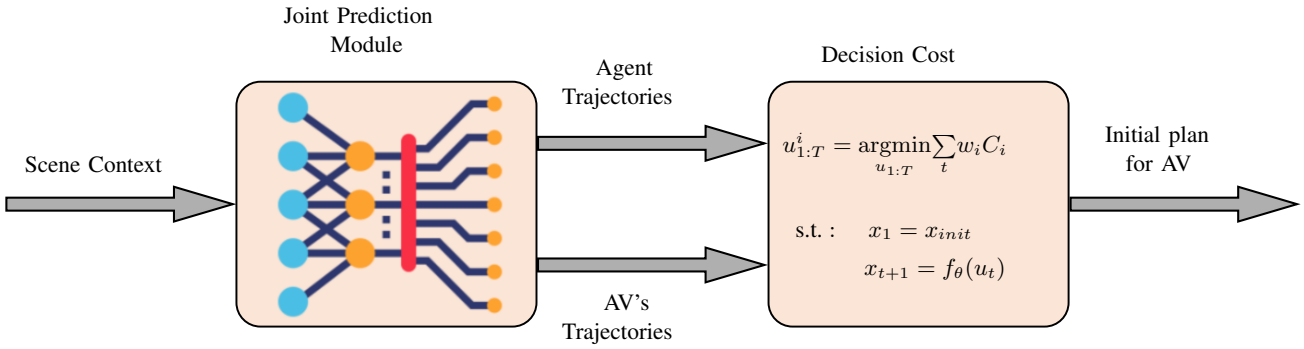


Fig. 1: An overview of our approach. We first generate consistent scenes using scene-consistent motion-predictor. These scenes are then systematically ranked based on predefined cost metrics. The trajectory of an autonomous vehicle (AV) associated with the scene with the least cost is subsequently chosen as the initial plan for downstream modules.

We thoroughly assess our proposed method using the open-source Argoverse2 dataset, demonstrating the effectiveness of various cost components. Additionally, we compare different approaches to goal integration. These claims are backed up by the paper and our experimental evaluation.

II. Related Work

Learning-based motion prediction models are gaining prominent in recent years, as deep convolution network has shown potential in handling complex agent interactions and road scenarios, a part of the credit goes to availability of large scale open-source datasets [2], [3], [5], [20], thus providing better accuracy for long-term prediction.

Nevertheless, most current models are designed to independently forecast each agent's future trajectory, which is computationally inefficient and may lead to inconsistent outcomes [10]. Recent approaches [7], [8], [15], [17] address this limitation by focusing on multi-agent joint prediction for consistent future trajectory generation. This method, known as joint prediction or scene-consistent prediction, assumes that well-learned interactions during the encoding stage implicitly contain information about future states [9], thereby aiding the planning task [10]. Therefore, in our approach we employ multi-agent joint prediction setting, specifically we extend upon an approach by Rowe et al. [17] due to its superior performance on competitive benchmarks [22].

In multi-modal motion prediction models, generating and selecting the most likely trajectory is crucial for evaluating performance and planning tasks. Most of the existing methods employ explicit score decoder [7], [8], [15], or rank trajectories based on Euclidean distance to the ground-truth data [17]. However, the current evaluation/scoring criteria have drawbacks, such as a lack of interpretability, suboptimal performance of the L2 distance metric at intersections as noted by Cui et al. [4], unavailability of ground-truth data in real-world scenarios, and, notably, a poor alignment with the downstream planning task. Conversely, we employ an explicit decision cost module to assess various modes. This module incorporates crucial planning cost functions that align effectively with subsequent tasks and enhance interpretability.

State-of-the-art learning-based motion prediction pipeline considers the agent's past states and road-network information is available to predict the future states of the agents present in the scenario. However, for AV, there's a vital piece of information often overlooked by many approaches—the goal information, indicating where the AV intends to be in the future. The paper by Rheinhart et al. [16] addresses this limitation by conditioning prediction on the agent's state. Ngiam et al. [15] masks the AV's future goal while encoding agent-context. Song et al. [19] generates explicit AV trajectories considering the goal, which are then used in the prediction model. However, these approaches focus on a specific end-state without considering various ways to achieve a goal. Our approach addresses this by concentrating on goal lanes instead of a fixed end-state or initial plan, acknowledging the practical uncertainties associated with AV's end-state.

In summary, our approach enhances the existing scene-consistent model by adding a decision cost module. This module evaluates predicted traffic scenes based on collision potential, traffic rule adherence, and goal information integration. Ranking scenes by these costs provides interpretability. The AV's trajectory with the lowest cost is forwarded to a planner for ego-motion planning. We also propose various methods to integrate high-level goal information without relying on a specific state.

III. Our Approach

Our approach evaluates the consistent scenes resulting from scene-consistent motion predictor and selects the optimal scene according to predefined decision cost, thus aiding the planning task. We first generate consistent scenes from a scene-consistent motion predictor; the selected model as explained in Sec. III-A. Then we evaluate the scenes using a pre-defined decision cost module; the key components of the module are explained in Sec. III-B. Finally, Sec. III-C explains briefly the overall scene cost. See Fig. 1 for our overall pipeline.

A. Joint Prediction Module

We use the factorized joint motion prediction (FJMP) approach proposed by Rowe et al. [17] as our joint prediction module, leveraging its state-of-the-art performance and open-source codebase. This method decomposes joint prediction into a series of marginal and conditional predictions, modeling future interactions among agents using a sparse directed interaction graph. In this graph, edges represent relationships between pairs of agents; their influencer-reactor relationship determines each edge's direction. The directed interaction graph is then transformed into a directed acyclic graph (DAG) through an efficient algorithm by Johnson [11]. Consequently, joint future predictions are generated through a sequence of conditional and marginal predictions based on the DAG's order. Marginal predictions are created for the source nodes in the DAG, while conditional predictions are generated for non-source nodes conditioned on their parents' predicted futures in the DAG. To facilitate this sequential future prediction through the DAG, the approach employs a directed acyclic graph neural network (DAGNN) architecture. We adopt this approach without any modification.

B. Decision Cost

We introduce an explicit decision cost module inspired by classical planning approaches to evaluate and rank these predicted traffic scenes. The decision cost module consists of various cost terms that encode different aspects of driving primarily for AV. The details about different cost terms are mentioned below.

Ride Comfort Costs. To quantify this comfort aspect, we introduce a cost metric associated with the predicted trajectory's longitudinal acceleration $\hat{v}$. This metric allows us to assess the predicted trajectories in terms of their impact on ride comfort, aligning with the passenger's preference for a comfortable driving experience. The longitudinal acceleration limits are $[-5, 5]\, m/s^2$.

$$c_{acc} = \frac{1}{\delta} \sum_{i=t}^{i=t+\delta} acc_{cost}(\hat{v_i}) \tag{1}$$

in which,

$$acc_{cost} = \begin{cases} 0, & \text{if } -5 \leq \hat{v} \leq 5 \\ (|\hat{v}| - 5)^2, & \text{otherwise} \end{cases} \tag{2}$$

Collision Costs. We explicitly model a collision cost to guarantee that the selected trajectory avoids collisions with other agents in the scenario. We adopt the approach from the INTERACTION dataset [22] to model collision cost with minor modifications. A list of circles defines each agent according to radius r_i and r_j, and the collision threshold ϵ_C is determined as per Eq. (3). Unlike the INTERACTION approach, we define a non-binary cost function. The cost is computed by calculating the Euclidean distance between agent i and j at each time step. The minimum distance, d_{min}, is then selected from all time steps. Given, d_{min} and ϵ_C, the cost is calculated as per Eq. (4). In a scenario involving multiple agents, the maximum of this cost is selected after calculating the collision cost concerning a specific agent i with all the other agents in the scene. This choice is made to encode the highest possible cost that a predicted trajectory may incur and mitigate its impact on downstream tasks.

$$\epsilon_C = \frac{r_i + r_j}{\sqrt{3.8}} \tag{3}$$

where r_i and r_j are defined according to the agent dimensions as given in the dataset.

$$c_{collision} = \begin{cases} \left(1 - \frac{d_{min}}{\epsilon_C}\right)^3, & \text{if } d_{min} \leq \epsilon_C \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Goal-following Costs. We model this cost to integrate high-level goal/planned route information that AV aims to reach. This integration allows us to understand better and evaluate the AV's predicted trajectory in relation to its goal, contributing to more informed and goal-aware decision-making processes. One method to integrate this information is to select a predicted mode based on Euclidean distance from the goal point (assuming it is known, which may not always be true). However, this method might lead to erroneous selection, as illustrated in Fig. 2. In contrast, our approach integrates goal information by assessing the overlap between the forward reachability from the predicted end-point and the goal point. Our approach acknowledges the possibility of multiple ways to achieve a given high-level goal. The underlying assumption is that if the predicted position and the goal location share the same reachable lanes, the trajectory is well-aligned with the goal direction. Given the size and information about the road network graph, we believe this measure effectively encodes goal-related information.

We define the goal-following cost as a binary cost, either 1 or 0, based on the alignment of the predicted trajectory with the specified high-level goal. The goal-following cost aims to identify all sets of lanes (denoted as L) that are reachable from the high-level goal. The cost is defined such that if the sets of forward-reachable lane segments from the predicted trajectory and the high-level goal overlap, the cost is 0; otherwise, it is 1. To calculate the goal-following cost, we assume the availability of a road-network graph $\mathcal{G}_{map} = \{\mathcal{V}, \mathcal{E}\}$, where nodes $\mathcal{V}$ represent lane-segments in the scenario, and edges $\mathcal{E}$ contain information about neighbors and the path of the lane-segments. Additionally, we have predicted and ground-truth end-points denoted as $\hat{y}_{i,t}$, y_{goal} along with their respective orientations $\hat{\psi}_{i,t}$, ψ_{goal}. For our approach, we assume the goal to be $y_{goal} = y_{i,t}$ (the end-point of the ground-truth trajectory for AV in the respective scenario) and associated orientation $\psi_{goal} = i, t$. We assume this information is available, as a navigation planner typically provides a set of lanes or at least knows the high-level goal that AV is trying to reach.

The goal-following cost calculation involves two main steps: first, finding the closest assigned lane segments $\forall p \in [\hat{y}_{i,t+\delta}, y_{goal}]$ in the road-network graph, and second, identifying all the successor lanes corresponding to the assigned

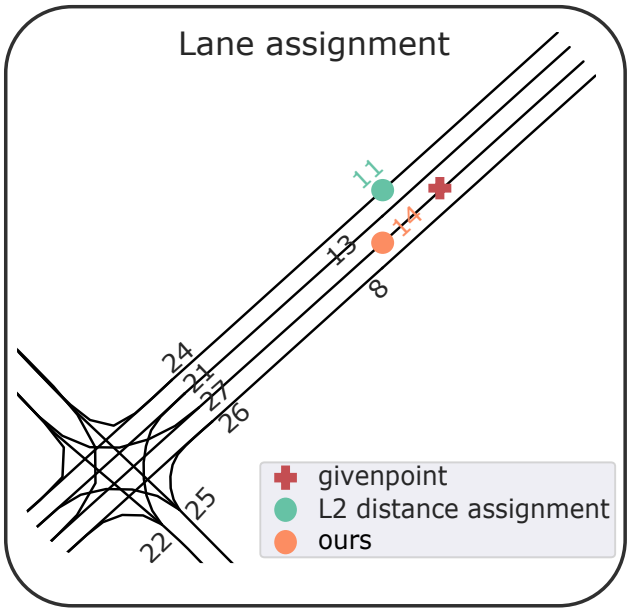


Fig. 2: The figure shows for given point p lane assignment using different strategies.

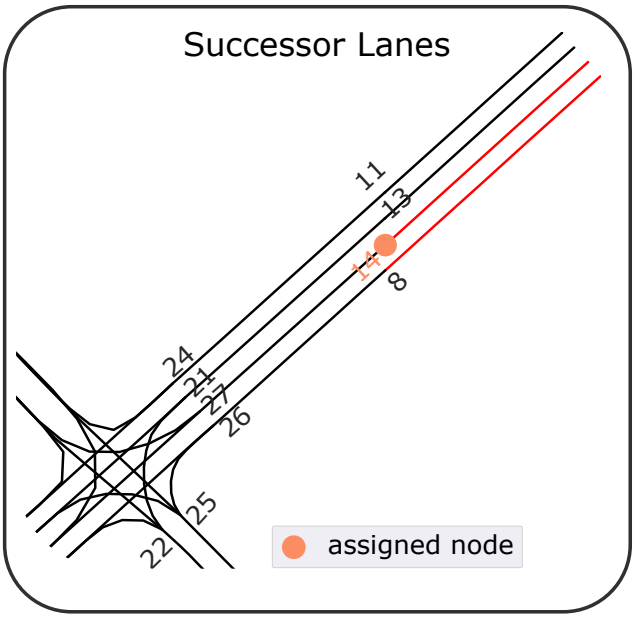


Fig. 3: In this diagram, we use red color to highlight the successor lanes of the assigned node, which is marked with an orange circle, using our approach.

lane segments. Then, based on the overlap between two sets of lanes, we assign a cost to the given predicted trajectory. The details of each step are explained using the point $p \in [\hat{y}_{i,t+\delta}, y_{goal}]$ and its associated orientation $\psi \in \hat{\psi}_{i,t+\delta}, \psi_{i,t}$ as an example. The overall process of goal-following cost is outlined in Alg. 1.

1) **Lane assignment:** To determine the reachable set of lane segments, we first identify the closest lane $n_i \in \mathcal{V}$ in the given map for the point p. This involves obtaining all lane segments within a $\pm 45°$ range from the associated orientation ψ to that point. Subsequently, we select the lane closest to the given point using L2 distance. This step of selecting lanes with similar orientation is crucial, as relying solely on L2 distance for the nearest lane segments could result in erroneous assignments, as seen in Fig. 2.
2) **Finding successor lanes:** After identifying the closest lane segment n_i, we proceed to find all the successor lane segments from v_i using the breath first search (BFS) algorithm. We also keep track of the neighbors of the lane segment n_i. Consequently, the reachable lane set L comprises successor lanes associated with the closest lane segment and their neighbors. We decide to accommodate lane changes while following the goal and consider the possibility of multiple lanes heading in the same direction toward the high-level goal. Fig. 3 highlights successor lanes given the assigned node.

C. Scene Cost

The overall scene cost consists of two components: the cost related to the ego-vehicle C_{ego}, i.e., the AV, and the cost associated with surrounding agents C_{agents}. Including agent cost allows us to capture the impact of the ego-vehicle on the agents in the scene, aligning with game-theoretic approaches where the optimal scenario involves every agent behaving optimally. Each cost component is the weighted sum of various components. It's important to note that the agent cost C_{agents} does not include the goal-following cost for obvious reasons – we may not be aware of the plans or high-level goals of the surrounding agents. Hence, the overall scene cost of each predicted scene is as per Eq. (5).

$$C_{scene} = C_{ego} + C_{agent} \tag{5}$$

where,

$$C_{ego} = w_a \cdot c_{acc} + w_c \cdot c_{collision} + w_g \cdot c_{goal}$$

$$C_{agent} = w_a \cdot c_{acc} + w_c \cdot c_{collision}$$

Here, w_a, w_c, and w_g are weights for ride comfort, collision, and goal cost, respectively. Thus, we have scene cost corresponding to each predicted scene, i.e., to each mode $k \in K$. The scene/mode with the least cost is then selected as the optimal scene, and the trajectory corresponding to AV is then passed on to downstream tasks.

IV. Experimental Evaluation

The main focus of this work is an approach to evaluating/ranking modes provided by scene-consistent motion predictors and selecting the best mode according to this evaluation. We present our experiments to show the capabilities of our method. The results of our experiments support key claims, which are: (i) We propose an explicit decision cost module comprising various pre-defined cost metrics to comprehensively evaluate/rank the evolutions of predicted

Algorithm 1 Goal-following cost as proposed in this thesis

Require: Road-network graph $\mathcal{G}$, Points $p \in [\hat{y}_{i,t}, y_{i,t}]$, Orientation $\theta = [\hat{\psi}_{i,t}, \psi_{i,t}]$

$lane\_assigned \leftarrow \emptyset$
for $\forall p \in [\hat{y}_{i,t}, y_{i,t}]$ and $\forall \psi \in [\hat{\psi}_{i,t}, \psi_{i,t}]$ **do**
    $v \leftarrow \text{AssignLane}(p, \psi)$
    $lane\_assigned \leftarrow lane\_assigned \cup v$
$\hat{\mathcal{L}} \leftarrow \text{FindSuccessorLanes}(lane\_assigned[0])$
$\mathcal{L} \leftarrow \text{FindSuccessorLanes}(lane\_assigned[1])$
if $\hat{\mathcal{L}} \cap \mathcal{L} \neq \emptyset$ **then**
    $c_{goal} \leftarrow 0$
else
    $c_{goal} \leftarrow 1$
return c_{goal}

scenes generated from a scene-consistent motion predictor. (ii) We propose methods to integrate goal information of AV. First, we include it in the decision cost and then modify the network to include the goal information of AV to get conditioned prediction on the high-level goal of AV.

A. Metrics

We conduct our experiments in an open-loop manner. In open-loop setting, we aim to compare the output from our approach i.e., consistently predicted scenes against the ground-truth data available from the dataset. The metrics set below for evaluation can be broadly classified into prediction metrics to assess the performance of the prediction model and planning metrics to assess the performance of the selected AV's trajectory corresponding to the best-predicted scene. We assess the following to evaluate suitability for initializing downstream modules such as planning.

Prediction Metrics: We access the performance of prediction models using commonly used average displacement error (ADE) and final displacement error (FDE) [20] of predicted multi-agent joint trajectories corresponding to the selected scene by our approach.

Planning Metrics: We select the planning metrics as per the state-of-the-art approach by Huang et al. [10], and it includes

- **Planning metric** is a metric to quantify human likeliness. This is calculated by comparing the L2 distance between the selected trajectory and the ground-truth trajectory at particular time steps (i.e., 2 s, 4 s, 6 s).
- **Comfort metrics** quantifies the rider's comfort and the feasibility of the selected trajectory. Three key quantities are introduced to capture and quantify these comfort metrics: longitudinal acceleration, jerk, and lateral acceleration. These values are computed as averages over time within a given overtime and ardently compared against the expert driving trajectory.
- **Collision rate** is a metric that gauges the safety performance of our approach, which is particularly relevant for downstream tasks. The collision is calculated based on AV selected trajectory and the ground-truth future trajectories of other agents present in the scenario.
- **Goal-check** metric assesses whether the selected trajectory aligns with the high-level goal of the AV, which is crucial for planning purposes. It is a binary metric with values 0 or 1, indicating whether the selected trajectory is not heading in the goal direction (0) or is aligned with the goal (1). The calculation of this metric is based on the goal-following costs, explained in Sec. III-B.
- **Final heading error (FHE)** is computed by calculating the absolute error between the heading of the ground-truth endpoint and the selected trajectory endpoint. This metric shows how closely the selected trajectory matches the expert's driving direction.
- **Progress** metric measures the advancement of the selected trajectory for the AV. It is determined by computing the Euclidean distance between the starting and ending points of the chosen trajectory. A higher value of this metric signifies greater progress in the given time. This metric is computed for each scene, and the final result is presented as the average of all scenes.

B. Dataset and Implementation Details

For the following experiments, we train and validate our framework on Argoverse2 [20], a large-scale open-source urban driving dataset. The dataset contains 250,000 non-overlapping scenarios. Each scenario has its associated HD-Map, 11 seconds of trajectory data (2D position, velocity, and orientation) recorded at 10 Hz of all the tracks observed by ego-vehicle, and an additional classification label for each track belonging to one of ten unique object category types. For each scenario, the 11 seconds time window is split as a 5 seconds observation horizon and 6 seconds as the prediction horizon. The total number of scenarios used for training is 199,908, while 24,988 are used for validation. Since we do not have access to the testing split from official Argoverse2. We further split the validation set, keeping 16,765 frames for testing and 8203 for validation.

We train the motion-prediction module as per details presented by Rowe et al. [17] to reproduce competitive results on the prediction benchmark. Specifically, we set $K = 6$ for the factorized decoder and $K = 15$ for the proposal decoder. We train our model on NVIDIA GeForce RTX 2080 Super using Adam optimizer [12]. We set the batch size to 32 and train for 36 epochs with a learning rate 1e-3, step-decayed by a factor of $1/10$ at epoch 32. The model training on a single GPU takes approximately 10 - 12 hours of time.

C. Baselines

In this section, we mention details about the chosen baselines to compare our proposed framework. We compare our framework with the following baselines to address the limitations and advantages of our approach.

FJMP (Goal): For this baseline, we modify the existing FJMP architecture to encode high-level goal information directly into the model architecture. Specifically, we make the following modifications to the network to integrate the goal information. First, during preprocessing, we identify

Experiments	Collision ↓	Goal-Check ↑	Progress ↑	FHE ↓	Acc. ↓	Jerk ↓	Lat. Acc. ↓	Planning Metrics (m) ↓			ADE ↓	FDE ↓
	(%)	(%)	(m)	(rad)	(m/s^2)	(m/s^3)	(m/s^2)	@2s	@4s	@6s	(m)	(m)
FJMP + RS	1.50	88.82	30.932	0.62	1.130	7.473	0.147	0.361	0.868	3.705	-	-
FJMP + C	**0.70**	89.12	**31.447**	0.58	1.130	7.365	0.148	0.353	0.843	3.568	1.19	3.11
FJMP + C + G	**0.70**	**94.54**	31.436	**0.57**	1.130	7.374	0.148	**0.352**	**0.835**	**3.485**	**1.18**	**3.09**
FJMP + C + G + A	1.32	**94.54**	29.820	0.62	**1.097**	**7.294**	**0.143**	0.370	0.895	3.883	1.23	3.55
Human-like	0.03	-	-	-	1.067	2.777	0.082	-	-	-	-	-

TABLE I: Evaluation of different cost components. RS: Random Selection, C, G, and A represent Collision, Goal-following, and Ride Comfort Costs, respectively

Experiments	Collision ↓	Goal-Check ↑	Progress ↑	FHE ↓	Acc. ↓	Jerk ↓	Lat. Acc. ↓	Planning Metrics (m) ↓			ADE ↓	FDE ↓
	(%)	(%)	(m)	(rad)	(m/s^2)	(m/s^3)	(m/s^2)	@2s	@4s	@6s	(m)	(m)
FJMP + C + G	**0.70**	**94.54**	**31.436**	**0.57**	**1.130**	**7.374**	**0.148**	0.352	0.835	3.485	**1.18**	**3.09**
FJMP(G) + C + G + A	0.72	92.65	29.708	0.75	1.227	10.441	0.158	**0.348**	**0.772**	**3.086**	1.28	3.33
Human-like	0.03	-	-	-	1.067	2.777	0.082	-	-	-	-	-

TABLE II: Comparison between different strategies for goal integration. FJMP(G): FJMP (Goal)

all the successor lanes given the high-level goal for each scenario, taking it as the ground-truth future endpoint of the AV's trajectory. The identification process is similar to the process we follow for goal-following cost described in Sec. III-B. Additionally, we find the route connecting the high-level goal and the last observed endpoint in the ground truth of the AV's trajectory, using BFS on the road-network graph $\mathcal{G}$. We aim to encode this information in training, as typically, this information is available through upstream tasks like planning. Thus, the route lanes include the route from the observed ground-truth endpoint to the high-level goal, and all the lanes are then reachable from the high-level goal. We encode the information about route lanes in the given HD-Map through a binary mask.

Additionally, we introduce an additional class representing the AV for each scenario and pass this information as input. We hypothesize that adding this information as input can drive the predictions to align with the high-level goal. The model is trained using the exact specifications and hyper-parameters mentioned in Sec. IV-B, ensuring that results in comparison are fair.

FJMP + RandomSelection: In this baseline, the explicit decision cost module is replaced by random selection. In other words, the mode is selected based on a discrete uniform distribution, introducing randomness into the mode selection process.

D. Quantitative Evaluation

We list the quantitative evaluations of our method and other baselines mentioned in Sec. IV-C based on open-loop metrics defined in Sec. IV-A. These evaluations are performed on the testing split that consists of 16,785 frames. We list the quantitative evaluation results on different cost components in Tab. I. The initial row in Tab. I highlights the necessity of incorporating the decision cost module into the network. Without it, randomly selecting consistently predicted scenes increases collisions and AV trajectories not aligning with the goal. The assessment, considering collision, goal-check, and planning terms, reveals that FJMP, including collision and goal components, performs better than others. Interestingly, adding the ride-comfort component leads to a notable increase in collision metrics. We observed a pattern in the network's predictions: scenes predicted by the network tend to represent specific behaviors. When we choose the mode with low acceleration cost, we favor trajectories where the AV and other agents make less progress compared to the ground truth, as seen by the progress metric. We also observe a similar pattern in prediction metrics, as noted in Tab. I. The collision and goal-check costs result in comparable prediction metrics. However, adding the acceleration component tends to favor low progress trajectories, leading to increased prediction metrics. Tab. II shows comparison to goal integration methods with decision cost module, it demonstrates comparable results in collision metrics but show a marginal difference in ride-comfort and planning metrics. Here all the components of decision-cost for FJMP (Goal) are chosen as it provides for superior performance when analyzed. The performance improvement in FJMP (Goal) can be attributed to utilizing information about where the AV will end up in the ground truth during training, serving as a strong prior for our network.

E. Qualitative Evaluation

In this section, we present qualitative results to validate further and compare the baselines and support the claims presented in Sec. IV-D. Fig. 4 displays the modes selected by the addition of different components of our decision cost module in one particular scenario. The importance of the goal-following cost module introduced in our approach is evident, especially for the AV as seen in Fig. 4. While the collision cost module effectively deals with collisions, it might lead to a predicted forecast of the AV that doesn't align with the high-level goal. Since we know the high-level goal of the AV, we are more interested in finding the prediction of surrounding agents conditioned on the AV plan, which the addition of goal-following cost as a post-processing step

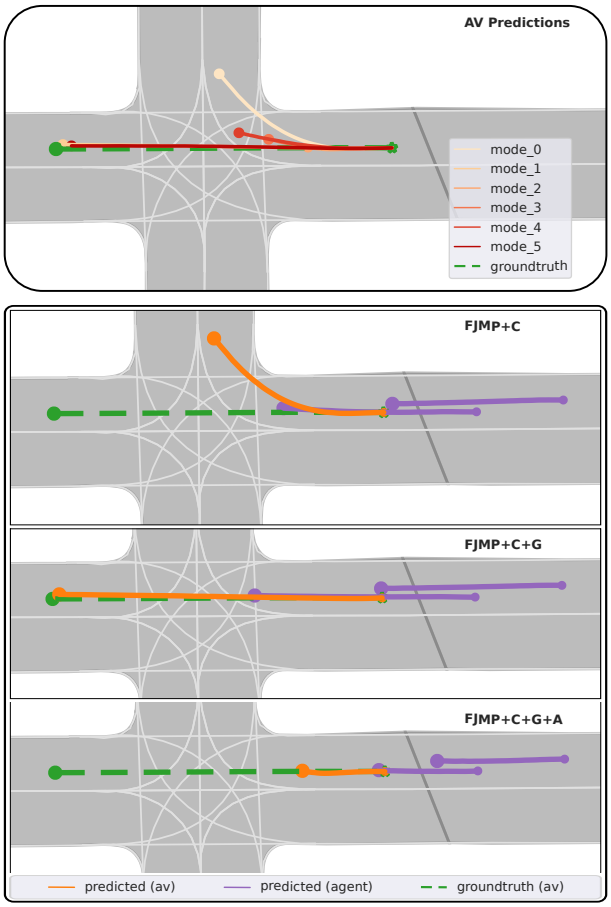


Fig. 4: The figure shows a comparison of different cost components on the selection of AV trajectory. The first row represents predicted modes of AV; for simplicity, modes corresponding to other agents are not shown.

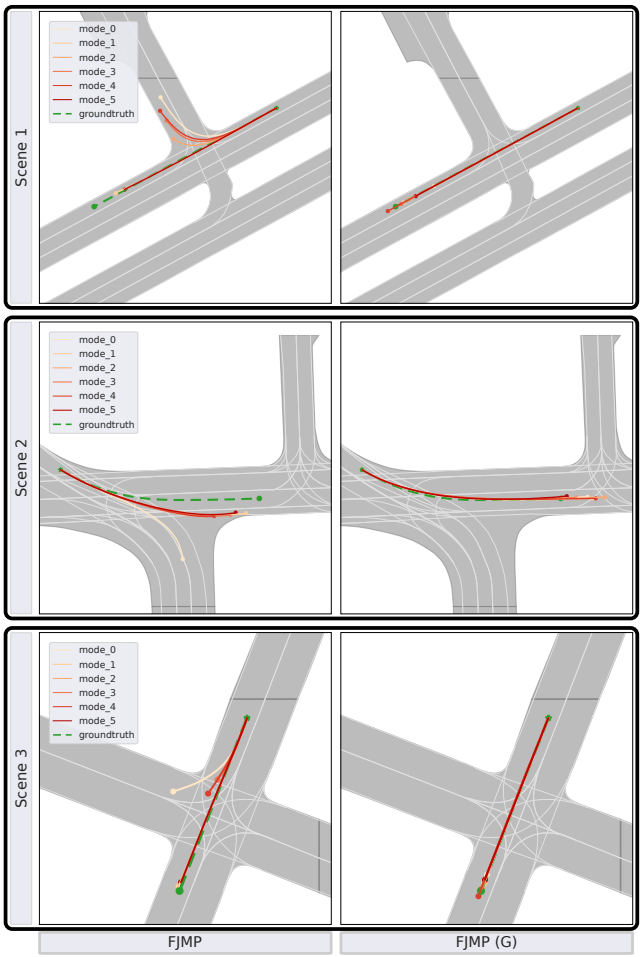


Fig. 5: The figures show the effects of goal integration in the FJMP model.

can effectively achieve. Additionally, the importance of joint-prediction or scene-consistent prediction can be seen in last row of Fig. 4, as the leading vehicle i.e., AV brakes the follower agents are also seen to be slowing down.

Fig. 5 shows a comparison between predicted modes of AV by the FJMP (Goal) and FJMP on three different scenarios. When comparing the modes, it is evident that the modes predicted from FJMP (Goal) align with the high-level goal information of the AV, and this also explains why FJMP (Goal) performs better in planning metrics as seen in Tab. II. However, this also highlights the limitation that adding strong prior i.e., lane information from the HD-Map, we may loose diversity in predictions of AV.

V. Conclusion

In this paper, we presented an approach to extend existing scene-consistent motion predictor with an explicit decision-cost module that allows for a comprehensive assessment of the predicted scenes abiding by the planning task. We further present methods to integrate goal information available of AV, leading to more informed predictions. We implemented and evaluated our approach on the Argoverse2 dataset, provided comparisons to other existing techniques, and supported all claims made in this paper. The experiments suggest that comprehensive assessment through the decision-cost module yields more informed predictions and offers interpretability. Moreover, adding goal information to the decision-cost module presents a cost-effective strategy for incorporating the high-level goal of the autonomous vehicle AV into prediction models without explicitly relying on a predefined route or endpoint. Through quantitative experiments on open-loop planning metrics, we hypothesize that the selected trajectory of AV can be utilized as a valid initialization for the planning module, as the selected trajectory adheres to the downstream planning task. Additionally, experiments with goal integration during training show improved alignment of AV-predicted trajectories with high-level goals.

References

[1] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social LSTM: Human Trajectory Prediction in Crowded Spaces. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, 2016.

[2] H. Caesar, V. Bankiti, A.H. Lang, S. Vora, V.E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal

dataset for autonomous driving. *arXiv preprint*, arXiv:1903.11027, 2019.

[3] M.F. Chang, J.W. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays. Argoverse: 3d tracking and forecasting with rich maps. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[4] H. Cui, V. Radosavljevic, F. Chou, T. Lin, T. Nguyen, T. Huang, J. Schneider, and N. Djuric. Multimodal Trajectory Predictions for Autonomous Driving using Deep Convolutional Networks. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 2090–2096, 2019.

[5] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C.R. Qi, Y. Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2021.

[6] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid. VectorNet: Encoding HD Maps and Agent Dynamics from Vectorized Representation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[7] R. Girgis, F. Golemo, F. Codevilla, M. Weiss, J.A. D'Souza, S.E. Kahou, F. Heide, and C. Pal. Latent Variable Sequential Set Transformers For Joint Multi-Agent Motion Prediction. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2022.

[8] B. He and Y. Li. Multi-future Transformer: Learning diverse interaction modesfor behaviour prediction in autonomous driving. *IET Intelligent Transport Systems*, 16(9):1249–1267, 2022.

[9] R. Huang, Z. Guirong, X. Lu, L. Xiong, S. Lu, and W. Tian. A Review of Deep Learning-Based Vehicle Motion Prediction for Autonomous Driving. *Sustainabilty*, 15(20):14716, 2023.

[10] 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.

[11] D.B. Johnson. Finding all the elementary circuits of a directed graph. *SIAM J. Comput.*, 4(1):77–84, 1975.

[12] D. Kingma and J.Ba. Adam: A method for stochastic optimization. *arXiv preprint*, abs/1412.6980, 2014.

[13] N. Lee, W. Choi, P. Vernaza, C.B. Choy, P.H.S. Torr, and M. Chandraker. DESIRE: Distant Future Prediction in Dynamic Scenes with Interacting Agents. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[14] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun. Learning Lane Graph Representations for Motion Forecasting. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020.

[15] J. Ngiam, B. Caine, V. Vasudevan, Z. Zhang, H.T.L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal, D. Weiss, B. Sapp, Z. Chen, and J. Shlens. Scene Transformer: A unified architecture for predicting multiple agent trajectories. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2022.

[16] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2019.

[17] L. Rowe, M. Ethier, E.H. Dykhne, and K. Czarnecki. FJMP: Factorized Joint Multi-Agent Motion Prediction over Learned Directed Acyclic Interaction Graphs. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[18] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone. Trajectron++: Dynamically-Feasible Trajectory Forecasting With Heterogeneous Data. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020.

[19] H. Song, W. Ding, Y. Chen, S. Shen, M.Y. Wang, and Q. Chen. Pip: Planning-informed trajectory prediction for autonomous driving. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020.

[20] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J.K. Pontes, D. Ramanan, P. Carr, and J. Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2021.

[21] ZF. Autonomous driving: An overview. Accessed on January 17th, 2023.

[22] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kummerle, H. Konigshof, C. Stiller, A. de La Fortelle, et al. Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps. *arXiv preprint*, arXiv:1910.03088, 2019.

Appendix B

Poster

Evaluation of Traffic Scene Evolutions Using Scene-Consistent Prediction

Dhagash Desai, Benedikt Mersch, Julian Schmidt, and Cyrill Stachniss

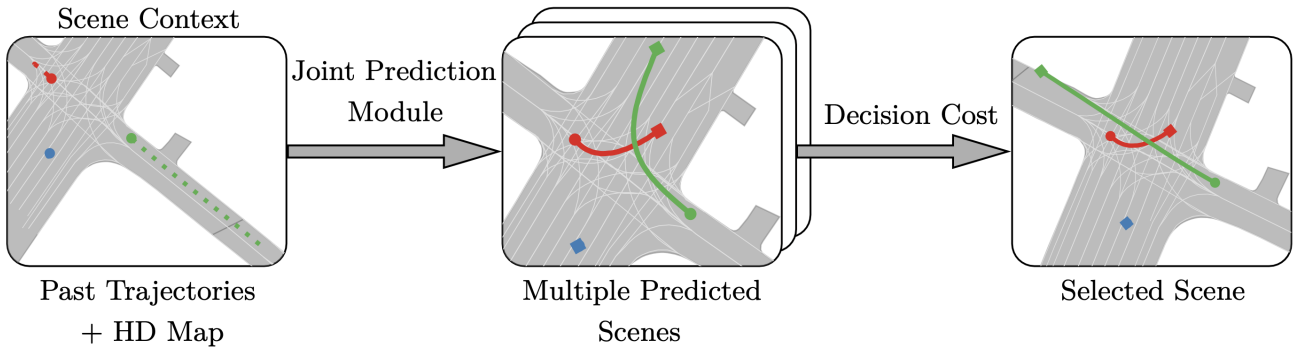
UNIVERSITÄT BONN

Abstract

- Prediction of surrounding agents is essential to navigate safely, reliably in urban environments
- Leveraging advances in scene-consistent motion prediction to provide initialization to downstream tasks
- We employ decision cost module to evaluate predicted scenes adhering to planning task

Method

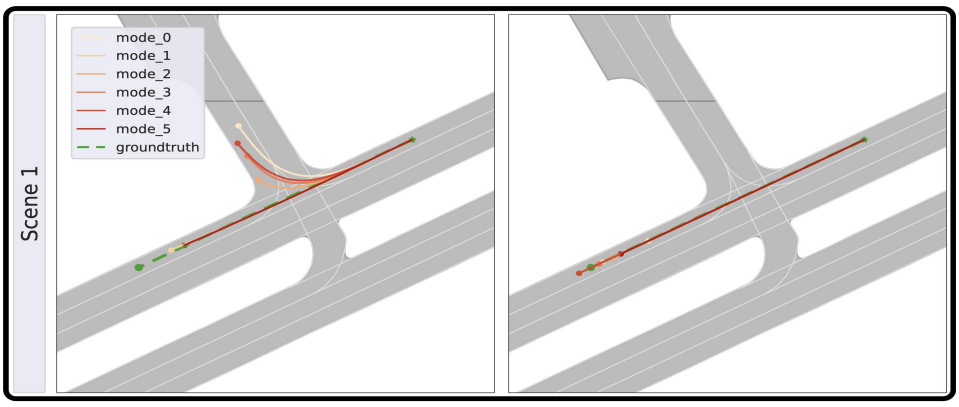
- We generate multiple possible evolutions of traffic scenes from predictor, where each predicted scene includes the future trajectory of all agents present in the scenario, including the autonomous vehicle (AV)
- We assign cost to each predicted scene based on various metrics such as collision, adherence to traffic rules and goal-alignment for AV
- The scene corresponding to least cost is then selected as most likely scene



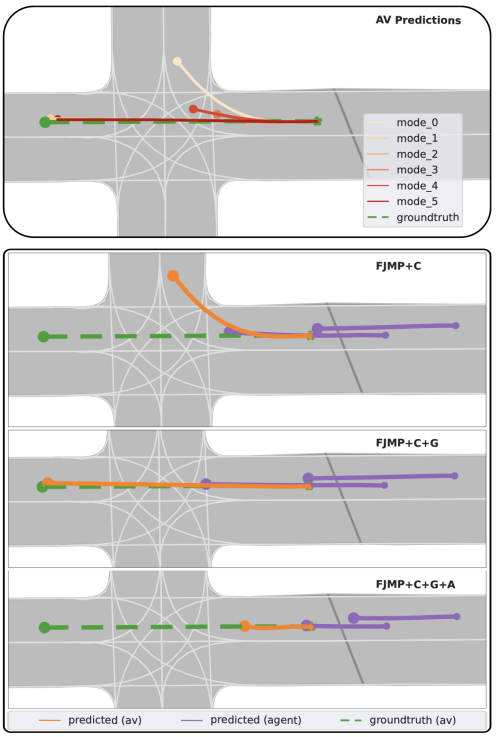
An overview of our proposed pipeline

Experiments and Results

- The scene selected using the decision cost module provides a more informed prediction and offers interpretability
- We provide two approaches of goal integration of AV, to generate goal conditioned predictions
- We hypothesize through experiments, selected trajectory of AV can be utilized as a valid initialization for the planning module



Comparison between models with and without goal conditioning. Left: Without goal. Right: With goal.



Effectiveness of decision cost module