



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

Ph.D. DISSERTATION

Learning-Based Autonomous Vehicle
Navigation Using Road Graph Networks

도로 그래프 네트워크를 이용한 학습 기반의 자율주행
네비게이션

BY
TAE OH HA

FEBRUARY 2023

DEPARTMENT OF ELECTRICAL AND COMPUTER
ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Learning-Based Autonomous Vehicle Navigation Using Road
Graph Networks

도로 그래프 네트워크를 이용한 학습 기반의 자율주행
네비게이션

지도교수 오 성 회

이 논문을 공학박사 학위논문으로 제출함

2023 년 1 월

서울대학교 대학원

전기·정보 공학부

하 태 오

하태오의 공학박사 학위논문을 인준함

2023 년 1 월

위 원 장	<u>최 진 영</u>
부 위원장	<u>오 성 회</u>
위 원	<u>조 남 익</u>
위 원	<u>양 인 순</u>
위 원	<u>최 성 준</u>

Abstract

This dissertation focuses on the learning-based autonomous navigation problem. To deploy an autonomous framework to the real world, it is necessary to design a general-purpose controller that can operate in various road environments. Previous methodologies have used case-specific methodologies, which means that a specific road environment is assumed for each case when designing a controller. A controller designed in this case-specific manner cannot properly operate in an unseen road environment which has not been addressed before. In addition, since the types of road environments are diverse, developing a general-purpose autonomous driving controller requires a lot of development time and cost. Therefore, a new kind of methodology, which is not limited to a specific road environment, is required to develop a general-purpose autonomous driving controller.

This dissertation aims to design a controller which operates universally without assuming a specific road environment. For this purpose, we employ a learning-based controller and focus on improving general driving performance through learning in various environments. In particular, to make the controller capture the features of various road environments, we suggest providing information about a road environment as input for the controller. Road environment information is encoded using a graph, and the controller enhances the driving performance by learning the encoded features of the road graph. This dissertation deals with three major issues related to utilizing a road graph.

First, this dissertation deals with a graph-based methodology for encoding road environment information. In general, the movement of a vehicle on the road is affected by the shape of the road. Therefore, when figuring out the location information of the vehicle, the shape of the road on which the vehicle is located must be considered together. For this purpose, we propose a representation method which reflects roads in the form of graphs and links the location information of vehicles to road graphs. We also propose a novel network structure capable of

processing such road graph representations. Experimental results demonstrate that encoding a state by a road graph helps to generalize the controller’s driving environment.

Second, this dissertation deals with a methodology that fuses road environment information with other sensor data. A road graph represents the structure of a road and the location of nearby vehicles but does not represent other information required to drive in a real world road environment. For example, information such as road signs and traffic signals is difficult to express by a road graph. To address this issue, we propose to combine a road graph with other sensor data such as images and LiDAR. Information that could not be obtained by a road graph is supplemented through other sensor data information. We also propose a novel network architecture which can fuse a road graph with various sensor data. Through the proposed network architecture, the controller succeeds in autonomous driving even in complex road environments. Experimental results demonstrate that the proposed fusion-based method helps to figure out the road environment state.

Finally, this dissertation deals with a methodology that can detect possible errors on road graphs and thereby prevent the degradation of autonomous driving performance. A road graph based controller requires a road graph database to be prepared in advance. However, roads can be continually changed due to several reasons, such as road construction. If these changes are not reflected in the road graph database, the controller receives incorrect road information as input. A slow update to the road graph can cause controller performance degradation, and therefore an error detection method is necessary to prevent such degradation. For this purpose, we propose a methodology which can detect road graph errors. We first define errors that may occur due to road changes and propose road graph change detection modules which can detect these errors. Experimental results show that road graph change detection can be used to improve the performance

of an autonomous driving controller.

Keywords: Autonomous Driving, Road Graph, Reinforcement Learning, Imitation Learning, State Representation, Navigation, Image Processing, Object Detection

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Organization of the dissertation	3
2	Background	7
2.1	Learning Algorithm for Autonomous Driving	7
2.1.1	Reinforcement Learning	8
2.1.2	Imitation Learning	11
2.2	State Representation for Autonomous Driving	15
2.2.1	Feature-Based Representation	15
2.2.2	Bird's Eye View Image Representation	21
2.2.3	Egocentric View Image Representation	23
2.2.4	Sensor Fusion Based Representation	27
3	Road Graphical Neural Networks for Autonomous Driving	31
3.1	Problem Setting	33
3.2	Proposed Method	40
3.2.1	Node-and-Edge-Level Encoding	42
3.2.2	Graph-Level Encoding	42
3.2.3	Time-Level Encoding	43
3.2.4	Learning Algorithm	44

3.3	Experiments	44
3.3.1	Environment and Network Details	44
3.3.2	Experimental Results	46
3.3.3	Qualitative Results	50
3.4	Chapter Summary	53
4	Road Graph and Image Attention Network for Autonomous Driving	59
4.1	Problem Setting	62
4.2	Proposed Method	63
4.2.1	Feature Encoder	63
4.2.2	Attention Network	67
4.2.3	Low-Level Controller	69
4.2.4	Learning Algorithm	70
4.3	Experiments	70
4.3.1	Experimental Settings	71
4.3.2	Dataset	71
4.3.3	Implementation Details	72
4.3.4	Baselines	74
4.3.5	Comparison Results	76
4.3.6	Attention Score Visualization	77
4.3.7	Road Graph Feature Analysis	77
4.3.8	Ablation Study	81
4.3.9	Qualitative Results	83
4.4	Chapter Summary	84
5	Road Graph Change Detection for Autonomous Driving	89
5.1	Problem Setting	92
5.2	Proposed Method	95

5.2.1	Controller Module	96
5.2.2	Road Change Detection Module	98
5.3	Experiments	102
5.3.1	Environments	102
5.3.2	Performance of Road Graph Based Controller Module . . .	103
5.3.3	Performance of Road Change Detection Module	109
5.3.4	Robustness of Controller under Road Change Condition . .	112
5.4	Chapter Summary	114
6	Conclusion	119
	Appendices	121
A	Collected map images of roundabout environment	123
B	Map image and road graph of CARLA Town05	125
C	Details of CARLA evaluation metrics	129
D	Detailed results of CARLA experiment in Chapter 4	133
E	Detailed results of CARLA experiment in Chapter 5	139
F	Examples of FMTC real-world dataset scenarios	143
G	Effect of localization error on road change detection accuracy	149
H	Analysis of detection accuracy according to the degree of road changes	153
I	Ablation study about performance consistency in CARLA environment	157

List of Figures

1.1	An example of road graph representation	2
2.1	An example of distance features	16
2.2	An example of longitudinal and lateral positions	17
2.3	An example of a Frenet coordinate frame	18
2.4	Two different example scenarios on a straight road environment . .	20
2.5	An example of bird's eye view images	22
2.6	An example of egocentric view images	26
2.7	An example of LiDAR sensor data in the CARLA simulator	29
2.8	An example of LiDAR sensor data in the real world	30
3.1	Overview of training a Road-GNN based controller and autonomous driving using Road-GNN in unseen environments	34
3.2	An example of a roundabout environment	36
3.3	The overall process of Road-GNN	41
3.4	Examples of collected map images	45
3.5	Learning curve of RL algorithms	48
3.6	Experiment with various difficulty levels	51
3.7	Roundabout environments used for showing qualitative results . .	54
3.8	Snapshots of a simulation result of Road-GNN	55
3.9	Snapshots of a simulation result of Road-GNN	56

3.10	Snapshots of a simulation result for comparison	57
3.11	Snapshots of a simulation result for comparison	58
4.1	Illustration of our main idea	61
4.2	The structure of the feature encoder module	64
4.3	The structure of the networks	69
4.4	Visualization of attention scores	79
4.5	Road graph feature analysis	80
4.6	Turn right and avoid a pedestrian scenario	85
4.7	Turn left and avoid a pedestrian scenario	86
4.8	Crossing an intersection scenario	87
4.9	Turn right and avoid a pedestrian scenario	88
5.1	An example of a road graph and road changes in the CARLA simulator	94
5.2	The structure of the proposed framework	96
5.3	Illustration of the proposed road change detection methods	99
5.4	Vehicle platform and sensor configuration	104
5.5	Snapshot of FMTC environment	105
5.6	Illustrations of four real-world experience scenarios	106
A.1	Examples of collected map images	124
B.1	HD-map and global road graph of Town05	126
B.2	A part of Town05 map example	127
F.1	FMTC dataset example (Scenario 1: Go straight)	144
F.2	FMTC dataset example (Scenario 2: Turn right)	145
F.3	FMTC dataset example (Scenario 3: Turn left)	146
F.4	FMTC dataset example (Scenario 4: Stop behind another vehicle)	147

G.1	Effect of localization error on road change detection accuracy . . .	151
H.1	Analysis of detection accuracy according to the degree of road changes	155

List of Tables

3.1	Network Architecture Parameters	46
3.2	Performance Comparison	49
3.3	Experiment with aggressive drivers	50
4.1	Performance Comparison	78
4.2	Ablation Study	82
5.1	Performance comparison of controller module (CARLA)	108
5.2	Average L_2 norm in meters between the expert waypoints and the waypoints calculated by controller module (FMTC Real-World) . .	108
5.3	Performance comparison of road change detection module by av- erage precision (CARLA)	111
5.4	Performance comparison of road change detection module by av- erage precision (FMTC Real-World)	113
5.5	Performance consistency of RIANet++ under road change condi- tions (CARLA)	115
5.6	Performance consistency of RIANet++ under road change condi- tions (FMTC Real-World)	116
C.1	Infraction and corresponding coefficient	130
D.1	Performance Comparison in Town05 Short	135

D.2	Performance Comparison in Town05 Long	136
D.3	Ablation Study in Town05 Short	137
D.4	Ablation Study in Town05 Long	138
E.1	Performance consistency of RIANet++ under road change conditions (CARLA Town05 Short)	140
E.2	Performance consistency of RIANet++ under road change conditions (CARLA Town05 Long)	141
I.1	Ablation study in Town05 Short	159
I.2	Ablation study in Town05 Long	160

Chapter 1

Introduction

1.1 Motivation

Autonomous vehicle navigation has been studied by a number of researchers for a long time. In particular, learning-based autonomous vehicle navigation has been widely studied with the development of deep neural network. Many existing studies about autonomous driving have been performed using a driving simulator for safety. One of the most popular examples of simulators is the highway simulator using the NGSIM dataset [34]. The NGSIM dataset was collected by taking videos from a highway traffic camera, and several works [17, 15, 41] have constructed an open-loop driving simulator by making vehicles follow the trajectories of the NGSIM dataset. Many other studies have suggested using a closed-loop driving simulator where the movements of vehicles are changed in response to the ego vehicle’s action. In these studies, road structures such as merging roads [59, 54], intersections [45], or city streets [24] have been used to test the driving performance of an autonomous vehicle. In most of their works, however, they have used the same road structure for both training and testing. This can cause a generalization issue when the vehicle controller is deployed into a real world environment. If the driving agent is tested in a new unseen environment, it is

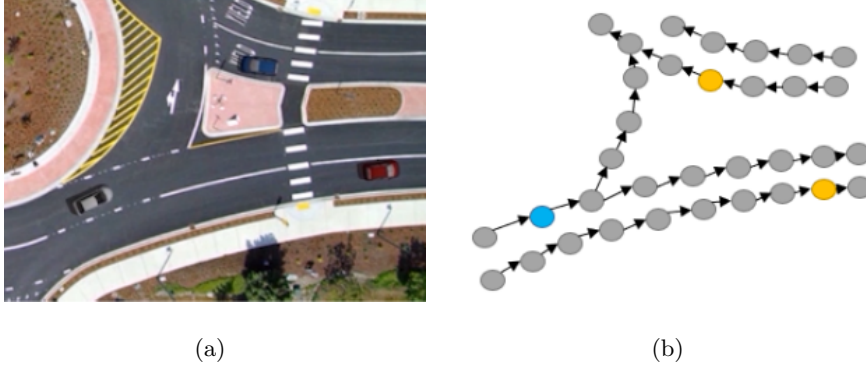


Figure 1.1: An example of road graph representation. (a) Bird's eye view image of a urban driving environment. (b) Corresponding road graph.

likely to fail to drive because it lacks the knowledge of the new environment.

When training a generalized framework which can drive in various road environments, one of the most important issues for a driving agent is how to capture the structural information of a road. Road graph representation is one of the methods to express the structural information of a road. Road graph is a graph which shows the shape and structure of a road. Figure 1.1 shows an example of the road graph. In a road graph, each node is a sampled point at which a vehicle can be located on the road. In addition, each edge is connected when the vehicle can move between each node. Through this road graph, it is possible to represent the road's positional and directional information. A road graph also can have various features for each node and edge. For example, node or edge features can contain road direction information, or traffic information.. Through this method, an autonomous driving agent can effectively obtain useful information.

Unlike the state presentation used in the existing autonomous driving problem, the road graph presentation has the advantage of effectively transmitting road information. In the case of the feature vector based presentation, there is a limit to expressing complex and various forms of roads with only simple vectors. In the case of image based representation, it requires more memory to fully show the road

structure, compared to the road graph. Therefore, the road graph presentation is a more suitable state representation for the autonomous driving problems.

Several works have represented this structural information in the form of a graph. Gao et al. [29] vectorized components on a road and expressed the relationship between each vector as a graph. Similarly, Liang et al. [52] proposed a graph representation which can represent lanes in a road. However, they only focused on the prediction of vehicle movements, and their works have not been applied in the field of vehicle control. In addition, their works have a limitation because their frameworks use the dataset [10] which is collected from relatively simple road environments such as an intersection.

In this dissertation, we focus on enhancing the generalized performance of autonomous driving by providing a road graph based state representation. We also provide a novel driving framework which can leverage both a road graph and other sensor data. In addition, we focus on several issues which can occur when a road graph representation is applied to a real world environment. For the robustness of our framework, we aim to detect errors in a road graph which can occur by road changes.

1.2 Organization of the dissertation

From Chapter 2 to Chapter 5, we cover several research topics about road graph presentation and its application. In this section, we explain the organization of the dissertation.

In Chapter 2, we introduce backgrounds of learning-based automatic vehicle navigation. We mainly cover learning-based frameworks which is widely used in autonomous driving. We introduce reinforcement learning and imitation learning framework. We then explain how each framework has been applied in the autonomous driving field. We also cover various types of state representation which

Chapter 1. Introduction

are used as inputs for learning-based driving frameworks. We describe the advantages and disadvantages of each representation method and explain how each method is used in autonomous driving.

In Chapter 3, we propose road graph neural network (Road-GNN), which is an autonomous driving framework using a road graph presentation. Road-GNN consists of several encoding processes. Road-GNN first encodes node and edge features of a road graph using a deep neural network. Road-GNN then encodes the road graph information. The encoded features are recorded in historical memory and combined together. Finally, Road-GNN takes the final encoded features as an input and is trained with a reinforcement learning framework. In the experiment, we demonstrate that the proposed representation method is more effective than other methods when training a learning-based driving agent.

In Chapter 4, we propose road image attention network (RIANet), which can solve an issue in a road graph based controller. In a road graph, crucial information for driving, such as a traffic light or sign, is ignored. Since Road-GNN uses only a road graph as an input, it does not consider this crucial information when driving in the real world. To solve this issue, we propose a fusion-based driving framework which can leverage information from both road graphs and other sensor data. In the experiment, we experimentally show that the proposed framework outperforms the other frameworks by fusing the sensor data.

In Chapter 5, we propose a detection framework which detect a road change error in a road graph. A road graph can be changed due to road construction, and this change can reduce the performance of a driving agent. To solve this issue, we propose several road change detection methods to filter out unreliable and inaccurate road graphs. We also suggest a robust driving framework named RIANet++, which combines the proposed road change detection modules with the road graph based controller module. Through the proposed framework, we can measure the unreliability of a road graph and consider it when controlling

the ego-vehicle.

Chapter 2

Background

In this chapter, we introduce learning-based frameworks which are widely used in autonomous driving. There are two major issues related to training an autonomous driving agent. First, deciding how to train an autonomous driving agent is an important issue because the driving performance of an agent is influenced by its learning algorithm. Second, deciding which information to give is another important issue because the control of an agent is influenced by its input. Those two issues are essential for training an agent and have been addressed in various studies. The remainder of this chapter covers those two issues and introduces various methods related to each issue.

2.1 Learning Algorithm for Autonomous Driving

There are several learning-based autonomous driving frameworks. Unlike other non-learning-based frameworks, such as model predictive control (MPC), learning-based frameworks enable improving performance without requiring complex physical vehicle modeling. Reinforcement learning and imitation learning are among the most frequently and widely used learning-based frameworks. These two frameworks are generally combined with deep learning for scalability. Unlike other

Chapter 2. Background

methods, deep learning based frameworks can handle a high dimensional input, such as a camera image or LiDAR cloud point data.

2.1.1 Reinforcement Learning

Reinforcement learning (RL) is a framework which trains an agent through a specific pre-defined reward. In a reinforcement learning framework, the autonomous driving problem is formulated using a Markov decision process (MDP). For each timestep t , the agent observes the current state s_t . The agent then uses the policy π to determine an action a_t . The action a_t is a control signal which can include a target speed, acceleration, or steering. Each action may be a useful control signal which drives a vehicle safely or, otherwise, a harmful control signal which causes an accident. As a result of the action, the agent's state is changed to the next state s_{t+1} . During the training, the agent receives a pre-determined reward r_t according to the result of the action. The goal of reinforcement learning is to maximize the expected sum of rewards $\mathbb{E}_\pi[\sum_t \gamma^t r_t]$, where γ is a discount factor.

One of the most popular reinforcement learning algorithms is Deep Q-Network (DQN) [56]. DQN is also known as a Q-learning based reinforcement learning framework. In DQN, a neural network named Q-network predicts a state-action value of each action, which is the expected sum of rewards. The state-action value Q is defined as follows:

$$Q(s, a) = \mathbb{E}[\sum_t \gamma^t r_t | s_t = s, a_t = a] \quad (2.1)$$

Here, the policy of the agent is defined as an argmax policy.

$$\pi(s_t) = \arg \max_a Q_\theta(s_t, a), \quad (2.2)$$

where Q_θ is a Q-network and θ is a training parameter. At each iteration, Q_θ is updated using the following loss function:

$$L = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}_{exp}} [(r + \gamma \max_{a'} Q_\theta(s', a') - Q_\theta(s, a))^2], \quad (2.3)$$

where \mathcal{D}_{exp} is a set of experiences of the agent.

In contrast to DQN, trust region policy optimization (TRPO) [72] is an actor-critic based reinforcement learning framework. Unlike Q-learning, an actor-critic based framework separates the network that calculates a policy from the network that calculates a value. Each network is named as a policy network π (actor) and a value network V (critic), respectively. In TRPO, the networks are trained while solving the following optimization problem.

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & \mathbb{E}\left[\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t\right] \\ \text{subject to} \quad & \mathbb{E}[\text{KL}[\pi_{\theta_{old}}(\cdot|s_t), \pi_{\theta}(\cdot|s_t)]] \leq \delta, \end{aligned} \quad (2.4)$$

where KL is a KL divergence, θ_{old} is a training parameter before the update, and $\hat{A}_t = -V(s_t) + \gamma^{T-t}V(s_T) + \sum_{k=t}^{T-1} \gamma^{k-t}r_k$ is an advantage estimator which indicates how much the expected sum of reward will be improved compared with the current state. The constraint of Equation 2.6 is called a trust region. The trust region constraint prevents the policy from being changed too much when it is updated. In this way, TRPO guarantees a monotonic improvement of the policy network π . The value network V is updated by a square-error loss L^{VF} .

$$L^{VF} = (V_{\theta}(s_t) - V_t^{target})^2, \quad (2.5)$$

where V_t^{target} is a target value computed from the sampled experience. PPO computes \hat{A}_t through the value network V for each iteration.

Proximal policy optimization (PPO) [73] simplifies the optimization process of TRPO. PPO denotes the probability ratio $\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ as $R(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$. Instead of KL divergence constraint, PPO uses a clipped surrogate objective as follows.

$$\underset{\theta}{\text{maximize}} \quad \mathbb{E}[\min(R_t(\theta)\hat{A}_t, \text{clip}(R_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)], \quad (2.6)$$

where clip is a clipping operator which clips the surrogate objective $R_t(\theta)\hat{A}_t$ between the interval $[(1 - \epsilon)\hat{A}_t, (1 + \epsilon)\hat{A}_t]$ with a small margin ϵ .

Chapter 2. Background

Reinforcement learning frameworks have been applied to several works in the field of autonomous driving. In many works, a driving simulator is used to train a deep learning based reinforcement learning policy. For example, DQN [56] was applied by Nishitani et al. [59] to train an agent which drives on a merging road. To enhance the learning efficiency, they also suggested training an embedding network which predicts the vehicle speed. In their framework, the reward function is designed to minimize the impact of the merging on traffic flow. The agent obtains a zero reward while the ego vehicle is on the ramp. After the merging is completed, the agent obtains the merging completion reward according to the average speed of all vehicles. To minimize the reward, the agent must merge smoothly and make no slowdown in the traffic flow. PPO [73] has also been used by several works to train a driving agent. Saxena et al. [70] used PPO to train a driving agent who can drive on a dense and crowded road. In their work, they suggested a reward function and an observation which are carefully designed for driving in dense traffic. Likewise, Osiński et al. [60] trained a driving agent using PPO. Compared to the other methods, they used a more realistic driving simulator which mimics a real world road structure and scenario.

There are two major difficulties when applying a reinforcement learning framework to the autonomous driving problem. First, it is difficult to give full state information to the agent. The state information of a real world environment can be obtained from a sensor such as a camera. However, it is impossible to grasp perfect state information because a sensor may be noisy and not perfect. For example, distortion of a camera lens, low resolution, or limited field of view can make it challenging to identify surrounding objects accurately. In addition, an occlusion by an object can block the field of view of a camera. Partially observable Markov decision process (POMDP) has been proposed to deal with this issue. In POMDP, the observation o_t , which represents the information from the sensor data, is used instead of the state. The observation o_t can include various sensor

data such as RGB camera image, LiDAR sensor data, or speedometer. The type of sensors can vary depending on the problem. Second, it is difficult to train a reinforcement learning policy in a real world environment. In reinforcement learning, an agent continuously interacts with the environment. An agent should avoid moving obstacles and should not violate the traffic light signal. An agent failing to follow traffic rules can occur dangerous situations such as collisions and injuries. However, since a reinforcement learning based agent does not work well at the beginning of training, it is not possible to train an agent without any accident. In addition, an agent cannot learn to avoid undesirable actions without an accident because it requires receiving a negative reward by experiencing such a dangerous situation beforehand. Therefore, it is not possible to train an agent in a real world environment. For safety, many existing works have used a driving simulator for training an agent. Now a day, driving simulators can provide a realistic road environment. For example, CARLA simulator [25] can simulate jaywalking pedestrians, traffic lights, or weather changes. However, even if a simulator is similar to a real world environment, there is inevitably a difference between a real world environment and a simulator. This difference makes it difficult to deploy a reinforcement learning based agent in a real world environment.

2.1.2 Imitation Learning

Imitation learning (IL) is a framework which trains an agent using a pre-collected expert dataset. The goal of imitation learning is to find an optimal policy from expert demonstrations. In detail, an agent is given a set of expert demonstrations, which is defined as follows:

$$\mathcal{D}_{\pi_E} = \{(s_0, a_0), (s_1, a_1), \dots, (s_T, a_T)\}, \quad (2.7)$$

where T is a demonstration length. \mathcal{D}_{π_E} is assumed to be collected by an expert policy π_E . The expert policy π_E maximizes the expected sum of rewards

Chapter 2. Background

$\mathbb{E}_\pi[\sum_t \gamma^t r_t]$, where r_t is a reward and γ is a discount factor. In contrast to reinforcement learning, however, the reward r_t is hidden and not known to the agent. Therefore, the agent should find an optimal policy without knowing the reward function.

One of the most straightforward frameworks for imitation learning is behavior cloning (BC). A behavior cloning framework trains a policy in a supervised learning manner. In a general BC framework, a policy is defined using a neural network. The network is trained with the following log-likelihood loss function:

$$L = -\mathbb{E}_{(s,a) \sim \mathcal{D}_{\pi_E}} [\log \pi_\theta(a|s)], \quad (2.8)$$

where θ is a training parameter.

Another example of imitation learning frameworks is generative advertising imaging learning (GAIL). Similar to [31], GAIL trains a policy by adversarial learning between a generator (policy) and a discriminator. The objective of GAIL is formulated as follows:

$$\min_{\pi} \max_D \mathbb{E}_{\pi} [\log(D(s, a))] + \mathbb{E}_{\pi_E} [\log(1 - D(s, a))] - \lambda H(\pi), \quad (2.9)$$

where $D : S \times A \mapsto (0, 1)$ is a discriminator, $H(\pi) = \mathbb{E}_{\pi} [-\log \pi(a|s)]$ is the causal entropy of the policy π , and λ is a scale parameter. The discriminator D returns the probability of whether a given demonstration is from the agent policy π or the expert policy π_E . During the optimization process, the discriminator D is trained to discriminate the input demonstrations while the policy π is trained to trick the discriminator and generate a demonstration which is similar to the expert demonstrations.

There are a number of examples where an imitation learning framework is applied to the autonomous driving problem. A behavior cloning framework is used to train a driving agent by several works. Dosovitskiy et al. [25] suggested using an urban scene simulator to train a driving agent. They provide a bench-

mark for urban scene driving and compare the training results of a reinforcement learning and imitation learning based agent. Codevilla et al. [20] improved the performance of urban driving by investigating the limitations of a behavior cloning based controller. In particular, they propose to divide the module into two parts: perception network and low-level controller. The perception network part calculates the future trajectory of the ego-vehicle through imitation learning, and the low-level controller part follows the calculated trajectory through PID control. Chen et al. [11] proposed a two-stage learning process which can overcome the limitation of behavior cloning. First, they train a privileged agent which can access privileged information. This privileged information includes a bird’s eye view map of the scene, which contains the indicators of objects, road features, and traffic lights. Second, they train a sensorimotor agent, which can only perceive the scene via sensors. The privileged agent works as a teacher to the sensorimotor agent and provides online expert trajectories. This two-stage process helps to boost the performance of a vision-based driving agent. GAIL [43] is also a widely used framework in autonomous driving. Bhattacharyya et al. [5] suggested a GAIL based driving policy generation framework which can simulate a multi-agent driving scenario. Lee et al. [50] also trained a GAIL based agent which can drive in a simulator. In their framework, the agent is trained not only using expert demonstrations but also negative demonstrations, which can teach the agent what should not be performed.

Compared to reinforcement learning, an imitation learning framework has two advantages. First, imitation learning does not require designing a reward function. In order to use reinforcement learning, it is necessary to design a reward suitable for the situation according to each state and action. For example, it can be designed to give a positive reward if the vehicle drives normally and a negative reward if the vehicle collides with another vehicle. However, it is difficult to define in which situations to give rewards and how many rewards to give. If the

Chapter 2. Background

reward is not adequately defined, the vehicle will not operate as a regular human driver. Therefore, the reward design is a complex task and requires much labor. On the other hand, imitation learning is relatively less complex because it only needs to collect expert demonstrations without a reward design. Second, imitation learning is relatively secure for safety issues which can occur during the training process. As explained in Section 2.1.1, reinforcement learning has a safety issue while training an agent in a real world environment. A reinforcement learning agent should experience a collision before learning to avoid it. This approach consequently leads to an accident in a real world environment. An imitation learning agent, on the other hand, does not have to experience an accident to learn to avoid it. Also, data for imitation learning can be collected by an expert policy, which is less likely to commit accidents. This makes the process of imitation learning relatively secure compared to reinforcement learning.

Imitation learning also has a disadvantage compared to reinforcement learning: an agent trained by imitation learning may be vulnerable to environmental changes because interactions with the environment are not performed when training the agent. For example, assume that expert data is collected while following the exact center of a road. If an agent is fully-trained and exactly follows the given trajectory, it can imitate the expert policy and successfully drive. However, suppose that the agent slightly deviates from the trajectory. In this case, the agent will not be able to successfully correct the deviation because those actions are not collected in the dataset. This data mismatching issue is generally called a covariance shift. This issue can be easily solved in reinforcement learning by giving a deviation value as a negative reward. It makes the agent learn how to recover the deviation when the vehicle is off the center of a road. In imitation learning, this issue can be solved by collecting sufficient data from scratch or by continually collecting additional data with the expert policy [69].

2.2 State Representation for Autonomous Driving

There are two major points to consider when defining a state representation. First, a state representation should be informative. There is essential information which is needed to solve an autonomous driving problem. For example, knowing the locations of vehicles or pedestrians are essential for safety. Therefore, this essential information should be contained in a state representation or at least inferred through it. Second, a state representation should consist of a compact data structure. There is a limit to the amount of data that a controller can process during driving. If the size of the data is too large, the time complexity of the control will inevitably increase. Since an autonomous driving agent must work in real-time, the data should be as simple as possible.

There are several types of state representation which have been used by existing autonomous driving studies. For example, the following state representations have been used by multiple researchers: feature-based representation, bird's eye view image representation, and egocentric representation. In the remainder of this section, we explain each state presentation and discuss its properties in terms of informativeness and compactness.

2.2.1 Feature-Based Representation

Feature-based representation method represents a state using a single feature vector. A feature vector can be defined in different ways depending on its problem. For example, the distance and relative speed to other vehicles is one of the frequently used features. Feature-based representation has been used in many existing autonomous driving studies because it is simple and easy to calculate in a simulator. Choi et al. [17] assumed a straight highway environment, and they designed a feature vector which contains the frontal distances to the vehicles. More specifically, the feature vector considers the three closest frontal distances on the

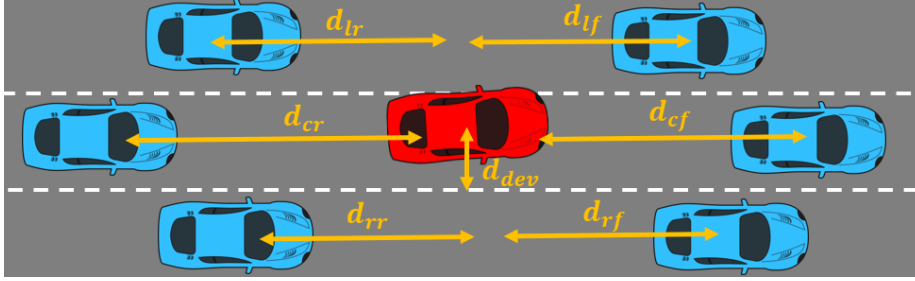


Figure 2.1: An example of distance features. A feature vector consists of the distances from the ego vehicle (red) to three frontal and three rearward vehicles (blue).

left (d_{lf}), center d_{cf} , and right lanes d_{rf} . In addition to the frontal distances, the feature vector contains the lane deviation distance (d_{dev}). Their feature representation is defined as follows:

$$f = (d_{lf}, d_{cf}, d_{rf}, d_{dev}) \quad (2.10)$$

In [18] and [15], a feature vector also contains the rearward distances to other vehicles on the left (d_{lr}), center (d_{cr}), and right lanes (d_{rr}). Their feature representation is extended from Equation 2.10 and defined as follows:

$$f = (d_{lf}, d_{cf}, d_{rf}, d_{lr}, d_{cr}, d_{rr}, d_{dev}) \quad (2.11)$$

Figure 2.1 illustrates each component of a feature vector in Equation 2.11.

Schmerling et al. [71] designed a feature vector using the 2D position of vehicles. In their framework, they assumed that two vehicles are interacting with each other on a highway. The positions of the vehicles are represented in the 2D coordinate system of a highway environment. Their feature vector is defined as follows:

$$f = (s_{ego}, t_{ego}, s_{other}, t_{other}), \quad (2.12)$$

where s_{ego} and s_{other} are longitudinal positions of the ego and the other vehicle, t_{ego} and t_{other} are lateral positions of the ego and the other vehicle. Each feature

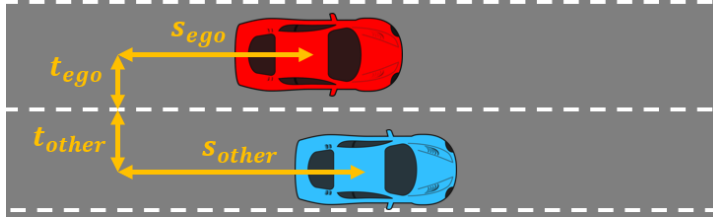


Figure 2.2: An example of longitudinal and lateral positions. The positions of the ego-vehicle (red) and the other vehicle (blue) are computed by the 2D coordinate system of a highway environment.

component is computed by the distance to the origin of the road. Figure 2.2 illustrates an example of a feature vector.

Werling et al. [82] used the position of a vehicle as a feature vector, which is represented by a Frenet coordinate frame [9]. The Frenet coordinate frame is a generalization of the Cartesian coordinate frame. Similar to the Cartesian coordinate, the position of a vehicle in the Frenet coordinate is represented using a longitudinal distance s and a lateral distance t .

$$f = (s, t) \quad (2.13)$$

Unlike the Cartesian coordinate, however, s and t are calculated from a relative distance to a reference path. In the Frenet coordinate, the s -coordinate is defined as the run length of the vehicle, and the t -coordinate is defined as the vertical deviation from the reference path. Figure 2.3 illustrates an example of a Frenet coordinate frame. It is possible for a reference path to have a curved shape. Therefore, the Frenet coordinate frame allows a more intuitive representation of a vehicle's position on a curved road.

There are several issues when adapting a feature-based representation to the autonomous driving problem. First, it is difficult to compute an exact value of a feature in a real world environment. In a simulator environment, features can be easily calculated. In a real world environment, however, it is necessary to first

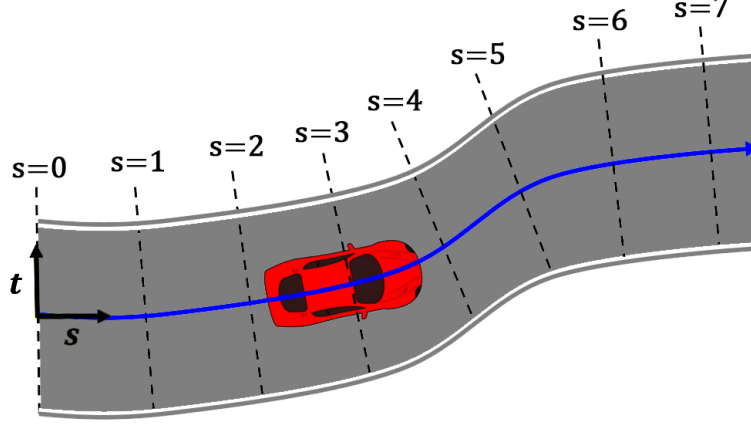
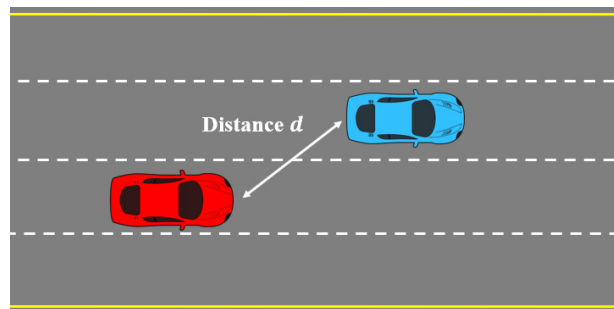


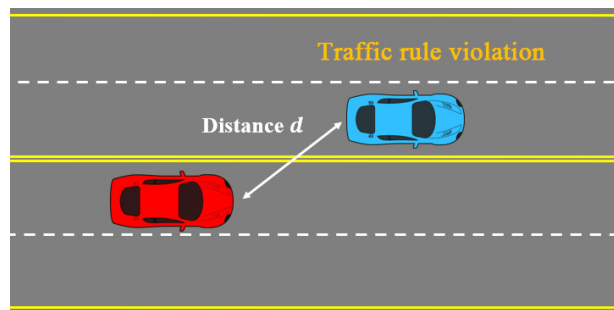
Figure 2.3: An example of a Frenet coordinate frame. A blue curved line indicates a reference path of a road. A s coordinate follows the direction of the reference path, and a t coordinate follows the orthogonal direction of the s coordinate.

perceive sensory data before calculating a feature vector. For example, suppose that there is an agent which tries to compute a positional feature of nearby vehicles through a sensor. In this case, the agent first needs to detect nearby vehicles through sensor data. The agent then needs to find the positions of nearby vehicles by estimating the distances to them. During this process, there is a possibility that detection may fail, or there may be a noise in the estimated distances. Existing studies about autonomous driving [17, 18, 15, 71, 82] have assumed that vehicles' positions can be estimated without an error. However, there is no guarantee that there are no errors when using positional features in a real world environment. This error interferes with the controller's learning process and degrades its performance. Second, it is difficult to quantify a state of a real world environment only with a simple vector feature. One of the features which are difficult to represent by a single vector is the vehicle type. Each vehicle on the road moves differently depending on the vehicle type. Motorcycles, for example, are easy to change lanes, and in some cases, they even pass through the boundary between two lanes. On

the other hand, large vehicles such as buses are relatively less likely to change lanes. However, it is difficult to vectorize the vehicle type because there are too many types of vehicles in the real world. One possible way is to vectorize the vehicle type is a one-hot vector encoding [74, 7]. However, one-hot vector encoding also has a limitation because it can only represent a fixed number of vehicle types. Zero-shot learning framework [2], which can classify unseen labels, has been suggested to solve this issue. However, zero-shot learning requires learning semantic information of the class before classification. Finally, it is difficult to define a feature vector which can reflect road structural information. The movement of a vehicle is affected by the shape of a road. Since the movement of the vehicle is greatly affected by the shape of the road, such road structural information must be fed to a controller when a vehicle is driven. For example, suppose that there are two different scenarios: a one-way road environment (Figure 2.4(a)) and a two-way road environment (Figure 2.4(b)). In Figure 2.4(a), each vehicle is currently driving safely. Since the two vehicles are not violating traffic rules, it can be said that the probability of an accident is low. In Figure 2.4(b), on the other hand, the blue vehicle is violating a traffic rule by crossing the center line. In this case, it can be easily predicted that the blue vehicle will either cause an accident with an oncoming vehicle or attempt to cut into the lane where the red vehicle is currently located. Therefore, the two scenarios are significantly different in terms of dangerousness. However, because the blue vehicle has the same distance as the red vehicle, the distance feature must be calculated equally in both scenarios. This makes it difficult for a controller to distinguish the state difference between the two scenarios. Therefore, a controller trained only with a feature vector input may not recognize the dangerousness in Figure 2.4(b). A positional feature which is specified by a Cartesian [71] or Frenet [82] coordinate can help solve this issue. By assigning a negative sign when a vehicle is on the other side of the road, a feature vector can represent the difference between the two scenarios. However,



(a)



(b)

Figure 2.4: Two different example scenarios on a straight road environment. (a) A one-way road. Two vehicles are safely driving on the road. (b) A two-way road. The blue vehicle is violating a traffic rule by crossing the center line.

the positional feature also cannot represent a road’s structural information when a road has a complicated shape, such as an intersection or roundabout. Because a reference path cannot have a branch, the positional features are only useful for representing a state on a straight or curved highway environment.

2.2.2 Bird’s Eye View Image Representation

Bird’s eye view (BEV) image representation method represents a state by an image which is captured from the top-down viewpoint. Figure 2.5(a) and Figure 2.5(c) shows an example of BEV images. A BEV image representation contains more information compared to a feature-based representation. For example, a direction and position of a lane, a position of a crosswalk, and a curvature of a road can be inferred through a BEV image.

It is not necessary to represent a bird’s eye view of the scene as a raw RGB image. For example, semantic information such as road lane and vehicle location can be included in a BEV image. Each object on the road can be represented with pixels by colorizing the area where an object occupies. This semantic image is also called a rasterized image. Figure 2.5(b) and Figure 2.5(d) shows an example of rasterized images. A rasterized image is more useful than a raw BEV image because it is relatively easy to identify an object in a rasterized image. Advances in image processing technologies such as object detection [64, 38, 39, 7] and semantic segmentation [68, 12, 13] have helped to extract this rasterized image from a raw BEV image.

Nishitani et al. [59] trained a reinforcement learning based agent using a BEV image input. In their work, the input is given as a rasterized image. Similarly, Henaff et al. [41] used a rasterized BEV image as an input to train an imitation learning based agent which can drive on a highway simulator.

There are several issues when applying BEV image representation to a real world autonomous driving problem. First, it is not possible to capture a BEV

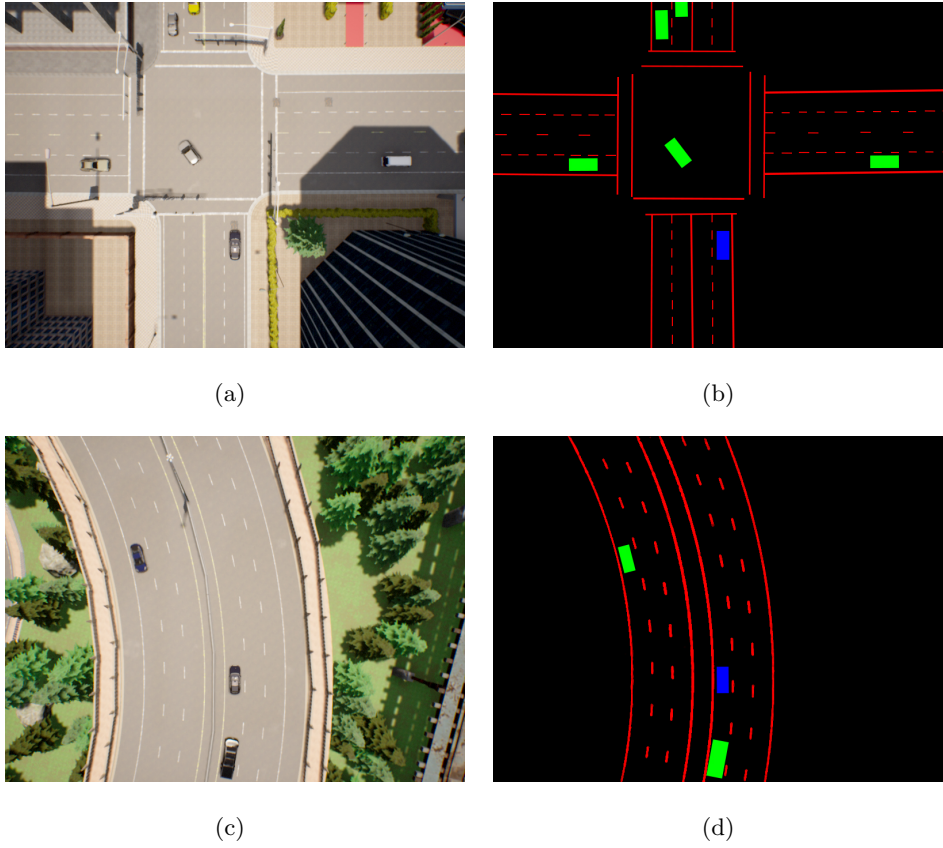


Figure 2.5: An example of bird's eye view images. (Left) Bird's eye view images in the CARLA simulator [25]. (Right) Corresponding rasterized images which contain semantic information. The red lines represent lanes on the road. The blue box represents the ego-vehicle. The green boxes represent the other vehicles.

image in a real world environment. Before taking a BEV image, a camera must be installed in a position which is much higher than that of a vehicle. When moving a vehicle in a real world environment, however, it is not possible to place a camera in the air. Therefore, it is hard to take and use a BEV image in a real world environment. There is another way to use a BEV image in a real world environment. A BEV image can be synthesized from multiple vehicle-mounted camera images [65]. However, in this case, the synthesized BEV image can be incomplete due to occlusions or perspective transform errors. Second, a BEV image does not contain information about traffic, such as traffic signs. The movement of a vehicle is restricted by the traffic signs on the road. For example, a vehicle is prohibited from turning left when there is a no left turn sign on the road. Similarly, if there is a speed limit sign on the road, the vehicle must drive at a speed below the speed limit. However, it is difficult to represent this traffic information on a BEV image. One of the solutions to this issue is to add an image channel, which represents a traffic rule, to the rasterized image. For example, Jain et al. [46] represented the information about traffic lights by three image channels which indicate the state of traffic lights. In their work, the pixels of a channel indicates the area which is controlled by red, yellow, or green traffic light, respectively. Their representation, however, only contains information about traffic lights, not about the other traffic signs, such as speed limit signs. The number of the types of traffic signs can be increased by increasing the number of image channels. However, if the number of image channels is increased, the BEV representation data size will also be increased. Therefore, it is difficult to represent traffic sign information as a BEV image without increasing the data size and complexity of the representation.

2.2.3 Egocentric View Image Representation

Egocentric view image representation method represents a state using an image which is captured from an egocentric camera. An egocentric image is also called

Chapter 2. Background

a first-person view image. A camera for an egocentric image is usually attached directly to a vehicle and shows the vehicle’s surrounding environment. Figure 2.7(a) shows an example of an egocentric view image. There are a number of types of egocentric view cameras which can be used in autonomous driving. Many recent works have used only a single forward-facing camera to drive a vehicle. For example, Dosvitskiy et al. [25] and Codevilla et al. [20] used a forward-facing camera image to train an autonomous driving agent in an urban driving simulator. Hawke et al. [36], on the other hand, trained an agent to drive using multiple camera image inputs. In their work, a state presentation consists of camera images captured from three different viewpoints: left, center, and right. To fuse three different images, they designed a deep learning based fusion network.

An egocentric view image representation not only uses a raw RGB image but also can use a pre-processed image. One of the examples of pre-processing is semantic segmentation [68, 12, 13]. In a semantic segmentation process, the pixels of an image are clustered and classified according to the object class they belong to. Figure 2.7(b) shows an example of semantic segmentation image, which is processed from Figure 2.7(a). In a semantic segmentation image, each pixel has one object class. In Figure 2.7(b), each pixel is colored in different colors according to the object class. For example, roads are purple, vehicles are blue, lane markings are green, and the pixels without class are colored black. Sobh et al. [75] used a semantic segmentation image as an input to a driving agent. In their work, they obtained a semantic segmentation image through U-net [68] architecture.

Object detection [64, 38, 39, 7] is another example of pre-processing methods. In recent years, object detection has been widely used in the field of autonomous driving. The goal of an object detection process is to find the sizes and locations of objects on a 2D image. Figure 2.6(c) shows an example of object detection result, which is processed from Figure 2.7(a). Each object in Figure 2.6(c) is bounded by a colored bounding box. There are two types of objects in Figure 2.6(c): vehicle

and traffic light. Here, two vehicles are surrounded by a red box, and a traffic light is surrounded by a blue box. Behl et al. [3] suggested using object detection based features when training a driving agent. In their work, they represented the positions of detected objects in the form of labeled bounding boxes. Object detection can be applied to various types of objects. For example, in recent years, pedestrian detection [35], and lane detection [53] have been studied by several works. It is not only possible to find the position of an object in 2D. 3D objection detection finds the position of an object in 3D coordinates. Li et al. [51] suggested a 3D object detection framework which can find the 3D position of an object with a single-view image. Wang et al. [81] also suggested a 3D object detection framework but used a multi-view image when finding the position of an object.

Depth estimation [28, 30, 4] is also one of the methods used for pre-processing. A depth image refers to an image which contains distance information. The goal of the depth estimation process is to find a depth image from a given RGB image. Figure 2.6(d) shows an example of depth estimation result, which is processed from Figure 2.7(a). In Figure 2.6(d), the positions of vehicles and traffic lights can be inferred from the depth image. Combined with an RGB image, a depth image is generally called an RGB-D image. Not only from depth estimation, an RGB-D image also can be obtained directly from an RGB-D camera. Xiao et al. [83] used RGB-D images to train an autonomous driving agent. In their work, the simulator gives an accurate depth image to an agent. However, they added noise to the depth image to make the simulation more realistic.

There are several issues when training an autonomous driving agent using egocentric view image representation. First, an agent can see only a local part of a road environment through an egocentric view image. Unlike a BEV image, an egocentric view image cannot show the overall appearance of a road. Therefore, it is difficult for an agent to recognize the structure or shape of a road through an egocentric view image. When a driving agent determines a global path and

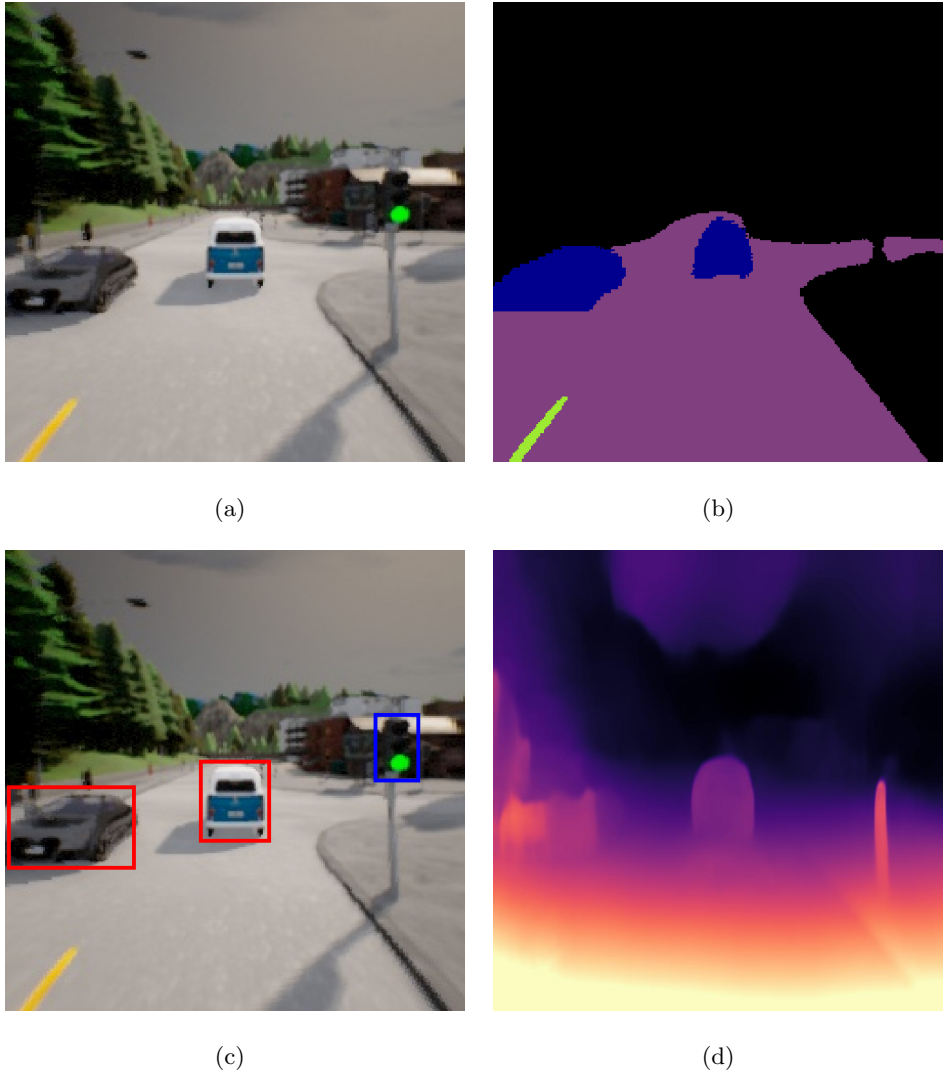


Figure 2.6: An example of egocentric view images. (a) An egocentric RGB image captured in the CARLA simulator. (b) A semantic segmentation result processed from the RGB image. (c) An object detection result processed from the RGB image. (d) A depth estimation result processed from the RGB image.

navigates in a real world environment, the entire structures and locations or roads are needed to be known. However, since an egocentric view image can only show a local part of a road, an agent cannot navigate a road only with an egocentric view image. Therefore, a high-level command [19, 20] or a pre-determined global path [23, 87] are required before an agent starts to drive. Second, the performance of the agent may be sensitive to the sensor location because the viewpoint of an egocentric view image varies depending on the sensor location. For example, suppose that an agent is trained with data collected with a single viewpoint. In this case, if data collected with a different viewpoint is given as new input, the performance of the agent can be decreased. The performance decrease by viewpoint change makes it difficult to reposition the sensor or apply a pre-trained network to another vehicle of a different size. This kind of issue is usually addressed by domain generalization [80].

2.2.4 Sensor Fusion Based Representation

Sensor fusion refers to a method of combining multiple sensors to reduce uncertainty and increase the performance of an agent. In recent years, many types of sensors have been used for autonomous driving. One of the most widely used sensors in the field of autonomous driving is LiDAR. LiDAR is a sensor that can detect surface information of surrounding objects through a laser. Surface information obtained through a LiDAR sensor is generally given through a data structure named a point cloud, which records the locations of a set of points in a 3D coordinate. Figure 2.7 shows an example of LiDAR sensor data in the CARLA simulator. In Figure 2.7, a LiDAR point cloud is projected on a 2D BEV image. An agent in a simulator can obtain precise LiDAR sensor data. However, in a real world environment, the quality of LiDAR sensor data varies depending on the spec of a LiDAR sensor. For example, an object which is too far away cannot be captured because a LiDAR sensor has a limitation in the detectable range.

Chapter 2. Background

Also, noise from a low-quality LiDAR sensor can prevent the recognition of an object. Figure 2.8 shows a LiDAR sensor data captured from a vehicle in the real world.

Several works have proposed a new framework which can fuse a camera image with LiDAR sensor data. Xu et al. [84] suggested a 3D object detection framework which uses an image and LiDAR sensor data. Because existing networks used for image data [37, 66] cannot be directly applied to a point cloud data structure, they used a pointnet architecture [63] to encode LiDAR sensor data before fusing it. Similarly, Vora et al. [79] suggested a 3D semantic segmentation framework which uses a sensor fusion method. Their architecture first obtains a semantic segmentation image from input and then uses a projection method to fuse this image with LiDAR data. Image and LiDAR sensor fusion method has also been applied to train an imitation learning based driving agent. Prakash et al. [62] proposed a fusion based framework for autonomous driving. In their work, LiDAR point cloud data is projected into a BEV image. They fused this LiDAR BEV image with a forward-facing camera image through the attention mechanism [78].

There are other sensors used for fusion, such as GPS, IMU, and speedometer sensor sensors. These sensors are generally used to estimate the location of a vehicle. The estimated location can be used to compute a high-level command and fuse it with an image input [20, 11].

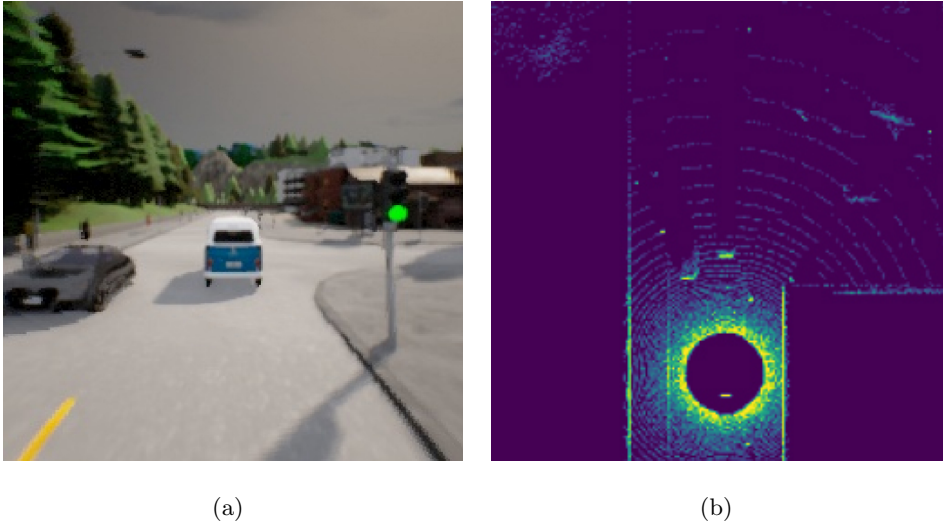


Figure 2.7: An example of LiDAR sensor data in the CARLA simulator. (a) A egocentric image from a camera. (b) LiDAR sensor data captured in the same scene. The LiDAR point cloud is projected into BEV.

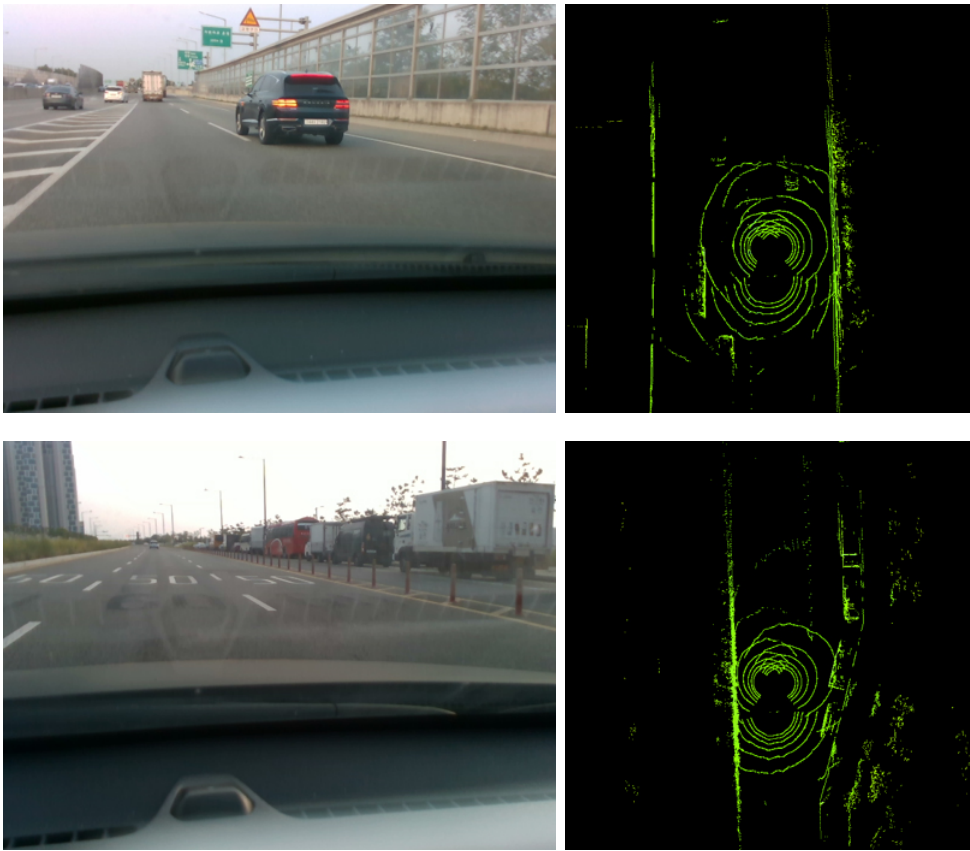


Figure 2.8: An example of LiDAR sensor data in the real world. (Left) Camera images. (Right) LiDAR sensor data captured in the same scene. The LiDAR point clouds are projected into BEV.

Chapter 3

Road Graphical Neural Networks for Autonomous Driving

Along with increasing interests in autonomous driving, the investigation of reinforcement learning (RL) using the driving simulator has been actively carried out by a number of researchers. However, most of the previous works cannot be generalized to other driving environments. One of the most widely used examples is the highway environment [17, 15], but their works are only limited to a single one-way road. There are more complex environments, such as merging roads [59, 54], intersections [45], or city streets [24]. The previous studies, however, use only a simple and fixed road environment for training and testing.

One of the main reasons for the generalization difficulty is that it is difficult to capture the generalized feature of the road environment. For example, [17] and [15] use the relative speed and position of vehicles as inputs features, but they cannot capture information about how vehicles move on a complex road. Another example is the semantic top-view image, which includes road lines [41].

Chapter 3. Road Graphical Neural Networks for Autonomous Driving

However, as shown in the recent studies [29, 52], it is not efficient to use images for representing the topology of roads.

In this chapter, we propose a generalized autonomous driving framework to use graphical representation of roads. The proposed graphical representation includes the information about the structure of roads, and the positional relationship between vehicles and the road. Unlike other works using the graph representation [29, 52], there are four distinctive features of the proposed framework.

- It is the first work which uses graph-based features to control vehicles to the best of our knowledge.
- We do not focus on a simple environment such as highways or merging roads but on various roundabout intersections, where the topology of the road is more complicated and multiple vehicles can enter from multiple directions simultaneously.
- The relationship between vehicles and the road is also considered when calculating graph features
- We use long short-term memory (LSTM) [44] along with graph neural networks (GNN) [47] to capture the historical features of vehicles on the road graph.

The proposed method uses a two-level controller: First, a low-level PID controller follows a trajectory determined by a graph-based path planner. The path planner perceives the road graph and finds a path based on the Dijkstra algorithm. Second, a high-level controller determines the target speed of the vehicle to follow the traffic flow. The high-level controller uses the encoded feature of the road graph, which is extracted by a road graphical neural network (Road-GNN). A Road-GNN compresses the graph features of the road into a latent feature used for the high-level controller. We use the RL framework to train the network, and

Chapter 3. Road Graphical Neural Networks for Autonomous Driving

it successfully controls the vehicle in various and complex unseen road environments. Figure 3.1 shows an overview of how Road-GNN works in training and test phases.

We trained the network from a set of road environments and tested it on a different set to show the generalizability of Road-GNN. We have collected satellite images of roundabouts for training and testing and applied them to our driving simulator to simulate a realistic road environments and structures. Note that the satellite images are only used to construct road models and get state representations, and we use a 3D simulator and a 3D dynamic model for the experiments. The image data are collected from Google search and NAVER Maps [57], an online mapping service. We have created road environments by pre-processing and reconstructing the collected images. The simulator is implemented within ROS Gazebo [76], and the 3D vehicle model we use is the ROS Prius model [27], which has throttle, brake, and steering control similar to a real vehicle. In the experiment, we show that our agent successfully drives in an unseen environment by learning road graph features. The proposed method outperforms the other methods, which use different features and networks.

3.1 Problem Setting

We consider a driving environment where multiple vehicles are moving on a road and interacting with each other. In the environment, vehicles other than the ego-vehicle, which is controlled by the agent, are controlled by a pre-defined controller. Also, each simulated vehicle follows a pre-defined path. The goal of the agent is to successfully follow the given path and make the ego-vehicle reaching the goal position safely. To follow the traffic flow, the agent must regulate the speed properly and know when and where to stop. This is only possible if the agent is sufficiently trained to recognize the surrounding circumstance.

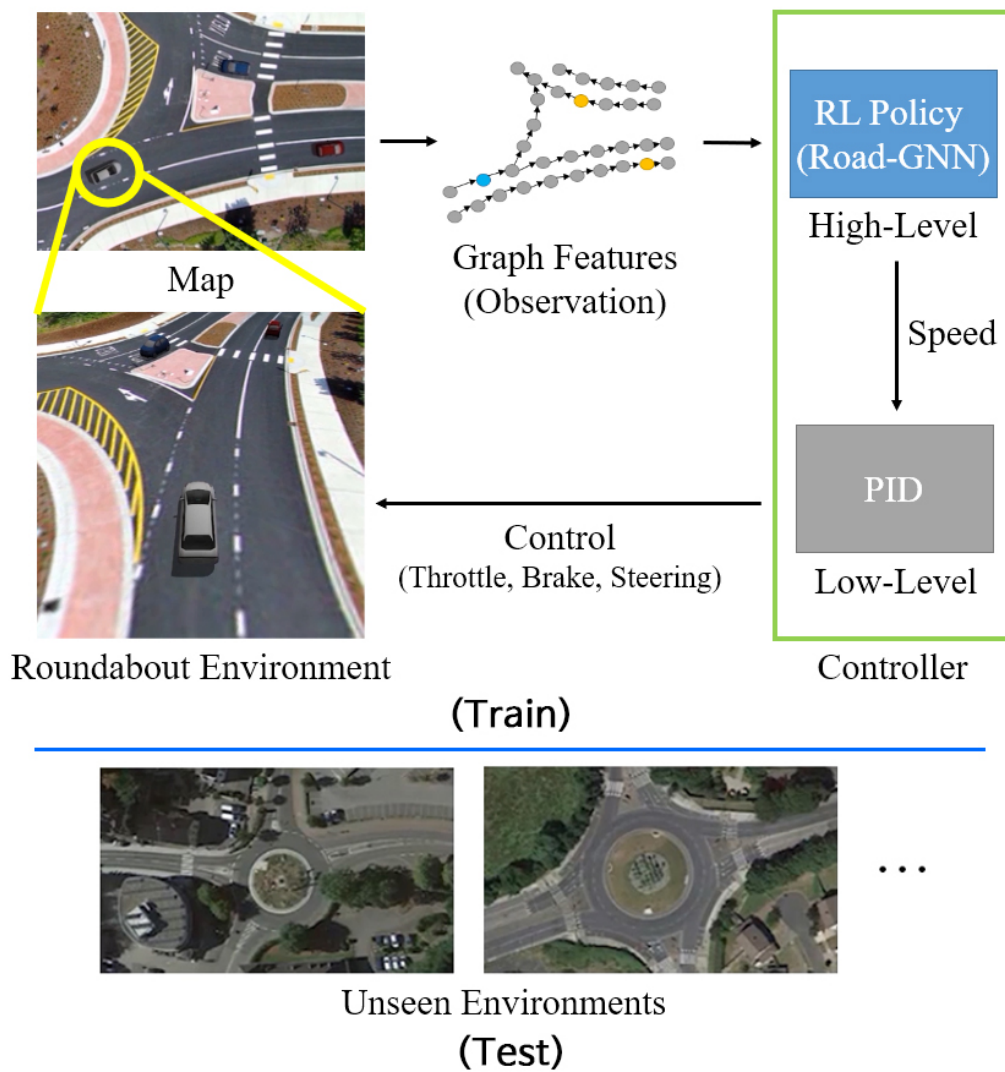


Figure 3.1: Overview of training a Road-GNN based controller and autonomous driving using Road-GNN in unseen environments.

Chapter 3. Road Graphical Neural Networks for Autonomous Driving

We formulate the problem by a Markov decision process (MDP). For every time step t , the policy of the agent π observes the current state s and executes an action a from the discrete action set. Then, the agent obtains a reward r and the next state s' from the environment as a result of the action a . We then optimize the policy π to maximize the expected sum of the rewards with a discount factor γ .

$$\mathbb{E}_\pi[\sum_t \gamma^t r(t)] \quad (3.1)$$

In the remainder of this section, we describe state, action, and reward setup. After that, we also explain how other vehicles move in the environment.

State. In our environment, the state s is represented by using a graph representation. An example of the graph representation is shown in Figure 3.7. The topology of the road can be expressed by both the point-level graph G_p and the segment-level graph G_{seg} . First, the point-level graph G_p contains each point p_i as a node, and each p_i represents a 2D point on the map M . The edge from the point p_i to the point p_j is represented by $e_{i \rightarrow j}$. The edges of G_p are connected along the direction in which vehicles are allowed to move. Second, the segment-level graph G_{seg} contains each segment seg_k as a node, and each segment seg_k represents a set of the points. Depending on the position, each point p_i belongs to one of the segment seg_k in G_{seg} . Similar to G_p , the graph G_{seg} is connected in the direction of the traffic flow.

Each edge $e_{i \rightarrow j}$ and node p_i of the graph G_p has its own features that reflect the vehicle and road conditions. For example, the edge feature of $e_{i \rightarrow j}$ is calculated from the relative position between p_i and p_j .

$$\text{Feature}(e_{i \rightarrow j}) := \text{Pos}(p_j) - \text{Pos}(p_i) \quad (3.2)$$

Here, $\text{Pos}(p_i)$ represents the 2D position vector of p_i on the map M . If the edge $e_{i \rightarrow j}$ does not exist in the graph G_p , the edge feature is considered to be a zero vector. The node feature of p_i is calculated depending on the vehicle conditions

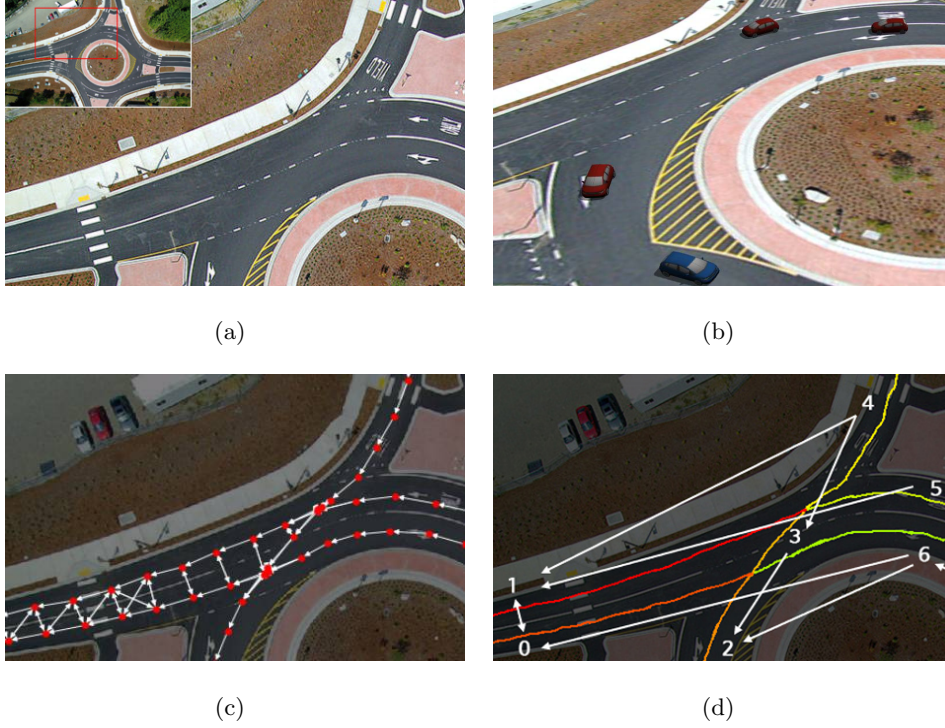


Figure 3.2: An example of a roundabout environment. (a) A part of the original roundabout image data. (b) The 3D simulator used in the experiments. The vehicle model we use is the ROS Prius model [27] whose dynamics is similar to a real vehicle. The roads in the simulator are reconstructed based on the satellite image data. (c) The point-level graph G_p , which is organized from the data. The red dots are the nodes in G_p and the white arrows are the edges connecting the nodes. (d) The segment-level graph G_{seg} . Each segment is colored with different colors and tagged with the numbers. The white arrows are the connection between the segments in G_{seg} .

Chapter 3. Road Graphical Neural Networks for Autonomous Driving

such as position and velocity. Here, we assume that there are a total of N vehicles including the ego-vehicle, and each vehicle is numbered as v_0, \dots, v_{N-1} . For a vehicle v_k , the node p_i has the features of v_k if the node p_i is the nearest point to the vehicle v_k on the graph G_p . The feature includes the position and velocity of v_k , and an occupancy indicator, which is 1 if p_i is the nearest point to the vehicle v_k and 0 otherwise.

$$\text{Feature}(p_i) := [F(v_0), F(v_1), \dots, F(v_{N-1})], \quad (3.3)$$

where

$$F(v_k) = \begin{cases} f(v_k), & \text{if } p_i = \arg \min_{p_j \in G_p} d(v_k, p_j) \\ \mathbf{0}, & \text{otherwise} \end{cases} \quad (3.4)$$

$$f(v_k) = [\text{Pos}_{R_k}(v_k)^T, \text{Vel}_{R_k}(v_k)^T, 1]^T \quad (3.5)$$

Here, $d(v_k, p_j)$ is the L2 distance between the vehicle v_k and the point p_j . $\text{Pos}_{R_k}(v_k)$ and $\text{Vel}_{R_k}(v_k)$ are the position and velocity vector of the vehicle v_k , respectively, which are relatively calculated from the nearest edge to the vehicle v_k . When $e_{nearest_k}$ is the nearest edge to the vehicle v_k , $p_{nearest_k}$ is the starting node of $e_{nearest_k}$, and θ_k is the angle of the direction of $e_{nearest_k}$, then

$$\text{Pos}_{R_k}(v_k) := R_k^{-1} [\text{Pos}(v_k) - \text{Pos}(p_{nearest_k})] \quad (3.6)$$

$$\text{Vel}_{R_k}(v_k) := R_k^{-1} \text{Vel}(v_k), \quad (3.7)$$

where

$$R_k = \begin{bmatrix} \cos \theta_k & -\sin \theta_k \\ \sin \theta_k & \cos \theta_k \end{bmatrix} \quad (3.8)$$

The matrix R_k is the rotation matrix of θ_k .

The agent observes the state s , where the observation is a subset of graph G_p , which includes the K nearest nodes to the ego-vehicle. The subset graph is given

Chapter 3. Road Graphical Neural Networks for Autonomous Driving

to the agent in the form of the adjacency matrix A , the node feature matrix A_N , and the edge feature matrix A_E . The node and edge feature matrix contain each node and edge features as elements. Since there are K nodes in the subset graph and N vehicles on the environment, A , A_E , A_N matrix have the size of $(K \times K)$, $(K \times N \times 5)$, and $(K \times K \times 2)$, respectively.

Action. For every time step, the agent controls the vehicle using a two-level controller. The low-level controller adjusts both the throttle and steering through PID control while the high-level controller determines the target speed of the vehicle. The action is defined as the target speed given to the vehicle. Similarly to other recent works which use an MDP formulation in driving [59, 58], we discretize the actions as the set of target speeds 0m/s, 3m/s, 6m/s, 9m/s, 12m/s.

The low-level controller is designed to follow a pre-determined path. For each episode, the start and goal segments are chosen randomly and the path is set to connect segments. The path is calculated from the entire graph representation and the Dijkstra algorithm. The detailed path finding process is as follows: First, the shortest path in the segment-level graph G_{seg} is calculated using the Dijkstra algorithm. Here, the weights for each edge is considered equal to ones. From the segment-level path, the point-level path is constructed from the points belonging to each segment. For example, when the segment-level path is $\{seg_0, seg_1, \dots, seg_n\}$, then the point-level path is defined as follows.

$$path = seg_0 \cup \dots \cup seg_n \quad (3.9)$$

Note that the segment seg_k is an ordered set of the points p . When $seg_i = \{p_0^i, \dots, p_{l_i-1}^i\}$, $seg_j = \{p_0^j, \dots, p_{l_j-1}^j\}$, and the two segments are connected in the direction of $seg_i \rightarrow seg_j$, then the two sets are combined as follows:

$$path = \{\dots, p_0^i, \dots, p_{l_i-1}^i, p_0^j, \dots, p_{l_j-1}^j, \dots\} \quad (3.10)$$

Here, l_i is the number of points in the segment seg_i . There can be cases where two segments seg_i and seg_j have edges in both directions $seg_i \rightarrow seg_j$ and $seg_j \rightarrow$

Chapter 3. Road Graphical Neural Networks for Autonomous Driving

seg_i . The physical meaning of it is that each segment represents one of the two lanes of a multi-lane road, and a vehicle can change the lanes between them. In these cases, the two segments are combined at a random point when calculating the path. For example, if a point p_{rand}^i is randomly selected from seg_i and the nearest point to p_{rand}^i in seg_j is p_{near}^j , then the two segments are combined as follows:

$$path = \left\{ \cdots, p_0^i, \cdots, p_{rand}^i, p_{near}^j, \cdots, p_{l_j-1}^j, \cdots \right\} \quad (3.11)$$

The low-level PID controller follows the point-level path by controlling both the throttle and steering of the vehicle. First, the steering is controlled to minimize the deviation of the vehicle from the path. The deviation is calculated by the nearest edge e , which connects the two points in the point-level path. Here, we define x_{pos} as the length of the line perpendicular from the vehicle to the nearest edge e . x_{direct} is the difference of the angle between the direction of the vehicle v_k and the nearest edge e . The steering PID controller minimizes the sum of two values: $x_{pos} + x_{direct}$. Second, the throttle is controlled to set the speed of the ego-vehicle to a given target speed. Using PID control, the throttle inputs minimize the difference between the ego-vehicle speed and the target speed. Since the throttle cannot take a negative input, the brake input is taken instead when the speed is higher than the target speed.

For safety, there is also a collision-avoidance system in the controller. First, the future position of each vehicle is estimated using a unicycle vehicle model. The vehicle steps on the brake when a collision is expected. After the vehicle has stopped completely, the system re-estimates the future positions assuming that the vehicle can move at a low speed. If a collision is not expected, then the vehicle moves again based on the agent and PID controller.

Reward. The goal of the agent is to make the vehicle moves from the start area to the goal area. We set a positive reward of +1 when the agent successfully reaches the goal area. The start and goal areas are pre-determined and randomly changed

Chapter 3. Road Graphical Neural Networks for Autonomous Driving

before an episode starts. To prevent the ego-vehicle from being stationary in one place, we also give a small negative reward of -0.01 for each time step before the agent reaches the goal area or exceeds the time limit.

Other Vehicle Movements. There are several other vehicles that move in the simulator. The simulator we use is a closed-loop system, which means that the movements of other vehicles changes according to the movement of the ego-vehicle. We use the same path planner and PID controller used in the ego-vehicle to simulate the other vehicle movements. Each vehicle is set up to follow different paths, which vary from episode to episode. Each controller for the simulated vehicle and the ego-vehicle differs in two ways: First, the simulated vehicles only move at the fixed target speed 9m/s while the ego-vehicle changes the target speed according to input actions. Note that although the target speed is constant, the speeds of the vehicles are not maintained due to the collision-avoidance system. Second, the simulated vehicle is re-spawned in a new random starting position when it reaches the goal area. This re-spawning repeatedly occurs until the ego-vehicle finally reaches its goal area.

3.2 Proposed Method

We now explain our road graphical neural network (Road-GNN), which is used to train the agent. Road-GNN processes a graph observation of the state and calculates the encoding used for the policy and value networks of the agent. Through this networks, we train the RL agent and control the ego-vehicle. Road-GNN takes three steps to encode the road graph: (1) node-and-edge-level encoding, (2) graph-level encoding, and (3) time-level encoding. The overall process of Road-GNN is illustrated in Figure 3.3.

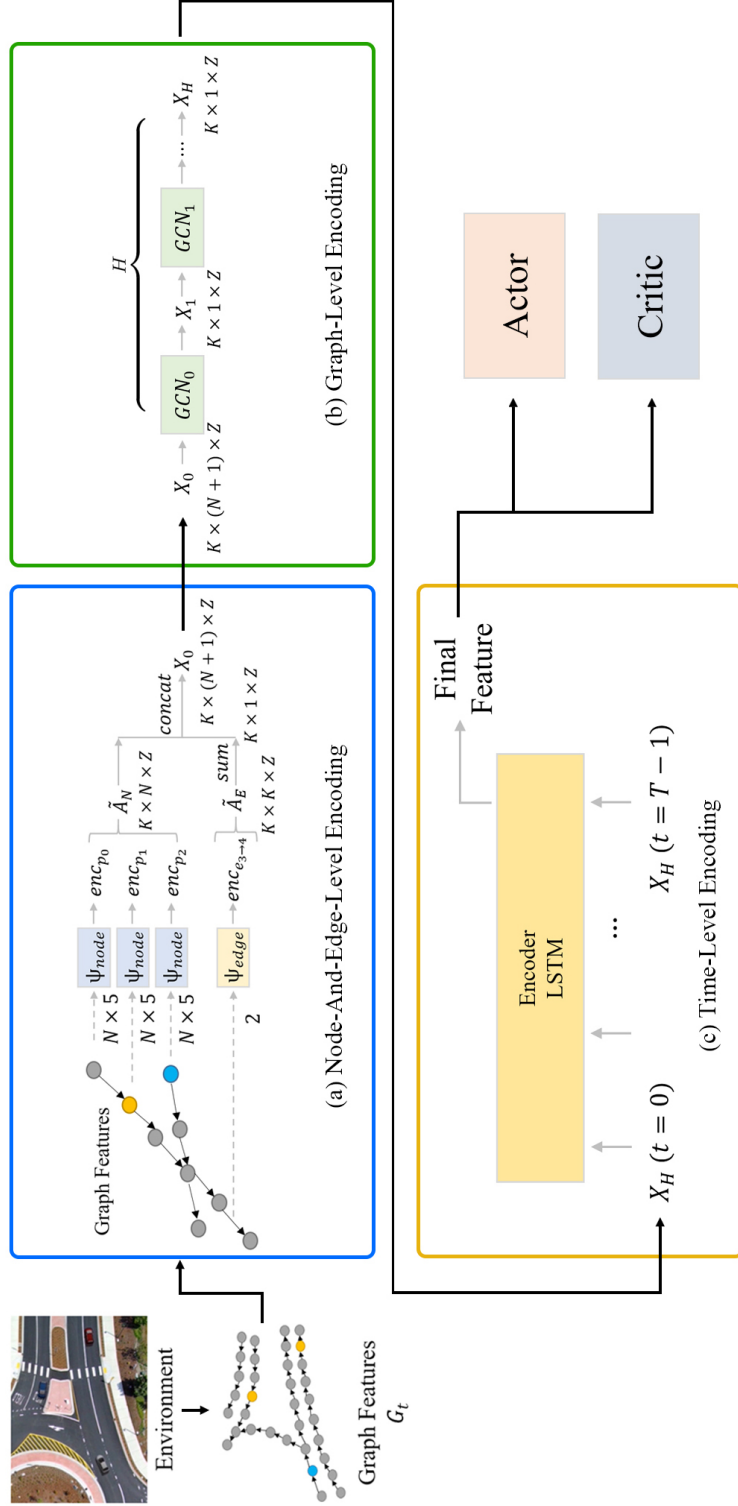


Figure 3.3: The overall process of Road-GNN. (a) Node-and-Edge-Level encoding. (b) Graph-Level encoding. (c) Time-Level encoding.

Chapter 3. Road Graphical Neural Networks for Autonomous Driving

3.2.1 Node-and-Edge-Level Encoding

Since each node and edge feature have different units, we first regularize them using the node and edge encoder networks, ψ_{node} and ψ_{edge} . If the graph observation at time t is G_t , and the number of the nodes in G_t is K , then the networks encoding for each node and edge are as follows:

$$enc_{p_i} = \psi_{node}(\text{Feature}(p_i)), \text{ for } 0 \leq i < K \quad (3.12)$$

$$enc_{e_{i \rightarrow j}} = \psi_{edge}(\text{Feature}(e_{i \rightarrow j})), \text{ for } 0 \leq i, j < K \quad (3.13)$$

In the case of node features, encoding is processed individually for each vehicle features.

$$enc_{p_i} = \psi_{node}(\text{Feature}(p_i)) \quad (3.14)$$

$$= [\psi_v(F(v_0)), \psi_v(F(v_1)), \dots, \psi_v(F(v_{N-1}))] \quad (3.15)$$

For the vehicle feature encoding network ψ_v , we use a simple fully-connected network. Here, the network ψ_v is not shared among all the vehicles v_k . To distinguish the features of the ego-vehicle from the other vehicles, we use ψ_{ego} for the ego vehicle v_{ego} , and ψ_{other} for all the other vehicles. The network ψ_{other} is shared among each vehicle v_k except for v_{ego} . By encoding the elements in the feature matrix A_N and A_E , we can get the encoded feature matrix \tilde{A}_N and \tilde{A}_E , which have the size of $(K \times N \times Z)$ and $(K \times K \times Z)$ respectively, where Z is the size of the encoded vector.

3.2.2 Graph-Level Encoding

After node-and-edge-level encoding, we use a GCN [47] based model for graph-level encoding. First, \tilde{A}_N and \tilde{A}_E are incorporated into a single matrix. The edge encoding matrix \tilde{A}_E is summed up along starting node index, and then

Chapter 3. Road Graphical Neural Networks for Autonomous Driving

concatenated with the node encoding matrix \tilde{A}_N .

$$\text{sum}(\tilde{A}_E) = \left[\sum_i \text{enc}_{e_{i \rightarrow 0}}, \dots, \sum_i \text{enc}_{e_{i \rightarrow K}} \right] \quad (3.16)$$

$$X_0 = \text{concat}(\tilde{A}_N, \text{sum}(\tilde{A}_E)), \quad (3.17)$$

where concat is a concatenation operator. Starting from X_0 , a GCN network ψ_{graph} iteratively calculates the next layer output X_{k+1} from X_k . The layers are stacked H times, and the final output of GCN is X_H .

$$X_{k+1} = \sigma(\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} X_k W_k + B_k), \quad (3.18)$$

where

$$\tilde{A} = A + I \quad (3.19)$$

$$\tilde{D}_{i,j} = \begin{cases} \sum_j \tilde{A}_{i,j}, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases} \quad (3.20)$$

Here, A is the adjacency matrix, σ is a non-linear function, W is a weight, and B is a bias. Only for the first layer GCN_0 , the results of each vehicle channel in X_0 is added into one as like a 3D convolution layer.

$$X_1 = \sigma\left(\sum_{j=0}^N \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} X_0^j W_0^j + B_0^j\right) \quad (3.21)$$

3.2.3 Time-Level Encoding

Each graph observation G_t at time t is encoded by the graph encoding network. In Road-GNN, the network uses the historical memory of G_t with a time length of T . To deal with a historical data, we use a long short-term memory (LSTM) [44] model. First, a readout layer ψ_{read} converts the graph-level encoding of each G_{t-T+1}, \dots, G_t into a series of vectors. Each vector is entered into a LSTM encoding network in the order of time. The final output of the LSTM network becomes the final feature which is used for the policy and value network of the RL agent.

Chapter 3. Road Graphical Neural Networks for Autonomous Driving

3.2.4 Learning Algorithm

We train the agent using proximal policy optimization (PPO) [73], which is a widely used reinforcement algorithm for many recent studies [70, 41]. In our PPO implementation, the agent has the actor and critic networks, and each network shares the common input features, as shown in Figure 3.3.

3.3 Experiments

In the experiment, we examined the generalizability of the proposed method. We have tested the proposed method in various realistic scenarios. The tests were performed on the roundabout environment we implemented.

3.3.1 Environment and Network Details

To implement the roundabout environment, we collected 30 satellite images of roundabouts. As illustrated in Figure 3.4, the collected images have various road structures. Each image is paired with a graph representation constructed from the map structure. We generated the graph representation of a map in the following steps: First, we manually drew the possible trajectories and posed the start and goal area on the map. The nodes of the graph are regularly sampled from the trajectories and grouped into the segments. The grouping process is also performed manually, and we divided the segments at the intersection points. The edges between the nodes and the segments are connected depending on the map structure and the road direction. When the graph is entered into Road-GNN, some additional edges are added to make GNN features flow through the edges. The edge is added if one of the following conditions is satisfied.

- The two nodes are close ($< 2\text{m}$).
- It is possible to change the lane between the two nodes.

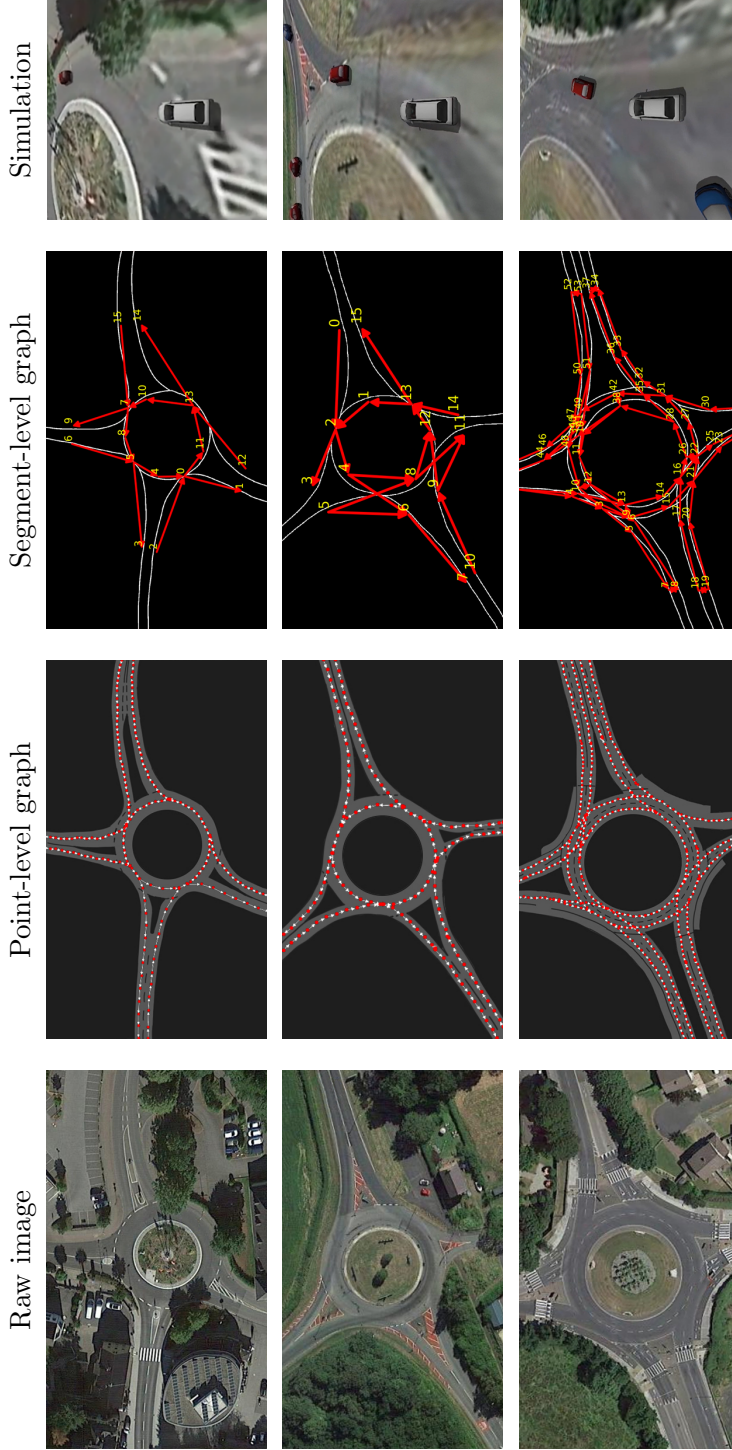


Figure 3.4: Examples of collected map images. The first column shows raw images of roundabouts. The second column shows point-level graphs extracted from raw images. Red dots represent nodes, and white arrows represent edges. The third column shows segment-level graphs. Numbers in yellow represent the index of each segment and red arrows represent the connection between segments. The last column shows how each roundabout appears in a simulator.

Chapter 3. Road Graphical Neural Networks for Autonomous Driving

Table 3.1: Network Architecture Parameters

Network		Value
ψ_{node}, ψ_{edge}		FC 5×8 , FC 2×8
GCN	K, N, Z, H	64, 8, 8, 3
History Length T		10
ψ_{read}		FC 512×64
Actor		FC 64×8 - FC 8×8 - FC 8×5
Critic		FC 64×8 - FC 8×8 - FC 8×1
Non-Linear function		Leaky-ReLU [55]

- The two nodes are occupied with vehicles.

The number of nodes, K , is selected considering the trade-off between the computational cost and the road representability. Empirically, we found that large K requires more computational time, while small K does not fully capture road information. In the environment, the number of vehicles is kept to eight, including the ego-vehicle. The PID controller of each vehicle is executed at 30Hz, while the action and its target speed are updated at 6Hz. For RL training, we used the implementation of [48] but converted it into a discrete version (PPO2) [73]. We implemented the encoder and readout networks, i.e., ψ_{node} , ψ_{edge} , and ψ_{read} , using fully-connected networks. The LSTM encoding network has two stacked hidden states with the size of 64×64 . The details of the network architecture is described in Table 3.1.

3.3.2 Experimental Results

In each experiment, we choose six different maps for training and three different maps for testing. The agent cannot see maps used for testing during training. We have trained the agent with three different random seeds and tested it with 100 episodes for each random seed. Each set of train and test maps is changed

Chapter 3. Road Graphical Neural Networks for Autonomous Driving

depending on the random seed. We use five baseline methods for comparison. Method A and B are not learning-based methods. Method A is a random controller that selects the target speed randomly. Method B is the controller that has the knowledge of the target speed of nearby and controls the ego-vehicle with the same target speed as other vehicles. To show the representative power of Road-GNN, we fix an RL algorithm and compare Road-GNN against other widely used representations and network models. Both MLP and LSTM methods use the history of all the vehicle features such as the velocity and positions, but no traffic information is used. CNN method rasterizes images of the map as described in [41]. The rasterized image channels include the road lines of the maps and the history of the locations of the ego and other vehicles. The network model in the CNN method perceives an area of $32m \times 24m$ around the ego-vehicle with resolution of 320×240 . Figure 3.5 shows learning curves of different algorithms we tested.

In the performance comparison experiment, we have first measured the mean speed and cumulative reward of the ego-vehicle to test if the agent can select a proper speed. As given in Table 3.2, the proposed method shows the highest performance in terms of both the speed and cumulative reward. Meanwhile, the other RL-based method, such as MLP, LSTM, and CNN methods, cannot reach the performance of Road-GNN and failed to drive properly. They were even much slower than not only Road-GNN but also Method B which uses the same controller as other simulated vehicles. The proposed method is the only method that has outperformed Method B in terms of both the speed and the cumulative reward. Therefore, it can be said that the graph representation and Road-GNN is useful for training an RL agent and for driving complicated road environments.

We also checked the number of parameters and flops of each network to test the efficiency of Road-GNN. While the proposed method uses about the same number of parameters as MLP and LSTM, it shows better performance than all compared

Chapter 3. Road Graphical Neural Networks for Autonomous Driving

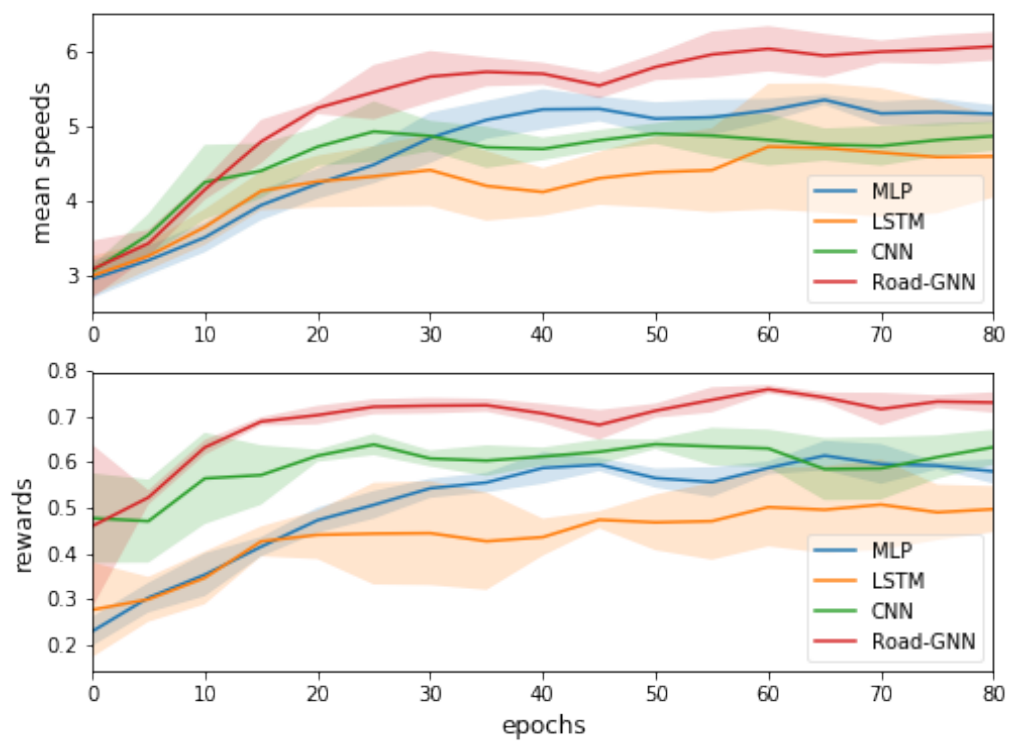


Figure 3.5: Learning curve of RL algorithms. The shaded regions represent the standard deviations of the results.

Chapter 3. Road Graphical Neural Networks for Autonomous Driving

Table 3.2: Performance Comparison

Method	Mean Speeds [m/s]	Rewards	#Param	Flops
Method A	3.035	0.407	-	-
Method B	5.525	0.727	-	-
MLP	5.009	0.582	0.104M	0.208M
LSTM	4.602	0.557	0.070M	1.396M
CNN [41]	4.747	0.641	11.528M	163.380M
Road-GNN	5.838	0.743	0.101M	3.631M
Road-GNN w/o LSTM	5.518	0.710	0.080M	2.369M

methods. However, it should be taken into account that MLP and LSTM methods do not consider road structural information. Compared with CNN [41], which also can consider road structures, Road-GNN shows a better performance than CNN even though it uses fewer parameters and flops.

As an ablation study, we also trained Road-GNN without an encoder LSTM [44] model which is described in Section 3.2.3. We instead encoded time-series features by simple fully-connected layers. In the ablation study, Road-GNN was more effective when it uses an LSTM model. The result shows that LSTM is helpful for Road-GNN to encode time-series features although it requires more parameters.

To show that the proposed method can be generally deployed in various driving environments, we conducted an experiment on eight different maps that have different driving difficulty levels. We tested each method 20 times per map. As a metric of the difficulty, we measured the traffic density of each map, which is defined by the number of vehicles divided by the area of the road. Figure 3.6 shows the results of the difficulty test. The speed of the vehicles tended to

Chapter 3. Road Graphical Neural Networks for Autonomous Driving

Table 3.3: Experiment with aggressive drivers

Method	Collision Rates [%]	Mean Speeds [m/s]	Rewards
Method A	48.3	2.749	0.031
Method B	19.3	5.175	0.488
MLP	18.0	4.838	0.395
LSTM	33.0	4.212	0.225
CNN [41]	22.3	4.547	0.436
Road-GNN	13.3	5.716	0.602

decrease as the traffic density increases. As shown in Figure 3.6, the proposed method shows the highest performance evenly regardless of the driving difficulty of the environment.

We conducted an additional experiment to test the stability of the proposed method. We assume a more hazardous scenario where some drivers ignore the safety rules. We changed the controllers of some simulated vehicles to move more aggressively on the simulator. The target speed of two simulated vehicles is changed to 12m/s, which is faster than the other vehicles. We also made them ignore the collision-avoidance system. The result is shown in Table 3.3. The proposed method shows the lowest collision rate and the highest performance, even though it has been trained in a safe environment. In conclusion, Road-GNN is stable and can drive more safely in hazardous environments.

3.3.3 Qualitative Results

We displayed simulation results to show how the proposed method drives a vehicle in the simulator. Figure 3.8 and Figure 3.9 show snapshots of simulations in a roundabout environment. The roundabout map used for each simulation is shown in Figure 3.7(a) and Figure 3.7(b). The snapshots are arranged in the order

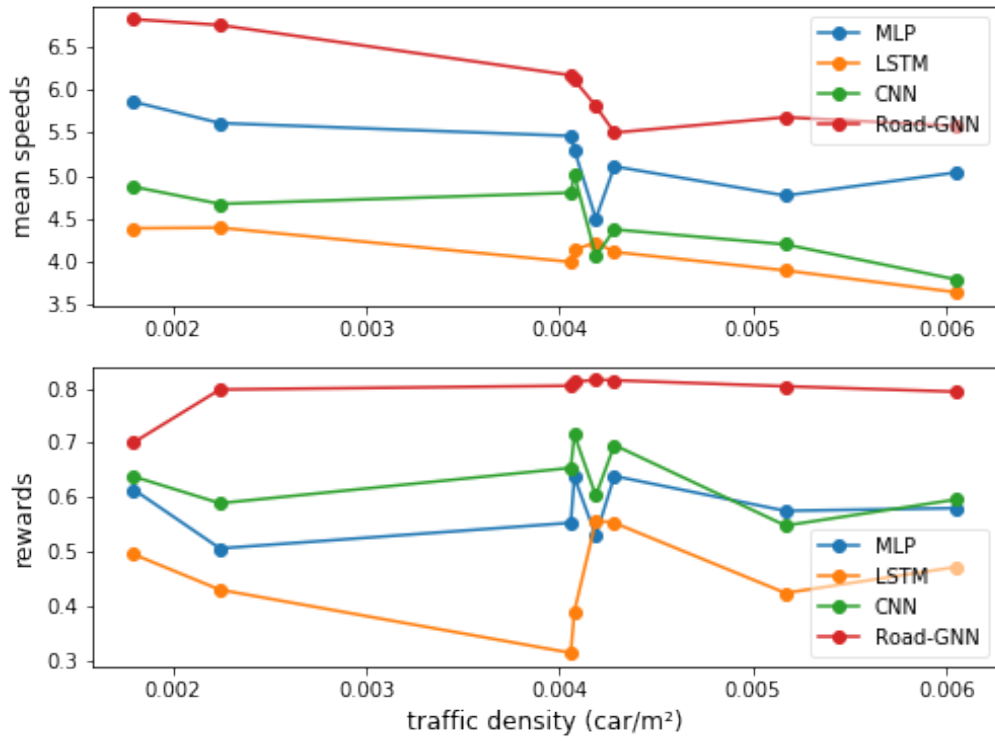


Figure 3.6: Experiment with various difficulty levels. Each dot represents a test result in one map. Each method is tested 20 times per map.

Chapter 3. Road Graphical Neural Networks for Autonomous Driving

of time with regular time intervals. In the simulations, the white ego-vehicle is controlled by a fully-trained Road-GNN. In Figure 3.8, the white ego-vehicle enters to a roundabout, moves along a curved path, and successfully exits from the roundabout. Because there is no interruption by other vehicles, the white-ego vehicle drives at a relatively constant speed between 5.849m/s and 7.523m/s. In Figure 3.9, the white ego-vehicle merges into a roundabout while there are already two simulated vehicles turning around the roundabout. During the merge, the white ego-vehicle slows down until the red vehicle passes completely. The white ego-vehicle recovers its speed while moving between the red and blue vehicle. The results show that the proposed method successfully trains an RL agent and makes it to drive in a complex road environment.

We also displayed simulation results which compares the proposed method with CNN [41]. Figure 3.10 and Figure 3.11 compares Road-GNN and CNN by showing snapshots of simulations. The roundabout map used for each simulation is shown in Figure 3.7(c) and Figure 3.7(d). For the comparison, we used the same low-level controller but changed the types of networks and their inputs. In Figure 3.10, Road-GNN selects a proper speed while merging the lanes. Road-GNN increases the vehicle speed up to 6.300m/s after the merge. On the other hand, CNN model requires relatively more time to merge and follow the lanes. In specifically, CNN shows slower speeds than Road-GNN in all time steps. In Figure 3.11, the simulation shows a similar result. Road-GNN follows the vehicle in front of it while maintaining its speed between 6.152m/s and 6.869m/s. On the other hand, CNN model does not accelerate even though there is no vehicle in front of it. The results show that Road-GNN can optimize the speed appropriately while CNN model can not.

3.4 Chapter Summary

In this chapter, we have proposed a new autonomous driving framework, which can be generalized to various road environments, using the GNN based structure named Road-GNN and an RL based controller. The proposed network, Road-GNN, is designed to perceive a graph representation of a road, which includes the road connection and vehicle information. In the experiments, the proposed method has outperformed the other methods which do not use the graph representation. The results have shown that the proposed method can generalize different road structures and the knowledge learned from roads in the training set can be easily transferred to unseen road structures.

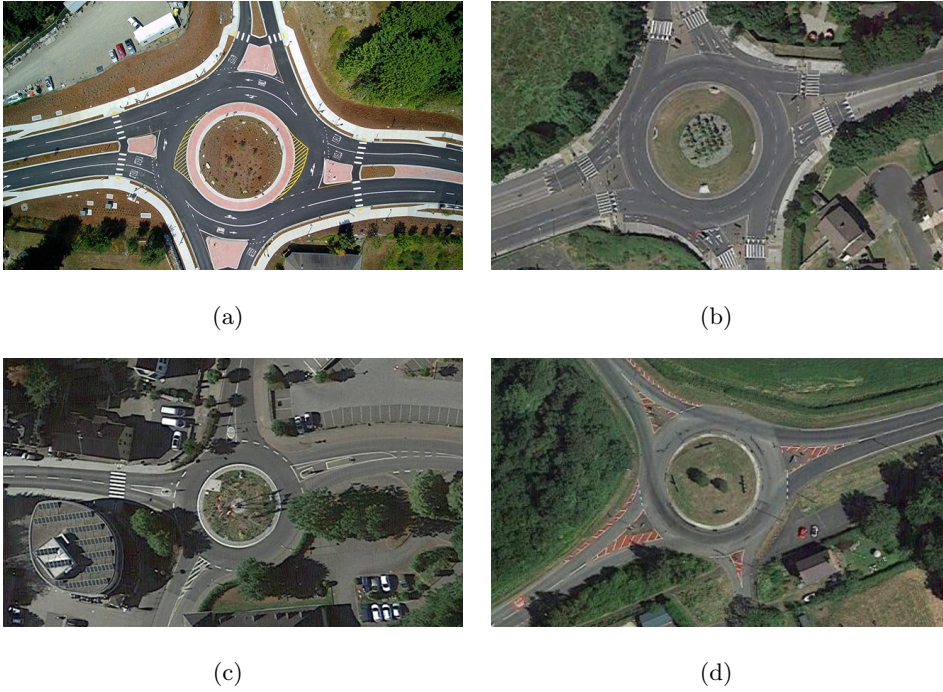


Figure 3.7: Roundabout environments used for showing qualitative results. (a) The roundabout map for the simulation in Figure 3.8. (b) The roundabout map for the simulation in Figure 3.9. (c) The roundabout map for the simulation in Figure 3.10. (d) The roundabout map for the simulation in Figure 3.11.


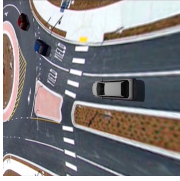
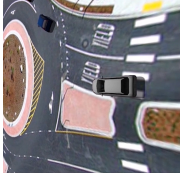
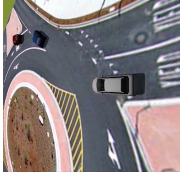

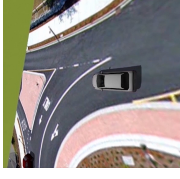
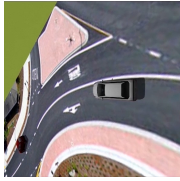

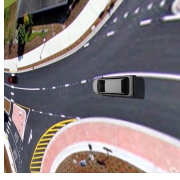
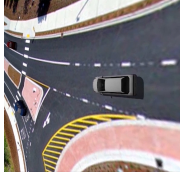


Time	$t = 2.0s$	$t = 4.0s$	$t = 6.0s$	$t = 8.0s$	$t = 10.0s$	$t = 12.0s$
Speed	6.296m/s	6.627m/s	7.176m/s	7.523m/s	7.221m/s	6.233m/s
Road-GNN (Ours)						
Time	$t = 14.0s$	$t = 16.0s$	$t = 18.0s$	$t = 20.0s$	$t = 22.0s$	$t = 24.0s$
Speed	6.450m/s	6.805m/s	7.233m/s	6.883m/s	5.849m/s	6.177m/s
Road-GNN (Ours)						

Figure 3.8: Snapshots of a simulation result of Road-GNN.


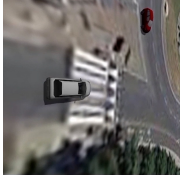
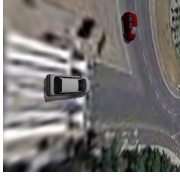
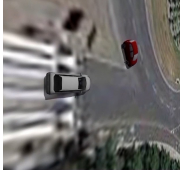
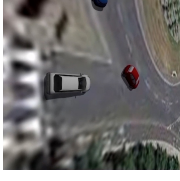
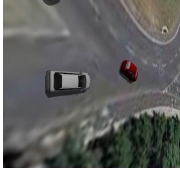

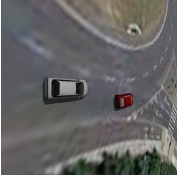
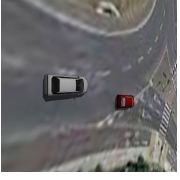


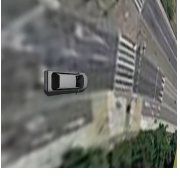
Time	$t = 2.0s$	$t = 4.0s$	$t = 6.0s$	$t = 8.0s$	$t = 10.0s$	$t = 12.0s$
Speed	6.298m/s	6.780m/s	4.557m/s	1.410m/s	3.611m/s	5.474m/s
Road-GNN (Ours)						
Time	$t = 14.0s$	$t = 16.0s$	$t = 18.0s$	$t = 20.0s$	$t = 22.0s$	$t = 24.0s$
Speed	6.300m/s	6.738m/s	6.194m/s	5.456m/s	5.899m/s	6.246m/s
Road-GNN (Ours)						

Figure 3.9: Snapshots of a simulation result of Road-GNN.



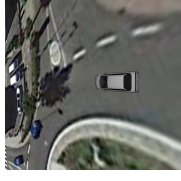
Time	$t = 6.0s$	$t = 12.0s$	$t = 18.0s$	$t = 24.0s$	$t = 30.0s$	$t = 36.0s$
Speed	3.583m/s	2.605m/s	3.277m/s	4.962m/s	6.000m/s	6.300m/s
Road-GNN (Ours)						
Speed	1.143m/s	2.137m/s	1.882m/s	3.349m/s	4.637m/s	5.892m/s
CNN [41]						

Figure 3.10: Snapshots of a simulation result for comparison.

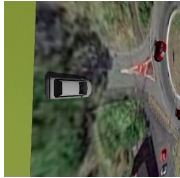

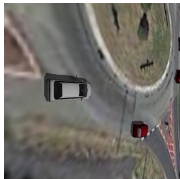

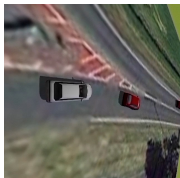


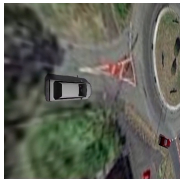
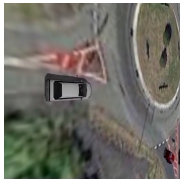
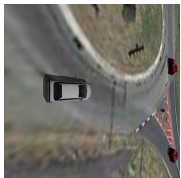
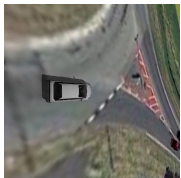
Time	$t = 4.0s$	$t = 8.0s$	$t = 12.0s$	$t = 16.0s$	$t = 20.0s$	$t = 24.0s$
Speed	1.649m/s	4.277m/s	6.869m/s	6.152m/s	6.180m/s	6.420m/s
Road-GNN (Ours)						
Speed	1.618m/s	1.749m/s	1.934m/s	4.268m/s	4.908m/s	5.827m/s
CNN [41]						

Figure 3.11: Snapshots of a simulation result for comparison.

Chapter 4

Road Graph and Image Attention Network for Autonomous Driving

In recent years, vision-based object recognition algorithms have shown a dramatic improvement in the field of autonomous driving. For example, pedestrian detection [35], lane detection [53], 3D object detection [89], and semantic segmentation [86] have been developed and shown remarkable performance. However, applying such vision-based methods to vehicle control is a challenging task. In most cases, it is required to elaborately design a handcraft rule-based controller to use the vision-based features.

There have been studies which applied vision-based features in a learning-based controller. Previous methods have used image-based features [25, 20, 11], semantic segmentation based features [75], or detection based features [3] to train an imitation learning based autonomous driving controller. However, these works have shown insufficient performance compared to commercialized handcraft controllers.

Chapter 4. Road Graph and Image Attention Network for Autonomous Driving

One of the most critical problems when applying the vision-based approaches to vehicle control is that the controller does not know which part of the image is more important. Each object in the image has different importance according to the scene context. For example, when a driver sees traffic lights at an intersection, not all traffic lights are equally important. The driver should focus more on the traffic lights at the front than the traffic lights at the crossroad. Likewise, drivers usually pay more attention to pedestrians jaywalking on the road than pedestrians walking on the sidewalk.

To compensate for such a problem, several works investigated the importance of image features along with the LiDAR sensor data. Zhao et al. [88] proposed a LiDAR and image fusion based 3D semantic segmentation method which can be used to identify the scene context of objects. In addition, Prakash et al. [62] used the attention mechanism [78] to capture the importance of the image feature according to the LiDAR sensor data. However, LiDAR-based attention is not enough to fully understand the scene context for two reasons. (1) It is difficult to directly extract meaningful information such as the road direction and connection because the LiDAR data is a high-dimensional representation. (2) The LiDAR data lack prior knowledge about the traffic (e.g., right-hand or left-hand traffic), which is important for determining whether an object is moving in a safe direction. As studied in previous works [29, 52], the vehicle movement is highly influenced by the direction of the road. Therefore, a new type of state representation, which can reflect the road structural context of the scene, is required to overcome the limitation of LiDAR-based attention.

In this chapter, we propose an autonomous driving framework named *road graph and image attention network* (RIANet), which considers the importance of object features according to the road structure. Figure 4.1 shows a brief explanation of our idea. To reflect the graph information of the road, we represent the road structure in the form of a graph which is called *road graph*. Unlike other

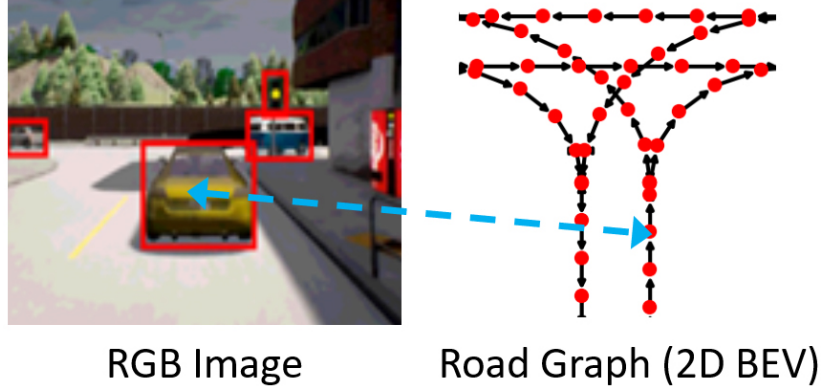


Figure 4.1: Illustration of our main idea. (Left) RGB camera image of the scene. (Right) The corresponding road graph seen from the bird's eye view (BEV). We propose to use the attention mechanism to consider the importance of image features according to the road graph features.

attention-based controllers, the proposed method leverages the road graphical features along with visual features extracted from image and LiDAR sensors. The attention mechanism allows the network to compute the scene context and prioritize the importance of objects according to the road structure.

We use imitation learning to train the network. The evaluation and data collection are conducted on a 3D urban scene driving simulator named CARLA [25]. The agent in the CARLA simulator is required to deal with dangerous situations such as lane changing, unexpected obstacle avoidance, crossing intersection, and unprotected turn. The experiment shows that the proposed method outperforms the baseline methods in terms of all the suggested metrics. Our contributions are summarized as follows:

- We propose a novel autonomous driving framework which considers the importance of objects according to the road structure.
- We demonstrate that the road graph features can effectively reflect the road

Chapter 4. Road Graph and Image Attention Network for Autonomous Driving

structural context of the scene

- We experimentally show that the driving performance can be improved by fusing the road graph and the other sensor data features

4.1 Problem Setting

We first clarify our problem settings before explaining the proposed framework. We consider an urban scene driving environment. The goal of the agent is to navigate a given route while following traffic rules. The route is composed of multiple goal locations in the road environment. At each time step t , the agent is given a high-dimensional observation o_t which consists of the following components.

Road Graph. The agent is given the topology information of the drivable road in the form of a graph. A road graph $G_{global} = (V, E)$ consists of nodes V and edges E . A node $n_i \in V$ represents a point on the road and is distributed along the centerline of the road segment. An edge $e_{i \rightarrow j} \in E$ connects the node n_i and the node n_j . The direction of $e_{i \rightarrow j}$ represents the direction of the road segment. We use the method in Chapter 3 to extract a road graph G_{global} from a road. We sample nodes on the road at intervals of 3m. We connect edges between nodes depending on the direction and connection of the roads. In addition, we connect two nodes if it is possible for a driver to change the lane along the edge. A node n_i contains a node feature f_{n_i} and an edge $e_{i \rightarrow j}$ contains an edge feature $f_{e_{i \rightarrow j}}$, respectively. In simulation and training, the agent only observes a subgraph G_t , which consists of the nearest nodes to the ego-vehicle. The detailed explanations about a road graph and subgraphs are described in Section 4.3.2.

Camera Image. The agent is given a front ego-view camera image which has a resolution of 400×300 with a 100° FOV. To remove radial distortion [62], we crop the image to 256×256 before entering the image into the feature encoder.

Additional Sensor Data. The agent is given LiDAR, GPS, IMU, and speedome-

Chapter 4. Road Graph and Image Attention Network for Autonomous Driving

ter sensor data. The LiDAR point cloud is pre-converted into a 2D BEV grid image by the method in [62, 67]. A 2D BEV grid image contains 256×256 pixels and covers a $32\text{m} \times 32\text{m}$ area in front of the ego-vehicle. A 2D BEV grid image has two channels: The first channel represents the points above the ground plane while the second channel represents the points below the ground plane. There are GPS, IMU, and speedometer sensor data as well. These sensor give the position and speed data of the ego-vehicle to the agent. The position and direction of the ego-vehicle are localized from the GPS and IMU sensor inputs. We use the extended Kalman filter for localization.

4.2 Proposed Method

In this section, we explain the road graph and image attention network (RIANet), which leverages the attention score [78] between a road graph and image features. The proposed framework is divided into three modules: (1) a feature encoder, (2) an attention network, and (3) a low-level controller. The feature encoder module encodes a road graph, a camera image, and additional sensor data into features. The attention network module takes these feature embeddings as inputs, applies the attention mechanism to fuse features, and extracts the context feature of the scene. The low-level controller module calculates the target speed from the context feature and controls the ego-vehicle using a PID controller.

4.2.1 Feature Encoder

The feature encoder module takes an observation $o_t = \{G_t, I_t, S_t\}$ as an input and encodes the road graph G_t , the camera image I_t , and the additional sensor data S_t into feature embeddings. The encoder for each component is a neural network and we use a different network architecture for each encoder. The entire network structure of the feature encoder module is shown in Figure 4.2.

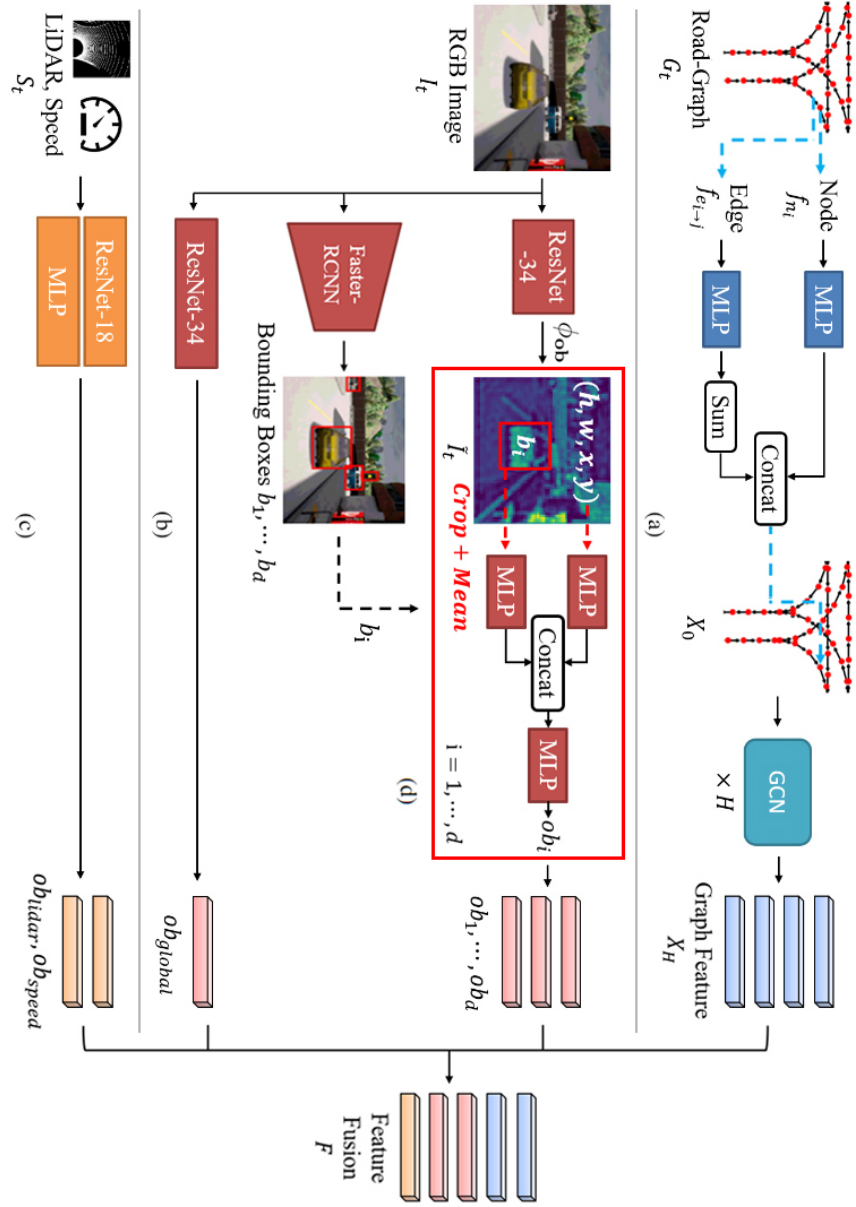


Figure 4.2: The structure of the feature encoder module. (a) Road graph encoder. (b) Camera image encoder. (c) Additional feature encoder. (d) The encoding process of an object feature.

Chapter 4. Road Graph and Image Attention Network for Autonomous Driving

Road Graph Encoder. The road graph encoder extracts a feature embedding of the road graph G_t using graph convolutional network (GCN) [47]. Similar to the graph encoding method in Section 3.2.1, we first regularize each node feature f_{n_i} and edge feature $f_{e_{i \rightarrow j}}$ using fully-connected networks ψ_{node} and ψ_{edge} . We then sum up the edge feature along the starting node index and concatenate it with the node feature. If the number of nodes in G_t is N , the encoding process is formulated as follows:

$$\tilde{f}_{n_i} = \psi_{node}(f_{n_i}), \text{ for } 1 \leq i \leq N \quad (4.1)$$

$$\tilde{f}_{e_{i \rightarrow j}} = \psi_{edge}(f_{e_{i \rightarrow j}}), \text{ for } 1 \leq i, j \leq N \quad (4.2)$$

$$\tilde{f}_i = \text{concat}(\tilde{f}_{n_i}, \sum_k \tilde{f}_{e_{k \rightarrow i}}), \text{ for } 1 \leq i \leq N \quad (4.3)$$

Here, concat is a concatenation operator. We get the embedded feature matrix $X_0 \in \mathbb{R}^{N \times Z}$ by applying the encoding process to each element in G_t . The i -th row of the X_0 is the feature $\tilde{f}_i \in \mathbb{R}^Z$, where Z is the feature size. We update the feature matrix X_0 using a H -layers GCN network [47]. The update process of the k -th layer of GCN is formulated as follows:

$$X_k = \sigma(\tilde{A}X_{k-1}W_k + B_k), \quad (4.4)$$

where $\tilde{A} \in \mathbb{R}^{N \times N}$ is a normalized adjacency matrix of G_t with self-loops, σ is a leaky-ReLU [55], and W_k and B_k are the weight and bias of the k -th GCN layer. We iteratively update X_0 and obtains the final graph feature embedding X_H .

Camera Image Encoder. We expect the camera image encoder to capture some important visual information such as traffic lights, pedestrians, and obstacles. To capture and encode the object features, we use a ResNet [37] based feature map and a Faster R-CNN [66]. The encoding process of an object feature is shown in Figure 4.2(d). We first extract the feature map of the image I_t using the ResNet based network. The ResNet based feature map $\phi_{ob}(I_t) = \tilde{I}_t$ has the channel size of C and the same height and width as the input image I_t . We then detect objects in

Chapter 4. Road Graph and Image Attention Network for Autonomous Driving

the image I_t using Faster R-CNN and obtain a bounding box set $\{b_i\}_{i=1}^d$, where b_i is a bounding box and d is the number of the detected bounding boxes. d is not a fixed value and can be changed according to the image I_t . Bounding box b_i has the corresponding classification feature c_i and positional feature p_i . The classification feature c_i is a classification result of the i -th object and encoded in a one-hot vector. The positional feature $p_i = (h, w, x, y)$ represents the positional information of a bonding box b_i , where h , w , x , and y represent the height, the width, the x and y position of the center of the bounding box b_i in the 2D pixel space. For each bounding box b_i , we crop the feature map \tilde{I}_t according to the size and location of the bounding box b_i . We get the spatial mean of the cropped feature map and incorporate it with c_i and p_i using MLP networks. For a cropped feature map with the size of $C \times h \times w$, the spatial mean has the size of C . The feature embedding for the i -th object is denoted as ob_i . The object feature encoding process is formulated as follows:

$$\tilde{I}_t = \phi_{ob}(I_t) \quad (4.5)$$

$$f_{img} = \psi_{img}(\text{mean}(\text{crop}(\tilde{I}_t; b_i))) \quad (4.6)$$

$$f_{lin} = \psi_{lin}(\text{concat}(c_i, p_i)) \quad (4.7)$$

$$ob_i = \psi_{ob}(\text{concat}(f_{img}, f_{lin})), \quad (4.8)$$

where ψ_{img} , ψ_{lin} , and ψ_{ob} are MLP networks, crop is a crop operator, and mean is a spatial mean operator.

In addition to each object feature embedding ob_i , the camera image encoder also encodes the global image feature embedding ob_{global} to capture the global feature of the image I_t . The image I_t is encoded into ob_{global} by the ResNet-based network ϕ_{global} as follows:

$$ob_{global} = \phi_{global}(I_t), \quad (4.9)$$

where ob_i and ob_{global} are one-dimensional vectors and have the same feature size

of Z .

Additional Feature Encoder. The additional feature encoder take the additional sensor data S_t as inputs. The S_t contains LiDAR data f_{lidar} and speed data f_{speed} . We encode each f_{lidar} and f_{speed} into feature embedding. First, LiDAR data is encoded by a ResNet model ϕ_{lidar} . As explained in Section 4.1, the LiDAR data f_{lidar} is a pre-processed 2D BEV grid image. The LiDAR data f_{lidar} is encoded into a feature ob_{lidar} by ϕ_{lidar} . The speed data f_{speed} is also encoded into a feature ob_{speed} by the MLP network ψ_{speed} . The encoding process of the additional feature encoder is formulated as follows:

$$ob_{lidar} = \phi_{lidar}(f_{lidar}) \quad (4.10)$$

$$ob_{speed} = \psi_{speed}(f_{speed}) \quad (4.11)$$

4.2.2 Attention Network

The attention network module outputs the scene context by incorporating all the encoded features from the feature encoder. To leverage the attention between each feature, we use the transformer model [78] and compute the attention scores between the features. The network architecture of the attention network module is shown in Figure 4.3(a). We first integrate all the feature embeddings in Section 4.2.1 into the feature fusion F as follows:

$$F_{ob} = [ob_1, \dots, ob_d, ob_{global}, ob_{lidar}, ob_{speed}] \quad (4.12)$$

$$F = \text{concat}(X_H, F_{ob}) \quad (4.13)$$

The feature F_{ob} contains the object feature embeddings $ob_1, \dots, ob_d \in \mathbb{R}^Z$, the global image feature embedding $ob_{global} \in \mathbb{R}^Z$, the LiDAR feature embedding $ob_{lidar} \in \mathbb{R}^Z$, and the speed embedding $ob_{speed} \in \mathbb{R}^Z$. Each feature has the same size of Z and these features are stacked into the feature F_{ob} . The feature $F_{ob} \in \mathbb{R}^{(d+3) \times Z}$ and the road graph feature $X_H \in \mathbb{R}^{N \times Z}$ are concatenated into the feature fusion $F \in \mathbb{R}^{(N+d+3) \times Z}$.

Chapter 4. Road Graph and Image Attention Network for Autonomous Driving

We use the attention mechanism [78] to obtain the attention scores between each feature in F . The attention function $\text{Attn}(\cdot, \cdot)$ is formulated as follows:

$$\text{Attn}(M_1, M_2) = \text{concat}(\text{head}_1, \dots, \text{head}_n)W^O, \quad (4.14)$$

$$\text{where } \text{head}_i = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{Z}}\right)V_i \quad (4.15)$$

$$Q_i, K_i, V_i = M_1 W_i^Q, M_2 W_i^K, M_2 W_i^V \quad (4.16)$$

Here, M_1 and M_2 are arbitrary inputs to the attention function, softmax is the softmax function, n is the number of heads, $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{Z \times Z}$, and $W_O \in \mathbb{R}^{(nZ) \times Z}$ are weight matrices. The transformer consists of an attention function, a MLP network ψ_{tran} , and a layer normalization LN. The transformer is formulated as follows:

$$\text{Transformer}(M_1, M_2) = \text{LN}(\psi_{tran}(\text{sub}) + \text{sub}), \quad (4.17)$$

$$\text{where } \text{sub} = \text{LN}(\text{Attn}(M_1, M_2) + M_1) \quad (4.18)$$

The transformers are stacked L times and compose a total of L transformer layers. Except for the last layer, the process of each layer is formulated as follows:

$$F_k = \text{Transformer}_k(F_{k-1}, F_{k-1}), \text{ for } 1 \leq k < L \quad (4.19)$$

Starting from $F_0 = F$, the fusion transformer module iteratively computes F_k . The last transformer layer computes the scene context feature c from $F_{L-1} \in \mathbb{R}^{(N+d+3) \times Z}$ and $\bar{F}_{L-1} \in \mathbb{R}^{1 \times Z}$ as follows:

$$c = \text{Transformer}_L(\bar{F}_{L-1}, F_{L-1}), \quad (4.20)$$

where \bar{F}_{L-1} is the feature mean of F_{L-1} . The size of the scene context feature $c \in \mathbb{R}^{1 \times Z}$ is invariant to the number of nodes N and the number of object features d .

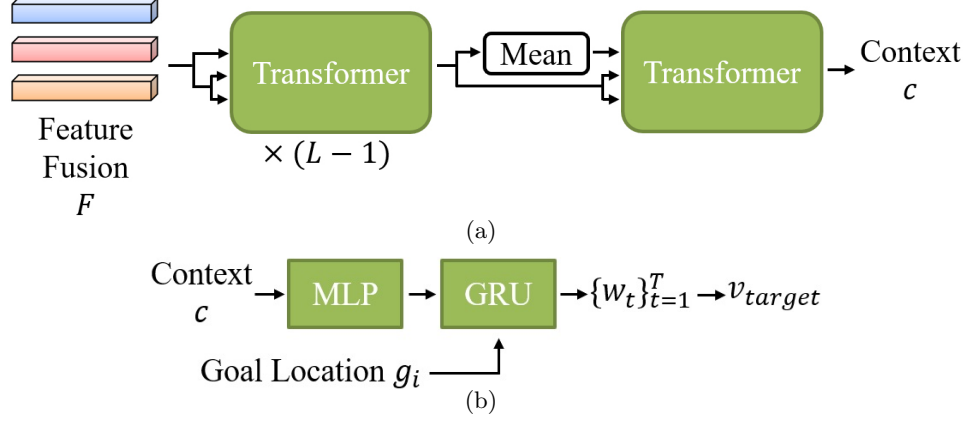


Figure 4.3: The structure of the networks. (a) The attention network module. (b) Waypoint prediction network.

4.2.3 Low-Level Controller

The low-level controller takes the scene context c as an input and computes the steering control value and the target speed for the ego-vehicle.

Steering. We compute the steering value which makes the ego-vehicle follow the pre-defined path ζ . The path ζ is calculated through the following process. Similar to [62], we assume that the goal is to navigate a given route $\gamma = \{g_i\}_{i=1}^l$, where g_i is a goal location and l is the number of the goal locations. We first find a node $n_{g_i}^{near} \in G_{global}$ which is the nearest node to g_i . We then use the Dijkstra algorithm and calculate the shortest path in G_{global} , which visits all the nodes $\{n_{g_i}^{near}\}_{i=1}^l$. The path ζ is constructed after applying a line smoothing method to the Dijkstra shortest path. The steer PID controller computes the deviation between the current ego-vehicle position and the path ζ and finds the steering value which makes the ego-vehicle follow the path ζ .

Target Speed. We also compute the target speed of the ego-vehicle from the future waypoints which is predicted from a waypoint prediction network [11, 62]. The network architecture of the waypoint prediction network is shown in Figure 4.3(b). As demonstrated in [11, 62], we also empirically found that it is

Chapter 4. Road Graph and Image Attention Network for Autonomous Driving

better to use the waypoint prediction network than to compute the target speed directly. The waypoint prediction network takes the scene context c as an input and predicts the waypoints in the ego-vehicle coordinate frame. The scene context c is first passed to the MLP network. The output of the MLP network is entered to the GRU [16] as a hidden state. Starting from the initial input of $w_0 = (0, 0)$, the GRU iteratively outputs the differential waypoints δw_t . During the iteration, the GRU incorporates the current goal location g_i in its inputs. The waypoints w_t are constructed as $w_t = \sum_{\tau=1}^t \delta w_\tau$ for the future time steps $t = 1, \dots, T$ and we use $T = 4$. After the waypoints are predicted, we compute the target speed $v_{target} = \|w_1 - w_0\|_2 / \delta t$, where δt is the time interval between the waypoints. We use a PID controller which has the same configuration as [11, 62] and provides the throttle and brake values to the ego-vehicle, which make the ego-vehicle have the target speed v_{target} .

4.2.4 Learning Algorithm

We train our network using imitation learning [11, 62]. The output of the waypoint prediction network is the waypoints $\{w_t\}_{t=1}^T$. We use L_1 loss between the predicted waypoints w_t and the expert waypoints w_{gt} . Given the expert waypoints $\{w_t^{gt}\}_{t=1}^T$, the loss function is formulated as follows:

$$loss = \sum_{t=1}^T \|w_t - w_t^{gt}\|_1 \quad (4.21)$$

As mentioned in [62], the waypoints w_t and w_{gt} are different from the goal locations g_i of the given route γ or the nodes in the pre-defined path ζ which the ego-vehicle follows.

4.3 Experiments

In experiment, we use a widely-used driving simulator named CARLA [25]. We train the proposed network with an expert dataset and compare the evaluation

results with the baseline methods. All data collection and experiments are conducted in the CARLA environment.

4.3.1 Experimental Settings

We use the CARLA 0.9.10 version for the experiments. There are eight types of maps in CARLA, and in each environment, the agent needs to complete the given route while handling challenging situations such as lane changing, unexpected obstacle avoidance, crossing intersection, and unprotected turn. We follow the evaluation settings in [62] and use Town05 for evaluation. We use two different types of scenarios: Town05 Short and Town05 Long. The scenarios in Town05 Short setting have 10 routes of 100-500m in length and the scenarios in Town05 Long setting have 10 routes of 1000-2000m in length.

4.3.2 Dataset

The training data is collected by a handcraft expert policy which uses privileged information about the environment. The dataset is provided by [62] and we especially use the ClearNoon dataset. In our setting, we do not use the data from Town10 for training because CARLA 0.9.10 does not provide the HD-map of Town10, which is required for constructing a road graph. We instead use Town01, Town02, Town03, Town04, Town06, and Town07.

The road graph data is not contained in the dataset of [62]. Therefore, we generate the road graph feature G_t according to the vehicle position and rotation in the dataset. We first construct the global graph G_{global} using HD-map data of each town. HD-map data in CARLA uses the OpenDRIVE format [1] and provides border information for each lane of roads. We obtain the center trajectory of each lane through borderline information and sample nodes from the center trajectory at intervals of 3m, including the endpoint of each center trajectory. HD-map also provides linkage information of lanes. We connect edges

Chapter 4. Road Graph and Image Attention Network for Autonomous Driving

so that nodes in the same lane can be sequentially linked. In addition, edges are connected between end nodes of linked lanes. Inspired by the method in Chapter 3, we define node and edge features as follows:

$$f_{n_i} = (x_{n_i}, y_{n_i}, \mathbb{1}_{ts}, \mathbb{1}_{tr}, \mathbb{1}_{oc}, \mathbb{1}_{oc} \cdot v) \quad (4.22)$$

$$f_{e_{i \rightarrow j}} = (x_{e_{i \rightarrow j}}, y_{e_{i \rightarrow j}}), \quad (4.23)$$

where (x_{n_i}, y_{n_i}) is the 2D position of the node n_i , $(x_{e_{i \rightarrow j}}, y_{e_{i \rightarrow j}})$ is the 2D direction of the edge $e_{i \rightarrow j}$, v is the speed of the ego-vehicle, and $\mathbb{1}_{ts}$, $\mathbb{1}_{tr}$, $\mathbb{1}_{oc}$ are the indicators. $\mathbb{1}_{ts}$ is the traffic signal indicator which is 1 if the node is on an intersection and 0 otherwise. $\mathbb{1}_{tr}$ is the path indicator which is 1 if the node is included in the pre-defined path ζ . $\mathbb{1}_{oc}$ is the occupancy indicator which is 1 if the node is the nearest node to the current ego-vehicle position. The node position (x_{n_i}, y_{n_i}) and edge direction $(x_{e_{i \rightarrow j}}, y_{e_{i \rightarrow j}})$ are computed in the ego-vehicle coordinate frame, where x -axis is parallel to the ego-vehicle direction. The position of the ego-vehicle can be inversely calculated from the position of each node and the occupancy indicator.

After constructing G_{global} , we extract the graph presentation G_t . We first select the K nearest nodes to the ego-vehicle position. We then select only the nodes in front of the ego-vehicle with a small margin μ . We use $K = 96$ and $\mu = 10\text{m}$ in our setting. The position and direction of the ego-vehicle are estimated by the extended Kalman filter and used to compute the ego-vehicle coordinate frame. The road graph is computed in real-time when evaluating the agent.

4.3.3 Implementation Details

In the road graph encoder, the MLP networks ψ_{node} and ψ_{edge} are constructed by a fully-connected layer with the size of 6×32 and 2×32 , respectively. We use $H = 3$ for the number of GCN [47] layers. The weight of each GCN layer has the size of 64×64 except for the last layer and the weight of the last GCN layer has

Chapter 4. Road Graph and Image Attention Network for Autonomous Driving

the size of 64×128 .

We use a pre-trained ResNet-34 model [37] for the network ϕ_{global} to encode the global image feature ob_{global} . The network ϕ_{global} is pre-trained on ImageNet [21] data and the last layer of ϕ_{global} is changed to a 512×128 size fully-connected layer. We use a ResNet-18 model for the network ϕ_{lidar} to encode the LiDAR feature ob_{lidar} . In contrast to ϕ_{global} , ϕ_{lidar} is not pre-trained. The last layer of ϕ_{lidar} is also changed to a 512×128 size fully-connected layer. We use 1×64 and 64×128 size fully-connected layers for the network ψ_{speed} to encode the speed data feature embedding ob_{speed} . All the feature embeddings ob_{global} , ob_{lidar} , and ob_{speed} have the same feature size of $Z = 128$.

We use a Faster R-CNN model [66] which is reimplemented and pre-trained on the CARLA dataset [26]. We use two separate networks for object detection: a traffic light detector and an obstacle detector. The traffic light detector detects the traffic light signal which can be labeled as Red, Yellow, Green, or Off. The obstacle detector detects the obstacles which can be labeled as Pedestrian, Car, Bicycle, or Motorcycle. The classification result of each detector is encoded as a one-hot vector. The bounding boxes of the detected objects of two detectors are combined into the bounding box set $\{b_i\}_{i=1}^d$. The feature map \tilde{I}_t is computed from the ResNet-34 model ϕ_{ob} [37] and we use the feature map from the first convolution groups named conv1. The ResNet-34 model ϕ_{ob} shares the parameters with the global image feature encoding network ϕ_{global} . Before applying the crop operation, the feature map from ϕ_{ob} is upsampled to the size of the original image I_t with the bi-linear interpolation. The MLP networks ψ_{img} and ψ_{lin} are constructed by a fully-connected layer with the size of 64×128 and 8×128 , respectively. The MLP network ψ_{ob} is constructed by two fully-connected layers which have the size of 256×128 and 128×128 . We use two separate networks for ψ_{img} , ψ_{lin} , and ψ_{ob} depending on whether the detected object belongs to a traffic light or an obstacle. The speed data encoding network ψ_{speed} is also an

Chapter 4. Road Graph and Image Attention Network for Autonomous Driving

MLP network and constructed by two fully-connected layers which have the size of 1×64 and 64×128 . All the MLP network uses the leaky-ReLU [55] for the non-linear function.

We stack $L = 5$ transformer layers in the attention network module. Each transformer in a layer has four heads. All the linear projection weights of each transformer have the size of 128×128 . The MLP network ψ_{tran} is constructed by two fully-connected layers which have the size of 128×512 and 512×128 . We use the same network architecture as [11, 62] for the waypoint prediction network.

4.3.4 Baselines

We compare the proposed method with several baselines.

- **CILRS** [20]. CILRS is a conditional imitation learning method which uses a single front camera image and a speedometer sensor data. The network architecture of CILRS contains a conditional module which takes a navigational command as the condition.
- **RBC** [26]. RBC is a rule-based control which identifies the location of the object through a Faster R-CNN [66] and controls the ego-vehicle based on the hard-coded rules. We used a similar method with [26] but reimplemented some rules to apply it to our setting. RBC detects traffic lights and obstacles as the same as the proposed method. However, unlike the proposed method, RBC estimates the 3D position of a detected object through LiDAR data. In our implementation, RBC uses the following method to find the 3D position of the detected object. First, the center points of the bounding box of objects are projected into the normalized image coordinate using the camera calibration matrix. Likewise, the 3D coordinates of the point clouds obtained from LiDAR data are projected into the normalized image coordinate. After that, the point cloud closest to the center point of the

Chapter 4. Road Graph and Image Attention Network for Autonomous Driving

object is selected on the normalized image coordinate. The 3D coordinate of the selected point cloud is used as the 3D coordinate of the detected object. Based on the estimated 3D coordinate, RBC finds the lane in which the obstacle is currently located. If the obstacle is on the current lane and the distance between the ego-vehicle and the obstacle is less than λ , RBC stops the ego-vehicle. We have fine-tuned the parameter λ and use the best value for evaluation. The distance is computed according to the Frenet coordinate frame of the lane. RBC also stops the vehicle if the traffic light closest to the center of the image I_t is detected as red or yellow. However, if the ego-vehicle has already entered an intersection, the vehicle does not stop according to the traffic light.

- **AIM** [62]. AIM uses only the camera image as an input. The image input is encoded through ResNet [37] based networks. The encoded features are entered into the waypoint prediction network. We test two type of low-level controller to check the effect of changing low-level controller. Vanilla AIM follows the waypoints predicted by the waypoint prediction network. AIM+R follows the pre-defined path as the same as the proposed method. In AIM+R, the waypoint prediction network only decides the target speed of the ego-vehicle.
- **Transfuser** [62]. Transfuser uses a camera image and a LiDAR feature as inputs. The image and LiDAR input are encoded through ResNet [37] based networks and fused together through the attention mechanism. The fused feature is then entered into the waypoint prediction network. Similar to AIM, we test Transfuser which is the vanilla version and Transfuser-R which uses a path-following low-level controller.

Chapter 4. Road Graph and Image Attention Network for Autonomous Driving

4.3.5 Comparison Results

We use three metrics in the evaluation: route completion, driving score, and infractions per km. Route completion (RC) is the percentage of the completed route length relative to the total route length. Driving score (DS) is the product between the route completion and the penalty weight. The penalty weight starts from 1.0 and decreases when the agent commits an infraction. The types of infractions include the following events: collisions, running a red light, and running a stop sign. There are also additional infractions which are not counted when computing the penalty weight: off-road driving, route deviation, agent blocked, route timeout. Instead of reducing the penalty weight, they affect the computation of the route completion or terminate the simulation. Infractions per km (IpKm) is the total number of infractions divided by the total distance in kilometers traveled. To consider cases when a vehicle is stopped and the agent is blocked, we count all types of infractions when computing IpKm. Higher is better in DS and RC, and lower is better in IpKm. The detailed explanation of each metric can be found from the CARLA official website [8].

We compare the proposed method with the other baselines. Each method is trained and evaluated with five different random seeds. Table 4.1 shows the mean and standard deviation of the result. We observe that the proposed method shows the highest performances in all three metrics. Especially, the proposed method shows higher performance than Transfuser which does not use the road graph for the attention mechanism. The result demonstrates that the use of the road graph feature can improve the performance of the autonomous driving controller. To show the effect of changing the low-level controller, we tested AIM-R and Transfuser-R which use the same low-level controller as RIANet. However, the proposed method shows better performance than AIM-R and Transfuser-R even they use the same low-level controller. In addition, the proposed method shows better performance than RBC which uses the same object detector model. The

result shows that the proposed method leverages the detected object features more effectively compared to RBC.

4.3.6 Attention Score Visualization

We visualized the attention score of the detected objects and the nodes of the road graph. The attention score is measured from the first layer of the attention network by taking the mean of the attention weights along the query features. Figure 4.4 shows two examples on Town05. On the camera images, we marked the object with the highest attention score among the objects in red and the other objects in blue. The results show that the attention network tends to pay more attention to important objects (e.g., a jaywalking pedestrian and a vehicle that crosses the intersection) than other objects. We also visualized the relationship between the image and node features. We estimated the 3D position of the object with the highest attention score using the estimation method of RBC [26] and plotted the position on the 2D coordinates. We also plotted the road graph on the same figure. Interestingly, the object with the highest attention score (red boxes) and the node with the highest attention score (red x marks) are distributed close to each other on the 2D coordinates. From the result, we can infer that the network considers the relationship between the road graph and the object when calculating the attention scores.

4.3.7 Road Graph Feature Analysis

We investigated how the structure of the road graph affects the attention in the image. Figure 4.5 shows an example scenario. We first fed a camera image (Figure 4.5(a)) and the corresponding road graph input (Figure 4.5(b)) to the network. In this case, the network assigned the highest attention score of 0.01256 to the jaywalking pedestrian (red box). We then fed a left-curved road graph (Figure 4.5(c)) to the network, which is randomly sampled from the evaluation

Chapter 4. Road Graph and Image Attention Network for Autonomous Driving

Table 4.1: Performance Comparison

Method	Town05 Short		
	DS \uparrow	RC \uparrow	IpKm \downarrow
CILRS [20]	21.693 ± 1.751	25.762 ± 0.807	98.141 ± 14.164
RBC [26]	49.947 ± 4.966	84.016 ± 3.246	33.527 ± 4.783
AIM [62]	69.917 ± 15.104	73.106 ± 19.000	10.315 ± 2.489
AIM+R [62]	79.969 ± 18.601	82.286 ± 19.714	8.592 ± 7.382
Transfuser [62]	64.495 ± 7.840	70.915 ± 7.200	12.059 ± 2.835
Transfuser+R [62]	67.031 ± 8.800	73.254 ± 9.837	10.881 ± 2.481
RIANet (Ours)	87.469 ± 2.363	93.881 ± 3.827	6.319 ± 1.419

(a) Town05 Short

Method	Town05 Long		
	DS \uparrow	RC \uparrow	IpKm \downarrow
CILRS [20]	7.889 ± 1.047	10.751 ± 0.358	17.020 ± 1.182
RBC [26]	14.692 ± 4.019	82.866 ± 7.282	6.669 ± 0.490
AIM [62]	39.558 ± 5.056	83.018 ± 11.477	3.424 ± 0.248
AIM+R [62]	39.873 ± 5.005	80.251 ± 17.594	2.753 ± 0.325
Transfuser [62]	36.438 ± 8.412	96.650 ± 5.420	3.473 ± 0.452
Transfuser+R [62]	37.283 ± 7.049	92.815 ± 4.607	3.209 ± 0.613
RIANet (Ours)	44.719 ± 2.513	96.934 ± 2.927	2.624 ± 0.163

(b) Town05 Long

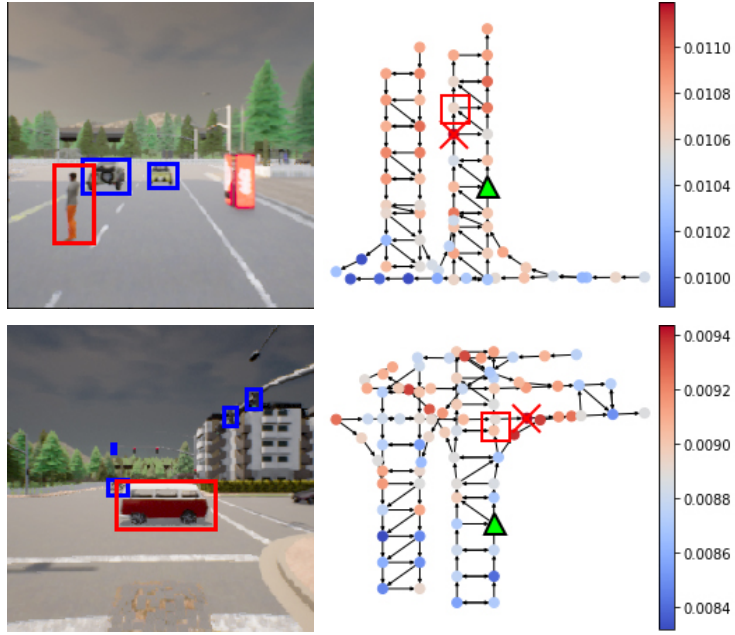


Figure 4.4: Visualization of attention scores. (Left) Detected objects on camera images. The object with the highest attention score is marked as red and the other objects are marked as blue. (Right) Road graph of scenes in 2D BEV. Green arrows indicate the position and direction of the ego-vehicle. Red boxes represent the estimated 2D position of the object with the highest attention scores and red x marks represent the 2D position of the node with the highest attention scores.

Chapter 4. Road Graph and Image Attention Network for Autonomous Driving

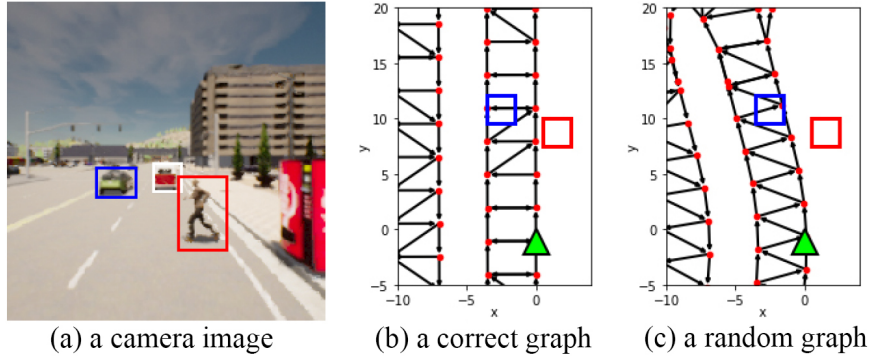


Figure 4.5: Road graph feature analysis. (a) A jaywalking pedestrian (red box) and cars (blue and white boxes) are detected on a camera image. (b) A correct road graph. (c) A random road graph. Green arrows indicate the position and direction of the ego-vehicle. Red boxes in (b) and (c) represent the estimated 2D position of the pedestrian. Also, blue boxes in (b) and (c) represent the estimated 2D position of the car marked in blue. The car marked in white box is not plotted in (b) and (c).

Chapter 4. Road Graph and Image Attention Network for Autonomous Driving

dataset. Interestingly, the network assigned the highest attention score of 0.00932 to the car (blue box) and the second-highest attention score of 0.00925 to the pedestrian (red box). It appears that the network pays less attention to the object if the object is estimated to be far from the ego-vehicle lane on the road graph. The result shows that the network can capture the structural feature of the road graph.

4.3.8 Ablation Study

As an ablation study, we tested how each input feature affects the performance of the agent. In the As an ablation study, we tested how each input feature affects the performance of the agent. In the default configuration, we used all the road graph, detected bounding boxes, global image, LiDAR, and speed features. However, in each case of no-graph, no-detection, no-global-image, and no-LiDAR setting, we ignored the corresponding feature embedding X_H , $\{ob_i\}_{i=1}^d$, ob_{global} , and ob_{lidar} , respectively. The feature fusion F is computed without the ignored feature. The network is trained once with each configuration and evaluated with three different random seeds. As shown in Table 4.2, The default configuration shows the highest DS and IpKm in both Town05 Short and Town05 Long settings. The default configuration shows a lower performance (1.671% and 3.066%) in RC but shows a higher performance in DS (3.885% and 12.757%) and IpKm (14.312% and 13.543%) compared to no-road-graph. From this comparison, it can be inferred that the agent drives more cautiously when using the road graph. In addition, the average performance of no-road-graph, no-detection, and no-LiDAR were higher than that of Transfuser in Table 4.1, because Transfuser does not use both road-graph and detection features. As a result, the ablation study shows that all the proposed feature inputs are essential for driving successfully.

Table 4.2: Ablation Study

Configure	Town05 Short		
	DS \uparrow	RC \uparrow	IpKm \downarrow
no-road-graph	84.198 \pm 1.971	95.477 \pm 3.575	7.374 \pm 2.358
no-detection	81.729 \pm 6.503	90.729 \pm 6.503	8.453 \pm 1.972
no-global-image	60.681 \pm 2.513	93.532 \pm 6.832	23.750 \pm 1.343
no-LiDAR	82.754 \pm 2.694	93.166 \pm 3.669	9.657 \pm 1.121
Default	87.469 \pm 2.363	93.881 \pm 3.827	6.319 \pm 1.419

(a) Town05 Short

	DS \uparrow	RC \uparrow	IpKm \downarrow
no-road-graph	39.660 \pm 3.092	100.000 \pm 0.000	3.035 \pm 0.203
no-detection	39.091 \pm 2.055	98.592 \pm 1.991	2.997 \pm 0.245
no-global-image	25.778 \pm 5.552	86.312 \pm 4.694	4.563 \pm 0.370
no-LiDAR	39.139 \pm 4.177	97.069 \pm 4.146	2.976 \pm 0.265
Default	44.719 \pm 2.513	96.934 \pm 2.927	2.624 \pm 0.163

(b) Town05 Long

4.3.9 Qualitative Results

We displayed simulation results of the proposed method in Figure 4.6 and Figure 4.7. In the simulations, the ego-vehicle was controlled by RIANet on CARLA Town05 Short. The network model of RIANet is fully-trained before running a simulation. We have captured the ego-vehicle in a third-person view (up) and a bird-eye view (down). The snapshots are arranged in the order of time with regular time intervals. In Figure 4.6, the ego-vehicle has stopped at the red light signal ($t = 3.0s$). Note that turning right on red is prohibited in the CARLA environment. After the signal turns green, the ego-vehicle has waited until the frontal vehicle started to move ($t = 6.0s$). After the frontal vehicle started to move, the ego-vehicle also started and has followed the frontal vehicle ($t = 9.0s$). The right turn of the ego-vehicle was completed while the signal was green ($t = 13.5s$). As a result, the ego-vehicle successfully made a right turn without violating traffic rules. Another example of a simulation result is shown in Figure 4.7. In Figure 4.7, the ego-vehicle first has changed the lane ($t = 3.0s$). The ego-vehicle then has turned left at the green light ($t = 7.5s$). The ego-vehicle did not violate the traffic signal because the ego-vehicle passed the intersection before the signal turned red. After passing the intersection, the ego-vehicle faced a jaywalking pedestrian ($t = 12.0s$). To avoid a collision with the pedestrian, the ego-vehicle has slowed down ($t = 13.5s$) and changed the lane ($t = 15.0s$). As a result, the ego-vehicle succeeded to drive without any accidents or signal violations.

We compared the proposed method with RBC [26] and Transfuser [62] by displaying simulation results. Figure 4.8 and Figure 4.9 shows snapshots of the scenarios used for the comparison. In Figure 4.8, the ego-vehicle tries to cross an intersection. RBC and RIANet successfully have driven the ego-vehicle and crossed the intersection. Although the traffic lights became yellow before crossing the intersection completely, the ego-vehicle did not violate the signal by making the first start at the green light. On the other hand, Transfuser failed to cross

Chapter 4. Road Graph and Image Attention Network for Autonomous Driving

the intersection without violating traffic sign. In the case of Transfuser, the ego-vehicle stopped once in front of the intersection ($t = 8.0s$). When the traffic signal turned red, the ego-vehicle suddenly started to move and violated the traffic signal ($t = 12.0s$). In Figure 4.9, the ego-vehicle tries to turn right while avoiding a collision with a jaywalking pedestrian. In this scenario, RBC failed to recognize the signal properly and did not move in front of the intersection. In the case of Transfuser, the ego-vehicle turned right on the intersection ($t = 7.5s$). However, the ego-vehicle turned right on the red, which is prohibited in the CARLA environment. On the other hand, RIANet successfully drove the ego-vehicle and finished turning before the traffic sign completely became red ($t = 7.5s$). In both cases of Transfuser and RIANet, the ego-vehicle faced a jaywalking pedestrian after passing the intersection. The ego-vehicle successfully avoided the pedestrian in each scenario. As a result, only RIANet succeeded to complete the path without any infraction.

4.4 Chapter Summary

In this chapter, we have presented a new driving framework that leverages the attention score between the road graph and image feature. We have proposed RIANet to capture relationship between sensed observations against the road structure via a road graph. The proposed network computes the scene context by incorporating a road graph, image, and additional features through the attention mechanism. In the experiments, we have shown that the proposed method outperforms the baseline methods in terms of all the metrics. Compared to the other methods, the proposed method can effectively leverage the attention between the road graph and image features. The results show that the use of the attention score improves the performance in urban autonomous driving.



Figure 4.6: Turn right and avoid a pedestrian scenario.

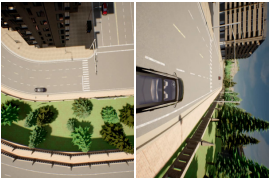

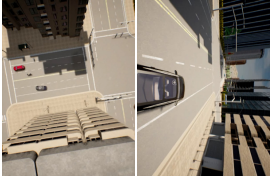
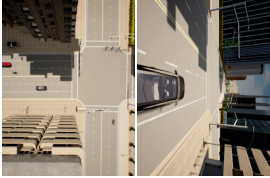
Time	$t = 1.5s$	$t = 3.0s$	$t = 4.5s$	$t = 6.0s$	$t = 7.5s$	$t = 9.0s$
Traffic light	Green	Green	Green	Green	Green	Yellow
RIANet						
Time	$t = 10.5s$	$t = 12.0s$	$t = 13.5s$	$t = 15.0s$	$t = 16.5s$	$t = 18.0s$
Traffic light	Yellow	-	-	-	-	-
RIANet						

Figure 4.7: Turn left and avoid a pedestrian scenario.

















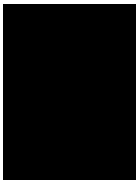
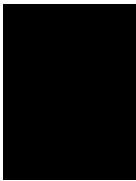
Time	$t = 2.0s$	$t = 4.0s$	$t = 6.0s$	$t = 8.0s$	$t = 10.0s$	$t = 12.0s$
Traffic light	Green	Green	Green	Yellow	Yellow	Red
RBC [26]						
Transfuser [62]						
RIANet (Ours)						

Figure 4.8: Crossing an intersection scenario.








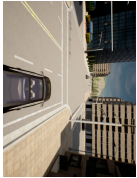

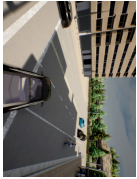








Time	$t = 2.5s$	$t = 5.0s$	$t = 7.5s$	$t = 10.0s$	$t = 12.5s$	$t = 15.0s$
Traffic light	Green	Yellow	Red	Red	Red	Red
RBC [26]						
Transfuser [62]						
RIANet (Ours)						

Figure 4.9: Turn right and avoid a pedestrian scenario.

Chapter 5

Road Graph Change Detection for Autonomous Driving

In the previous works, it is demonstrated that using a road graph representation has advantages over a feature-based representation [17, 18, 15, 71, 82], BEV image representation [59, 41], or egocentric image based representation [19, 20]. In particular, using a road graph representation has two major advantages when training a driving agent. First, a road graph can help an agent grasp the road information. For example, a road graph can give information about road connections. Second, a road graph is efficient for representing a state of an environment. A road graph has a smaller data size than an image-based input. It reduces the time complexity required for data processing.

In Chapter 3 and Chapter 4, we have presented a driving framework using a road graph representation. As a result, the road graph based controllers showed improved performances than previous learning-based controllers. However, applying a road graph representation to the real world environment is still challenging because of road change issues. The structure of roads in the real world can be changed for several reasons such as road construction or temporary traffic control.

Chapter 5. Road Graph Change Detection for Autonomous Driving

For example, the location of the road can be changed after road construction, or some of the road lanes can be blocked during a temporary traffic control. Because road graphs for RIANet were loaded from a pre-constructed road database and not constructed on the fly, our framework was vulnerable to these road change issues. This made our prior frameworks perceive inaccurate and unreliable road structural information, which resulted in performance degradation in the real world environment.

Road change detection has been studied in the field of autonomous driving. However, compared with the proposed method, there are the following differences in previous works. Pannen et al. [61] suggested a particle filter based localization system and road change detection algorithm. In their work, however, they assumed that the accurate positions of landmarks, such as lane markings or traffic signs, are provided to the ego-vehicle controller. Ding et al. [22] proposed a LiDAR based localization system and appended a local road change detection module to enhance their localization accuracy. However, their method cannot recognize visual road changes such as road marking or lane width changes because their detection module only relies on a LiDAR sensor. Heo et al. [42] suggested using deep metric learning to detect mislabeled road markings in an HD map. Likewise, Lambert et al. [49] suggested several fusion-based network architectures which can detect newly added road markings in an HD map. Unlike prior work, the proposed framework focuses on detecting structural road changes such as lane addition or lane width changes.

In this chapter, we focus on the unreliability problem of input road graphs, which is caused by road changes. For this purpose, we introduce an improved version of our framework named RIANet++. Unlike [33], which uses only a controller module, RIANet++ uses a road change detection module as well. The road change detection module detects a road change by checking the similarity score between image sensor data and pre-constructed query road graph data. By detect-

Chapter 5. Road Graph Change Detection for Autonomous Driving

ing a road change, we are able to increase the worst-case performance of the agent by filtering out unreliable road graph input. For the road change detection module, we suggest two types of detection methods. The semantic matching method converts the query road graph into a semantic image and measures the similarity score in the semantic image domain. Likewise, the graph matching method converts the image sensor data into a road graph and measures the similarity score in the road graph domain. In the experiment, we show the effects of the proposed detection methods on the robustness of the controller. The experiment is performed in two driving environments: the CARLA urban driving simulator [25] and the Future Mobility Technology Center (FMTC) real-world environment. In both cases, the proposed method successfully detects road changes and shows improved driving performance compared to other baselines, including our prior work [33].

Our contributions are summarized as follows:

- We propose a driving framework which can consider the importance of objects and the scene context by leveraging road graph data.
- To solve the unreliability issue of road graph data, we propose to combine the controller with the road change detection module, which can find the road graph error from an egocentric image and query road graph.
- The proposed detection methods successfully detect road changes and help the overall framework outperform the other baselines in both detection and driving performance.
- We also experimentally demonstrate that the proposed driving framework can overcome the performance degradation issue, which occurs by unexpected road changes, through the road change detection module.

5.1 Problem Setting

In this section, we first define the problem where the proposed framework is applied. In our problem setting, the goal is to train an urban autonomous driving agent to visit a sequence of goal locations while keeping traffic rules, such as collision avoidance and stopping at a red light. We denote the goal location set as $\{g_i\}_{i=1}^l$, where g_i is a goal location and l is the number of the goal locations. The agent needs to reach each goal location in the given order. The driving vehicle is installed with several sensors such as cameras, LiDAR, GPS, IMU, and a speedometer sensor. The agent observes its state from these sensor data at every time step t . Also, the agent is given the position and direction of the ego-vehicle from the localization system. The localization system uses GPS and IMU sensor data and estimates the vehicle odometry using the extended Kalman filter. The detailed specification of each sensor and environment setting is explained in Section 5.3.

In addition to egocentric sensor data, the driving agent is also given road graph data. A road graph is a graph which represents the topological information of a local map. In this paper, we use a graph representation method, which is defined in [32, 33]. The agent observes a road graph G_t for every time step t . A road graph G_t is a subgraph of a global graph $G_{global} = (V, E)$. The nodes and edges of this global graph G_{global} represent the topological information of the global map data. A node $n_i \in V$ represents a drivable point on the map. These points are sampled along the centerline of road segments with an interval of 3m. An edge $e_{i \rightarrow j}$ represents the connection between the node n_i and n_j . This connection is only valid if the ego-vehicle is allowed to drive from n_i to n_j . The position of node n_i and the direction of an edge $e_{i \rightarrow j}$ are stored in the database. During training or test, the agent only observes a subgraph G_t , which contains $K = 96$ nearest nodes from the agent's position. These nearest nodes in G_t are only selected among the

Chapter 5. Road Graph Change Detection for Autonomous Driving

nodes in front of the ego-vehicle with a small backward margin $\mu = 10\text{m}$. A node feature f_{n_i} and an edge feature $f_{e_{i \rightarrow j}}$ in G_t are calculated depending on the ego-vehicle state and the topological structure of G_t . In details, f_{n_i} and $f_{e_{i \rightarrow j}}$ are defined as follows:

$$f_{n_i} = (x_{n_i}, y_{n_i}, \mathbb{1}_{ts}, \mathbb{1}_{tr}, \mathbb{1}_{oc}, \mathbb{1}_{oc} \cdot v) \quad (5.1)$$

$$f_{e_{i \rightarrow j}} = (x_{e_{i \rightarrow j}}, y_{e_{i \rightarrow j}}), \quad (5.2)$$

where (x_{n_i}, y_{n_i}) is the node position, $(x_{e_{i \rightarrow j}}, y_{e_{i \rightarrow j}})$ is the edge direction, v is the ego-vehicle speed, and $\mathbb{1}_{ts}$, $\mathbb{1}_{tr}$, $\mathbb{1}_{oc}$ are the scalar-valued binary indicators. $\mathbb{1}_{ts}$ is the traffic signal indicator meaning that the node is at an intersection. $\mathbb{1}_{tr}$ is the path indicator meaning that the node is a part of the pre-defined vehicle path. $\mathbb{1}_{oc}$ is the occupancy indicator meaning that the node is the closest node to the ego-vehicle position. We use the ego-vehicle-centered coordinate frame to compute the node position (x_{n_i}, y_{n_i}) and edge direction $(x_{e_{i \rightarrow j}}, y_{e_{i \rightarrow j}})$. The x -axis of the coordinate frame is parallel to the orientation of the ego-vehicle.

The structure of roads in the real world can be changed due to road construction or temporary traffic control. Therefore, in our problem setting, we assume that some part of a global road graph G_{global} can contain a structural error due to those road changes. Because G_t is stored and loaded from a pre-constructed database of G_{global} , the structure of G_t could be different from the structure of the actual road. In this paper, we assume that the road graph G_t can be changed to G'_t by road changes. The actual road graph of the environment is G'_t , but an agent can only access to the pre-constructed road graph G_t . Figure 5.1 explains how each type of a road change can be applied to a road graph. In our problem setting, we define six different types of road changes as follows:

- **Lane Deviation (LD).** A road can deviate in a perpendicular direction to the road.

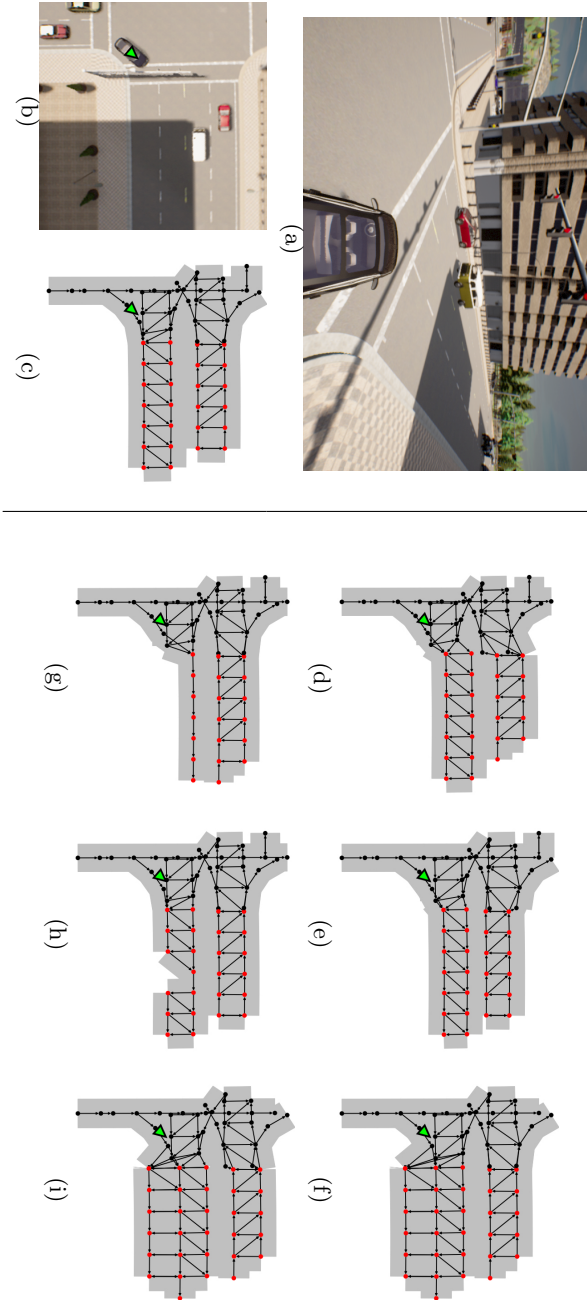


Figure 5.1: An example of a road graph and road changes in the CARLA simulator. During test, an agent is met with the actual road graph G'_t (left) but can only access to the pre-constructed road graph G_t (right). (a) A scene of a road during test. (b) Corresponding BEV image. The position of the ego-vehicle is marked as a green arrow. (c) Corresponding actual road graph G'_t . The nodes which are targets of changes are marked in red, and the other nodes are marked in black. (d) Lane deviation. (e) Lane width change. (f) Lane removal. (g) Lane addition. (h) Node Missing. (i) Combination.

Chapter 5. Road Graph Change Detection for Autonomous Driving

- **Lane Width Change (LWC).** A lane width of a road is increased or decreased.
- **Lane Removal (LR).** Compared with the road graph G_t , a road segment in the actual road graph G'_t has fewer lanes. It means that the number of lanes is decreased after the road change.
- **Lane Addition (LA).** Compared with the road graph G_t , a road segment in the actual road graph G'_t has additional lanes. It means that the number of lanes is increased after the road change.
- **Node Missing (NM).** A static object, such as traffic barriers, can hide a part of the road. In this case, nodes of the road graph G_t are missing.
- **Combination (Comb).** Some of the other road change types can be simultaneously applied to the road graph G_t .

During the evaluation, the agent is met with the actual road graph G'_t but can access only the pre-constructed road graph G_t . The detailed explanation of each road change type is described in Section 5.3.1.

5.2 Proposed Method

In this section, we explain the proposed driving framework (RIANet++). Unlike our prior framework in Chapter 4, the proposed framework assumes the unreliability of a road graph and focuses on handling it. To this end, we suggest a detection module which can detect errors on the road graph caused by road changes. The overall flow of the proposed framework is explained in Figure 5.2. The road change detection module (Figure 5.2(a)) determines whether a road change has occurred in the scene through the input camera image and the query road graph. If a road change is not detected, the query road graph is considered

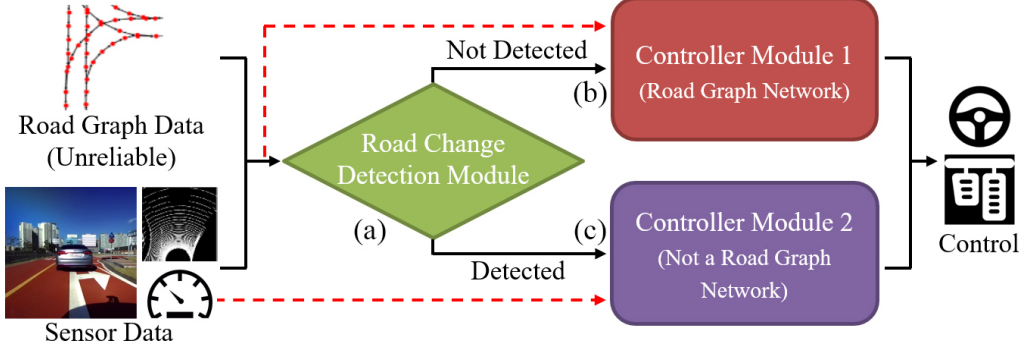


Figure 5.2: The structure of the proposed framework. (a) Road change detection module. (b) Controller module for normal cases. (c) Controller module for road change cases.

reliable. In this case, we use the same road graph based controller module as RIANet (Figure 5.2(b)). However, if a road change is detected, the query road graph is considered incorrect, and we use another controller module which ignores this road graph data (Figure 5.2(c)). The details of the controller module and road change detection module are explained in Section 5.2.1 and Section 5.2.2, respectively.

5.2.1 Controller Module

As mentioned in Section 5.2, we use a road-graph controller module (Figure 5.2(b)) when a road change is not detected. However, if a road change is detected, we use a no-road-graph controller module which is trained without road graph data (Figure 5.2(c)).

Road-Graph Controller Module for Normal Cases. When a road change is not detected, we use the same road graph based controller which is suggested in Section 4.2 (road-graph controller module). The road-graph controller module takes both road graph and ego-centric sensor data as inputs. As explained in Sec-

Chapter 5. Road Graph Change Detection for Autonomous Driving

tion 4.2, we use an attention network [78] for our controller module. We compute the attention score between a road graph and image features and leverage it to reflect the scene context. The controller module consists of three sub-modules named a feature encoder, an attention network, and a low-level controller. First, in the feature encoder module, each road graph and ego-enteric sensor data is encoded into each feature embedding. Second, in the attention network module, all the feature embeddings are fused by the attention mechanism. Finally, the low-level controller module computes the target speed of the ego-vehicle and controls it with a PID controller.

No-Road-Graph Controller Module for Road Change Cases. When a road change is detected, we control the vehicle using a different network trained without road graph data (no-road-graph controller module). The differences between the road-graph and no-road-graph controller modules are as follows: First, the no-road-graph controller module does not take a road graph as an input to the network. Both road-graph and no-road-graph controller modules have similar network structures. However, the no-road-graph controller module uses the feature fusion vector $F = F_{ob}$ instead of (4.13). Second, for the steering control, the no-road-graph controller module does not use the pre-defined path ζ when determining the steering control value. As explained in Section 4.2.3, the road-graph controller module uses path ζ to determine the steering control value. However, since the path ζ is calculated from the road graph G_{global} , there is a concern that a road change in G_{global} also occurs an inaccuracy in the path ζ . Therefore, instead of following the path ζ , the no-road-graph controller module controls the steering of the vehicle to follow the predicted waypoints $\{w_t\}_{t=1}^T$. These waypoints are predicted by the waypoint prediction network [11, 62] (Section 4.2.3) of the no-road-graph controller module.

5.2.2 Road Change Detection Module

In this section, we describe a road change detection module which is mentioned in Section 5.2. The road change detection module uses a camera image to determine whether the query road graph stored in the database matches with the road graph of the current environment. In this paper, we suggest two types of matching methods for our road change detection module: graph matching and semantic matching. Figure 5.3 illustrates the difference between two detection methods. Each method has its own advantages depending on the environment.

Graph Matching Method. In the graph matching method, the detection module first reconstructs a road graph of the current environment from a camera image. The detection module then compares the reconstructed road graph with the query road graph. To reconstruct the road graph, we use a method similar to Sat2Graph [40]. First, we convert a given camera image into a 2D BEV image which covers $32\text{m} \times 32\text{m}$ area in front of the ego-vehicle with a resolution of 256×256 . This 2D BEV image is entered into the deep layer aggregation (DLA) [85] network and processed into a grid image of 19 channels. The first output channel of DLA indicates the node existence probability for each pixel location (p_x, p_y) . In addition, the $(3i + 1)$ -th channel indicates the probability that there exists the i -th edge in the corresponding node at (p_x, p_y) , and the $(3i + 2)$ -th and $(3i + 3)$ -th channels indicate the direction of the i -th edge. The DLA network is trained from the dataset in a supervised learning manner. The road graph is reconstructed from the output image of DLA. We set a threshold to determine whether a node or edge exists in each pixel location. We use 0.1 as a threshold for the node existence. If the probability of node existence does not exceed 0.1, it is also determined that the edge does not exist in the node. If the probability of node existence exceeds 0.1, we use the following distance function to determine

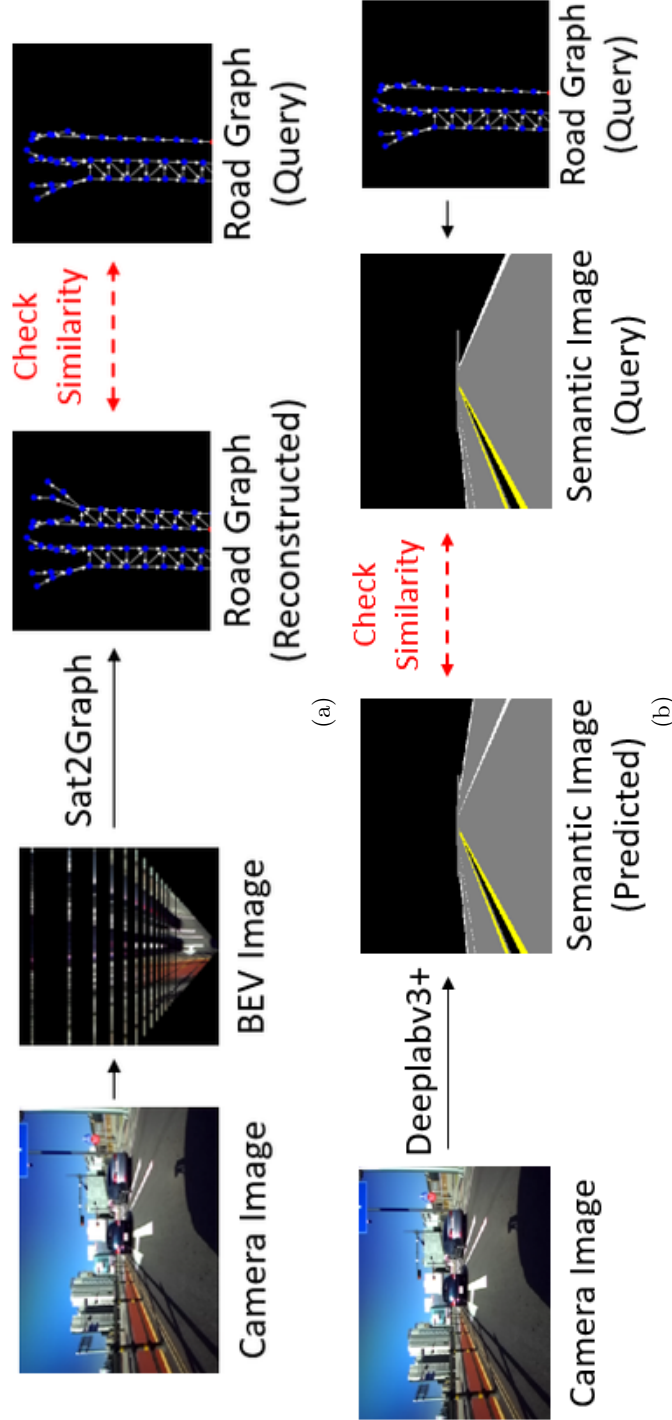


Figure 5.3: Illustration of the proposed road change detection methods. (a) Graph matching method. (b) Semantic matching method.

Chapter 5. Road Graph Change Detection for Autonomous Driving

whether the i -th edge of the node u is connected to the surrounding nodes v .

$$\begin{aligned} \text{dist}_i(u, v) = & \|p_u + \Delta p_u^i - p_v\|_2 \\ & + \alpha \cdot \cos_{\text{dist}}(\Delta p_u^i, (p_u - p_v)), \end{aligned} \quad (5.3)$$

where p_u and p_v are the pixel locations of the nodes u and v , Δp_u^i is the predicted direction of the i -th edge of the node u , \cos_{dist} is the cosine distance between two vectors, and α is the weight of the cosine distance. In the distance function, the first term indicates how much the predicted position of the outgoing node ($p_u + \Delta p_u^i$) matches with the node position (p_v) and the second term indicates how much the predicted edge direction (Δp_u^i) matches with the edge direction ($p_v - p_u$). In the implementation, we set α to 100. We connect the node u and v if the i -th edge existence probability of the node u is larger than a threshold 0.1 and the distance function $\text{dist}_i(u, v)$ is smaller than a threshold 3.0.

To determine whether the reconstructed road graph matches the query road graph, we check the similarity score between two road graphs. We use TOPO [6] and APLS [77] as similarity scores because they are among the most widely used metrics for checking graph similarity. To measure TOPO, we first find maximum one-to-one matching between the nodes of two graphs G_1 and G_2 . In this case, the distance between the matched nodes should not exceed 4m in the world coordinate frame. TOPO is measured through the following equation:

$$\text{TOPO}(G_1, G_2) = \frac{2 \cdot M(G_1, G_2)}{V(G_1) + V(G_2)}, \quad (5.4)$$

where $M(G_1, G_2)$ is the number of the matched nodes between G_1 and G_2 , and $V(G_1)$ and $V(G_2)$ are the number of the nodes in G_1 and G_2 , respectively.

To measure APLS, we sample two nodes u and v in the graph G_1 and find the shortest path between u and v . We then find the nodes $u', v' \in G_2$ which are the nearest nodes to u and v , respectively. APLS is measured through the following

Chapter 5. Road Graph Change Detection for Autonomous Driving

equation:

$$\text{APLS}(u, v) = 1 - \min \left(1, \frac{|L(u, v) - L(u', v')|}{L(u, v)} \right), \quad (5.5)$$

where $L(u, v)$ is the shortest path length between the node u and v . If the path between u' and v' does not exist, $L(u', v')$ is considered as an infinite value. To calculate APLS between two graphs, we sample all the valid node pairs in G_1 and use the mean value of APLS as the similarity score.

TOPO and APLS range between zero and one. Also, their values become higher when the two graphs are more similar. Therefore, we can use both metrics to compute the similarity score between two graphs. The graph matching based detection module determines road changes when the similarity score between the reconstructed road graph and the query road graph becomes below a detection threshold. In the experiment, both metrics are evaluated under different conditions.

Semantic Matching Method. In the semantic matching method, the detection module first predicts the camera-based semantic segmentation image using a DeepLabv3+ [14] network. To ignore occlusions by obstacles and only consider the road structure, we define four classification types: empty, driving area, yellow line, and white line. During training and test, the DeepLabv3+ network ignores occlusion and predicts the structural information of the road. In addition, the network ignores the difference between a dotted line and a straight line and predicts only the position of lines. The detection module also synthesizes the semantic segmentation image from the query road graph. The synthesized semantic image is compared with the semantic image predicted by DeepLabv3+. As a result, the detection module discerns the road change through the similarity score between two semantic images.

The process of synthesizing a semantic image from a query road graph is as follows: First, we find the positions of the left and right boundaries for each lane

Chapter 5. Road Graph Change Detection for Autonomous Driving

in the graph. The width of the road is calculated from the width between the side lane. For a one-way-one-lane road, we use a fixed lane width value of 3.5m to find the boundaries. For a road with multiple lanes, we can find the center line of the road from the leftmost boundary among the lanes. We draw a yellow line along this road center line on the 2D plane. We also draw a white line for the rest of the lane boundary positions. In the road environment setting we use, yellow lines always take place in the center of the road, and there is no yellow line on the edge of the road. Finally, the synthesized semantic image is transformed into a given camera viewpoint.

To check the similarity score between the predicted semantic image and the synthesized semantic image, we measure the mean average precision (mAP) and mean intersection over union (mIoU), which are widely used metrics for evaluating the performance of semantic segmentation. The semantic matching based detection module determines road changes when the similarity score between two images becomes below a detection threshold. We also compare the detection performance of both mAP and mIoU in the experiments.

5.3 Experiments

In the experiments, we first show that the road graph based controller is effective for urban autonomous driving. However, we also show that a road change in the environment can harm the performance of the controller module. We then demonstrate that this harmful effect can be reduced by the proposed road change detection module in RIANet++.

5.3.1 Environments

We use two environments for our experiment: the CARLA simulator and FMTC real-world driving environment. The CARLA simulator [25] is one of the most

Chapter 5. Road Graph Change Detection for Autonomous Driving

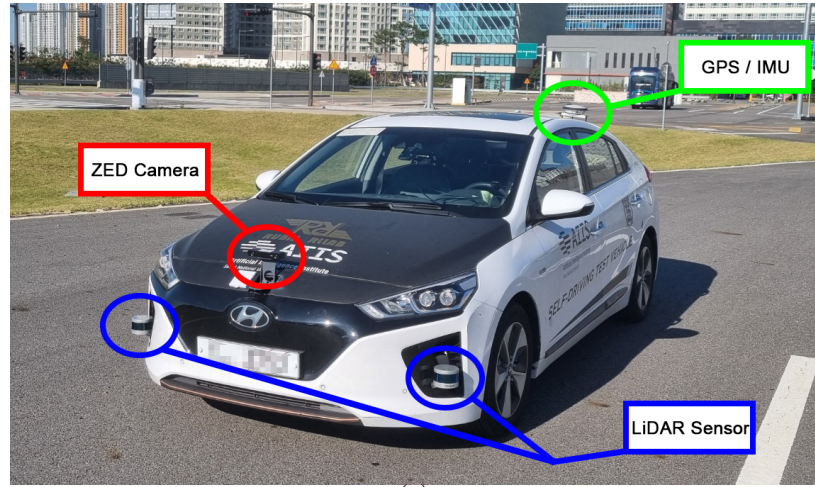
widely used urban simulators. CARLA contains various map environments from Town01 to Town07 and includes complex scenarios such as pedestrian avoidance and crossing intersections. To train the controller and road change detection modules, we use a dataset collected by [62]. We use Town05 for the evaluation to check the performance of the controllers and detection modules. The Town05 environment is divided into Town05 Long and Town05 Short scenarios depending on the length of the route distance. The road graph G_{global} is constructed from the HD-map data [1] by following the method in [33]. Future Mobility Technology Center (FMTC) is a real-world test driving facility for self-driving vehicles in Siheung, Korea. FMTC includes urban driving scene components such as intersections, crosswalks, and traffic lights. Figure 5.5 shows a snapshot of FMTC and the corresponding road graph we constructed. We collected expert data using our test vehicle in FMTC. Figure 5.4 shows the vehicle platform and sensor configuration we used for the experiment. The FMTC dataset includes four driving scenarios which are illustrated in Figure 5.6. For the reality of the scenarios, each data is collected by changing the driving lanes and allowing one or two other vehicles to drive near the ego-vehicle. The collected dataset contains a total of 14,406 frames with 0.5s time intervals. We take a satellite image of FMTC and manually draw the centerline of the roads to extract a road graph from it. The positional information in the driving dataset is also manually adjusted to reduce the localization error. For all experiments, the networks of the controller and road change detection modules are trained once using the entire training dataset and evaluated for each evaluation scenario.

5.3.2 Performance of Road Graph Based Controller Module

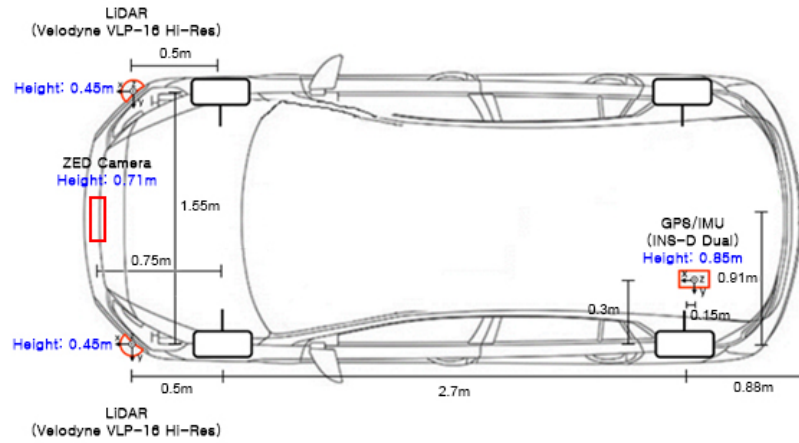
We first conduct the experiments in environments without road change conditions to measure the performance of the road graph based controller module itself.

CARLA Environment. In the CARLA environment, we compare the proposed

Chapter 5. Road Graph Change Detection for Autonomous Driving



(a)



(b)

Figure 5.4: Vehicle platform and sensor configuration. (a) Hyundai Ioniq platform used for collecting the road graph and expert dataset. The vehicle is equipped with a ZED camera, two Velodyne LiDARs, and a GPS+IMU based localization system. (b) The illustration of the installed sensor configuration.

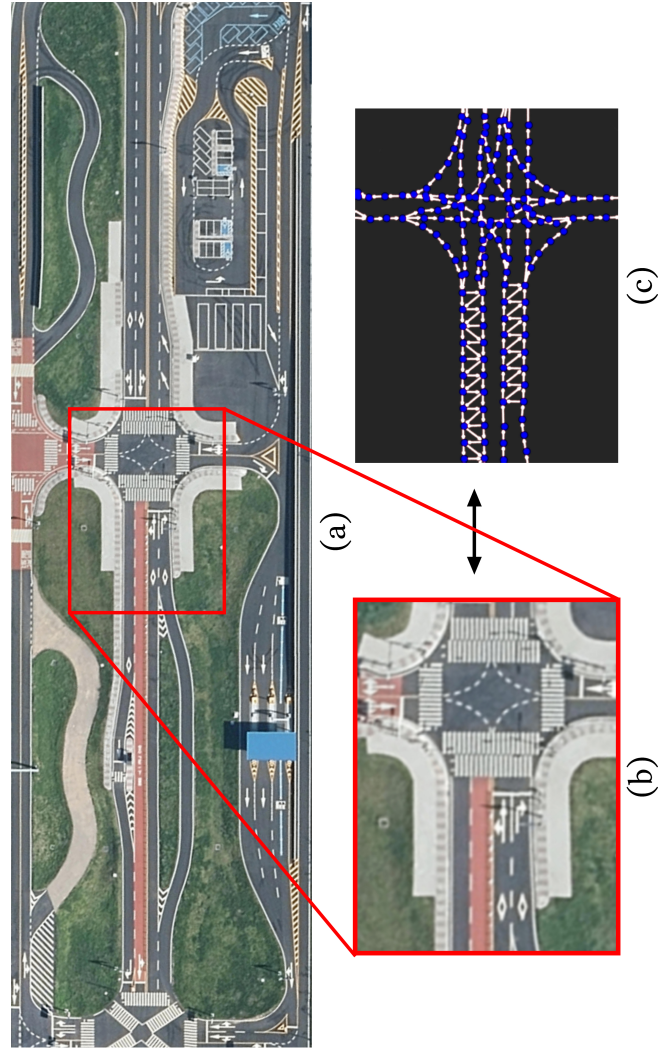


Figure 5.5: Snapshot of FMTC environment. (a) The entire map image of FMTC. (b) A part of the FMTC environment. (c) Corresponding road graph.

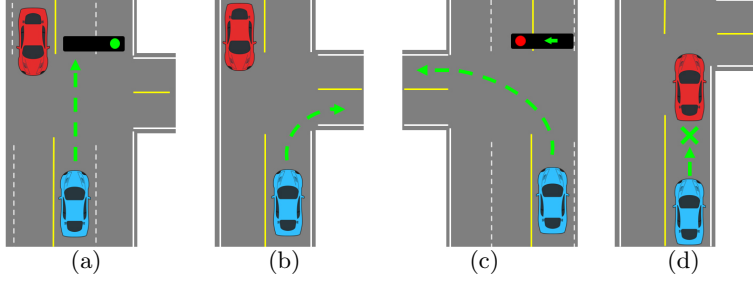


Figure 5.6: Illustrations of four real-world experience scenarios. Various types of roads are used for each scenario. (a) Go straight on a straight road or an intersection. (b) Turn right on an intersection. (c) Turn left on an intersection. (d) Stop behind another vehicle.

controller module with the other baselines which do not use a road graph as an input to the network. CILRS [20] is an imitation learning based method which uses only a camera image, a speedometer sensor data, and a navigational command as inputs to the network. AIM [62] is also an imitation learning based method, but it uses LiDAR 2D BEV grid image as an additional input. Likewise, Transfuser [62] takes the same input as AIM, but it uses the attention mechanism [78] to fuse image and LiDAR inputs to consider the relationship between them. While vanilla AIM and Transfuser directly follow the predicted waypoints, AIM+R and Transfuser+R are designed to follow the pre-defined path ζ as done in the proposed method. The predicted waypoints of AIM+R and Transfuser+R are only used for calculating the target speed. RBC [26] is a rule-based controller which uses a Faster R-CNN [66] to identify the locations of objects on the scene. The details of the rules for RBC are described in Section 4.3.4.

We use three metrics for the comparison in the CARLA environment: Route completion (RC) is the percentage of the completed route. Driving score (DS) is the product of the route completion and the penalty weight. The penalty weights range from 1.0 to 0.0 and decrease when the agent fails to follow the traffic rules.

Chapter 5. Road Graph Change Detection for Autonomous Driving

Infractions per km (IpKm) is the number of committed infractions divided by the total traveling distance in kilometers. Examples of infractions include collisions, running on red, and off-road driving. Higher is better in DS and RC, and lower is better in IpKm. The details of each metric are described on the CARLA official website [8].

Table 5.1 shows the performance comparison results between each baseline controller module. Each method is tested with five different random seeds. In the experiment, the proposed controller module shows the highest performance in all cases. In particular, the proposed controller module shows higher performance than Transfuser and AIM. Also, even though AIM+R and Transfuser+R follow the same path ζ as the proposed controller module, they still show lower performance. The results show that the proposed controller module outperforms the other baseline methods by using the road graph data as an input to the network.

FMTC Real-World Driving Environment. We also compare the performance of the controller modules in the FMTC real-world driving environment. The results are shown in Table 5.2. Due to the limitations of our vehicle platform and safety issues, we do not use the same comparison method as the CARLA experiment. We instead compare the L_2 norm between the waypoints predicted by the waypoint prediction network [11, 62] and the expert waypoints in the evaluation dataset. This L_2 norm is noted in meters and shows how well each controller network module is trained to imitate the expert’s action. Because only AIM and Transfuser use the waypoint prediction network, we compare the proposed controller module with these two baseline networks. In the result, we observe that the proposed controller module shows the lowest L_2 norm. It means that the waypoints predicted by the proposed controller module are most similar to the expert’s waypoints. Therefore, it can be said that the road graph network is most effective for training urban driving controllers.

Chapter 5. Road Graph Change Detection for Autonomous Driving

Table 5.1: Performance comparison of controller module (CARLA)

Method	Town05 Short		
	DS \uparrow	RC \uparrow	IpKm \downarrow
CILRS [20]	21.693 \pm 1.751	25.762 \pm 0.807	98.141 \pm 14.164
RBC [26]	49.947 \pm 4.966	84.016 \pm 3.246	33.527 \pm 4.783
AIM [62]	69.917 \pm 15.104	73.106 \pm 19.000	10.315 \pm 2.489
AIM+R [62]	79.969 \pm 18.601	82.286 \pm 19.714	8.592 \pm 7.382
Transfuser [62]	64.495 \pm 7.840	70.915 \pm 7.200	12.059 \pm 2.835
Transfuser+R [62]	67.031 \pm 8.800	73.254 \pm 9.837	10.881 \pm 2.481
RIANet++ (Ours)	87.469 \pm 2.363	93.881 \pm 3.827	6.319 \pm 1.419

(a) Town05 Short

Method	Town05 Long		
	DS \uparrow	RC \uparrow	IpKm \downarrow
CILRS [20]	7.889 \pm 1.047	10.751 \pm 0.358	17.020 \pm 1.182
RBC [26]	14.692 \pm 4.019	82.866 \pm 7.282	6.669 \pm 0.490
AIM [62]	39.558 \pm 5.056	83.018 \pm 11.477	3.424 \pm 0.248
AIM+R [62]	39.873 \pm 5.005	80.251 \pm 17.594	2.753 \pm 0.325
Transfuser [62]	36.438 \pm 8.412	96.650 \pm 5.420	3.473 \pm 0.452
Transfuser+R [62]	37.283 \pm 7.049	92.815 \pm 4.607	3.209 \pm 0.613
RIANet++ (Ours)	44.719 \pm 2.513	96.934 \pm 2.927	2.624 \pm 0.163

(b) Town05 Long

Table 5.2: Average L_2 norm in meters between the expert waypoints and the waypoints calculated by controller module (FMTC Real-World)

Method	Scenario 1	Scenario 2	Scenario 3	Scenario 4
AIM	1.026	0.674	0.640	0.362
Transfuser	0.745	0.499	0.272	0.375
RIANet++ (Ours)	0.349	0.285	0.231	0.156

5.3.3 Performance of Road Change Detection Module

We now test the performance of the road change detection module through the experiments. We measure the similarity score between the camera image and the query road graph in each experiment. For positive samples, we use query road graphs in which road change processes described in Section 5.1 are applied. On the other hand, for negative samples, we use query road graphs in which the road change processes are not applied. It means that the road graph in a negative sample shows the accurate road graph of the scene. To validate the performance of the road change detection module, we set a determining threshold for the similarity score. We then obtain a precision-recall curve of the detection module by changing this threshold. In each experiment, we show the average precision computed from each precision-recall curve.

To compose training and evaluation data for road change detection, we apply the road changes, which are described in Section 5.1, to the collected road graph data. In the case of lane deviation change, each node in a road moves in a direction perpendicular to the road's tangent line. A road can deviate in both left or right directions. The degree of a road deviation is randomly sampled from a uniform distribution between 1.5m and 3.0m. In the case of lane width change, the degree of a lane width change is randomly sampled from a uniform distribution between 1.0m and 1.5m. This lane width change is equally applied to all lanes on one road segment. When a road has two or more lanes, the distance between the lane center lines also is changed when the lane width is changed. In the case of lane removal and lane addition, the number of lanes decreased or increased by one. In the case of node missing, we assume that a road graph G_t can have a missing node up to one compared with the actual road graph G'_t . In the case of combination, lane deviation and lane width change are applied to a road graph simultaneously. In addition, one additional lane change type is randomly selected among lane removal, lane addition, and node missing and then applied to the road graph.

Chapter 5. Road Graph Change Detection for Autonomous Driving

We validate two types of the proposed detection methods described in Section 5.2.2: graph matching and semantic matching. For the graph matching method, we test TOPO [6] and APLS [77] as the similarity scores. For the semantic matching method, we test mAP and mIoU as the similarity scores. We also test two baseline methods for comparison. ResNet discriminator [49] uses a ResNet-34 [37] model and takes the camera image and the query semantic image as inputs. The query semantic image is synthesized from the query road graph and stacked with the camera image at a channel level. The ResNet-34 network is trained to predict the probability of road change occurrence in the scene. By applying a threshold on this probability, we calculate the average precision of the ResNet discriminator. Deep metric learning [42] also uses the camera image and the query semantic image as inputs. However, in deep metric learning, two different networks encode both camera image and query semantic image into the same feature space, and the similarity score is calculated from the cosine similarity between the encoded features. The encoder networks are trained with a triplet loss to make the semantic image without road changes have higher cosine similarity than the semantic image with road changes. We also calculate the average precision of the deep metric learning method using the similarity score.

CARLA Environment. In Table 5.3, the mIoU-based semantic matching method outperforms the other methods in terms of average precision. In Town05 Short, the semantic matching method shows the highest performance for all road changes. In Town05 Long, the mIoU-based semantic matching method also shows the highest performance except for lane deviation and node missing cases. However, even in those two cases, the mIoU-based semantic matching method shows no significant difference in performance from the method with the highest performance. As a result, in the CARLA environment, we find that the proposed mIoU-based semantic matching method shows the highest performance at road change detection.

Table 5.3: Performance comparison of road change detection module by average precision (CARLA)

Configure	Town05 Short					
	LD	LWC	LR	LA	NM	Average
ResNet Discriminator [49]	0.453	0.436	0.399	0.492	0.411	0.437
Deep Metric Learning [42]	0.852	0.812	0.563	0.747	0.578	0.675
Graph Matching (TOPO)	0.588	0.593	0.573	0.586	0.570	0.584
Graph Matching (APLS)	0.626	0.601	0.566	0.646	0.615	0.611
Semantic Matching (mAP)	0.951	0.887	0.580	0.901	0.543	0.792
Semantic Matching (mIoU)	0.953	0.911	0.734	0.913	0.605	0.838

(a) Town05 Short

Configure	Town05 Long					
	LD	LWC	LR	LA	NM	Average
ResNet Discriminator [49]	0.468	0.478	0.436	0.487	0.456	0.466
Deep Metric Learning [42]	0.875	0.824	0.638	0.695	0.677	0.756
Graph Matching (TOPO)	0.626	0.613	0.610	0.608	0.589	0.609
Graph Matching (APLS)	0.610	0.605	0.566	0.627	0.597	0.602
Semantic Matching (mAP)	0.959	0.870	0.604	0.824	0.570	0.787
Semantic Matching (mIoU)	0.953	0.895	0.760	0.839	0.648	0.835

(b) Town05 Long

Chapter 5. Road Graph Change Detection for Autonomous Driving

FMTC Real-World Driving Environment. In Table 5.4, the mIoU-based semantic matching method also shows the highest performance in lane deviation and lane width change cases. However, on average, the method with the highest performance is the TOPO-based graph matching method. Even though it does not show the highest average precision, the mIoU-based semantic matching method still shows a higher performance compared with ResNet discriminator [49] and deep metric learning [42] in average cases. We analyze the causes of degradation in the semantic matching method in the FMTC environment. Compared with the CARLA dataset, it is observed that the FMTC dataset contains more localization errors. Even though we manually adjust the errors in the training data, the errors are not adjusted for the evaluation data. The localization errors can cause viewpoint discrepancy when synthesizing a semantic image from a query road graph. Therefore, localization errors can result in the synthesis of the wrong semantic images, which consequently interferes with accurate calculations of the similarity score and the performance of the detection module.

5.3.4 Robustness of Controller under Road Change Condition

In this experiment, we verify that the road change detection module can actually increase the robustness of the controller under the road change condition. The proposed method (RIANet++) uses both the detection module and controller module as described in Section 5.2 while RIANet uses only the controller module and does not detect road changes.

CARLA Environment. In the CARLA environment, we randomly select three road segments among the roads on the entire route and apply road changes. In addition, the types of applied road changes are also randomly selected for each scenario. For the road change detection module in RIANet++, we use the semantic matching method in the CARLA environment. We use mIoU as the similarity score, and the detection threshold is set to 0.46. The results of the experiment

Table 5.4: Performance comparison of road change detection module by average precision (FMTC Real-World)

Configure	FMTC Real-World					
	LD	LWC	LR	LA	NM	Comb
ResNet Discriminator [49]	0.355	0.367	0.408	0.348	0.368	0.343
Deep Metric Learning [42]	0.414	0.434	0.457	0.433	0.403	0.420
Graph Matching (TOPO)	0.669	0.701	0.648	0.726	0.699	0.753
Graph Matching (APLS)	0.683	0.609	0.640	0.665	0.622	0.683
Semantic Matching (mAP)	0.811	0.706	0.549	0.604	0.557	0.698
Semantic Matching (mIoU)	0.853	0.770	0.614	0.415	0.696	0.586
						0.656

Chapter 5. Road Graph Change Detection for Autonomous Driving

are shown in Table 5.5. We observe that RIANet shows a performance degradation under the road change conditions, and it demonstrates that the performance of the road graph based controller can be vulnerable to road changes. On the other hand, RIANet++ shows robustness against road changes, and it shows the performance which is similar to the case when there is no road change. The result demonstrates that the road change detection module in RIANet++ can relieve the controller module’s performance degradation which is caused by road changes.

FMTC Real-World Driving Environment. We also conduct a similar experiment in FMTC. We test RIANet and RIANet++ in both road change condition cases and without road change condition cases. In the road change condition case, road changes are applied to one randomly selected road on the route, and the type of road change is also randomly selected. For the road change detection module in RIANet++, we use the graph matching method in the FMTC environment. We use TOPO [6] as the similarity score, and the detection threshold is set to 0.49. Similar to Section 5.3.2, we compare the L_2 norm accuracy of the predicted waypoints between RIANet and RIANet++. We show the average experiment results for each scenario in Table 5.6. In the results, RIANet and RIANet++ generally show lower performance when road changes are applied. However, the errors of RIANet are increased by 5.779% on average in the road change condition, while the errors of RIANet++ are only increased by 2.411% increase on average. The results show that the proposed RIANet++ is more robust to road changes and shows more consistent performances than RIANet.

5.4 Chapter Summary

In this chapter, we have proposed a road graph based autonomous driving framework which is robust to road changes. For urban driving, the proposed framework

Table 5.5: Performance consistency of RIANet++ under road change conditions (CARLA)

Configure	Road Changes	Town05 Short		
		DS \uparrow	RC \uparrow	IpKm \downarrow
RIANet	no	87.469 ± 2.363	93.881 ± 3.827	6.319 ± 1.419
	yes	74.012 ± 3.902	90.372 ± 4.956	16.009 ± 2.722
Performance Drop		-13.457	-3.509	-9.690
RIANet++	no	86.930 ± 3.960	97.032 ± 2.657	6.737 ± 1.616
	yes	83.224 ± 3.547	94.964 ± 3.961	8.921 ± 2.287
Performance Drop		-3.706	-2.068	-2.184

(a) Town05 Short

Configure	Road Changes	Town05 Long		
		DS \uparrow	RC \uparrow	IpKm \downarrow
RIANet	no	44.719 ± 2.513	96.934 ± 2.927	2.624 ± 0.163
	yes	27.724 ± 7.514	79.653 ± 6.314	4.135 ± 1.083
Performance Drop		-16.995	-17.281	-1.511
RIANet++	no	41.714 ± 3.988	100.000 ± 0.000	3.024 ± 0.227
	yes	39.726 ± 7.336	96.152 ± 4.728	3.312 ± 0.688
Performance Drop		-1.988	-3.848	-0.288

(b) Town05 Long

Table 5.6: Performance consistency of RIANet++ under road change conditions (FMTC Real-World)

	Scenario 1		Scenario 2		Scenario 3		Scenario 4		Average L_2 Norm Increase Due To Road Changes (%)
Road Changes	no	yes	no	yes	no	yes	no	yes	
RIANet	0.349	0.364	0.285	0.300	0.231	0.247	0.156	0.169	5.779
RIANet++	0.393	0.392	0.209	0.226	0.210	0.211	0.142	0.148	2.411

Chapter 5. Road Graph Change Detection for Autonomous Driving

captures the scene context by fusing road graph image data through the attention mechanism. In addition, we introduce road change detection methods to solve the unreliability issue of road graphs. By measuring the similarity between the road graph and image data, the proposed framework can consider the reliability of the road graph when deciding the control output. We successfully applied it to driving environments such as the CARLA simulator and the FMTC real-world driving environment. In the experiments, we have shown that the detection module in the proposed framework can successfully detect road changes in road graphs. As a result, the proposed framework outperformed other baselines and showed stable control performance even in conditions where the road graphs are inaccurate.

Chapter 6

Conclusion

In this dissertation, we have investigated several methods for utilizing road information in autonomous driving. We have proposed a representation method which provides road information in the form of a graph. We have also proposed effective network architectures which can handle the proposed representation. In addition, we have dealt with several issues which can arise when using a road graph in the real world.

In Chapter 3, we have proposed an autonomous driving framework, named road graphical neural network (Road-GNN), which can leverage road information for driving. The road information is represented by a graph which includes the road connection and vehicle features. In the experiment, we evaluated the baseline methods in various roundabout environments. We experimentally demonstrated that the proposed method outperforms the other non-road-graph-based methods. The results have shown that representing the road environment state with a road graph can be effective for autonomous driving.

In Chapter 4, we have improved Road-GNN and proposed a sensor fusion based driving framework named road and image attention network (RIANet). In RIANet, we have represented the road environment state by combining road

Chapter 6. Conclusion

information, camera image, and other sensor data. The proposed network fuses the input data by attention mechanism [78]. Through the attention mechanism, the proposed network is designed to consider the importance of objects according to the road structure. In the experiment, we have demonstrated that the fusion-based method can enhance driving performance. In addition, we have shown that the proposed network can successfully attend to important features when driving in a complex road environment.

In Chapter 5, we have proposed a method that can detect an error in a road graph and thereby prevent performance degradation caused by the error. The proposed method can detect a changed part of a road graph and ignore the road graph when a road change is detected. We also have combined the proposed detection method with RIANet. By combining the road change detection module and the controller module, we have suggested a robust driving framework named RIANet++. In the experiment, we have shown that the proposed driving framework can improve the worst-case performance of the controller by selecting between a road-graph-based and non-road-graph-based network.

Appendices

Appendix A

Collected map images of roundabout environment

Figure A.1 shows examples of roundabout images we collected.

Appendix A. Collected map images of roundabout environment

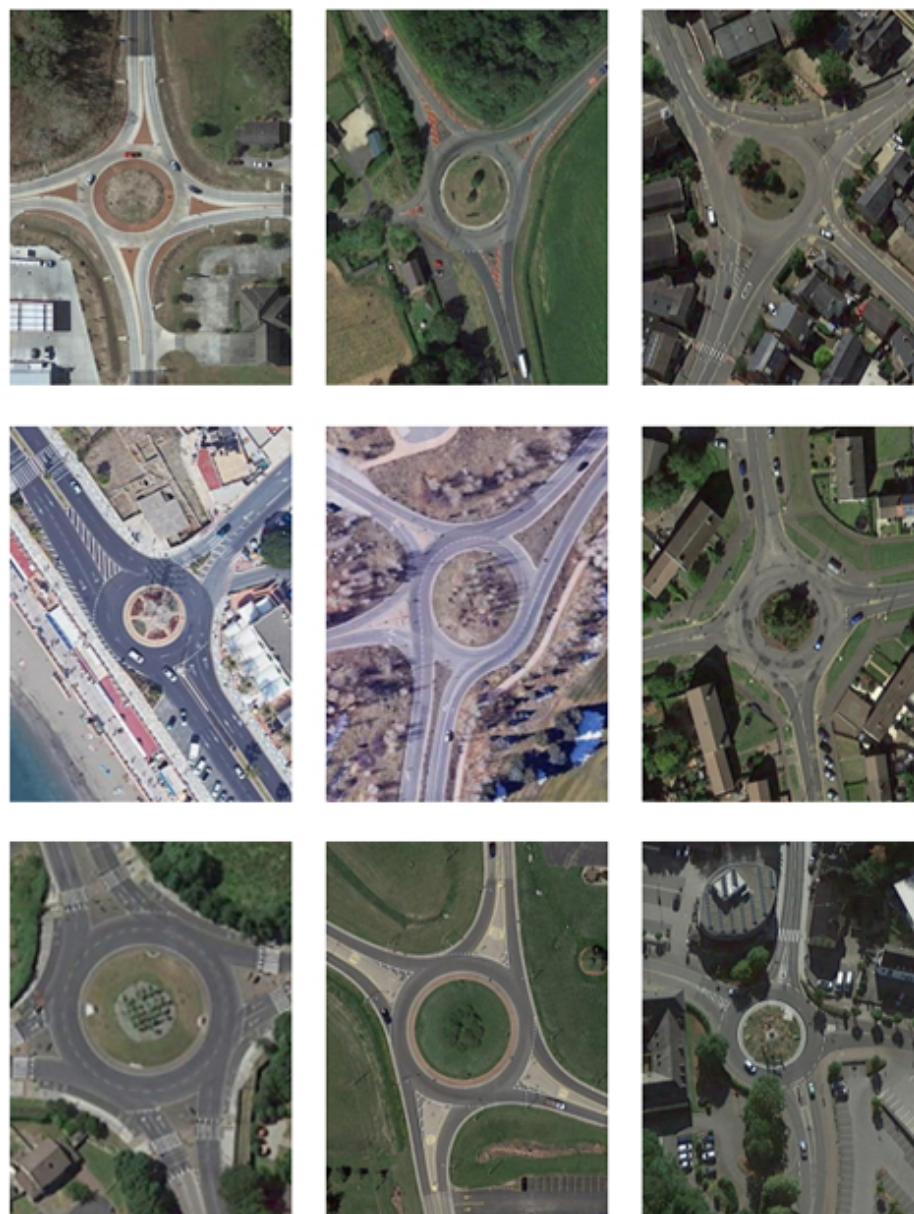


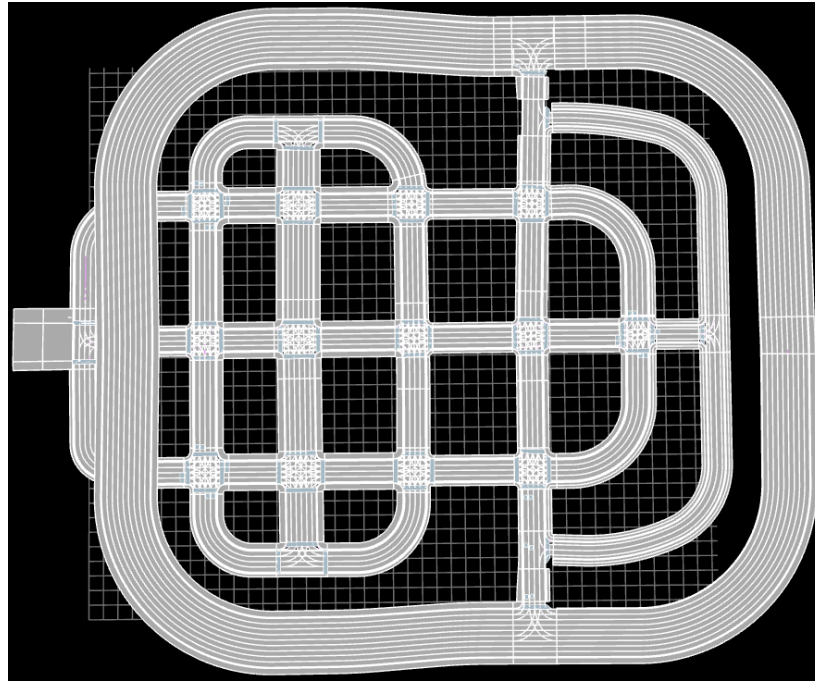
Figure A.1: Examples of collected map images.

Appendix B

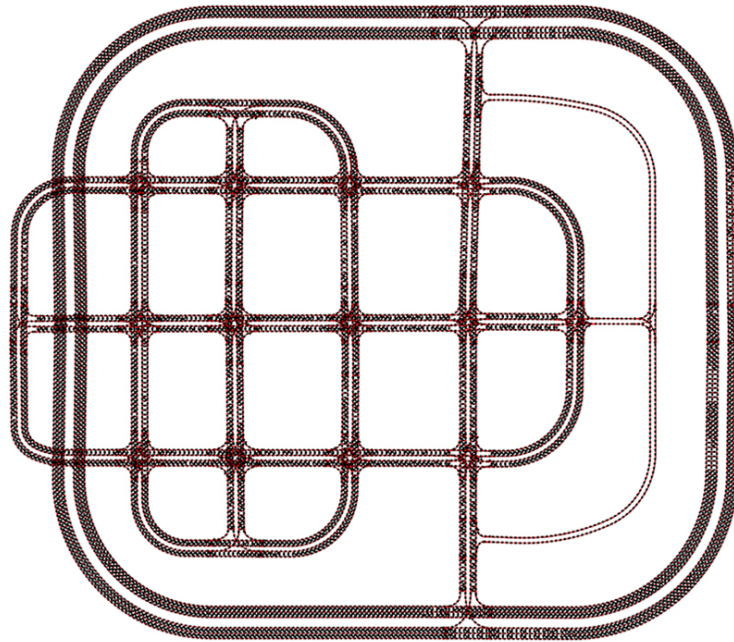
Map image and road graph of CARLA Town05

As explained in Section 4.3.2, a global road graph G_{global} is constructed from HD-map data of a CARLA town. Figure B.1 shows an example of HD-map data and the corresponding G_{global} , which are from the CARLA Town05 [25].

A more detailed example of a road graph construction is shown in Figure B.2. Figure B.2(a) visualizes a part of HD-map in the CARLA Town05. The corresponding road graph is shown in Figure B.2(b). In the CARLA simulator, the captured HD-map and road graph represent a four-way intersection. Figure B.2(c) shows a BEV image of this intersection in the CARLA simulator.



(a)



(b)

Figure B.1: HD-map and global road graph of Town05. (a) HD-map. (b) Global road graph.

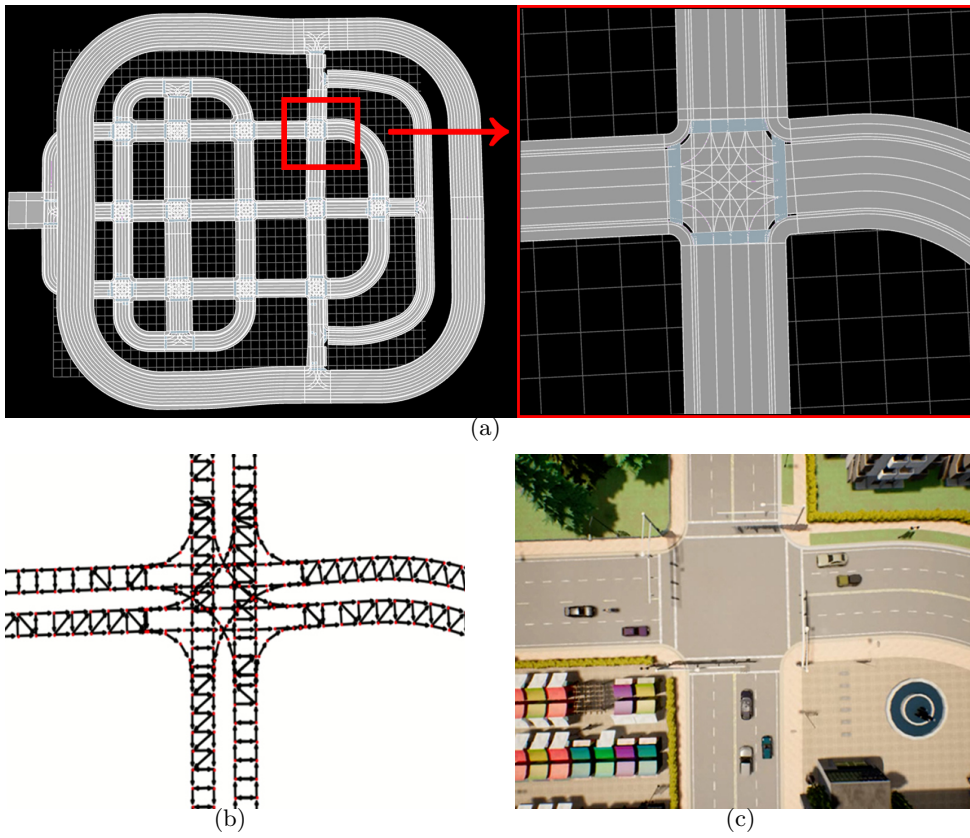


Figure B.2: A part of Town05 map example (a) A part of HD-map. (b) Corresponding road graph. (c) Corresponding BEV image of the road in the CARLA simulator.

Appendix B. Map image and road graph of CARLA Town05

Appendix C

Details of CARLA evaluation metrics

As described in Section 4.3.5, we use three metrics for evaluation in the CARLA simulator: route completion (RC), driving score (DS), infractions per km (IpKm). Here, we describe the details of the metrics. We follow the descriptions on the CARLA official website [8].

Route completion (RC) is the percentage of the completed route length relative to the total route length. The route completion of the i -th route is denoted as R_i .

Driving score (DS) is the product between the route completion and the penalty weight. The driving score of i -th route is formulated as follows:

$$DS = R_i P_i, \quad (\text{C.1})$$

where P_i is the penalty weight of the i -th route. The penalty weight P_i starts from 1.0 and is multiplied by a coefficient when the agent commits an infraction. The penalty weight P_i is formulated as follows:

$$P_i = \prod_j p_j^{\text{\#infractions}_j}, \quad (\text{C.2})$$

Appendix C. Details of CARLA evaluation metrics

Table C.1: Infraction and corresponding coefficient

Infraction	Coefficient
Collisions with pedestrians	0.5
Collisions with other vehicles	0.60
Collisions with static elements	0.65
Running a red light	0.70
Running a stop sign	0.80

(a) Infraction with a coefficient

Infraction	Coefficient
Off-road driving	-
Route deviation	-
Agent blocked	-
Route timeout	-

(b) Infraction without a coefficient

where j is the type of an infraction, p_j is the corresponding coefficient of an infraction, and $\#\text{infractions}_j$ is the number of an infraction committed. The types of infractions and corresponding coefficients are shown in Table C.1. Here, a infraction without a coefficient is not counted when computing P_i . Instead of reducing the penalty weight, a infraction without a coefficient affects the computation of the route completion (off-road driving) or terminates the simulation (route deviation, agent blocked, and route timeout). If an agent causes an off-road driving infraction, the distance traveled with off-road driving is not considered when computing a route completion R_i . In addition, if an agent causes a route deviation, agent blocked, or route timeout infraction, the simulation is terminated.

Infractions per km (IpKm) is the total number of infractions divided by the total distance traveled. All the infractions shown in Table C.1 are considered

Appendix C. Details of CARLA evaluation metrics

when computing the infractions per km regardless of whether a infraction has a coefficient or not.

Appendix C. Details of CARLA evaluation metrics

Appendix D

Detailed results of CARLA experiment in Chapter 4

We report details of the experiment results which are provided in Section 4.3.5 and Section 4.3.8. As described in Section C, there are eight types of infractions in the CARLA simulator: collisions with pedestrians (Ped), collisions with other vehicles (Veh), collisions with static elements (Static), running a red light (Red), running a stop sign (Stop), off-road driving (Off), route deviation (Dev), agent blocked (Blocked), route timeout (TO). We report infractions per km for each type of infraction (Ped pKm, Veh pKm, Static pKm, Red pKm, Stop pKm, Off pKm, Dev pKm, Blocked pKm, TO pkm). The number of each infraction is divided by the total distance in kilometers traveled. We also report the total distance traveled (Distance), but Distance here is reported in meters. In Table D.1 and D.2, we provide the mean and standard deviation of the results of the performance comparison on Town05 Short and Town05 Long. Each method is trained and evaluated with five different random seeds in the performance comparison. In Table D.3 and D.4, we provide the mean and standard deviation of the results of the ablation study on Town05 Short and Town05 Long. Each method is trained

Appendix D. Detailed results of CARLA experiment in Chapter 4

and evaluated with three different random seeds in the ablation study.

Table D.1: Performance Comparison in Town05 Short

Method	Ped pKm ↓	Veh pKm ↓	Static pKm ↓	Red pKm ↓	Stop pKm ↓
CIRLS [20]	0.000 ± 0.000	4.438 ± 4.071	20.980 ± 4.660	7.776 ± 2.964	0.000 ± 0.000
RBC [26]	0.000 ± 0.000	8.257 ± 6.972	0.000 ± 0.000	18.526 ± 3.133	0.000 ± 0.000
AIM [62]	0.000 ± 0.000	0.000 ± 0.000	0.309 ± 0.618	1.465 ± 1.830	0.000 ± 0.000
AIM+R [62]	0.000 ± 0.000	0.309 ± 0.618	0.927 ± 0.757	0.610 ± 0.747	0.000 ± 0.000
Transfuser [62]	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	4.409 ± 2.466	0.000 ± 0.000
Transfuser+R [62]	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	4.146 ± 2.575	0.000 ± 0.000
RIANet (Ours)	0.000 ± 0.000	0.283 ± 0.566	0.000 ± 0.000	2.999 ± 0.918	0.000 ± 0.000

Method	Off pKm ↓	Dev pKm ↓	Blocked pKm ↓	TO pKm ↓	Distance (m) ↑
CIRLS [20]	15.493 ± 4.557	27.475 ± 0.868	2.156 ± 2.640	19.824 ± 3.046	18.217 ± 0.570
RBC [26]	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	6.743 ± 0.254	59.408 ± 2.295
AIM [62]	0.765 ± 0.965	0.000 ± 0.000	3.841 ± 3.255	3.823 ± 3.396	51.694 ± 13.436
AIM+R [62]	0.000 ± 0.000	0.000 ± 0.000	3.255 ± 5.013	3.490 ± 3.885	58.185 ± 13.941
Transfuser [62]	0.000 ± 0.000	0.000 ± 0.000	4.638 ± 2.529	3.013 ± 2.246	50.144 ± 5.091
Transfuser+R [62]	0.000 ± 0.000	0.000 ± 0.000	5.046 ± 3.092	1.689 ± 2.090	51.798 ± 6.956
RIANet (Ours)	0.000 ± 0.000	0.000 ± 0.000	0.313 ± 0.626	2.724 ± 1.810	66.384 ± 2.706

Appendix D. Detailed results of CARLA experiment in Chapter 4

Table D.2: Performance Comparison in Town05 Long

Method	Ped pKm ↓	Veh pKm ↓	Static pKm ↓	Red pKm ↓	Stop pKm ↓
CIRLS [20]	0.000 ± 0.000	2.356 ± 1.258	0.895 ± 0.030	2.846 ± 1.000	0.000 ± 0.000
RBC [26]	0.000 ± 0.000	1.547 ± 0.290	0.019 ± 0.039	3.550 ± 0.312	1.278 ± 0.146
AIM [62]	0.000 ± 0.000	0.426 ± 0.241	0.036 ± 0.072	1.194 ± 0.381	1.102 ± 0.243
AIM+R [62]	0.000 ± 0.000	0.301 ± 0.158	0.000 ± 0.000	0.945 ± 0.427	1.020 ± 0.361
Transfuser [62]	0.000 ± 0.000	0.455 ± 0.262	0.016 ± 0.032	1.446 ± 0.210	1.226 ± 0.202
Transfuser+R [62]	0.000 ± 0.000	0.482 ± 0.287	0.000 ± 0.000	1.594 ± 0.345	0.989 ± 0.183
RIANet (Ours)	0.000 ± 0.000	0.308 ± 0.202	0.000 ± 0.000	1.087 ± 0.196	1.178 ± 0.133

Method	Off pKm ↓	Dev pKm ↓	Blocked pKm ↓	TO pKm ↓	Distance (m) ↑
CIRLS [20]	1.971 ± 0.693	6.260 ± 0.549	2.506 ± 0.371	0.184 ± 0.369	111.853 ± 3.872
RBC [26]	0.000 ± 0.000	0.000 ± 0.000	0.255 ± 0.126	0.019 ± 0.038	991.296 ± 102.054
AIM [62]	0.281 ± 0.071	0.017 ± 0.034	0.342 ± 0.288	0.024 ± 0.049	1005.030 ± 156.295
AIM+R [62]	0.000 ± 0.000	0.000 ± 0.000	0.456 ± 0.418	0.032 ± 0.063	961.508 ± 237.002
Transfuser [62]	0.236 ± 0.141	0.000 ± 0.000	0.094 ± 0.107	0.000 ± 0.000	1149.862 ± 92.545
Transfuser+R [62]	0.000 ± 0.000	0.000 ± 0.000	0.145 ± 0.077	0.000 ± 0.000	1127.488 ± 77.393
RIANet (Ours)	0.000 ± 0.000	0.000 ± 0.000	0.035 ± 0.040	0.000 ± 0.000	1188.881 ± 44.585

Table D.3: Ablation Study in Town05 Short

Configure	Ped pKm ↓	Veh pKm ↓	Static pKm ↓	Red pKm ↓	Stop pKm ↓
no-road-graph	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	6.362 ± 2.427	0.000 ± 0.000
no-detection	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	4.700 ± 0.324	0.000 ± 0.000
no-global-image	0.000 ± 0.000	1.049 ± 0.747	0.000 ± 0.000	20.531 ± 2.493	0.000 ± 0.000
no-LiDAR	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	5.546 ± 0.517	0.000 ± 0.000
Default	0.000 ± 0.000	0.283 ± 0.566	0.000 ± 0.000	2.999 ± 0.918	0.000 ± 0.000

Configure	Off pKm ↓	Dev pKm ↓	Blocked pKm ↓	TO pKm ↓	Distance (m) ↑
no-road-graph	0.000 ± 0.000	0.000 ± 0.000	0.517 ± 0.731	0.495 ± 0.701	67.513 ± 2.528
no-detection	0.000 ± 0.000	0.000 ± 0.000	2.187 ± 1.548	1.567 ± 0.108	64.154 ± 4.599
no-global-image	0.000 ± 0.000	0.000 ± 0.000	1.121 ± 1.586	1.049 ± 0.747	66.137 ± 4.831
no-LiDAR	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	4.111 ± 1.635	65.879 ± 2.595
Default	0.000 ± 0.000	0.000 ± 0.000	0.313 ± 0.626	2.724 ± 1.810	66.384 ± 2.706

Table D.4: Ablation Study in Town05 Long

Configure	Ped pKm ↓	Veh pKm ↓	Static pKm ↓	Red pKm ↓	Stop pKm ↓
no-road-graph	0.000 ± 0.000	0.515 ± 0.167	0.000 ± 0.000	1.382 ± 0.230	1.138 ± 0.176
no-detection	0.000 ± 0.000	0.490 ± 0.112	0.000 ± 0.000	1.226 ± 0.058	1.253 ± 0.129
no-global-image	0.000 ± 0.000	0.434 ± 0.177	0.000 ± 0.000	2.854 ± 0.379	1.030 ± 0.279
no-LiDAR	0.000 ± 0.000	0.268 ± 0.163	0.000 ± 0.000	1.481 ± 0.235	1.196 ± 0.075
Default	0.000 ± 0.000	0.308 ± 0.202	0.000 ± 0.000	1.087 ± 0.196	1.178 ± 0.133

Configure	Off pKm ↓	Dev pKm ↓	Blocked pKm ↓	TO pKm ↓	Distance (m) ↑
no-road-graph	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	1230.079 ± 0.000
no-detection	0.000 ± 0.000	0.000 ± 0.000	0.028 ± 0.039	0.000 ± 0.000	1222.604 ± 10.571
no-global-image	0.000 ± 0.000	0.000 ± 0.000	0.213 ± 0.036	0.032 ± 0.046	1092.108 ± 43.847
no-LiDAR	0.000 ± 0.000	0.000 ± 0.000	0.032 ± 0.045	0.000 ± 0.000	1168.486 ± 87.105
Default	0.000 ± 0.000	0.000 ± 0.000	0.035 ± 0.040	0.000 ± 0.000	1188.881 ± 44.585

Appendix E

Detailed results of CARLA experiment in Chapter 5

We report details of the experiment results which are provided in Section 5.3.4. We use the same metrics which are explained in Appendix D: Ped pKm, Veh pKm, Static pKm, Red pKm, Stop pKm, Off pKm, Dev pKm, Blocked pKm, TO pkm, and Distance. In Table E.1 and E.2, we provide the mean and standard deviation of the results of the performance comparison on Town05 Short and Town05 Long. Each method is trained and evaluated with five different random seeds in the performance comparison.

Appendix E. Detailed results of CARLA experiment in Chapter 5

Table E.1: Performance consistency of RIANet++ under road change conditions (CARLA Town05 Short)

Configure	Road Changes	Ped pKm ↓	Veh pKm ↓	Static pKm ↓	Red pKm ↓	Stop pKm ↓
RIANet	no	0.000 ± 0.000	0.283 ± 0.566	0.000 ± 0.000	2.999 ± 0.918	0.000 ± 0.000
	yes	0.000 ± 0.000	3.058 ± 1.505	0.316 ± 0.631	6.311 ± 2.201	0.000 ± 0.000
RIANet++	no	0.000 ± 0.000	0.301 ± 0.602	0.000 ± 0.000	5.260 ± 1.492	0.000 ± 0.000
	yes	0.000 ± 0.000	0.583 ± 0.714	0.318 ± 0.635	5.302 ± 2.096	0.000 ± 0.000

Configure	Road Changes	Off pKm ↓	Dev pKm ↓	Blocked pKm ↓	TO pKm ↓	Distance (m) ↑
RIANet	no	0.000 ± 0.000	0.000 ± 0.000	0.313 ± 0.626	2.724 ± 1.810	66.384 ± 2.706
	yes	1.536 ± 1.396	0.342 ± 0.684	0.000 ± 0.000	4.447 ± 1.411	63.903 ± 3.504
RIANet++	no	0.000 ± 0.000	0.000 ± 0.000	0.300 ± 0.600	0.875 ± 0.715	68.612 ± 1.879
	yes	0.601 ± 0.738	0.000 ± 0.000	0.000 ± 0.000	2.118 ± 0.827	67.150 ± 2.801

Table E.2: Performance consistency of RIANet++ under road change conditions (CARLA Town05 Long)

Configure	Road Changes	Ped pKm ↓	Veh pKm ↓	Static pKm ↓	Red pKm ↓	Stop pKm ↓
RIANet	no	0.000 ± 0.000	0.308 ± 0.202	0.000 ± 0.000	1.087 ± 0.196	1.178 ± 0.133
	yes	0.000 ± 0.000	0.960 ± 0.520	0.085 ± 0.083	1.347 ± 0.237	1.268 ± 0.284
RIANet++	no	0.000 ± 0.000	0.293 ± 0.065	0.000 ± 0.000	1.301 ± 0.178	1.138 ± 0.230
	yes	0.000 ± 0.000	0.512 ± 0.235	0.034 ± 0.041	1.279 ± 0.298	1.129 ± 0.126

Configure	Road Changes	Off pKm ↓	Dev pKm ↓	Blocked pKm ↓	TO pKm ↓	Distance (m) ↑
RIANet	no	0.000 ± 0.000	0.000 ± 0.000	0.035 ± 0.040	0.000 ± 0.000	1188.881 ± 44.585
	yes	0.129 ± 0.177	0.047 ± 0.058	0.257 ± 0.085	0.042 ± 0.051	946.302 ± 90.000
RIANet++	no	0.293 ± 0.083	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	1230.079 ± 0.000
	yes	0.305 ± 0.078	0.000 ± 0.000	0.053 ± 0.071	0.000 ± 0.000	1192.082 ± 47.786

Appendix E. Detailed results of CARLA experiment in Chapter 5

Appendix F

Examples of FMTC real-world dataset scenarios

We display examples of each FMTC real-world dataset scenario described in Section 5.3.1 and Figure 5.6. Figure F.1 shows examples of the FMTC dataset in scenario 1 (Go straight). Figure F.2 shows examples of the FMTC dataset in scenario 2 (Turn right). Figure F.3 shows examples of the FMTC dataset in scenario 3 (Turn left). Figure F.4 shows examples of the FMTC dataset in scenario 4 (Stop behind another vehicle).

Appendix F. Examples of FMTC real-world dataset scenarios









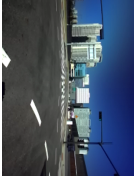











Time	$t = 1.5s$	$t = 3.0s$	$t = 4.5s$	$t = 6.0s$	$t = 7.5s$
Scenario 1-1					
Time	$t = 9.0s$	$t = 10.5s$	$t = 12.0s$	$t = 13.5s$	$t = 15.0s$
Scenario 1-1					
Time	$t = 1.5s$	$t = 3.0s$	$t = 4.5s$	$t = 6.0s$	$t = 7.5s$
Scenario 1-2					
Time	$t = 9.0s$	$t = 10.5s$	$t = 12.0s$	$t = 13.5s$	$t = 15.0s$
Scenario 1-2					

Figure F.1: FMTC dataset example (Scenario 1: Go straight).

Appendix F. Examples of FMTC real-world dataset scenarios









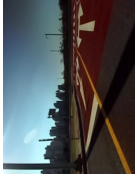
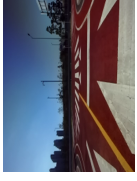










Time	$t = 1.5s$	$t = 3.0s$	$t = 4.5s$	$t = 6.0s$	$t = 7.5s$
Scenario 2-1					
Time	$t = 9.0s$	$t = 10.5s$	$t = 12.0s$	$t = 13.5s$	$t = 15.0s$
Scenario 2-1					
Time	$t = 1.5s$	$t = 3.0s$	$t = 4.5s$	$t = 6.0s$	$t = 7.5s$
Scenario 2-2					
Time	$t = 9.0s$	$t = 10.5s$	$t = 12.0s$	$t = 13.5s$	$t = 15.0s$
Scenario 2-2					

Figure F.2: FMTC dataset example (Scenario 2: Turn right).

Appendix F. Examples of FMTC real-world dataset scenarios





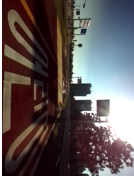
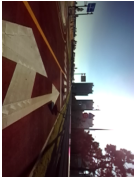
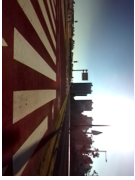
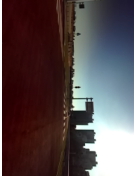
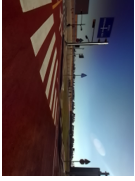
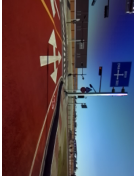









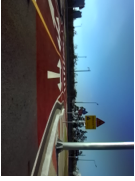
Time	$t = 1.5s$	$t = 3.0s$	$t = 4.5s$	$t = 6.0s$	$t = 7.5s$
Scenario 3-1					
Time	$t = 9.0s$	$t = 10.5s$	$t = 12.0s$	$t = 13.5s$	$t = 15.0s$
Scenario 3-1					
Time	$t = 1.5s$	$t = 3.0s$	$t = 4.5s$	$t = 6.0s$	$t = 7.5s$
Scenario 3-2					
Time	$t = 9.0s$	$t = 10.5s$	$t = 12.0s$	$t = 13.5s$	$t = 15.0s$
Scenario 3-2					

Figure F.3: FMTC dataset example (Scenario 3: Turn left).
















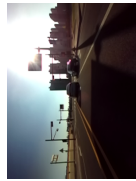




Time	$t = 1.5s$	$t = 3.0s$	$t = 4.5s$	$t = 6.0s$	$t = 7.5s$
Scenario 4-1					
Time	$t = 9.0s$	$t = 10.5s$	$t = 12.0s$	$t = 13.5s$	$t = 15.0s$
Scenario 4-1					
Time	$t = 1.5s$	$t = 3.0s$	$t = 4.5s$	$t = 6.0s$	$t = 7.5s$
Scenario 4-2					
Time	$t = 9.0s$	$t = 10.5s$	$t = 12.0s$	$t = 13.5s$	$t = 15.0s$
Scenario 4-2					

Figure F.4: FMTC dataset example (Scenario 4: Stop behind another vehicle).

Appendix F. Examples of FMTC real-world dataset scenarios

Appendix G

Effect of localization error on road change detection accuracy

We investigate the effect of localization error on road change detection accuracy. Figure G.1 shows the detection module’s average precision according to the localization error in the CARLA environment. The average precision is calculated as the mean of the average precision for all types of road changes. The experiments are conducted in Town05 Short (Figure G.1(a) and Figure G.1(c)) and Town05 Long (Figure G.1(b) and Figure G.1(d)) environments. To show the effect of localization error, we add zero-mean Gaussian noise to the estimated vehicle positions, which are used to synthesize the query road graph in Section 5.2.2. Gaussian noise is added to each position value (Figure G.1(a) and Figure G.1(b)) and the yaw value (Figure G.1(c) and Figure G.1(d)), respectively. We change the standard deviation of the noise to observe how the average precision of the detection module is degraded according to the noise level.

In each experiment, RD stands for ResNet discriminator [49], and DML stands for deep metric learning [42]. GM-TOPO and GM-APLS stand for the graph matching methods, which use TOPO [6] and APLS [77] as similarity scores. Also,

Appendix G. Effect of localization error on road change detection accuracy

SM-mAP and SM-mIoU stand for the semantic matching methods, which use mAP and mIoU as similarity scores. In the experiments, the semantic matching method shows the highest performance when there are zero noises. However, as the noise level increases, the semantic matching method shows a significant decrease in performance. On the other hand, the graph matching method is not significantly affected by the noise level. The results show that the graph matching method is more robust to localization errors than the semantic matching method.

Appendix G. Effect of localization error on road change detection accuracy

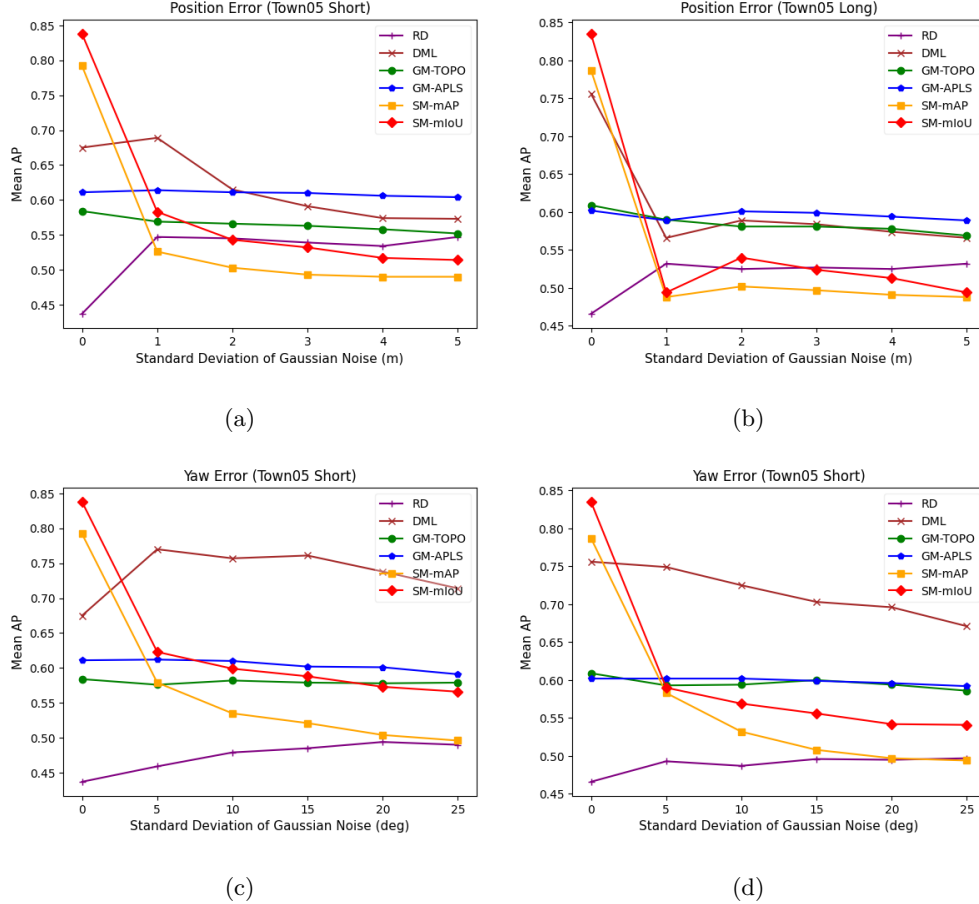


Figure G.1: Effect of localization error on road change detection accuracy. (a) Effect of position error in Town05 Short. (b) Effect of position error in Town05 Long. (c) Effect of yaw error in Town05 Short. (d) Effect of yaw error in Town05 Long.

Appendix G. Effect of localization error on road change detection accuracy

Appendix H

Analysis of detection accuracy according to the degree of road changes

We conduct a quantitative analysis of the detection accuracy according to the degree of road changes. Figure H.1 shows the detection module’s average precision according to the degree of road changes in the CARLA environment. The experiments are conducted in Town05 Short (Figure H.1(a) and Figure H.1(c)) and Town05 Long (Figure H.1(b) and Figure H.1(d)) environments. Figure H.1(a) and Figure H.1(c) show how the detection accuracy is changed according to the degree of lane deviation. Likewise, Figure H.1(b) and Figure H.1(d) show how the detection accuracy is changed according to the degree of lane width change.

In each experiment, RD stands for ResNet discriminator [49], and DML stands for deep metric learning [42]. GM-TOPO and GM-APLS stand for the graph matching methods, which use TOPO [6] and APLS [77] as similarity scores. Also, SM-mAP and SM-mIoU stand for the semantic matching methods, which use mAP and mIoU as similarity scores. For both graph matching and semantic

Appendix H. Analysis of detection accuracy according to the degree of road changes

matching methods, the average precision increases as the degree of lane deviation increases. As the lane deviation increases, the difference between the accurate and changed road graphs increases as well. For this reason, the accuracy of the detection module that distinguishes the two differences between them seems to increase. In lane width change cases, the accuracy of the graph matching method increases as the degree of lane width change increases. However, the accuracy of the semantic matching method does not show increases when the degree of the width change becomes large. From this result, we can conclude that an increase in the degree of road change generally increases the discriminating performance of the detection module, but does not always increase depending on the type of road change.

Appendix H. Analysis of detection accuracy according to the degree of road changes

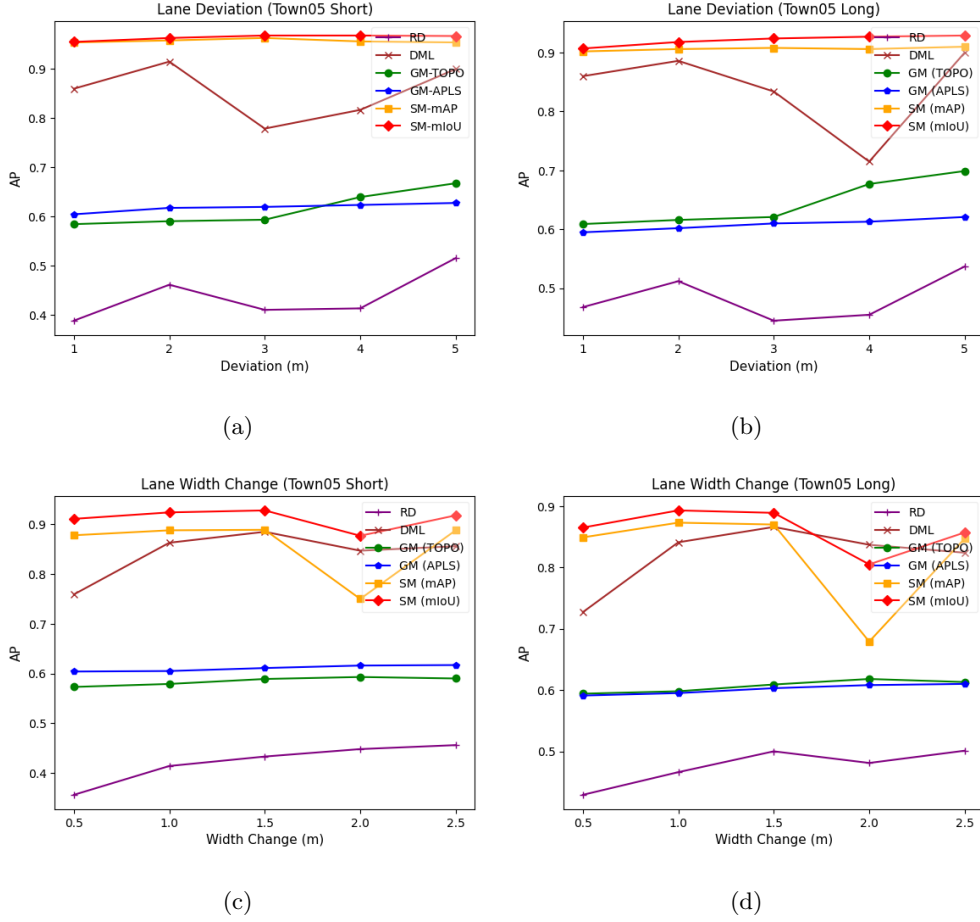


Figure H.1: Analysis of detection accuracy according to the degree of road changes. (a) Effect of lane deviation in Town05 Short. (b) Effect of lane deviation in Town05 Long. (c) Effect of lane width change in Town05 Short. (d) Effect of lane width change in Town05 Long.

Appendix H. Analysis of detection accuracy according to the degree of road changes

Appendix I

Ablation study about performance consistency in CARLA environment

For ablation studies, we compare the performance of Road-GNN (Chapter 3), RIANet (Chapter 4), and RIANet++ (Chapter 5) in the CARLA simulator. Similar to the experience in Section 5.3.4, we conduct each method both with and without road change conditions. In Road-GNN, the controller follows the pre-defined path as like RIANet and RIANet++. However, Road-GNN only takes the road graph and vehicle speed as inputs to the controller and does not use image and LiDAR sensor data. Instead of using a vanilla Road-GNN structure, we use a network model which takes only a single time-step data. The structure is similar to the network model explained in Section 4.3.8, which ignores other sensor data and takes the road graph and vehicle speed as inputs. For RIANet and RIANet++, we use the same results described in Section 5.3.4.

In Table I.1 and Table I.2, we show the experiment results conducted in Town05 Short and Town05 Long, respectively. In Town05 Short (Table I.1), RIANet++

Appendix I. Ablation study about performance consistency in CARLA environment

shows the lowest performance degradation for all the metrics. In Town05 Long (Table I.2), Road-GNN shows the lowest performance degradation for RC and IpKm. However, compared to the other methods, Road-GNN also shows the lowest performance for all the metrics. From this point of view, it can be said that RIANet++ is the most effective controller for the CARLA environment under road change conditions.

**Appendix I. Ablation study about performance consistency in
CARLA environment**

Table I.1: Ablation study in Town05 Short

Configure	DS \uparrow	RC \uparrow	IpKm \downarrow
Road-GNN	33.627 ± 2.960	54.033 ± 5.046	44.416 ± 7.577
RIANet	87.469 ± 2.363	93.881 ± 3.827	6.319 ± 1.419
RIANet++	86.930 ± 3.960	97.032 ± 2.657	6.737 ± 1.616

(a) No road change condition

Configure	DS \uparrow	RC \uparrow	IpKm \downarrow
Road-GNN	25.045 ± 8.727	44.435 ± 10.415	64.377 ± 15.261
RIANet	74.012 ± 3.902	90.372 ± 4.956	16.009 ± 2.722
RIANet++	83.224 ± 3.547	94.964 ± 3.961	8.921 ± 2.287

(b) Road change condition

Configure	DS \uparrow	RC \uparrow	IpKm \downarrow
Road-GNN	-8.582	-9.598	-19.961
RIANet	-13.457	-3.509	-9.69
RIANet++	-3.706	-2.068	-2.184

(c) Performance drop

Appendix I. Ablation study about performance consistency in CARLA environment

Table I.2: Ablation study in Town05 Long

Configure	DS \uparrow	RC \uparrow	IpKm \downarrow
Road-GNN	7.965 ± 1.292	19.208 ± 6.554	12.857 ± 4.151
RIANet	44.719 ± 2.513	96.934 ± 2.927	2.624 ± 0.163
RIANet++	41.714 ± 3.988	100.000 ± 0.000	3.024 ± 0.227

(a) No road change condition

Configure	DS \uparrow	RC \uparrow	IpKm \downarrow
Road-GNN	7.139 ± 0.627	18.571 ± 2.509	13.381 ± 1.289
RIANet	27.724 ± 7.514	79.653 ± 6.314	4.135 ± 1.083
RIANet++	39.726 ± 7.336	96.152 ± 4.728	3.312 ± 0.688

(b) Road change condition

Configure	DS \uparrow	RC \uparrow	IpKm \downarrow
Road-GNN	-0.826	-0.637	-0.524
RIANet	-16.995	-17.281	-1.511
RIANet++	-1.988	-3.848	-0.288

(c) Performance drop

Bibliography

- [1] Association for Standardization of Automation and Measuring Systems. Opendrive. URL <https://www.asam.net/standards/detail/opendrive/>.
- [2] Ankan Bansal, Karan Sikka, Gaurav Sharma, Rama Chellappa, and Ajay Divakaran. Zero-shot object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 384–400, 2018.
- [3] Aseem Behl, Kashyap Chitta, Aditya Prakash, Eshed Ohn-Bar, and Andreas Geiger. Label efficient visual abstractions for autonomous driving. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2338–2345, 2020.
- [4] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4009–4018, 2021.
- [5] Raunak P Bhattacharyya, Derek J Phillips, Blake Wulfe, Jeremy Morton, Alex Kuefler, and Mykel J Kochenderfer. Multi-agent imitation learning for driving simulation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1534–1539. IEEE, 2018.
- [6] James Biagioni and Jakob Eriksson. Inferring road maps from global posi-

Bibliography

- tioning system traces: Survey and comparative evaluation. *Transportation research record*, 2291(1):61–71, 2012.
- [7] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11621–11631, 2020.
- [8] CARLA. Carla autonomous driving leaderboard. URL <https://leaderboard.carla.org/>.
- [9] Manfredo Perdigao do Carmo. *Differential geometry of curves and surfaces*. Dover Publications, Inc., Mineola, New York, revised and updated 2nd edition edition, 2016.
- [10] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8748–8757, 2019.
- [11] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on Robot Learning (CoRL)*, 2019.
- [12] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.
- [13] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam.

- Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.
- [14] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 801–818, 2018.
- [15] Kyunghoon Cho, Timothy Ha, Gunmin Lee, and Songhwai Oh. Deep predictive autonomous driving using multi-agent joint trajectory prediction and traffic rules. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2076–2081, 2019.
- [16] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.
- [17] Sungjoon Choi, Kyungjae Lee, Sunbin Lim, and Songhwai Oh. Uncertainty-aware learning from demonstration using mixture density networks with sampling-free variance modeling. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6915–6922, 2018.
- [18] Sungjoon Choi, Kyungjae Lee, and Songhwai Oh. Robust learning from demonstrations with mixed qualities using leveraged gaussian processes. *IEEE Transactions on Robotics*, 35(3):564–576, 2019.
- [19] Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *IEEE international conference on robotics and automation (ICRA)*, pages 4693–4700, 2018.

Bibliography

- [20] Felipe Codevilla, Eder Santana, Antonio M López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9329–9338, 2019.
- [21] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.
- [22] Wendong Ding, Shenhua Hou, Hang Gao, Guowei Wan, and Shiyu Song. Lidar inertial odometry aided robust lidar localization system in changing city scenes. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4322–4328, 2020.
- [23] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, and James Diebel. Practical search techniques in path planning for autonomous driving. *AAAI Workshop - Technical Report*, 2008.
- [24] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning (CoRL)*, pages 1–16, 2017.
- [25] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of Conference on Robot Learning (CoRL)*, pages 1–16, 2017.
- [26] ERDOS. Pylot. URL <https://github.com/erdos-project/pylot>.
- [27] Open Source Robotics Foundation. Demo of prius in ros/gazebo. https://github.com/osrf/car_demo, 2019.

- [28] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2002–2011, 2018.
- [29] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11525–11533, 2020.
- [30] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3828–3838, 2019.
- [31] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems (NeurIPS)*, 27, 2014.
- [32] Timothy Ha, Gunmin Lee, Dohyeong Kim, and Songhwai Oh. Road graphical neural networks for autonomous roundabout driving. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 162–167, 2021.
- [33] Timothy Ha, Jeongwoo Oh, Hojun Chung, Gunmin Lee, and Songhwai Oh. Rianet: Road graph and image attention network for urban autonomous driving. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.
- [34] John Halkias and James Colyar. Ngsim interstate 80 freeway dataset, 2006.

Bibliography

- [35] Irtiza Hasan, Shengcai Liao, Jinpeng Li, Saad Ullah Akram, and Ling Shao. Generalizable pedestrian detection: The elephant in the room. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11328–11337, 2021.
- [36] Jeffrey Hawke, Richard Shen, Corina Gurau, Siddharth Sharma, Daniele Reda, Nikolay Nikolov, Przemysław Mazur, Sean Micklethwaite, Nicolas Griffiths, Amar Shah, et al. Urban driving with conditional imitation learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 251–257, 2020.
- [37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [38] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017.
- [39] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4918–4927, 2019.
- [40] Songtao He, Favyen Bastani, Satvat Jagwani, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, Mohamed M Elshrif, Samuel Madden, and Mohammad Amin Sadeghi. Sat2graph: Road graph extraction through graph-tensor encoding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 51–67, 2020.
- [41] Mikael Henaff, Alfredo Canziani, and Yann LeCun. Model-predictive policy learning with uncertainty regularization for driving in dense traffic. In *International Conference on Learning Representations (ICLR)*, 2019.

- [42] Minhyeok Heo, Jiwon Kim, and Sujung Kim. Hd map change detection with cross-domain deep metric learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10218–10224. IEEE, 2020.
- [43] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 29, 2016.
- [44] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [45] David Isele, Reza Rahimi, Akansel Cosgun, Kaushik Subramanian, and Kikuo Fujimura. Navigating occluded intersections with autonomous vehicles using deep reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2034–2039, 2018.
- [46] Ajay Jain, Sergio Casas, Renjie Liao, Yuwen Xiong, Song Feng, Sean Segal, and Raquel Urtasun. Discrete residual flow for probabilistic pedestrian behavior prediction. In *Conference on Robot Learning (CoRL)*, pages 407–419, 2020.
- [47] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations (ICLR)*, 2017.
- [48] Ilya Kostrikov. Pytorch implementations of reinforcement learning algorithms. <https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail>, 2018.
- [49] John Lambert and James Hays. Trust, but verify: Cross-modality fusion for hd map change detection. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

Bibliography

- [50] Gunmin Lee, Dohyeong Kim, Wooseok Oh, Kyungjae Lee, and Songhwai Oh. Mixgail: Autonomous driving using demonstrations with mixed qualities. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5425–5430, 2020.
- [51] Buyu Li, Wanli Ouyang, Lu Sheng, Xingyu Zeng, and Xiaogang Wang. Gs3d: An efficient 3d object detection framework for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [52] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. In *European Conference on Computer Vision (ECCV)*, pages 541–556, 2020.
- [53] Lizhe Liu, Xiaohao Chen, Siyu Zhu, and Ping Tan. Condlanenet: a top-to-down lane detection framework based on conditional convolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3773–3782, 2021.
- [54] Yiwei Lyu, Chiyu Dong, and John M Dolan. Fg-gmm-based interactive behavior estimation for autonomous driving vehicles in ramp merging control. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1250–1255, 2020.
- [55] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of International Conference on Machine Learning (ICML) Workshops*, 2013.
- [56] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dhharshan Kumaran, Daan Wierstra, Shane Legg,

- and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.
- [57] NAVER. Naver maps, 2020. URL <https://map.naver.com/>.
- [58] Kentaro Nishi and Masamichi Shimosaka. Fine-grained driving behavior prediction via context-aware multi-task inverse reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2281–2287, 2020.
- [59] Ippei Nishitani, Hao Yang, Rui Guo, Shalini Keshavamurthy, and Kentaro Oguchi. Deep merging: Vehicle merging controller based on deep reinforcement learning with embedding network. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 216–221, 2020.
- [60] Błażej Osiński, Adam Jakubowski, Paweł Ziecina, Piotr Miłoś, Christopher Galias, Silviu Homoceanu, and Henryk Michalewski. Simulation-based reinforcement learning for real-world autonomous driving. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6411–6418, 2020.
- [61] David Pannen, Martin Liebner, and Wolfram Burgard. Hd map change detection with a boosted particle filter. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2561–2567, 2019.
- [62] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. Multi-modal fusion transformer for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7077–7087, 2021.
- [63] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings*

Bibliography

- of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017.
- [64] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.
- [65] Lennart Reiher, Bastian Lampe, and Lutz Eckstein. A sim2real deep learning approach for the transformation of images from multiple vehicle-mounted cameras to a semantically segmented image in bird’s eye view. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–7, 2020.
- [66] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 28:91–99, 2015.
- [67] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. Pre-cog: Prediction conditioned on goals in visual multi-agent settings. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2821–2830, 2019.
- [68] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.
- [69] Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 627–635, 2011.

- [70] Dhruv Mauria Saxena, Sangjae Bae, Alireza Nakhaei, Kikuo Fujimura, and Maxim Likhachev. Driving in dense traffic with model-free reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5385–5392, 2020.
- [71] Edward Schmerling, Karen Leung, Wolf Vollprecht, and Marco Pavone. Multimodal probabilistic model-based planning for human-robot interaction. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3399–3406, 2018.
- [72] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, pages 1889–1897, 2015.
- [73] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [74] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10529–10538, 2020.
- [75] Ibrahim Sobh, Loay Amin, Sherif Abdelkarim, Khaled Elmadawy, Mahmoud Saeed, Omar Abdeltawab, Mostafa Gamal, and Ahmad El Sallab. End-to-end multi-modal sensors fusion system for urban automated driving. In *Advances in Neural Information Processing Systems (NeurIPS) Workshops*, 2018.
- [76] Stanford Artificial Intelligence Laboratory et al. Robotic operating system. URL <https://www.ros.org>.

Bibliography

- [77] Adam Van Etten, Dave Lindenbaum, and Todd M Bacastow. Spacenet: A remote sensing dataset and challenge series. *arXiv preprint arXiv:1807.01232*, 2018.
- [78] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.
- [79] Sourabh Vora, Alex H Lang, Bassam Helou, and Oscar Beijbom. Point-painting: Sequential fusion for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4604–4612, 2020.
- [80] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip Yu. Generalizing to unseen domains: A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering*, 2022.
- [81] Yue Wang, Vitor Campagnolo Guizilini, Tianyuan Zhang, Yilun Wang, Hang Zhao, and Justin Solomon. Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In *Proceedings of the Conference on Robot Learning (CoRL)*, volume 164, pages 180–191, 2022.
- [82] Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. Optimal trajectory generation for dynamic street scenarios in a frenet frame. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 987–993, 2010.
- [83] Yi Xiao, Felipe Codevilla, Akhil Gurram, Onay Urfalioglu, and Antonio M López. Multimodal end-to-end autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

- [84] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 244–253, 2018.
- [85] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2403–2412, 2018.
- [86] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In *European Conference on Computer Vision (ECCV)*, pages 173–190, 2020.
- [87] Yu Zhang, Huiyan Chen, Steven L Waslander, Jianwei Gong, Guangming Xiong, Tian Yang, and Kai Liu. Hybrid trajectory planning for autonomous driving in highly constrained environments. *IEEE Access*, 6:32800–32819, 2018.
- [88] Lin Zhao, Hui Zhou, Xinge Zhu, Xiao Song, Hongsheng Li, and Wenbing Tao. Lif-seg: Lidar and camera image fusion for 3d lidar semantic segmentation. *arXiv preprint arXiv:2108.07511*, 2021.
- [89] Wu Zheng, Weiliang Tang, Li Jiang, and Chi-Wing Fu. Se-ssd: Self-ensembling single-stage object detector from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14494–14503, 2021.

Bibliography

초 록

본 학위 논문은 학습 기반의 자율주행 네비게이션 문제에 대해 다룬다. 자율주행 기술을 현실에 적용하기 위해서는 여러 도로환경에서 동작할 수 있는 범용적인 컨트롤러를 설계하는 것이 필요하다. 기존의 방법론은 사례 중심적인 방식을 사용하였다. 이는 각 사례별로 특정 도로 환경을 가정하고 컨트롤러를 설계하는 것을 의미한다. 이러한 사례 중심적인 방식으로 만들어진 컨트롤러는 이전에 다루어지지 않은 도로 환경에 대해서는 동작하지 않을 수 있다. 또한 도로 환경은 그 종류가 다양하기 때문에, 기존의 방법론으로 범용적인 자율주행 컨트롤러를 개발하는 것은 많은 개발시간과 비용을 요구하게 된다. 따라서 범용적인 자율주행 기술 개발을 위해서는 도로 환경에 구애되지 않는 새로운 종류의 방법론이 요구된다.

본 학위 논문에서는 특정 도로 환경을 가정하지 않고, 범용적으로 동작하는 컨트롤러를 설계하는 것을 목표로 한다. 이를 위해서 본 학위 논문은 학습 기반의 컨트롤러를 이용하고자 하며, 다양한 환경에서의 학습을 통해 범용적인 주행 성능을 높이하고자 한다. 특히 다양한 도로 환경에 대한 특성을 파악할 수 있도록 학습시키기 위해서, 본 학위 논문은 도로 환경에 대한 정보를 컨트롤러에 입력으로 줄 것을 제안한다. 도로 환경 정보를 그래프 형태를 이용하여 인코딩 되며, 컨트롤러는 인코딩된 도로 그래프 정보를 학습하여 도로주행 성능을 높이게 된다. 본 학위 논문에서는 도로 그래프 활용과 관련하여 크게 세 가지 주제를 다룬다.

첫째로, 본 학위 논문은 도로 환경 정보를 그래프 기반으로 인코딩 하는 방법론에 대해 다룬다. 일반적으로 도로 위 자동차의 움직임은 도로 모양에 영향을 받는다. 따라서 자동차의 위치 정보에 대해 파악할 때는 자동차가 위치한 도로의 형태를 함께 고려해주어야 한다. 이를 위해 본 학위 논문은 도로를 그래프 형태로 나타내고, 자동차의 위치 정보를 도로 그래프와 연관지어 나타내는 방법을 제안한다. 본 학위 논문은 또한 이러한 도로 그래프를 처리할 수 있는 새로운 네트워크 구조를 제안한다. 실험 결과들은 도로 그래프를 이용하여 상태를 인코딩하는 방식이 컨트롤러의 도로 주행 환경 일반화에 도움이 된다는 사실을 증명한다.

둘째로, 본 학위 논문은 도로 환경 정보와 기타 센서 데이터를 융합하는 방법론에

대해 다룬다. 도로 그래프는 도로의 형태와 주변 차량의 위치를 나타낼 수 있지만, 실제 도시 환경에서 도로를 주행할 때 필요한 다른 정보들을 나타내지는 못한다. 예를 들어 도로 표지판이나 교통 신호와 같은 정보들은 도로 그래프로 표현하기 어렵다. 이를 해결하기 위해, 본 학위 논문에서는 도로 그래프 뿐만이 아니라 이미지, LiDAR와 같은 기타 센서 데이터를 조합하여 함께 입력으로 사용할 것을 제안한다. 도로 그래프만으로는 얻을 수 없던 정보들은 다른 센서 데이터 정보를 통해 보완될 수 있다. 본 학위 논문에서는 또한 도로 그래프와 다양한 센서 데이터를 융합할 수 있는 새로운 네트워크 구조에 대해 제안한다. 제안된 네트워크 구조를 통해, 컨트롤러는 복잡한 도로 환경에서도 자율주행을 성공한다. 실험 결과들은 제안된 융합 기반의 방식이 도로 환경 상태를 파악하는데 도움을 준다는 사실을 증명한다.

마지막으로, 본 학위 논문은 도로 그래프에 발생할 수 있는 오류를 탐지하여, 이로 인한 자율주행 성능 저하를 방지할 수 있는 방법론에 대해 다룬다. 도로 그래프 기반의 컨트롤러는 도로 그래프 데이터베이스를 사전에 미리 준비할 것을 요구한다. 하지만 도로는 도로 공사 등의 이유로 계속해서 형태가 변형될 수가 있다. 이러한 변화가 도로 그래프 데이터베이스에 반영되지 않는다면, 컨트롤러는 잘못된 도로 정보를 입력으로 받게 된다. 그렇기에 도로 그래프의 느린 업데이트는 컨트롤러의 성능 저하를 유발할 수 있고, 오류 탐지 기술은 이러한 성능 저하를 방지하기 위해 필요하다. 본 학위 논문에서는 이를 위해 도로 그래프 오류를 탐지할 수 있는 방법론에 대해 제안한다. 먼저, 도로의 변화로 인해 발생할 수 있는 오류에 대해 정의하고, 이러한 오류를 탐지할 수 있는 도로 그래프 변화 탐지 모듈을 제안한다. 실험 결과들은 도로 그래프 변화 탐지 기술이 실제 자율주행 컨트롤러의 성능 향상에 이용될 수 있음을 보인다.

주요어: 자율 주행, 도로 그래프, 강화 학습, 모방 학습, 상태 표현, 네비게이션, 이미지 처리, 객체 인식

학 번: 2017-20972

감사의 글

6년간의 대학원 생활을 마무리하며 대학원에서의 저의 삶은 어떠했는지 되돌아보게 되었습니다. 돌이켜보면 훌륭하신 교수님과 친절한 선배님들, 그리고 착한 동기들과 좋은 후배들 덕분에 나름대로 나쁘지 않고 좋은 대학원 생활을 보낼 수 있었다는 생각이 듭니다. 하지만 한편으로는 반대로 제가 과연 친절한 선배였는지, 착한 동기였는지, 그리고 좋은 후배였는지에 대해서는 의문이 들 때가 많습니다. 하지만 그렇기에 그만큼, 이런 부족한 저를 아껴주고 따뜻하게 대해준 연구실 내의 사람들, 그리고 제 주변에서 저를 지지해준 사람들이 있었기에 제가 대학원을 무사히 마칠 수 있었다는 생각이 듭니다. 연구실 안밖에서의, 그리고 주변에서의 격려와 도움이 없었다라면, 지금의 저의 모습은 결코 없었을 것입니다. 그러한 분들 덕분에 무사히 졸업 논문까지 작성할 수 있었고, 대학원 생활을 마무리 할 수 있었던 것 같습니다.

구체적으로는 먼저 저의 지도교수님이신 오성희 교수님께 감사드리고 싶습니다. 지금 돌아보면, 처음 연구실에 들어왔을 때만 하더라도 제가 부족했던 점이 많았다는 생각이 듭니다. 하지만 이런 부족한 제자를 항상 감내해주시고, 좋은 방향으로 이끌어주신 교수님 덕분에 한 사람의 연구자로 성장할 수 있었던 것 같습니다. 교수님 연구실에 들어올 수 있었고, 6년간 교수님의 지도를 받을 수 있었다는 사실이, 저에게 있어서는 무척이나 다행스러운 일이 아니었나 싶습니다. 그렇기에 이 자리를 빌어 교수님께 다시 한 번 감사의 말씀을 드리고 싶습니다. 두번째로 저의 논문의 위원장을 맡아주신 최진영 교수님께도 감사드리고 싶습니다. 신입생 시절, 세미나에서 교수님의 열정적인 모습을 볼때마다, 저런 연구자가 되고 싶다는 생각을 많이 하게 되었던 것 같습니다. 그러한 분께 논문심사를 받을 수 있어서 영광이었고, 다시 한 번 논문 심사를 맡아주심에 감사드립니다. 예심과 초심에서 논문 심사위원을 맡아주신 조남익 교수님께도 감사드리고 싶습니다. 예심 때 코로나로 편찮으셨을텐데, 그런 와중에도 논문 심사를 맡아주셔서 감사드립니다. 교수님의 친절하시면서도 통찰력 있으신 코멘트는 논문 작성에 많은 도움이 되었습니다. 또한 양인순 교수님께도 감사드리고 싶습니다. 제가 신입생 시절, 교수님께서 처음 부임하셔서 제가 수업을 듣게 되었는데, 그 때 많은 것들을 배우게 된 기억이 떠오릅니다. 이번 논문 심사에서도

큰 도움을 주신 교수님께 다시금 감사드리고 싶습니다.

이외에도 연구실의 많은 선배님들께도 감사드리고 싶습니다. 제가 연구실 생활을 무사히 마무리 할 수 있었던 것은, 연구실 선배분들의 도움이 없었으면 결코 불가능한 일이었을 것입니다. 가장 먼저, 졸업 논문 심사에 심사위원까지 맡아주시며, 많은 도움을 주셨던 성준 형께 감사드리고 싶습니다. 항상 연구에 열정적이신 모습에 존경스러운 점이 많다고 생각했고, 후배인 입장에서 연구자로서의 자세에 대해 많이 배울 수 있었습니다. 또한, 언제나 성실하시고 졸업후에도 연구실에 좋은 영향력을 끼치시던 은우 형. 연구실을 항상 유쾌한 분위기로 이끌어주시던 인환 형. 항상 따뜻하게 연구실 분들을 대해주시던 윤선 누나. 즐겁게 일하시면서도 항상 뛰어난 모습을 보여주시던 동훈 형. 언제나 재밌으시면서도 위트가 있으시고 후배와 잘 놀아주시던 건호 형. 밝은 성격으로 연구실을 언제나 즐거움이 넘치게 만들고, 그러면서도 후배들에게 따스한 조언을 아끼지 않던 헤민 누나. 제가 연구가 막힐 때마다 직접적으로 도움을 주시고, 저에게 연구자로서의 방향성에 대해 가르쳐 주신 경훈 형. 항상 유머러스하시고 착하시면서도, 언제나 후배들에게 연구 내외적으로 큰 도움을 아낌없이 나눠주시던 경재 형. 연구실의 중심이자 큰 형으로써, 항상 재미있고 좋은 분위기로 연구실을 이끌어 나가시던 승규 형. 제가 모르는게 있을때마다 무엇이든 친절하게 가르쳐주시고, 졸업 논문 준비와 발표에도 큰 도움과 조언을 아끼지 않아주셨던 찬호 형 등. 연구실에서 뵈었던 많은 선배님들께 감사 말씀 드리고 싶습니다. 선배님들의 따뜻한 조언과 격려가 저에게 있어서는 큰 힘이 되었습니다.

그리고 저의 동기들과 그리고 후배들에도 감사드리고 싶습니다. 6년간 항상 같이 함께 일하면서 많은 일들이 있었지만, 동기와 후배들 덕분에 즐거운 대학원 생활을 할 수 있었던 것 같습니다. 항상 밝고 연구실 분위기를 잘 이끌어 나가면서도, 한편으로 같이 졸업 논문 준비를 하면서 도움을 많이 주었던 누리. 쾌활하면서도 누구에게든 친하게 대해주고, 동기로서 대학원 생활동안 많은 도움을 주었던 휘연. 착하고 활동적이고, 나보다 어리긴 하지만 오히려 항상 배울 점들이 많았던 윤호. 재미있으면서도 선후배 상관없이 항상 서스럼없는 모습으로 분위기를 유쾌하게 만들어주던 재구. 언제나 밝은 모습이면서도 열정적으로 노력하며 연구를 계속 해오던 오빈. 장난기가 많긴 하지만 재미있고, 한편으로 과제 관련해서도 항상 수고를 많이

해와줬던 민의. 자율주행 관련된 연구를 같이 하면서 도움을 많이 주고받고, 논문 관련해서도 수고를 많이 해주었던 건민. 밝으면서도 항상 연구실에 재미있는 에피소드를 만들어주었던 민재. 방장일로 항상 바쁘게 수고해주었고, 연구실에서 궂은일을 도맡아 해주었던 호건. 연구를 열정적으로 하면서도, 자율주행 관련 과제로 수고를 많이 해주었던 도형. 자기 하는 일에 열심이고, 항상 재미있게 일하는 모습 보여주던 정호. 논문이나 BMRR 과제로 항상 우직하게 수고해주던 우석. 착하면서도 재미있는 연구실 분위기를 만들어주었던 홍중. 자율주행 차량 관련해서 열심히 세팅도 도와주고 고마운 점이 많았던 정우. 남과 잘 어울리고 밝은 연구실 분위기를 만들어가던 재연. 연구실에서 항상 서스럼없이 친절하게 대해주시던 호웅 형. 언제나 밝고 항상 자신있는 모습을 보여주던 민영. 연구 관련해서 도움도 많이 주었고, 반대로 의지해주는 점도 많아서 고마웠던 재석 등. 연구실에서 만났던 많은 후배분들께도 감사드리고 싶습니다. 연구실에서 좋은 추억과 기억을 가지고 가게 되는 것 같아, 다시 한 번 감사드립니다.

마지막으로 항상 대학원 생활을 물심양면으로 지원해주시던 부모님과 가족들에게도 감사드리고 싶습니다. 덕분에 많은 힘이 되었고, 헌신적인 지지와 도움 덕분에 대학원 생활을 무사히 마칠 수 있었던 것 같습니다. 그 밖에 감사드리고 싶은 분들이 많지만 다 말로 표현하기에는 어려움이 있는것 같습니다. 제 주변에 항상 좋은 분들이 계셨고, 그분들 덕분에 제가 여기까지 올 수 있었던 것 같습니다.

이제 저는 10년간 재학했던 학교를 떠나게 됩니다. 4년간의 학사과정, 그리고 6년간의 대학원과정을 통해, 제 자신의 능력과 한계를 모두 경험할 수 있었던 것 같습니다. 아쉬운 부분도 많았고 후회되는 일도 많았지만, 돌이켜보면 그래도 제가 어느 정도 성장했음을 느끼게 되는 것 같아 다행이라는 생각이 듭니다. 박사라는 것은, 그 사람이 뛰어난 전문가라는 것을 드러내는 자격이라기 보다는, 그 분야에서 새로운 문제를 해결할 수 있고 연구자로서 한 사람 몫을 해낼 수 있다는 것을 인증하는 자격이다라는 말을 들은 적이 있습니다. 제가 박사과정 기간동안 여러 연구를 하기는 했지만, 제 연구보다 뛰어난 업적을 세우신 분들도 많고, 저보다 능력이 뛰어난 분들도 많이 있을 것입니다. 그런 분들과 비교해서 부족하기는 하지만, 적어도 한 사람의 박사로서, 그리고 그에 걸맞은 한 사람 몫을 해내는 연구자로서, 앞으로의 제 삶을

살아갈 수 있도록 노력하겠습니다. 감사합니다.