N° d'ordre NNT : **2020LYSE1331**

THÈSE de DOCTORAT DE L'UNIVERSITÉ DE LYON

Opérée au sein de :

l'Université Claude Bernard Lyon 1

Ecole Doctorale 512

Informatique et Mathématiques

Spécialité de doctorat : Informatique

Soutenue publiquement le 16/12/2020, par :

Zine el abidine Kherroubi

Novel off-board decision-making strategy for connected and autonomous vehicles (Use case: Highway on-ramp merging)

Devant le jury composé de :

Prashant Doshi	**Rapporteur**
Professeur, Université de Georgia, USA	
Leila Merghem-Boulahia	**Rapportrice**
Professeur des Universités, Université de technologie de Troyes	
Hamamache Kheddouci	**Examinateur**
Professeur des Universités, Université Claude Bernard Lyon 1	
Rene Mandiau	**Examinateur**
Professeur des Universités, Universite de Polytechnique Hauts-de-France	
Samir Aknine	**Directeur de thèse**
Professeur des Universités, Université Lyon 1	
Rebiha Bacha	**Directrice de thèse**
Connected Services Product Owner, Groupe Renault	

UNIVERSITÉ CLAUDE BERNARD LYON 1

Novel off-board decision-making strategy for connected and autonomous vehicles (Use case: Highway on-ramp merging)

by

Zine el abidine Kherroubi

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

LIRIS - Laboratoire d'Informatique en Image et Système d'information

Department of Computer Science

April 2021

Declaration of Authorship

I, Zine el abidine Kherroubi, declare that this thesis titled, 'Novel off-board decision-making strategy for connected and autonomous vehicles (Use case: Highway on-ramp merging)' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- The present work is part of an industrial research project for the French multinational automobile manufacture ***Renault S.A***.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. Except for such quotations, this thesis is entirely my work.
- I have acknowledged all main sources of help.

Signed: _____

Date: _____

"We're going to make it happen. As God is my bloody witness, I'm hell-bent on making it work."

- Elon Musk

UNIVERSITÉ CLAUDE BERNARD LYON 1

Abstract

LIRIS - Laboratoire d'Informatique en Image et Système d'information
Department of Computer Science

Doctor of Philosophy

by Zine el abidine Kherroubi

Merging in the highway on-ramp is a significant challenge toward realizing fully automated driving (*level 4* of autonomous driving). The combination of communication technology and autonomous driving technology, which underpins the notion of Connected Autonomous Vehicles (*CAVs*), may improve greatly safety performances when performing highway on-ramp merging. However, even with the emergence of *CAVs* vehicles, some key constraints should be considered to achieve a safe on-ramp merging. First, human-driven vehicles will still be present on the road, and it may take decades before all the commercialized vehicles will be fully autonomous and connected. Also, on-board vehicle sensors may provide inaccurate or incomplete data due to sensors' limitations and blind spots, especially in such critical situations. To resolve these issues, the present thesis introduces a novel solution using an off-board Road-Side Unit (*RSU*) that has perception sensors and Vehicle-To-Everything (*V2X*) communication, to realize fully automated highway on-ramp merging for connected and automated vehicles. Our proposed approach is based on an Artificial Neural Network (*ANN*) to predict drivers' intentions. This prediction is used as an input state to a Deep Reinforcement Learning (*DRL*) agent that outputs the longitudinal acceleration for the merging vehicle. To achieve this, we first show how the road-side unit may be used to enhance perception in the on-ramp zone. We then propose a driver intention model that can predict the behavior of the human-driven vehicles in the main highway lane, with *99%* accuracy. We use the output of this model as an input state to train a Twin Delayed Deep Deterministic Policy Gradients (*TD3*) agent that learns *"safe"* and *"cooperative"* driving policy to perform highway on-ramp merging. We show that our proposed decision-making strategy improves performance compared to the solutions proposed previously.

UNIVERSITÉ CLAUDE BERNARD LYON 1

Résumé

LIRIS - Laboratoire d'Informatique en Image et Système d'information
Department of Computer Science

Doctor of Philosophy

by Zine el abidine Kherroubi

L'insertion sur autoroute est un défi pour réaliser une conduite entièrement automatisée (*Niveau 4* de conduite autonome). La combinaison des technologies de communication et de conduite autonome, qui sous-tend la notion de Connected Autonomous Vehicle (*CAV*), peut améliorer considérablement les performances de sécurité lors de l'insertion sur autoroute. Cependant, même avec l'émergence des véhicules *CAVs*, certaines contraintes clés doivent être prises en compte afin de réaliser une insertion sécurisée sur autoroute. Tout d'abord, les véhicules conduits par des conducteurs humains seront toujours présents sur la route, et il faudra peut-être des décennies avant que tous les véhicules commercialisés ne soient entièrement autonomes et connectés. Aussi, les capteurs embarqués des véhicules peuvent fournir des données inexactes ou incomplètes en raison des limites des capteurs et des angles morts, en particulier dans de telles situations de conduite critiques. Afin de résoudre ces problèmes, la présente thèse propose une nouvelle solution utilisant une unité de bord de route (Road-Side Unit *(RSU)*) qui dispose de capteurs de perception et de communication Vehicle-To-Everything (*V2X*), permettant une insertion entièrement automatisée sur autoroute pour des véhicules connectés et autonomes. Notre approche est basée sur un réseau de neurones artificiels (*ANN*) pour prédire l'intention des conducteurs. Cette prédiction est utilisée comme état d'entrée pour un agent Deep Reinforcement Learning (*DRL*) qui fournit l'accélération longitudinale pour le véhicule qui s'insère. Afin d'y parvenir, nous montrons d'abord comment l'unité Road-Side Unit peut-être utilisée pour améliorer la perception dans la zone d'insertion sur autoroute. Ensuite, nous proposons un modèle de reconnaissance d'intention du conducteur qui peut prédire le comportement des véhicules conduits par des conducteurs humains sur la voie principale de l'autoroute, avec une précision de *99%*. Nous utilisons la sortie de ce modèle comme état d'entrée pour entrainer un agent Twin Delayed Deep Deterministic Policy Gradients *(TD3)* qui apprend une politique de conduite *"sûre"* et *"coopérative"* pour effectuer l'insertion sur autoroute.

Nous montrons que notre stratégie de prise de décision améliore les performances par rapport aux solutions proposées dans l'état de l'art.

Acknowledgements

I would like to express my very great appreciation to my supervisor, Professor *Samir Aknine*, who guided me through my three years of research. He has helped me a lot with his patience, support, availability, and encouragement.

I would also like to offer my special thanks to my co-supervisor, *Mrs. Rebiha Bacha*, with whom I have spent a wonderful time working. Her availability, kindness, and professional network have helped the advancement of my research work.

My special thanks go to the reviewers, *Pr. Prashant Doshi* and *Pr. Leila Merghem-Boulahia*, and the examiners, *Pr. Hamamache Kheddouci* and *Pr. Rene Mandiau*, for their constructive comments and advice, before and during the defense. I am particularly grateful for the support and good times given by my Algerians friends, in Paris and everywhere else. Their friendship has surely made my years less stressful.

I am particularly grateful for the support and good times are given by my colleagues and friends in the connectivity and multimedia innovation service at *Renault's technocentre* for every interesting discussion that we have had, inside and outside the workplace.

To my beloved family for all the support through my study. My special thanks go to my parents and my sisters for all their belief, their prayers, their moral support, and their unconditional love.

Contents

Declaration of Authorship i

Abstract iii

Abstract iv

Acknowledgements vi

List of Figures ix

List of Tables xi

Abbreviations xii

1 Introduction 2
- 1.1 Problem definition . . . 4
  - 1.1.1 Road-Side Unit perception . . . 5
  - 1.1.2 Road-Side Unit communication . . . 7
  - 1.1.3 Advantages of using Road-Side Unit to support Autonomous Driving 9
- 1.2 Research challenges . . . 10
- 1.3 Research contributions . . . 12
- 1.4 Thesis outline . . . 15

2 State-of-the-Art 17
- 2.1 Introduction . . . 17
- 2.2 Drivers' intentions estimation . . . 18
- 2.3 Decision making . . . 29
  - 2.3.1 Classical methods . . . 29
  - 2.3.2 Reinforcement learning based methods . . . 39
- 2.4 Conclusion . . . 50

3 Drivers' intentions estimation 53
- 3.1 Introduction . . . 53
- 3.2 Drivers' intentions model . . . 54

3.3 Probabilistic classifiers . . . 58
3.3.1 Logistic regression . . . 58
3.3.2 Naïve-Bayes . . . 59
3.3.3 Tree Augmented Naïve-Bayes . . . 59
3.3.4 General Bayesian Network . . . 60
3.3.5 K-Nearest Neighbor . . . 61
3.3.6 Artificial Neural Networks . . . 61
3.4 Experimental Evaluation . . . 62
3.4.1 Training Dataset . . . 62
3.4.2 Results . . . 64
3.4.2.1 Driver intention model extension . . . 67
3.4.2.2 On-board driver intention model . . . 69
3.5 Conclusion . . . 72

4 Decision making **74**
4.1 Introduction . . . 74
4.2 Preliminaries on reinforcement learning . . . 75
4.2.1 Twin Delayed DDPG (TD3) . . . 77
4.3 Highway on-ramp merging modeling . . . 79
4.4 Experimental setup . . . 82
4.5 Results . . . 84
4.5.1 Using a safe controller . . . 84
4.5.2 Using the driver intention model . . . 87
4.5.3 Using the Twin Delayed Deep Deterministic Policy Gradient (*TD3*) agent . . . 91
4.6 Conclusion . . . 94

5 Conclusions and future work **97**
5.1 Future Work . . . 98

A SUMO simulator **101**
A.1 Creating the Network in netedit . . . 102
A.2 Demand Generation in netedit . . . 103
A.2.1 Creating a Route . . . 104
A.2.2 Adding a vehicle . . . 104
A.3 Visualizing in sumo-gui . . . 105

B Interfacing TraCI from Python **107**
B.1 importing **traci** in a script . . . 107
B.2 First Steps . . . 108
B.3 Subscriptions . . . 108
B.4 Context Subscriptions . . . 109
B.5 Controlling parallel simulations from the same TraCI script . . . 109
B.6 Controlling the same simulation from multiple clients . . . 110

Bibliography **111**

List of Figures

1.1 The role of perception and communication in the autonomy of vehicles [1]: the merge of advanced autonomous technology with communication technology leads to collaborative autonomy. . . . 3
1.2 Highway on-ramp merging for connected and autonomous vehicles using off-board Road-Side Unit (*RSU*) with *CAVs* vehicles technologies. . . . 5
1.3 Vehicles' trajectories extraction using the 360-degree (rotating) *LiDAR* [2]. 6
1.4 Autonomous highway on-ramp merging: a. Using *RSU*, b. Without using *RSU*. . . . 10

2.1 Modeling and analysis of acceptability for merging vehicle at highway junction [3]. . . . 18
2.2 Probabilistic graphical model of the social behavior of an autonomous vehicle: V_n is the current speed, V_i is the speed at the previous time step; T_m, T_h are the current time-to-arrival for merging and host car respectively; I is the latent intention which needs to be estimated [4]. . . . 19
2.3 Classification of gap choice [5]. . . . 21
2.4 Schematic illustrating input variables [6]. . . . 22
2.5 Geometric Characteristics of Tapered Entrance Ramps on *I-95* at (a) *J.T. Butler NB-WB* Approach, and (b) *J.T. Butler SB-EB* Approach, and Parallel Entrance Ramps *on I-95* at (c) *Phillips NB*, and (d) *Baymeadows NB* [7]. . . . 24
2.6 Comparison of observed and simulated trajectories at *Hamazaki-bashi*. (Left) Ramp lane *1*. (Right) Ramp lane *2*. [8] . . . 25
2.7 Data flow and architecture of the proposed *GBNN* [9]. . . . 26
2.8 Comparison of different correlation methods (*Pearson*, *Kendall/Spearman*, and *random*), which are used for setting the weights of gate, on the effects of the prediction accuracy of *"non-merge"* and *"merge"* events [9]. 28
2.9 The slot-changing problem [10]. . . . 29
2.10 On-ramp merging [10]. . . . 30
2.11 Ramp metering configuration [11]. . . . 32
2.12 Merging roads with connected and automated vehicles controlled by a centralized controller [12]. . . . 34
2.13 Observation of the driving behavior in the main lane [13]. . . . 36
2.14 Graph of the *Q-function* approximator [14]. . . . 41
2.15 Reinforcement learning procedure [14]. . . . 41
2.16 Curve of single total rewards of ramp vehicles [14]. . . . 42

2.17 **a)** The *3-car* system for merging. Merging vehicle (*Car-0*) should ideally merge midway between the following vehicle (*Car-2*) and leading vehicle (*Car-1*). dx_{10} and dv_{10} denote *Car-0*'s relative position and velocity from *Car-1*. **b)** A typical freeway-merge situation. There are three mergeable spots: Spot 1, 2, and 3. The merging vehicle needs to determine a spot from a set of the candidates and controls input to merge [15]. 44
2.18 Illustration of the vehicles observed by the ego vehicle. The observation vector (or feature vector) contains information on the position and velocity of the observed vehicles [16]. 46
2.19 Schematic for merging [17]. 48

3.1 Motion modeling overview [18]. 54
3.2 Directed graphical model used for drivers' intentions estimation at the highway on-ramp merging situation. 56
3.3 Contextual vector for the vehicle in the merge lane and the vehicle in the main lane. 57
3.4 Naïve-Bayes structure. 59
3.5 Tree Augmented Naïve-Bayes structure. 60
3.6 General Bayesian Network structure. 61
3.7 A typical neural network architecture. 62
3.8 Study area schematic and camera coverage [19]. 63
3.9 Data preprocessing to identify the merging vehicle *ID*. 64
3.10 Time to arrival at the merging point when the *ANN* model outputs the first *TRUE* intention's prediction. 66
3.11 (a) The follower vehicle's intention model (b) Context situation vector. . . 67
3.12 Time to arrival for the first *True* prediction of the follower vehicle's model. 69
3.13 On-board highway on-ramp driver intention model. 70

4.1 The Reinforcement Learning Framework [20]. 75
4.2 Highway on-ramp merging modeling. 80
4.3 Ramp merge (a) simulated scenario under *SUMO* and (b) real-world location schematic. 82
4.4 DRL architecture (a) DDPG agent and (b) DDPG agent with safe controller (*DDPG-SC*). 85
4.5 Average undiscounted reward over *100* episodes during training 86
4.6 Vehicle architecture: (a) *DDPG* agent with safe controller (*DDPG-SC*) (b) *DDPG* agent with safe controller and driver intention model (*DDPG-SC-I*) 89
4.7 Average undiscounted reward over *100* episodes during training 90
4.8 Vehicle architecture: (a) Agent with safe controller (*DDPG*, *TD3*) (b) Agent with safe controller and driver intention model (*DDPG-I*, *TD3-I*) . 92
4.9 Average undiscounted episode reward during training. 93

List of Tables

2.1 Features extracted from *NGSIM* for modelling an *MLC* at on-ramps of highways [9]. . . . 27
2.2 Testing results [17]. . . . 49
2.3 Drivers' intentions estimation approaches at highway on-ramp situation. . . . 50
2.4 Decision-making methods at highway on-ramp situation. . . . 51

3.1 Vector C features. . . . 57
3.2 Models' performances. . . . 65
3.3 Models' accuracy. . . . 66
3.4 Vector C features for the follower vehicle. . . . 68
3.5 Performance of the follower vehicle's model. . . . 68
3.6 Vector C features for on-board model. . . . 70
3.7 On-board main lane model's performances. . . . 71
3.8 On-board merge lane model's performances. . . . 71

4.1 *DDPG* agent training parameter values using for algorithm 1. . . . 86
4.2 *DDPG-SC* agent testing results . . . 87
4.3 *DDPG* agents testing results . . . 90
4.4 Agents training parameter values. . . . 92
4.5 Agents testing results . . . 94

Abbreviations

CAV	**C**onnected **A**utonomous **V**ehicles
CV	**C**onnected **V**ehicles
DAS	**D**river **A**ssistance **S**ystems
AD	**A**utonomous **D**riving
RSU	**R**oad **S**ide **U**nit
V2V	**V**ehicle **T**o **V**ehicle communication
V2I	**V**ehicle **T**o **I**nfrastructure communication
V2X	**V**ehicle **T**o **E**verything communication
AI	**A**rtifical **I**ntelligence
PGM	**P**robabilistic **G**raphical **M**odel
LRM	**L**ogistic **R**egression **M**odel classifier
NB	**N**aive **B**ayes
TAN	**T**ree **A**ugmented **N**aive-Bayes
GBN	**G**eneral **B**ayesian **N**etwork
KNN	**K**-**N**earest **N**eighbors
ANN	**A**rtificial **N**eural **N**etwork
NN	**N**eural **N**etwork
TP	**T**rue **P**ositives
TN	**T**rue **N**egatives
FP	**F**alse **P**ositives
FN	**T**alse **N**egatives
RL	**R**einforcement **L**earning
DQN	**D**eep **Q**-**N**etwork
DRL	**D**eep **R**einforcement **L**earning
DDPG	**D**eep **D**eterministic **P**olicy **G**radients

TD3	**T**win **D**elayed **D**eep **D**eterministic Policy Gradients
SC	**S**afe **C**ontroller
DIM	**D**river **I**ntention **M**odel
SUMO	**S**imulation of **U**rban **Mo**bility
TraCI	**Tra**ffic **C**ontrol **I**nterface

Dedicated to my mother, my father, and my sisters ...

Chapter 1

Introduction

Interest in intelligent vehicles research has been growing in the last two decades, which significantly improves transportation security and comfort [1]. Early *DAS*[1] were based on proprioceptive sensors, i.e. sensors measuring the internal status of the vehicle, such as wheel velocity, acceleration, or rotational velocity. These enable the control of vehicle dynamics to follow the trajectory requested by the driver in the best possible way. A high level of autonomy is achieved with technologies such as road/lane detection, vehicle detection, and tracking, localization, and mapping. Most current autonomous driving systems are perception-based and rely on a myriad of on-board sensors. However, sensor fusion alone cannot guarantee the safety of *AD*[2] cars in a complex traffic environment [21]. Sharing this on-board data would be beneficial to other vehicles on the road.

Inter-CAV communication is the enabling technology for enhancing efficiency and safety of *CAVs*; without reliable *inter-CAV* communications, it is going to be extremely hard to achieve intersection control and collision avoidance. In fact, the *NHSTA*[3] predicts that effectively applying vehicle-to-vehicle (*V2V*) and vehicle-to-infrastructure (*V2I*) communications could potentially reduce and/or eliminate up to *80%* crashes of any type from non-impairment (*NHTSA*, *2017*) [22]. However, the communication requirements for cooperative perception and maneuvering are yet to be understood in detail. Advances in communication technology such as *DSRC*[4], *Wi-Fi*[5], and *LTE*[6] have paved the way for connected vehicles, which will bring the intelligent transportation field towards collaborative autonomy.

[1] Driver-Assistance Systems.
[2] Autonomous Driving.
[3] National Highway Traffic Safety Administration.
[4] Dedicated Short Range Communications.
[5] Wireless Fidelity.
[6] Long Term Evolution.

In full collaborative autonomy, on-board sensors from individual cars and data sharing between connected vehicles are used in conjunction to increase the overall *"intelligence"* of traffic [1][23]. The relationship between perception-based autonomy, communication-based autonomy, and collaborative autonomy is shown in Figure 1.1.

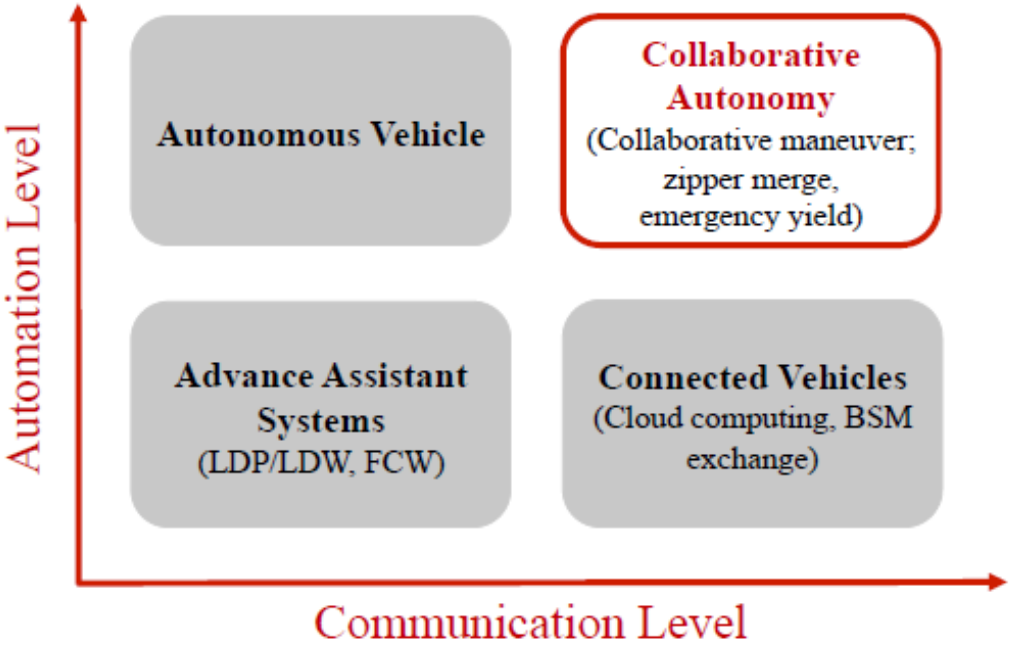


FIGURE 1.1: The role of perception and communication in the autonomy of vehicles [1]: the merge of advanced autonomous technology with communication technology leads to collaborative autonomy.

Connected and automated vehicle (*CAV*) is a transformative technology that has great potential to realize successful autonomous driving. Nevertheless, even with the emergence of such *CAVs* vehicles, performing fully autonomous driving requires taking into account several essential issues:

- First, vehicle horizon is limited to the range of its sensors (*camera*, *RADAR*[7] and *LiDAR*[8]). Moreover, the processing of data collected by all sensors in the vehicle demands a lot of computational resources.
- Second, even if vehicle-to-vehicle (*V2V*) communication technology allows us to overcome sensors range limitation, it can be used only if all vehicles are equipped with *V2V* capabilities. This is not the case in the real world because it may take decades before that all the vehicles become connected. Vehicles driven by

[7] A sensor for determining the presence and location of an object by measuring the time for the echo of a radio wave to return from it and the direction from which it returns.
[8] A sensor that is similar in operation to RADAR but emits pulsed laser light instead of microwaves.

humans will still be present on the road and they cannot be controlled directly. The behavior of their drivers should be considered by the autonomous vehicle.

- Last but not least, the exploitation of vehicle data using communication technology to improve driving automation requires taking into account current communication technology limitations, namely in particular communication latency.

The connected vehicles (*CV*) network relies on the communication of different roadside and on-board sensors, so all the road users can share their real-time information. However, the current *CV* application is constrained by the resolution of data input [24]. The *CV* network requires high-resolution micro traffic data (*HRMTD*), which means second-by-second real-time traffic data of all individual road users [25], [26]. Since, as stated above, it takes time to build a whole connected system (especially vehicle-to-vehicle communication), supplemental data provided by the roadside infrastructure are required to help the deployment of the *CV* network [27]. Automation of complex traffic scenarios is expected to rely on input from a roadside infrastructure to complement the vehicles' environment perception [28].

To exploit the advantages of combining *CAV* vehicle technology with the roadside infrastructure, we study the highway on-ramp merging situation. Although there are lots of driving situations to be considered, the merging task at highway junctions is now recognized as a key task to realize the automated drive on the highway [29]. Ramp merging is a critical maneuver for road safety and traffic efficiency. According to the *US Department of Transportation*, nearly *300,000* merging accidents occur every year with *50,000* being fatal [30]. Even, the prototype vehicles of the leading self-driving car company, *Waymo*, have reportedly been seen unable to merge autonomously [31].

The present thesis answers the following research problematic:

- How the combination of road infrastructure and *CAVs* vehicles technologies could improve the automated driving level, particularly in highway on-ramp merging?

1.1 Problem definition

The present work is part of an industrial research project for the French multinational automobile manufacture ***Renault S.A***. This thesis addresses the use of off-board Road-Side Unit (*RSU*), i.e. road infrastructure, with *CAV* vehicle technologies to perform autonomous highway on-ramp merging. The use case is illustrated in Figure 1.2.

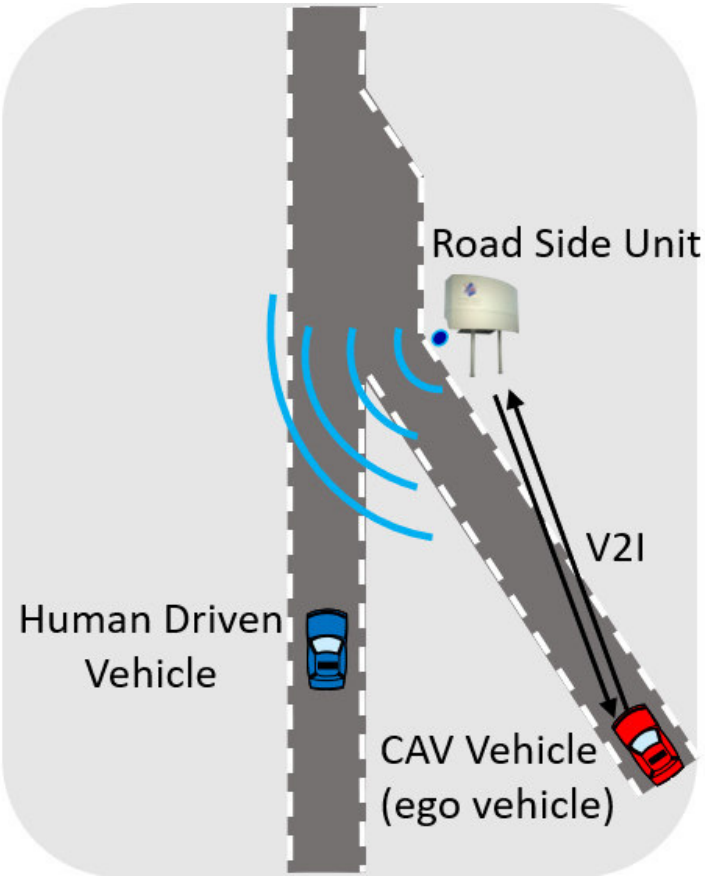


FIGURE 1.2: Highway on-ramp merging for connected and autonomous vehicles using off-board Road-Side Unit (*RSU*) with *CAVs* vehicles technologies.

The main advantages of such an approach are:

- First, it allows an increased perception that exceeds the limit of embedded vehicle sensors, thanks to the *RSU* sensors (*camera*, RADAR, or LiDAR).
- Moreover, it can be used in mixed traffic situation with the presence of human-driven vehicles that does not have *V2V* communication technologies. This is only possible using road-side unit that owns *V2I* communication capabilities.
- Last but not least, this configuration offers more computational power at the off-board road-side unit side in comparison to on-board vehicle calculators. This allows to implement more sophisticated algorithms, such as *Edge AI*[9] models.

1.1.1 Road-Side Unit perception

Road-side unit equipped with sensors (*camera*, *RADAR* and *LiDAR*) increases the perception range and reliability compared to on-board vehicle sensors.

[9] *Edge AI* means that *AI* algorithms are processed locally on a hardware device.

Authors in [2] propose a roadside *LiDAR-enhanced* infrastructure as a new application of the 360-degree (rotating) *LiDAR* (Figure 1.3). Considering the massive deployment of roadside *LiDAR* in the future, the roadside *LiDAR* has to provide the *HRMTD*[10] in an extended range compared to the on-board *LiDAR* serving autonomous vehicles. The authors also developed a data processing procedure for detecting and tracking vehicle trajectories with a roadside *LiDAR* sensor. The validation results suggest that the tracking speed matches real driving speed accurately.

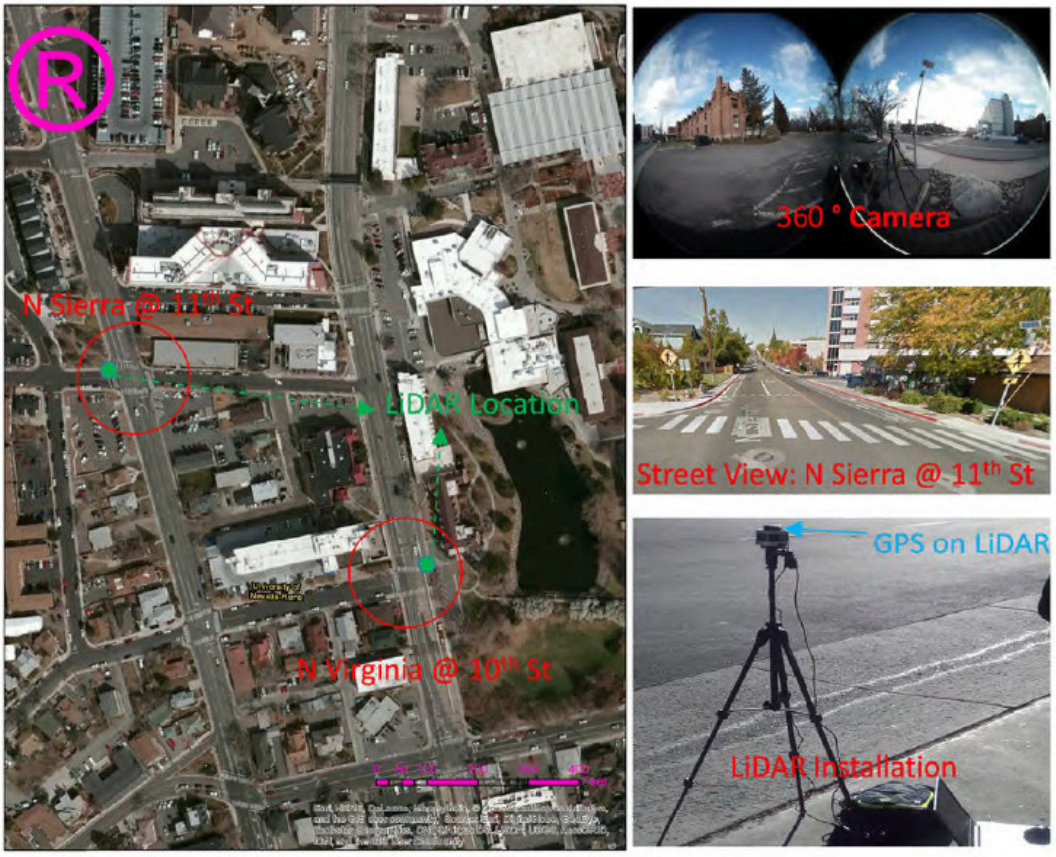


FIGURE 1.3: Vehicles' trajectories extraction using the 360-degree (rotating) *LiDAR* [2].

Authors in [32] explore design requirements for a prototype setup of virtual vision or *RADAR* sensors along one roadside. They modeled cameras with almost perfect detection capabilities, that act as a reference detection frame. Moreover, they modeled detection by RADAR and studied object tracking based on the *RADAR* detections. The perfect cameras and the generic *RADAR* sensors have exhibited very similar detection capabilities. They demonstrated also that tracking improves the completeness of detections especially for poor road coverage (e.g. small sensor range) since a vehicle trajectory

[10] High-resolution micro traffic data.

can still be predicted for some time after the object has left the sensors *FoV*[11]. Complete traffic detection up to a degree of *99%* can be readily achieved with feasible sensor parameters (*camera*, *RADAR* or *RADAR tracking*) in the low traffic density scenario. Therefore, the authors in [32] considered the sensor infrastructure to be an instrument for supporting *AD*, e.g. in the form of augmenting agent-based vision or for improved traffic orchestration.

1.1.2 Road-Side Unit communication

The road-side unit equipped with stationary sensors obtains object information from mounted stationary sensors: cameras, RADARs, and LiDARs, to provide any assistance to compliant cooperative vehicle. *RSU* could detect and track vehicles in the road and provides characteristics such as position, heading, and speed for every obstacle using *V2X* wireless communication technologies. Current *V2X* systems are based on two main technologies:

- **Dedicated short-range communications (*DSRC*):** *DSRC* is a type of wireless technology developed for the automotive platform, specifically for *CAV* usage [33]. *DSRC* is the pathway that allows *CAVs* to communicate with each other and the infrastructure. *DSRC* uses radio frequencies that lie in a range of the *5.9 GHz* band, which is controlled and allocated by the *Federal Communications Commission* (*FCC*)[12]. In terms of communication range, *DSRC* covers a maximum of *500 feet* in all weather conditions. *IEEE 802.11p*, the IEEE standard for *DSRC*, has been implemented and endorsed by various automotive manufacturers and the *U.S. Department of Transportation*. A *DSRC* system is composed of the on-board unit (*OBU*) and the road-side unit (*RSU*) [34]. *DSRC* can be one of the leading reasons why *CAVs* will help the traffic congestion caused by the human facet of driving [35], and the gradual increase of implementing *V2V* communications for collision avoidance. In *DSRC* standard, safety messages are sent intermittently with an estimated time period of $T = 0.1\ s$. One significant drawback of *DSRC* powered *V2X* communications is its low scalability, i.e. when the number of nearby vehicles is high. Moreover, *DSRC* is mostly designed for quick transmission of short-range basic safety messages and does not offer high bandwidth and low latency communication channels for more advanced *V2X* applications such as autonomous driving, especially in high-speed scenarios [33].

[11] Field of vision.

[12] *The Federal Communications Commission (FCC)* is an independent agency of the United States government that regulates communications by radio, television, wire, satellite, and cable across the United States.

- **Cellular based *V2X* (*C-V2X*):** An alternative to *DSRC* is cellular-based *V2X* (*C-V2X*) communications. What is really promising, and future proof is the *5G*[13] *C-V2X*, i.e., release *15* and *16 C-V2X*. Compared with *DSRC* and *3GPP LTE*[14], *5G C-V2X* offers very higher throughput and reliability (*99.999%*), longer range (*443 m* line of sight and *107 m* none line of sight), and very lower latency (*10 ms* end-to-end and *1 ms* over-the-air) [36]. Besides, *5G C-V2X* provides direct messaging services among *CAVs*, allowing short-range communication when cellular towers are unavailable. While *DSRC* is mostly targeted for basic safety applications, *5G C-V2X* will enable more advanced *CAV* features, such as autonomous driving and highway platoon. Currently, *5G C-V2X* products are being developed, and it is important to test, verify, and improve performance when they become available.

Regarding characteristics and limitations of each technology, a potential solution for communicated *RSU* is to use a hybrid communication that combines *DSRC* and *5G C-V2X* to benefit from the advantages of each technology and compensate for their drawbacks.

The information to be exchanged between the road-side unit and the *CAV* vehicle is packed up in the Collective Perception Message (*CPM*). Collective Perception (*CP*) is the concept of sharing the perceived environment of a station based on perception sensors. The Collective Perception Service aims at enabling intelligent transport system stations (*ITS-Ss*) to share information about other road users and obstacles that were detected by local perception sensors such as RADARs, cameras, and alike. In that sense, it aims at increasing awareness between *ITS-Ss* by mutually contributing information about their perceived objects to the individual knowledge base of the *ITS-S*. The service does not differentiate between detecting connected or non-connected road users. The *CPM* message provides generic data elements to describe detected objects in the reference frame of the disseminating *ITS-S*. The *CPM* is transmitted cyclically with adaptive message generation rates (minimum message every *100 ms*) to decrease the resulting channel load while focusing on reporting changes in the dynamic road environment. Overall, a *host-ITS-S* should generate *CPMs* for surrounding objects it detected with sufficient level of confidence [37]. The ability for *RSU* to share their local perception using Collective Perception Message (*CPM*) and for vehicles to receive such data extends the information that can be exploited for ensuring road safety of *CAVs*.

[13] The fifth-generation technology standard for cellular networks.

[14] *LTE* (Long Term Evolution) is an evolution of GSM/EDGE, CDMA2000, TD-SCDMA and UMTS mobile telephony standards, defined by *3GPP* (The 3rd Generation Partnership Project), which is a co-operation of a number of standards organizations that develop protocols for mobile telecommunications.

1.1.3 Advantages of using Road-Side Unit to support Autonomous Driving

As stated above, the new generation of the road-side unit can perceive the road environment using its sensors and output perceived objects with their attributes (position, speed ...). This information is packed as Collective Perception Message (*CPM*) and could be provided to the *CAV* vehicle through wireless communication technologies (*DSRC* and *5G C-V2X*). An infrastructure-based solution for autonomous highway on-ramp merging using road-side unit (Figure 1.4.a) is already feasible and would be practical for real-world implementation. Highway on-ramp merging solution based on *RSU* has the following advantages:

- **Computational power:** Using *RSU* offers more computational power at the off-board road infrastructure in comparison to the on-board vehicle calculators. This allows us to implement more sophisticated algorithms, such as *Edge AI* models.
- **Perception range:** Using *RSU* allows an increased perception that exceeds the limits of embedded vehicle sensors. Sensor fusion alone cannot guarantee the safety of the *CAVs* vehicles in a complex traffic environment, such as the highway on-ramp.
- ***V2X* deployment:** Performing autonomous driving in highway on-ramp without *RSU* requires that all the vehicles present on the road are equipped with *V2V* capability to provide their states for surrounding vehicles because sensor fusion alone cannot guarantee the full perception of the environment. On the contrary, using *RSU* based solution requires that only the highway on-ramp zone is equipped with off-board sensors and *V2I* technology to communicate with *CAV* vehicle, even in the presence of non-connected vehicles.
- **Cost:** Without using *RSU*, the cost of equipping all the vehicles with *V2V* communication technology is more considerable than the cost of equipping only highway on-ramp infrastructure with the road-side unit.
- **Communication latency:** Without using *RSU*, the communication between vehicles through *V2V* is done in decentralized way. In this case, minimum latency is not guaranteed. When using *RSU*, information is provided directly from the road-side unit to the merging vehicle. Hence, communication latency is controlled and measurable.
- **Security & privacy:** Without using *RSU*, it is difficult to trust messages received from different connected vehicles. When using a *RSU* based solution, privacy is

more guaranteed since the information is provided only by the infrastructure that is managed by a road operator.

- **Interoperability:** Without using *RSU*, it is more difficult to deal with interoperability between different car manufacturers. When using *RSU*, it is easier to ensure interoperability between the road operator, that is responsible for the *RSU* infrastructure, and each car manufacturer.

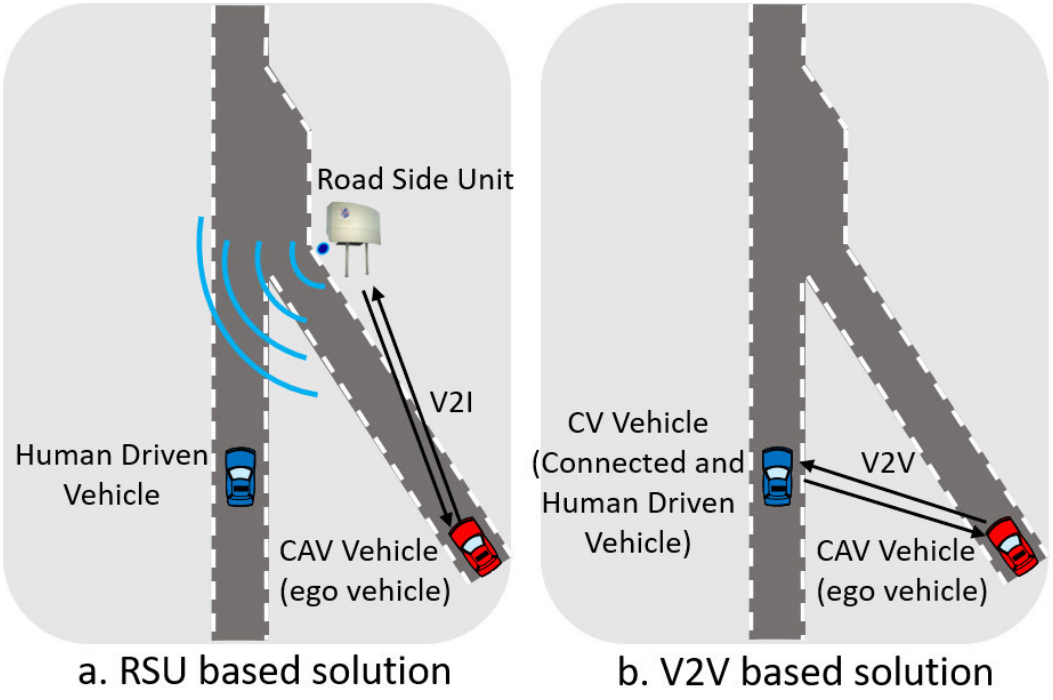


FIGURE 1.4: Autonomous highway on-ramp merging: a. Using *RSU*, b. Without using *RSU*.

1.2 Research challenges

Using infrastructure to support *CAVs* to perform autonomous highway on-ramp merging can be quite challenging. We can highlight some of the main challenges as follows:

Safety: Before taking into account any performance metric, decision-making at highway on-ramp needs to be safe. Safety is, indeed, the key to any transportation system currently and in the future. In the Intelligent Transportation System (*ITS*) where vehicles follow their crossing plan, safety would become even more crucial. Therefore, at any point in time after the vehicle has reached the highway on-ramp merging point, the system must be collision-free when the driving policy is tested. Safety could be improved by incorporating a safe controller that evaluates the safety of the autonomous

system's action according to some predefined security rules, such as minimal distance to the preceding vehicle.

Human-driven vehicles: Vehicle-to-vehicle (*V2V*) communication technology can be used only if all vehicles are equipped with *V2V* capabilities. This is not the case in the real world because it may take decades so that all the vehicles become connected. Vehicles driven by humans will still be present on the road. Such vehicles cannot be controlled directly. The behavior of their drivers should be considered by the autonomous driving system. Therefore, a probabilistic model could be used to predict the behaviors of human drivers at the highway on-ramp zone. This information may be used to make a cooperative decision by the autonomous vehicle. The driving policy cooperativeness can be evaluated by the number of emergency brakings performed by the surroundings vehicles after that the autonomous vehicle reaches the on-ramp merging point.

Communication latency: Wireless communication technology (*DSRC*, *5G C-V2X*) is the enabling technology for enhancing efficiency and safety of *CAVs* vehicles. However, the communication requirements for cooperative perception and maneuvering are yet to be understood in detail. *V2X* communication systems consider latency to be the most important performance metric as some applications (such as *precrash sensing*) require very stringent requirements [38]. Hence, the latency must be evaluated when information is communicated through *V2X* technologies between infrastructure and *CAVs* vehicles. We consider that the off-board road infrastructure should provide information to the *CAV* vehicle by a minimum time of TR=0.5 *sec* before reaching the merging point. This estimation is based on the following time constraints:

- *0.4 sec*, for the vehicle's dynamic response time [39].
- *0.1 sec*, for the communication latency time [40].

In addition to these main challenges, there might be some minor problems, such as communication data loss and computational requirements from the infrastructure. These problems need to be addressed later when deploying our proposed approach in the real world.

Thereafter, we discuss the contributions of this thesis regarding current state-of-the-art.

1.3 Research contributions

Against these challenges, the contributions of this thesis are as follows:

- A novel infrastructure-based decision-making strategy to perform autonomous highway on-ramp merging for connected and autonomous vehicles (*CAVs*). This solution ensures increased perception compared to vehicles' sensors, and more computational power compared to onboard vehicles' calculators. This solution can be used even in the presence of human-driven vehicles (*non-CAV*), which is advantageous in terms of deployment cost.

- A directed probabilistic graph model to predict drivers' intentions at highway on-ramp zone. This model takes into account the contextual situation for each vehicle to estimate the probability of *merging* or *not merging* for the vehicle in the merge lane and the probability of *yielding* or *not yielding* for the vehicle in the main lane.

- The training and testing of this model are done using existing probabilistic classifiers and real-world database. The comparison of these classifiers' performances shows the best values for the Logistic Regression Model classifier *(LRM)* and the Artificial Neural Network *(ANN)* classifier, which yields accuracy and precision of around *99%* for predicting drivers' intentions in highway on-ramp merging situation. The obtained results outperform the existing models proposed in the literature. The proposed method predicts the intention earlier than *0.5 sec* before reaching the highway on-ramp merging point, which is sufficient time for decision-making regarding communication latency and the vehicle's dynamic delay. In short, an off-board unit that uses *V2X* communication and sensors in the highway on-ramp can predict drivers' behaviors more accurately than the on-board vehicle's model, independently from the used probabilistic classifiers and drivers' driving styles.

- A formulation of the highway on-ramp merging scenario as a reinforcement learning problem with a reward function that motivates the merging vehicle to merge with similar speed as its preceding vehicle and at midway distance regarding the first preceding and the first following vehicle.

- A new driving architecture that incorporates main lane driver intention prediction as input state to Twin Delayed Deep Deterministic Policy Gradients (*TD3*) agent that performs autonomous highway on-ramp merging. Moreover, a safe controller is proposed to evaluate the safety of the control action according to some predefined security rules.

- The architecture that uses driver's intention estimation as input state to *TD3* agent accelerates significantly the policy training convergence. It improves safety performances and cooperativeness of the learned highway on-ramp merging driving policy.

When taken together, this work proposes a novel decision-making strategy supported by the road infrastructure. The solution combines a data-driven model at the off-board *RSU*, to predict drivers' intentions, with an autonomous driving system at the on-board vehicle. The driver's intention is predicted using an artificial neural network that provides accurate estimation in real-time regarding communication latency and the vehicle's dynamic delay. This driver's intention estimation is provided as an input state to a Twin Delayed Deep Deterministic Policy Gradients (*TD3*) agent that performs autonomous highway on-ramp merging. This novel architecture could be used in mixed traffic where non-connected human-driven vehicles are present on the road. The proposed solution accelerates convergence when learning autonomous driving policy, and improves safety performances and cooperativeness.

Parts of our work during the *PhD* has led to the publication of the following paper:

- *Student abstract:* Zine El Abidine Kherroubi, Samir Aknine, Rebiha Bacha. A Dynamic Bayesian Network Based Merge Mechanism for Autonomous Vehicles. *AAAI-19 Conference on Artificial Intelligence*, *January 2019*, *Hawaii*, *USA*. This work presents for the first time drivers' intentions model trained by the Logistic Regression Model (*LRM*). The paper was selected as finalist for the *3 minutes* presentation at the Thirty-Third *AAAI* Conference on Artificial Intelligence (*AAAI-19*) student abstract program which was held between *January 27* and *February 1*, *2019* at the *Hilton Hawaiian Village*, *Honolulu*, *Hawaii*, *USA*.

- *Long student abstract:* Zine El Abidine Kherroubi, Samir Aknine, Rebiha Bacha. Dynamic and intelligent control of autonomous vehicles for highway on-ramp merge. *AAMAS (International Conference on Autonomous Agents and Multiagent Systems)*, *Mai 2019*, *Montreal*, *Canada*. In this paper, we verify the robustness of the proposed drivers' intentions model.

- *Extended student abstract:* Zine El Abidine Kherroubi, Samir Aknine, Rebiha Bacha. New Off-board Solution for Predicting Vehicles' Intentions in the highway On-Ramp using Probabilistic Classifiers. *AAAI-2020 Conference on Artificial Intelligence*, *February 2020*, *New York*, *USA*. In this paper, we train and compare the driver intention model using existing probabilistic classifiers. Also, we verify the real-time feasibility of the proposed solution.

- *Conference paper (accepted):* Zine El Abidine Kherroubi, Samir Aknine, Rebiha Bacha. Novel Off-board Solution for Predicting Drivers' Intentions using Probabilistic Classifiers. *The 2020 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, *December 2020*, *Virtual Conference*. In this conference paper, we compare the performances of probabilistic classifiers for estimating drivers' intentions. Also, we verify the real-time feasibility of the proposed solution and compare it to the on-board vehicle approach.

- *Student abstract (accepted):* Zine El Abidine Kherroubi, Samir Aknine, Rebiha Bacha. Leveraging on Deep Reinforcement Learning for Autonomous Safe Decision-Making in Highway On-ramp Merging. *The Thirty-Fifth AAAI Conference on Artificial Intelligence*, *February 2021*, *Virtual Conference*. In this paper, we propose using the driver's intention prediction as input state to *DRL* agent. We demonstrate that this approach improves safety and cooperativeness of autonomous driving policy in highway on-ramp merging situation.

1.4 Thesis outline

The remainder of this thesis is organized as follows:

In Chapter 2, we explore the existing methods for motion modeling and drivers' behavior estimation in the highway on-ramp scenario. We discuss each method and show its limitations. We also review the existing solutions that were proposed for decision-making in the merging problem. These works studied the highway on-ramp merging problem using either classical deterministic methods such as slot-based and optimal-control or novel deep reinforcement learning algorithms such as deep deterministic policy gradient (*DDPG*). We show the limitations of each proposed methods.

In chapter 3, we show how the infrastructure could, due to its increased perception range, support autonomous driving by estimating drivers' intentions. We propose an off-board model to predict the intention of the driver at the main lane and the driver at the merge lane. We review the different probabilistic classifiers that are used to train this model. We then compare the performances of these classifiers for predicting drivers' intentions at the highway on-ramp. We show that this off-board model can reach a prediction accuracy higher than *99%* and can provide prediction information in real-time. Finally, we give a comparison between such off-board implementation and on-board vehicle's model.

In chapter 4, we show how the driver intention model could be used with a deep reinforcement learning agent to improve decision-making at the highway on-ramp. We first review preliminaries on continuous state-action reinforcement learning algorithms: *DDPG* and the twin delayed deep deterministic policy gradient (*TD3*). We then formulate the highway on-ramp merging problem as a reinforcement learning framework. The proposed approach is trained and validated using a traffic simulator with real-world traffic-conditions. The results are discussed. A safe controller is a key component for ensuring convergence of the deep reinforcement learning agent. Also, the use of the driver intention model as input to the *TD3* agent, that outperforms the *DDPG* agent, allows learning safe and cooperative autonomous driving policy.

Chapter 5 concludes the work presented in this thesis and outlines possible improvements in the future.

Chapter 2

State-of-the-Art

2.1 Introduction

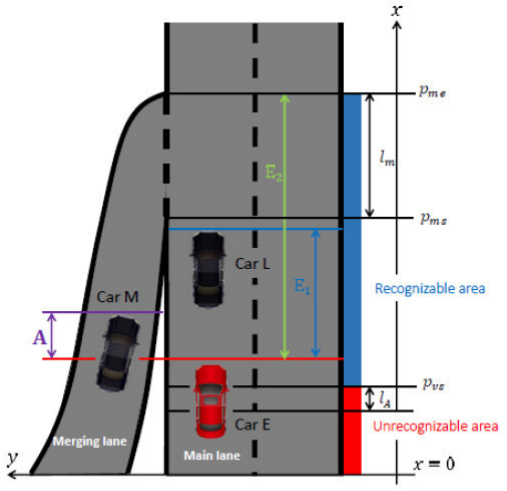
Although *CAVs* vehicles technology could enable safe autonomous driving, it will take time to build a whole connected system (especially vehicle-to-vehicle communication) [27]. Meanwhile, vehicles driven by human drivers will be present in the road even with the emergence of *CAVs* vehicles. These vehicles cannot be controlled directly. Hence, autonomous driving systems should take into consideration the behaviors of such vehicles. Many works studied the problem of drivers' intentions recognition and behaviors' prediction in the highway on-ramp situation. In the following, we will review the existing models and methods that were proposed to estimate drivers' intentions and behaviors in such a driving situation. We will show how each approach could be used and discuss the limitations of each proposed solution.

Moreover, many works have focused on centralized and decentralized approaches for coordinating connected and autonomous vehicles (*CAVs*) in highway on-ramp merging situations. These works are either classical rule-based approaches that include heuristics, optimal control, and model predictive control, or novel reinforcement learning-based methods. As for drivers' intention recognition, we will review the details of each decision-making method that was proposed and discuss their limitations.

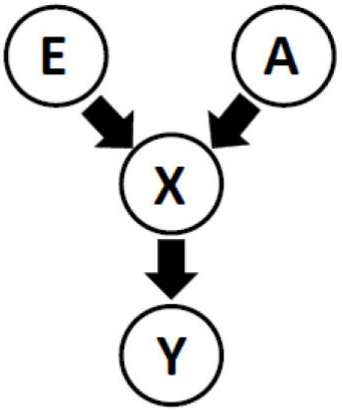
2.2 Drivers' intentions estimation

Many works studied the problem of drivers' intentions recognition in highway on-ramp situation.

Reference [3] proposes a model that analyzes the acceptability for merging a vehicle at highway junctions, for the driver driving in the main car lane (cf. Figure 2.2.A), using a Bayesian network and logistic regression based on the observation of the driving data on a driving simulator.



(A) Merging task on highway junction



(B) Bayesian network to estimate acceptability state

FIGURE 2.1: Modeling and analysis of acceptability for merging vehicle at highway junction [3].

The driver's acceptability for merging is represented by the Bayesian network as in Figure 2.2.B. The model is composed of input variables *(A, E)*, state variable *(X)* and output variable *(Y)*. There are two kinds of input variables, A and E where A shows the information related to cars M, and E shows the information related to Car E and Car L. The state variable X is the acceptability of the driver in the main lane and it can take three values: *Accept* (the driver allows the Car M to cut in between car E and car L), *Reject* (the driver *DOES NOT* allow the Car M to cut in at the front of Car E), *Unknown* (the driver does not decide either *'Accept'* or *'Reject'* yet). The output variable Y is the acceleration in each acceptability state. The conditional probabilities, *P(X|E,A)* and *P(Y|X)*, are modeled by using logistic regression model (*LRM*) and Gaussian distribution respectively. The model shows high accuracy of around *87%* even though its simplicity.

Although the proposed work analyses the acceptability of the driver in the main lane, it did not model the behavior of the vehicle in the merge lane. Besides that, the model was trained and tested with a traffic simulator that provides only simulated data obtained by five drivers who experimented with the merging scenario. This model should be further trained and validated using real-world data before it could be used in the real world. Besides, the parameters of the trained model varied from one driver to another. Hence, the accuracy and parametric form of the model depend on each driver's driving style. This is a key limitation when using such a model with a real merging assistance system that considers the driver's acceptability.

Reference [4] estimates drivers' intentions using a probabilistic graphical model (*PGM*) that organizes historical data and latent intentions and determines predictions (cf. Figure 2.2). The most important part of this method is to understand the cause-effect relationship between previous states and intention. To simplify and abstract this dependency, the authors applied a probabilistic graphical model. There are three kinds of nodes in this proposed model: *(1)* state nodes, which are the time-to-arrival for each car; *(2)* an intention node, which is either *"Yield"* or *"Not Yield"*; *(3)* speed nodes, which contain the speed history of the target vehicle.

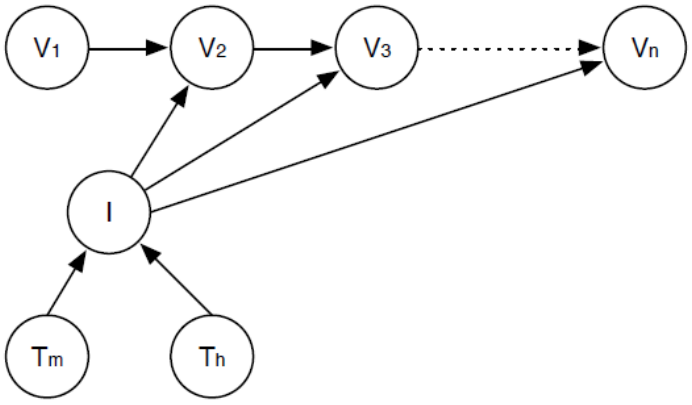


FIGURE 2.2: Probabilistic graphical model of the social behavior of an autonomous vehicle: V_n is the current speed, V_i is the speed at the previous time step; T_m, T_h are the current time-to-arrival for merging and host car respectively; I is the latent intention which needs to be estimated [4].

The probability of the merging car's *Yielding* or *Not Yielding* to the autonomous car is calculated using Bayes rules:

$$P(I \mid V, T_m, T_h) \; \alpha \; P(V, T_m, T_h \mid I).\; P(I) \tag{2.1}$$

Although this model has better performance when it is combined with the Adaptive Cruise Control (*ACC*[1]) compared to other control methods that do not include historical driver's data and latent intentions, it shows a failure rate of *7.6%* when it is tested on the *I-80* highway on-ramp of the *NGSIM*[2] database that is used in our thesis. This is a high failure rate for such driving systems where safety is the most critical performance metric. This is due, either, to the fact that the model uses the time to arrive for the merging car (T_m) and that of the host car (T_h), which is neither measured directly by sensors nor which can be observed by human drivers. Otherwise, it can be due to the use of historical data (speed) for current intention estimation, which makes it dependent on each driver's driving style.

References [41], [5], and [6] analyze and explore the theory of gap acceptance for merging behavior. The principle of the gap acceptance theory is that a driver assesses an offered gap (distance or time between two vehicles on the main road that it is driving next to). In this assessment, the gap is compared to a so-called *"critical gap"*: if the offered gap is larger than the critical gap, the gap will be accepted, otherwise it is rejected, and the driver will look for another offered gap. The critical gap depends on the characteristics of the traffic participant, the vehicle, and the road, and can be expressed either in time or in distance.
Reference [41] proposes a model for accepting or rejecting a gap during a merging maneuver. This stochastic model of gap rejection and acceptance was obtained by applying logistic regression analysis of the merging behavior, to express the probability of accepting or rejecting a gap as a function of the distance towards the end of the acceleration lane, the length in meters of the offered gap, the difference in speed between the putative leader and the putative follower, and the difference in speed between the merging vehicle and the putative follower. The predictive power of the model, assessed on two real data-sets, is *98%*.
Reference [5] analyzes and models gap choice behavior taking account of the effects of mainline traffic conditions, acceleration lane length, and the reactions of merging vehicles to the traffic in the mainline. The gap choice is classified into three patterns considering the reactions of merging vehicles to the mainline's adjacent gap at the decision point, as illustrated in Figure 2.3.

[1] Adaptive Cruise Control (*ACC*) is an available cruise control system for road vehicles that automatically adjusts the vehicle speed to maintain a safe distance from vehicles ahead.
[2] Next Generation Simulation: a database which provides vehicle trajectories and supporting data.

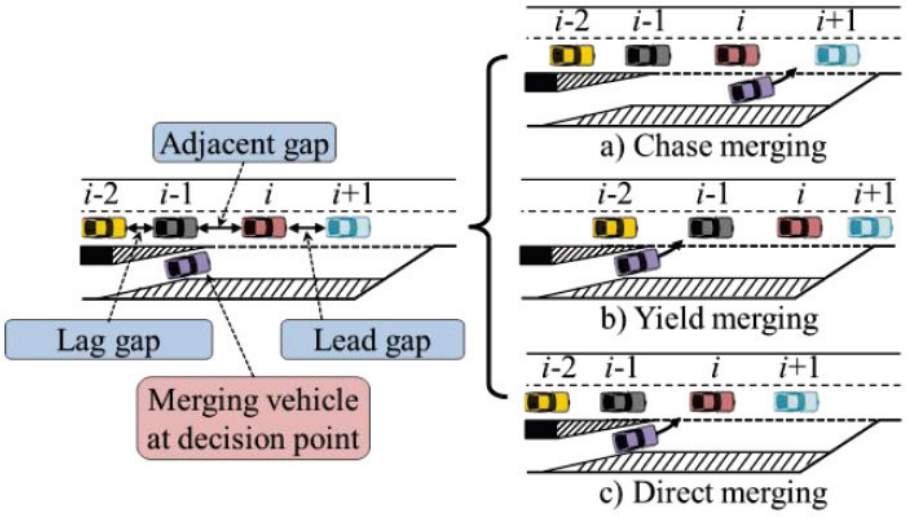


FIGURE 2.3: Classification of gap choice [5].

Empirical analysis showed that mainline traffic conditions significantly affect the proportion of gap choice. *"Yield"* choice can be observed more frequently in traffic conditions with high mainline speeds, while *"chase"* choice can be observed mostly in congested traffic conditions. The gap choice behavior was modeled by applying the binary logit model. Generally, it is found that relative speed, clearance between mainline vehicles and merging vehicles, traffic conditions, acceleration lane length, and remaining distance to the end of the acceleration lane are the most significant influencing factors. This model has an accuracy of *97%*. The proposed model is limited only to the acceleration lane located in the middle of the expressway carriageways.
Authors of [6] propose a lane-changing assistance system that advises drivers of safe gaps for making mandatory lane changes at lane drops. Their model uses five factors or dimensions that were found to affect a driver's merging decision, which is considered to be an input variable. These factors are shown in Figure 2.4 and are defined as follows:

- $\Delta V_{lead}(m/s)$ is the speed difference between the lead vehicle in the target lane and the merging vehicle.
- ΔV_{lag} (m/s) is the speed difference between the lag vehicle in the target lane and the merging vehicle.
- D_{lead} (m) is the gap distance between the lead vehicle in the target lane and the merging vehicle.
- D_{lag} (m) is the gap distance between the lag vehicle in the target lane and the merging vehicle.
- S (m) is the distance from the merging vehicle to the beginning of the merge lane.

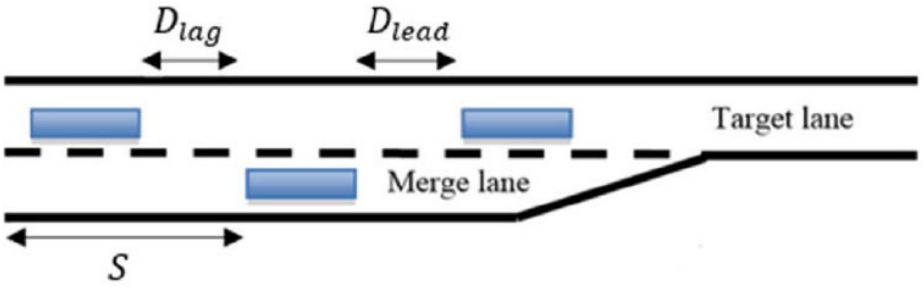


FIGURE 2.4: Schematic illustrating input variables [6].

The publicly available *NGSIM* vehicle trajectory dataset that consists of traffic conditions approaching congestion and congested conditions was used for model development and testing. This dataset is also used in our thesis. The best results is obtained when both Bayes and decision-tree classifiers were combined into a single classifier using a majority voting principle. The prediction accuracy was *94.3%* for *"non-merge"* events and *79.3%* for *"merge"* events. In a lane change assistance system, the accuracy of *"non-merge"* events is more critical than merge events. Misclassifying a *"non-merge"* event as a *"merge"* event could result in a traffic crash, whereas misclassifying a *"merge"* event as a *"non-merge"* event would only result in a lost opportunity to merge. Sensitivity analysis performed by assigning higher misclassification cost for *"non-merge"* events resulted in even higher accuracy for *"non-merge"* events but lower accuracy for *"merge"* events. The cost of misclassification can be treated as a surrogate to driver conservativeness. The greater the cost, the more conservative or less aggressive a driver is in working toward the gap to change lane. Hence, the accuracy of the proposed model is limited to *96.7%* and is very sensitive regarding the cost of *"non-merge"* misclassification.

As stated above, a model based on the theory of gap acceptance can only be used in the auxiliary acceleration lane. Therefore, it is more adapted for congested traffic and low-speed scenario. Such a model cannot be used in high-speed highway scenario since it cannot predict drivers' behaviors in the merge lane, before that the merging vehicle reaches the auxiliary acceleration lane. This is mandatory in the high-speed scenario to have a sufficient time lapse for decision-making. Moreover, the critical gap is obtained using an analytic model that assumes a probabilistic distribution of the critical gap values for the population of drivers. Hence, this model depends on the characteristics of the traffic participant, which differs from one driver to another. Using such model in practice expects that all the drivers behave in exactly the same way, which is not true in the real world.

Reference [7] studies driver behavior at the freeway-ramp merging areas to understand the merging process from the driver's perspective. The data collection undertaken for

their study entails observations of thirty-one participants driving an instrumented vehicle and simultaneous video observations of the freeway during experiments. The merging maneuvers were categorized as *"free"*, *"cooperative"* and *"forced"* merge, depending on the degree and the type of observed interaction between the ramp and the freeway vehicle. Merging maneuvers during both uncongested and congested conditions were observed. In uncongested conditions, participants were involved in *273* maneuvers as the ramp vehicle and *109* as the through vehicle. In congested conditions, participants were involved in *42* maneuvers as the ramp vehicle and *3* maneuvers as the through vehicle. In uncongested conditions, the participants performed all types of merging maneuvers, the majority of which were *"free"* maneuvers. When participants received cooperation, usually it was through deceleration rather than lane-changing. In congested conditions, *"free"* merges were not observed while participants were involved mostly with *"cooperative"* rather than *"forced"* merging. In almost all cases, the interacting vehicles would decelerate, possibly because gaps were not available on the inside lane due to dense traffic conditions.

Also, this work studies the effect of geometry and ramp design on merging. Observations of the participants merging on the four ramp junctions shown in Figure 2.5 revealed interesting findings related to their merging positions and speeds. It was found that compared to *"parallel-type"* on-ramps drivers used a higher percentage of the acceleration lane length on the *"tapered"* on-ramps before merging. It was also found that the average merging speed is higher in *"tapered"* than *"parallel"* type ramps and the observed variation is also higher.

Moreover, this work identifies the driver's behavior for each participant involved in the experiment. Three types of driver behavior were considered: *"aggressive"*, *"average"* and *"conservative"* behavior. For this task, the actual observed driver behavior was evaluated considering both qualitative and quantitative factors based on the field observations. The average age group for the entire sample falls between *35* and *45* years old. The *"aggressive"* drivers are in the *25-35* years old group, whereas *"average"* and *"conservative"* drivers are in the older group *(35-45)*. Also, the majority of men fall into the *"aggressive"* and *"average"* driver types, and women are more often found in the *"conservative"* driver type category.

Although this study was based on instrumented vehicle observations, it gives results in statistical form. It does not propose a specific parametric or analytic model for predicting drivers' intentions and behaviors at the on-ramp merging situation. Therefore, it cannot be implemented and used in the real world.

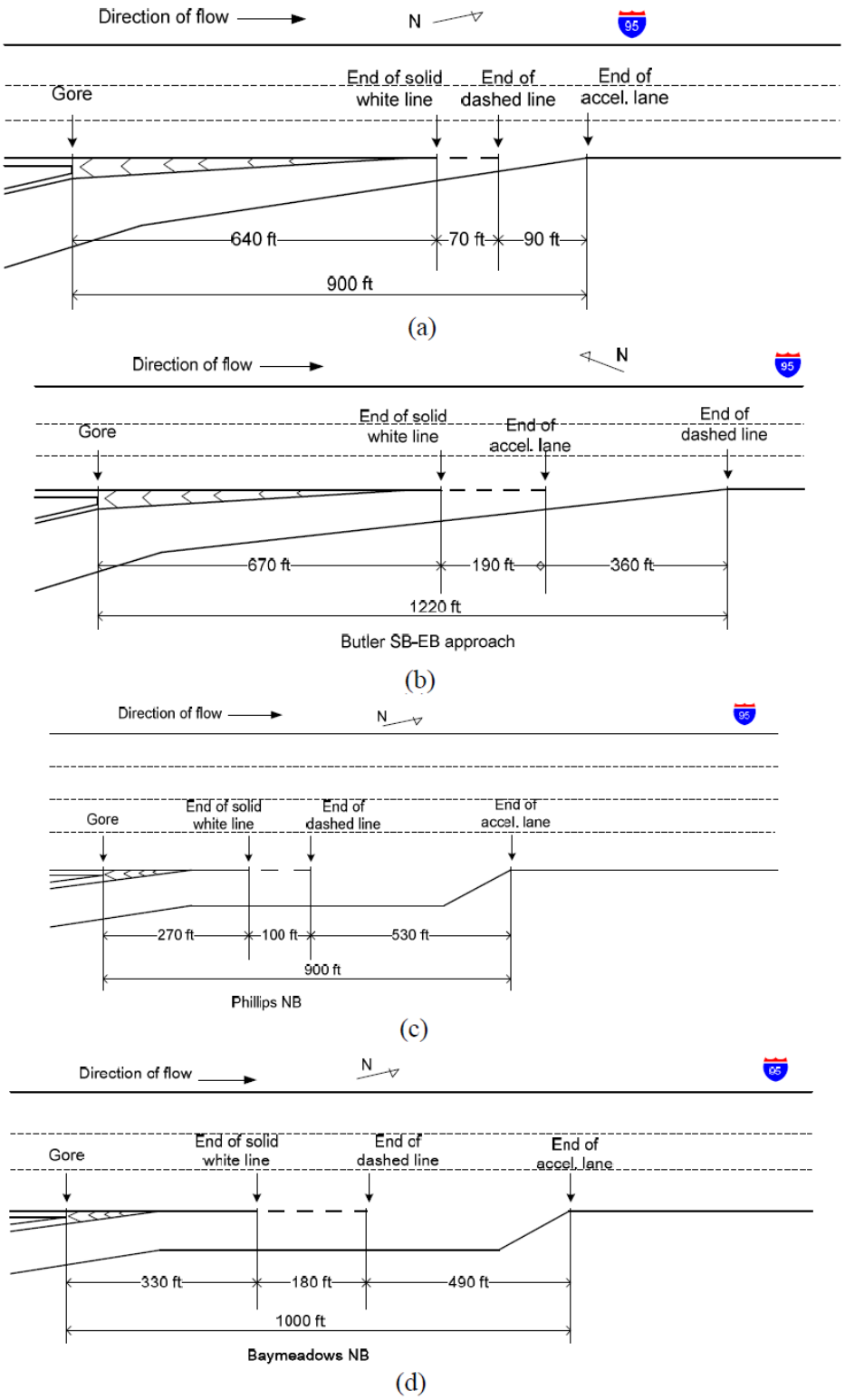


FIGURE 2.5: Geometric Characteristics of Tapered Entrance Ramps on *I-95* at (a) *J.T. Butler NB-WB* Approach, and (b) *J.T. Butler SB-EB* Approach, and Parallel Entrance Ramps *on I-95* at (c) *Phillips NB*, and (d) *Baymeadows NB* [7].

Reference [8] describes a microsimulation program developed to study freeway ramp

merging phenomena under congested traffic conditions. The results of extensive macroscopic and microscopic studies are used to establish a model for the behavior of merging drivers. It proposes a theoretical framework for modeling the ramp and freeway lag driver acceleration-deceleration behavior. The acceleration-deceleration profile of ramp vehicles in acceleration lanes is much more complicated than the types of behavior described by conventional car-following models. In fact, it shows that the fundamental psychophysical concept of car-following models (*Driver Response (t + T) = Sensitivity factors(t) * Stimulus(t)*, where t is the time, and T is the reaction time) remains appropriate, provided that the stimuli can be well specified. Three stimuli affect ramp driver behavior: speed relative to the freeway leader, speed relative to the freeway lag vehicle, and the distance from the freeway leader. Also, a theoretical framework for modeling the acceleration-deceleration behavior of a freeway lag vehicle (approaching the ramp area from the freeway) is then built. In congested traffic situations, four stimuli are considered for evaluating the freeway lag vehicle driver response: relative speed regarding the freeway leader, relative speed regarding the ramp vehicle, spacing regarding the freeway leader, and spacing regarding the ramp vehicle.

To overcome the major limitation of most of the existing microscopic simulation models that employ a global car-following model to capture the acceleration characteristics of drivers in all driving situations, the authors of reference [8] propose a Freeway Merging Capacity Simulation Program (*FMCSP*) capable of modeling the complex acceleration characteristics of ramp drivers in the acceleration lane and the significant interaction of ramp vehicles and freeway vehicles in congested conditions. A *C++* programming platform and a periodic sampling method at intervals of *0.05 s* is used for this microsimulation model. The *FMCSP* simulation includes the merging section and the upstream/downstream sections. The validation of *FMCSP* was performed at both microscopic and macroscopic levels using the traffic flows and lane-changing maneuvers observed at the *Hamazaki-bashi* merging section (*Japan*). In the microscopic analysis, trajectories from the *FMCSP* were compared with those from the field data (Fig. 2.6).

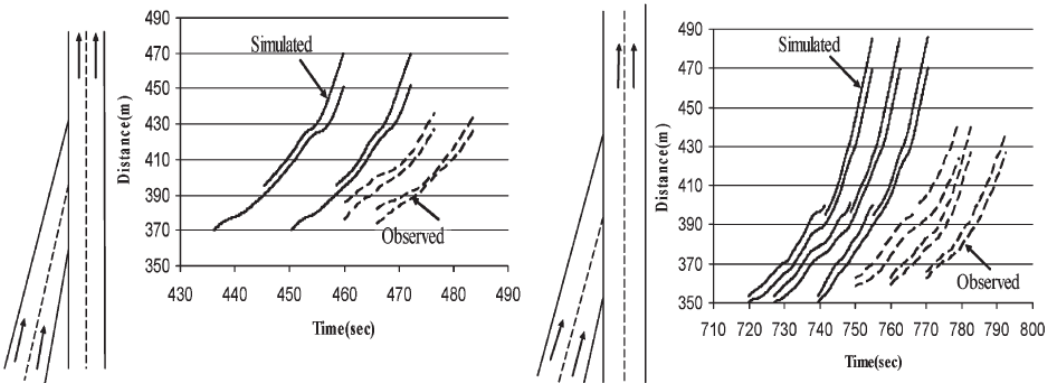


FIGURE 2.6: Comparison of observed and simulated trajectories at *Hamazaki-bashi*. (Left) Ramp lane *1*. (Right) Ramp lane *2*. [8]

Figure 2.6 shows a comparison between the simulated and observed trajectories of vehicles for ramp lanes *1* and *2* respectively. Each pair of lines in this figure represents the ramp vehicle and its freeway lead vehicle. The slopes of the trajectory lines (speeds) for the simulated vehicles before and after the merging process are consistent with the observed slopes. Also, the average speeds of the ramp vehicle and its freeway leader, the average time (headway) between the ramp vehicle and its freeway leader, and the lane-changing maneuvers of vehicles are in good agreement between simulation and real-world data. Finally, the developed simulation program was applied successfully to investigate a variety of freeway and ramp merging strategies. These strategies could reduce the high incident risk involved in lane-changing maneuvers.

Although the proposed framework models accurately the acceleration-deceleration behavior and shows good consistency with real-world data, it is limited only to the auxiliary acceleration lane of the on-ramp situation and under congested traffic situations. It has not been designed for high-speed highway on-ramp. Moreover, the framework is an analytical model which has a deterministic form. Hence, it cannot model uncertainties relative to drivers' behaviors diversity. A probabilistic model is more adapted for Modeling the behavior of on-ramp merging drivers.

Reference [9] uses a gated branch neural network (*GBNN*) for mandatory lane changing (*MLC*) suggestions in highway on-ramp situation. The real-world data-set *U.S. Highway 101 (US 101)* and *Interstate 80 (I-80)* are utilised from the *Federal Highway Administration's Next Generation Simulation (NGSIM)* program. The architecture of *GBNN* is shown in Figure 2.7. It is composed of two stages. Stage *I* is for data pre-processing and stage *II* includes a compact neural network. In the first stage, *16* features are extracted as inputs to the algorithm as listed in Table 2.1.

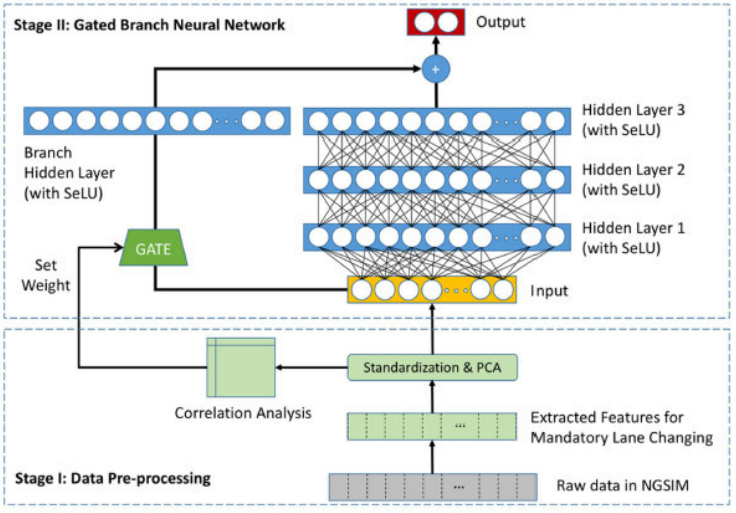


FIGURE 2.7: Data flow and architecture of the proposed *GBNN* [9].

NO.	Features	Unit	Meaning
1	Δx_ego	feet	lateral coordinate deviation of the merging vehicle with respect to the centre line of the merging lane
2	y_ego	feet	longitudinal coordinate of the merging vehicle with respect to the left-most entry edge, where the vehicle enters into the study area
3	v_ego	feet/s	velocity of the merging vehicle
4	a_ego	feet/s^2	acceleration of the merging vehicle
5	d_lead	feet	longitudinal gap between the front vehicle in merging lane and the merging vehicle
6	Δv_lead	feet/s	velocity difference between the front vehicle in merging lane and the merging vehicle
7	Δa_lead	feet/s^2	acceleration difference between the front vehicle in merging lane and the merging vehicle
8	d_lag	feet	longitudinal gap between the rear vehicle in merging lane and the merging vehicle
9	Δv_lag	feet/s	velocity difference between the rear vehicle in merging lane and the merging vehicle
10	Δa_lag	feet/s^2	acceleration difference between the rear vehicle in merging lane and the merging vehicle
11	d_lead_t	feet	longitudinal gap between the front vehicle in target lane and the merging vehicle
12	Δv_lead_t	feet/s	velocity difference between the front vehicle in target lane and the merging vehicle
13	Δa_lead_t	feet/s^2	acceleration difference between the front vehicle in target lane and the merging vehicle
14	d_lag_t	feet	longitudinal gap between the rear vehicle in target lane and the merging vehicle
15	Δv_lag_t	feet/s	velocity difference between the rear vehicle in target lane and the merging vehicle
16	Δa_lag_t	feet/s^2	acceleration difference between the rear vehicle in target lane and the merging vehicle

TABLE 2.1: Features extracted from *NGSIM* for modelling an *MLC* at on-ramps of highways [9].

Figure 2.8 shows the effects of different correlation methods on the *"non-merge"* accuracy and *"merge"* accuracy.

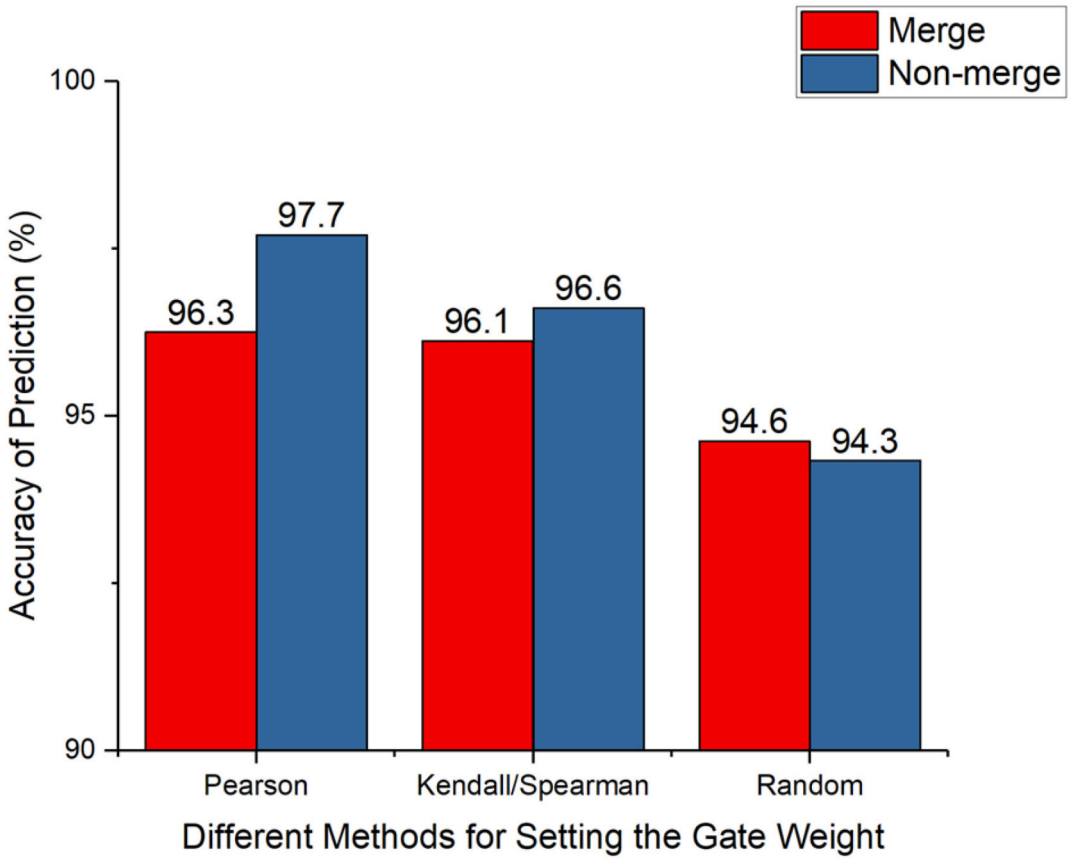


FIGURE 2.8: Comparison of different correlation methods (*Pearson*, *Kendall/Spearman*, and *random*), which are used for setting the weights of gate, on the effects of the prediction accuracy of *"non-merge"* and *"merge"* events [9].

Results show that as the number of neurons increases, the accuracy of *"non-merge"* events increases and achieves the highest value of *97.7%* at *384* neurons. The accuracy of *"merge"* events is not the highest at that point, but still has a good value of *96.3%*. Since the accuracy of *"non-merge"* events is prior due to its relation to driving safety, *384* neurons are chosen. Hence, the proposed gated branch neural network (*GBNN*) algorithm achieves limited accuracy in predicting both *"non-merge"* events (*97.7%*) and *"merge"* events (*96.3%*), outperforming the state-of-the-art binary classifiers reported in *MLC* applications.

Although the proposed model gives good accuracy, it is designed for congested situations. Moreover, it can be used only in the auxiliary acceleration lane and cannot predict drivers' behaviors at any point from the merge lane to be used for high-speed on-ramp merging scenarios.

2.3 Decision making

In recent years, many works have focused on either centralized or decentralized approaches for coordinating connected and autonomous vehicles (*CAVs*) in highway on-ramp merging situations.

2.3.1 Classical methods

Authors of reference [10] use the idea of a slot-based approach that employs cooperation between vehicles within the main motorway as well as between motorway and on-ramp vehicles to achieve a highly efficient merging. A slot based traffic management system (*TMS*) is based on the concept of slot S which is defined as $S = \{z, p, t, b\}$, where z represents the size of the slot, p represents its position (including the lane number) at time t and b the predefined behaviour as a constant speed. It is the task of the *TMS* to generate slots at a specific frequency and to provide the slot information to vehicles using vehicle-to-infrastructure (*V2I*) communication. The slot generation frequency determines the headway between slots. During the *"journey"* of a slot along the motorway, the slot's occupancy status can vary between *"free"* and *"occupied"*. The occupancy status of a slot is required when a vehicle wants to change its slot. This necessitates both knowing that the target slot is not currently occupied by another vehicle as well as knowing that another vehicle is not on its way to moving into that slot, as depicted in Figure 2.9. As such, the slot information is extended to $S = \{z, p, t, b, o\}$, where o represents the occupancy status of a slot.

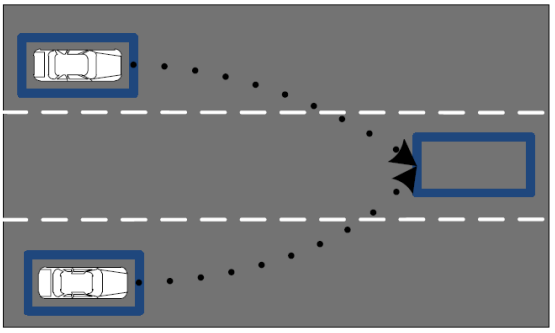


FIGURE 2.9: The slot-changing problem [10].

Merging on-ramp traffic with the slot-based traffic on the main road requires an algorithm that performs a mapping of on-ramp vehicles to empty slots on the main road. This is achieved in two phases: slot selection and moving into the slot. In the first phase, a suitable slot needs to be selected. Such a slot must be empty and located on the first lane of the motorway. Furthermore, at the time of selection, the distance

between the selection point and the actual point where the roads merge must be large enough to allow the on-ramp vehicle to get into the slot before the actual merging point is reached. A road-side unit (*RSU*) is located at the slot selection point and acts as a proxy between vehicles on the main road and the on-ramp vehicles. The *RSU* can sense the location of vehicles on the main road and can coordinate with them using *V2V* communication to determine suitable slots. When such slots are found, the *RSU* marks them as occupied for on-road vehicles, effectively blocking any attempt of any vehicle on the main road to move into that slot. On-ramp vehicles communicate with the *RSU* using *V2I* communication and request a slot. Once such a slot becomes available, the *RSU* communicates the slot information to the vehicle.

After the on-ramp vehicle has received a suitable slot, the moving into the slot phase commences. For this purpose, the vehicle creates a virtually identical slot clone, the only difference being that the cloned slot is located on the on-ramp rather than on the main road. The vehicle then moves into the cloned slot before the merging point is reached. Right before the merging point, the original and cloned slot are in perfect alignment and the vehicle performs a lane change towards the first lane of the main road and changes its target slot to the original slot, moves into this slot, and finishes the merging procedure. This process is described in Figure 2.10.

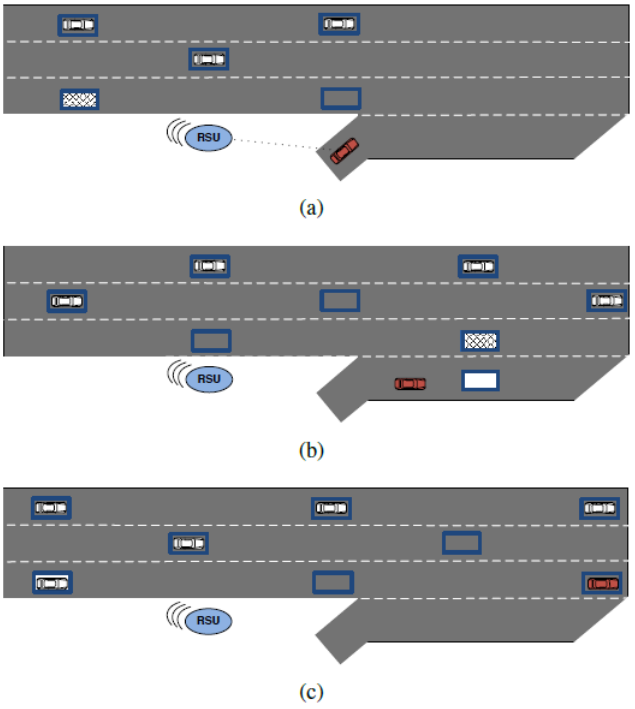


FIGURE 2.10: On-ramp merging [10].

The slot-based merging algorithm was evaluated and compared against a simulation of human drivers as performed by *VISSIM's Wiedemann '99 model*[3]. The efficiency of the merging was evaluated concerning:

- *Throughput:* the maximum number of vehicles that can merge from the on-ramp into the main road within an hour.
- *Delay:* the average delay experienced by vehicles on the on-ramp is calculated as the difference between the average travel time and the ideal travel time (no other vehicles in the network).

under:

- *Medium traffic conditions:* main road flow of *3600 vehicles/hour*.
- *Heavy traffic conditions:* main road flow of *4700 vehicles/hour*.

Under medium traffic conditions, the slot based driving without cooperation achieves a *41%* increase in throughput when compared to human drivers. Slot-based driving with cooperation performs better and achieves a *106%* increase in throughput compared to human drivers. The two algorithms perform even better under heavy traffic conditions compared to human drivers: *230%* and *452%* increase for slot-based driving without and with cooperation respectively. For manual drivers, the delay increases exponentially concerning the on-ramp flow. For the two slot-based driving algorithms the delay remains small and increases slightly as the on-ramp flow increases. The traffic on the main road increases the delay for human drivers but has very little effect on the other two slot-based approaches, thanks to the high throughput and efficient merging of the slot-based approach.

Although the slot-based approach increases traffic efficiency and achieves very high throughput and low delay on the highway on-ramp, it has many majors limitations. First, the slot-based driving model assumes that all vehicles on the road have identical capabilities and are equipped with RADAR, DGPS[4], wireless communication and are (semi)-autonomous. This assumption is not true in the real world since human-driven vehicles that are not equipped with such requirements will still be present in the road for decades. Moreover, the model validation does not take into account either real communication limits such as latency or fault tolerance issues such as malfunctioning vehicles.

[3]Originally formulated in *1974* by *Rainer Wiedemann*. This model is known for its extensive use in the microscopic multi-modal traffic flow simulation software, *VISSIM*.

[4]Differential Global Positioning System.

Last but not least, the approach was tested and validated using only simulator data. Real-world data from an on-ramp merging situation is mandatory to validate the slot-based approach.

Ramp metering is a common method used to regulate the flow of vehicles merging into freeways to decrease traffic congestion [11]. Ramp meters are traffic signals installed on freeway on-ramps to control the frequency at which vehicles enter the flow of traffic on the freeway. Ramp metering reduces overall freeway congestion by managing the amount of traffic entering the freeway and by breaking up platoons that make it difficult to merge onto the freeway. As seen in Figure 2.11, vehicles traveling from an adjacent arterial onto the ramp form a queue behind the stop line.

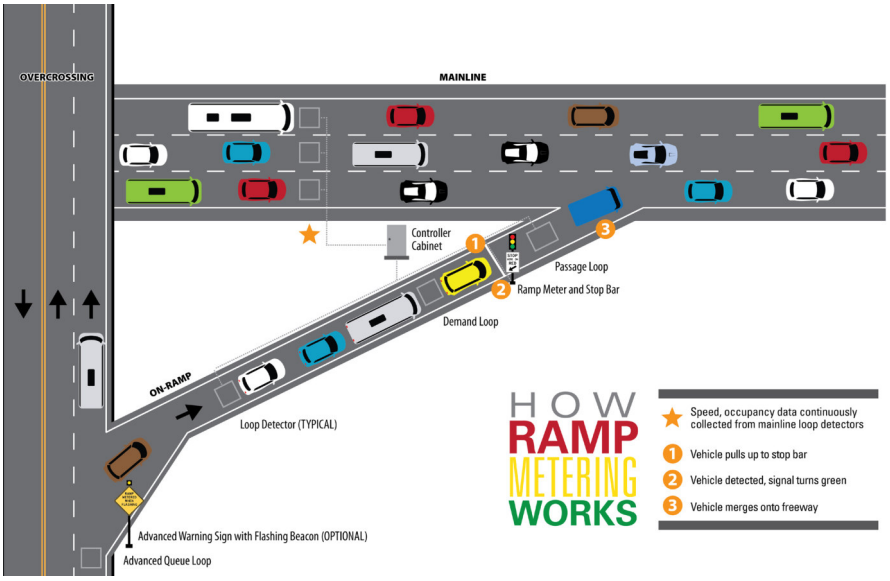


FIGURE 2.11: Ramp metering configuration [11].

The widespread benefits of ramp metering, relative to its costs, make it one of the most cost-effective freeway management strategies. Ramp metering reduces mainline congestion and overall delay while increasing mobility through the freeway network and traffic throughput. Also, Ramp meters help break up platoons of vehicles that are entering the freeway and competing for the same limited gaps in traffic. By allowing for smooth merging maneuvers, collision on the freeway can be avoided. Finally, Ramp meters smooth the flow of traffic entering the freeway so vehicles can merge with mainline traffic with minimal disruption to traffic flow. Eliminating prolonged periods of stop-and-go conditions due to congestion can reduce vehicle emissions and fuel consumption on the freeway.

Depending on the existing infrastructure, constraints, or objectives of the ramp metering program, various ramp metering approaches may be used. The following is a high-level overview of commonly used control approaches for ramp metering:

- **Single or Multi-Lane Metering:** Single lane metering allows only one vehicle to enter the freeway during each signal cycle. Multi-lane metering requires two or more lanes to be provided on the ramp and a signal head dedicated to each lane. After the stop bar, the lanes are required to merge into a single lane before merging onto the freeway.
- **Single or Dual Release Metering:** One vehicle per green (or single release metering), operates with a shorter green time than with two vehicles per green (or dual release) approach. The dual release allows for two vehicles to enter the freeway each cycle but requires a longer green time. The dual release metering approach usually increases ramp capacity under metering.
- **Freeway-to-Freeway Connections:** Ramp metering on freeway-to-freeway ramps is less common due to the high travel speeds and the perceived increased potential for vehicle collision because of vehicle queues where drivers may not expect them. Geometric constraints also exist such as limited sight distance along a curved roadway and limited provisions to provide the required storage for queued vehicles on-ramps. Freeway-to-freeway metering, if possible, can significantly improve the ability to manage traffic on a freeway because a greater share of entering traffic is controlled by meters.
- **Bypass Lanes:** Bypass lanes allow a specific class of vehicle (usually an *HOV*[5], a bus or, in some locations, a truck) to avoid delay at ramp meters and have the right of way to merge directly on to the freeway.

Although it has been shown that ramp metering aims at improving the overall traffic flow and safety on freeways, several challenges are associated with its usage. First, ramp metering is not possible in all ramp metering locations due to the configuration and structure of their ramps. Because ramp metering requires space for vehicles to merge into mainline traffic and to wait in a queue, not all ramp configurations are suitable for ramp metering. Moreover, despite the benefits of ramp metering, there are monetary costs for deploying and maintaining ramp metering systems.

Authors of reference [12] address the problem of optimally coordinating *CAVs* at merging roadways by ramp metering to achieve smooth traffic flow without stop-and-go driving.

[5] High-Occupancy Vehicle.

They propose an analytical closed-form solution using *"Hamiltonian"* analysis for vehicle coordination under the hard constraint of collision avoidance. They formulate the problem of optimal vehicle coordination at merging roadways in terms of fuel consumption under the hard constraint of collision avoidance, and then to derive online a closed-form solution in a centralized fashion (cf. Figure 2.12).

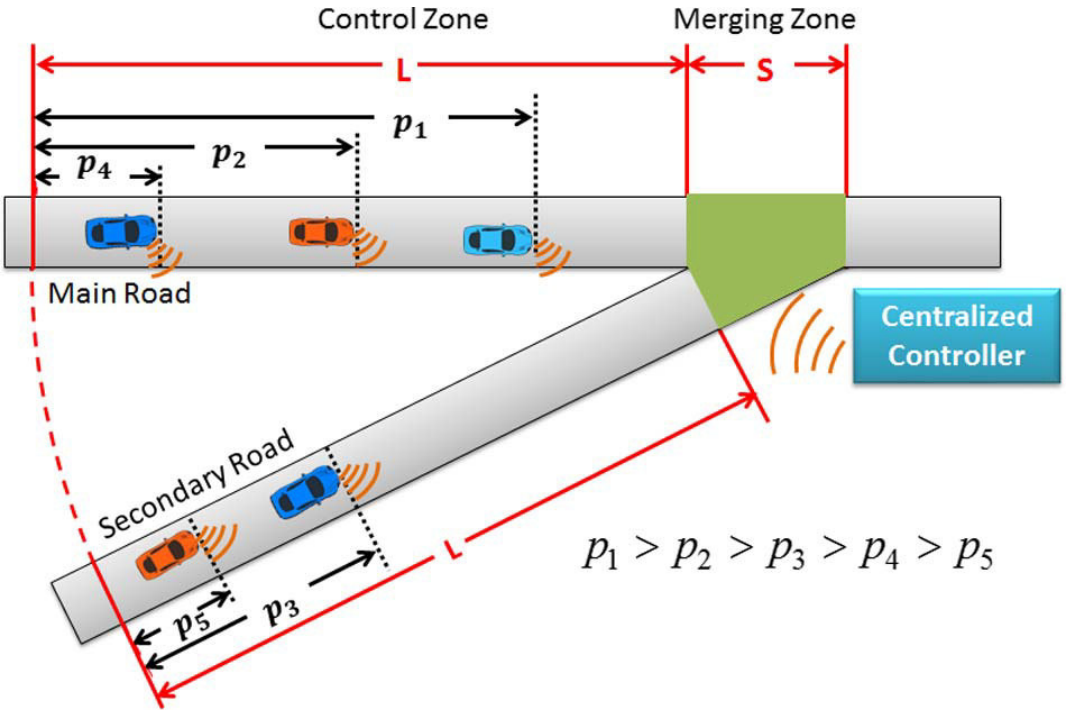


FIGURE 2.12: Merging roads with connected and automated vehicles controlled by a centralized controller [12].

Let $\mathcal{N}(t) = \{1,...,N(t)\}$ be the queue associated with the control zone. They model each vehicle i, $i \in \mathcal{N}(t)$, as a point mass moving along a specified lane with a state equation:

$$\dot{x}_i = f(t, x_i, u_i), \qquad x_i(t_i^0) = x_i^0 \tag{2.2}$$

where $t \in R^+$ is the time, $x_i(t)$, $u_i(t)$ are the state of the vehicle and control input, u_i^0 is the time that vehicle i enters the control zone, and x_i^0 is the value of the initial state. For simplicity, they assume that each vehicle is governed by a second order dynamics:

$$\begin{aligned} \dot{p}_i &= v_i(t) \\ \dot{v}_i &= u_i(t) \end{aligned} \tag{2.3}$$

where $p_i(t)$, $v_i(t)$, and $u_i(t)$, respectively, denote the position, speed and acceleration/deceleration (control input), respectively, of each vehicle i. Fuel consumption is expressed by the use of the polynomial metamodel as a function of the speed, v,

and control input, u:

$$\dot{f}_v = \dot{f}_{cruise} + \dot{f}_{accel} \tag{2.4}$$

where $t \in R^+$ is the time, $\dot{f}_{cruise}$=q_0+q_1.$v(t)$+q_2.$v^2(t)$+q_3.$v^3(t)$ estimates the fuel consumed by a vehicle traveling at a constant speed $v(t)$, and $\dot{f}_{accel}$=$u(t)$.(r_0+r_1.$v(t)$+r_2.$v^2(t)$) is the additional fuel consumption caused by acceleration $u(t)$. The polynomial coefficients q_n, n=0,...,3, and r_m, m = 0, 1, 2 are calculated from experimental data. The objective was to solve an optimization problem for each vehicle in the queue separately:

$$\begin{gathered} \min_{u_i} \frac{1}{2} \int_{t_i^0}^{t_i^m} u_i^2 dt \\ Subject\,to: (2), (4) \quad \forall i \in \mathcal{N}(t). \end{gathered} \tag{2.5}$$

For the analytical solution and online implementation of the problem, authors apply Hamiltonian analysis. This approach yields the optimal solution as long as the control input and speed of each vehicle is within the imposed limits. From the condition 2.5 and the state equations 2.3, the *"Hamiltonian"* function was formulated for each vehicle $i \in \mathcal{N}$(t) as follows:

$$H_i(t, x(t), u(t)) = L_i(t, x(t), u(t)) + \lambda^T.f_i(t, x(t), u(t)) \tag{2.6}$$

Thus

$$H_i(t, x(t), u(t)) = \frac{1}{2}u_i^2 + \lambda_i^P.v_i + \lambda_i^v.u_i \tag{2.7}$$

where λ_i^P and λ_i^v are the co-state components. The necessary condition for optimality is:

$$\frac{\partial H_i}{\partial u_i} = u_i + \lambda_i^v = 0 \tag{2.8}$$

and the optimal control is given by:

$$u_i + \lambda_i^v = 0, \quad i \in \mathcal{N}(t). \tag{2.9}$$

The effectiveness of the efficiency of the analytical solution was validated by simulating the merging scenario under *MATLAB*. The authors considered four case studies: *(1)* coordination of *4* vehicles, *2* for each road, *(2)* coordination of *30* vehicles, *15* for each road, *(3)* coordination of *30* vehicles assuming the vehicles on the secondary road reach the control zone at a lower speed of *11.2 m/s*, and *(4)* coordination of *30* vehicles that enter the control zone with *29 m/s*. In particular, optimal vehicle coordination improves overall fuel consumption by *52.7%* for the case study *2*, and *48.1%* for the case study *3* compared to the baseline scenario. The total travel time is also improved by *7.1%*, and *13.5%* respectively. For case *4*, the authors considered a scenario where the vehicles enter the control zone at *29 m/s*. The maximum and minimum speed limits inside

the control zone were specified to be equal to *31.3 m/s* and *22.4 m/s* respectively. In this case, however, the controller was unable to satisfy the safety constraints within the length of the control zone and the speed limits.

Although the proposed approach allows the vehicles to merge without creating congestions under the hard constraint of collision avoidance, it has some major limitations. First, it cannot satisfy safety merging in high-speed scenarios where the vehicles enter the control zone at *29 m/s* with a speed range of *[22.4 m/s, 31.3 m/s]*, which is the most critical situation at highway on-ramp. Moreover, the feasibility of the solution must be investigated. The control action is based on an analytic form solution that includes the dynamic model of each vehicle in the control zone. This approach is deterministic: it assumes that all the vehicles are connected, autonomous and equipped with this control strategy. This is far from the real-world situation, and cannot handle uncertainties related to other vehicles dynamics and behaviors.

Reference [13] proposed a new control strategy for merging tasks at highway junctions based on a predictive control model in which the decision entropy of drivers in the main lane was explicitly considered to be the cost function.

At first, the model of the driving behavior of the human-driven cars (Car E in Figure 2.13) in the main lane was developed. The model consists of two kinds of models; one reflects the decision-making of the driver whether to accept or reject the merging car (acceptance model), and the other one represents whether to accelerate or decelerate (motion model) based on the decision-making.

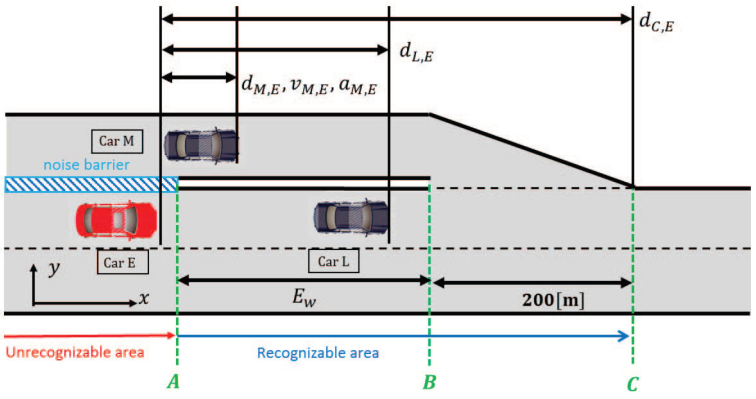


FIGURE 2.13: Observation of the driving behavior in the main lane [13].

The acceptance model is expressed as the stochastic discrete model which outputs the binary variables representing that the driver accepts or rejects the merging car based on the physical relationship between cars and using a logistic regression model. On the

other hand, the motion model expresses the motion control aspect of the driver of Car E, such as acceleration or deceleration. 5-fold cross-validation is applied to evaluate the estimation performance of this model using a virtual environment for data collection which is a driving simulator. The model shows an average success rate of *75.6%*.

The control strategy for the merging task was developed to minimize the decision entropy of the driver in the main lane considering the safety constraint. This control strategy is realized by the model predictive control (MPC), which minimizes:

$$J(t) = \sum_{k=1}^{K} \sum_{i}^{N} S_i(k|t) \tag{2.10}$$

where t is current time, k is predicted time index in the horizon, and i is the car index. K is the length of the predicted horizon, and N is the number of cars driving in the main lane. The cost function *J(t)* evaluates the accumulated entropy ($= S_i(k|t)$) of the acceptance state of drivers in the main lane over predicted horizon. Due to the nonlinearity of the formulated MPC problem, it is difficult to find the solution in real-time by standard nonlinear optimization computation. To overcome this problem, authors adopted a randomized model predictive control ($RMPC$) approach. Simulation results confirmed that the proposed method can produce more smooth merging speed pattern to the drivers in the main lane.

Although this work proposes the design of an advanced driver assistance system with cooperative behavior, it has major limitations. First, since the entropy function is nonlinear, the computational burden is an important problem in the implementation and must be addressed. The authors claim that the computational time must be dramatically shortened by using the code optimization, vectorization, and/or parallel computation technique. Moreover, authors assume that information (position and velocity) of cars in the main lane are measurable or given by vehicle-to-vehicle ($V2V$) communication although they are not explicitly controllable by the controller implemented on the merging car. This assumption implies that all vehicles present on the road are connected, which is not the case in the real world. Last, the proposed solution was trained and validated using only a virtual environment for data collection under a driving simulator. Using real-world driving data is mandatory before exploiting such a driving assistance system in the real world.

Reference [42] proposed an optimization-based ramp control strategy that can effectively coordinate all merging vehicles in freeway on-ramp situations and substantially

improve safety and efficiency. A nonlinear optimization model was developed to optimally coordinate the movements of freeway and ramp vehicles in a complex and realistic setting. The model takes the second-by-second accelerations of all vehicles as the decision variables and tries to maximize the total speed of all vehicles over the next short period. It also ensures that when a vehicle arrives at the merging point, the distance headways between it and adjacent vehicles are greater than a minimum value to guarantee safety. A decision interval of *10 seconds* is considered. This interval is further divided into *10 1-second* decision steps. At the beginning of each *1-second* decision step, each vehicle needs to decide its acceleration rate, which is a decision variable of the above optimization model. By optimizing these acceleration rates, the optimal control model aims to maximize the total speed of all merging vehicles in each decision step subject to the following constraints:

- Ensure that each vehicle maintains a non-negative speed that is no greater than the speed limit.
- Ensure that the distance between two consecutive vehicles in the same lane must be greater than a minimum value.
- Ensure that any pair of freeway and ramp vehicles maintains a safe distance at the end of the decision interval. This is achieved by projecting ramp vehicles onto the freeway using the merging point as the reference.
- Limit the acceleration rate changes of each vehicle between two consecutive time steps to prevent aggressive driving behaviors.
- Ensure that each vehicle maintains an acceleration rate that is within a limit range at each time step.

To evaluate to what extent the optimal control model can improve traffic operations at freeway on-ramps, a simulation platform was developed integrating *VISSIM*[6], *MATLAB*, and the *Car2X* module[7] in *VISSIM*. The proposed optimal ramp control strategy was solved using the *MATLAB* optimization toolbox. Three case studies are conducted to validate the effectiveness of the developed optimal control model and the simulation platform. The proposed optimal control algorithm was further compared with a do-nothing strategy and a gradual speed limit strategy for controlling a typical freeway on-ramp. Various levels of freeway and on-ramp traffic flows are considered: *low traffic flow*, *medium traffic flow* and *heavy traffic flow*. These three ramp control strategies were compared in terms of average delay time, average speed, and traffic throughput.

[6]The world's most advanced and flexible traffic simulation software.
[7]A module of *VISSIM* for network simulations.

When either the freeway or the on-ramp traffic flow is low, there is no significant difference among the three control strategies in terms of throughput. This is likely because ramp vehicles can all find a safe gap to join the freeway without causing long-standing queues. For the remaining considered scenarios, the optimal control strategy substantially outperforms the other two strategies. When the freeway traffic is heavy and the on-ramp traffic is light, the gradual speed limit strategy performs even worse than not considering any control. This gradual speed limit strategy works when the freeway traffic flow is low and the on-ramp has a medium to heavy traffic.

Although the results demonstrate the potential effectiveness of the proposed optimization-based ramp control strategy, this control strategy is based on a strict assumption that all vehicles are connected via Dedicated Short-Range Communications (*DSRC*), and controlled automatically by the control strategy (no human drivers). This assumption is not true in the real world. Moreover, the model was validated using only simulation tools that model traffic participants. Using real-world traffic data and considering non-connected vehicles are mandatory before using such a control strategy in the real world.

2.3.2 Reinforcement learning based methods

All previous works are classical rule-based approaches that include heuristics, optimal-control, and model predictive control. These methods require accurate modeling of the environment, where calculations are a burden. Moreover, most of these approaches assume that all the vehicles are connected and autonomous. Hence, they cannot handle uncertainties and unforeseen situations, which is not practical in real-world situations. Recently, studies investigate automated on-ramp merging using reinforcement learning. In reference [14], authors formulate the high-speed (*29 m/s*) merging problem within a reinforcement learning framework that treats state-space and action-space as continuous as in a real-world situation. Since in the on-ramp merging problem, it is difficult to prescribe an accurate model of the environment with a state transition matrix, the authors resort to *Q-learning*, a model-free approach, for finding an optimal driving policy. A *Q-function* was used to evaluate the long-term return *G(s,a)* based on the current and next step information *(s,a,r,s')*. *Q(s,a)* is called the action-state value and is approximated by neural networks for this use case where both state-space (driving environment) and action space (vehicle control) are continuous. Authors design the format of the *Q-function* approximator as a quadratic function to ensure that there is always a global optimal action for a given state at every moment. The state-space was defined to include the dynamics of the ego vehicle, the gap-front vehicle, and the

gap-back vehicle. The continuous state-space is therefore defined as:

$$s = (v_{ev}, p_{ev}, v_{gfv}, p_{gfv}, v_{gbv}, p_{gbv}) \tag{2.11}$$

where v_{ev} and p_{ev} are the speed and position of the ego vehicle; v_{gfv} and p_{gfv} are the speed and position of the gap front vehicle; v_{gbv} and p_{gbv} are the speed and position of the gap back vehicle. The action space is the vehicle longitudinal control (e.g. acceleration or deceleration). After the reinforcement learning agent takes an action in a given state, its impact on the environment is fed back as an immediate reward. The effect is reflected by the smoothness, safeness, and promptness of the merging maneuver. Smoothness represents the comfort of the merging maneuver and was measured by the absolute value of the acceleration. The safeness was estimated by the distance to the surrounding vehicles. The promptness was assessed by the time that the ego vehicle will take to complete the merging process, where the current vehicle speed was used to account for the contribution of promptness in the immediate reward. Consequently, the composition of the immediate reward was expressed by equations:

$$R(s,a) = R_1(acceleration) + R_2(distance) + R_3(speed) \tag{2.12}$$

$$R_1(acceleration) = f_1 * abs(acceleration) \tag{2.13}$$

$$R_2(distance) = f_2 * g_2(distance) \tag{2.14}$$

$$R_3(speed) = f_3 * speed \tag{2.15}$$

where f_1, f_2 and f_3 are factors accounted for each part of reward. The quadratic format of *Q-function* approximator was specified as follows:

$$Q(s,a) = A(s) * (B(s) - a)^2 + C(s) \tag{2.16}$$

where A, B and C are trainable parameters and designed with the neural network structure with environment state as inputs. An illustration is shown in Figure 2.14. There are two graphs concealed in this form of the *Q-function* approximator. One is the graph for obtaining an optimal action (a^*=*B(s)*) in a given state, where *B(s)* is learned based on the current state s. The other is the graph for calculating the *Q-value* for a given state and action.

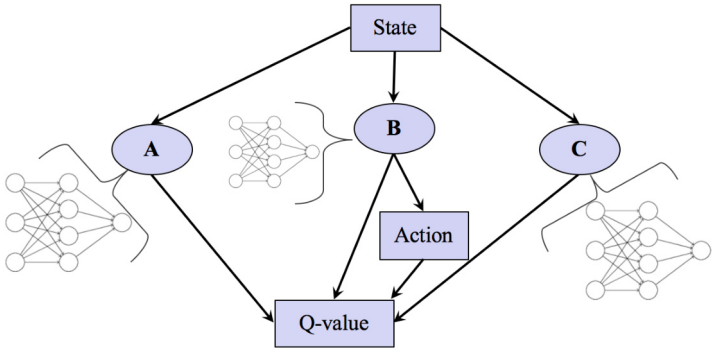


FIGURE 2.14: Graph of the *Q-function* approximator [14].

In the learning process, *Q-network* is updated with the following loss function.

$$Loss = \sum_{i=1}^{n} (r + \gamma \cdot \max_{a'} Q(s', a', \theta) - Q(s, a, \theta))_i^2 \tag{2.17}$$

where $r+\gamma \cdot \max_{a'} Q(s', a', \theta)$ is called the target *Q-value* and $Q(s,a,\theta)$ is called the predicted *Q-value*. θ is a set of *Q-network* parameters. When the agent is trained based on equation 2.17, stability issues and correlations in the observed sequence are factors affecting the learning performance. Hence, experience replay and a second *Q-network* techniques were used to alleviate the problem. The step-by-step learning procedure is shown in Figure 2.15.

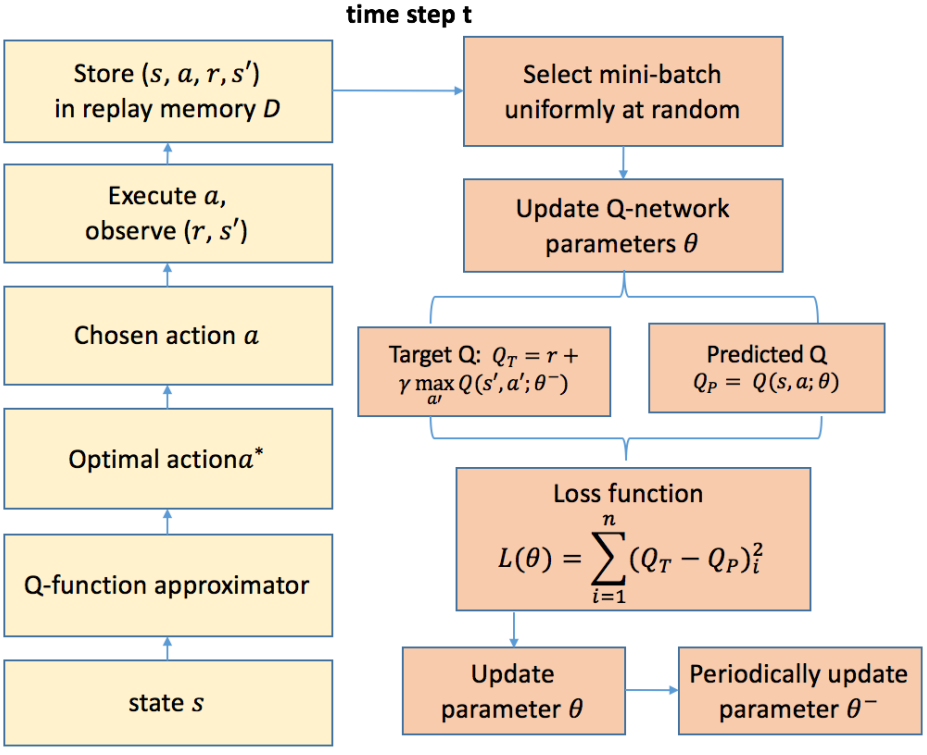


FIGURE 2.15: Reinforcement learning procedure [14].

The training procedure is set to *1,600,000* steps, during which around *8,000* ramp vehicles are performed ramp merging behavior. Figure 2.16 shows the total reward (named single total reward) of all the *8000* vehicles in the simulation.



FIGURE 2.16: Curve of single total rewards of ramp vehicles [14].

The authors claim that the single total rewards converge at the end of the training. However, when analyzing each individual reward, only the reward curves of distance to the gap-front vehicle and vehicle acceleration show apparent convergence. In contrast, reward curves of the distance to the gap-back vehicle and the acceleration show a higher level of fluctuations.

Although the authors propose a reinforcement learning approach to learn a safely, smoothly, and timely driving policy, their approach has some limitations and unrealistic considerations.
First, the method to select an appropriate gap is done in a deterministic way by estimating the arrival time to the merging section of the ego vehicle and of the other vehicles on the highway. The authors did not show details about how this estimation was made, because it is difficult in the real world to estimate arrival time due to the varying dynamics and behaviors of each vehicle on the road.
Moreover, The ego vehicle is supposed to be equipped with a suite of sensors including *LiDAR*, *RADAR*, *camera*, a digital map, *DGPS*[8] and *IMU*[9], and can gather the vehicle dynamic information of its own and its surrounding vehicles within a vicinity of *150 m* that was also assumed to be accurate enough to meet the implementation requirements. These assumptions are far from the realistic situations where the observation range may

[8]Differential Global Positioning System.
[9]Inertial Measurement Unit.

be partially occluded and the measurements are shortened, imprecise, or inaccurate. Moreover, Authors said that the optimal action is selected using *B(s)* network (cf. Figure 2.14). However, they did not show either how the network is trained as in the *actor-critic* algorithm and what guarantees the optimality of the selected action.
Last but not least, the authors did not show results details when testing the trained driving policy, such as the average merging speed. Moreover, they did not show details about the effects of merging on the other vehicles present on the road, such as the emergency brakings performed by the gap back vehicle. Such detail evaluates the cooperativeness of the learned driving policy.

The authors of reference [15] present a method for freeway merge based on multipolicy decision-making coupled with a reinforcement learning technique called *passive actor-critic* (*pAC*), which learns with less knowledge of the system and without active exploration. Multi-policy decision-making (*MPDM*) is a method to select the best policy in a set of applicable policies. Authors present a novel *MPDM* algorithm that utilizes *pAC*. *MPDM* determines control input by selecting a policy among multiple candidates using the scores of each policy. While the previous *MPDM* algorithm requires forward simulation to score each policy candidate, their proposed algorithm scores the candidates without the simulation, instead of using state values estimated with *pAC*. The pseudo-code of *MPDM* with *pAC* is shown in Algorithm *1*. It populates the set Π of available policies. A score c for each candidate, which is calculated using state value estimated with *pAC*, is added to the set of scores C. Finally, the policy associated with the minimum score is returned as the best policy.

Algorithm 1 *MPDM* Policy Selection with *pAC*.

---

1. Set Π of available policies.
2. $C \leftarrow \emptyset$.
3. **for** $\pi \in \Pi$ **do**
4. Set current state: $X_k \leftarrow x$.
5. Calculate score of a policy π learned by pAC:
6. $c \leftarrow \hat{V}_i(X_k)$.
7. $C \leftarrow C \cup \{<\pi, c>\}$.
8. **end for**
9. Choose the best policy $\pi^* \in \Pi : \pi^* \leftarrow argmin_{\pi \in \Pi}\ C$.
10. **return** π^*

While the *actor-critic* method usually operates using samples collected actively in the

environment, *pAC* finds a converged policy without exploration. Instead, it uses samples of passive state transitions and a known control dynamics model. The *pAC* follows the usual two-step approach of *actor-critic*: a state evaluation step (*critic*), and a policy improvement step (*actor*):

- **1) Critic:** Estimates the *Z-value* and the average cost from the linearized *Bellman* equation using samples under passive dynamics.
- **2) Actor:** Improves a control policy by optimizing the *Bellman* equation given the known control dynamics model, and the *Z-value* and cost from the critic.

The authors then apply Algorithm *1* to the problem of properly merging into a freeway. The algorithm learns a policy and a state value function to merge into a predetermined spot using *pAC* on a collected data set in advance. The algorithm then determines a merging spot from a set of candidates and control input with the learned model when an autonomous vehicle is driving on a merging ramp. They model the merging problem using a *3-car* system as shown in Figure 2.17.a. The state of the problem is sufficiently modeled using the following variables: $X = [dx_{12}, dv_{12}, dx_{10}, dv_{10}]^T$ where dx_{ij} and dv_{ij} denote the horizontal signed distance and relative velocity between cars i and $j \in [0, 1, 2]$. The state cost is designed to motivate *Car-0* to merge midway between *Car-1* and *Car-2*, and with the same velocity as *Car-1*.

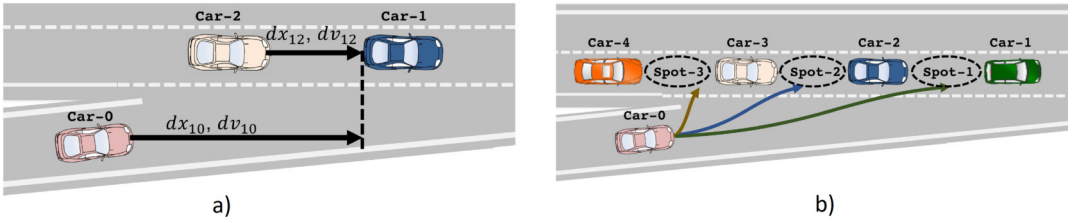


FIGURE 2.17: **a)** The *3-car* system for merging. Merging vehicle (*Car-0*) should ideally merge midway between the following vehicle (*Car-2*) and leading vehicle (*Car-1*). dx_{10} and dv_{10} denote *Car-0*'s relative position and velocity from *Car-1*. **b)** A typical freeway-merge situation. There are three mergeable spots: Spot 1, 2, and 3. The merging vehicle needs to determine a spot from a set of the candidates and controls input to merge [15].

The figure 2.17.b shows a typical freeway-merge situation. In this situation, there are three possible mergeable spots: Spot-1, 2, and 3. The *pAC* is utilized to learns the policy for merging into one of these spots, and the corresponding value function for that state is obtained. The best policy is then selected as in Algorithm *1*. The value function learned to merge into a predetermined spot can be used to calculate the score of any spot candidate because the *MDP* is the same and only the states are different between these candidates.

The authors evaluated the performance of their proposed approach with numerical experiments and real-world congested traffic data (*NGSIM*). The *pAC* with the neural network achieved *97%* success rate on the numerical experiments. In the real traffic data, *pAC* with the neural network achieved *93%* by combining an approximate nearest neighbor to mitigate data imbalance and sparseness. Evaluating on real-world congested traffic data, the combined *MPDM* with *pAC* achieved a *92%* success rate, which is comparable to merging success when the spot is selected by human drivers. The success rate of the proposed method is much higher than that of merging into a fixed spot specified in advance.

Although the novelty of the proposed approach and his good success rate, it has some limitations. First, it considers only congested freeway conditions. High-speed merging is a more critical driving scenario that should be considered to test this solution. Moreover, the *passive actor-critic* algorithm requires data under passive dynamics and accurate dynamics model of the vehicle. This control model should be known accurately in advance, which may increase the complexity when implementing this approach. Last, their approach selects the policy based on the assumption that surrounding vehicles behave in the same way on average. This assumption is far from reality where each driver has different behavior and driving style. Authors admit that the approach would not be able to cope with situations when a surrounding vehicle deviates significantly from the average, and this is also an important future challenge to consider.

The authors of reference [16] present a reinforcement learning approach to learn how to interact with drivers with different cooperation levels. They focus on dense traffic situations where cars drive slowly (around *5 m/s*) and very close to each other (the gaps can be below *2 m*). The merging scenario was modeled as a Partially Observable Markov decision process (*POMDP*). The complete state of the environment consists of the collection of the individual states of each vehicle present. The physical state of each vehicle corresponds to distance to the merge point, longitudinal velocity, acceleration, and cooperation level (c). The behavior is characterized by this parameter c, and the state of the ego vehicle (the controlled agent) does not contain this behavior parameter. The authors assume that the ego vehicle has limited sensing capabilities and cannot measure the internal states of other vehicles. Hence, the observation is restricted as illustrated in Figure 2.18 to the longitudinal position and velocity of four neighbor cars: the front neighbor of the ego vehicle, the vehicle right behind the merge point, the rear neighbor, and front neighbor of the projection of the ego vehicle in the main lane.

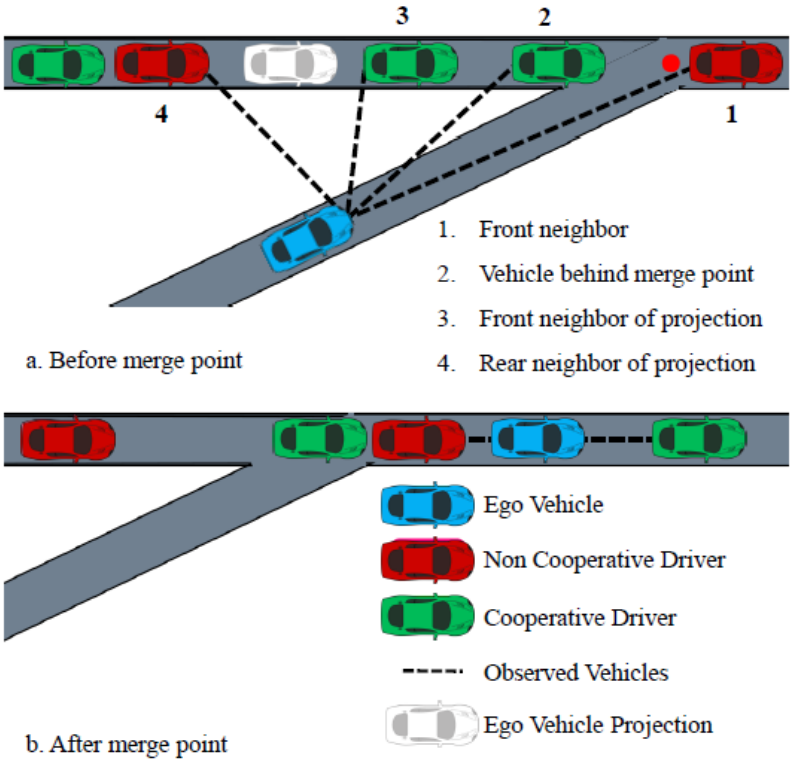


FIGURE 2.18: Illustration of the vehicles observed by the ego vehicle. The observation vector (or feature vector) contains information on the position and velocity of the observed vehicles [16].

The ego vehicle controls its motion by applying a change in acceleration. At each time step, the acceleration is updated as $a_t = a_{t+1} + \Delta a$, where Δa is the action chosen by the agent in the set *-1* m/s^2, *-0.5* m/s^2, *0* m/s^2, *0.5* m/s^2, *1* m/s^2. The agent can also apply a hard braking action and releasing action which instantaneously sets the acceleration to *-4* m/s^2 and *4* m/s^2 respectively. Hence, the action space is discrete with *7* possible actions at each time step. The reward function was designed such that the optimal policy maximizes safety and efficiency. The agent receives a penalty of *-1* for colliding with other traffic participants and receives a bonus of *+1* for reaching a goal position defined *50 m* after the merge point. Authors model the behavior of drivers in the main lane by introducing a cooperation level $c \in [0,1]$. This parameter is a scalar controlling the reaction to the merging vehicle state. $c = 1$ represents a driver who slows down to yield to the merging vehicle if he/she predicts that the merging vehicle will arrive ahead of time. $c = 0$ represents a driver who completely ignores the merging vehicle until it traverses the merge point. Since the cooperation level cannot be directly measured, the ego car maintains a belief in the cooperation level of the observed drivers. Authors propose a simple belief updater, that is acting as if the cooperation level was binary although it can take a continuous value. The belief at time t is composed of the fully observable part of the state, o_t, and θ_i for $i = 1...n$, where n is the number of

observed drivers. θ_i^t represents the probability that vehicle i has a cooperation level of *1*. At time $t+1$, the ego vehicle observes o_{t+1} and updates its belief on the cooperation level of vehicle i as follows:

$$\theta_{t+1}^i = \frac{Pr(o_{t+1}|o_t, c_i = 1)\theta_t^i}{Pr(o_{t+1}|o_t, c_i = 1)\theta_t^i + Pr(o_{t+1}|o_t, c_i = 0)(1-\theta_t^i)} \tag{2.18}$$

Equation 2.18 consists of simulating forward the previous state with the two possible hypothesis: $c_i = 1$ and $c_i = 0$, and comparing the outcome with the current observation. Authors propose to use this belief state as input to the Deep Q-network (DQN) reinforcement learning policy.

The proposed approach was simulated under a dense traffic environment (average speed of *5 m/s*). The reinforcement learning (RL) policy without access to information on the cooperation level had *2%* collisions at test time and the RL policy that has information about cooperation level (either directly or through the prediction given by the belief state) performed similarly with around *0.6%* collisions. Authors said that previous works have shown that only relying on deep RL is not sufficient to achieve safety. Moreover, they claimed that the deployment of those policies would require the addition of a safety mechanism.

Although the study confirms that an autonomous agent can benefit from reasoning about the interaction with other drivers, it has some main limitations. First, the authors consider only urban driving environments with dense traffic where the vehicle speeds are around *5 m/s*. It would be more interesting and safety-critical if they used and validated their approach in the high-speed highway scenario, which is more critical. Moreover, the driver cooperation level was approximated using a simple binary state predictor, which might not represent accurately how human drivers behave in the real world. Authors admitted that it is mandatory to consider more complex filtering techniques such as multi-hypothesis filters, interacting multiple models, or data-driven approaches to estimate the driver cooperation level from observation. Another key limitation of this approach is the use of a Deep Q-Network, which provides only discrete action control. This is not practical for real-world implementation where the state-action space is continuous. Last, the authors use a basic sparse reward that penalizes collision and gives a bonus for successful merging. The reward design did not consider some essentials safety criteria such as distance from the front and the rear vehicles.

The work conducted in reference [17] studies high-speed on-ramp merging decision-making and control for an automated vehicle using deep reinforcement learning. It

considers no vehicle to everything (*V2X*) wireless communication and the merging vehicle relies on its sensors to obtain the states of other vehicles and the road information to merging from on-ramp to the main road. The authors use the deep deterministic policy gradient (*DDPG*) method to train the merging policy. *DDPG* assumes a deterministic policy and outputs continuous actions for decision-making and control. The merging environment is created in the Simulation of Urban Mobility (*SUMO*) simulator, where a control zone is defined for the merging vehicle that is *100 m* to the merging point on the on-ramp and *50 m* from the merging point on the main road, as shown in Figure 2.19.

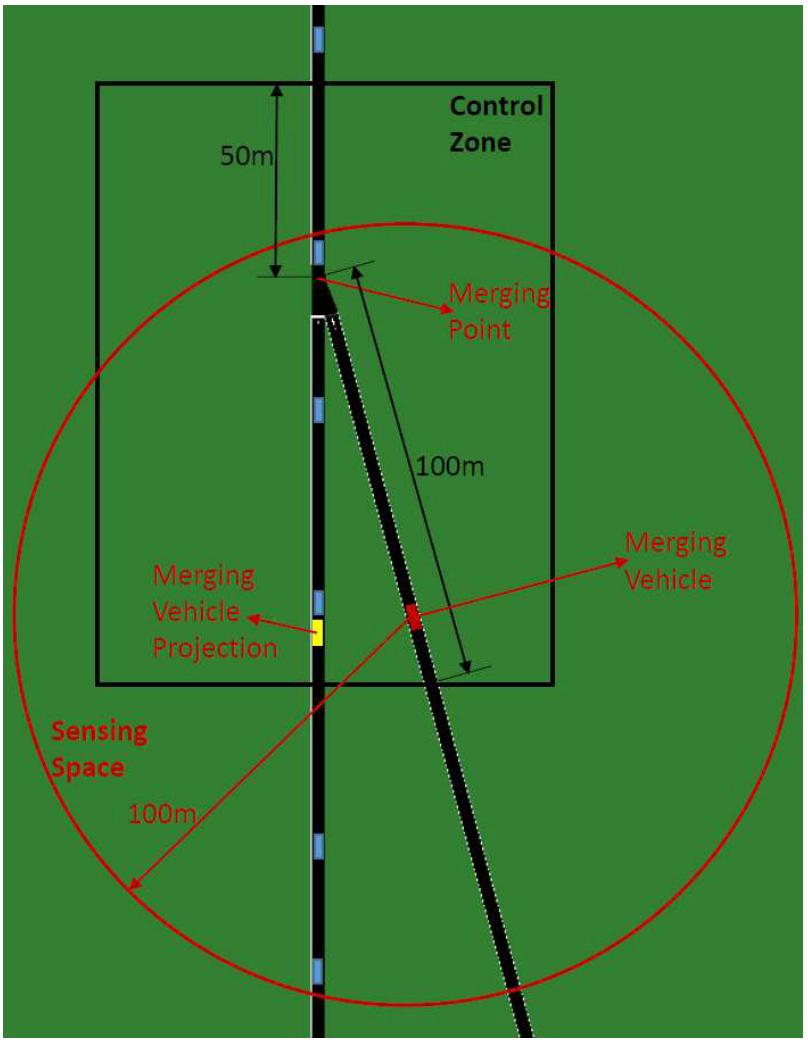


FIGURE 2.19: Schematic for merging [17].

The authors design a reinforcement learning framework such as the environment state is composed of the distance to the merging point, velocity and acceleration of the merging vehicle projection, two preceding vehicles, and two following vehicles. The action of the reinforcement learning framework is the acceleration control input to the merging vehicle, while the reward is designed to motivate the vehicle to merge midway between the

preceding and the following vehicles, with the same speed as the first preceding vehicle. The reward gives a bonus for successful merging, and penalize collision, stop, and hard braking of the first following vehicle.

The proposed *DRL* framework trained the merging vehicle for *1 million* simulation time steps wherein a reasonable convergence of the cumulative episode reward was observed. The trained policy was tested for another *1 million* simulation time steps. Table 2.2 summarizes the total number of episodes and the numbers of stops, collisions, successes, and the emergency brakings times wherein the merging vehicle decelerates to *-9* m/s^2.

Total number of episodes	16975
Number (%) of stops by merging vehicle	0 (0%)
Number (%) of emergency brakings by first following vehicle	197 (1.161%)
Number (%) of collision involving merging vehicle	1 (0.006%)
Number (%) of successful episodes	16024 (99.994%)
Average speed of merging vehicle	24.22 m/s

TABLE 2.2: Testing results [17].

Table 2.2 shows that the trained policy resulted in zero stops during testing. Additionally, the average speed is *24.22 m/s*, which is relatively high. This means that the trained agent sought to minimize the travel time. The merging vehicle made only *1 collision*. However, it caused the first following vehicle to brake emergently *197* times.

Although this work proposes a decision-making and control solution via *DDPG*, it has main limitations that do not allow it to be used in the real world. First, the trained policy did not eliminate all the collision cases. As safety for the learned policy is not guaranteed, authors admit that trajectory prediction or a safe controller may be needed to exclude unsafe actions. Moreover, the trained driving policy shows many hard brakings performed by the first following vehicle during the evaluation of the model (*1.161%*), although the reward penalizes such maneuvers. This means that the learned policy is not cooperative with humans' driven vehicles, even if their behaviors are implicitly considered in the reward function through the hard braking penalty. Hence, a deep reinforcement learning framework is unable to learn implicitly a *"cooperative"* driving policy.

To summarize, many works studied the highway on-ramp merging problem using either classical deterministic methods such as slot-based and optimal-control or novel deep reinforcement learning algorithms such as *DDPG*. Yet, no previously work has shown perfect performances that allow the implementation of such systems in the real world.

2.4 Conclusion

We surveyed the different approaches proposed for drivers' intentions estimation in a highway on-ramp situation. These approaches are summarized in the following table:

Approach	Limitations
[3] Bayesian network and logistic regression (*87%* accuracy)	- Trained and validated using only simulator data. - Dependencies on driver's driving style.
[4] Probabilistic Graphical Model (*PGM*) (*92.4%* success rate)	- Dependencies on estimating the time to arrival to the merging point. - Dependencies on driver's driving style.
[41][5][6] The theory of gap acceptance (*98%* accuracy)	- Used only in the auxiliary acceleration lane.
[7] Statistical form	- Not a parametric nor an analytic model.
[8] Theoretical framework for modeling freeway ramp merging behavior	- Used only in the auxiliary acceleration lane. - Analytical form that cannot model drivers' behaviors diversity and uncertainties.
[9] Gated branch neural network (*GBNN*) (*97.7%* accuracy)	- Used only in congested situations. - Used only in the auxiliary acceleration lane.

TABLE 2.3: Drivers' intentions estimation approaches at highway on-ramp situation.

As shown in Table 2.3, the best accuracy was obtained using the theory of gap acceptance or the Gated branch neural network. These techniques can be used only in the auxiliary acceleration lane under congested traffic. Yet, no previously work has proposed a prediction model that can be implemented at the off-board infrastructure, and which uses real-world traffic data.

We also reviewed the different methods that were proposed for decision-making at the highway on-ramp situation. These methods are summarized in Table 2.4. As shown in this table, the first works that were proposed for highway on-ramp decision-making were classical rules-based approaches that include heuristics, optimal-control, and model predictive control. These methods require accurate modeling of the environment, where calculations are a burden. Moreover, they cannot handle uncertainties and unforeseen situations, which is not practical in real-world situations. Recently, studies investigate automated merging using deep reinforcement learning. Although these methods learn driving policy directly from trial-and-errors without the need for an accurate system model, they did not guarantee safe and cooperative driving using only reward formulation. Yet, no previously work has shown perfect performance for highway on-ramp merging so that it allows the implementation of such driving systems in the real world.

	Method	Limitations
Rules-based methods	[10] Slot-based approach	- Assumes that all the vehicles are autonomous. - Tested and validated using only simulation data.
	[11] Ramp metering	- Not adapted to all ramp locations. - Monetary cost.
	[12] Analytical solution using Hamiltonian analysis	- Cannot satisfy safety merging at high-speed scenario ($29\ m/s$). - Control action based on deterministic model that cannot handles uncertainties.
	[13] Predictive control model	- Computational burden. - Assumes that all the vehicles are connected. - Tested and validated using only simulation data.
	[42] Optimization-based ramp control strategy	- Assumes that all the vehicles are autonomous and connected. - Tested and validated using only simulation data.
Reinforcement learning methods	[14] Q-learning	- Gap selection using deterministic method. - Vehicle perception range is not realistic. - Some implementation and results details are not shown.
	[15] Passive actor-critic (pAC)	- Used only in congested freeway conditions. - Requires accurate model for the control dynamic of the vehicle. - Expectation that surrounding vehicles behave in the same way on average.
	[16] Deep Q-Network with cooperation level	- Used only in urban driving environments with dense traffic. - Simple binary state predictor to estimate the driver cooperation level. - Provides only discrete action control. - Uses basic sparse reward.
	[17] Deep deterministic policy gradient ($DDPG$)	- Did not eliminate all the collision cases. - Many hard brakings performed by the first follower vehicle.

TABLE 2.4: Decision-making methods at highway on-ramp situation.

Chapter 3

Drivers' intentions estimation

3.1 Introduction

Despite connected and autonomous vehicles (*CAVs*) technology may improve driving safety and efficiency, it takes time to build a whole connected system (especially vehicle-to-vehicle communication). Human-driven vehicles, that cannot be controlled directly, will be present on the road. Their behaviors should be taken into account by the autonomous driving system at the highway on-ramp. Since the infrastructure increases the perception range and the reliability compared to on-board vehicle sensors [2][32], we propose to use an off-board model to predicts drivers' intentions at the highway on-ramp. This model should be implemented in the road-side unit (*RSU*).
For that, we will first review the different probabilistic classifiers that will be used to train this model. We will then train and validate it using a real-world database that extracts vehicles' information at the highway on-ramp through infrastructure sensors. The performances of these classifiers will be discussed and compared. The real-time application of this approach will be then validated. Finally, a comparison between such off-board implementation and an on-board model will be provided.

3.2 Drivers' intentions model

A survey of existing methods for motion prediction was cited in [18]. These approaches for motion modeling and prediction were classified into three levels with an increasing level of abstraction as shown in Figure 3.1.

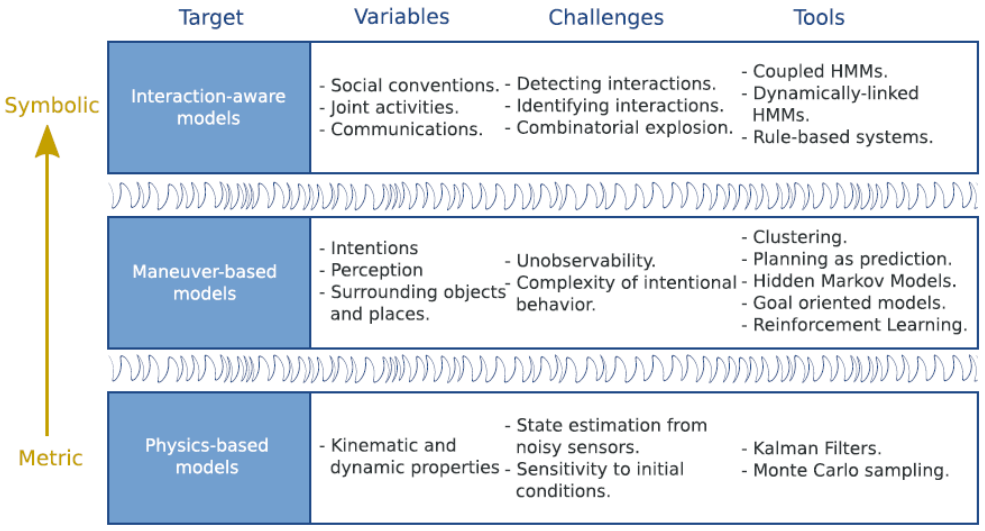


FIGURE 3.1: Motion modeling overview [18].

1. Physics-based motion models: *Physics-based* motion models represent vehicles as dynamic entities governed by the laws of physics. Future motion is predicted using dynamic and cinematic models linking some control inputs (e.g. steering, acceleration), car properties (e.g. weight), and external conditions (e.g. friction coefficients of the road surface) to the evolution of the state of the vehicle (e.g. position, heading, speed). These models remain the most commonly used motion models for trajectory prediction and collision risk estimation in the context of road safety. The models are more or less complex depending on how fine the representation of the dynamics and kinematics of a vehicle is, how uncertainties are handled, whether or not the geometry of the road is taken into account,etc.

Since these models only rely on the low-level properties of motion (dynamic and kinematic properties), *Physics-based* motion models are limited to short-term (less than a second) motion prediction. Typically, they are unable to anticipate any change in the motion of the car caused by the execution of a particular maneuver (e.g. slow down, turn at a constant speed, then accelerate to make a turn at an intersection) or changes caused by external factors (e.g. slowing down because of a vehicle in front).

2. Maneuver-based motion models: *Maneuver-based* motion models represent vehicles as independent maneuvering entities, i.e. they assume that the motion of a vehicle on the road network corresponds to a series of maneuvers executed independently from

the other vehicles. A maneuver is defined as *"a physical movement or series of moves requiring skill and care"*. Trajectory prediction with *Maneuver-based* motion models is based on the early recognition of the maneuvers that drivers intend to perform. If one can identify the maneuver intention of a driver, one can assume that the future motion of the vehicle will match that maneuver. Thanks to this a priori, trajectories derived from this scheme are more relevant and reliable in the long term than the ones derived from *Physics-based* motion models.
In practice, the assumption that vehicles move independently from each other does not hold. Vehicles share the road with other vehicles, and the maneuvers performed by one vehicle will necessarily influence the maneuvers of the other vehicles. Inter-vehicle dependencies are particularly strong at road intersections, where priority rules force vehicles to take into account the maneuvers performed by the other vehicles. Disregarding these dependencies can lead to erroneous interpretations of the situations, and affects the evaluation of the risk.

3. Interaction-aware motion models: Interaction-aware motion models represent vehicles as maneuvering entities that interact with each other, i.e. the motion of a vehicle is assumed to be influenced by the motion of the other vehicles in the scene. Taking into account the dependencies between vehicles leads to a better interpretation of their motion compared to the *Maneuver-based* motion models. As a result, it contributes to a better understanding of the situation and more reliable evaluation of the risk.
The Interaction-aware motion models are the most comprehensive models proposed so far in the literature. They allow longer-term predictions compared to *Physics-based* motion models, and are more reliable than *Maneuver-based* motion models since they account for the dependencies between vehicles. However, this exhaustiveness has some drawbacks: computing all the potential trajectories of the vehicles with these models is computationally expensive and not compatible with real-time risk assessment.

Regarding the characteristics and limitations of each model, we observe that the *"Interaction-aware"* motion model is the best suited to predict drivers' behaviors from the off-board infrastructure. The *RSU* could perceive vehicles data resulting from vehicles dynamics (position, speed, etc.) using its sensors and data related to maneuvers performed by drivers (throttle position, brake, etc.) through *V2X* communication messages. The processing of this data allows to extract the driving contextual situation such as the relative distance and the relative speed between vehicles. This implicitly explains the interaction between drivers at the highway on-ramp to estimate their intentions probability and to have a long-term prediction.

We propose to use a directed graphical model with factored states to estimate drivers' intentions. The decision of merging on highways is determined mainly by a set of contextual parameters that drivers perceive, such as the distance to the merging point and the relative speed from the vehicle inserted. Using factored states allows reducing calculation burdens. Figure 3.2 shows the model structure for the vehicle in the merge lane and the vehicle in the main highway lane (first lane as illustrated in Figure 3.3). The network is composed of three layers: vector X which contains the vehicle data (mainly dynamic data), vector C which contains the vehicle situation context, and finally, the output I which is the intention probability of merging or not merging for the vehicle's driver.

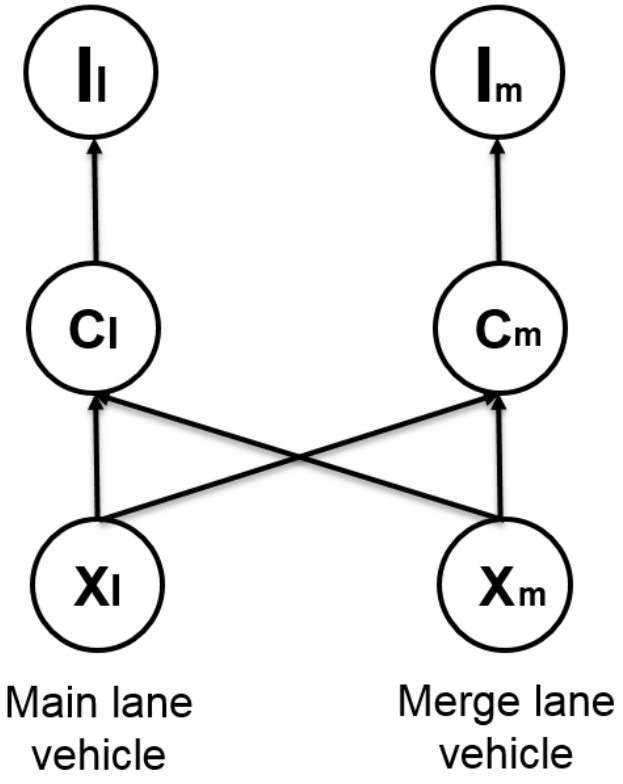


FIGURE 3.2: Directed graphical model used for drivers' intentions estimation at the highway on-ramp merging situation.

- **Vector X:** contains the vehicle states: position, speed, acceleration. This information can be retrieved in real-time by the road-side unit (*RSU*) using, either its sensors *(camera, RADAR, or LiDAR)* that exceeds the limit of embedded vehicle sensors, or (*V2I*) communication.

- **Vector C:** contains the features of the local situation context. These features are extracted from the merge lane (X_m) and the main lane (X_l) vehicles states. The function that outputs C_m (resp. C_l) vector can be expressed mathematically as a

Dirac distribution δ_{C_m} (resp. δ_{C_l}), with X_m and X_l as input arguments:

$$\delta_{C_m}(X_m, X_l); \quad \delta_{C_l}(X_m, X_l)$$

The features of this vector for the vehicle in the merge lane (resp. main lane) are summarized in table 3.1, and illustrated in Figure 3.3.

Merge lane	*Main lane*	*Feature*
C_1	C'_1	*Distance from the merging point*
C_2	C'_2	*Speed*
C_3	C'_3	*Acceleration*
C_4	C'_4	*Relative distance between the vehicle in the main lane and the vehicle in the merge lane*
C_5	C'_5	*Relative speed between the vehicle in the main lane and the vehicle in the merge lane*
C_6	C'_6	*Relative acceleration between the vehicle in the main lane and the vehicle in the merge lane*
C_7	C'_7	*Relative distance from the vehicle above the merging point in the main lane*
C_8	C'_8	*Relative speed from the vehicle above the merging point in the main lane*
C_9	C'_9	*Relative acceleration from the vehicle above the merging point in the main lane*

TABLE 3.1: Vector C features.

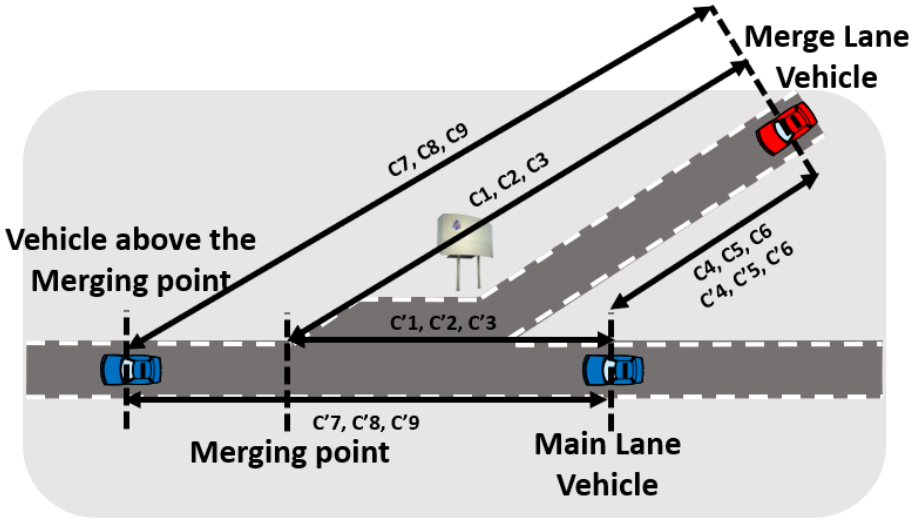


FIGURE 3.3: Contextual vector for the vehicle in the merge lane and the vehicle in the main lane.

- **Vector *I*:** contains the intention's probability of *"merging"* or *"not merging"* for the vehicle in the merge lane (I_m), and the intention's probability of *"passing"* or *"not passing"* for the vehicle in the main lane (I_l). The probability of *"merging"*

(resp. *"passing"*) is deduced from the situation context vector Pr[1](I_m/C_m) (resp. $Pr(I_l/C_l)$). An output probability of I_m (resp. I_l) with a value close to *1* means that the vehicle in the merge lane (resp. the main lane) has the intention to *"merge"* (resp. *"pass"*) before the vehicle in the main lane (resp. the merge lane) at the highway on-ramp.

3.3 Probabilistic classifiers

Since drivers behaviors and intentions are highly random, stochastic and non-deterministic, different probabilistic classifiers were used to predict this intention and identify the merged vehicle: Classic logistic regression (LRM) is used as a discriminative classifier [43][44]. The Naïve-Bayes model (NB) and two of its variants (the Tree Augmented Naïve-Bayes (TAN) and the General Bayesian Network (GBN)) are used as generative classifiers [45]. Moreover, the k-Nearest Neighbors classifier (KNN) and the Artificial Neural Network (ANN) are included in this comparison.

3.3.1 Logistic regression

Logistic regression is a discriminative regression model. A discriminative algorithm simply categorizes a given input (C vector). Discriminant classifiers directly model the posterior $Pr(I/C)$ (intention I) or learn a direct model from the input to the output. This model solves the problem $Pr(I/C)$ directly. It is based on the logistic function that is a sigmoid function, that takes any real input t, $t \in R$, and an output which takes a value between zero and one. For the *logit* function, this is interpreted as log-odds input and output probabilities:

$$t = logit(Pr) = ln(\frac{Pr}{1-Pr}) \qquad Pr \in]0;1[\tag{3.1}$$

The logistic function $\sigma(t)$ is defined as follows:

$$\sigma(t) = \frac{e^t}{1+e^t} = \frac{1}{1+e^{-t}} \tag{3.2}$$

Suppose that t is a linear function of a single explanatory variable C (situation context vector). We can then express t as follows: $t = \beta_0 + \beta_1 C$, and the logistic function can be written as:

$$Pr(I/C) = \frac{1}{1+e^{-\beta_0+\beta_1 C}} \tag{3.3}$$

[1] Pr denotes probability.

Note that *Pr(I/C)* is interpreted as the probability that the driver's intention I corresponds to a *"merge"* rather than *"Yield"*. The parameters $[\beta_0, \beta_1]$ can be determined either for maximizing the conditional likelihood on the training data set or to minimize training error [43][44].

3.3.2 Naïve-Bayes

A Naïve-Bayes (*NB*) is a classifier that learns from training data the conditional probability of each attribute C_i (resp. C'_i) given the class label I_m (resp. I_l). Classification is then done by applying Bayes rule to compute the probability of I_m (resp. I_l) given the particular instance of C_1,...,C_9 (resp. C'_1,...,C'_9), and then predicting the class with the highest posterior probability. This computation is rendered feasible by making a strong independence assumption: all the attributes C_i (resp. C'_i) are conditionally independent given the value of the class I_m (resp. I_l). By independence means probabilistic independence, that is, C_i (resp. C'_i) is independent of C_j (resp. C'_j) given I_m (resp. I_l) whenever $Pr(C_i|C_j,I_m) = Pr(C_i|I_m)$ (resp. $Pr(C'_i|C'_j,I_l) = Pr(C'_i|I_l)$) for all possible values of i and $j \in \{1{:}9\}$, whenever $Pr(I_m) > 0$ (resp. $Pr(I_l) > 0$). A Naïve-Bayes (*NB*) is a simple structure where the classification node is the parent node of all other nodes as shown in Figure 3.4. That is to say the intention for the vehicle in the merge lane (resp. main lane) is the parent for the contextual situation features. No other connections are allowed in a Naïve-Bayes structure.

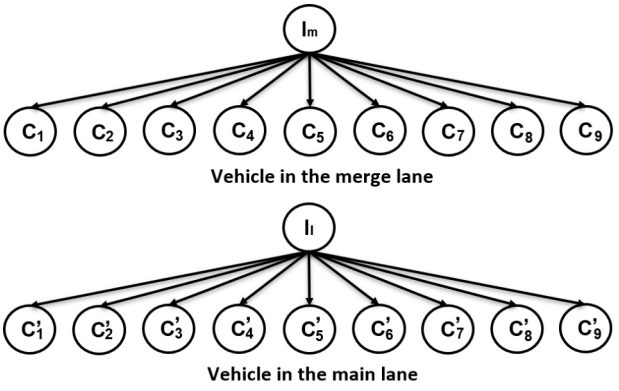


FIGURE 3.4: Naïve-Bayes structure.

3.3.3 Tree Augmented Naïve-Bayes

The Tree Augmented Naïve-Bayes (*TAN*) allows tree-like structures to be used to represent dependencies among attributes. In a tree-augmented naïve bayesian (*TAN*) network, the class variable has no parents and each attribute has as parents the class variable

and at most one other attribute. Let C_m= $\{C_1,...,C_9,I_m\}$ (resp. C_l= $\{C'_1,...,C'_9,I_l\}$) represents the node set for the vehicle in the merge lane (resp. main lane), where I_m (resp. I_l) is the classification node of the data. The algorithm for learning *TAN* classifiers [45] first learns a tree structure over $C_m \backslash I_m$ (resp. $C_l \backslash I_l$), using mutual information tests conditioned on C_m (resp. C_l). It then adds a link from the classification node to each feature node, similar to a Naïve-Bayes structure (i.e. the classification node is a parent of all other nodes)(cf. Figure 3.5).

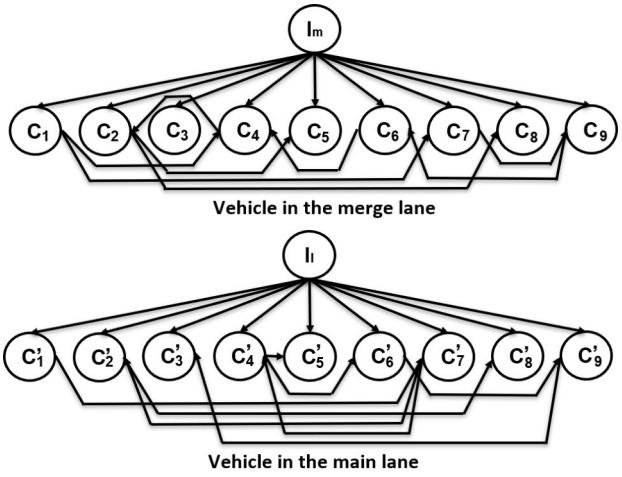


FIGURE 3.5: Tree Augmented Naïve-Bayes structure.

3.3.4 General Bayesian Network

The General Bayesian Network (*GBN*) is an unrestricted Bayesian network. In general Bayesian network, all nodes treat as a normal node and can have a present node and also be a child node of some attribute node. *K2* and *Hill Climb* (*HC*) algorithms are adopted to generate the *GBN*. *K2* algorithm is a simple and fast algorithm, a kind of greedy algorithm. It starts with a given ordering of the nodes. *Hill Climb* algorithm starts from an empty or random network. If there is no information on the conditional probability distribution of the data in the network structure learning is required. An important element in the network structure learning is actual sample data for each event. *GBN* can have reasoning ability throughout the network structure learning based on the actual sample data. A typical method for structural learning is the *score-based* learning, which is used to learn the general Bayesian network for our use case (Figure 3.6). This method is to maximize the score according to the degree of matching of generated network and actual data [46].

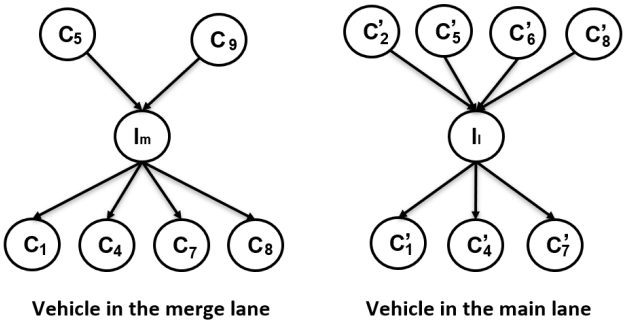


FIGURE 3.6: General Bayesian Network structure.

3.3.5 K-Nearest Neighbor

The K-Nearest Neighbor (*KNN*) is one of the classification techniques using a machine learning algorithm. The *KNN* is known as a simple but robust classifier and produced high-performance results even for complex applications. The *KNN* uses a distance of features in a dataset to determine the data belongs to which group. The close distance between features means the features in the same group while the long distance between features means that the features in the different groups. Therefore *KNN* is a non-parametric procedure to determinate the appropriate group which close in *Euclidian* distance [47]. For our use case, that means that all the data sets should be stored at the road-side unit (*RSU*) to use such classifier for drivers' intentions estimation, which is more complex and expensive in terms of hardware requirement.

3.3.6 Artificial Neural Networks

In recent times, artificial neural networks (*ANN*) have become a popular and helpful model for classification, clustering, pattern recognition, and prediction in many disciplines. *ANN* are one type of model for machine learning (*ML*) and have become relatively competitive to conventional regression and statistical models regarding usefulness. Nowadays, *ANN* are mostly used for universal function approximation in numerical paradigms because of their excellent properties of self-learning, adaptivity, fault tolerance, non-linearity, and advancement in input to an output mapping. *ANN* can learn by example like people. In some cases, *ANN* can be designed for a specific application like data classification or pattern recognition through the learning process, such as predicting drivers' intentions for our use case. An architecture of a typical *NN* is showed in Figure 3.7. Neural network (*NN*) layers are independent of one another; that is, a specific layer can have an arbitrary number of nodes. This arbitrary number of nodes is called a *bias node*. The bias nodes are always set as equal to one. In analogy, the bias nodes are like the offset in linear regression given as; $y = a.x + b$, where "a" is

the coefficient of independent *"x"* and then *"b"* is called slope. A bias major function is to provide a node with a constant value that is trainable, in addition to the normal inputs received by the network node. Importantly, a bias value enables one to move the activation function either to the right or the left, which can be analytical for *ANN* training success. When the *NN* is used as a classifier, the input and the output nodes will match input features and output classes. However, when the *NN* is used as a function approximation, it generally has an input and an output node. However, the number of designed hidden nodes essential greater than those of input nodes [48]. For our highway on-ramp drivers' intentions prediction, a neural network with an input layer of *9* nodes, a single hidden layer of *24* nodes, and an output layer of *12* nodes are used.

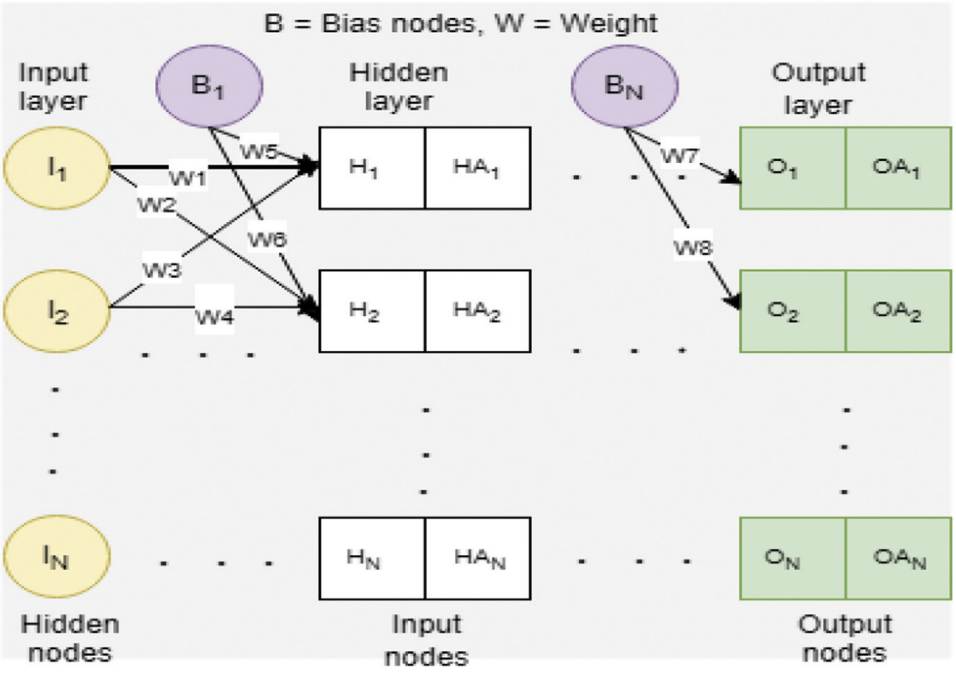


FIGURE 3.7: A typical neural network architecture.

3.4 Experimental Evaluation

3.4.1 Training Dataset

The proposed model was trained and validated using the Next Generation Simulation *(NGSIM)* [19], which provides vehicle trajectories and supporting data. The data used to train our model correspond to vehicles' trajectories on a segment of interstate *80* in *Emeryville (San Francisco), California* collected between *4:00 p.m.* and *4:15 p.m.* on *April 13, 2005*. Data represent travel on the northbound direction of *Interstate 80* in *Emeryville*, *California*. This data was collected using video cameras mounted on a *30-story* building, *Pacific Park Plaza*, which is located in *6363 Christie Avenue* and is

adjacent to the interstate freeway *I-80*.

Figure 3.8 provides a schematic illustration of the location for the vehicle trajectory dataset. The site was approximately *1650 feet* in length, with an on-ramp at *Powell Street*.

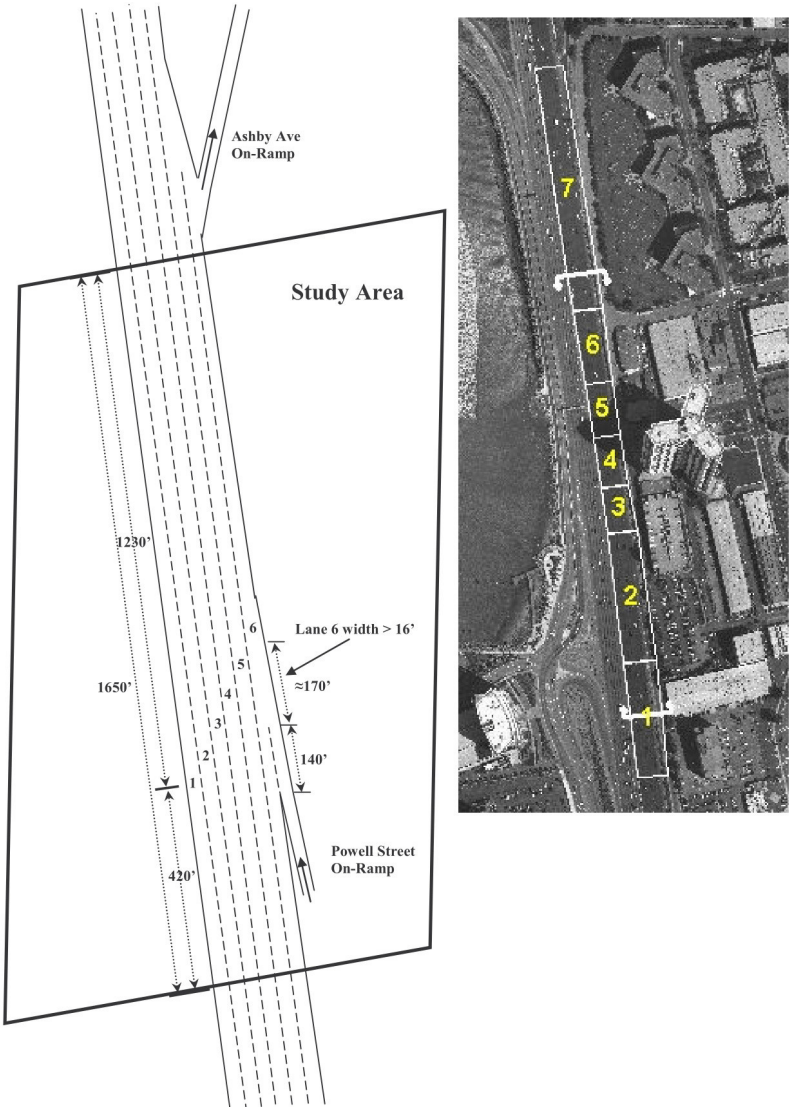


FIGURE 3.8: Study area schematic and camera coverage [19].

Data were collected through a network of synchronized digital video cameras *NGVIDEO*, a customized software application developed for the *NGSIM* program, transcribed the vehicle trajectory data from the video. These vehicle trajectory data provided the precise location of each vehicle within the study area every one-tenth of a second, resulting in detailed lane positions and locations relative to other vehicles [19]. Notice that vehicles data are obtained in the same way as the road-side unit (*RSU*) perceives and provides vehicles information, which allows us to use such dataset for our use case. There are *191* merging-host-leading groups in the *I-80* dataset between *4:00 p.m.* and *4:15 p.m.*.

We use these data because they correspond to non-congested traffic conditions with an average speed of *58 km/h*, which is the study's objective of our thesis. Each group contains a host vehicle, a leading vehicle and multiple merging vehicles. The data were preprocessed to detect, at each time interval, which vehicle *ID*[2] reaches the merging point. Figure 3.9 shows that, at each instant, the vehicle *ID* reaching the merging point corresponds to either a vehicle *ID* in the merge lane or a vehicle *ID* in the main lane. This determines the output probability for each vehicle driver's model. Following data preprocessing, the features for the vehicle in the merge lane were extracted, which compose the situation context vector C_m. Also, the features were extracted for the vehicle in the main lane, which compose the situation context vector C_l.

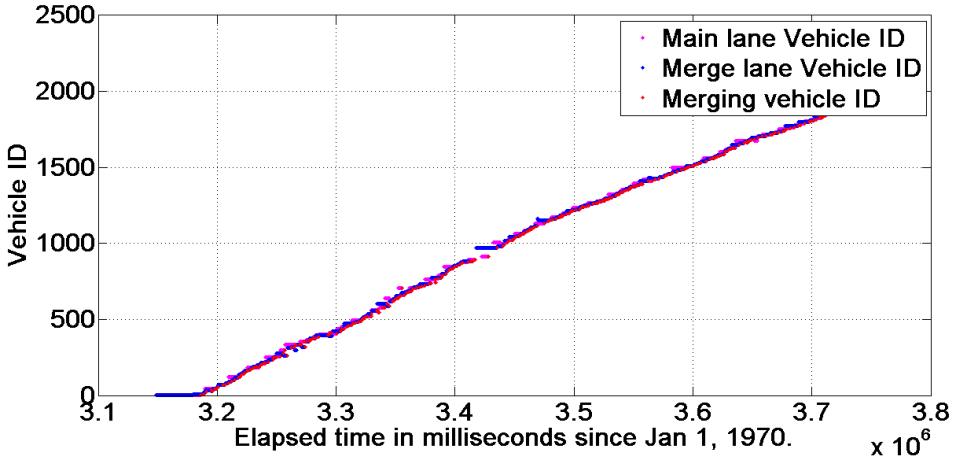


FIGURE 3.9: Data preprocessing to identify the merging vehicle *ID*.

3.4.2 Results

Each classifier is used to evaluate the probability for both vehicles in the main lane and the merge lane at each instant. The merging vehicle *ID* is then predicted by choosing the highest probability. Each model was evaluated five times for each different set of test data in a form of *5-fold* cross-validation [15]. To evaluate the performance of each classifier, we calculate the *Accuracy*, *Precision*, *Recall* and *F1* score metrics. These parameters are defined as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{3.4}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.5}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.6}$$

[2]Vehicle's identifier

$$F1\,Score = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{3.7}$$

Where: *TP*, *TN*, *FP* and *FN* are the true positives, true negatives, false positives and false negatives respectively.

The results of each classifier are summarized in Table 3.2.

Classifier	*Accuracy*	*Precision*	*Recall*	*F1 Score*
LRM	99.59 %	99.67 %	99.37 %	99.52 %
NB	92.54 %	91.95 %	90.85 %	91.27 %
TAN	98.37 %	98.28 %	97.82 %	98.05 %
GBN	97.11 %	96.09 %	96.94 %	96.47 %
KNN	96.52 %	97.93 %	94.24 %	95.92 %
ANN	99.17 %	99.63 %	98.40 %	99.01 %

TABLE 3.2: Models' performances.

From Table 3.2, the discriminative classifiers *LRM* and the Artificial Neural Network (*ANN*) give the best performances over the generative classifiers (*NB*, *TAN* and *GBN*) and *KNN* classifier. In fact, the accuracy and precision of *LRM* and *ANN* classifiers are over *99%*, which is a good ratio regarding the state-of-the-art. Naïve Bayes, which is the simplest generative classifier, yields slightly lower performances (around *92%*). Unlike *LRM* classifier, the Artificial Neural Network may have different structures. The used neural network contains one single hidden layer of *24* nodes and an output layer of *12* nodes. The performances of this classifier can be improved using a more complex structure such as more hidden layers and more nodes. The accuracy and the precision of the *ANN* were around *96%* when we previously used only *12* nodes in the hidden layer and *6* nodes in the output layer. Thereby, It has the advantage of being a more flexible classifier compared to the *LRM* classifier, which has a fixed structure and limited performance. However, *ANN* has the drawback of having more complex implementation regarding required computing power and memory storage. Moreover, *ANN* explainability must be verified. Some existing deployed systems and regulations make the need for explanatory systems urgent and timely. With impending regulations like the European Union's *"Right to Explanation"* [49], there has been a recent explosion of interest in interpreting the representations and decisions of black-box models. Concerning the *Recall* criteria, both the *LRM* and *ANN* classifiers show good values. This means that in all the situations where the vehicle in the merge lane takes priority and merges before the main lane vehicle, we can predict most of these situations accurately to use such information for decision-making strategy. Finally, the *F1* score takes both *FALSE* positives and *FALSE* negatives into account. The negative prediction refers to the intention to *"pass"* for the vehicle in the main lane (model outputs *0*), while the positive prediction refers to the vehicle intention to *"not pass"* at the main lane (model outputs *1*). Good value for the *F1* score means that the model can, also, predict the intention accurately

for the vehicle in the main lane. The accuracy of *"pass"* events is more safety critical than *"not pass"* events. Misclassifying a *"pass"* event as a *"not pass"* event may motivates the merge lane vehicle to take priority, and could result in a traffic crash, whereas misclassifying a *"not pass"* event as a *"pass"* event would only result in a lost opportunity to merge. Table 3.3 contains a comparison between the accuracy of our proposed model trained by *ANN* and *LRM* classifiers and the previous state-of-the-art approaches. The accuracy of our proposed off-board model outperforms all previously proposed solutions.

	LRM	***ANN***	*Ref. [3]*	*Ref. [6]*	*Ref. [9]*
Accuracy	**99.59 %**	**99.17 %**	87%	96.7%	97.7 %

TABLE 3.3: Models' accuracy.

To ensure that this model can be used in practice by the off-board road-side unit, the true intention prediction must be provided earlier than the arrival of the merge lane vehicle at the merging point. Therefore, we calculate the time to arrive at the merging point for the vehicle in the merge lane (T_m), and the time to arrive at the merging point for the vehicle in the main lane (T_l) at the instant when the model outputs the first *TRUE* intention prediction of the driver (cf. Figure 3.10). To use the intention's prediction for decision making, we estimate that it must be provided to the *CAV* before a minimum time of T_R=*0.5 sec*, based on the following time requirements:

- *0.4 sec*, for the vehicle's dynamic response time [39].
- *0.1 sec*, for the communication latency [40].

The values of the time to arrive at the merging point when the model outputs the first *TRUE* prediction are shown in Figure 3.10 for the *ANN* classifier.

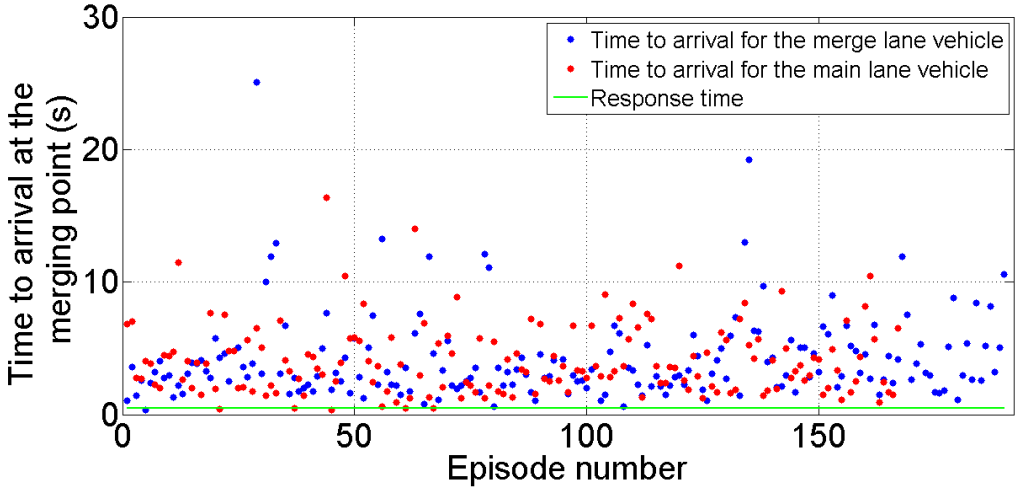


FIGURE 3.10: Time to arrival at the merging point when the *ANN* model outputs the first *TRUE* intention's prediction.

For each one of the previously classifiers, we notice that the model predicts the intention of the vehicles before they reach the on-ramp merge point by a mean of *3.96 sec* for the vehicle in the merge lane, and *4.08 sec* for the vehicle in the main lane, which we estimate to be sufficient for decision making. Moreover, the model predicts the intention below the vehicle response time (T_R) only in three prediction episodes: one corresponds to a vehicle following another vehicle that changes the main lane close to the merging point, while the two other cases correspond to a vehicle that follows immediately the merging vehicle in the merge lane. These cases may be solved by extending the proposed model to predict the intention of the follower vehicle in the main lane and the merge lane.

In short, the performance metrics show the best values for the *LRM* and *ANN* classifiers that offer an accuracy and precision of around *99%* for predicting drivers' intentions at the highway on-ramp merging situation. The model is robust for predicting both the main lane and the merge lane vehicles' intentions. Also, we demonstrated that the proposed method predicts the intention before a time interval sufficient for decision making.

3.4.2.1 Driver intention model extension

To get a higher prediction time horizon, we extend the driver intention model to the follower vehicle at the main lane (see Figure 3.11).

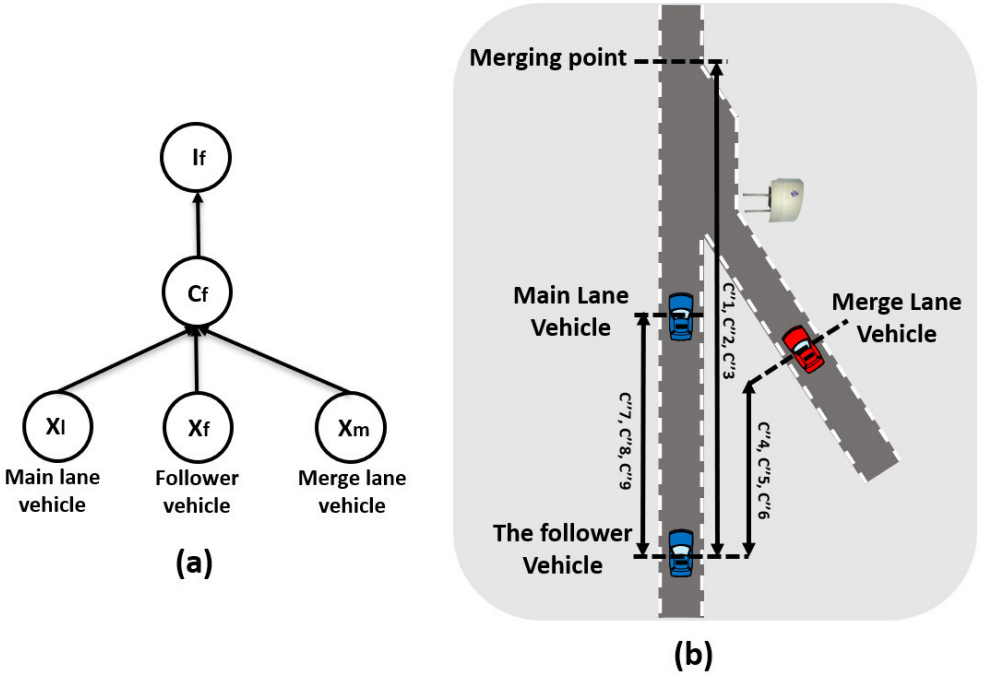


FIGURE 3.11: (a) The follower vehicle's intention model (b) Context situation vector.

The model for the follower vehicle is composed of:

- ***Vector*** X_f***:*** contains the vehicle states: *position, speed, acceleration.*
- ***Vector*** C_f***:*** contains the features of the local situation context, which are summarized in Table 3.4.

Follower model	*Feature*
C_1''	*Distance from the merging point*
C_2''	*Speed*
C_3''	*Acceleration*
C_4''	*Relative distance from the vehicle in the merge lane*
C_5''	*Relative speed from the vehicle in the merge lane*
C_6''	*Relative acceleration from the vehicle in the merge lane*
C_7''	*Relative distance from the vehicle in the main lane*
C_8''	*Relative speed from the vehicle in the main lane*
C_9''	*Relative acceleration from the vehicle in the main lane*

TABLE 3.4: Vector C features for the follower vehicle.

- ***Vector*** I_f***:*** contains the intention of the follower vehicle. An output probability with a value close to *1* means that the follower vehicle has the intention to *"pass"* before the vehicle in the merge lane at the highway on-ramp.

As for the main lane vehicle and the merge lane vehicle, we calculate the output for the follower vehicle's model using previous classifiers. The results are summarized in the Table 3.5.

Classifier	*Accuracy*	*Precision*	*Recall*	*F1 Score*
LRM	97.38 %	96.86 %	91.12 %	93.59 %
NB	92.20 %	86.87 %	78.02 %	81.27 %
TAN	90.42 %	92.61 %	61.87 %	68.70 %
GBN	93.28 %	97.21 %	70.39 %	80.24 %
KNN	97.66 %	95.01 %	94.40 %	94.39 %
ANN	97.27 %	93.61 %	94.23 %	93.26 %
Combined	97.66 %	96.01 %	94.10 %	94.21 %

TABLE 3.5: Performance of the follower vehicle's model.

From Table 3.5, the performances of the follower vehicle model are lower than the previous model. The best accuracy is obtained when using *LRM*, *KNN*, and *ANN* classifiers, and it has the value *97%* that is, *2%*, lower than the accuracy for the vehicle in

the main lane below the merging point. To improve prediction performance, we proposed combining the best three classifiers (*LRM*, *KNN*, and *ANN*), where each classifier vote according to its output. The method, then, selects the prediction with the majority vote. From the table, we notice that the combination method average the performance metrics that were fluctuated when using each individual classifier. When combining the best classifiers, better performance values are obtained, that may be exploited later for decision making.

Extending the driver's intention model to the follower vehicle at the main highway lane aims to increase the prediction time horizon before that the merge lane vehicle reaches the on-ramp merging point. To check that, figure 3.12 shows the values of the time to arrival at the merging point for the first *True* prediction of the follower vehicle model. We notice that the model predicts the intention above the vehicle response time (T_R). The model predicts the intention of the follower vehicle before it reaches the on-ramp merging point by a mean of *8.38 sec*.

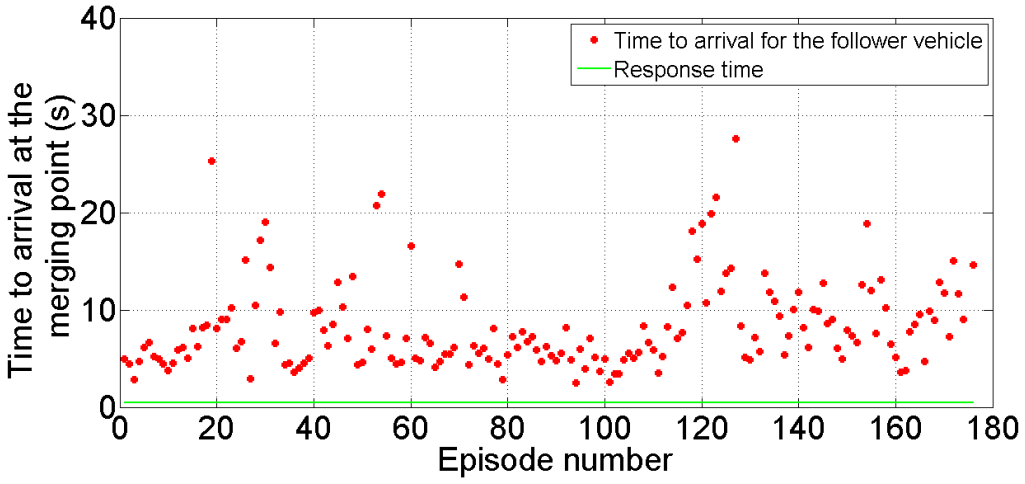


FIGURE 3.12: Time to arrival for the first *True* prediction of the follower vehicle's model.

3.4.2.2 On-board driver intention model

To compare the performances of the proposed off-board model with its on-board counterpart, we trained and tested an on-board model that predicts drivers' intentions at the highway on-ramp situation (cf. Figure 3.13). This model uses only embedded data of the subject vehicle. The data about the vehicle in the other lane (either the main lane or the merge lane) is not available in the case of a non-connected car, and cannot be perceived directly by autonomous vehicle due to the limited perception range of their sensors at the highway on-ramp. In contrast, when using an off-board solution, data about vehicles are available either by communication (*V2I*) or by road-side unit (*RSU*)

of second generation.

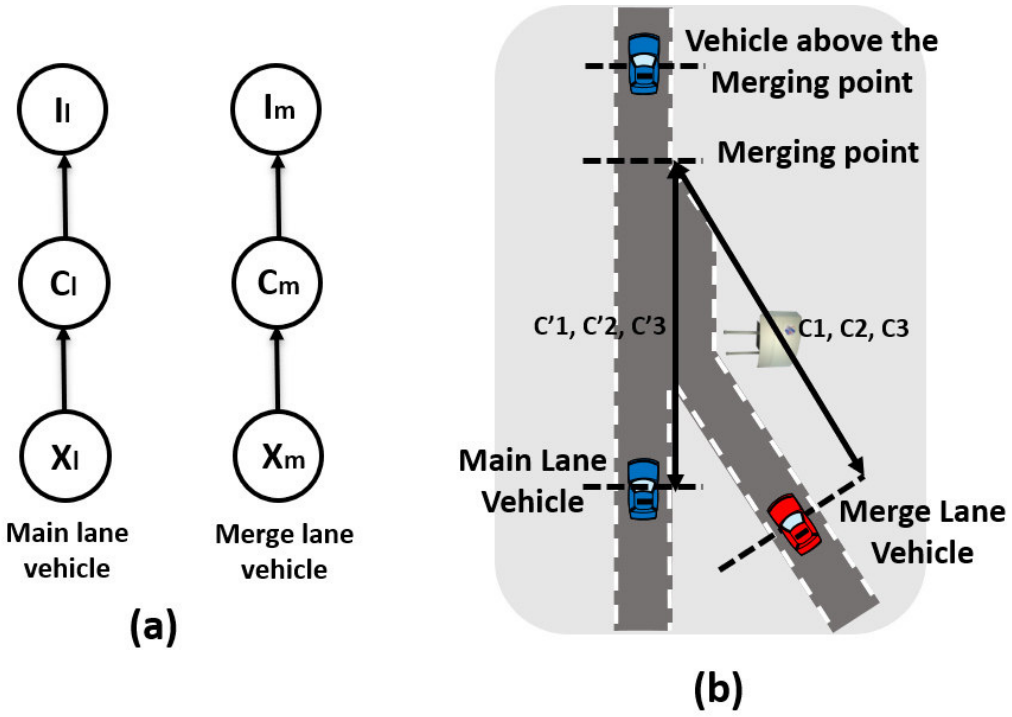


FIGURE 3.13: On-board highway on-ramp driver intention model.

The on-board model of each vehicle is composed of:

- ***Vector X:*** contains the vehicle states: *position*, *speed*, *acceleration*.
- ***Vector C:*** contains the features of the local situation context, which are summarized in Table 3.6. These are the only on-board parameters available for describing drivers' behaviors.

Merge lane	*Main lane*	*Feature*
C_1	C'_1	*Distance from the merging point*
C_2	C'_2	*Speed*
C_3	C'_3	*Acceleration*

TABLE 3.6: Vector C features for on-board model.

- ***Vector I:*** contains the intention of the vehicle. It outputs the probability of *"merging"* or *"not merging"* for the merge lane vehicle, and the probability of *"passing"* or *"not passing"* for the main lane vehicle. The probability of each vehicle is totally uncorrelated from the probability of the other vehicle. The prediction output of each vehicle's intention is determined by a probability threshold of *50 %*.

Each model was trained using the same *NGSIM* dataset with the different classifiers stated earlier. The performances are summarized in Tables (3.7) and (3.8).

Classifier	Accuracy	Precision	Recall	F1 Score
LRM	83.08 %	83.28 %	88.44 %	85.73 %
NB	74.99 %	72.93 %	92.16 %	80.73 %
TAN	80.20 %	77.18 %	93.27 %	84.36 %
GBN	80.29 %	77.47 %	92.91 %	84.39 %
KNN	78.05 %	80.17 %	81.86 %	80.92 %
ANN	73.01 %	70.10 %	97.80 %	81.20 %

TABLE 3.7: On-board main lane model's performances.

Classifier	Accuracy	Precision	Recall	F1 Score
LRM	82.33 %	79.15 %	81.63 %	79.65 %
NB	67.66 %	60.74 %	85.82 %	69.80 %
TAN	79.43 %	76.51 %	78.71 %	76.43 %
GBN	79.14 %	76.13 %	78.27 %	76.06 %
KNN	80.63 %	77.99 %	77.50 %	77.29 %
ANN	74.21 %	93.24 %	72.57 %	79.69 %

TABLE 3.8: On-board merge lane model's performances.

The accuracy of the on-board model is lower than the accuracy of the off-board model by more than *15%*. The other performance metrics are, also, far lower than the previous model. This is due to the lack of information about the vehicle in the other lane, and the dependency of the on-board model to the driving style, which varies from one driver to another. Furthermore, accuracy changes considerably from one probabilistic classifier to another, in contrast to the off-board model where accuracy was approximately constant for all the classifiers.

This comparison allows us to confirm that the off-board unit that uses communication and sensors at the highway on-ramp can predict drivers' behaviors more accurately than the classic embedded implementation, independently from the used probabilistic classifiers or the drivers' driving styles.

3.5 Conclusion

In summary, off-board drivers' intentions model provides prediction accuracy greater than *99%* using either Logistic Regression Model (*LRM*) or Artificial Neural Network (*ANN*). This model can be implemented on the road-side unit (*RSU*), and can provide prediction information in real-time for decision making.

In contrast to the embedded model, the off-board model can predict drivers' behaviors more accurately, independently from the used probabilistic classifiers or the drivers' driving styles, as was shown in section 3.4.2.2. Implementing such a model at the road-side unit allows learning the intentions and behaviors of drivers at the highway on-ramp using a data-driven approach rather than a deterministic analytic form or an embedded model, that may depend on each driver's style.

In the next chapter, we show how the prediction provided by this off-board model could be used to improve the decision making of the autonomous vehicle at the highway on-ramp merging scenario.

Chapter 4

Decision making

4.1 Introduction

Classical rule-based approaches for decision making that include heuristics, optimal-control, and model predictive control require accurate modeling of the environment, where calculations are a burden. Besides, they cannot handle uncertainties and unforeseen situations, which is not practical in the real world. Reinforcement learning (*RL*) methods are more adapted for real-world situations such as highway on-ramp merging, where there are uncertainties relatives to drivers' behaviors and vehicles' dynamic models. In this chapter, we present a novel architecture that combines the previous driver intention model with a deep reinforcement learning framework that performs autonomous highway on-ramp merging *"safely"* and *"cooperatively"*.
We first review the theoretical framework of deep reinforcement learning. We then formulate the high-speed highway on-ramp merging as a *RL* problem. We will combine the deep *RL* agent with a safe controller and the driver intention model. Our solution is tested and validated using the traffic simulator *"SUMO"*. We show that our new architecture accelerates learning of the deep *RL* agent, improves safety performance and learns a cooperative autonomous driving policy.

4.2 Preliminaries on reinforcement learning

The *RL* problem for maximizing the long-term reward while operating in an environment can be represented as a Markov Decision Process (*MDP*). Formally, an *MDP* is represented by the tuple <*S*, *A*, *T*, *R*>, where *S* is the set of states, *A* is the set of actions, and $T(s_t, a_t, a_{t+1})$ represents the stochasticity in the underlying environment and provides the probability of transitioning from state s_t to state s_{t+1} on taking action a_t (cf. Figure 4.1).

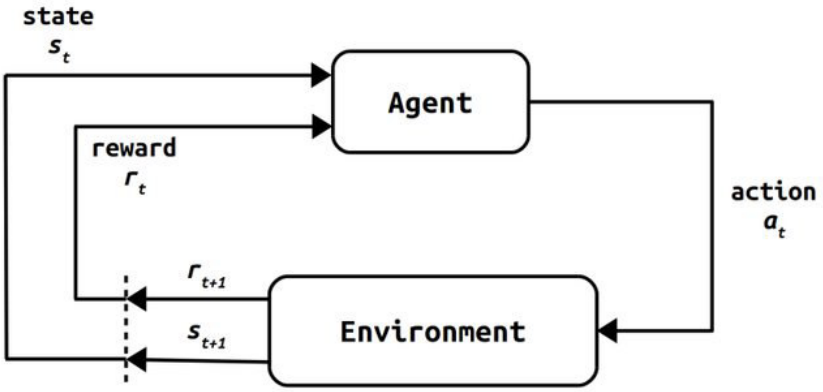


FIGURE 4.1: The Reinforcement Learning Framework [20].

$R(s_t, a_t)$ represents the reward obtained on taking action a_t in state s_t. The *RL* problem is to learn a policy that maximizes the long-term reward from experiences without knowing the exact model of transitions and rewards. An experience is defined as a tuple (s_t, a_t, s_{t+1}, r_t) and, typically, learning occurs over a batch of experiences (referred to as an episode) that ends when s_{t+1} is a terminal state. Q-learning represents the value function for being in state s_t and taking action a_t:

$$Q(s_t, a_t) = E_{s_{t+1}, r_t}\ [\ r_t + \gamma . max_{a_{t+1}}\ Q(s_{t+1}, a_{t+1})] \tag{4.1}$$

where the expectation, *E* is over the stochasticity in the environment concerning transitions and also reward.

Recently, a novel deep reinforcement learning algorithm, called the deep deterministic policy gradient (*DDPG*) [50], has achieved good performances in many simulated continuous control problems. In *DDPG*, we have a critic function *Q* parameterized by θ^Q that approximates the state-action-value function. We also have an actor μ parameterized by θ^μ that outputs the deterministic action in a continuous space given the current state. Let *N* denote the size of the batch of total experiences $e_i = (s_i, a_i, s_{i+1}, r_i)$, $i =$

1...N collected in an episode. The critic is updated by minimizing the loss:

$$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - Q(s_i, a_i|\theta^Q))^2 \tag{4.2}$$

where:

$$y_i = r_i + \gamma.Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'}) \tag{4.3}$$

where Q' and μ' are target networks parameterized by $\theta^{Q\prime}$ and $\theta^{\mu\prime}$ respectively. The parameters of these target networks are made to slowly track the parameters of the original networks: $\theta' \leftarrow \tau\theta + (1 - \tau)\,\theta'$, with $\tau \ll 1$. The purpose of this is to avoid making targets y_i non-stationary, and improve the stability of updates. Next, the actor policy μ is updated by using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_{i} \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i} \tag{4.4}$$

When learning from low dimensional feature vector observations, the different components of the observation may have different physical units (for example, positions versus velocities) and the ranges may vary across environments. The *DDPG* addresses this issue by adopting a recent technique from deep learning called batch normalization [51]. This technique normalizes each dimension across the samples in a mini-batch to obtain zero-mean and unit variance. Besides, a major challenge of learning in continuous action spaces is exploration. An advantage of off-policy algorithms such as *DDPG* is that we can treat the problem of exploration independently from the learning algorithm. In [52], an exploration policy μ was constructed by adding noise sampled from a noise process $\mathcal{N}$ to the actor policy:

$$\mu(s_t) = \mu(s_t|\theta_t^\mu) + \mathcal{N} \tag{4.5}$$

In the present work, $\mathcal{N}$ will be chosen as an *"Ornstein-Uhlenbeck"* process to generate temporally correlated exploration noise for exploration's efficiency in physical control problems with inertia, such as vehicle dynamic control. This allows to better explore the effects of control actions on the vehicle's behavior.

The *DDPG* algorithm is summarized below: first initialize the actor and critic networks, their targets and the replay buffer (lines *1* to *3*). At each time step of each episode, select and execute action according to equation 4.5 (lines *8* to *9*). Afterward, observe and store the environment transition (line *10*). Then, update the critic network according to equations 4.2 and 4.3 (lines *11* to *13*) and update the actor network according to equation 4.4 (line *14*). Finally, update the target networks smoothly (line *15*).

Algorithm 1 *DDPG algorithm*

1: Randomly initialize critic network Q(s,a|θ^Q) and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
2: Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$.
3: Initialize replay buffer R.
4: **for** episode = 1, M **do**
5: Initialize a random process $\mathcal{N}$ for action exploration.
6: Receive initial observation state s_1.
7: **for** t = 1, T **do**
8: Select action $a_t = \mu(s_t|\theta^\mu_t) + \mathcal{N}$ according to the current policy and exploration noise.
9: Execute action a_t and observe reward r_t and observe new state s_{t+1}.
10: Store transition (s_t,a_t,r_t,s_{t+1}) in R.
11: Sample a random mini-batch of N transitions (s_i,a_i,r_i,s_{i+1}) from R.
12: Set $y_i = r_i + \gamma.Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$.
13: Update critic θ^Q by minimizing the loss: $L = \frac{1}{N}\sum_{i=1}^{N}(y_i - Q(s_i, a_i|\theta^Q))^2$
14: Update the actor θ^μ policy using the sampled policy gradient:
$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$
15: Update the target networks:
$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
16: **end for**
17: **end for**

4.2.1 Twin Delayed DDPG (TD3)

While *DDPG* can achieve great performances sometimes, it can suffer from the function approximation errors which lead to overestimated values and suboptimal policies. Thus, the twin delayed deep deterministic policy gradient (*TD3*) algorithm [53] was proposed to address this problem. It borrows the idea of double Q-learning to build an additional Q network, and take the minimum value between the pair of Q networks when setting the target Q value. It also suggests delaying the target network update steps. Twin Delayed DDPG (*TD3*) is an algorithm that addresses the following *DDPG* issues:

- **Overestimation Bias:** The *TD3* uses Clipped Double-Q Learning, where it learns two Q-functions $Q_1^{'}$ and $Q_2^{'}$ and takes the minimum between the two estimates, to give the target update of our Clipped Double Q-learning algorithm:

$$y_i = r_i + \gamma.min_{j=1,2}Q_j^{'}(s_{i+1}, \mu^{'}(s_{i+1})) \tag{4.6}$$

  With Clipped Double Q-learning, the value target cannot introduce any additional overestimation over using the standard Q-learning target.

- **Variance:** Besides the impact of variance on overestimation bias, high variance estimates provide a noisy gradient for the policy update. This is known to reduce

learning speed as well as hurt performance in practice. Since target networks can be used to reduce the error over multiple updates, and policy updates on high-error states cause divergent behavior, then the policy network should be updated at a lower frequency than the value network, to first minimize error before introducing a policy update. *TD3* algorithm proposes delaying policy updates until the value error is as small as possible. The modification is to only update the policy and target networks after a fixed number of updates (d) to the critic. By sufficiently delaying the policy updates, *TD3* limits the likelihood of repeating updates concerning an unchanged critic. The less frequent policy updates that do occur will use a value estimate with lower variance, and in principle, should result in higher quality policy updates.

- **Bootstrap off similar state-action pairs:** When updating the critic, a learning target using a deterministic policy is highly susceptible to inaccuracies induced by function approximation error, increasing the variance of the target. This induced variance can be reduced through regularization. *TD3* enforces the notion that similar actions should have similar value. While the function approximation does this implicitly, the relationship between similar actions can be forced explicitly by adding a small amount of random noise to the target policy and averaging over mini-batches. This makes the modified target update:

$$y_i = r_i + \gamma.min_{j=1,2}Q_j^{'}(s_{i+1}, \mu^{'}(s_{i+1}) + \epsilon) \tag{4.7}$$

$$\epsilon \sim clip(N(0, \sigma), -c, +c) \tag{4.8}$$

where the added noise is clipped to keep the target close to the original action.

The *TD3* algorithm is summarized below: As for the *DDPG* algorithm, first initialize the actor, the two critic networks, their targets and the replay buffer (lines *1* to *3*). At each time step of each episode, select and execute action according to equation 4.5 (lines *8* to *9*). Afterward, observe and store the environment transition (line *10*). After sampling a random mini-batch (line *11*), add a small amount of random noise to the target policy (line *12*) according to equations 4.7 and 4.8, in order to bootstrap of similar state-action pairs. Then, calculate the target (line *13*) by taking the minimum of the two Q-function according to equation 4.6 for reducing overestimation bias. Update the critic network according to equation 4.2 (lines *14*). At each d steps (for reducing variance), update the actor network according to equation 4.4 (line *16*). Finally, update the target networks smoothly (line *17*).
Yet, no previously published work has used the *TD3* algorithm for autonomous highway on-ramp merging.

Algorithm 2 *TD3 algorithm*

1: Randomly initialize the two critic networks Q_1(s,a|θ_1^Q), Q_2(s,a|θ_2^Q) and actor $\mu(s|\theta^\mu)$ with weights θ_1^Q, θ_2^Q and θ^μ.
2: Initialize target network Q_1', Q_2' and μ' with weights $\theta_1^{Q'} \leftarrow \theta_1^Q$, $\theta_2^{Q'} \leftarrow \theta_2^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$.
3: Initialize replay buffer R.
4: **for** episode = 1, M **do**
5: Initialize a random process $\mathcal{N}$ for action exploration.
6: Receive initial observation state s_1.
7: **for** t = 1, T **do**
8: Select action $a_t = \mu(s_t|\theta_t^\mu) + \mathcal{N}$ according to the current policy and exploration noise.
9: Execute action a_t and observe reward r_t and observe new state s_{t+1}.
10: Store transition (s_t,a_t,r_t,s_{t+1}) in R.
11: Sample a random mini-batch of N transitions (s_i,a_i,r_i,s_{i+1}) from R.
12: $\tilde{a} \leftarrow \mu'(s_{i+1}) + \epsilon, \quad \epsilon \sim clip(N(0,\sigma), -c, +c)$
13: Set $y_i = r_i + \gamma.minQ'_{j=1,2}(s_{i+1}, \tilde{a}|\theta_j^{Q'})$.
14: Update critic $\theta_{j=1,2}^Q$ by minimizing the loss: $L = \frac{1}{N}\sum_{i=1}^{N}(y_i - Q_j(s_i, a_i|\theta_j^Q))^2$
15: **if** t **mod** d **then**
16: Update the actor policy θ^μ using the sampled policy gradient:
$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q_1(s, a|\theta_1^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$
17: Update the target networks:
$\theta_1^{Q'} \leftarrow \tau\theta_1^Q + (1-\tau)\theta_1^{Q'}$
$\theta_2^{Q'} \leftarrow \tau\theta_2^Q + (1-\tau)\theta_2^{Q'}$
$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
18: **end if**
19: **end for**
20: **end for**

4.3 Highway on-ramp merging modeling

In our highway on-ramp merging problem, the merging vehicle should find an appropriate gap according to its time to arrive at the merging point and then execute proper actions to merge safely and smoothly. There is uncertainty in other drivers' behaviors and potentially in their interactions with the merging vehicle. Also, vehicles driven in the main highway lane are of various types (car, bus, truck...etc.), and have different dynamic models depending on their manufacturer. Since control actions at one time-step have an impact on the subsequent steps to achieve safe on-ramp merging, and there is transitional uncertainty between each action, *RL* is an ideal model for the problem of highway on-ramp merging.

The merging vehicle (vehicle of interest) is connected and automated. As was assumed in [17], the decision for the highway on-ramp merging is determined only by the projection of the merging vehicle in the main lane *(V_m)* and only the two preceding vehicles *(P_1, P_2)* and the two following vehicles *(F_1, F_2)* in the main lane, as illustrated in Figure 4.2. Also, we assume that the merging vehicle is fully informed about the states

of the surrounding vehicles using its sensors, *V2V* communication, and the road-side unit *(RSU)*[1]. This makes our environment state fully observable even in the presence of non-automated and non-connected vehicles in the main highway lane.

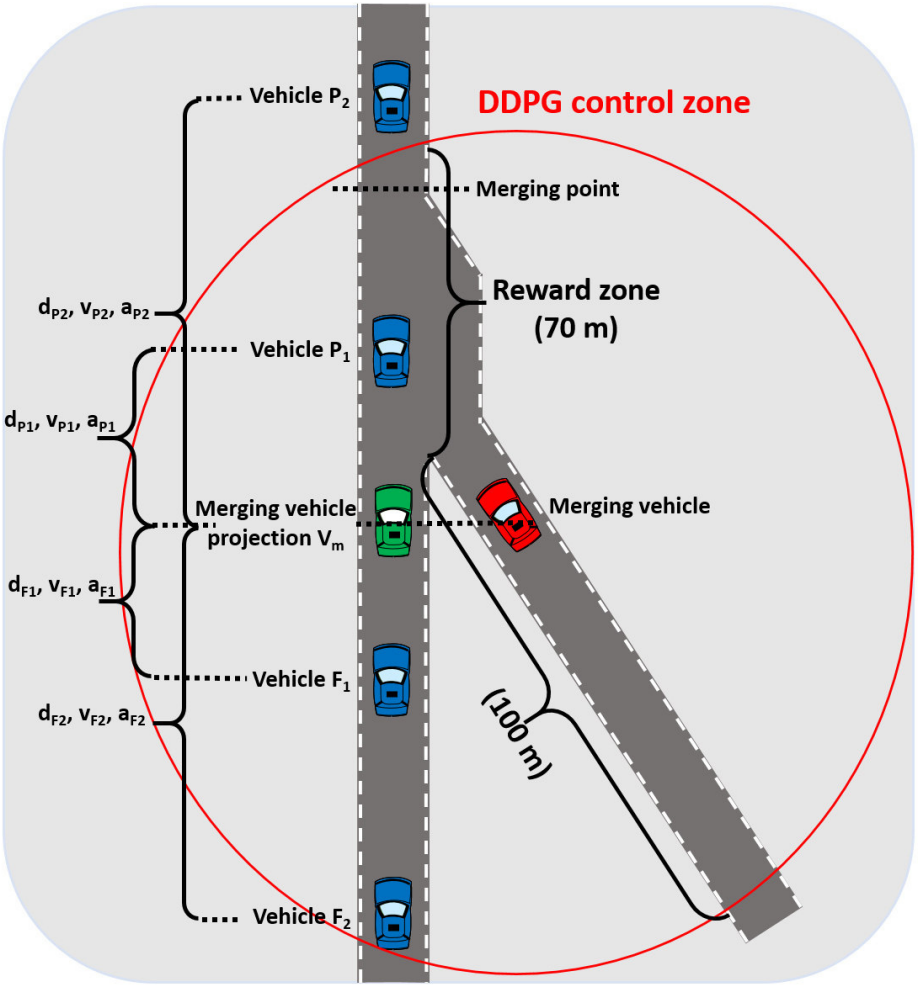


FIGURE 4.2: Highway on-ramp merging modeling.

We consider that the *RL* agent learns a policy that controls the merging vehicle in a zone starting from *100 m* distance before the beginning of the acceleration lane, which is the required distance to perform a complete stop from maximum speed (*29 m/s*)[2] and at maximum deceleration (*-5 m/s^2*)[3]. The control of the merging vehicle ends (*20 m*) after the end of the acceleration lane (the merging point in Fig. 4.2), which has a total length of (*50 m*)[4]. That means that the policy will be trained to find an appropriate gap, execute proper merging maneuvers and make proper car-following actions as vehicles on the highway usually do.

[1]The off-board Road-Side Unit *(RSU)* contains its sensors (camera, RADAR) and has the *V2I* communication capabilities.

[2]Speed limit at the highway on-ramp section of *US Interstate Highway I-80* in *Emeryville* (*San Francisco*), *California*, *USA*.

[3]The maximum normal deceleration range of a main road vehicle [54].

[4]Geometry of the highway on-ramp section of *US Interstate Highway I-80* in *Emeryville* (*San Francisco*), *California*, *USA*.

The set of states *(S)*, set of actions *(A)* and set of rewards *(R)*, of the highway on-ramp merging deep reinforcement agent are given by:

- **State *S*:** Each state $s \in S$ is a tuple:
  $< d_m, v_m, a_m, ..., d_{F_2}, v_{F_2}, a_{F_2} >$ where d_m, v_m, a_m are, respectively, the distance between the merging vehicle projection and the merging point, the speed of the projected merging vehicle and the acceleration of the projected merging vehicle. d_i, v_i, a_i $(i \in \{P_2, P_1, F_1, F_2\})$ are, respectively, the relative distance, the relative speed and the relative acceleration between vehicle i and the merging vehicle projection.

- **Action *A*:** Each action $a \in A$ is a tuple $<a>$ where a is the longitudinal acceleration control input to the merging vehicle. The acceleration control input for the merging vehicle is strictly within *[-5 m/s^2, 3 m/s^2]*, which is the same normal acceleration range of a main road vehicle [16].

- **Reward *R*:** Each reward $r \in R$ is defined as:

  - if the merging vehicle V_m is at the acceleration lane (reward zone in Figure 4.2):
    $$r = -0.05.(\frac{|d_{F_1} - d_{P_1}|}{|d_{F_1} + d_{P_1}|} + \frac{|v_{P_1} - v_m|}{10}) \tag{4.9}$$
  - if V_m is within *100 m* before the acceleration lane (as illustrated in Figure 4.2):
    $$r = 0 \tag{4.10}$$
  - if V_m reaches the end of the control zone (success) and the episode ends:
    $$r = +1 \tag{4.11}$$
  - if V_m performs a stop and the episode ends:
    $$r = -0.5 \tag{4.12}$$
  - if V_m performs a collision and the episode ends:
    $$r = -1 \tag{4.13}$$

  The reward term in equation *(4.9)* motivates the merging vehicle to merge midway between the preceding vehicle and the following vehicle to maximize the

safety distance, by minimizing the ratio between the distance from the midway point ($\frac{|d_{F_1} - d_{P_1}|}{2}$) and the available gap ($\frac{|d_{F_1} + d_{P_1}|}{2}$). Also, it motivates merging with the same speed as the preceding vehicle by minimizing the speed difference ($|\ v_{P_1}$ - $v_m\ |$), and normalizing by a speed difference of *10 m/s*.

4.4 Experimental setup

In order to train the autonomous highway on-ramp merging agent, we create the highway scenario under the environment *SUMO* (*Simulation of Urban MObility*, see Appendix A). We use *SUMO*, an open-source, microscopic traffic simulator, for its ability to handle large, complicated road networks at a microscopic (vehicle-level) scale, as well as easily query, control, and extend the simulation through *TraCI* (see Appendix B), which will be used in our thesis. As a microscopic simulator, *SUMO* provides several car-following and lane-change models to dictate the longitudinal and lateral dynamics of individual vehicles [55]. Under *SUMO*, we created a highway on-ramp situation with similar geometry as the highway on-ramp section of US Interstate Highway *I-80* in *Emeryville (San Francisco), California, USA,* as illustrated in Figure 4.3.

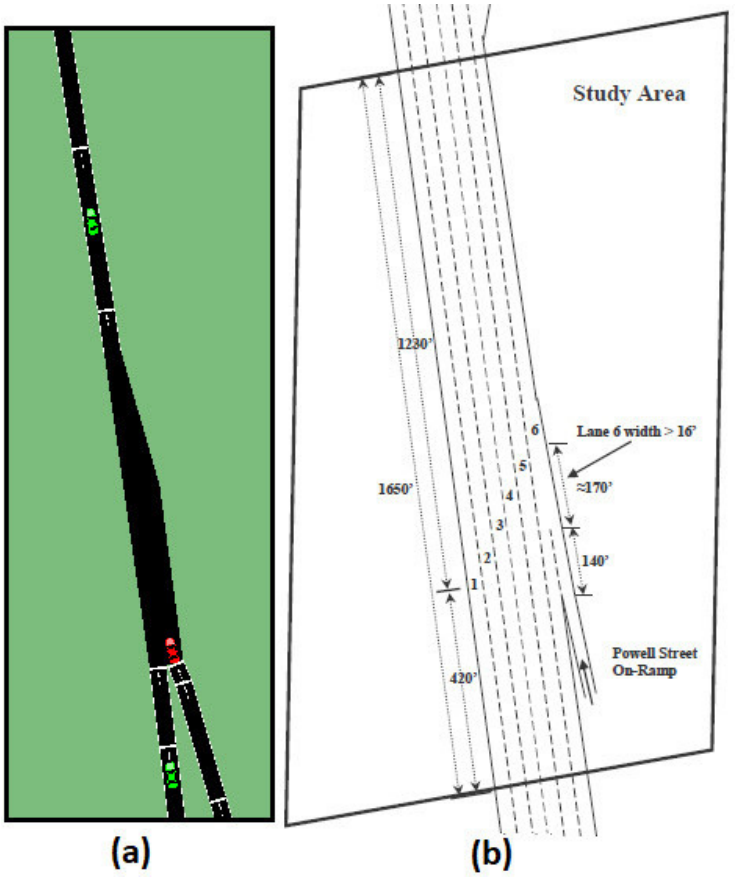


FIGURE 4.3: Ramp merge (a) simulated scenario under *SUMO* and (b) real-world location schematic.

The training is based on such simulated scenarios of real-world merging areas. Also, the traffic flow under *SUMO* corresponds to the vehicles' trajectories on a segment of interstate *80* in *Emeryville (San Francisco), California* collected between *4:00 p.m.* and *4:15 p.m.* on *April 13, 2005*, extracted from the Next Generation Simulation (*NGSIM*) database. Main-road vehicles are intelligent such that they can perform car following and collision avoidance based on the *Krauss' model* [56]. Main-road speed limit is v_{limit} = *29.06 m/s*. Each vehicle on the main road has a desired average speed in the range of *[22 m/s, 34 m/s]*. Regarding the traffic flow characteristics extracted from the *NGSIM* database, the flow of vehicles on the main road is around one vehicle entry at each *3.25 seconds*, and follows a normal distribution with a mean of *3.25* and a standard deviation of *0.1*. That means that a vehicle will enter the main lane every *3* to *3.5 seconds*. All the main-road vehicles have normal acceleration values in *[-5 m/s*^{2}*, 3 m/s*^{2}*]*. When abnormal situations occur, such as the merging vehicle merges too closely to the following vehicle, the latter can decelerate further to the minimum *-9 m/s*^{2} which is the emergency braking deceleration. These definitions for the main-road vehicles remain the same during training and testing. For the vehicles in the main lane, we consider a driver model that controls the longitudinal acceleration of the vehicle while taking into account the projection of the merging vehicles (green color in Figure 4.2). As was proposed in [16], we introduce a cooperation level $C \in [C_{min};\ 1]$, which is a scalar parameter controlling the reaction to the merging vehicle state when it is on the acceleration lane. $C = 1$ represents a driver who slows down to yield to the merging vehicle if he/she predicts that the merging vehicle will arrive ahead of time. $C = C_{min}$ represents a driver who is less cooperative regarding the merging vehicle. This model relies on estimating the time to reach the merge point *(TTM)* for the car in the main lane (TTM_l) and the car in the merge lane (TTM_m), to decide whether or not the merging vehicles should be considered. Once the time to merge for both vehicles has been estimated, three cases are considered:

- If $TTM_m < C\ .\ TTM_l$, the vehicle in the main lane takes into account the merging vehicle by considering its projection in the main lane as its front vehicle.
- If $TTM_m >= C\ .\ TTM_l$, the vehicle in the main lane ignores the merging vehicle and follows its standard behavior.
- In the absence of or at a distance from the merging vehicle, the driver in the main lane follows his/her standard behavior.

These rules for main-road vehicles model the real-world traffic where various vehicle types depending on each manufacturer, and different drivers' behaviors.

4.5 Results

4.5.1 Using a safe controller

In *RL*, techniques for selecting actions during the learning phase are called exploration/ exploitation strategies. Most exploration methods are based on heuristics, rely on statistics collected from sampling the environment or have a random exploratory component (e.g., *Ornstein-Uhlenbeck* process noise). However, most of those exploration methods are blind to the risk of actions [57] and can become expensive, particularly when learning on a physical platform such as a robotic platform [58]. For example, while training a highway driving agent, use of unguided exploration could frequently lead to collision or near-miss scenarios. This may result in simulation resets, thus slowing down the learning process. Additionally, even after convergence, due to the function approximation by the Q-network, the trained agent may choose a non-safe maneuver [59]. In order to address these issues, we add a safe controller (SC) that evaluates the control action (a) outputted by the *DDPG* agent according to some predefined security rules, and provides alternative safe action (a_s) if these rules are not respected. In these cases, the safe action (a_s) replaces the agent control action (a) also in Algorithms **1** and **2**. The SC is a key component that is used both while learning and during the exploitation phase. For the highway on-ramp merging use-case, we propose that the safe action (a_s) is obtained according to the following rules:

- ***Rule 1:*** $a_s = 0\ m/s^2$ if $((v_m > v_{limit})\ \&\ (a_m > 0))$
- ***Rule 2:*** $a_s = a_{min}$ if $((d_{p_1} - T_{min}.v_{P_1}) > d_{min})$
- ***Rule 3:*** $a_s = a_{max}$ if $(v_m < v_{rcmnd})$

Rule 1 prevents the vehicle from exceeding the highway speed limit (v_{limit}). Rule 2 ensures that the merging vehicle performs a minimum deceleration (a_{min}) to keep a relative gap from the preceding vehicle (d_{p_1}) above a *minimum-safety distance* of d_{min}= *40 m* for a *minimum time to collision* of $T_{min} = 2\ s$. Last but not least, Rule 3 ensures that the merging vehicle performs a maximum acceleration (a_{max}) until speed reachs at least a recommended speed $v_{rcmnd} = 22.5\ m/s$. These values are based on the standard of the US Department of Transportation [60].

In order to show the role of safe controller for ensuring convergence of the *RL* agent, we train a *DDPG* agent and then add a safe controller *(SC)* that provides alternate safe actions. The *DRL* architecture used for the autonomous highway on-ramp driving is given in Figure 1.4.a: vehicles' state information including the merging vehicle (V_m) and

its surrounding vehicles (F_1, F_2, P_1 and P_2) is provided as inputs to the *DDPG* agent. The *DDPG* agent outputs a longitudinal acceleration control action *(a)*. For the DDPG agent with a safe controller (*DDPG-SC* in Figure 1.4.b), the safe controller checks if this action respects safety requirements. If not, it outputs an alternative control action (a_s) to the merging vehicle that respects security rules stated above, and which is as close as possible to the *DDPG's* action *(a)*.

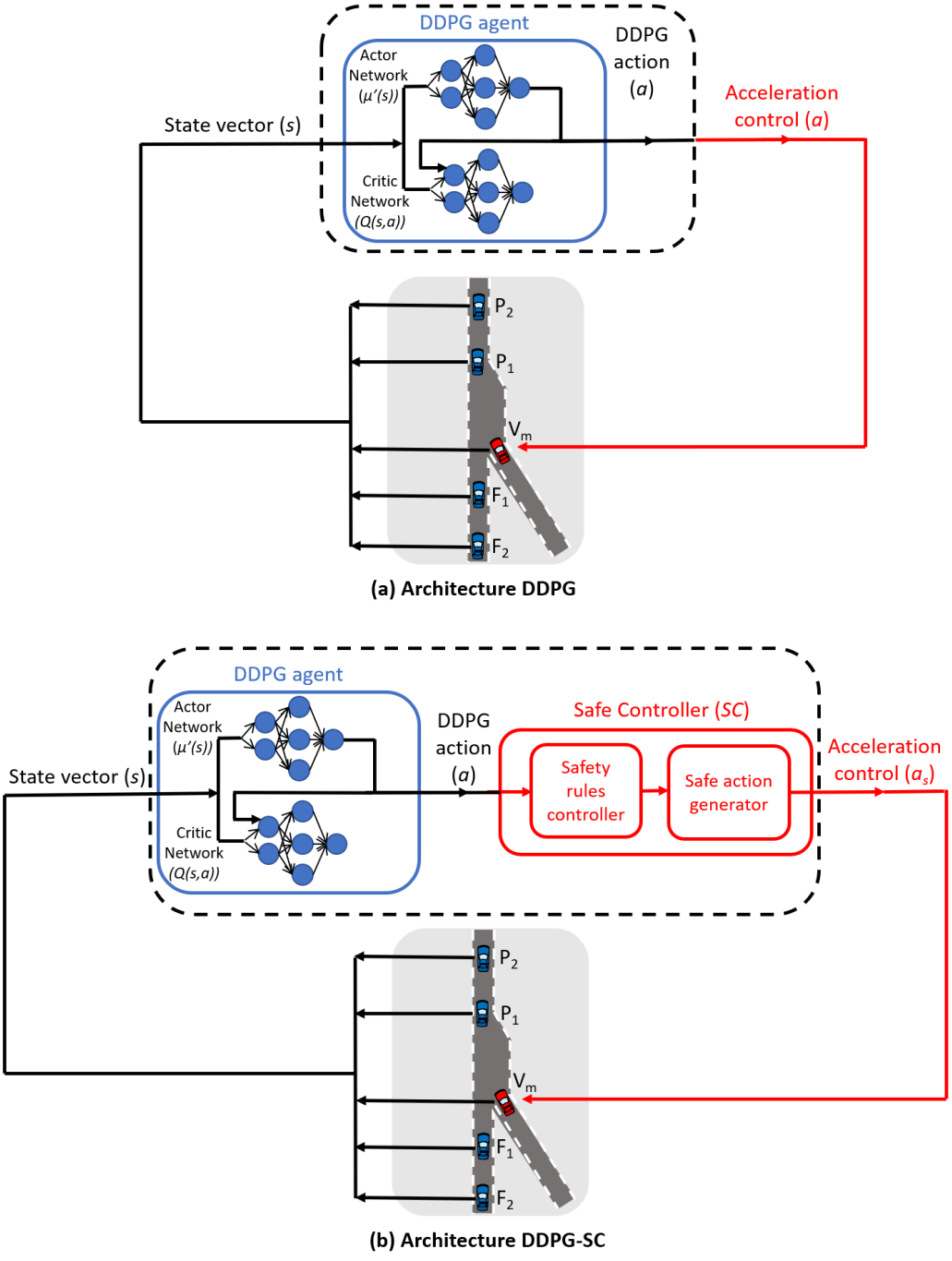


FIGURE 4.4: DRL architecture (a) DDPG agent and (b) DDPG agent with safe controller (*DDPG-SC*).

The *DDPG* and *DDPG-SC* agents were trained for *1* million simulation time steps with the parameters in Table 4.1, which were used for autonomous driving on the highway [16] [17].

Target network update coefficient τ	0.001
Reward discount factor γ	0.99
Actor learning rate	0.0001
Critic learning rate	0.001
Experience replay memory size R	400000
Mini-batch size N	64
Ornstein-Uhlenbeck σ	0.1
Ornstein-Uhlenbeck Θ	0.05

TABLE 4.1: *DDPG* agent training parameter values using for algorithm 1.

Figure 4.5 shows the average undiscounted reward (over *100* episodes) when we train the DDPG agent without the safe controller (*DDPG*), and with the safe controller (*DDPG-SC*).

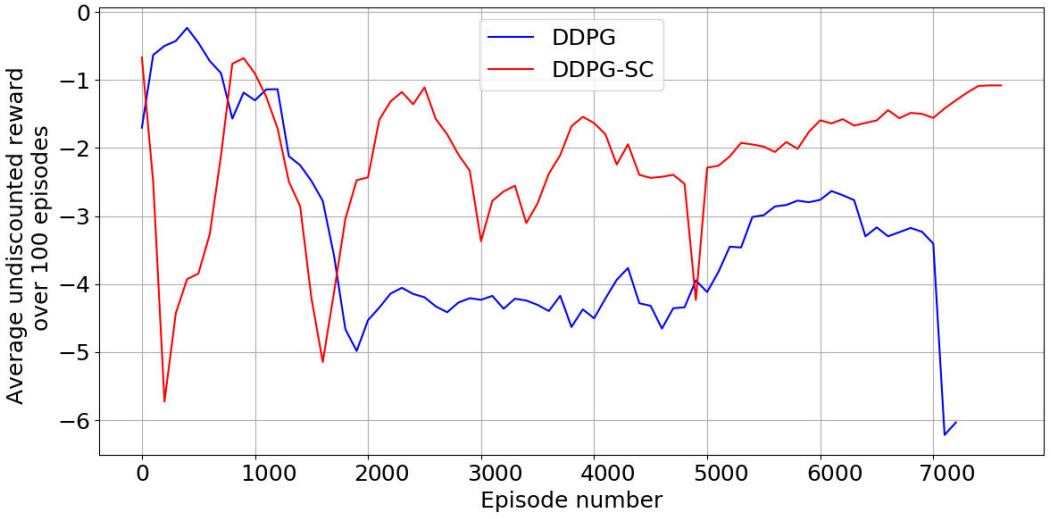


FIGURE 4.5: Average undiscounted reward over *100* episodes during training

In Figure 4.5, when the *DDPG* agent is trained without the safe controller, the average reward drops toward negative values. After adding a safe controller that replaces unsafe actions, a reasonable convergence of the average cumulative episode reward occurs after *6000* episodes, since the safe controller reduces the number of episodes where there is a collision or stop. As shown in Figure 4.5, at the initial training phase, the *DDPG-SC* agent had episode rewards that fluctuate very often toward negative values, indicating many stops or collisions. At the end of the training, as the agent learned a meaningful policy, the episode rewards increased and converged to an optimal value. This shows that the safe controller is a key component in learning a meaningful policy and accelerating

convergence.

We tested the trained policy for another 1 million simulation time steps and recorded the total number of episodes and the number of stops, collisions, and successes. We also recorded the emergency brakings times wherein the merging vehicle decelerates to *-9* m/s^2. The training results are summarized in the following Table 4.2.

Total number of episodes	8015
Number of stops	0 (0 %)
Number of collisions	1 (0,000125 %)
Number of emergency brakings by F_1	8 (0,001 %)
Number of successful episodes	8014 (99,98 %)
Average speed	23.4 m/s
Average gap from midway	23.2 m
Average speed difference from P_1 (v_{P_1})	10.5 m/s

TABLE 4.2: *DDPG-SC* agent testing results

From Table 4.2, there are *1* collision and *8* emergency brakings for *8015* merging episodes, despite the midway safety criteria in the reward function. This means that a simple deep reinforcement learning approach cannot learn a safe and cooperative driving strategy. Yet, such a solution cannot be implemented in a real-world system.

4.5.2 Using the driver intention model

As shown before, a deep *RL* agent with a safe controller cannot learn, implicitly, a safe and cooperative driving policy. To improve the performance of the proposed *DDPG-SC* agent, we improve our architecture by incorporating a model that predicts the behavior of the vehicle below the merging point at the main highway lane (cf. Figure 3.3), with the assumption that this may increase the cooperativeness, hence safety, of the merging vehicle regarding the vehicle at the main highway lane. We use the driver's intention model (*DIM*) with the same architecture as stated in section 3.2. This model is trained by the artificial neural network using *SUMO* environment, where it shows same performance values as in Table 3.2. Our proposed *DRL* architecture is given in Figure 4.6.b: vehicles' state information is provided to the *DDPG* agent and to the Driver Intention Model *(DIM)*. The driver model outputs the intention and provides it as an input state to the *DDPG* agent. The *DDPG* agent then outputs a control action that will be checked by the safe controller (*SC*) and replaced if it does not respect safety rules. The intention (I_l) of the driver below the merging point in the main lane, provided by the model *DIM*, is used as a new input to the *DDPG* agent. Hence, the state tuple becomes $s = < d_m,\ v_m,\ a_m,..., d_{F_2},\ v_{F_2},\ a_{F_2},\ I_l >$. Moreover, the intention can be considered in the reward function by assigning a negative reward (*-0.5*) when

the intention of the driver in the main lane changes near to the on-ramp merging zone. This motivates the *DDPG* agent to learn a cooperative merging policy regarding the behavior of the human driver in the main highway lane. We thus trained and compared three *DDPG* agents:

- ***DDPG-SC:*** DDPG agent with safe controller. This agent is considered as the classic approach of reference, and the baseline performance.
- ***DDPG-SC-I:*** DDPG agent with safe controller and driver intention as an input state.
- ***DDPG-SC-IR:*** DDPG agent with safe controller and driver intention as an input state and in the reward, by assigning a negative value as stated above.

These agents were trained using the same parameters as in Table 4.1 and were compared with the results obtained by the previous agent (*DDPG-SC*). Figure 4.7 shows the average undiscounted reward (over *100* episodes) for each agent.

In Figure 4.7, when using driver intention as an input to the state vector, the *DDPG* agent (*DDPG-SC-I*) converges toward higher reward values after only *450K* simulation steps (*3200* episodes in Fig.4.7). This is considerably faster, by *55%*, than the first agent (*DDPG-SC*) that did not include the intention, and which converges after *1* million simulation steps (*7500* episodes in Fig.4.7). Also, the average reward of the agent that includes driver intention (*DDPG-SC-I*) converges to a final value of *-0.4* (as in Fig.4.7), which is greater than the (*DDPG-SC*) agent's value (*-1* in Fig.4.7), and presents fewer fluctuations at the end of training. Inversely, when training the *DDPG* agent with included intention both as an input state and in the reward term (*DDPG-SC-IR*), the algorithm converges after *1.5* million simulation steps (*11400* episodes in Fig.4.7), which is *50%* slower than the first *DDPG-SC* agent. This is because the complexity of the reward objective increased after adding a negative penalty concerning driver's intention. As the *DDPG-SC-I* agent, the average reward has also a greater final value (*-0.8* in Fig.4.7) and fewer fluctuations than the *DDPG-SC* agent.

To compare and validate the performances of each agent, we tested each one during *1* million steps. The results are summarized in Table 4.3, and are compared to the simulator built-in controller. From this table, the simulator shows a very high number of emergency braking (*204* cases), and a very high average speed (*28 m/s*), which means that a rule-based controller does not guarantee meaningful merging policy. The classic *RL* approach (*DDPG-SC* agent) shows better results over the simulator's controller.

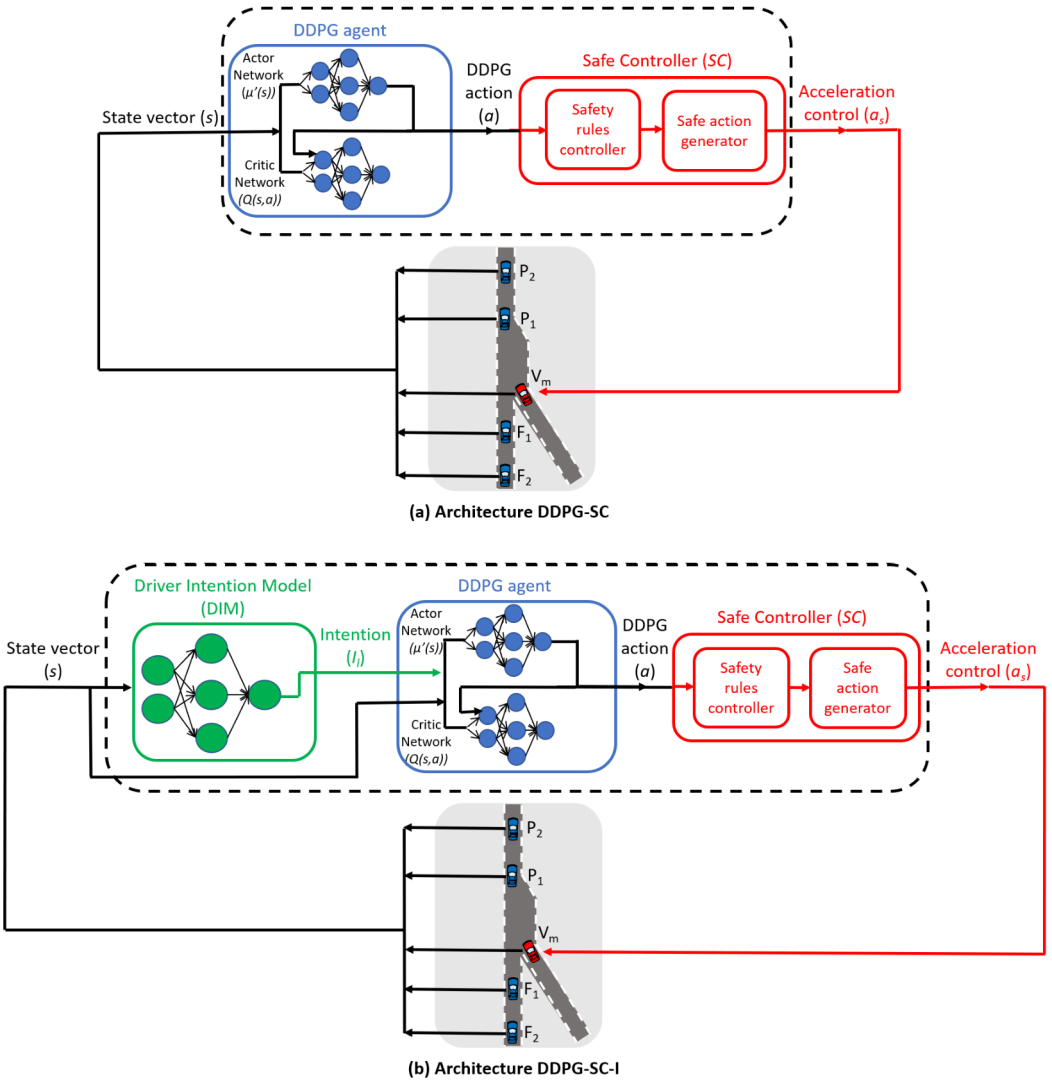


FIGURE 4.6: Vehicle architecture: (a) *DDPG* agent with safe controller (*DDPG-SC*) (b) *DDPG* agent with safe controller and driver intention model (*DDPG-SC-I*)

When adding the intention estimation as an input state to the *DDPG* agent (*DDPG-SC-I*), the number of collisions drops to zero (unlike the *DDPG-SC* agent that did not include driver intention estimation). Moreover, the number of emergency brakings drops to only *3* cases (*8* cases when we use the *DDPG* agent without driver intention estimation). When the driver's intention is added as an input state and in the reward (*DDPG-SC-IR*), the collision case disappears. However, the number of emergency brakings and the average merging speed increase, which means that the agent learns a more aggressive driving policy. This is because the complexity of the reward objective increased after adding a negative penalty concerning the driver's intention, compared to the previous reward formulation where the agent learns only safety-related policy. When

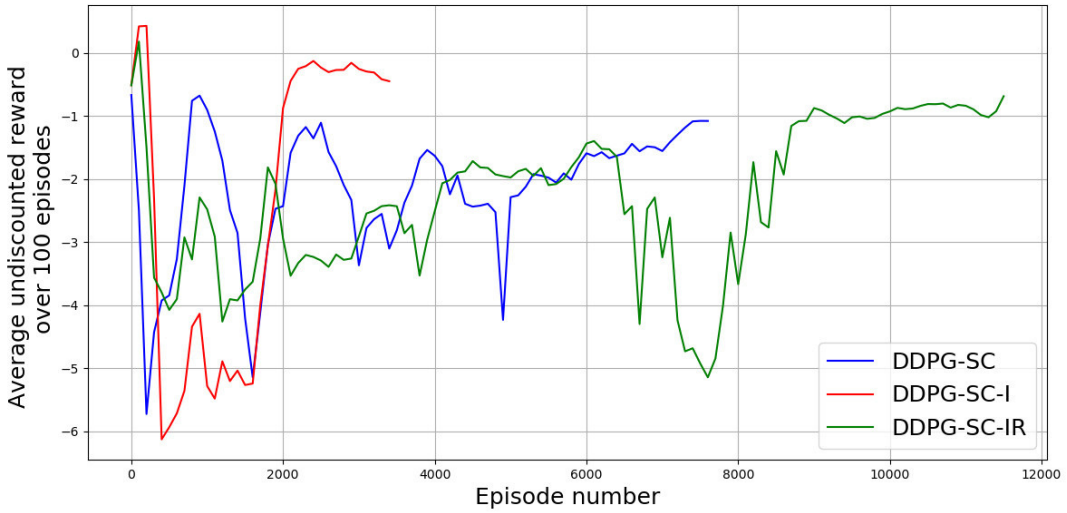


FIGURE 4.7: Average undiscounted reward over *100* episodes during training

comparing the other performance metrics, the *DDPG-SC-I* agent has the best performances in following the speed of the preceding vehicle (*7.8 m/s*). Moreover, it has the highest average merging speed (*26 m/s*) compared to the *DDPG-SC* and *DDPG-SC-IR* agents. This means that the learned driving policy is the most adapted for high-speed highway scenarios.

Performances	*Simulator*	*DDPG-SC*	*DDPG-SC-I*	*DDPG-SC-IR*
Number of episodes	8459	8015	8100	8081
Number of stops	0	0	0	0
Number of collisions	0	1	0	0
Number of emergency brakings by vehicle F_1	204	8	3	11
Number of successful episodes	8459	8014	8100	8081
Average speed (m/s)	28	23.4	26	25.5
Average gap from midway point (m)	24.7	23.2	24.5	23.7
Average speed difference from vehicle P_1 (m/s)	5.8	10.5	7.8	8.4

TABLE 4.3: *DDPG* agents testing results

We conclude that adding a driver intention prediction as an input state to the *DDPG* agent improves safety performance by eliminating collision cases and reducing the number of emergency brakings. Moreover, training is, *55%*, faster than a *DDPG* agent that does not consider the driver intention model.

4.5.3 Using the Twin Delayed Deep Deterministic Policy Gradient (*TD3*) agent

While *DDPG* algorithm achieves better performance when combining with the driver's intention estimation, it can suffer from the function approximation errors which lead to overestimated values and suboptimal policies. As stated earlier, the twin delayed deep deterministic policy gradient (*TD3*) algorithm was proposed to address these issues for continuous state-action control problems. Yet, no previously published work has used the *TD3* algorithm for autonomous highway driving. In order to check the performances of such an algorithm for highway on-ramp merging, we train and compare four autonomous agents:

- ***DDPG:*** DDPG agent without driver's intention estimation. The input state is $s = [d_i, v_i, a_i]$ where $i \in \{m, P_2, P_1, F_1, F_2\}$
- ***DDPG-I:*** DDPG agent with the main lane driver's intention estimation (I_l) provided by the off-board road-side unit. The input state is $s = [d_i, v_i, a_i\ ;\ I_l]$ where $i \in \{m, P_2, P_1, F_1, F_2\}$
- ***TD3:*** TD3 agent without driver's intention estimation. The input state is $s = [d_i, v_i, a_i]$ where $i \in \{m, P_2, P_1, F_1, F_2\}$
- ***TD3-I:*** TD3 agent with the main lane driver's intention estimation (I_l) provided by the off-board road-side unit. The input state is $s = [d_i, v_i, a_i\ ;\ I_l]$ where $i \in \{m, P_2, P_1, F_1, F_2\}$

The architecture of each agent is illustrated in Figure 4.8. As shown in Figure 4.8.a: vehicles' state information including the merging vehicle (V_m) and its surrounding vehicles (F_1, F_2, P_1 and P_2) is provided as an input to the *DDPG* (resp. *TD3*) agent. The *DDPG* (resp. *TD3*) agent outputs a longitudinal acceleration control action. The safe controller checks if this action respects safety requirements. If not, it outputs an alternative control action to the merging vehicle that respects security rules, and resembles as closely as possible the *DDPG*'s (resp. *TD3*'s) action. In order to improve the performance of the proposed agents *(DDPG, TD3)*, we incorporate the driver's intention model (*DIM*) as stated in section 4.5.2. Our proposed *DRL* architecture becomes as shown in Figure 4.8.b: vehicles' state information is provided to the *DDPG-I* (resp. *TD3-I*) agent at the on-board vehicle unit, and to the Driver Intention Model at the off-board road-side unit. The driver model outputs the intention and provides it as an input state to the *DDPG-I* (resp. *TD3-I*) agent. The *DDPG-I* (resp. *TD3-I*) agent then outputs a control action that will be checked by the safe controller to be replaced if it does not respect safety rules.

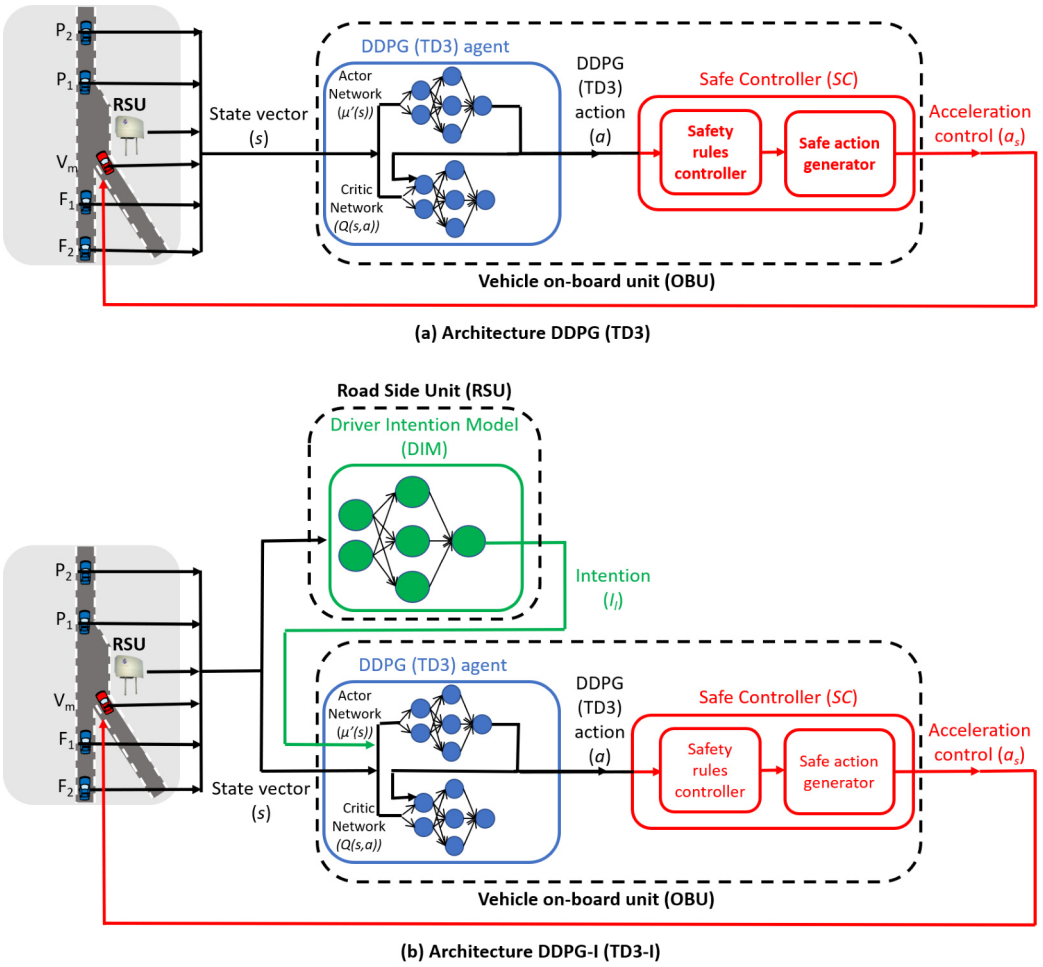


FIGURE 4.8: Vehicle architecture: (a) Agent with safe controller (*DDPG*, *TD3*) (b) Agent with safe controller and driver intention model (*DDPG-I*, *TD3-I*)

The agents were trained using the following parameters:

Target network update coefficient (*DDPG*) τ	0.001
Target network update coefficient (*TD3*) τ	0.005
Reward discount factor γ	0.99
Actor learning rate	0.0001
Critic learning rate	0.001
Experience replay memory size R	400000
Mini-batch size N	64
Ornstein-Uhlenbeck σ	0.1
Ornstein-Uhlenbeck Θ	0.05
Policy delay (*TD3*) d	2

TABLE 4.4: Agents training parameter values.

Figure 4.9 below shows the average undiscounted reward (over 100 episodes) for each agent. As was shown in section 4.5.2, the *DDPG-I* agent converges, *55%*, faster with greater finals values and fewer fluctuations at the end of training, and that when using

driver intention as an input to the state vector. On the other hand, the *TD3* agent converges after *1.8* million simulation steps (*13200* episodes in Fig. 4.9) when driver's intention is not provided as an input state, which is, *80%*, slower compared to the *DDPG* agent. This is due to the fact that the policy network (actor) of the *TD3* agent is updated less often than the critic network. When the driver's intention estimation is added as an input state, the *TD3-I* agent converges after *1* million simulation steps (*7200* episodes in Fig.4.9) which is, *55%*, faster than the *TD3* agent that do not include the driver's intention estimation. As for the *DDPG-I* agent, the *TD3-I* converges to a final value of *-0.7* (as in Fig.4.9), which is greater than the (*TD3*) agent's value (*-1.25* in Fig.4.9), and presents fewer fluctuations at the end of training.

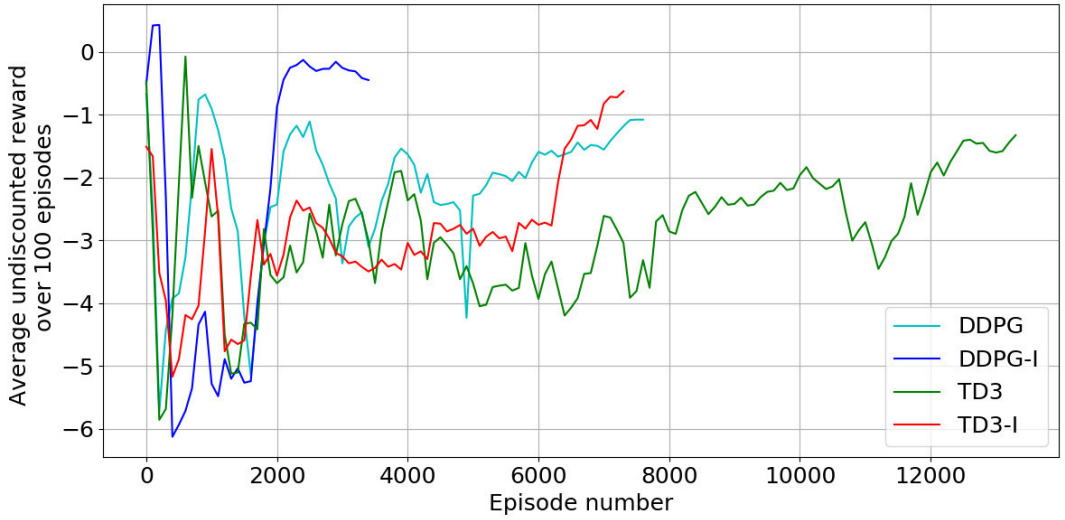


FIGURE 4.9: Average undiscounted episode reward during training.

To check and validate the performance of these agents, we tested each trained policy for *1* million simulation time steps and recorded the total number of successful episodes, the number of stops, and the number of collisions. We also recorded the emergency brakings times wherein the merging vehicle decelerates to *-9* m/s^2. The testing results are summarized in Table 4.5 below.

From table 4.5, there are *1* collision and *8* emergency brakings for *8015* merging episodes when using the *DDPG* agent, even with the midway safety criteria in the reward function. As was shown in section 4.5.2, the number of collisions drops to zero and the number of emergency brakings drops to only *3* cases when adding the intention estimation as an input state to the *DDPG-I* agent. The *TD3* agent shows no collision and only 2 cases of emergency brakings, which is better than its counterparts: *DDPG* and *DDPG-I*. When adding a driver's intention as an input state to the *TD3-I* agent, the learned policy shows neither collision nor emergency brakings cases. This is the best safety performances compared to all the trained agents. Moreover, the *TD3-I* agent

Performances	*DDPG*	*DDPG-I*	*TD3*	*TD3-I*
Number of episodes	8015	8100	7808	8026
Number of stops	0	0	0	0
Number of collisions	1	0	0	0
Number of emergency brakings by vehicle F_1	8	3	2	0
Number of successful episodes	8014	8100	7808	8026
Average speed (m/s)	23.4	26	22.5	25.35
Average gap from midway point (m)	23.2	24.5	23.4	28.54
Average speed difference from vehicle P_1 (m/s)	10.5	7.8	11.36	8.53

TABLE 4.5: Agents testing results

has an average speed of *25.35 m/s*, which is higher than the average speed of *DDPG* and *TD3* agents. This means that the *TD3-I* driving policy is the most adapted for high-speed highway on-ramp scenario.

In short, we conclude that the *TD3* algorithm gives better performance than its counterpart *DDPG* in all cases. Adding the driver's intention is a key component for accelerating convergence and learning *"safe"* and *"cooperative"* autonomous driving policy.

4.6 Conclusion

This chapter presented a novel architecture for autonomous highway on-ramp merging.

First, using a safe controller (*SC*) is a key component to ensure convergence during training, and reduces the risk of collisions after deploying an autonomous driving agent in the real world, as was shown in section 4.5.1. This safe controller checks the action provided by the *DRL* agent according to some predefined safety rules, and replaces the unsafe actions.

Then, section 4.5.2 presented a novel architecture that combines the prediction provided by the driver intention model (*DIM*) as an input state to the deep deterministic policy gradient (DDPG) agent. This approach accelerates the convergence of the learned autonomous driving policy by *55%*. Moreover, it eliminates the collision cases and reduces the number of emergency brakings maneuvers. This means that using a driver's model that feeds the *DRL* agent is a key element to learn a *"safe"* and *"cooperative"* autonomous driving policy.

Finally, section 4.5.3 presented the use of the Twin Delayed DDPG (*TD3*) algorithm that was not used yet for high-speed highway on-ramp merging. The *TD3* outperforms the *DDPG* algorithm, and eliminates all the collisions and emergency brakings cases, when combined with the driver intention model (*DIM*).

In summary, an architecture that combines a driver's model implemented at the off-board road-side unit, and provides intention prediction as input to a *TD3* agent implemented at the on-board vehicle, could perform autonomous highway on-ramp merging *"safely"* and *"cooperatively"*.

Chapter 5

Conclusions and future work

In this thesis, we proposed a novel architecture for autonomous high-speed highway on-ramp merging. Highway on-ramp is one of the riskiest and challenging situations for autonomous driving. Our new approach is based on the collaboration between the off-board road infrastructure and the on-board vehicle's autonomous driving system to achieve safe and cooperative highway on-ramp merging, even in the presence of non-connected human-driven vehicles. Although some previous works have proven that the road infrastructure has an increased perception range, and it may be a key support for autonomous driving (*AD*), yet no work has studied how the road infrastructure could be used to improve autonomous driving for connected and autonomous vehicles.

In chapter 3, we proved how a probabilistic model, implemented at the off-board road-side unit, could be used to predict drivers' intentions at the highway on-ramp, with an accuracy that exceeds *99%*. This model was trained and validated using real-world data provided by the road infrastructure (*NGSIM* database). The main advantage of using such a data-driven model at the off-board infrastructure is that it learns the intentions of drivers for a specific driving situation (highway on-ramp) and geographic location. Therefore, the prediction provided by the model is robust concerning the variety of drivers' behaviors, regardless of the used probabilistic classifier. The comparison between such an off-board model and its on-board counterpart confirms the importance of using road infrastructure to increase perception range and predict drivers' intentions accurately. Also, we demonstrated that this prediction can be provided from the road-side unit to the autonomous vehicle at the merge lane in real-time considering current *V2X* communication latency and vehicle's dynamic. This driver's model can be extended to the follower driver at the main lane to increase the prediction time horizon.

In chapter 4, we proposed a new architecture for autonomous highway on-ramp merging that combines the off-board driver intention model with an on-board deep reinforcement learning agent. First, we proved that a continuous action-state reinforcement learning framework is the most adapted for such a driving scenario where there is uncertainty in other drivers' behaviors and their dynamic models. We showed that using a safe controller (*SC*) is a key component to ensure convergence during training, and reduces the risk of collisions after deploying an autonomous driving agent in the real world. The safe controller checks the action provided by the *DRL* agent according to some predefined safety rules and replaces the unsafe actions. Then, we presented a novel architecture that combines the prediction provided by the driver intention model (*DIM*) as an input state to the deep deterministic policy gradient (*DDPG*) agent. This approach accelerates the convergence of the learned autonomous driving policy by *55%*. Moreover, it eliminates the collision cases and reduces the number of emergency brakings maneuvers. This means that: *"classic deep reinforcement learning approach does not learn, implicitly, a cooperative driving policy. Using a driver's model that feeds the DRL agent is a key element to learn "safe" and "cooperative" autonomous driving policy"*. A comparison between the twin delayed deep deterministic policy gradient (*TD3*) and *DDPG* algorithms shows that the *TD3* effectively outperforms the *DDPG* agent, although it needs more training steps. Using driver's model intention as input state to the *TD3* agent shows zero collisions and emergency brakings cases, which is the best performances so far.

When taken together, this thesis presented a novel solution for using the road infrastructure with *V2X* communication to improve the performances of autonomous driving systems, even with the presence of human-driven vehicles. The novel architecture combines a data-driven driver's model at the off-board infrastructure that provides information to a *TD3* agent at the vehicle's on-board to perform autonomous high-speed highway on-ramp merging, *"safely"* and *"cooperatively"*.

5.1 Future Work

In the scope of this thesis, we studied our novel solution at the highway on-ramp situation, which is one of the riskiest and dangerous scenarios for autonomous driving systems. Future work may be dedicated to applying our approach to various other challenging scenarios, such as properly entering roundabouts, dealing with intersections, and other driving situations.

In chapter 3, we presented a model that predicts the intentions of *"Yield"* or *"Merge"* for the drivers at the highway on-ramp. This driver's model may be further improved and extended to predict deeper human behaviors' information such as time to arrive at the merge point, speed, and acceleration profiles. Such information may be used for decision making to improve safety performances. Moreover, the proposed model can be used for other driving scenarios like, as was mentioned, roundabouts and intersections. In each driving case, the context situation vector of the driver's model should be adapted by choosing the appropriate features (For example distance from the intersection point, relative speed from the vehicle at the roundabout...etc.). The model may be further improved by incorporating information provided by the *V2X* communication messages, such as the position of the accelerator or the brake position. Such information cannot be perceived directly by the road-side unit. However, it may greatly improve the accuracy and robustness of the prediction. Finally, the driver's model can be extended to the followers drivers at the main lane to increase the prediction horizon and improve the performance of the decision.

In chapter 4, the deep reinforcement learning framework presented in this thesis can be used for other driving situations by adapting the state vector. For example, when using our approach for intersections, the state vector of the *DRL* agent should contain information about the traffic-light system and the presence or absence of vehicles on the other lanes. For each driving situation, the geometry and scenario should be created on the simulation tool, and the *DRL* agent should be trained. Moreover, the *RL* reward presented in our thesis contains safety-related term only, which is the main performance criteria for driving systems. Reward formulation may be further improved by incorporating terms to reduce the acceleration jerk and to improve fuel consumption.
Since we proposed to use communication hybridization in section 1.1.2, the approach presented in this thesis may be further simulated and validated using a more advanced simulation environment that includes deeper *V2X* communication details such as current *DSRC* and *5G C-V2X* limitations (latency, data loss ...etc.). Also, more sophisticated vehicles' dynamic models may be used so that the *DRL* agent learns better driving policy during the exploration phase.

The big challenge is, of course, to test our decision-making strategy deployed on real connected and autonomous vehicles (*CAV*) with a real road-side unit (*RSU*).

Appendix A

SUMO simulator

The details below were extracted from the official *"SUMO"* website:
https:\\sumo.dlr.de\docs\Tutorials\Hello_World.html [61].

"Simulation of Urban MObility", or *"SUMO"* for short, is an open-source, highly portable, microscopic, multi-modal traffic simulation. It allows us to simulate how a given traffic demand which consists of single vehicles moves through a given road network. The simulation allows us to address a large set of traffic management topics [61]. It is purely microscopic: each vehicle is modeled explicitly, has an own route, and moves individually through the network. Simulations are deterministic by default but there are various options for introducing randomness.

To perform a very basic simulation in *SUMO*, it is required to have at least the following elements (files):

- Network
- Route
- SUMO configuration file

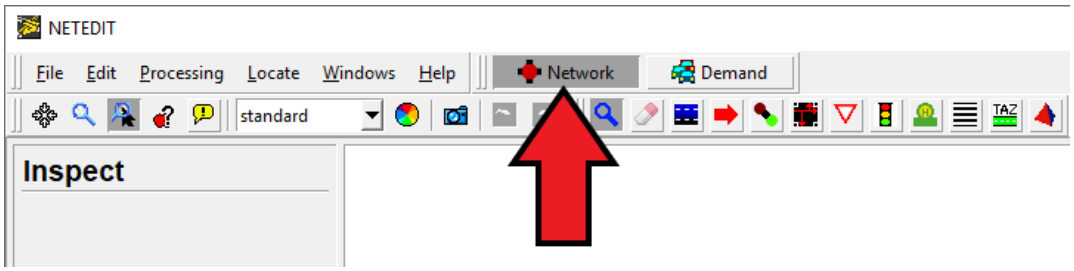
In *SUMO* a street network consists of nodes (junctions) and edges (streets connecting the junctions). In this tutorial, we will use *"netedit"* to create our basic net.

Routes are defined by connecting edges and assigning Vehicles that pass through them. In this tutorial, we will use *netedit* to create this.

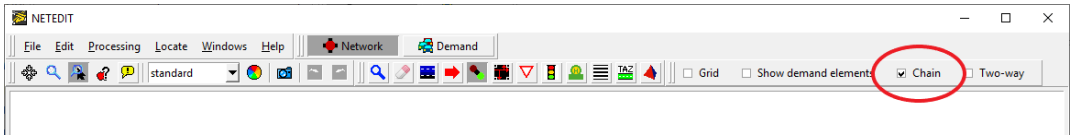
The *SUMO* Configuration file is where certain options and all files (Network, Route, etc.) are being listed so that *SUMO* can find and use them.

A.1 Creating the Network in netedit

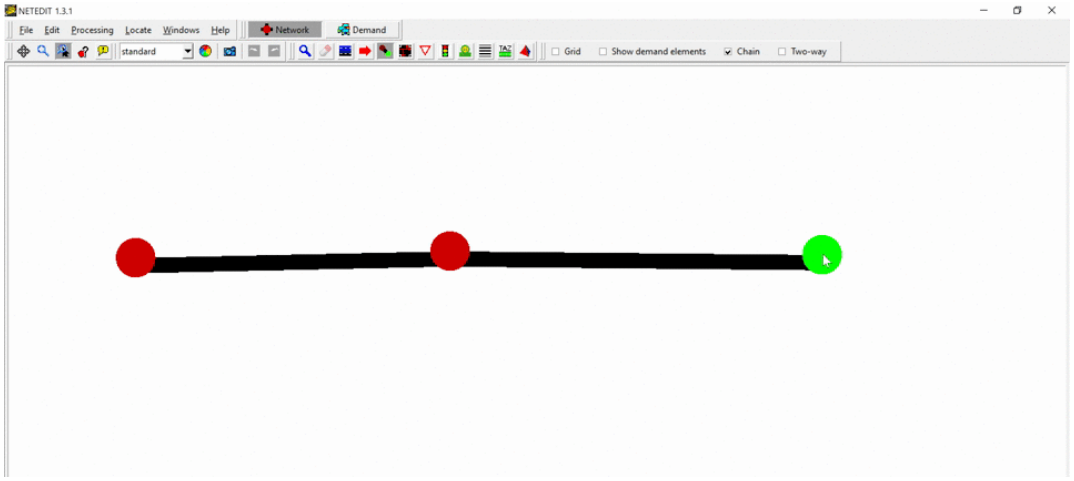
Open *netedit* and create a new network by selecting *File→New* Network or using the shortcut *Ctrl + N*. Make sure that **Network** is selected.



Enter **Edge Mode** by selecting *Edit→Edge* mode, using the shortcut *E* or by clicking on the button. In Edge Mode, make sure that **Chain** is selected. This will facilitate creating multiple nodes and their connecting edges with fewer clicks.



Nodes are created by clicking on empty spaces (when in *Edge Mode*). Insert 3 nodes (aka Junctions) by clicking on three different places at the white blank space. After inserting the last Node, press <*ESC*> to unselect that last node.

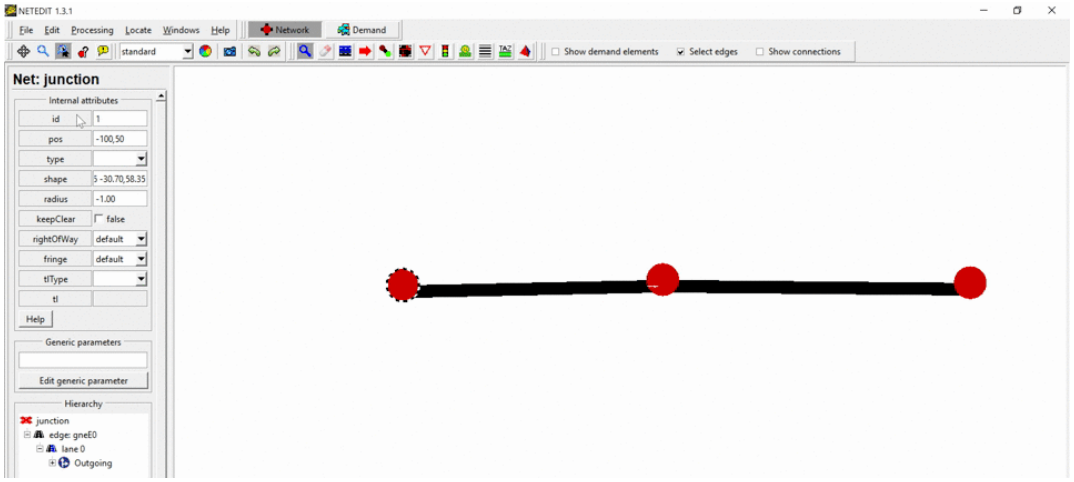


Now we want to rename our recently inserted Junctions and Edges (which get arbitrary id's when created) and also make our simple network prettier, by aligning all nodes. To do so, enter **Inspect** Mode by selecting *Edit Inspect* mode, using the shortcut *I* or by clicking on the button.

In *Inspect* mode you can select different type of elements, such as Nodes and Edges. If an element is selected, its properties will appear on the left side. Let's rename (change their **id**) the nodes to *"1"*, *"2"* and *"3"* (from left to right) and the edges to *"1to2"* and *"out"* (also left to right).

Replace the position (**pos**) of the nodes with the following values:

id	pos
1	-100,50
2	0,50
3	100,50



Our very basic network is done! We just need to save it . Use *File* → *Save Network* (*Ctrl* + *S*) or *File* → *Save Network As* (*Ctrl* + *Shift* + *S*) and give it a proper name (such as *"helloWorld.net.xml"*).

Do not close *netedit* yet, the demand still needs to be generated.

A.2 Demand Generation in netedit

Now, select the **Demand** supermode in *netedit*.



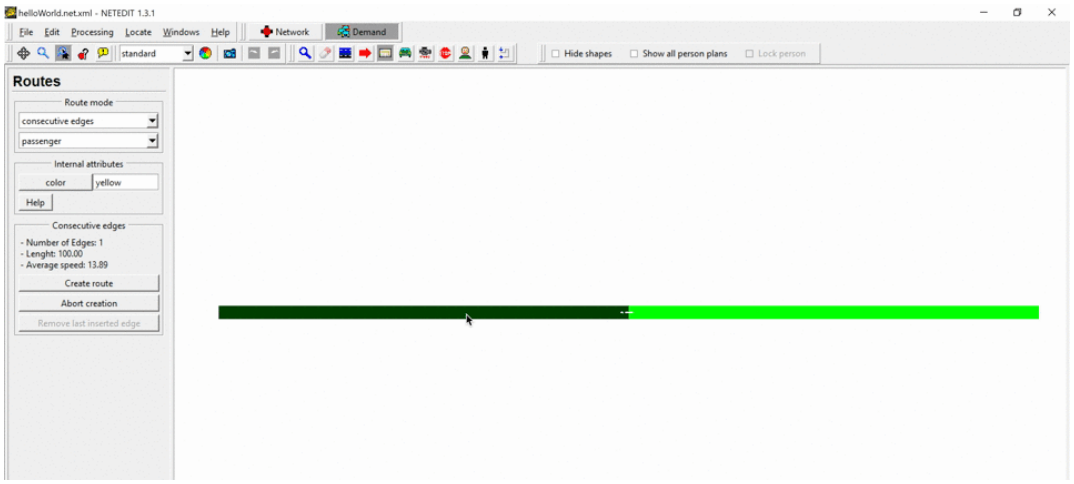
A.2.1 Creating a Route

Enter **Route Mode** by selecting *Edit* → *Route mode*, using the shortcut *R* or by clicking on the  button.

Creating a route is as simple as clicking on the Edges that will compose it. When selecting an Edge, its color will change.

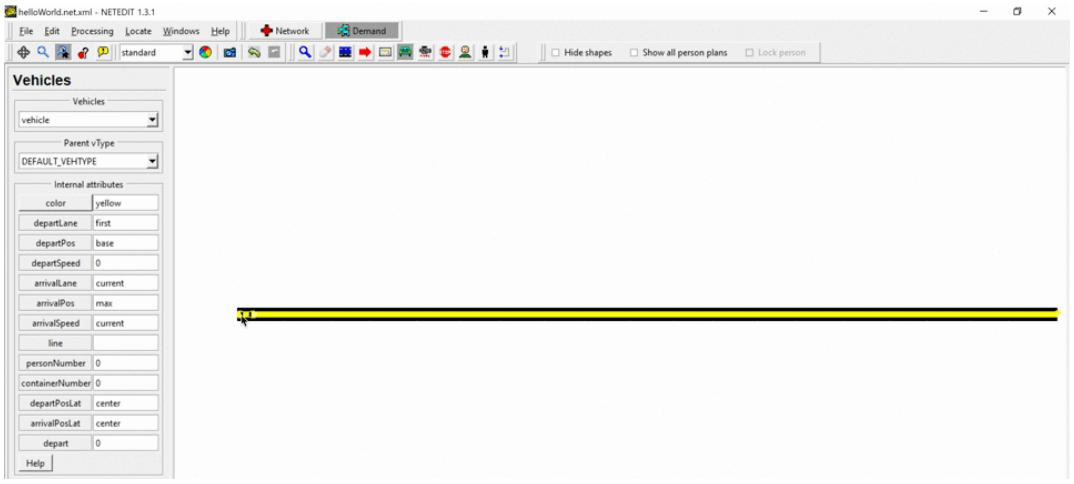
- Selected Edge
- Possible selectable edges

After selecting all the edges that will compose the desired route, click on *Create route*.



A.2.2 Adding a vehicle

Finally, enter **Vehicle Mode** by selecting *Edit* → *Vehicle mode*, using the shortcut *V* or by clicking on the  button. To insert a Vehicle, just click on the beginning of the route. A car will appear. On the left side bar you can change the vehicle's attributes such as id and even the color (just for fun change it to blue).



Now save the Demand (route + vehicle) file . Use *File* → *Demand elements* → *Save demand elements* (Ctrl + Shift + D) or *File* → *Demand elements* → *Save demand elements as* and give it a proper name (such as *"helloWorld.rou.xml"*).

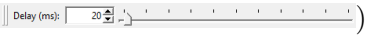
Do not close *netedit* yet.

A.3 Visualizing in sumo-gui

We will open *"sumo-gui"* from *netedit*. To do so, go to *Edit* → *Open in sumo-gui* (Ctrl + T). This will open *sumo-gui* and load our recently created network and demand files.

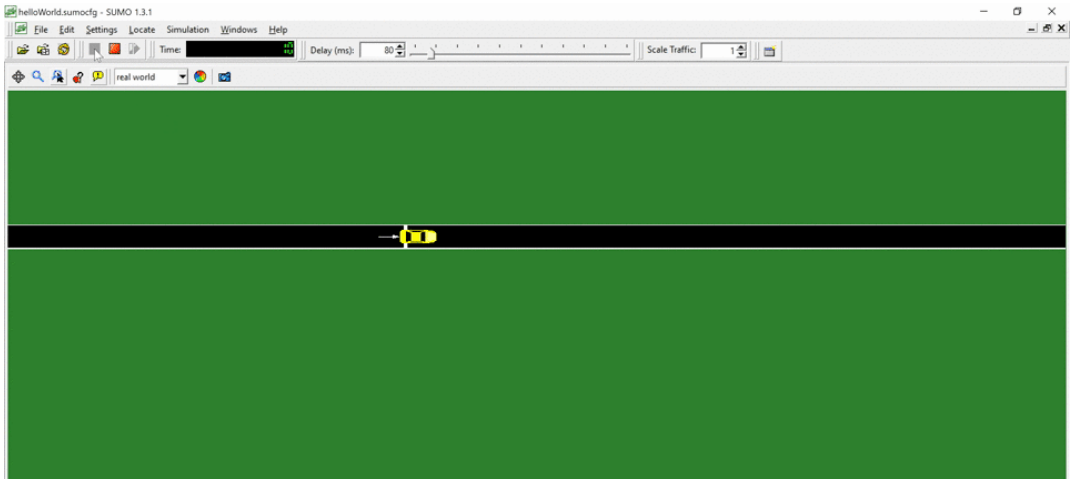
As soon as *sumo-gui* opens, let's save the *SUMO* configuration file (that relates the network and demand files) . *File* → *Save Configuration* (Ctrl + Shift + S). Give it a proper name (such as *"helloWorld.sumocfg"*).

Now you can close *netedit* if you wish.

Before starting the simulation, make sure that the Delay (

) is set to at least *80 ms*, otherwise, the simulation would happen very fast and we would not be able to see our only vehicle in our tiny network.

Click on Run  (Ctrl + A) to start the simulation.



From now on, if we want to run this scenario again we only have to open the SUMO Configuration file *(*.sumocfg)* in *sumo-gui* or *sumo*.

That's it! You have your first simulation scenario in *SUMO* :)

Appendix B

Interfacing TraCI from Python

The details below were extracted from the official *"SUMO"* website: https:\\sumo.dlr.de\docs\TraCI\Interfacing_TraCI_from_Python.html [62].

TraCI is the short term for "**Tra**ffic **C**ontrol **I**nterface". Giving access to a running road traffic simulation, it allows to retrieve values of simulated objects and to manipulate their behavior "on-line". The TraCI commands are split into the 13 domains gui, lane, poi, simulation, trafficlight, vehicletype, edge, inductionloop, junction, multientryexit, polygon, route, person and vehicle, which correspond to individual modules [62].

B.1 importing traci in a script

To use the library, the <**SUMO_HOME**>/tools directory must be on the python load path. This is typically done with a stanza like this:

```
import os, sys
if 'SUMO_HOME' in os.environ:
    tools = os.path.join(os.environ['SUMO_HOME'], 'tools')
    sys.path.append(tools)
else:
    sys.exit("please declare environment variable 'SUMO_HOME'")
```

This assumes that the environment variable **SUMO_HOME** is set before running the script. Alternatively, you can declare the path to *sumo/tools* directly as in the line

```
sys.path.append(os.path.join('c:', os.sep, 'whatever', 'path', 'to', 'sumo', 'tools'))
```

B.2 First Steps

In general it is very easy to interface with SUMO from Python:

First you compose the command line to start either **sumo** or **sumo-gui** (leaving out the option which was needed before 0.28.0):

```
sumoBinary = "/path/to/sumo-gui"
sumoCmd = [sumoBinary, "-c", "yourConfiguration.sumocfg"]
```

Then you start the simulation and connect to it with your script:

```
import traci
traci.start(sumoCmd)
step = 0
while step < 1000:
   traci.simulationStep()
   if traci.inductionloop.getLastStepVehicleNumber("0") > 0:
       traci.trafficlight.setRedYellowGreenState("0", "GrGr")
   step += 1

traci.close()
```

After connecting to the simulation, you can emit various commands and execute simulation steps until you want to finish by closing the connection. by default, the close command will wait until the sumo process finishes. You can disable this by calling

```
traci.close(False)
```

B.3 Subscriptions

Subscriptions can be thought of as a batch mode for retrieving variables. Instead of asking for the same variables over and over again, you can retrieve the values of interest automatically after each time step. TraCI subscriptions are handled on a per-module basis. That is you can ask the module for the result of all current subscriptions after each time step. To subscribe for variables you need to know their variable ids which can be looked up in the traci/constants.py file.
The values retrieved are always the ones from the last time step, it is not possible to retrieve older values.

```
import traci
import traci.constants as tc

traci.start(["sumo", "-c", "my.sumocfg"])
traci.vehicle.subscribe(vehID, (tc.VAR_ROAD_ID, tc.VAR_LANEPOSITION))
print(traci.vehicle.getSubscriptionResults(vehID))
for step in range(3):
   print("step", step)
   traci.simulationStep()
   print(traci.vehicle.getSubscriptionResults(vehID))
traci.close()
```

B.4 Context Subscriptions

Context subscriptions work like subscriptions in that they retrieve a list of variables automatically for every simulation stop. However, the do so by setting a reference object and a range and then retrieving variables for all objects of a given type within range of the reference object.

TraCI context subscriptions are handled on a per module basis. That is you can ask the module for the result of all current subscriptions after each time step. In order to subscribe for variables you need to the domain id of the objects that shall be retrieved and the variable ids which can be looked up in the traci/constants.py file. The domain id always has the form CMD_GET_<DOMAIN>_VARIABLE. The following code retrieves all vehicle speeds and waiting times within range (42m) of a junction (the vehicle ids are retrieved implicitly).

```
import traci
import traci.constants as tc

traci.start(["sumo", "-c", "my.sumocfg"])
traci.junction.subscribeContext(junctionID, tc.CMD_GET_VEHICLE_VARIABLE, 42, [tc.VAR_SPEED, tc.VAR_WAITING_TIME])
print(traci.junction.getContextSubscriptionResults(junctionID))
for step in range(3):
   print("step", step)
   traci.simulationStep()
   print(traci.junction.getContextSubscriptionResults(junctionID))
traci.close()
```

The values retrieved are always the ones from the last time step, it is not possible to retrieve older values.

B.5 Controlling parallel simulations from the same TraCI script

The TraCI python library can be used to control multiple simulations at the same time with a single script. The function *traci.start()* has an optional label argument which allows you to call it multiple times with different simulation instances and labels. The function *traci.switch()* can then be used to switch to any of the initialized labels:
If you prefer a more object oriented approach you can also use connection objects to

```
traci.start(["sumo", "-c", "sim1.sumocfg"], label="sim1")
traci.start(["sumo", "-c", "sim2.sumocfg"], label="sim2")
traci.switch("sim1")
traci.simulationStep() # run 1 step for sim1
traci.switch("sim2")
traci.simulationStep() # run 1 step for sim2
```

communicate with the simulation. They have the same interface as the static traci. calls but you will still need to start the simulation manually for them:

```
traci.start(["sumo", "-c", "sim1.sumocfg"], label="sim1")
traci.start(["sumo", "-c", "sim2.sumocfg"], label="sim2")
conn1 = traci.getConnection("sim1")
conn2 = traci.getConnection("sim2")
conn1.simulationStep() # run 1 step for sim1
conn2.simulationStep() # run 1 step for sim2
```

B.6 Controlling the same simulation from multiple clients

To connect with multiple clients, the number of clients must be known in advance and specified with sumo option –**num-clients** ¡*INT*¿. Also, the connection port must be known to all clients. After deciding on a port it can be made available to the clients via arguments or configuration files. A free port can be obtained by

```
from sumolib.miscutils import getFreeSocketPort
port = sumolib.miscutils.getFreeSocketPort()
```

One client may use method traci.start() to start the simulation and connect to it at the same time while the other client only needs to connect. After establishing client order, each client must continuously call simulationStep to allow the simulation to advance:

```
#client1
# PORT = int(sys.argv[1]) # example
traci.start(["sumo", "-c", "sim.sumocfg", "--num-clients", "2"], port=PORT)
traci.setOrder(1) # number can be anything
while traci.simulation.getMinExpectedNumber() > 0:
   traci.simulationStep()
   # more traci commands
traci.close(
```

```
# client2
# PORT = int(sys.argv[1]) # example
traci.init(PORT)
traci.setOrder(2) # number can be anything as long as each client gets its own number
while traci.simulation.getMinExpectedNumber() > 0:
   traci.simulationStep()
   # more traci commands
traci.close()
```

Bibliography

[1] Guchan Ozbilgin, Umit Ozguner, Onur Altintas, Haris Kremo, and John Maroli. Evaluating the requirements of communicating vehicles in collaborative automated driving. *IEEE Intelligent Vehicles Symposium (IV): pp. 1066-1071*, June 2016.

[2] Jingrong Chen, Sheng Tian, Hao Xu, Rui Yue, Yuan Sun, and Yuepeng Cui. Architecture of vehicle trajectories extraction with roadside lidar serving connected vehicles. *IEEE Access, vol. 7, pp. 100406-100415*, 2019.

[3] Hiroyuki Okuda, Kota Harada, Tatsuya Suzuki, Shintaro Saigo, and Satoshi Inoue. Modeling and analysis of acceptability for merging vehicle at highway junction. *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016.

[4] Chiyu Dong, John M. Dolan, and Bakhtiar Litkouhi. Intention estimation for ramp merging control in autonomous driving. *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017.

[5] Tien Dung Chu, Hideki Nakamura, and Miho Asano. Modeling gap choice behavior at urban expressway merging sections. *Journal of Japan Society of Civil Engineers, Ser. D3 (Infrastructure Planning and Management), Vol. 72, No.2, pp. I-881–I891*, 2016.

[6] Yi Hou, Praveen Edara, and Carlos Sun. Modeling mandatory lane changing using bayes classifier and decision trees. *IEEE Transactions on Intelligent Transportation Systems, Vol. 15, No. 2*, 2014.

[7] Alexandra Kondyli. Driver behavior at freeway-ramp merging areas. *Transportation Research Record Journal of the Transportation Research Board*, 2010.

[8] Majid Sarvi and Masao Kuwahara. Microsimulation of freeway ramp merging processes under congested traffic conditions. *IEEE Transactions on Intelligent Transportation Systems, Vol. 8, No. 3*, 2007.

[9] Yangliu Dou, Yihao Fang, Chuan Hu, Rong Zheng, and Fengjun Yan. Gated branch neural network for mandatory lane changing suggestion at the on-ramps of highway. *IET Intelligent Transport Systems, Vol. 13, No. 1*, 2019.

[10] Dan Marinescu, Jan Čurn, Mélanie Bouroche, and Vinny Cahill. On-ramp traffic merging using cooperative intelligent vehicles: A slot-based approach. *Intelligent Transportation Systems (ITSC)*, 2012.

[11] Arianne Mizuta, Kim Roberts, Les Jacobsen, and Nick Thompson. Ramp metering: A proven, cost-effective operational strategy - a primer. Technical report, U.S. Dept. Transp. Federal Highway Admin., 2014.

[12] Jackeline Rios Torres and Andreas A. Malikopoulos. Automated and cooperative vehicle merging at highway on-ramps. *IEEE Transactions on Intelligent Transportation Systems, vol 18, pp. 780-789*, 2017.

[13] Hiroyuki Okuda, Kota Harada, Tatsuya Suzuki, Shintaro Saigo, and Satoshi Inoue. Design of automated merging control by minimizing decision entropy of drivers on main lane. *IEEE Intelligent Vehicles Symposium (IV), pp. 640-646*, 2017.

[14] Pin Wang and Ching-Yao Chan. Autonomous ramp merge maneuver based on reinforcement learning with continuous action space. *ArXiv 2018 - Computer Science*, 2018.

[15] Tomoki Nishi, Prashant Doshi, and Danil Prokhorov. Merging in congested freeway traffic using multipolicy decision making and passive actor-critic learning, vol. 4, pp. 287-297. *IEEE Transactions on Intelligent Vehicles*, 2019.

[16] Maxime Bouton, Alireza Nakhaei, Kikuo Fujimura, and Mykel J. Kochenderfer. Cooperation-aware reinforcement learning for merging in dense traffic. *IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2019.

[17] Yuan Lin, John McPhee, and Nasser L. Azad. Decision-making and control for freeway on-ramp merging using deep reinforcement learning. *arXiv preprint arXiv:1909.12967*, 2019.

[18] Stephanie Lefevre, Dizan Vasquez, and Christian Laugier. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH Journal 1(1): 1-15*, 2014.

[19] Inc Cambridge Systematics. Ngsim i-80 data analysis (4:00 p.m. to 4:15 p.m.). Technical report, Federal Highway Administration, 2005.

[20] Hongbign Wang, Xin Chen, Qin Wu, Qi Yu, Xingguo Hu, Zibin Zheng, and Athman Bouguettaya. Integrating reinforcement learning with multi-agent techniques for adaptive service composition. *ACM Transactions on Autonomous and Adaptive Systems 12(2):1-42*, 2017.

[21] Zhangjing Wang, Yu Wu, and Qingqing Niu. Multi-sensor fusion in automated driving: A survey. *IEEE Access : vol. 8, pp. 2847-2868*, December 2019.

[22] David Elliott, Walter Keen, and Lei Miao. Recent advances in connected and automated vehicles. *Journal of Traffic and Transportation Engineering : vol. 6, pp. 109-131*, April 2019.

[23] Klaus Bengler, Klaus Dietmayer, Berthold Farber, Markus Maurer, Christoph Stiller, and Hermann Winner. Three decades of driver assistance systems: Review and future perspectives. *IEEE Intelligent Transportation Systems Magazine : vol. 6, pp. 6-22*, October 2014.

[24] Lv Bin, Xu Hao, Wu Jianqing, Tian Yuan, Tian Sheng, and Suoyao Feng. Revolution and rotation-based method for roadside lidar data integration. *Optics and Laser Technology, Vol. 119, article id. 105571*, 2019.

[25] Jianqing Wu, Yuan Tian, Hao Xu, Rui Yue, Aobo Wang, and Xiuguang Song. Automatic ground points filtering of roadside lidar data using a channel-based filtering algorithm. *Optics and Laser Technology, Vol. 115, pp. 374-383*, 2019.

[26] Jingrong Chen, Hao Xu, Jianqing Wu, Rui Yue, Changwei Yuan, and Lu Wang. Deer crossing road detection with roadside lidar sensor. *IEEE Access, vol. 7, no. 1, pp. 65944-65954*, 2019.

[27] Jianqing Wu, Hao Xu, and Jianying Zheng. Automatic background filtering and lane identification with roadside lidar data. *Intelligent Transportation Systems (ITSC)*, 2017.

[28] Florian Geissler, Sören Kohnert, and Reinhard Stolle. Designing a roadside sensor infrastructure to support automated driving. *Intelligent Transportation Systems (ITSC)*, 2018.

[29] Gurulingesh Raravi, Vipul Shingde, Krithi Ramamritham, and Jatin Bharadia. Merge algorithms for intelligent vehicles. *Next Generation Design and Verification Methodologies for Distributed Embedded Control Systems, Proceedings of the GM RD Workshop, Springer, pp. 51–65*, 2007.

[30] National Highway Traffic Safety Administration. National motor vehicle crash causation survey. Technical report, U.S. Department of Transportation, 2008.

[31] https://www.thetruthaboutcars.com/2018/05/thwarted-ramp-waymo-driverless-car-doesnt-feel-urge-merge.

[32] Florian Geissler, Sören Kohnert, and Reinhard Stolle. Designing a roadside sensor infrastructure to support automated driving. *Intelligent Transportation Systems (ITSC)*, 2018.

[33] David Elliott, Walter Keen, and Lei Miao. Recent advances in connected and automated vehicles. *Journal of Traffic and Transportation Engineering, vol. 6, pp. 109-131*, 2019.

[34] Ji Fang, Ruichen Xu, Yuan Yang, Xiaofan Li, Sha Zhang, Xiao Peng, and Xiaoyong Liu. Introduction and simulation of dedicated short range communication. *International Symposium on Electromagnetic Compatibility*, 2017.

[35] Bryan Thomas. U.s. dot advances deployment of connected vehicle technology to prevent hundreds of thousands of crashes. Technical report, U.S National Highway Traffic Safety Administration (NHTSA), 2016.

[36] Cellular v2x communications towards 5g. Technical report, 5G Americas, 2018.

[37] ETSI. Intelligent transport systems (its); vehicular communications; basic set of applications; analysis of the collective perception service (cps). Technical report, Intelligent Transport Systems (ITS), 2019.

[38] Valerian Mannoni, Vincent Berg, Stefania Sesia, and Eric Perraud. A comparison of the v2x communication systems: Its-g5 and c-v2x. *IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, 2019.

[39] Alexandre Armand. *Situation understanding and risk assessment framework for preventive driver assistance*. PhD thesis, University of Paris-Saclay, 2016.

[40] ETSI. Vehicular communications : Basic set of applications. part 2: Specification of cooperative awareness basic service. Technical report, Intelligent Transport Systems (ITS), 2014.

[41] Florian Marczaka, Winnie Daamenb, and Christine Buissona. Key variables of merging behaviour: empirical comparison between two sites and assessment of gap acceptance theory. *20th International Symposium on Transportation and Traffic Theory (ISTTT 2013)*, 2013.

[42] Yuanchang Xie, Huixing Zhang, Nathan Gartner, and Tugba Arsava. Collaborative merging behaviors and their impacts on freeway ramp operations under connected vehicle environment. *Symposium Celebrating 50 Years of Traffic Flow Theory, pp. 392-408*, 2014.

[43] Andrew Y. Ng and Michael I. Jordan. Discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in Neural Information Processing Systems, vol. 14, pp. 841-848*, 2001.

[44] D. W. Hosmer and S. Lemeshow. *Applied Logistic Regression. Wiley series in Probability and Statistics*. 2005.

[45] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning, vol. 29, pp. 131–163*, 1997.

[46] Dahee Chung, Kun Chang Lee, and Seung Chang Seong. General bayesian network approach to health informatics prediction: Emphasis on performance comparison. *Procedia - Social and Behavioral Sciences 81:465-468*, 2013.

[47] Ni'am Fuad, Mohd Nasir Taib, Rozita Jailani, and Muhammad Marwan. Brainwave classification for brain balancing index (bbi) via 3d eeg model using k-nn technique. *International Journal of Electrical and Computer Engineering*, 2014.

[48] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon 4*, 2018.

[49] B. Goodman and S. Flaxman. European union regulations on algorithmic decision-making and a" right to explanation. *arXiv preprint arXiv:1606.08813*, 2016.

[50] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2016.

[51] Ioffe, Sergey, Szegedy, and Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[52] G. E. Uhlenbeck and L. S. Ornstein. On the theory of the brownian motion. *Physical review, vol. 36, pp. 823*, 1930.

[53] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *International Conference on Machine Learning (ICML)*, 2018.

[54] Prashant Shridhar Bokare and Akhilesh Kumar Maurya. Acceleration-deceleration behaviour of various vehicle types. *Transportation Research Procedia, vol. 25, pp. 4737-4753*, 2017.

[55] Nishant Kheterpal, Kanaad Parvate, Cathy Wu, Aboudy Kreidieh, Eugene Vinitsky, and Alexandre M Bayen. Flow: Deep reinforcement learning for control in sumo. *EPiC Series in Engineering, vol. 2, pp. 134-151*, 2018.

[56] S. Krauss, P. Wagner, and C. Gawron. Metastable states in a microscopic model of traffic flow. *PHYSICAL REVIEW E, vol. 55, p. 5597*, 1997.

[57] Javier Garcia and Fernando Fernandez. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research, vol. 16, pp. 1437-1480*, 2015.

[58] Javier Garcia and Fernando Fernandez. Safe exploration of state and action spaces in reinforcement learning. *Journal of Artificial Intelligence Research, vol. 45, pp. 515–564*, 2012.

[59] Subramanya Nageshrao, Eric Tseng, and Dimitar Filev. Autonomous highway driving using deep reinforcement learning. *International Conference on Systems, Man and Cybernetics (SMC)*, 2019.

[60] AASHTO. A policy on geometric design of highways and streets. Technical report, The American Association of State Highway and Transportation Officials, 2011.

[61] German Aerospace Center (DLR) et al. Tutorials/Hello World. https://sumo.dlr.de/docs/Tutorials/Hello_World.html, 2020.

[62] German Aerospace Center (DLR) et al. TraCI/Interfacing TraCI from Python. https://sumo.dlr.de/docs/TraCI/Interfacing_TraCI_from_Python.html, 2020.