Vehicle-in-Virtual-Environment (VVE) Method for Developing and Evaluating VRU Safety of Connected and Autonomous Driving

Haochong Chen (https://orcid.org/0009-0000-5461-0822)
Xincheng Cao (https://orcid.org/0009-0008-2525-9031)
Bilin Aksun-Guvenc (https://orcid.org/0000-0003-0836-9286)
Levent Guvenc (https://orcid.org/0000-0001-8823-1820)

FINAL RESEARCH REPORT – July 31, 2024

Contract # 69A3552344811

Contents

Chapter 1: Overview **3**

Chapter 2: Vulnerable Road User (VRU) Detection **6**

2.1 Camera Based Vulnerable Road User (VRU) Detection 6

2.2 Lidar Data Process and Vulnerable Road User (VRU) Trajectory Prediction 9

2.3 Conclusion 11

Chapter 3: Disturbance Observer (DOB) and PID based Path Tracking **12**

3.1 Introduction 12

3.2 Path Generation 12

3.3 Linear Path-tracking Model 13

3.4 Disturbance Observer 15

3.5 Simulation Study 19

3.6 Conclusions 24

Chapter 4: Deep Reinforcement Learning (DRL) Based Collision Avoidance **25**

4.1 Introduction 25

4.2 Methodology 27

4.2.1 Vehicle Model 27

4.2.2 Low-level PID Pure Pursuit Path Tracking Controller Design 31

4.2.3 High-level DRL Based Collision Avoidance Controller Design 32

4.3 Results 38

4.3.1 Test Case 1 39

4.3.2 Test Case 2 41

4.4 Conclusions 44

Chapter 5: Vehicle-in-Virtual-Environment (VVE) 46

5.1 Introduction 46

5.2 Methodology 47

5.2.1 VVE System Structure 47

5.2.2 Frame Transformation/Conversion for Motion Synchronization 49

5.2.3 V2P System Structure 53

5.3 Results 55

5.4 Conclusions 57

Chapter 6: Future Work 59

References 61

Chapter 1

Overview

With rapid urbanization and technology development, the number of privately owned vehicles has dramatically increased each year. The excessive numbers of private vehicles have led to traffic congestion and car accidents, which gradually become a new set of challenges that every modern city must confront. According to the Global Status Report on Road Safety released by the World Health Organization (W.H.O.), over 50 million people get injuries, and 1.3 million individuals lose their lives in car accidents worldwide each year [1]. In the United States alone, over 2.3 million people are injured, and around 40,000 lives are lost in car accidents [2]. Among these car accidents, around 75% are attributed to human errors, such as drowsy driving, driving under the influence (DUI), and distracted driving. The use of an Automated Driving System (ADS) that benefits from powerful and robust autonomous driving algorithms may significantly reduce car accidents caused by human mistakes, thereby becoming a potential solution to these urgent traffic challenges. The Society of Automotive Engineers (SAE International) has categorized autonomous vehicles into six levels, ranging from Level 0 (fully manual driving) to Level 5 (fully autonomous driving). Particularly, vehicles at SAE Levels 4 and 5 have the capability to dramatically decrease accidents caused by human mistakes since their automated driving algorithms should have robust and steady performance under all traffic condition. In order to increase the level of autonomy of road vehicles, extensive research and testing have been conducted in the field in recent years. However, alongside these developments, new challenges and potential issues have also emerged especially related to rare and extreme events that usually end in collisions or near collisions.

The current approach to testing and evaluating connected and autonomous driving algorithms typically relies on model-in-the-loop (MIL) and hardware-in-the-loop (HIL) simulations, where the effectiveness of these methods is highly dependent on the accuracy of the simulated vehicle models that are used. Following these simulations, there is limited use of proving grounds, which is then followed by public road deployment of beta versions of the software and technology. This final public road development process forces other road

users to involuntarily participate in the development and evaluation of these beta-level autonomous driving functions. This approach is unsafe, costly, and inefficient, leading to numerous issues during the deployment of autonomous vehicles and contributes to a significant loss of public trust.

To address the aforementioned challenges, the Vehicle-in-Virtual-Environment (VVE) method is proposed as a safe, efficient, and cost-effective solution for the development, evaluation, and demonstration of connected and autonomous driving functions. The VVE method places the vehicle in a highly realistic virtual environment with accurate virtual sensor feeds, while the actual vehicle operates in a large, empty test area. VVE method can synchronize real vehicle motions in the real world and virtual vehicle motions in a virtual environment, to allow for the creation of various virtual traffic scenarios for safe and resource-efficient testing.

In this two-year project, we are focused on applying the Vehicle-in-Virtual-Environment (VVE) method to develop, evaluate, and demonstrate safety functions for Vulnerable Road Users (VRUs). In the first year, our primary focus was on pedestrian safety and the results are presented in this final report. We analyzed five key pedestrian crash scenarios identified by the Fatality Analysis Reporting System (FARS), an organization under the National Highway Traffic Safety Administration (NHTSA) that compiles vehicle crash data. The pedestrian crash scenarios we examined include: "Crossing Roadway, Vehicle Not Turning" (FARS 750), "Walking/Running Along Roadway" (FARS 400), "Dash/Dart-Out" (FARS 740), "Crossing Roadway, Vehicle Turning" (FARS 790), and "Crossing Expressway" (FARS 910). These scenarios formed the basis of our research during the first year. We recreated these five traffic crash scenarios using CARLA (Car Learning to Act) virtual environment to ensure realistic and accurate simulations for our analysis. The detailed traffic crash scenarios simulation videos are provided next. Videos Link: 1. FARS400: https://youtu.be/3kpJUbqACxg 2. FARS740: https://youtu.be/3r9mYsyRZA8 3. FARS750: https://youtu.be/5RVWsK6CnLw 4. FARS790: https://youtu.be/F0ZQcdZlh-4 5. FARS910: https://youtu.be/cb-_dqDf1oM.

Therefore, we designed a comprehensive autonomous driving solution to ensure pedestrian safety. In Chapter 2, we explore an RGB camera and LiDAR-based perception method to identify, locate, and predict the trajectory of pedestrians in traffic environments. Chapter 3 demonstrates a disturbance observer integrated with a PID path tracking controller which provides precise and accurate path tracking for the collision free path. Following this, Chapter 4 introduces a deep reinforcement learning (DRL) based approach for path planning and collision avoidance. In Chapter 5, we discuss the principles of the Vehicle-in-Virtual-Environment (VVE) method and detail its implementation for testing and evaluating the ADAS system proposed in the previous chapters. Finally, in Chapter 6, we conclude our first-year research and outline future work.

The publications that summarize some of our research findings with project support acknowledgment are:

1. H. Chen and B. A. Guvenc, "Deep Reinforcement Learning Based Collision Avoidance of Automated Driving Agent," SAE International, Warrendale, PA, SAE Technical Paper 2024-01–2556, Apr. 2024. [Online]. Available: https://www.sae.org/publications/technical-papers/content/2024-01-2556/
2. X. Cao, H. Chen, S. Y. Gelbal, B. A. Guvenc, and L. Guvenc, "Vehicle-in-Virtual-Environment Method for ADAS and Connected and Automated Driving Function Development, Demonstration and Evaluation," SAE International, Warrendale, PA, SAE Technical Paper 2024-01–1967, Apr. 2024. [Online]. Available: https://www.sae.org/publications/technical-papers/content/2024-01-1967/
3. H. Chen, X. Cao, L. Guvenc, and B. Aksun-Guvenc, "Deep-Reinforcement-Learning-Based Collision Avoidance of Autonomous Driving System for Vulnerable Road User Safety," Electronics, vol. 13, no. 10, Art. no. 10, Jan. 2024, doi: 10.3390/electronics13101952. [Online]. Available: https://www.mdpi.com/2079-9292/13/10/1952

Chapter 2

Vulnerable Road User Detection

2.1 Camera Based Vulnerable Road User (VRU) Detection

In this section, the use of a computer vision based pedestrian perception strategy for VVE method to test autonomous driving functions is investigated using readily available and well-established approaches. Plenty of research has already done in the pedestrian detection field. The first approach is to perform object detection using a two-stage process: proposal generation and object detection. The proposal generation step uses a selective search algorithm to generate multiple region proposals which indicate potential object locations. Then, a neural network is applied to classify the object within the proposed region and refine its bounding box. As an example of this approach, Girshick et al. introduced R-CNN, a novel framework that leverages rich feature hierarchies from pre-trained convolutional neural networks to perform accurate object detection and semantic segmentation [3]. Building on the foundation of R-CNN, Girshick further refined the model with Fast R-CNN by integrating the region proposal and feature extraction steps. This integration was achieved by introducing a Region of Interest (RoI) pooling layer that extracts a fixed-length feature vector from the feature map for each object proposal, followed by fully connected layers that classify the features into object categories and regress the bounding box coordinates [4]. Ren et al. explored the concept further and proposed Faster R-CNN. This algorithm incorporated a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals [5]. Sun et al. introduced Sparse R-CNN which simplifies the previously complex pipeline and reduces the dependency on heuristic design, pushing the boundaries of object detection with a sparse set of highly effective proposals [6]. However, the major limitations of the R-CNN architecture are its deficiency in real-time performance caused by complicated procedure and computational complexity.

The second approach is to merge the aforementioned two stages by applying a single

neural network to the whole image, dividing the image into regions and predicting bounding boxes and probabilities for each region simultaneously. Redmon introduced a convolutional neural network (CNN)-based architecture known as "YOLO" (You Only Look Once) for multi-object detection. This architecture segments an image into multiple small grids, assigning each grid the task of detecting objects whose center points fall within its boundaries. For each grid cell, the model predicts multiple bounding boxes and assigns labels to these boxes, each with associated class probabilities. To enhance the accuracy and reduce redundancy, the model employs Non-Maximum Suppression (NMS) to eliminate overlapping bounding boxes that detect the same object [7]. Currently, since its debut, the YOLO architecture has undergone numerous iterations and enhancements, evolving to its eighth generation—YOLOv8. These iterations have not only improved the model's accuracy and speed but also enabled YOLO to effortlessly recognize a wide variety of objects [8]. Additionally, there are many pre-trained YOLO models available for us to choose from. Therefore, in this chapter, we propose a training pipeline that utilizes a pre-trained, high-performance YOLO model and applies transfer learning to our dataset. One of the challenges in performing pedestrian detection is that the VVE method conducts traffic tests in a virtual environment, meaning all sensors are virtual. Therefore, unlike traditional pedestrian detection tasks that utilize real traffic images, our RGB camera is limited to process only virtual images. To address this challenge, the task is separated into two stages: 1. Find powerful pre-trained YOLO model. 2. Build CARLA VRU detection dataset and perform transfer learning on the dataset.

The YOLO architecture we implemented for pedestrian detection is YOLO v8, which can identify pedestrians and bicyclists [9]. However, the original model sometimes fails to accurately recognize vulnerable road users (VRUs) and other vehicles in the virtual environment. Therefore, we retrained the pre-trained the model using the CARLA Object Detection Dataset to enhance its performance. The specific datasets used for training included versions 20, 18, and 16 from the CARLA Object Detection Dataset [10]. The training parameters were set to 20 epochs, approximately 7 hours of training time, with an image size of 640, while other parameters remained at their default settings. After transfer learning, we observed a significant improvement in performance, achieving a processing speed of 130 ms per frame, which meets real-time performance criteria.

Figures 2.1 and 2.2 clearly demonstrate that the model, after undergoing transfer learning, can accurately detect vehicles and multiple vulnerable road users (VRUs), including pedestrians and bicyclists. This enhanced ADAS's detection capability by letting the system accurately interpret traffic information. Additionally, by integrating Lidar data, the precise locations of each detected VRU can be calculated, enabling the prediction of pedestrian trajectories. This integration of Lidar further enhances the system's ability to anticipate and respond to potential hazards on the road. Lidar data processing technology will be discussed in the next section. The demo video link is: https://youtu.be/N4Q_0O1uVuA.

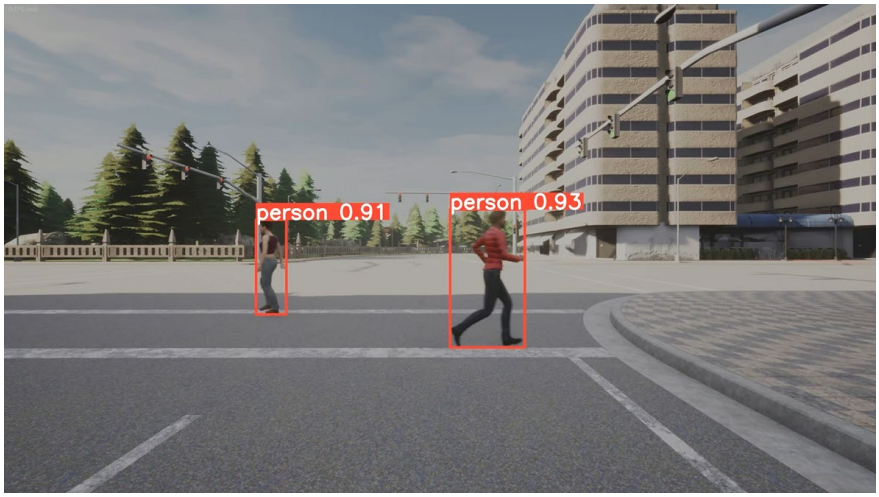


Figure 2.1: Yolo-v8 pedestrian detection. The first pedestrian is walking to the left while the second pedestrian is running to the right. Both pedestrians are crossing the street near a T-shaped intersection using the crosswalk.

Figure 2.2: Yolo-v8 bicyclist and vehicle detection. There is a bicycle followed by a vehicle. The bicyclist is also identified as a person. Both the bicyclist and vehicle are moving along the cross street in a T-shaped intersection.

2.2 Lidar Data Process and Vulnerable Road User (VRU) Trajectory Prediction

Lidar technology offers significant advantages over traditional front-facing vehicle cameras, particularly in the field of autonomous vehicles. Unlike cameras, which are limited to capturing visual data from the front of the vehicle, a three-dimensional Lidar provides a 360-degree view, allowing it to detect obstacles all around the vehicle. This comprehensive coverage is important for the complex decision making required in autonomous driving. Lidar sensors work by emitting laser beams and measuring the time it takes for the reflection to return, thereby creating detailed and accurate three-dimensional maps of the environment. This capability makes Lidar exceptionally good at detecting and tracking vulnerable road users, even in challenging conditions such as low light or obstructed views. Consequently, there has been substantial research and development in the field, focusing on leveraging Lidar for vulnerable road user detection, which is important for improving safety and operational efficiency in autonomous vehicle technologies.

Muhammad et al. focused on enhancing object detection in autonomous driving through a neural network approach that integrates visual data with Lidar point clouds. They proposed a framework aiming to address the inaccuracies common in Lidar detections by

using separate processing streams for visual and Lidar data, which allows for a lightweight Lidar-only setup during runtime, if needed. The approach they used is designed to work in real-time on embedded platforms, suggesting significant potential for practical applications in dynamic environments [11]. Sahba et al. introduced an effective method for three-dimensional object detection using Lidar data through the PointPillars network. Their study utilizes the nuScenes dataset to train the model for detecting cars, pedestrians, and buses, demonstrating that increasing the number of Lidar sweeps substantially improves detection performance. Their research emphasizes the potential of integrating different types of sensor data to further enhance the encoder's effectiveness in autonomous vehicle applications [12]. Liu et al. developed a Lidar-camera fusion algorithm for three-dimensional object detection, focusing on autonomous driving applications. Their proposed FuDNN network used a two-dimensional backbone for image feature extraction and an attention-based fusion sub-network for integrating features from camera and Lidar data. Their model was tested on the KITTI dataset and has shown high accuracy in detecting cars, reflecting significant improvements over existing Lidar-camera fusion techniques [13]. Naich et al. introduced a Lidar-based intensity-aware three-dimensional object detection approach for outdoor environments. They proposed a voxel encoder that generates intensity histograms to enhance the feature set for robust detection, integrated within a single-stage detector. Their method was evaluated using the KITTI dataset which not only matched but in some cases surpassed state-of-the-art performance, especially in detecting pedestrians and cyclists, while maintaining high frame rates during inference [14]. Zhu et al. introduced the Spatio-Temporal Graph Transformer Network (STGFNet) for predicting multi-pedestrian trajectories, leveraging both spatial and temporal data. Their proposed model integrated a novel decoder structure and a memory mechanism to enhance trajectory continuity and used HuberLoss for the first time as a loss function, showing notable improvements in prediction accuracy across multiple datasets. This research exemplified the usefulness of combining transformer architectures with graph neural networks to address the dynamic complexities of pedestrian movement in crowded spaces [15].

2.3 Conclusion

In this chapter, we explored various advanced methodologies for detecting VRUs using camera-based and Lidar technologies for autonomous driving. Our literature review indicates that YOLOv8 is not only efficient for real-time pedestrian and bicyclist detection but also straightforward to implement. By leveraging pre-trained models and applying transfer learning, our model has demonstrated the capability to accurately identify various VRUs in a simulated environment. Additionally, we have found that Lidar technology, in comparison to cameras, provides superior accuracy in locating the positions of road users due to its ability to generate precise three-dimensional maps of the surrounding environment. Therefore, we conducted an extensive review of the latest studies in Lidar technology and pedestrian trajectory prediction, which will undoubtedly enhance our foundation for future research in enhancing the safety and efficiency of autonomous driving systems. The vehicle-in-virtual-environment development and evaluation method of our research uses sensor feeds from the virtual environment of vehicle and vulnerable road user interactions of high collision risk.

Chapter 3

Disturbance Observer (DOB) and PID based Path Tracking

3.1 Introduction

In the previous chapter, the process of detecting vulnerable road users was described. This chapter focuses on path tracking for following a collision free path in the presence of vulnerable road users.

3.2 Path Generation

The collision avoidance maneuver in this case is assumed to be a single lane-change. The overall procedure of obtaining such a reference path can be condensed to the following: (a) generate limited number of sample waypoints to ascertain the general shape of the path; (b) generate dense waypoints based on the sample waypoints to complete the path waypoint design; (c) apply segmentation to the dense waypoints to cut the path into several segments, ideally with each segment containing a minimum amount of features (i.e. corners); and (d) perform polynomial fit optimization to obtain the desired path expression that guarantees smooth curvature within each segment as well as smooth transition between the segments. The detailed steps of this procedure are outlined in reference [16]. In Figure 3.1, an example of the optimized reference path and its path curvature are displayed to demonstrate the smoothness of such a path generated using this approach.

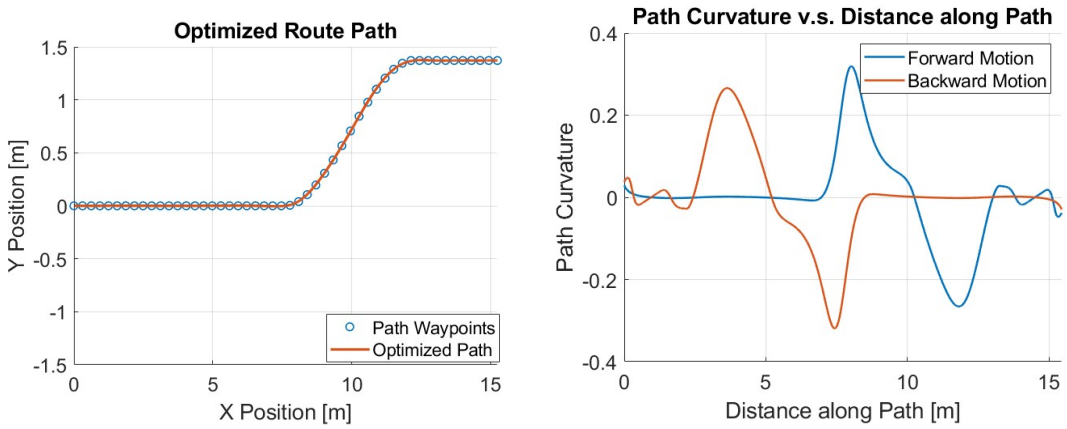


Figure 3.1: Reference path optimization: (a) optimized path; (b) path curvature of the optimized path [16].

3.3 Linear Path-tracking Model

This section presents the linear path-tracking model that serves as the basis for the proposed control routine. The detailed derivation of this model can be found in [17]. This linear path-tracking model contains two components: a (linear) lateral single-track model and a path-tracking model augmentation. The path-tracking scenario is illustrated in Figure 3.2, and the resulting linear path-tracking model is described in Equation (3.1). The parameters of this model are specified in Table 3.1. It can be observed that, for generality, the model presented in Equation (3.1) has both front and rear steering angles δ_f and δ_r as inputs. In our case, the vehicle is assumed to be front-wheel-steer only. It can also be noticed that path curvature ρ_{ref} and yaw moment disturbance M_{zd} enter the model as external disturbances. Additionally, the preview distance l_s is chosen to be a linear function of vehicle velocity. It should also be remarked that vehicle speed can be scheduled according to the refence path curvature to make sure vehicle lateral acceleration stays within an acceptable limit.

$$\begin{bmatrix} \dot{\beta} \\ \dot{r} \\ \Delta\dot{\psi}_p \\ \dot{e_y} \end{bmatrix} = \begin{bmatrix} \frac{-C_f-C_r}{MV} & -1+\frac{C_r l_r - C_f l_f}{MV^2} & 0 & 0 \\ \frac{C_r l_r - C_f l_f}{I_z} & \frac{-C_f {l_f}^2 - C_r {l_r}^2}{I_z V} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ V & l_s & V & 0 \end{bmatrix} \begin{bmatrix} \beta \\ r \\ \Delta\psi_p \\ e_y \end{bmatrix} + \begin{bmatrix} \frac{C_f}{MV} & \frac{C_r}{MV} \\ \frac{C_f l_f}{I_z} & \frac{C_r l_r}{I_z} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_f \\ \delta_r \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -V \\ -l_s V \end{bmatrix} \rho_{ref} + \begin{bmatrix} 0 \\ \frac{1}{I_z} \\ 0 \\ 0 \end{bmatrix} M_{zd} \quad (3.1)$$

where: $l_s = KV$, K is a constant

Table 3.1: Explanation of linear path-tracking vehicle model parameters.

Model	Explanation
β	Vehicle side slip angle
r	Vehicle yaw rate
$\Delta\psi_p$	Heading error
e_y	Path-tracking error
C_f	Front tire cornering stiffness
l_f	Distance between CG and front axle
C_r	Rear tire cornering stiffness
l_r	Distance between CG and rear axle
M	Vehicle mass
V	Vehicle velocity
l_s	Preview distance
I_z	Vehicle yaw moment of inertia
ρ_{ref}	Reference path curvature
M_{zd}	Yaw moment disturbance
K	Preview distance scheduling constant

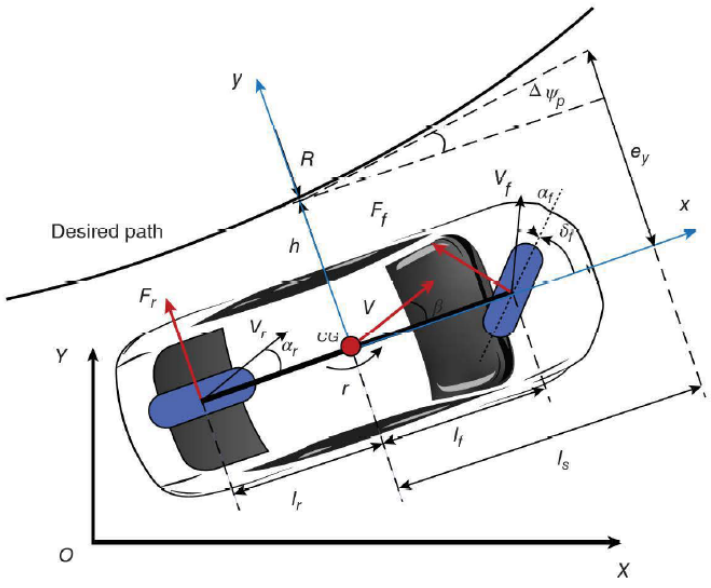


Figure 3.2: Path-tracking scenario showing the desired path, the vehicle offset distance and orientation error with respect to the desired path and the preview distance [17].

3.4 Disturbance Observer

This section presents a general overview of the disturbance observer (DOB). In automotive path-tracking applications, this is also referred to as a curvature rejection filter [17]. In general, the disturbance observer has two main functions: disturbance rejection and model regulation. To observe these effects, one can first consider a simple input-output system consisting of a plant (G) with multiplicative model uncertainty (Δ_m) and an external disturbance (d) applied at the output, as displayed in Figure 3.3. It should be noted that G_n is called the nominal plant. One can then write Equation (3.2) to describe this system, and further obtain Equation (3.3) by defining the extended disturbance *e.*

$$y = Gu + d = \big(G_n(1 + \Delta_m)\big)u + d = G_n u + (G_n \Delta_m u + d) = G_n u + e \tag{3.2}$$

$$e = y - G_n u \tag{3.3}$$

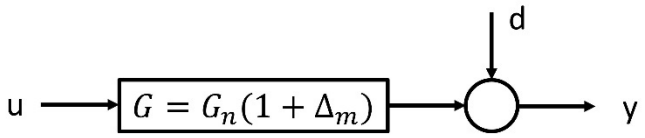


Figure 3.3: Sample input-output system with disturbance and model uncertainty [16].

Since the purpose of the DOB is to achieve disturbance rejection and model regulation, the end goal is to have a system without model uncertainty and external disturbance, which can be described in Equation (3.4).

$$y = G_n u_n \tag{3.4}$$

To determine the input (u) that is necessary to achieve this, one can combine Equation (3.2) and Equation (3.4), and further apply Equation (3.3) to get the result described in Equation (3.5).

$$u = u_n - \frac{y}{G_n} + u \tag{3.5}$$

It must be noted, however, that the result in Equation (3.5) is not implementable for the following two reasons: 1) The input u is on both sides of the expression; 2) $1/G_n$ is not proper since G_n is proper for most physical systems. To make this result implementable, Equation (3.5) is modified to Equation (3.6) where Q is introduced as a unity gain low-pass filter of the appropriate order such that Q/G_n is proper.

$$u = u_n - \frac{Q}{G_n} y + Qu \tag{3.6}$$

Equation (3.6) can be represented in the form of the system block diagram shown in Figure 3.4. It should be noted that an additional external input n is added as sensor noise. Using this block diagram, several transfer functions of interest can be derived as given in Equation 3.7.

$$\begin{cases} \frac{y}{u_n} = \frac{G_n G}{G_n(1-Q)+GQ} \\ \frac{y}{d} = \frac{G_n(1-Q)}{G_n(1-Q)+GQ} \\ \frac{y}{n} = \frac{-GQ}{G_n(1-Q)+GQ} \end{cases} \tag{3.7}$$

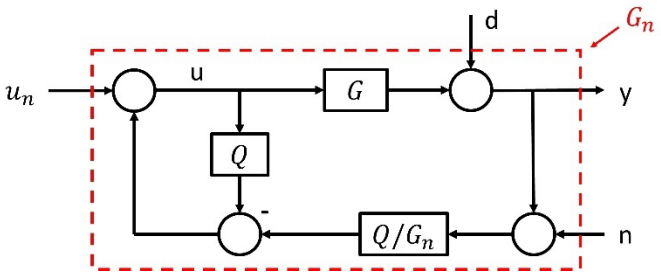


Figure 3.4: The DOB block diagram is shown here. The DOB regulated system within the dashed box acts like the nominal model G_n and disturbances d are rejected within the bandwidth of the Q filter [16].

From Equation (3.7), it can be observed that at low frequencies when $Q = 1$, $y/u_n = G_n$ and $y/d = 0$, demonstrating that model regulation and disturbance rejection are both achieved. Additionally, at high frequencies where $Q = 0$, $y/n = 0$, meaning that high frequency sensor noises can be rejected. It can thus be concluded that this system structure can achieve the desired effect.

In general, to design a DOB, one should construct a unity gain low-pass filter Q and a nominal plant G_n. As discussed previously, the Q filter must be of appropriate order such that Q/G_n is proper. Additionally, the Q filter should also have an appropriate bandwidth to ensure its performance, as a bandwidth too low will result in poor disturbance rejection and model regulation, while a bandwidth too high will cause high frequency sensor noises to enter the system and will also cause stability robustness issues. The nominal or desired plant model/behavior G_n should also be designed carefully as its dynamics must not be too drastically different from that of the uncertain plant G, otherwise the design will fail to achieve its goals.

In the case of applying this control structure to the collision avoidance (treated as a single-lane change here) path-tracking problem, the plant model G can be represented as the transfer function from front steering angle δ_f to path-tracking error e_y as we assume the vehicle is front-wheel-steer. The external disturbance entering the system in this case is the reference path curvature as yaw moment disturbance is assumed to be zero. Hence, u_n and y as displayed in Figure 3.4 are front steering angle δ_f and path-tracking error e_y, respectively. From the linear path-tracking model in Equation (3.1), one can derive that plant G is proper and has a relative order of two, which means that the Q filter must be at least second order to guarantee that Q/G_n is proper. Thus, the Q filter is chosen as a second order system with unity gain in the form displayed in Equation (3.8). Since the control system can be speed-scheduled and the path-tracking model in Equation (3.1) includes vehicle speed and preview distance, system parameters can change as the vehicle moves along the path.

$$Q = \frac{{\omega_n}^2}{s^2+2\xi\omega_n s+{\omega_n}^2} \tag{3.8}$$

The DOB is usually applied together with an additional feedback controller to smooth out its operations further. Figure 3.5 displays the system block diagram with this additional feedback control loop, where C denotes the feedback controller which is designed for the desired nominal plant model G_n based on the model regulation property of DOB.

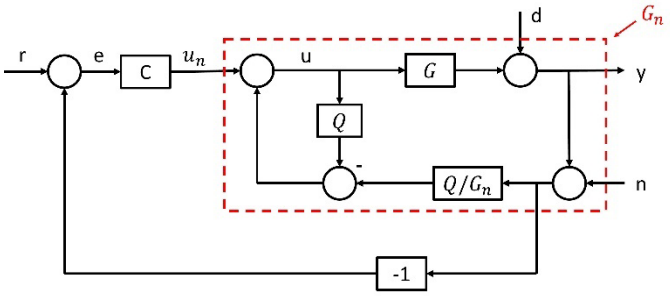


Figure 3.5: Control system with DOB and feedback controller with the DOB regulated plant shown within the dashed rectangle [16].

The feedback controller mentioned above can be of any design. We present an example design that features a speed-scheduled, parameter-space PID controller, the details of which can be found in [16]. The parameter-space method is discussed in detail in [18]. The reference input (denoted as *r* in Figure 3.5) should be zero in this case considering that the goal of the control system is to eliminate path-tracking error e_y. The form of the controller is presented in Equation (3.9). The controller gains (k_p, k_i, k_d) are the parameters to be tuned. Since the controller is speed-scheduled in this case as well, the tunable parameter set has four elements: (V, k_p, k_i, k_d). A D-stability region, as displayed in Figure 3.6, is established for pole placement. An example of the admissible controller gain region at a certain scheduled speed is shown in Figure 3.7. During the process of controller gains value selection, a general rule of thumb is to choose the gains to be as small as possible within the admissible region so that the control effort can be minimized while the energy efficiency maximized.

$$C(s) = k_p + \frac{k_i}{s} + k_d s \tag{3.9}$$

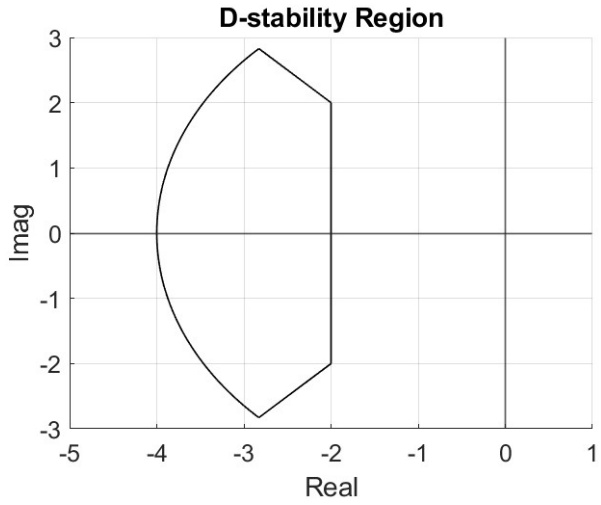


Figure 3.6: D-stability region for desired settling time, desired overshoot and maximum bandwidth constraints on closed loop system pole locations [16].

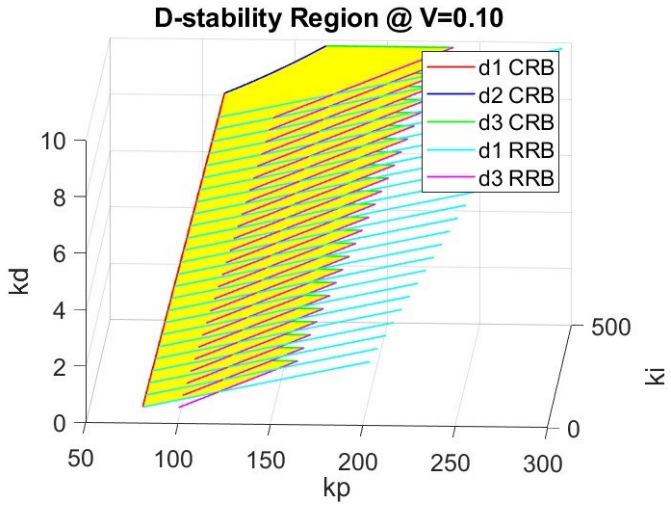


Figure 3.7: Admissible control region at a certain speed showing the controller gain space where D-stability is satisfied [16].

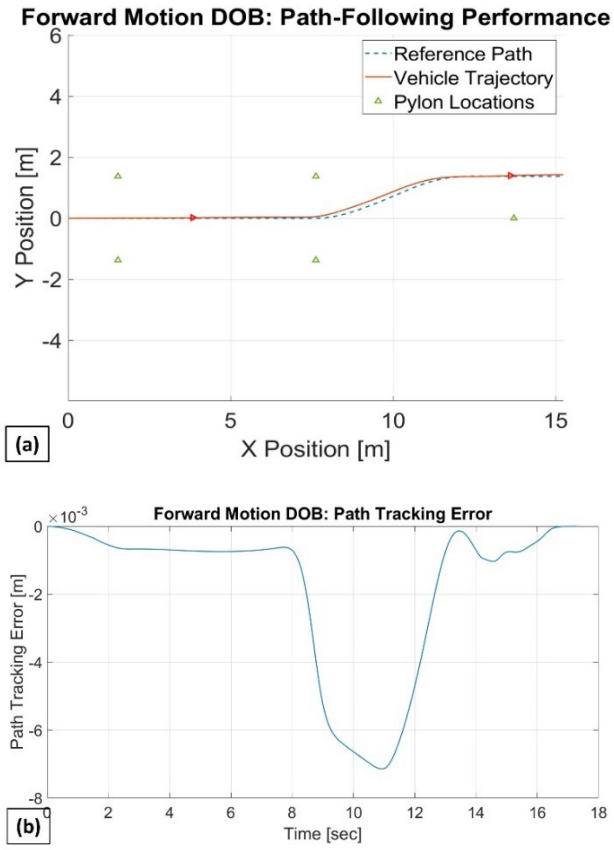
3.5 Simulation Study

Simulation studies are conducted to demonstrate the efficacy of the proposed control design. A Simulink model is constructed to simulate the motions of the vehicle. The parameter values used in the simulations are listed in Table 3.2.

Table 3.2: Parameter values used in the simulation.

Parameter [unit]	Value
C_f [N/rad]	3e5
l_f [m]	2
C_r [N/rad]	3e5
l_r [m]	2
M [kg]	3000
I_z $[kg \cdot m^2]$	5.113e3
ω_n [rad/sec]	100
ζ [unitless]	0.707

The simulation results for standalone DOB, standalone PID and combined PID and DOB control systems are displayed in Figure 3.8, Figure 3.9 and Figure 3.10, respectively.



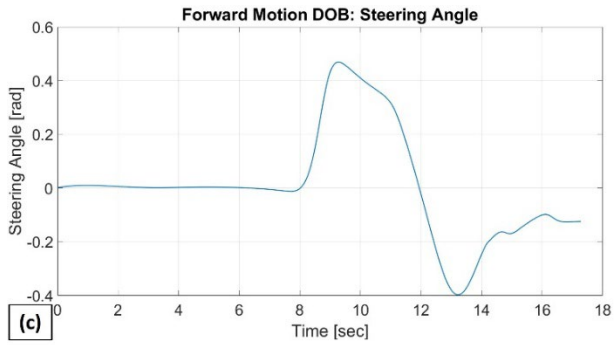
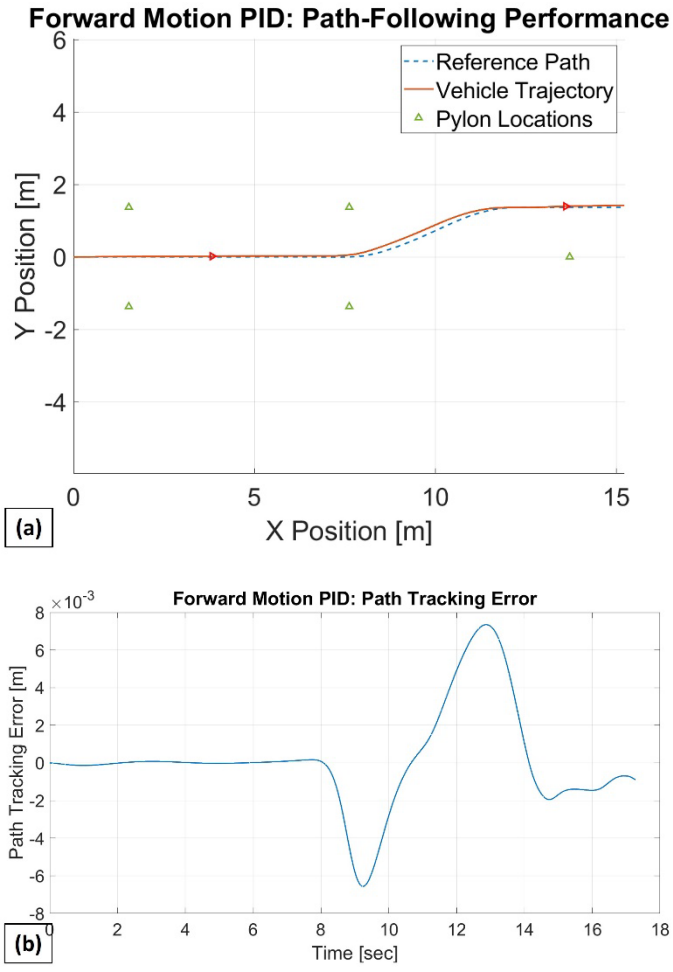


Figure 3.8: Single lane change collision avoidance DOB simulation results [16].



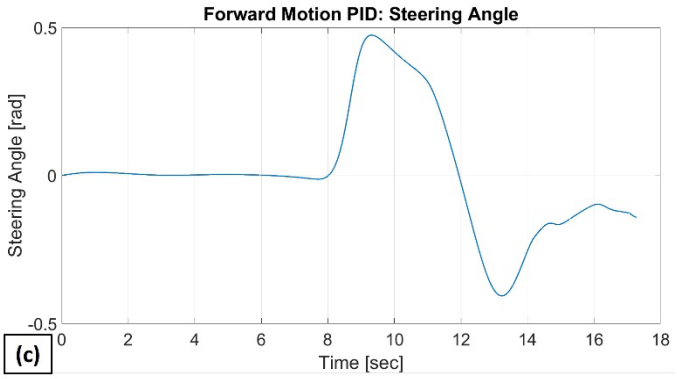
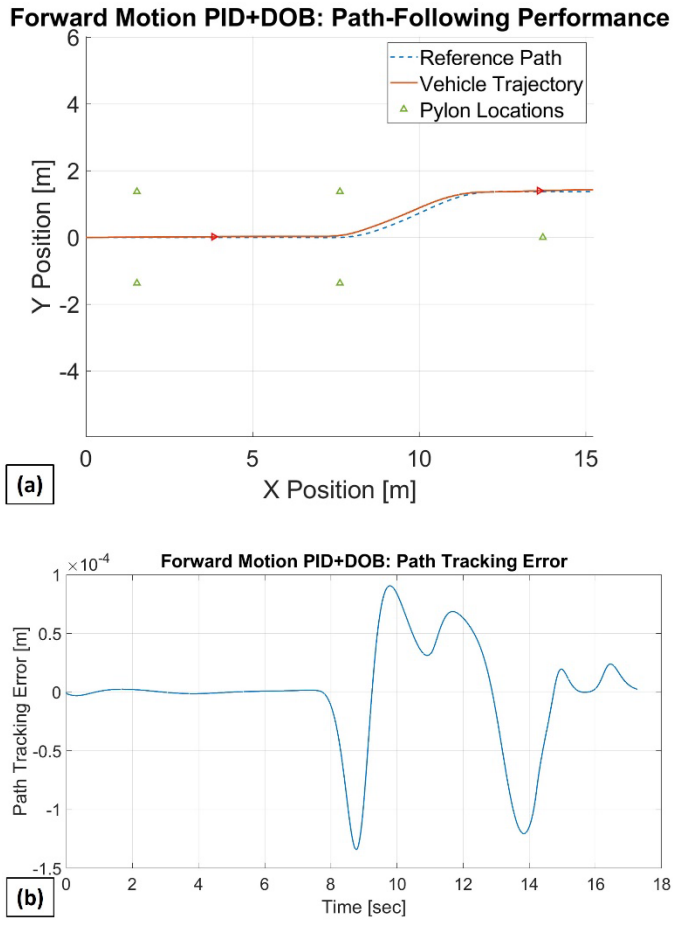


Figure 3.9: Single lane change collision avoidance PID simulation results [16].



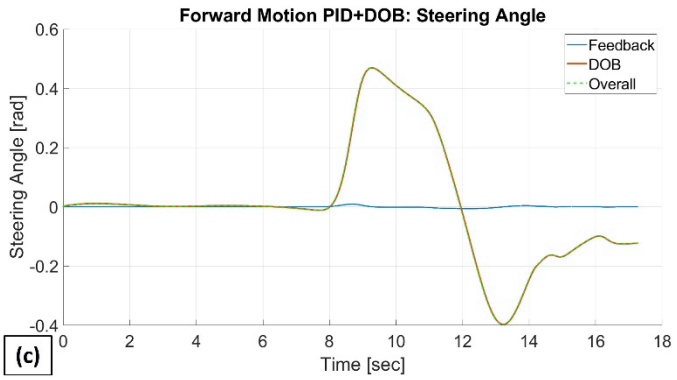


Figure 3.10: Single lane change collision avoidance PID+DOB simulation results [16].

It can be observed from Figures 3.8-3.10 that all three simulations display satisfactory single lane change collision avoidance path-tracking performance. To better compare the results, Table 3.3 is constructed to record maximum absolute path-tracking error, RMS path-tracking error, maximum absolute steering angle as well as maximum absolute steering rate.

Table 3.3: Forward motion simulation results evaluation [16].

Parameter	DOB	PID	PID+DOB
Max absolute path-tracking	0.0071	0.0073	1.3399e-4
RMS path-tracking error	0.0029	0.0027	4.4357e-5
Max absolute steering angle [rad]	0.4692	0.4746	0.4697
Max absolute steering rate [rad/sec]	0.6597	2.4348	2.4538

Results in Table 3.3 show that the control system that combines PID and DOB exhibits the best performance in terms of minimizing path-tracking errors. It is also seen from Figure 3.10 that with the combined DOB and PID control system, the DOB provides the majority of the control action while the PID controller serves mainly to smooth out the steering input

generated by the DOB.

3.6 Conclusions

This chapter reviewed the use of classical controllers along with the DOB method for tracking a collision avoidance path which was chosen as a single lane change maneuver. The parameter space method was used to design the classical controller gains, a PID controller in this case. Simulation results showed the effectiveness of this approach which executes a single lane change type avoidance if a VRU is detected. A learning controller is presented in the next chapter as the focus of this research. The classical approach of following a pre-planned collision avoidance path was presented to show the advantages of the learning control approach which can adapt easily to different encounter situations between the vehicle and the VRU.

Chapter 4

Deep Reinforcement Learning (DRL) Based Collision Avoidance

4.1 Introduction

Path planning and collision avoidance algorithms, as very crucial components of the ADS, largely affect the overall performance of autonomous driving. Unlike traditional car navigation, autonomous driving path planning must consider not only the overall path planning from the starting point to the destination but also local collision avoidance along the path. Therefore, the purpose of this chapter is to present the design of a reliable and robust path planning and collision avoidance algorithm for ADS which can help to increase the overall performance of autonomous driving and enhance the SAE autonomous driving level.

Collision avoidance first requires detection of VRU(s) [11-14] followed by collision avoidance maneuver planning and maneuver tracking. Currently, there are two major approaches to design path planning and collision avoidance algorithms. The first method is the optimization-based approach, which typically treats the path planning and collision avoidance issue as an optimization problem with constraints and then seeks to solve the defined problem. Wang et al. approached collision avoidance as a waypoint position optimization problem, applying the Elastic Band algorithm to iteratively generate a new collision-free trajectory that maintains a socially acceptable distance from Vulnerable Road Users (VRUs) [19]. Morsali et al. introduced a Support Vector Machine (SVM)-based spatiotemporal planning method to compute collision free regions within the spatiotemporal domain. By integrating this method with the A* path search algorithm and SVM-based heuristics, an optimal collision-free path was calculated in complex traffic scenarios [20]. Zhu et al. framed collision-free path searching as an optimization problem on a quintic spline and utilized a look-up table to enhance computational efficiency [21]. Chen et al. proposed a spatiotemporal obstacle avoidance algorithm that improves the efficiency and performance of the hybrid A* algorithm by leveraging a 3D spatiotemporal grid map

[22]. However, the major limitations of the optimization approach are its deficiency in real-time performance caused by computational complexity and its shortage on control feasibility. The second approach is the machine learning based one, which typically treats the path planning and collision avoidance problem as a Markov Decision Process (MDP) and applies reinforcement learning to seek the optimal solution [23-26].

Kendall et al. pioneered the application of the deep-reinforcement-learning (DRL) framework in autonomous driving, innovatively proposing an end-to-end model structure for autonomous driving [27]. Yurtsever et al. proposed an innovative hybrid deep-reinforcement-learning framework to develop Automated Driving Systems (ADS) [28]. Aksjonov et al. proposed a control framework that combines the strengths of both traditional rule-based approaches and machine learning to develop an autonomous driving system [29]. Furthermore, Makantasis et al. introduced an innovative driving policy based on Double Deep Q-Networks (DDQN) that is adaptable to mixed driving environments. They conducted a series of tests to evaluate the algorithm's efficacy across varying levels of market penetration [30]. Nageshrao et al. integrated DDQN with a short-horizon safety mechanism to design an autonomous driving system tailored for highway conditions, tested under various traffic density settings. [31]. Peng et al. developed an end-to-end ADS using a Dueling Double Deep Q-Network (DDDQN) framework. They validated the efficiency and effectiveness of this method using The Open Racing Car Simulator (TORCS) [32]. Jaritz et al. introduced an Asynchronous Actor-Critic (A3C) based method for autonomous driving which maps RGB images from the front camera to driving actions using a realistic rally racing game environment for training. The approach demonstrates faster convergence and more robust performance compared to other DRL based end-to-end methods, indicating its potential for practical applications in autonomous vehicles [33]. In order to handle critical pre-accident scenarios in emergency situations, Merola et al. proposed a Deep Q-Network (DQN) based approach to design ADS and training the system to execute emergency maneuvers to minimize or avoid damage [34]. Cao et al. introduced a hierarchical reinforcement and imitation learning (H-REIL) approach for autonomous driving to handle near-accident scenarios. By integrating a low-level imitation learning controller with a high-level reinforcement learning controller, their approach demonstrated capability of balancing

safety and efficiency[35]. However, a notable disadvantage of the machine learning-based approach is the instability in model performance under normal traffic conditions due to the absence of hard-coded safety protocols.

To address the critical challenges outlined earlier, we introduce a novel hybrid hierarchical DRL framework designed to enhance collision avoidance performance in autonomous driving in this chapter. This framework uniquely combines a path following controller with DRL-based collision avoidance algorithms. Under typical VRU-free conditions, the vehicle employs a low-level path-tracking controller to maintain precise navigation. However, when VRUs are detected nearby, a high-level DRL collision avoidance controller is activated, enabling the vehicle to adjust its speed or trajectory to avoid potential collisions. The primary contribution of this research lies in the seamless integration of traditional path following control techniques with advanced machine learning-based collision avoidance strategies. This hybrid approach significantly improves both path tracking and collision avoidance capabilities, providing a robust foundation for future research in hybrid DRL based control for autonomous vehicles.

4.2 Methodology

4.2.1 Vehicle Model

The vehicle model we used for model-in-the-loop (MIL) simulation in this chapter is a simplified linear vehicle model. We combined the linear longitudinal vehicle model with lateral single-track vehicle model to create this linear enhanced model. Figure 4.1 displays the geometry of the longitudinal vehicle model. The elements depicted include: 1) F_a = aerodynamic drag due to headwind at velocity V_{wind}; 2) F_r = rolling resistance; 3) θ = road grade; 4) Mg = the gravitational force of the vehicle; 5) F_g = the component of the gravitational force acting along the slope of the road; 6) F_x = the force exerted by the tires in the longitudinal direction. The input-output configuration of this model is shown in Figure 4.2, where inputs such as throttle and brake pedal positions, headwind velocity, and road slope are used to determine the vehicle's longitudinal speed. It should be noted that for simplification, the model assumes no headwind or road grade. Figure 4.3 details the

structure of this model, with Figure 4.3(a) outlining how longitudinal forces translate into vehicle speed, and Figure 4.3(b) showing how input engine and brake torques are converted into longitudinal tire force. Table 4.1 enumerates the parameters featured in Figure 3.

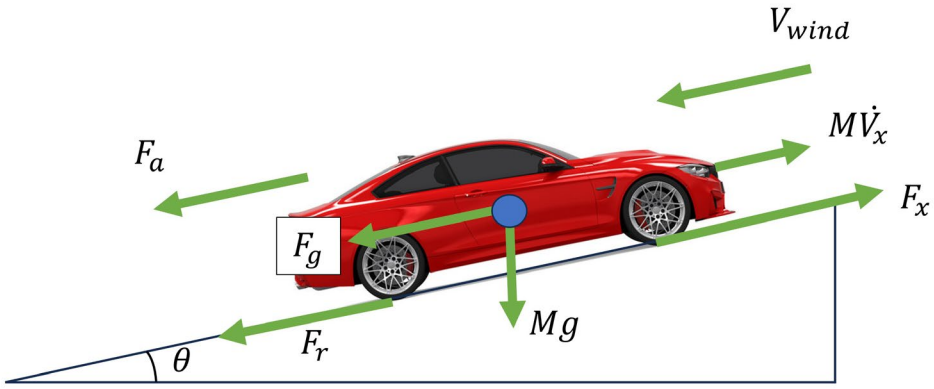


Figure 4.1: Longitudinal vehicle model geometry [36].

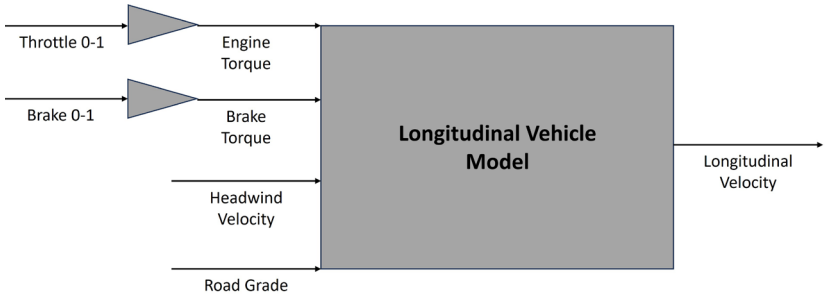
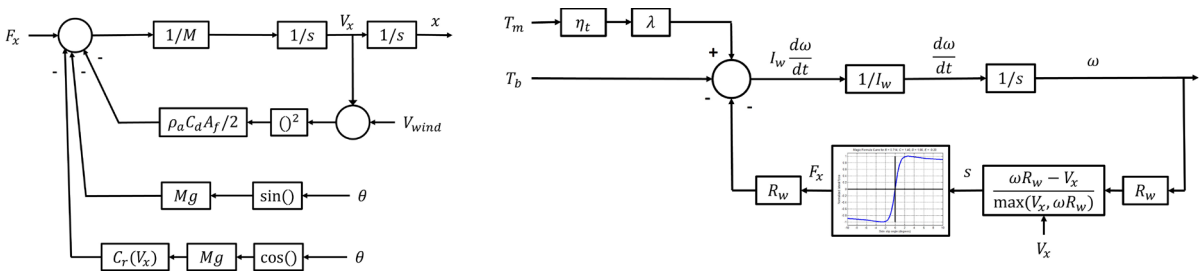


Figure 4.2: Longitudinal vehicle model input-output structure [36].



(a) (b)

Figure 4.3: Detailed components in the longitudinal vehicle model with (a) Longitudinal forces to longitudinal vehicle velocity calculation and (b) Input torques to longitudinal tire force calculation [36].

Table 4.1: Longitudinal model parameters [36].

Symbol	Parameter
F_x	Longitudinal tire force
M	Vehicle mass
V_x	Vehicle longitudinal velocity
X	Vehicle longitudinal position
ρ_a	Air density
C_d	Air drag coefficient
A_f	Vehicle cross-sectional area
V_{wind}	Headwind velocity
θ	Road grade
C_r	Rolling resistance coefficient
T_m	Motor torque
T_b	Brake torque
η_t	Transmission efficiency
λ	Gear ratio
I_w	Wheel moment of inertia
ω	Wheel angular velocity
R_ω	Wheel radius
s	Longitudinal tire slip

Figure 4.4 presents the geometry of the lateral single-track model while Equation 4.1 [17] provides the state-space form of the simplified linear single-track model. The parameters for this lateral model are demonstrated in Table 4.2. The model inputs include the steering angles of the front and rear wheels, along with any vehicle yaw disturbances. The outputs of the model are the vehicle's side-slip angle and yaw rate. For simplification, this model considers a front-wheel-steering vehicle with no external yaw disturbances, meaning the primary input is the front wheel steering angle. Furthermore, in this simplified version, the vehicle's longitudinal speed is assumed to remain constant.

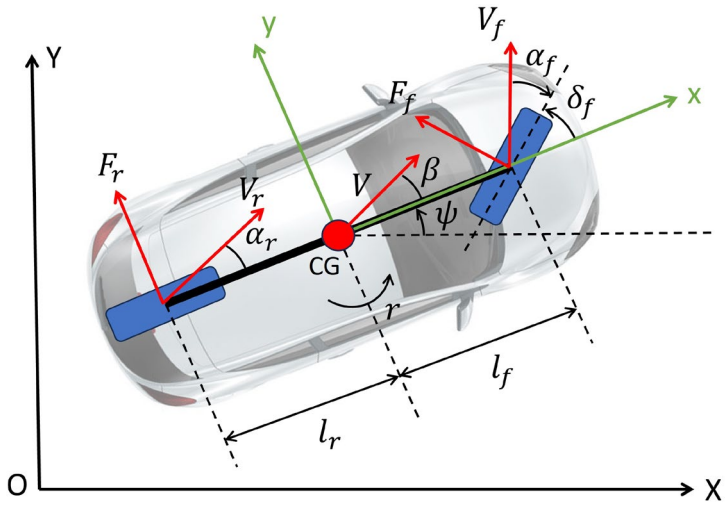


Figure 4.4: Lateral single-track vehicle model geometry [36].

$$\begin{bmatrix}\dot{\beta}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}\frac{-C_f-C_r}{MV} & -1+\frac{C_rl_r-C_fl_f}{MV^2}\\ \frac{C_rl_r-C_fl_f}{I_z} & \frac{-C_fl_f{}^2-C_rl_r{}^2}{I_zV}\end{bmatrix}\begin{bmatrix}\beta\\ r\end{bmatrix} + \begin{bmatrix}\frac{C_f}{MV} & \frac{C_r}{MV}\\ \frac{C_fl_f}{I_z} & \frac{C_rl_r}{I_z}\end{bmatrix}\begin{bmatrix}\delta_f\\ \delta_r\end{bmatrix} + \begin{bmatrix}0\\ \frac{1}{I_z}\end{bmatrix}M_{zd} \qquad (4.1)$$

Table 4.2: Lateral model parameters [36].

Symbol	Parameter
X, Y	Earth-fixed frame coordinate
x, y	Vehicle-fixed frame coordinate
V	Vehicle center-of-gravity (CG) velocity
M	Vehicle Mass
I_z	Vehicle yaw moment of inertia
β	Vehicle side-slip angle
ψ	Vehicle yaw angle
r	Vehicle yaw rate
M_{zd}	Yaw disturbance moment
δ_f, δ_r	Front & rear wheel steer angle
F_f, F_r	Front & rear lateral tire force
V_f, V_r	Front & rear axle velocity
α_f, α_r	Front & rear tire slip angle
l_f, l_r	Distance between vehicle CG and front & rear axle
C_f, C_r	Front & rear tire cornering stiffness

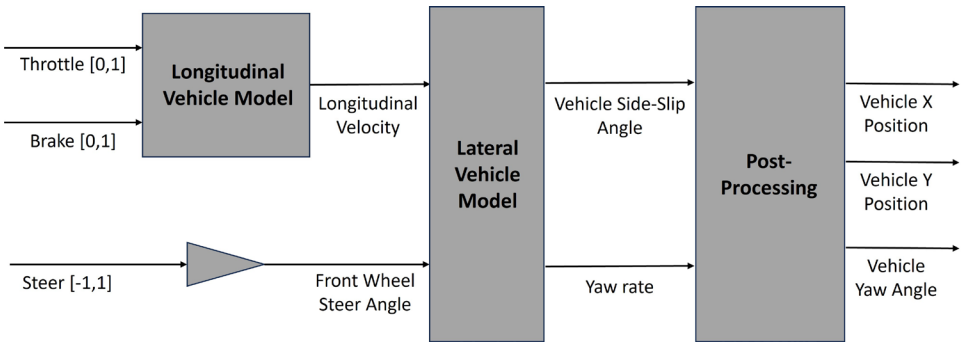


Figure 4.5: Full vehicle model structure [36].

4.2.2 Low-level PID Pure Pursuit Path Tracking Controller Design

The low-level controller used in this chapter is a PID pure pursuit path tracking controller which enables accurate path following in normal traffic conditions. This PID path tracking controller contains a longitudinal velocity controller and a lateral tracking controller. The longitudinal PID controller primarily manages the vehicle's speed by using

the speed differential between the vehicle's current speed and the target speed, as detailed in Equation 4.2, to produce throttle and brake command. Conversely, the lateral PID controller is responsible for steering control. It uses the angular difference between the vehicle's current trajectory and the desired path direction, outlined in Equation 4.3, to generate a steering command. Figure 4.6 illustrates the method used to determine the angle difference for the lateral PID controller. The PID gains (K_p, K_i, K_d) for both the longitudinal and lateral PID controllers have been manually adjusted to achieve optimal performance tailored to the vehicle model and CARLA simulator.

$$\varepsilon_{longitudinal} = v_{vehicle} - v_{desired} \tag{4.2}$$

$$\varepsilon_{lateral} = \theta_{vehicle} - \theta_{path} \tag{4.3}$$

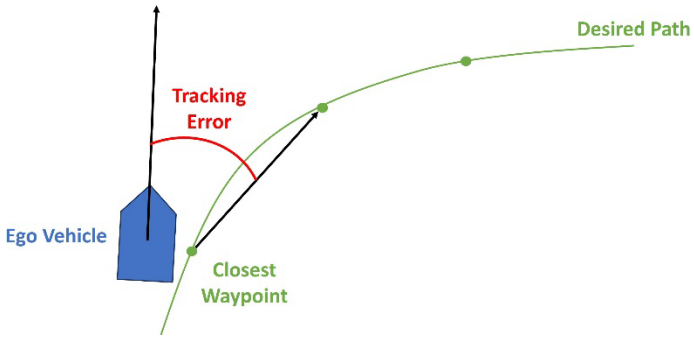


Figure 4.6: Lateral PID controller path tracking error illustration [36].

4.2.3 High-level DRL Based Collision Avoidance Controller Design

The collision avoidance process for an autonomous vehicle (AV) is a dynamic, ongoing decision-making task. Initially, the AV identifies the positions, velocities, and trajectories of surrounding road users. Based on its own location and velocity, the AV then makes quick, precise decisions to navigate safely around these users. This entire decision-making process is like a Markov Decision Process (MDP), suggesting that collision avoidance can be approached as a classic MDP problem. MDPs are useful for modeling and addressing the uncertainties in traffic environments.

To address this MDP, deep reinforcement learning (DRL) is employed to develop an autonomous driving system designed to optimize decisions to maximize expected rewards while enhancing safety and efficiency. The fundamental elements of an MDP include states (S), actions (A), transition probabilities (P), and rewards (r).

- State Space (S): The state space of the model contains a set of states, each representing specific information about the current traffic environment. There exist four key components in each state. The first one is the occupancy grid which represented by a 2-D array. The occupancy grid maps the surrounding obstacles relative to the vehicle. This grid identifies road users as obstacles within the predefined detection ranges of 20 meters ahead, 5 meters behind, and 7 meters to each side. Different weights are assigned to each grid cell based on the importance of nearby road users, with higher weights given to vulnerable road users (VRUs). Figures 4.7 and 4.8 illustrate the occupancy grid for vehicles with zero and nonzero yaw angles, respectively. The grid uses cross symbols to denote various elements: white crosses indicate collision-free areas, black crosses mark potential pedestrian collisions, red denotes the vehicle's geometric center, and blue represents the vehicle's coordinates. The second component is the ego vehicle's data which includes current information about the ego vehicle such as its location, orientation, and velocity. This component is crucial for dynamic decision-making and navigation. The third component is path tracking information which includes target tracking waypoints. The last component is obstacle Information which contains the time-to-collision zones (TTZ) for both vehicles (TTZ_v) and pedestrians (TTZ_p), as well as the difference between these zones (TTZ_diff), providing a measure of imminent threat from different obstacles.
- Action Space (A): The action space contains a variety of discrete actions that the ego vehicle can execute under different traffic situations. Each action is defined as a tuple of control commands, including steering, throttle, and brake inputs. The design of the action space is tailored to meet the specific demands of various test scenarios, ensuring that the vehicle can adaptively respond to different driving conditions.
- Transition Model (P): The transition model is an important component of the traffic simulation framework, designed to project next states based on the execution of specific

actions at the current state. In this chapter, the transition model is separated into two major components. The first part is a SIMULINK vehicle model which is used to simulate the dynamics of the ego vehicle, providing a detailed analysis of its motion based on the input actions. The specifics of the vehicle model, including its operational parameters and integration into the simulation environment have been presented above in the vehicle model sections of this chapter. The second part is the CARLA simulator which is used to replicate the traffic environment and the movements of other road users. Detailed discussions on the functionalities and contributions of the CARLA simulator to the overall traffic simulation are provided in the case study section.

- Reward (r): The reward function calculates immediate rewards based on the transition from the current state to the next after an action is executed. The detailed discussions of rewards function design are also provided in the case study section.

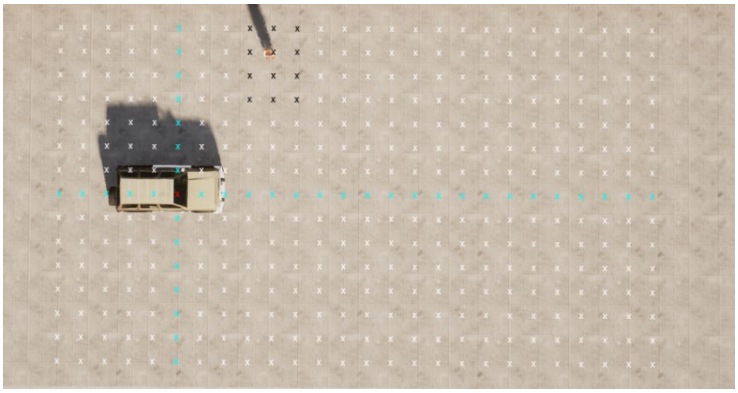


Figure 4.7: Occupancy grid 2-D array with zero yaw angle [36].

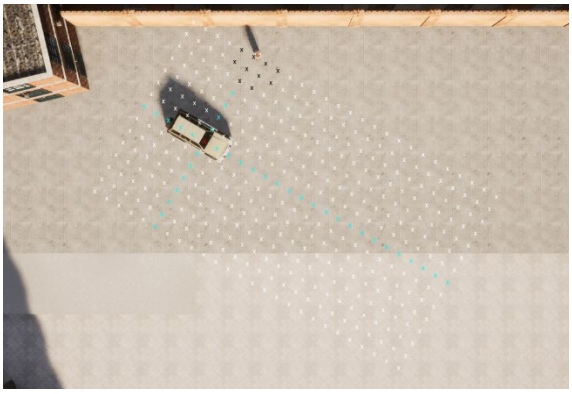


Figure 4.8: Occupancy grid 2-D array with 30° yaw angle [36].

To enhance performance, the Double Deep Q-Network (DDQN) method is utilized since it reduces the overestimation of action values common in DQN by separating action selection from target Q-value generation. Unlike traditional optimization or supervised learning methods, DDQN directly learns from environmental interactions, making it highly effective for autonomous driving tasks. This approach not only handles high-dimensional inputs like the occupancy grid but also learns without the need for pre-labeled data, offering a significant advantage over other collision avoidance techniques. A comparison of traditional and DDQN methods is detailed in Table 4.3, illustrating the benefits of the DDQN approach in autonomous vehicle navigation.

Table 4.3: Comparison to traditional optimization-based approach [36].

Approaches	**Pros**	**Cons**
Elastic Band [19]	1. Easy to implement. 2. Can avoid getting stuck at local minimum 3. Flexibility in local path modification.	1. Path shape may be irregular especially in complex environment. 2. Computational complexity may increase with number of obstacles. 3. Path may be control infeasible
Potential Field related	1. Easy to implement. 2. Can achieve real time performance. 3. Path easy to visualize and understand.	1. Sometimes stuck at local minimum, especially in complex environment. 2. Oscillations may occur around obstacles. 3. Path may be control infeasible.
SVM based optimization [20]	1. Path Planning in Spatial-Temporal region. 2. Can find optimal, efficient and control feasible path.	1. Sometimes stuck at local minimum, especially in complex environment. 2. Oscillations may occur around obstacles. 3. Path may be control infeasible.
Other Optimization Based Method [11], [35]	1. Can generate control feasible and optimal (either time or fuel efficient) path. 2. Can adapt to different traffic scenarios.	1. Computational inefficient and may not achieve real time performance. 2. Performance of the optimization might be sensitive to the tuning of parameters.
Proposed DDQN Based ADS	1. Learning ability. 2. Model can achieve real time performance. 3. Can adapt to different traffic scenarios.	1. Training requires good computational resources. 2. Performance of the model depends on training data quality.

Compared to other DRL algorithms, DDQN offers distinctive advantages. Currently, there are primarily two DRL based approaches: policy-based and value-based methods. Policy-based methods including Asynchronous Advantage Actor Critic (A3C) and Proximal Policy Optimization (PPO) concentrate on directly optimizing the policy that generates the agent's actions, aiming to enhance expected long-term rewards. These methods excel in environments with high-dimensional or continuous action spaces, offering robustness and stability during training. However, they tend to be less training-efficient than value-based methods. This inefficiency comes from their need to interact more frequently with the environment and their requirement to discard old data after policy updates, which can slow the learning process. Value-based methods, such as DDQ and DDQN, focus on estimating the values of actions (Q value) from each state, thereby indirectly optimizing a policy by selecting actions that maximize these estimated values. Unlike policy-based methods, these methods are off-policy which allows for the reuse of previously generated data, significantly improving training efficiency. This attribute makes value-based methods particularly useful in fields like collision avoidance because they can quickly adapt to dynamic traffic environments. Additionally, Deep Deterministic Policy Gradient (DDPG) merges elements of both policy-based and value-based approaches. It employs a policy network for action generation and a value network for action evaluation and is suitable for continuous action spaces. However, DDPG is complex to implement and highly sensitive to hyperparameter settings, which make it less ideal for complicated traffic environment simulation. Therefore, in this chapter, DDQN is selected as the preferred DRL framework for developing ADAS for collision avoidance.

Figure 4.9 shows the neural network architecture utilized in the DDQN. This network contains four fully connected hidden layers: three layers each containing 128 units, followed by a final layer with 32 units. The network processing starts with the flattening of the occupancy grid. Then the flattened occupancy grid is sequentially fed through the first three 128-unit layers. After processing through these layers, additional sensor data including the ego vehicle’s status, path tracking waypoints, and information of other road users are integrated with the output from the third layer and fed as input into the final 32-unit layer. This architectural is intentional, aiming to preserve important information that might be lost

in earlier layers, thus ensuring the network considers vital details in its outputs. Furthermore, the detailed workflow of the DDQN algorithm is demonstrated in Table 4.4. The DDQN employs a dual neural network structure to optimize its decision-making capabilities. The online network Q is responsible for generating optimal actions for current states. Concurrently, the target network $\hat{Q}$, used for gradient descent updates its parameters only after specified intervals. This staggered updating helps stabilize the training process by reducing rapid shifts in learning targets. Training data generated from the SIMULINK vehicle model and the CARLA traffic environment are stored in a replay buffer which breaks the correlation between consecutive steps and decreases variance in model training process. This method enhances the efficiency of data utilization and guarantees a more balanced and comprehensive sampling from the replay buffer throughout the training phases. Key implementation details for the DDQN training include: 1. Learning Rate which is set at 0.01 to moderate the speed of learning. 2. Initial Exploration Probability which starts at 1, allowing for maximum exploration initially. 3. Decay Period such that the exploration rate decays over 200,000 steps, gradually shifting focus from randomly exploration towards exploiting learned strategies. 4. Final Exploration Probability which is reduced to 0.05 at the end, balancing the need for exploration with exploitation. 5. Reward Discount Factor which is fixed at 0.9 and this parameter prioritizes immediate over distant rewards, affecting the strategy's short-term focus. 6. Episode Limit where each training episode is capped at 4,000 steps to keep the training duration per episode manageable.

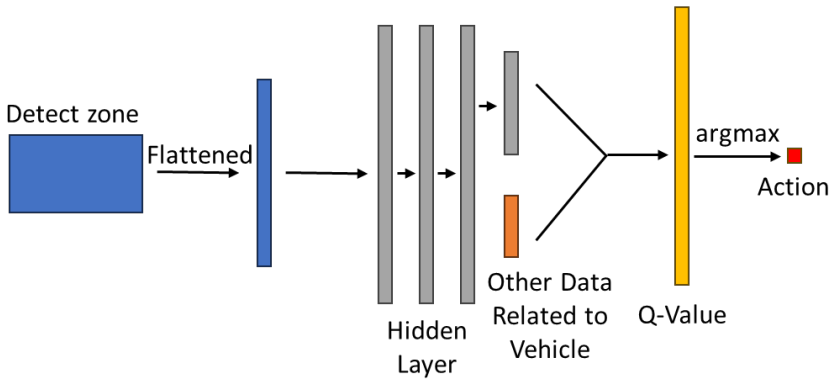


Figure 4.9: DDQN framework neural network structure [36].

Table 4.4: DDQN algorithm flowchart [36].

Algorithm 1

Initialize replay memory D
Initialize target network $\hat{Q}$ and Online Network Q with random weights θ
for each episode **do**
    Initialize traffic environment
    for t = 1 to T **do**
        With probability ϵ select a random action a_t
        Otherwise select $a_t = \max_a Q^*(s_t, a; \theta)$
        Execute a_t in CARLA and extract reward r_t and next state s_{t+1}
        Store transition (s_t, a_t, r_t, s_{t+1}) in D
        if t mod training frequency == 0 **then**
            Sample random minibatch of transitions (s_j, a_j, r_j, s_{j+1}) from D
            Set $y_j = r_j + \gamma \max_{a_{j+1}} \hat{Q}(s_{j+1}, \text{argmax}_{a_{j+1}} Q(s_j, a_{j+1}; \theta); \theta)$
            for non-terminal s_{j+1}
            or $y_j = r_j$ for terminal s_{j+1}
            Perform a gradient descent step to update θ
            Every N steps reset $\hat{Q} = Q$
        end if
        Set $s_{t+1} = s_t$
    end for
end for

4.3 Results

To validate the effectiveness of the proposed routine, we introduce two pedestrian collision avoidance scenarios designed for model-in-the-loop (MIL) evaluation. The testing is conducted using the CARLA simulator which provides a realistic traffic environment coupled with the SIMULINK vehicle dynamics model to simulate vehicle behavior accurately. This setup enables a comprehensive assessment of the routine under controlled, yet realistic traffic conditions, ensuring a robust evaluation of its capabilities.

4.3.1 Test Case 1

The first traffic scenario, illustrated in Figure 4.10, involves a pedestrian entering the crosswalk as a vehicle approaches. In this scenario, the crosswalk becomes a potential collision zone, requiring the vehicle to gradually slow down and stop before reaching the zone's edge. It's important to note that the DRL module's action space in this scenario will be limited to throttle and brake actions as the situation only necessitates a longitudinal motion maneuver. Additionally, to enhance the learning efficiency of the DRL module, several exemplary slowing down speed profiles are provided. These profiles help guide the DRL module by reducing its reliance on random action selection, significantly shortening the model training time.

Figure 4.11 illustrates the progression of step rewards as the number of training episodes increases. The rewards increase with the episodes and tend to stabilize after 1000 episodes, suggesting convergence and the successful identification of the optimal policy. This particular test case is relatively straightforward and allows for a scenario where even randomly generated actions stand a reasonable chance of completing the driving task successfully. Consequently, the reward progression shows limited improvement across training episodes after the initial period, reflecting the simplicity of the task. Nonetheless, the proposed deep reinforcement learning (DRL) model proves its effectiveness by consistently enabling the vehicle to successfully complete test cases, demonstrating its robust capability in typical traffic scenarios.

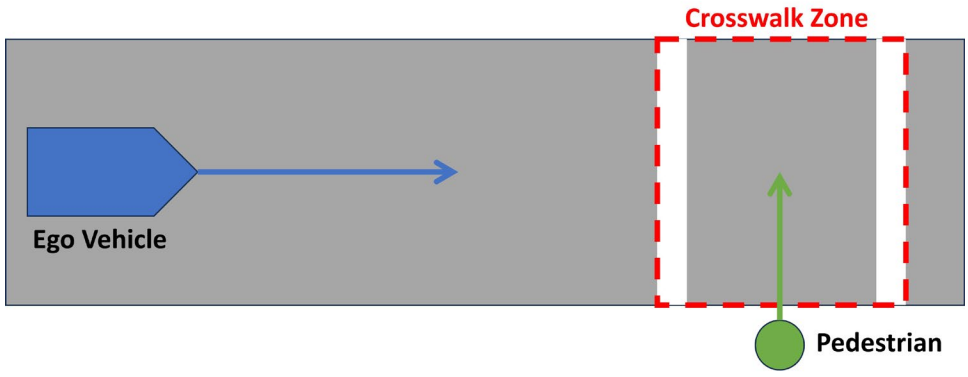


Figure 4.10: Traffic Scenario 1 setup [36].

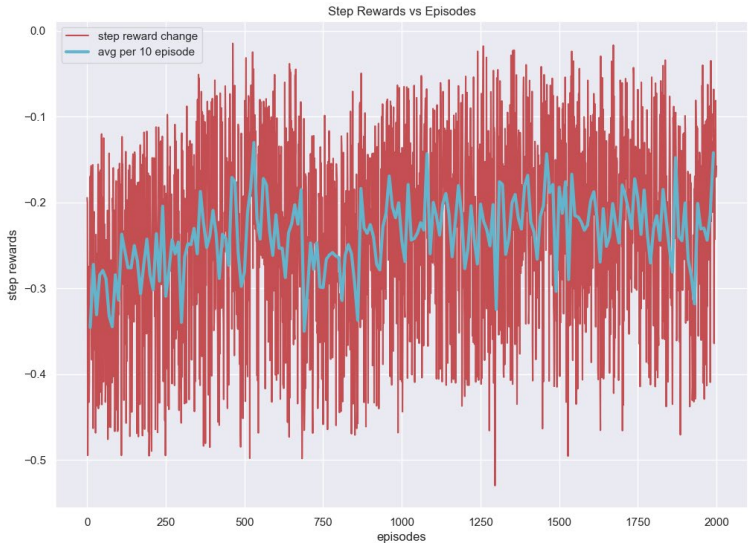


Figure 4.11: Scenario 1 step reward vs. training episodes [36].

Figure 4.12 displays the evolution of the Time-To-Collision Zone (TTZ) for both the pedestrian and the vehicle. The three colored arcs represent the boundaries of the severity levels, with level one being the most critical and requiring immediate actions to avoid likely collisions, and level three being the least urgent such that the vehicle has enough time and space to react to potential collision risks. The consistently maintained TTZ of over four seconds for both the pedestrian and the vehicle indicates a low likelihood of collision, further validating the efficacy of the optimal policy. Scenario 1 speed following is shown in Figure 4.13.

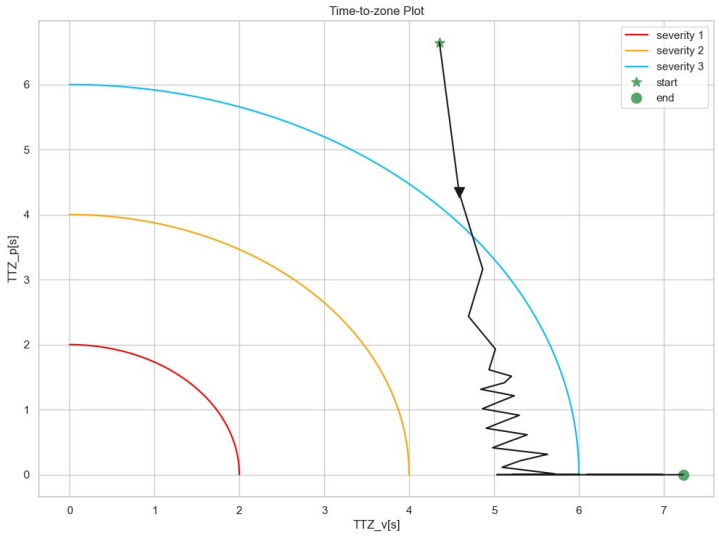


Figure 4.12: Scenario 1 TTZ progression [36].

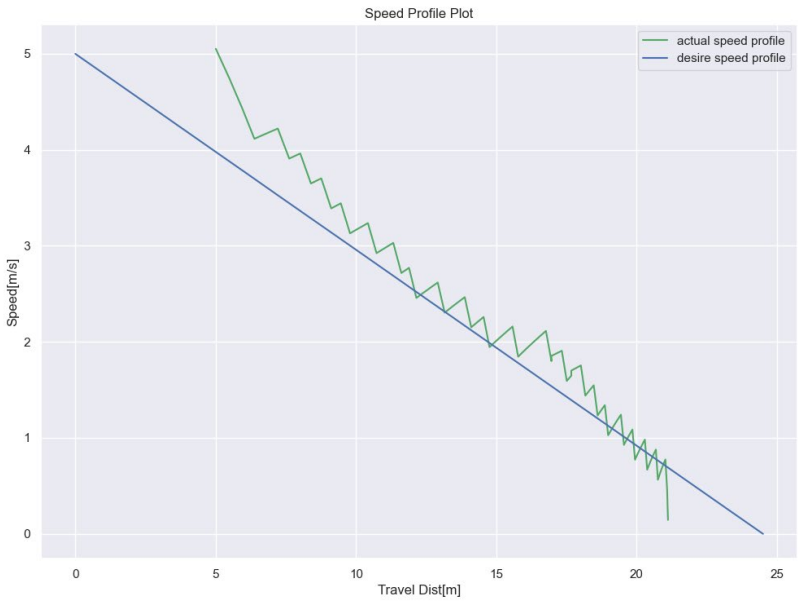


Figure 4.13: Scenario 1 speed following performance. [36].

4.3.2 Test Case 2

Figure 4.14 illustrates the second traffic scenario where a pedestrian unpredictably enters the road as the vehicle approaches, creating a potential collision risk. In this scenario, the pedestrian is considered a moving obstacle, and the vehicle must maneuver around the pedestrian by steering to avoid a collision. Consequently, the action space for the DRL module in this scenario expands to include not only longitudinal controls, such as throttle and brake inputs, but also steering actions.

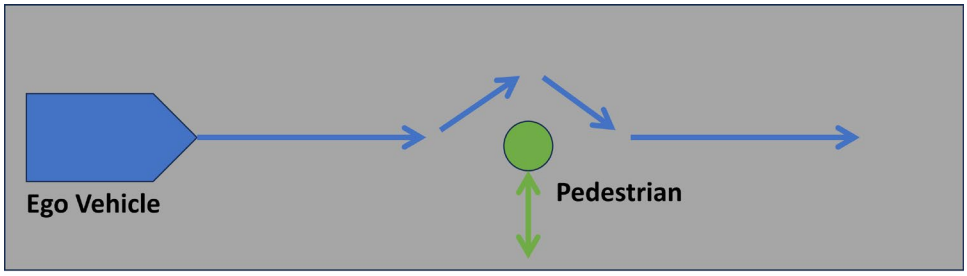
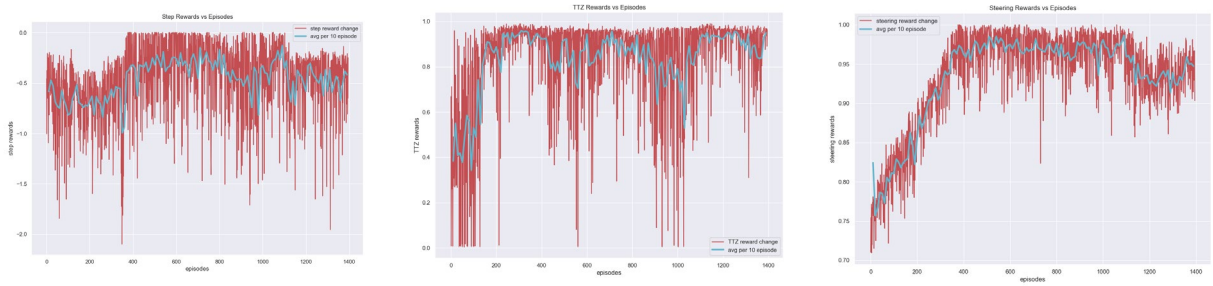


Figure 4.14: Traffic Scenario 2 setup [36].

Figure 4.15 tracks the reward progression during the training of scenario 2. Rewards increase with the completion of more episodes and eventually stabilize at a high level, indicating successful optimization of the policy. Specifically, the convergence of the Time-To-Collision Zone (TTZ) reward indicates that the agent effectively learns to avoid collisions with pedestrians, as further evidenced by Figure 4.16 which shows low collision risk. Additionally, the convergence of steering rewards suggests that the agent consistently selects appropriate steering angles during maneuvers. Moreover, as demonstrated in Figure 4.17, the agent's behavior includes slowing down as it approaches the pedestrian and then accelerating once the collision risk has passed. This reaction pattern is desirable for safely handling such maneuvers. The link to the demo video of Scenario 2 is provided here: https://youtu.be/CmGtaAjZ_x4. This video serves to further illustrate the practical application and effectiveness of the DRL model in real-world-like simulations, showcasing its capability to navigate complex traffic scenarios safely and efficiently.



(a) (b) (c)

Figure 4.15: Training reward vs. episodes: (a) Step reward; (b) TTZ reward; (c) Steering reward [36].

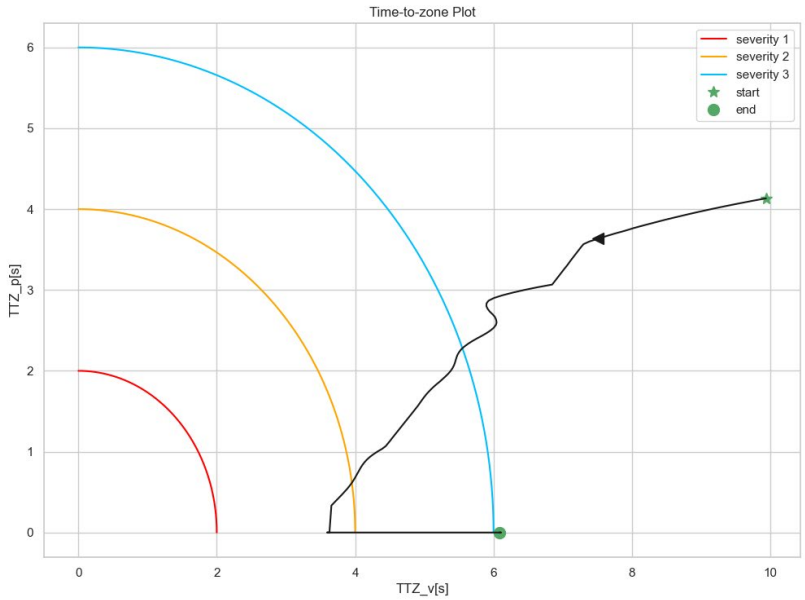


Figure 4.16: Scenario 2 TTZ progression [36].

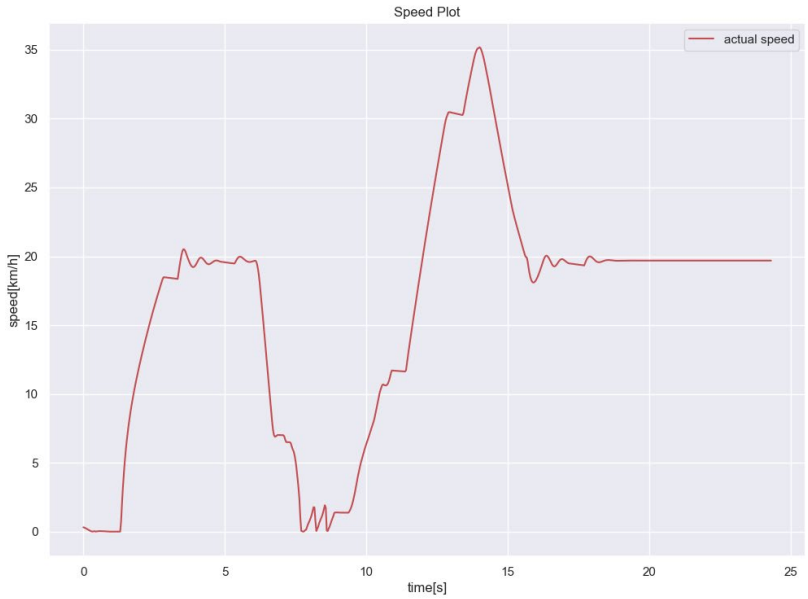


Figure 4.17: Scenario 2 vehicle speed progression [36].

4.4 Conclusions

Effectively planning a trajectory for connected autonomous vehicles that is both collision-free and control-feasible can help increase the vehicle's SAE autonomous driving level, thereby reducing car accidents caused by human error. This chapter proposed an innovative hybrid hierarchical autonomous driving controller which integrates DRL based local collision avoidance controller with global PID path following. Specifically, the DRL based collision avoidance component employs a Double Deep Q-Network (DDQN) framework to train autonomous driving agents. The agents are trained to not only prevent collisions but also maintain a socially acceptable distance from Vulnerable Road Users (VRUs) when potential collision may happen. The PID based pure pursuit controller ensures the vehicle to perform precision path tracking on a pre-calculated optimal path under standard traffic conditions.

The proposed method was evaluated by model-in-the-loop (MIL) simulations which demonstrated superior performance compared to other methods. It is mainly because the proposed method took the strengths of both model-based modular control and machine learning-based collision avoidance. Under normal traffic conditions, hard coded protocol and high-performance path tracking controller can guarantee the precision navigation. Under emergency traffic condition, flexible DRL based collision avoidance controller can perform collision avoidance to dodge the pedestrian by either decelerating and/or changing the path. In addition, the application of the MIL approach enables comprehensive testing of the proposed method across a wide array of edge cases, which makes results more convincing and persuasive. However, there are still some limitations of the proposed method. The majority of shortcoming of the proposed method came from its hybrid structure. Under certain traffic conditions, the switch from the PID-based path tracking controller to the emergency DRL-based collision avoidance controller can result in abrupt maneuvers. These sudden changes in movement may cause discomfort to passengers. Moreover, the real-time performance of the proposed algorithm under different traffic conditions requires further study.

In general, this chapter proposed a novel approach to designing the ADS which not only fulfill the gaps in the autonomous driving path planning research in hybrid controller

but also offers insight for future study. The future research should focus more on smooth transition between different controllers and improving real-time performance under all traffic conditions. Moreover, the proposed method requires further testing and evaluation using hardware-in-the-loop (HIL) and vehicle-in-virtual environment (VVE) simulations. The details of vehicle-in-virtual environment implementation are presented in the next chapter.

Chapter 5

Vehicle-in-Virtual-Environment (VVE)

5.1 Introduction

The typical development process for ADAS systems employed by OEMs generally follows these sequential steps: 1) extensive model-in-the-loop (MIL) simulation; 2) hardware-in-the-loop (HIL) testing; 3) proving ground testing; and 4) public road testing. As testing progresses, increasing levels of hardware and realism are introduced into the system. For instance, model-in-the-loop (MIL) is a purely virtual simulation tool, detailed in [38], while hardware-in-the-loop (HIL) testing begins to incorporate physical components, as discussed in [37-38]. Both MIL and HIL simulations rely on validated vehicle models with varying levels of fidelity that incorporate longitudinal [41], lateral [42], and vertical dynamics [43]. However, the VVE approach eliminates the need for these models since it uses the actual vehicle.

Public road testing, as described, is not only expensive but also cost-ineffective, as encountering rare and extreme scenarios may require millions of miles of driving. Moreover, this type of testing inherently involves other road users as involuntary participants, raising significant safety concerns. To address these issues, this chapter introduces the vehicle-in-virtual-environment (VVE) method as an intermediate step before public road deployment. The VVE method offers the realism of actual vehicle dynamics combined with the safety of a controlled testing environment. It also enables the safe testing of rare traffic scenarios without requiring extended periods of public road testing, as these scenarios can easily be replicated in a virtual environment.

One area where the VVE method is particularly advantageous is in vehicle-to-pedestrian (V2P) communication for enhancing Vulnerable Road User (VRU) safety. V2P communication involves the exchange of information between vehicles and pedestrians to promote traffic safety. Through various communication protocols, information can be transmitted between vehicles and mobile devices carried by pedestrians, such as cell phones.

This enables drivers to anticipate the presence of pedestrians in advance, allowing them to take preventive actions and avoid potential traffic accidents.

5.2 Methodology

5.2.1 VVE System Structure

The VVE architecture is illustrated in Figure 5.1. The real vehicle operates in a secure, open area like a parking lot, with its movements being synchronized with a virtual vehicle in a highly realistic, 3D-rendered virtual environment that can be easily adjusted. The real vehicle's onboard sensors are replaced by virtual sensor inputs generated from the virtual environment, reflecting the perspective of the synchronized virtual vehicle. This setup allows the control unit on the real vehicle to respond to traffic scenarios within the virtual environment, calculating control commands accordingly. These commands are then executed by the real vehicle, with its movements being mirrored back into the virtual vehicle, thus closing the feedback loop.

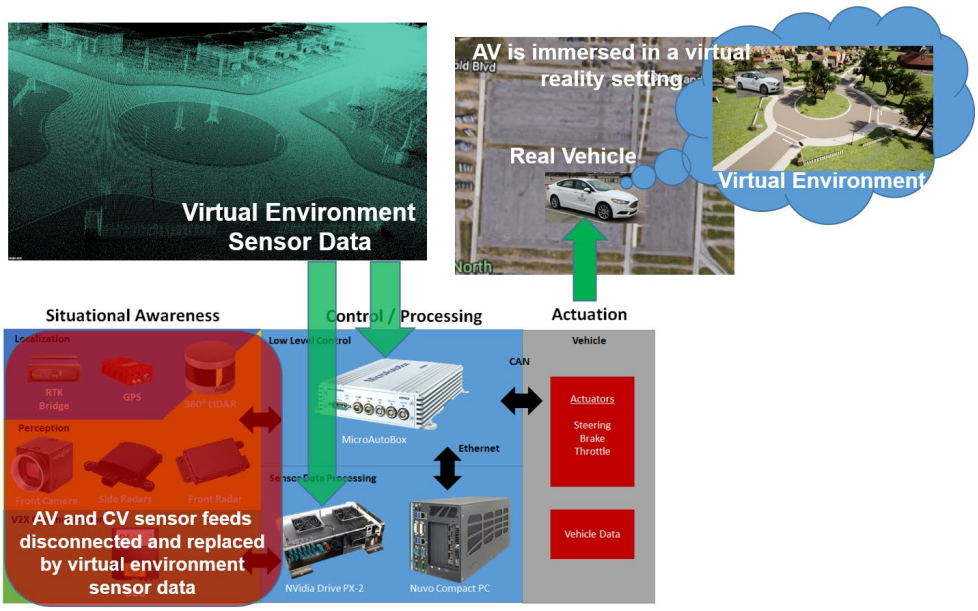


Figure 5.1: Illustration of VVE Architecture [44].

Our current implementation architecture of the VVE system is demonstrated in Figure 5.2 with the components onboard our test vehicle shown in Figure 5.3. The real vehicle is equipped with an RTK GPS unit (OxTS xNAV500) that includes differential antennas, providing accurate position and heading information even when the vehicle is stationary. This data is received and processed by the onboard control unit, the dSpace microautobox (MABX) electronic control unit, and then transmitted to the onboard simulation computer which runs a CARLA-based virtual environment built on the Unreal Engine, via the Ethernet UDP Protocol.

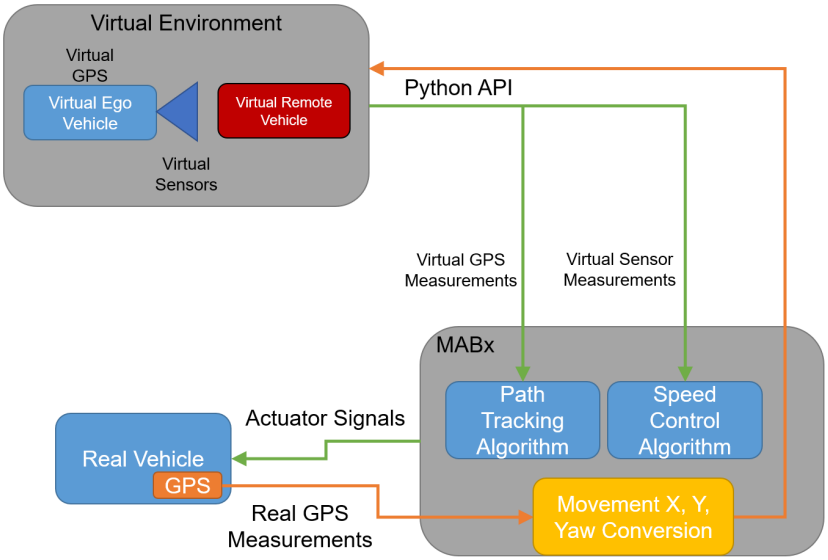


Figure 5.2: Implemented VVE Architecture [44].

The virtual environment in our implementation replicates the Linden area in Columbus, Ohio, where a recent autonomous shuttle deployment occurred [45]. Upon receiving the real GPS data, the virtual environment performs frame transformation and conversion operations to synchronize the movements of the virtual vehicle with those of the real vehicle. Virtual sensors within the environment gather measurements and transmit this data back to the MABX electronic control unit via Ethernet UDP Protocol. It's important to note that data from virtual perception sensors such as radar and Lidar are transmitted directly to the MABX electronic control unit while data from virtual localization sensors like GPS positions require inverse frame transformation and conversion operations before being

sent back. The MABX electronic control unit then uses this information to generate actuator signals which control the real vehicle via a CAN bus connection. The real-virtual frame transformation and conversion operations are crucial for ensuring precise motion synchronization between the real and virtual vehicles and will be discussed in detail in a separate section.

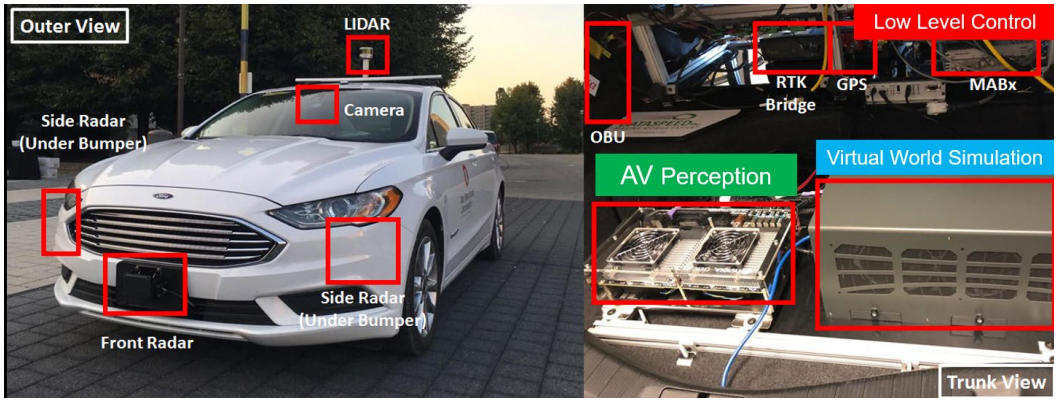


Figure 5.3: AV used for VVE implementation [44].

5.2.2 Frame Transformation/Conversion for Motion Synchronization

As highlighted in the previous section, frame transformation and conversion operations are essential for achieving accurate motion synchronization between the real and virtual vehicles. Therefore, this section outlines the procedure we used to accurately represent real-world motions within the virtual environment. Motion synchronization generally begins at a specific time, referred to as the 'reset time,' during which the virtual vehicle's location and heading are initialized to a predetermined set of values, and the real vehicle's location and heading are recorded for future reference. As the real vehicle moves, the changes in its location and heading relative to the reset time are subjected to a series of frame transformation and conversion operations. These operations adjust the real-world changes so that they are accurately represented in the virtual environment's frame of reference, thereby achieving motion synchronization. It is important to note that once the position of the vehicle's center of gravity and its heading are determined through the frame transformation and conversion process, the coordinates of the entire vehicle body can be

reconstructed accordingly. The frame transformation and conversion process involve several components, each of which will be detailed in the following paragraphs.

The first component of the process is a frame conversion operation illustrated in Figure 5.4. The OxTS GPS in the vehicle used provides the longitude, latitude and heading angle data in degrees, where the heading angle is recorded in the range of -180 deg and 180 deg. Converting the longitude and latitude data into unit of meter, one can construct a coordinate frame, namely the real vehicle (F_r) frame, as displayed in the left-hand side of Figure 5.4. It should be noted that the positive X-axis in this frame is assumed to be aligned with positive latitude direction, where zero heading angle corresponds to. It should also be noted that the positive heading angle in this frame is defined as clockwise (CW) in correspondence to the heading angle outputted by the OxTS GPS unit. In order to accommodate frame transformation operations in other components of the process, a new frame named GPS (F_g) frame is constructed as displayed in the right-hand side of Figure 5.4. In this F_g frame, zero heading is defined to be along the positive X-axis and positive heading angle is in the counterclockwise (CCW) direction. The frame conversion can thus be carried out as follows:

$$\begin{cases} X_g = X_r \\ Y_g = -Y_r \\ \psi_g = -mod(\psi_r + 360, 360) \end{cases} \tag{5.1}$$

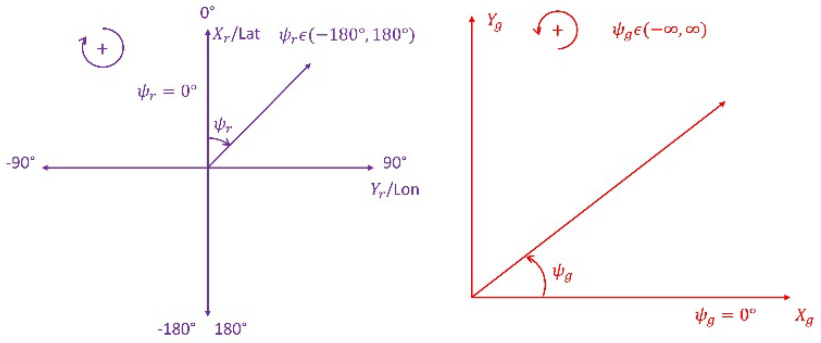


Figure 5.4: Frame transformation/conversion component 1: conversion between F_r frame and F_g frame [45].

The second component of the process is a frame transformation operation illustrated in Figure 5.5. The F_g frame is the same as described in Figure 5.4. At reset time t_0, an

intermediate (F) frame is created at the vehicle location with positive X-axis pointing at zero heading direction and positive heading defined in the CCW direction. As the vehicle moves to a new location at time t_1, its location and heading can be represented in the F frame as illustrated in Equation 5.2. It should be noted that set $(X_{g0}, Y_{g0}, \psi_{g0})$ and set $(X_{g1}, Y_{g1}, \psi_{g1})$ can be obtained by using Equation 5.1 on vehicle states at the reset time and at current time, respectively.

$$\begin{cases} \psi_1 = \psi_0 + \psi_{g1} - \psi_{g0} \\ \begin{bmatrix} X_1 \\ Y_1 \end{bmatrix} = \begin{bmatrix} X_0 \\ Y_0 \end{bmatrix} + R_{Fg \to F} \begin{bmatrix} X_{g1} - X_{g0} \\ Y_{g1} - Y_{g0} \end{bmatrix} \end{cases} \tag{5.2}$$

where: $R_{Fg \to F} = \begin{bmatrix} cos\,(\psi_{g0}) & sin(\psi_{g0}) \\ -sin\,(\psi_{g0}) & cos(\psi_{g0}) \end{bmatrix}$.

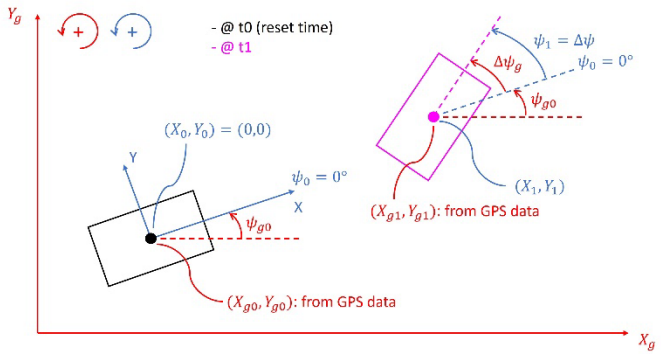


Figure 5.5: Frame transformation/conversion component transformation between F_g frame and F frame [45].

The third component of the process is another frame conversion operation displayed in Figure 5.6. The coordinate frame used in the virtual environment, named as the F_c frame, is shown on the right-hand side of Figure 5.6, where the initial location and heading of the virtual vehicle is defined at reset time t_0. An additional coordinate frame, named the vehicle (F_v) frame, is defined as shown on the left-hand side of Figure 5.6, with zero heading pointing

in the positive X-axis direction and positive heading angle in the CCW direction. The frame transformation operation can thus be carried out as follows:

$$\begin{cases} X_c = X_v \\ Y_c = -Y_v \\ \psi_c = -\psi_v \end{cases} \tag{5.3}$$

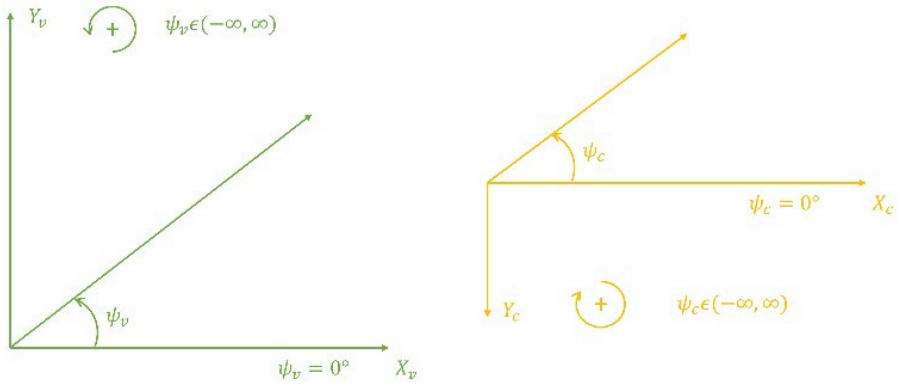


Figure 5.6: Frame transformation/conversion component 3: conversion between F_v frame and F_c frame [45].

The final component of the process is illustrated in Figure 5.7. This frame transformation procedure connects the previously discussed intermediate F frame and vehicle F_v frame. The location and heading of the vehicle at time t_1 can be represented in the F_v frame as illustrated in Equation 5.4. It should be noted that the set $(X_{v0}, Y_{v0}, \psi_{v0})$ can be obtained by using Equation 5.3 on the initial position and heading of the virtual vehicle defined in the Fc frame. Once $(X_{v1}, Y_{v1}, \psi_{v1})$ set is obtained from Equation 5.4, Equation 5.3 can be applied again to represent the current vehicle location and heading information in the F_c frame, which will complete the motion synchronization procedure.

$$\begin{cases} \psi_{v1} = \psi_{v0} + \psi_1 - \psi_0 \\ \begin{bmatrix} X_{v1} \\ Y_{v1} \end{bmatrix} = \begin{bmatrix} X_{v0} \\ Y_{v0} \end{bmatrix} + R_{F \to Fv} \begin{bmatrix} X_1 - X_0 \\ Y_1 - Y_0 \end{bmatrix} \end{cases} \tag{5.4}$$

where: $R_{F\to Fv} = \begin{bmatrix} cos\,(\psi_{v0}) & -sin(\psi_{v0}) \\ sin\,(\psi_{v0}) & cos(\psi_{v0}) \end{bmatrix}$.

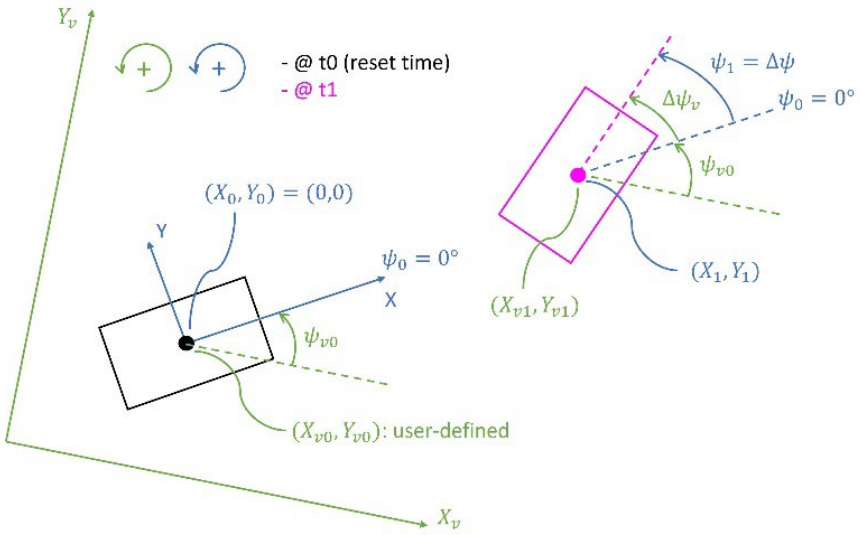


Figure 5.7: Frame transformation/conversion component transformation between F frame and F_v frame [45].

5.2.3 V2P System Structure

To illustrate the multi-actor capabilities of the proposed VVE approach, a Vehicle-to-VRU communication is implemented. The system architecture is displayed in Figure 5.8. The pedestrian carries a mobile phone equipped with onboard IMU and GPS sensors. A mobile application, adapted from [46], is implemented on the phone, where the GPS location and heading of the pedestrian are broadcasted as part of personal safety messages (PSM) via Bluetooth low-energy (BLE) connection. A Bluetooth reception board listens to the PSM and transmits pedestrian motion information to the virtual environment via Ethernet connection between the board and the simulation computer onboard the real vehicle. Frame transformation/conversion operations like those described in the previous section are carried out in the virtual environment to synchronize the virtual pedestrian motions to the real pedestrian motions. With this architecture, it is possible to initialize virtual pedestrian position and orientation in the virtual environment such that traffic testing scenarios can be

constructed, while the real pedestrian operates in a safe space out of the harm's way, ensuring the safety of the test. Figure 5.9 demonstrates one of such possible test setups. Figure 5.10 presents the information flow of our VVE V2P implementation.

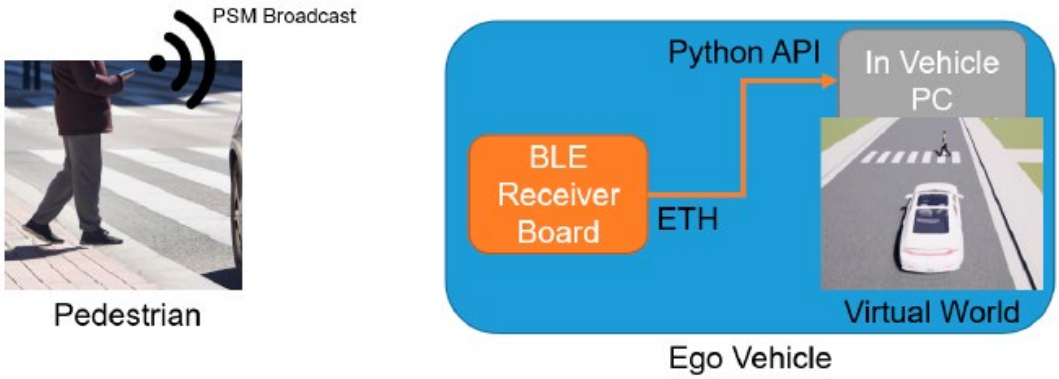


Figure 5.8: Implemented Vehicle-to-VRU architecture [44].

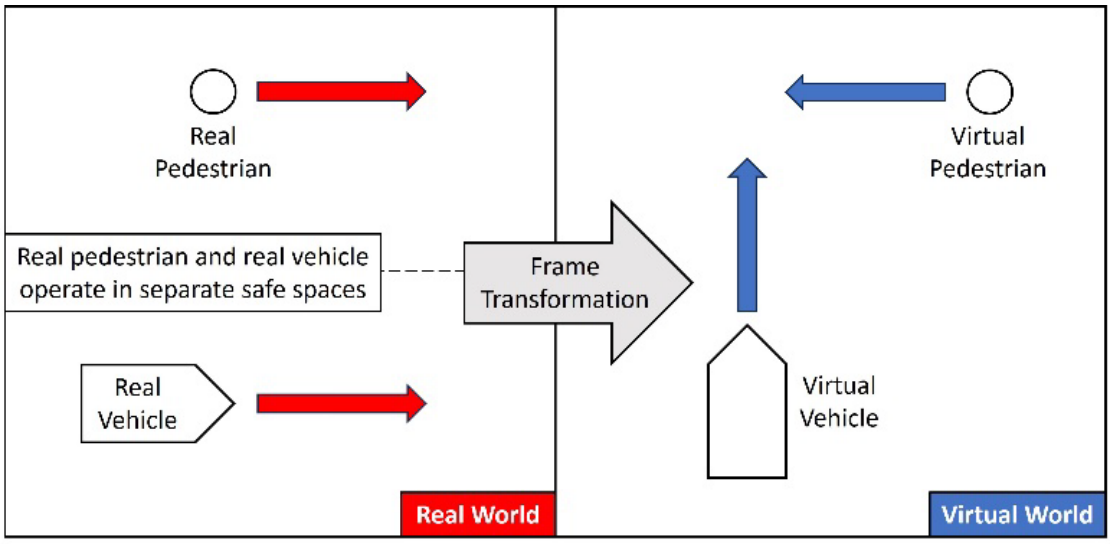


Figure 5.9: One possible V2P test setup with VVE method. The arrow directions represent the actors' directions of travel in the real (red) and the virtual (blue) world [45].

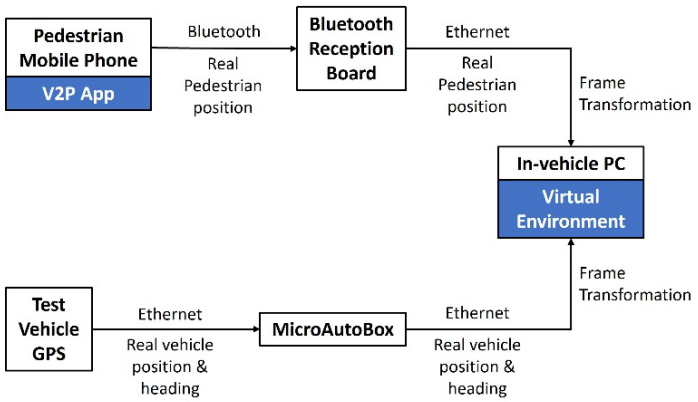


Figure 5.10: VVE V2P information flow [45].

5.3 Results

The capability of real-virtual vehicle motion synchronization using the proposed VVE method was tested with the setup shown in Figure 5.11. In this setup, a screen inside the vehicle is connected to the simulation computer, displaying the virtual vehicle's motion within the virtual environment while the real test vehicle operates in an open parking lot. The results of this testing are presented in Figure 5.12, where the motion trajectory of the real vehicle is plotted on the real-world coordinate frame, and the corresponding virtual vehicle's motion trajectory is plotted in the virtual environment. The results demonstrate that satisfactory synchronization between the real-world and virtual-world motions has been achieved.

Figure 5.11: VVE real-virtual motion synchronization [45].

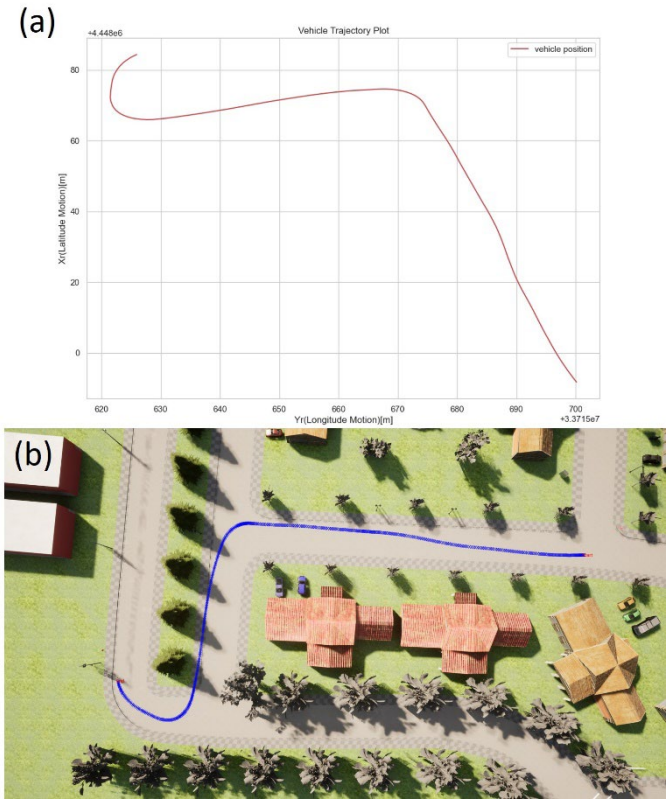


Figure 5.12: VVE motion synchronization test results: (a) Motion trajectory in the real world; (b) Motion trajectory in the virtual world [45].

To demonstrate the Vehicle-to-VRU communication capability with the VVE method, the test is conducted with the setup displayed in Figure 5.14. The virtual pedestrian is spawned in the virtual environment and the initial position and orientation is set in the virtual environment in relation to the pose of the virtual vehicle. The real pedestrian walks in the vicinity of the real test vehicle while holding the mobile phone with the V2P safety application enabled. The position and heading of the real pedestrian are sent to the virtual environment via Bluetooth connection and after frame transformation/conversion operations, the location and heading of the virtual pedestrian are updated accordingly. It should be noted that the virtual pedestrian location and heading can be easily reset in the virtual environment, so that the real pedestrian can operate at a safe distance from the test vehicle and in a direction that does not cross path with the test vehicle. Such capability is illustrated in Figure 5.15 where the real pedestrian is walking behind the test vehicle while

the virtual pedestrian is walking in front of the virtual vehicle.

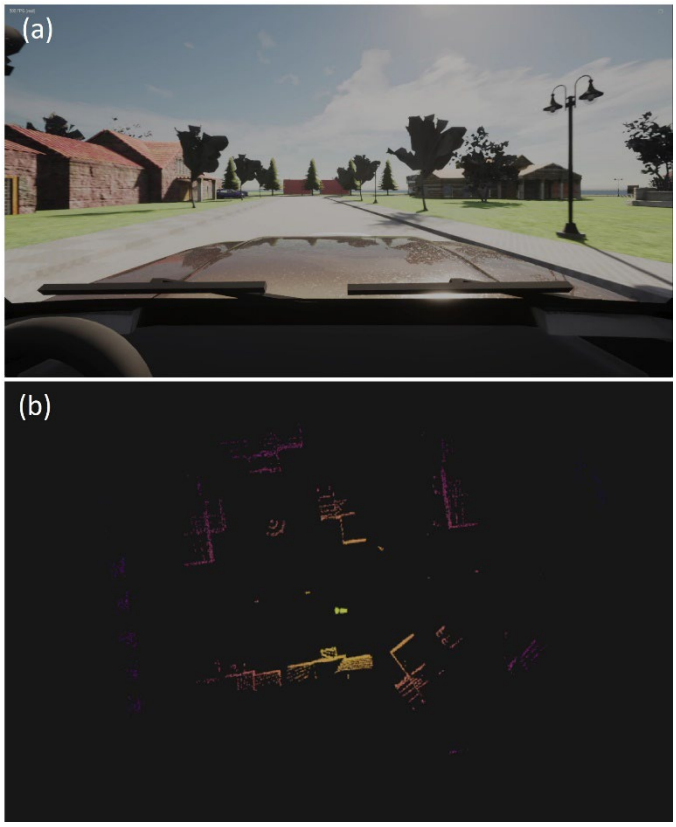


Figure 5.13: VVE virtual sensor data: (a) RGB camera; (b) LiDAR (figure not to scale) [45].

5.4 Conclusions

This chapter introduced a novel approach called Vehicle-in-Virtual-Environment (VVE) for the safe, efficient, and cost-effective development, demonstration, and evaluation of ADAS and Connected Autonomous Driving (CAD) systems. The VVE method synchronizes the movements of a real test vehicle, operating in a secure environment, with those of a virtual vehicle embedded in a realistic 3D virtual environment. This setup allows for the easy creation and testing of various traffic scenarios. The paper details the implementation of this method with a focus on the frame transformation and conversion operations that ensure precise motion synchronization. Additionally, a Vehicle-to-Vulnerable Road User (VRU) communication setup is implemented to showcase the VVE method's capability to handle

multi-actor interactions.

Figure 5.14: V2P connectivity in VVE test setup [45].

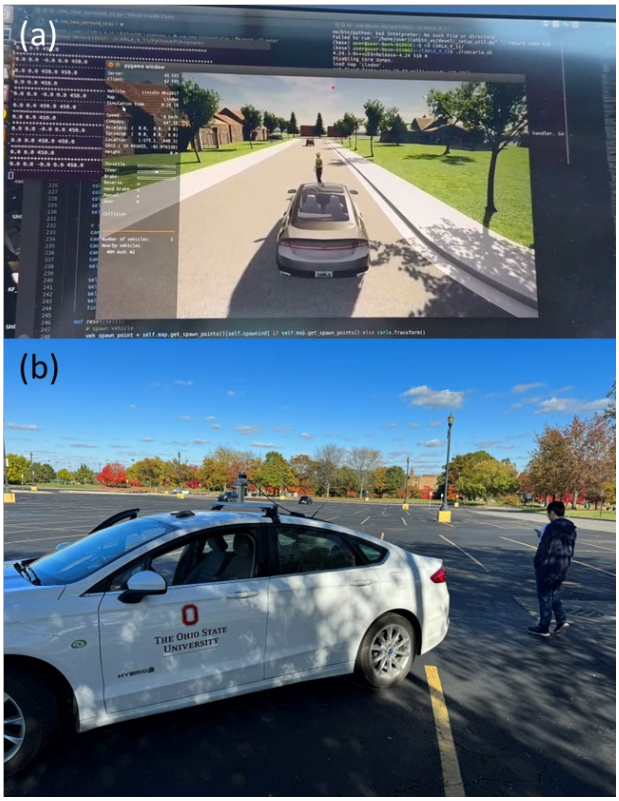


Figure 5.15: Safe operation of V2P connectivity test with (a) Virtual pedestrian walks in front of the virtual vehicle; (b) Real pedestrian walks behind the real test vehicle [45].

Chapter 6

Future Work

In conclusion, the rapid urbanization and increase in privately owned vehicles have exacerbated traffic congestion and car accidents, posing significant challenges for modern cities. Autonomous driving systems (ADS) offer a promising solution to mitigate these issues by reducing the human errors responsible for a substantial number of accidents. However, the current methods for testing and evaluating these systems, including Model-in-the-Loop (MIL) and Hardware-in-the-Loop (HIL) simulations, are limited and can lead to unsafe public road deployments, eroding public trust. To address these challenges, the Vehicle-in-Virtual-Environment (VVE) method is introduced as a more effective approach. This method allows for safe, efficient, and realistic testing of autonomous driving functions by synchronizing real vehicle motions with virtual scenarios.

In the first year of our project, we focused primarily on using the VVE method to enhance pedestrian safety. Our research began with the analysis and simulation of key pedestrian crash scenarios identified by the Fatality Analysis Reporting System (FARS). We then implemented RGB camera-based detection of vulnerable road users (VRUs) to accurately identify pedestrians around the vehicle during navigation. A Disturbance Observer (DOB)-integrated PID path tracking controller was also introduced for precise and efficient path tracking. Additionally, we developed a deep reinforcement learning (DRL)-based collision avoidance framework to handle emergency situations. The VVE method was then employed to test and evaluate these autonomous vehicle functions related to pedestrian safety.

Looking ahead to the second year, our focus will shift to improving bicyclist safety. Compared to pedestrians, bicyclists move at higher speeds and are more prone to collisions, necessitating even greater caution in our approach. This year, we reviewed extensive literature on Lidar data processing and trajectory prediction for VRUs, laying the groundwork for our upcoming efforts. In the second year of the project, we will work on

implementing these findings. Also, we will have a particular focus on developing robust and delay-tolerant trajectory control. This will involve the VVE method for testing and evaluation, ensuring that the calculated collision-free vehicle trajectories—whether they require slowing down, braking, or steering—are executed with precision and safety.

References

[1] World Health Organization, *Global status report on road safety 2015*. Geneva: World Health Organization, 2015. Accessed: Oct. 24, 2023. [Online]. Available: https://iris.who.int/handle/10665/189242

[2] "813428.pdf." Accessed: Aug. 25, 2024. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/813428

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," Oct. 22, 2014, *arXiv*: arXiv:1311.2524. doi: 10.48550/arXiv.1311.2524.

[4] R. Girshick, "Fast R-CNN," Sep. 27, 2015, *arXiv*: arXiv:1504.08083. doi: 10.48550/arXiv.1504.08083.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," Jan. 06, 2016, *arXiv*: arXiv:1506.01497. doi: 10.48550/arXiv.1506.01497.

[6] P. Sun *et al.*, "Sparse R-CNN: End-to-End Object Detection with Learnable Proposals," Apr. 26, 2021, *arXiv*: arXiv:2011.12450. doi: 10.48550/arXiv.2011.12450.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," May 09, 2016, *arXiv*: arXiv:1506.02640. doi: 10.48550/arXiv.1506.02640.

[8] Q. Liu, H. Ye, S. Wang, and Z. Xu, "YOLOv8-CB: Dense Pedestrian Detection Algorithm Based on In-Vehicle Camera," *Electronics*, vol. 13, no. 1, Art. no. 1, Jan. 2024, doi: 10.3390/electronics13010236.

[9] G. Jocher, A. Chaurasia, and J. Qiu, *Ultralytics YOLO*. (Jan. 2023). Python. Accessed: Aug. 26, 2024. [Online]. Available: https://github.com/ultralytics/ultralytics

[10] "CARLA - v20 carla_v20," Roboflow. Accessed: Aug. 26, 2024. [Online]. Available: https://universe.roboflow.com/alec-hantson-student-howest-be/carla-izloa/dataset/20

[11] S. Muhammad and G.-W. Kim, "Visual Object Detection Based LiDAR Point Cloud Classification," in *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Feb. 2020, pp. 438–440. doi: 10.1109/BigComp48618.2020.00-32.

[12] R. Sahba, A. Sahba, M. Jamshidi, and P. Rad, "3D Object Detection Based on LiDAR Data," in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, Oct. 2019, pp. 0511–0514. doi: 10.1109/UEMCON47517.2019.8993088.

[13] L. Liu, J. He, K. Ren, Z. Xiao, and Y. Hou, "A LiDAR–Camera Fusion 3D Object Detection Algorithm," *Information*, vol. 13, no. 4, Art. no. 4, Apr. 2022, doi: 10.3390/info13040169.

[14] A. Y. Naich and J. R. Carrión, "LiDAR-Based Intensity-Aware Outdoor 3D Object Detection," *Sensors*, vol. 24, no. 9, Art. no. 9, Jan. 2024, doi: 10.3390/s24092942.

[15] "A Spatio-Temporal Graph Transformer Network for Multi-Pedestrain Trajectory Prediction | IEEE Conference Publication | IEEE Xplore." Accessed: Aug. 29, 2024.

[Online]. Available: https://ieeexplore-ieee-org.proxy.lib.ohio-state.edu/document/9927798

[16] X. Cao and L. Guvenc, "Path Planning and Robust Path Tracking Control of an Automated Parallel Parking Maneuver," SAE International, Warrendale, PA, SAE Technical Paper 2024-01–2558, Apr. 2024. doi: 10.4271/2024-01-2558.

[17] "Autonomous Road Vehicle Path Planning and Tracking Control | IEEE eBooks | IEEE Xplore." Accessed: Oct. 24, 2023. [Online]. Available: https://ieeexplore.ieee.org/book/9645932

[18] L. Guvenc, B. Aksun-Guvenc, B. Demirel, and M. T. Emirler, *Control of Mechatronic Systems*. IET Digital Library, 2017. doi: 10.1049/PBCE104E.

[19] H. Wang, A. Tota, B. Aksun-Guvenc, and L. Guvenc, "Real time implementation of socially acceptable collision avoidance of a low speed autonomous shuttle using the elastic band method," *Mechatronics*, vol. 50, pp. 341–355, Apr. 2018, doi: 10.1016/j.mechatronics.2017.11.009.

[20] M. Morsali, E. Frisk, and J. Åslund, "Spatio-Temporal Planning in Multi-Vehicle Scenarios for Autonomous Vehicle Using Support Vector Machines," *IEEE Trans. Intell. Veh.*, vol. 6, no. 4, pp. 611–621, Dec. 2021, doi: 10.1109/TIV.2020.3042087.

[21] S. Zhu, "Path Planning and Robust Control of Autonomous Vehicles," Ph.D., The Ohio State University, United States -- Ohio, 2020. Accessed: Oct. 24, 2023. [Online]. Available: https://www.proquest.com/docview/2612075055/abstract/73982D6BAE3D419APQ/1

[22] G. Chen *et al.*, "Emergency Obstacle Avoidance Trajectory Planning Method of Intelligent Vehicles Based on Improved Hybrid A*," *SAE Int. J. Veh. Dyn. Stab. NVH*, vol. 8, no. 1, pp. 10-08-01–0001, Nov. 2023, doi: 10.4271/10-08-01-0001.

[23] W. B. Knox, A. Allievi, H. Banzhaf, F. Schmitt, and P. Stone, "Reward (Mis)design for autonomous driving," *Artif. Intell.*, vol. 316, p. 103829, Mar. 2023, doi: 10.1016/j.artint.2022.103829.

[24] B. R. Kiran *et al.*, "Deep Reinforcement Learning for Autonomous Driving: A Survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4909–4926, Jun. 2022, doi: 10.1109/TITS.2021.3054625.

[25] F. Ye, S. Zhang, P. Wang, and C.-Y. Chan, "A Survey of Deep Reinforcement Learning Algorithms for Motion Planning and Control of Autonomous Vehicles," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, Jul. 2021, pp. 1073–1080. doi: 10.1109/IV48863.2021.9575880.

[26] Z. Zhu and H. Zhao, "A Survey of Deep RL and IL for Autonomous Driving Policy Learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 14043–14065, Sep. 2022, doi: 10.1109/TITS.2021.3134702.

[27] A. Kendall *et al.*, "Learning to Drive in a Day," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 8248–8254. doi: 10.1109/ICRA.2019.8793742.

[28] E. Yurtsever, L. Capito, K. Redmill, and U. Ozgune, "Integrating Deep Reinforcement Learning with Model-based Path Planners for Automated Driving," in *2020 IEEE*

Intelligent Vehicles Symposium (IV), Oct. 2020, pp. 1311–1316. doi: 10.1109/IV47402.2020.9304735.

[29] A. Aksjonov and V. Kyrki, "A Safety-Critical Decision-Making and Control Framework Combining Machine-Learning-Based and Rule-Based Algorithms," *SAE Int. J. Veh. Dyn. Stab. NVH*, vol. 7, no. 3, pp. 10-07-03–0018, Jun. 2023, doi: 10.4271/10-07-03-0018.

[30] "Deep reinforcement-learning-based driving policy for autonomous road vehicles - Makantasis - 2020 - IET Intelligent Transport Systems - Wiley Online Library." Accessed: Oct. 24, 2023. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/iet-its.2019.0249

[31] S. Nageshrao, H. E. Tseng, and D. Filev, "Autonomous Highway Driving using Deep Reinforcement Learning," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, Oct. 2019, pp. 2326–2331. doi: 10.1109/SMC.2019.8914621.

[32] B. Peng *et al.*, "End-to-End Autonomous Driving Through Dueling Double Deep Q-Network," *Automot. Innov.*, vol. 4, no. 3, pp. 328–337, Aug. 2021, doi: 10.1007/s42154-021-00151-3.

[33] M. Jaritz, R. de Charette, M. Toromanoff, E. Perot, and F. Nashashibi, "End-to-End Race Driving with Deep Reinforcement Learning," Aug. 31, 2018, *arXiv*: arXiv:1807.02371. doi: 10.48550/arXiv.1807.02371.

[34] F. Merola, F. Falchi, C. Gennaro, and M. Di Benedetto, "Reinforced Damage Minimization in Critical Events for Self-driving Vehicles:," in *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, Online Streaming, --- Select a Country ---: SCITEPRESS - Science and Technology Publications, 2022, pp. 258–266. doi: 10.5220/0010908000003124.

[35] Z. Cao *et al.*, "Reinforcement Learning based Control of Imitative Policies for Near-Accident Driving," Jun. 30, 2020, *arXiv*: arXiv:2007.00178. doi: 10.48550/arXiv.2007.00178.

[36] H. Chen, X. Cao, L. Guvenc, and B. Aksun-Guvenc, "Deep-Reinforcement-Learning-Based Collision Avoidance of Autonomous Driving System for Vulnerable Road User Safety," *Electronics*, vol. 13, no. 10, Art. no. 10, Jan. 2024, doi: 10.3390/electronics13101952.

[37] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-Based Collision Avoidance," *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 3, pp. 972–983, May 2021, doi: 10.1109/TCST.2019.2949540.

[38] L. Pariota *et al.*, "Integrating tools for an effective testing of connected and automated vehicles technologies," *IET Intell. Transp. Syst.*, vol. 14, no. 9, pp. 1025–1033, 2020, doi: 10.1049/iet-its.2019.0678.

[39] S. Chen, N.-N. Zheng, Y. Chen, and S. Zhang, "A Novel Integrated Simulation and Testing Platform for Self-Driving Cars With Hardware in the Loop," *IEEE Trans. Intell. Veh.*, vol. PP, pp. 1–1, May 2019, doi: 10.1109/TIV.2019.2919470.

[40] Ş. Y. Gelbal, S. Tamilarasan, M. R. Cantaş, L. Guvenc, and B. Aksun-Guvenc, "A connected and autonomous vehicle hardware-in-the-loop simulator for developing automated driving algorithms," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct. 2017, pp. 3397–3402. doi: 10.1109/SMC.2017.8123155.

[41] M. R. Cantas, S. Gelbal, L. Guvenc, and B. Aksun-Guvenc, *Cooperative Adaptive Cruise*

Control Design and Implementation. 2023.

[42] M. T. Emırler *et al.*, "Lateral stability control of fully electric vehicles," *Int. J. Automot. Technol.*, vol. 16, no. 2, pp. 317–328, Apr. 2015, doi: 10.1007/s12239-015-0034-1.

[43] D. Ozcan, U. Sonmez, and L. Guvenc, "Optimisation of the Nonlinear Suspension Characteristics of a Light Commercial Vehicle," *Int. J. Veh. Technol.*, vol. 2013, pp. 1–16, Feb. 2013, doi: 10.1155/2013/562424.

[44] X. Cao, H. Chen, S. Y. Gelbal, B. Aksun-Guvenc, and L. Guvenc, "Vehicle-in-Virtual-Environment (VVE) Method for Autonomous Driving System Development, Evaluation and Demonstration," *Sensors*, vol. 23, no. 11, Art. no. 11, Jan. 2023, doi: 10.3390/s23115088.

[45] X. Cao, H. Chen, S. Y. Gelbal, B. A. Guvenc, and L. Guvenc, "Vehicle-in-Virtual-Environment Method for ADAS and Connected and Automated Driving Function Development, Demonstration and Evaluation," SAE International, Warrendale, PA, SAE Technical Paper 2024-01–1967.

[46] S. Y. Gelbal, M. R. Cantas, B. A. Guvenc, L. Guvenc, G. Surnilla, and H. Zhang, "Mobile Safety Application for Pedestrians Utilizing P2V Communication over Bluetooth," SAE International, Warrendale, PA, SAE Technical Paper 2022-01–0155, Mar. 2022. doi: 10.4271/2022-01-0155.

1. Report No. .	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Vehicle-in-Virtual-Environment (VVE) Method for Developing and Evaluating VRU Safety of Connected and Autonomous Driving		**5. Report Date** July 31, 2024 **6. Performing Organization Code** .
7. Author(s) Haochong Chen. (https://orcid.org/0009-0000-5461-0822) Xincheng Cao. (https://orcid.org/0009-0008-2525-9031) Bilin Aksun-Guvenc, Ph.D. (https://orcid.org/0000-0003-0836-9286) Levent Guvenc, Ph.D. (https://orcid.org/0000-0001-8823-1820)		**8. Performing Organization Report No.** .
9. Performing Organization Name and Address Automated Driving Lab Ohio State University 1320 Kinnear Rd, 43212, OH		**10. Work Unit No.** **11. Contract or Grant No.** Federal Grant No. 69A3552344811
12. Sponsoring Agency Name and Address Safety21 University Transportation Center Carnegie Mellon University 5000 Forbes Avenue Pittsburgh, PA 15213		**13. Type of Report and Period Covered** Final Report (July 1, 2023-June 30, 2024) **14. Sponsoring Agency Code** USDOT
15. Supplementary Notes Conducted in cooperation with the U.S. Department of Transportation. .		

16. Abstract
This report focuses on pedestrian safety and starts by considering five FARS pedestrian crash scenario use cases. A hybrid pedestrian safety system that switches from a conventional path following mode to double deep reinforcement learning pedestrian collision avoidance maneuvering after detection of collision risk interactions with nearby pedestrians is developed and demonstrated using simulations in a virtual environment. The Vehicle-in-Virtual-Environment (VVE) method is further developed to test the considered use cases with moving real vehicle(s) and real pedestrian(s) in a safe and repeatable manner. The VVE method developed is demonstrated using the pedestrian crossing roadway – vehicle not turning case with vehicle-to-pedestrian communication using mobile phones to convey pedestrian motion to the vehicle. Results will be extended to bicyclist safety in future work

17. Key Words	18. Distribution Statement
vulnerable road users, pedestrian safety, crash avoidance systems, deep reinforcement learning, vehicle-in-virtual-environment	No restrictions.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	65	

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized