CHALMERS
UNIVERSITY OF TECHNOLOGY

Unity 3D Environment for Vision in the Loop

Master's Thesis in MPCOM

Di Pan

Tiantong Chen

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Master's thesis 2019:EENX30

Unity 3D Environment for Vision in the Loop

Di Pan

Tiantong Chen

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2019

Unity 3D Environment for Vision in the Loop
Di Pan, Tiantong Chen

Supervisor: Siddhant Gupta, Volvo Car Corporation;
Francesco Costagliola, Volvo Car Corporation;
Advisor: Nima Hajiabdolrahim, Chalmers University of Technology;
Examiner: Fredrik Brännström, Chalmers University of Technology;

Master's Thesis 2019:EENX30
Department of Electrical Engineering
Communication Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Virtual image extracted from Unity showing the appearance of the built virtual scene.

Gothenburg, Sweden 2019

Unity 3D Environment for Vision in the Loop
Di Pan, Tiantong Chen
Department of Electrical Engineering
Chalmers University of Technology

Abstract

In this master thesis, an open-loop detection test is evaluated with Unity (a real-time 3D development platform) virtual images and SPAS (Simulation Platform for Active Safety, a virtual testing platform built in Matlab and Simulink) sensor data. The aim of this project is to figure out the possibility of producing real and detailed enough Unity virtual images for the open-loop detection test of the AD (Autonomous Driving) function.

The SPAS virtual scene is designed based on a real-world field test held by Volvo Cars. All the main information in SPAS virtual scene such as the host car's velocity and position as well as lane markings' position are set to be similar to the real world test environment. Unity virtual scene gets into construction once the SPAS virtual scene is built up with the capability of generating sensor data. Unity environment is set based on the SPAS setting to maintain the consistency between image data and sensor data. Then, the camera model is adjusted according to the real vehicle camera to generate photo-realism virtual images.

With Unity virtual image data and SPAS sensor data, open-loop detection software can provide a reliable object detection result. After validating the feasibility of this system, more variations could be made in Unity virtual scene. Variables of light conditions such as light intensity, light direction and shadow will be adjusted, and objects' color can also be changed. By comparing the detected output and the original setting, the effect of these variables could be figured out. The most important output parameters to analyze are the detected lane markings and objects' positions.

Keywords: Unity, Open-loop Test, SPAS, Virtual Scene, Object Detection

Acknowledgements

We would like to thank for the kind help and detailed technical support from the Active Safety Group in Volvo Cars Corporation, especially for our supervisor Siddhant Gupta, Francesco Costagliola, Kristoffer Helander – expert on V-ray, Mukesh Muralidharan – expert on virtual testing and Anders Ödblom. We would also like to appreciate our advisor in Chalmers, Nima Hajiabdolrahim, and our examiner Fredrik Brännström.

Di Pan, Tiantong Chen, Gothenburg, Jun 2019

Contents

List of Figures

List of Tables

Glossary

AD	Autonomous Driving
Ra-Cam	A sensor system with radar and camera
SPAS	Simulation Platform of Active Safety
VCC	Volvo Cars Corporation
Host Car	VCC AD test car
Real Log	Recorded data from VCC test car
Real Log Coordinate	Coordinate system of Real Log data
SPAS Coordinate	Coordinate system of SPAS data
Unity World Coordinate	Coordinate system of Unity
Longitude	Horizontal value on X axis in SPAS Coordinate
Latitude	Vertical value on Y axis in SPAS Coordinate
FOV	Field Of View
MSE	Mean Square Error

1
Introduction

This thesis is carried out based on the autonomous driving project of Volvo Cars Corporation. Volvo Cars Corporation is one of the top Swedish companies, focusing on the development of a safe driving environment.

Autonomous driving has been one of the most popular topics in recent years, with the goal of providing a safe and time effective driving. Volvo Cars Corporation also does many kinds of researches and developments for this long-term goal. To guarantee that autonomous driving function (abbreviated to AD function) is safe and reliable enough, extensive testings are necessary for all possible conditions. However, testing in the real environment usually has obvious limitations such as cost and efficiency, so a proper alternative is testing in the virtual scene instead. A photo-realism and well-detailed virtual environment can make the AD function show the same reaction as in the real world, but make it easier to test and develop the algorithm of the AD function.

Figure 1.1: Screenshot of Volvo previous work of V-Ray virtual scene

As depicted in Fig. 1.1, Volvo Cars Corporation has already done some re-simulation researches for the open-loop AD function test with virtual scenes. 'Re-simulation' here means the test is performed as a virtual simulation for the real field test. To be more specific, a virtual scene is built similarly to the real testing environment with software like V-Ray, then it will be used to test the same AD function in the

real field test. If the results of the virtual test and the real test are reasonable and similar, the re-simulation test has the potential to replace the real test. Thus, the AD function can be developed mainly based on the output of the re-simulation test, which could improve efficiency significantly.

The previous open-loop test result with V-Ray is satisfactory as the AD function performs well with the high-quality virtual images. As a ray tracing engine, V-Ray calculates the reflection and diffusion of each ray to get the final high-quality photo-realism virtual image, but this technique could also cause disadvantages such as time-inefficiency and over-detailed images for the AD function.

To avoid problems of V-Ray, a new method of fast virtual image processing is developed based on Unity software. The basic idea is trying to improve efficiency by sacrificing some unimportant details, which raises our thesis project. This thesis is trying to figure out whether Unity is able to provide detailed enough virtual images for the AD function. If so, more details about how the vision components affect the test results should be excavated. We first focus on building up the virtual scene in SPAS and Unity according to the real test output. Then the open-loop AD function is tested with the virtual images and sensor data extracted from Unity and SPAS. With a preliminary conclusion that Unity can take the duty of providing virtual images with enough quality, we made variations on the Unity virtual scene to obtain a rough impression of how different vision components affect the final test result.

1.1 Aim

The aim of this thesis is to develop a re-simulation tool chain based on Unity for the open-loop detection software and to evaluate whether Unity can provide virtual images with enough quality. Then the effect of different Unity factors on the detection result is examined.

1.2 Scope

The scope of this thesis is making open-loop detection test with the generated virtual input data. The input data is obtained from two platforms, so the virtual images extracted from Unity and the virtual sensor data generated by the SPAS platform should be matched. In other words, the pictures and the sensor data should depict the same environment and vehicle movement. The correspondence between them could be guaranteed since the virtual scene is built in both Unity and SPAS using the same real field test as a reference.

The open-loop detection test can be seen as a part of the algorithm separated from the AD function. It takes both virtual images and virtual sensor data as the input, then produces the detected object at the output. Due to the privacy policy of Volvo

Cars Corporation, the majority of the numerical values of the output are replaced by relative values.

1.3 Limitations

This thesis project mainly focuses on the possibility of using Unity virtual images in open-loop tests and the basic analysis about which factor will affect the detection output. Thus the algorithm and process of the open-loop detection or detailed quantitative analysis of the output is not involved. Moreover, creating a varying or more complex virtual scene is not included either.

1.4 Thesis Outline

The report is structured to show all aspects of the thesis project. Chapter 2 introduces the software and the platform involved in the project. The pre-processing for the original data is provided in Chapter 3. In Chapter 4 and Chapter 5, the methods of developing a virtual scene in SPAS and Unity are discussed. Chapter 6 illustrates the detection result of the open-loop test while Chapter 7 provides a preliminary discussion of how the detection result is influenced. Eventually, Chapter 8 offers the final conclusion and presents some suggestions for future works.

2
Background

To help readers understand the thesis project better, this chapter describes the software platforms and some basic concepts that are used in this thesis.

2.1 Volvo Host Car

In Volvo Cars Corporation (VCC), a huge amount of field tests have been done by using Volvo test vehicles to collect data for AD function. As a kit of sensors combining Radar and Camera, Ra-Cam[1] is equipped on the Volvo test vehicles to collect the scenario data during the whole period of driving. The word 'scenario' here refers to a traffic scenario, including all the stuff like vehicles, lane markings, traffic signs and pedestrians. The collected data can be categorized in two classes: image data obtained by vehicle camera as well as sensor data collected by radar and other sensors. The data collected from different sources are mixed together through sensor fusion before being processed by the AD function. The whole process is shown in Fig. 2.1.

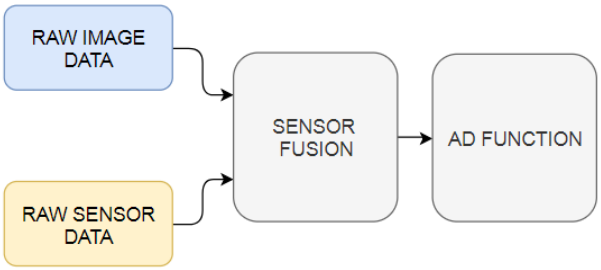

Figure 2.1: Basic structure of the real field test

2.2 SPAS[2]

SPAS is the abbreviation for 'Simulation Platform of Active Safety', which is a virtual simulation platform developed by Active Safety Group of Volvo Cars Corporation. As a Matlab/Simulink based platform, SPAS enables engineers to simulate real test driving in a virtual scene, which can be executed in both Windows and Linux environment [1]. The SPAS environment consists of models for vehicles and

[1] A short combination of Radar and Camera terms.
[2] Abbreviation for Simulation Platform of Active Safety

other traffic elements like lane markings and sensors on the car. In this thesis, SPAS is used to simulate the whole driving period and to generate the sensor data for the open-loop detection test.

2.3 Unity Game Engine

A game engine is a software-development environment designed for people to build video games. Unity is a cross-platform game engine developed by Unity Technologies, providing an easy way for both 2D and 3D virtual scene development [2]. Unity is chosen in this project as an alternative for V-Ray to build a less realistic virtual scene but with higher speed. Even though called 'less realistic', the virtual scene of Unity can still closely resemble the original real driving scenes [3]. Besides, Unity has a strong community offering many publicly available 'assets', which can be easily reused with simple adjustments. Moreover, Unity could easily control all vision components with *C#*, which makes it possible to simulate the movement during the whole period of driving and to extract virtual images automatically.

2.4 Open-Loop Test Software

The AD function could include various modules such as object detection, decision making and movement control. The open-loop test software used in this thesis can be interpreted as a detection module separated from the AD function. The basic structure of the software is shown in Fig. 2.1, which takes image data and sensor data as the input data to do the initial object detection. The image data shows the scene photographed by a camera on the test car, while sensor data contains vehicle movement information as well as radar information.

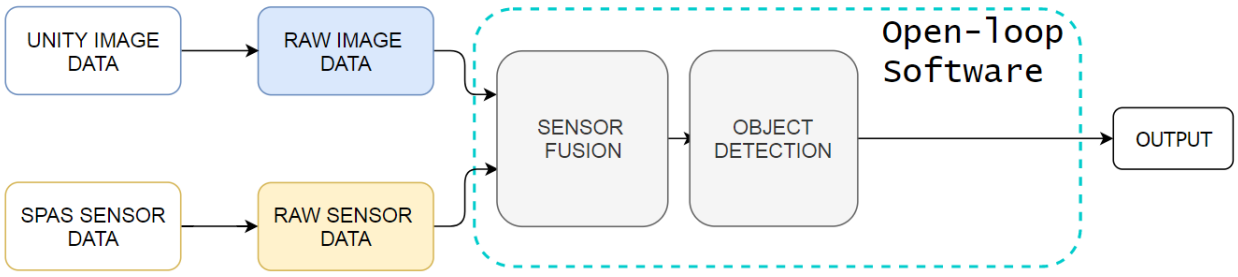

Figure 2.2: Structure of the Open-loop test

Figure. 2.2 illustrates the open-loop detection test's structure in this thesis project. The open-loop test software provides no feedback to modify the status of driving but only detects the objects. It is a third-party software and due to the secret policy, parts of the detection result such as numerical value and parameters' name are hidden.

3

Pre-processing Before Scene Setting

In this thesis, a segment of a field test video is chosen as the reference to build the virtual scene. This chapter briefly describes the chosen reference for the virtual scene development and the data extracted from this reference.

3.1 Real Log

This thesis aims to validate the feasibility of the system shown in Fig. 2.2, so it would be enough to just consider a simple scenario in the beginning. Although a virtual scene can be set arbitrarily, it would be better to set it similar to a real testing environment, since it could make the final result more reasonable and reliable. To achieve that, it is essential to obtain the original numerical value of the field test area such as lane width, lane markings' position, and the distance to targets. The original value is also called ground truth value, which means exactly accurate data of the environment. However, it is impossible to get the ground truth value of the real-world, so an alternative way would be setting the virtual scenario according to the detection output of the real field test. Therefore, the pre-processing is necessary before all the scene settings.

One real field test usually records a long driving period. Since a longer time period means a more complicated scenario, only a segment of a field test is chosen as the reference for virtual scenario settings. In this segment, the host car is slowly heading to a parked target car along a straight road and eventually brakes in front of the target car. The field test was held on a closed test road, so there was no other vehicle but only the parked target car. Besides, there are no pedestrians or traffic signs in this test area, and all the surroundings are mainly covered by trees and bushes. This simplified test environment reduces a great deal of calculation work of the AD function as it mainly needs to consider lane markings and the parked target car.

In the real field test, the host car is VCC's test car, equipped with a Ra-Cam system to record the whole driving process. The process output is named as Real Log, which contains the video log recorded by the camera, the raw numerical data collected by other sensors and detected result after sensor fusion and AD function. Screenshots from the Real Log depicted in Fig. 3.1 intuitively illustrate the field

test environment.

Figure 3.1: Screenshots of Real Log from the chosen field test

3.2 Data Extraction

Volvo Car Corporation has already obtained its own AD function detection result (Real Log) for this simple real filed test, and a segment of the Real Log is used as the reference of virtual scene development. The segment contains not only the host car's movement information but also the environment data produced by AD function with sensor fusion. Several parameters of this data could be used to approximate the ground truth value of the chosen field test. This section mainly describes which kinds of parameters are extracted from the Real Log for future steps as well as their included information:

- **Real Log Coordinate**
  Real Log Coordinate is the coordinate system for all stored data. The Real Log Coordinate is built based on the position and posture of the host car at the initial time of the chosen test segment. The initial position of the host car's centre is considered as the origin, while X, Y, Z axes point to the front, left and up side of the host car respectively, based on the right-hand rule.

- **Road Information**
  Road information extracted from the Real Log contains lane markings' positions, indicating the distance from the host car centre to both the left and the right adjacent lane markings. Road information also includes the lane width and the host car's relative position within the lane. In the selected Real Log segment, the average distance between the left lane marking and the host car is 1.6m, while the right lane marking is located at the average distance of 1.9 m from the host car. The recorded data has fluctuation around the average, since there is unavoidable detection noise and the real-world lane markings or driving path is not perfectly straight.

- **Vehicle Information**
  Vehicle information extracted from the Real Log contains position and velocity information of the host car as well as distance information for the parked target car. Position and velocity information is stored with the format of three-dimension arrays, consisting of X, Y and Z values along each axis of

the Real Log Coordinate. For the parked target car, there is no ground truth value to describe its position. Instead, only detected relative distance from the host car to the target car, which is obtained by the AD function detection, is available.

- **Camera Posture**
  Considering the movement of the host car and the unevenness of the road in the real world, the host car camera cannot be perfectly stable. In fact, the camera's posture will experience slight random fluctuation during the driving period. This posture changing is recorded as a relative rotation matrix to the previous camera posture in the Real Log Coordinate. To be more specific, the camera is no longer oriented to the exact 'front' side of the initial posture but suffered a slight erratic rotation with a small angle. This erratic rotation of the camera in the 3D space could be separated into three independent rotations along X, Y and Z axes of the Real Log Coordinate [4].

  For a clockwise rotation θ along each axis, the rotation matrices can be written as numerical formulas below [4], where R_X, R_Y and R_Z refer to the rotations along X, Y and Z axes, respectively:

$$R_X(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{3.1}$$

$$R_Y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{3.2}$$

$$R_Z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

  As mentioned above, a random rotation can be seen as three separated rotations along three axes. The final combined rotation matrix could be expressed by matrix multiplication:

$$R = R_Z(\gamma)R_Y(\beta)R_X(\alpha). \tag{3.4}$$

  Here α, β and γ are Euler Angles which stand for the rotation angles along X, Y and Z axes. The Camera Posture information here is the final combined rotation matrix R, which can be used to reconstruct a more vivid host car behaviour in next steps.

- **Feature Points**
  Feature points are marks of relative important positions in the 3D space to help recognize and locate objects, which are mainly located around the contour of the surroundings or boundary between the bright area and the dark area, but it is hard to determine what each point exactly stands for. These points

extracted from the Real Log are produced by the AD function after sensor fusion. Each feature point is recorded as a three-dimension array $[P_{fx}, P_{fy}, P_{fz}]$, where P_{fx}, P_{fy}, and P_{fz} are the positions along X, Y and Z axes of the Real Log Coordinate, respectively. The Real Log contains a large number of feature points scattered in the whole test field, that could be used as a reference to reconstruct the virtual test environment in the next step.

3.3 Result

Fig. 3.2 shows all the data extracted from the Real Log, consisting of basic data about road, vehicle and camera posture and feature points. All these data are used as references to build virtual scene in both SPAS and Unity.

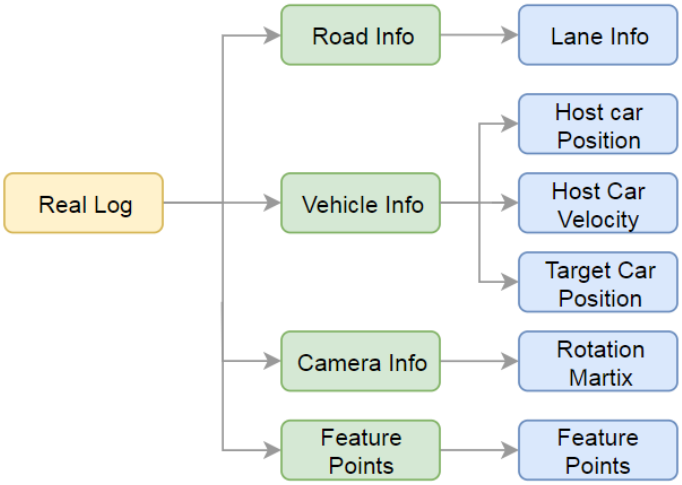

Figure 3.2: Data extraction from Real Log

4

SPAS Setting and Sensor Data Extraction

Once the reference data was extracted, the next step shall be setting the virtual scenario based on this reference. Scenario settings are important as the generated image data and sensor data will be the input of the open-loop test process. Besides, the image data and the sensor data should match each other, since they aim to describe the same segment of the field test. As the system structure depicted in Fig. 2.2, image data is extracted from Unity software and sensor data is produced by the SPAS system. In order to match image data and sensor data, the scenario settings of SPAS and Unity should be matched first. Because the Unity setting has more details, it is more convenient to first set the basic scenario in SPAS, including road, lane, and vehicle movement. After that, the initial setting parameters are applied to Unity, then more details can be included in the setting to improve the realism of visualization.

In the SPAS system, an ideal scenario is built, and environmental elements and parameters will be controlled strictly. Furthermore, only a limited number of all the parameters and factors could be included in software, thus the parameters setting in SPAS could not be exactly the same as the real world. However, once the scenarios of Unity and SPAS could match, it is unnecessary to develop an exactly same scenario as the real world. First, the final goal of the virtual test is to evaluate the open-loop detection tool chain based on Unity, so a scenario closer to the real-world would be better but is not essential. Second, the reason to use Unity is to achieve effective real-time processes by ignoring factors with lower level of importance. Thus, the selected real field test segment is only a template to design scenario, and a virtual environment with similar basic parameter settings is enough for the test.

4.1 SPAS Coordinate

In SPAS, the right-hand rule coordinate is the same as the Real Log Coordinate, which uses X, Y and Z axes to point the front, left and up sides of the host car respectively. The numerical value on X and Y axes are also named as 'Longitude' and 'Latitude' respectively, which will be used in data analysis.

The coordinate in SPAS system is shown in Fig. 4.1. The vehicle's position is set with the center of the front wheel axis, while the recorded radar data is set as the

distance from the center of the host car's rear wheel axis to the nearest edge of the target car. The camera is located between the front wheel axis and the rear wheel axis, and the position could be calculated with the information of the vehicle position and the relative distance between the camera and the front-wheel axis. For different types of vehicles, the relative position of camera and wheel axes inside the car will also be modified.

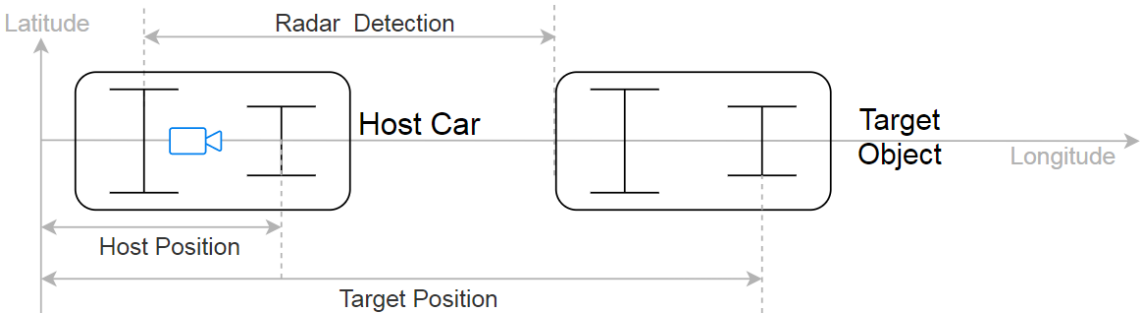

Figure 4.1: SPAS Coordinate

4.2 Basic Setting in SPAS

The most basic and important elements in SPAS scenario are road, lanes, the host car and the target car. These elements are set according to the selected Real Log segment, which is described in Fig. 4.2 and Table 4.1.

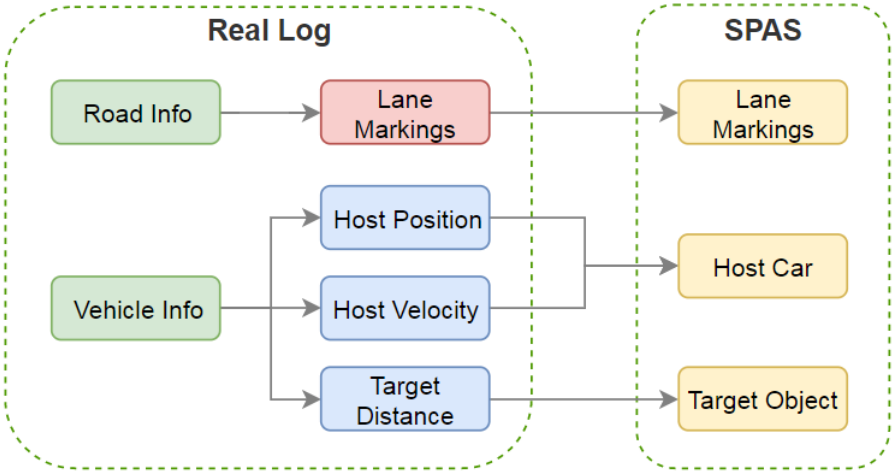

Figure 4.2: SPAS setting according to Real Log

Parameter	Setting
Road Shape	Straight
Road Length	1000 m
Number of Lanes	2
Number of Lane Markings	3
Lane Width	3.5 m
Host Car Initial Position	(0, 1.9, 0)m
Host Car Latitude Speed	0
Host Car Longitude Speed	as depicted in Fig.4.4
Number of Targets	1
Target Velocity	0
Target Position	(136.7, 1.75, 0)m

Table 4.1: Basic parameters setting in SPAS

- **Road**
  Even though the road in the selected field test segment has been assumed to be straight, the actual one might still be slightly tilted, curved and fluctuated. In SPAS, the road is set to be an ideal flat straight road with enough length (1000 m) for vehicle movement.

- **Lane**
  According to the lane detection result of the Real Log, the average detected distance between two lane markings is around 3.5 m, which is equal to the lane width. Three straight and parallel lane markings with the same distance are set in SPAS. As illustrated in Fig.4.3, The middle lane marking sits on the road center, and where it starts is set as the origin.

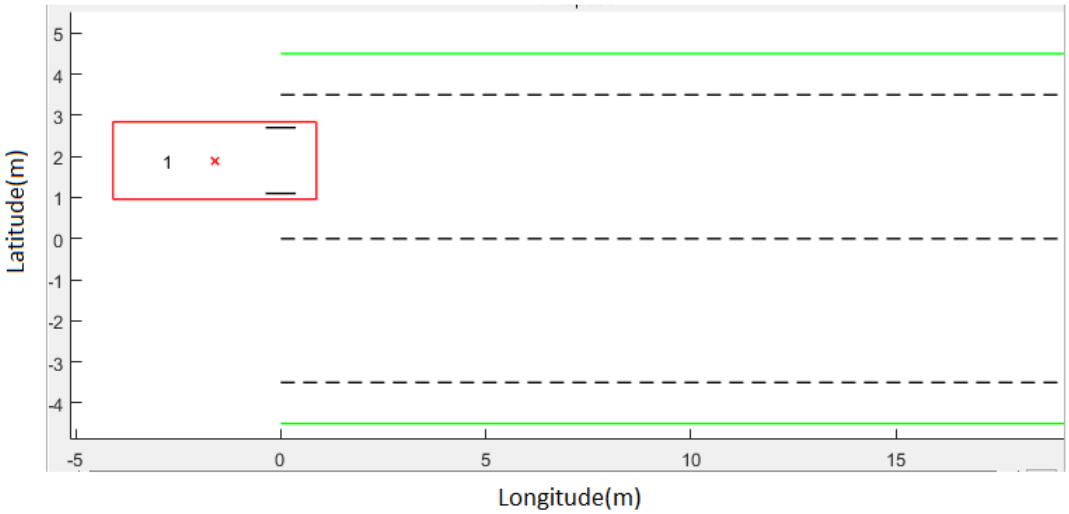

Figure 4.3: Lane setting and original position of the Host Car

- **Host Car**
  The host car refers to the VCC test vehicle equipped with the Ra-Cam system

to record data. The absolute position of the vehicle does not matter, therefore the initial position of the host car could be set at the beginning of the road. In the field test, the detected distance to the left and the right lane markings are around 1.6 m and 1.9 m respectively, so the initial position of the host car is set at (0, 1.9 m, 0), as depicted in Fig. 4.3. Since there is no lane change and Latitude shifts are small, the horizontal movement of the host car could be ignored in the ideal setting. The Longitude speed is set by the Real Log detection result, as shown in Fig. 4.4. By utilizing the initial position and the vehicle velocity, the whole movement of the host car could be recovered in SPAS virtual scene.

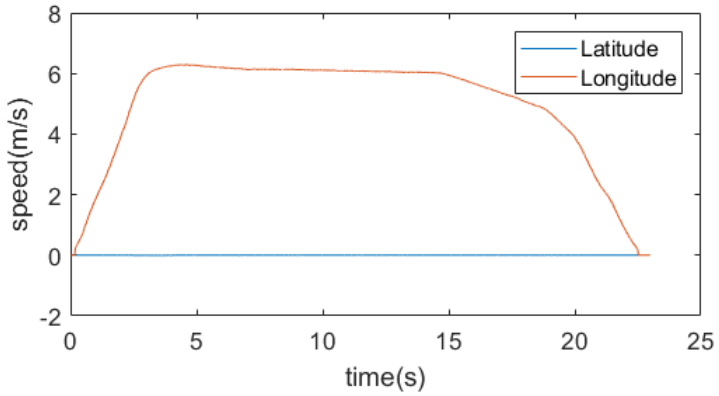

Figure 4.4: Longitude speed of virtual host car

- **Target Car**
  In the selected simple field test, the target car is a stationary vehicle, parking in front of the host car with a long distance. The detected distance to the target car could be attained from the Real Log. However, the large fluctuation of the detection result at the initial time means that the data is unreliable for long distance. When the host car finally stops in front of the target car, the average detected distance (R) is used to calculate the location of the target (P_o) in SPAS. Considering the coordinate transformation for Fig. 4.1, host car position (P_h) and the distance from the rear wheel axis to the end edge of the car ($D_{rear2end}$) should be used to calculate the position of the target:

$$P_o = P_h + R + D_{rear2end}. \tag{4.1}$$

4.2.1 SPAS Sensor Data

As shown in Fig. 4.5, after setting the scenario and running SPAS process, sensor data will be generated. Ego Data includes all information of host car movement, such as velocity, acceleration and rotation. Radar Data will record parameters related to the detected target, and the most important information is the distance between the host car and the target car.

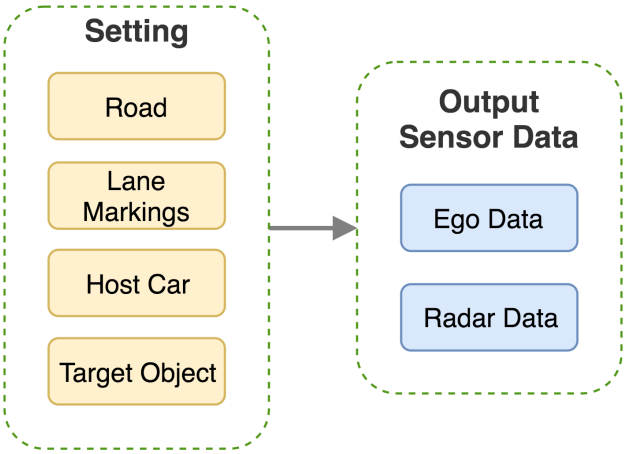

Figure 4.5: Brief process structure of SPAS

5

Unity Development

Once the SPAS virtual scene is built and the required sensor data is generated, the next step is to re-simulate the same field test segment in the Unity platform. This Unity development consists of both environment building as well as the motion control for the virtual cars and the virtual camera. The virtual image data for the future open-loop test is extracted from the developed Unity virtual scene. In this project, the virtual scene is built initially based on SPAS settings, then extra manual settings are added to improve realism. After that, the behaviour of the virtual host car in Unity is determined in the same way as that in SPAS. Finally, photo-realistic virtual images, as alternatives for the raw images data, are extracted from Unity.

5.1 Scene Development

Scene development in Unity refers to the process of making the virtual 3D world in Unity similar to the original field test environment. The original field test scene can be mainly decomposed into several different visual components, and most components can be reconstructed separately in Unity with given 'assets' including models and materials. The positions and orientations of some visual components could be set according on the Real Log data while other less important elements and parameters can be just set manually.

- **Unity World Coordinate**
  Unity has its own built 3D coordinate system following the left-hand rule. This coordinate makes environment development convenient because the previous Real Log Coordinate can be easily transferred into the Unity World Coordinate once the initial position of the host car is set at the origin in both of these coordinates.

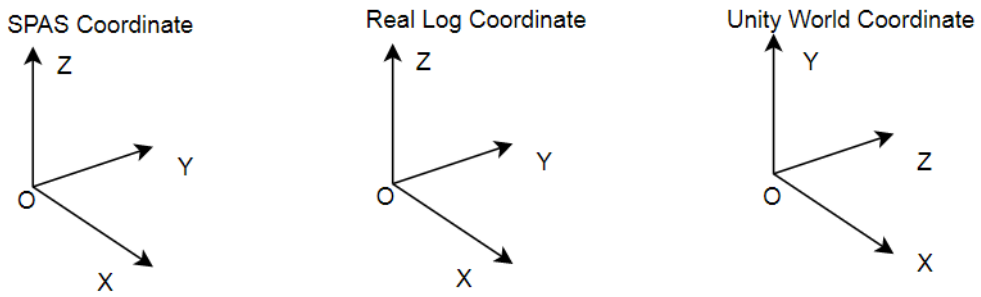

Figure 5.1: SPAS Coordinate, Real Log Coordinate, and Unity World Coordinate

Fig.5.1 illustrates the relationship among three different coordinates. The transformation from the Real Log Coordinate (or SPAS Coordinate) to the Unity World Coordinate is just swapping the array sequence from $[X, Y, Z]$ to $[X, Z, Y]$.

- **Road Setting**
  As described in Chapter 4, lane markings are fixed to certain positions along the whole straight road in SPAS. In Unity, a visually realistic straight road is built with all the virtual lane markings located at the same positions as the SAPS setting.

- **Target Car**
  In Unity, the target car model is a virtual prefab component built by Volvo Car Corporation. The distance in the Unity World Coordinate from the origin to the front wheel axis of the target car is set the same value as in the SPAS Coordinate.

- **Surroundings Setting**
  Unlike basic elements as lane markings or the target car, surrounding elements that are not on the driving path may not have a noticeable influence on the AD function. Thus, surrounding elements such as bushes, trees and barriers seem to be less important. However, well-detailed surroundings will make the final virtual scene much more resemble the original field test environment, which will also make the open-loop test result more reasonable [5].

  The surroundings are built up in two steps. First, all the feature points extracted from the Real Log are imported as the contour of the surrounding elements. Then, 3D models including terrain, bushes, trees and fence are set manually based on the contour of feature points. All these vision components will be finally adjusted according to the Real Log video. This manual development costs a lot of labour to set up and adjust to make the virtual scene good enough.

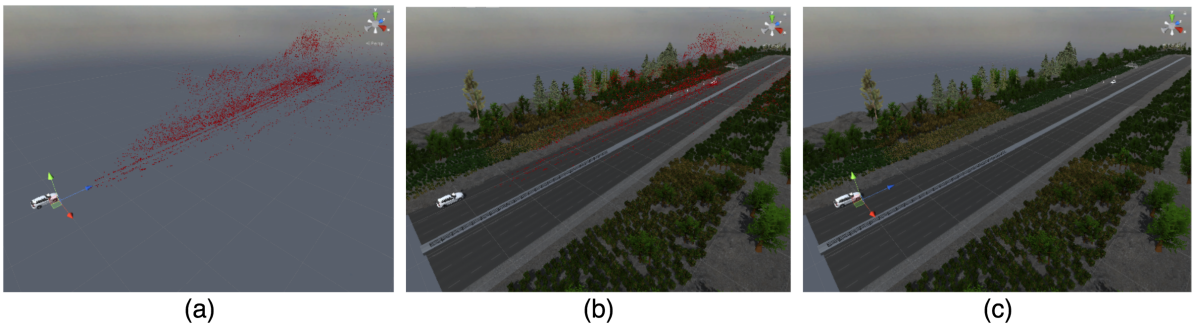

Figure 5.2: Sequence of manually built virtual scene

Fig. 5.2 illustrates how manual development is accomplished. Figures (a) to (c) are screenshots of individual imported feature points, built scene along with feature points and the final virtual scene, respectively. The red dots

scattering all over the virtual world are referenced feature points, which are hidden once the scene development is completed.

5.2 Camera Setting

Same as the real-world photos photographed by the real camera, Unity also has a virtual camera to take images from the virtual world. Unity's virtual camera can adjust its position, orientation and camera physical model to simulate the real camera. This section focuses on how the virtual camera in Unity is set according to the real camera used in the field test.

- **Position & Orientation**
  Camera's position is set based on the host car's position considering the relative distance from the camera to the front wheel axis of the host car. Camera's orientation is controlled with the rotation matrix extracted from the Real Log.

- **Camera Physical Model**
  The camera model describes the mathematical relationship between the scene and the final extracted image. In the real world, plenty of parameters might affect the final captured photos. While in Unity, there are only a limited number of parameters that could be included and adjusted. To be more specific, only focal length, sensor size and FOV (field of view) are calculated in the Unity camera model setting. Focal length is a physical property that measures the light concentration ability of a lens. Collimated parallel light rays will be concentrated to a point named 'focus' by a lens, and the distance from the focus point to the lens center is defined as focal length [6]. Sensor size, also named as film back size, indicates the size of the image plane and determines the aspect ratio and resolution of the final image. FOV can be calculated according to sensor size and focal length, which restricts the visible range size. Fig. 5.3 shows the relationship among these three parameters:

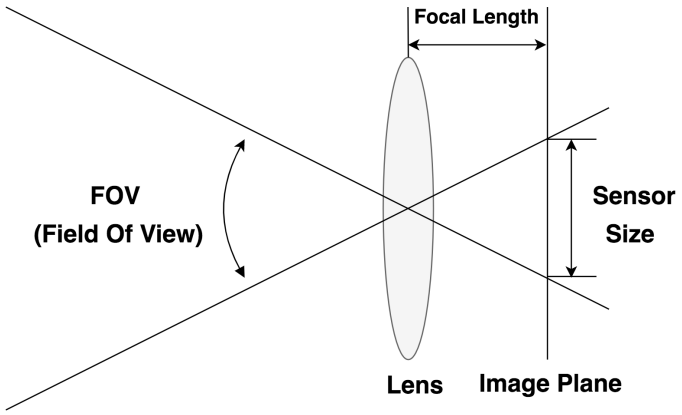

Figure 5.3: Illustration of focal length, FOV and sensor size

In Unity, the camera treats a lens as a pinhole, which simplifies the numerical

relationship of parameters as the formula:

$$SensorSize = 2 * FocalLength * tan(FOV/2). \tag{5.1}$$

5.3 Movement Control

After constructing the virtual scene and importing camera settings, $C\#$ code script is developed to control the visual components [3] [7]. This script controls the virtual host car to simulate the real host car's movement and extract the virtual images of the virtual scene. To be more specific, the camera's posture changes according to the extracted camera rotation matrix, while the virtual car's driving path is set based on the extracted position information from the Real Log. Each time the virtual car's position is updated, the virtual camera captures an image of the current virtual view, which is illustrated in Fig. 5.4. Finally, generated virtual images have the same frame amount as the Real Log segment.

Figure 5.4: Examples of extracted virtual images sequence

5.4 Result

Eventually, a well-detailed virtual scene in Unity is constructed. Furthermore, photo-realistic virtual images are generated from Unity. Environment variables like light condition and weather, which may remarkably affect the detection result, can be easily adjusted, and other vision components such as pedestrians and vehicles can also be introduced.

6
Open-Loop Detection

Chapters 3 to 5 describe the tool chain to produce all necessary input data for the open-loop software. This chapter briefly describes the test output and makes a basic data analysis to validate the feasibility of this development tool chain.

The detection software is named as open-loop detection since there is no feedback from the output to its status. In fact, the open-loop software only does object detection in this thesis project. After analyzing the output, it could be validated if Unity can be the source of image data. If the feasibility is proven, more detailed research could be done, including what factors could affect the detection result.

6.1 Software Output

All the input data for the open-loop detection are set based on the same field test segment. As the structure shown in Fig. 2.2, image data stands for the virtual images extracted from Unity, while sensor data is generated by SPAS. The output of the open-loop detection includes a visible video with markings on the original images as well as a *.mat* file containing the numerical results of the detection.

- **Vision Video**

Figure 6.1: Extracted virtual image sequence

Figure. 6.1 is a screenshot of the output video, making it clear that how the

output video looks like. The green lines on the leftmost and the rightmost sides are placed on the detected road edge. Next to the road edge are the yellow lines standing for the detected lane markings. The red box marked in the center illustrates the location that the target car is detected. There might also be other markings in different colors for other types of objects, but the detection output of the simple scene in this thesis only consists of these three kinds of markings.

- **Numerical Detection Result**
  Besides video log, the output also has a *.mat* file stores all the detected numerical result. The *.mat* file that stores the basic movement information such as the host car's position, velocity, acceleration, orientation as well as the detected positions of lane marking, road edge and other objects. As for result analysis in this thesis, only the most important parts of data are selected to analyze.

6.2 Test Output

To validate whether Unity has the ability to generate detailed enough virtual images for the open-loop test, comparison between the test output and the original settings seems necessary. The detection results and parameters settings could be plotted together to obtain a clear visualized comparison.

- **Lane Markings**

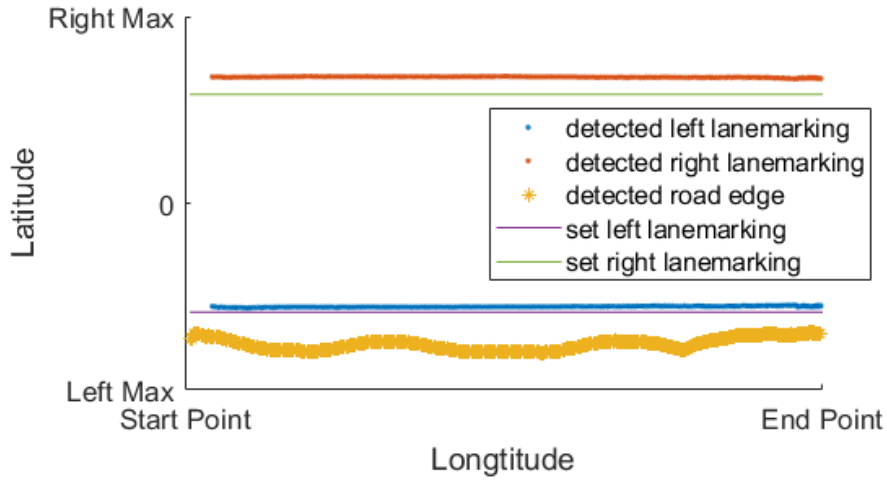

Figure 6.2: Detection result of lane markings compared to origin setting

Fig. 6.2 illustrates the comparison between positions of the detected lane markings and the original setting, and the detected road edge is also included. The slight gap between setting and detection is acceptable due to VCC's standard, showing that the AD function still works when image data are replaced by extracted Unity virtual images. Meantime the right road edge is lost because of its long distance to the host car. This is also acceptable as a far

road edge is not so important for autonomous driving once the adjacent lane markings have been detected.

- **Target Object Car**
  Fig. 6.3 illustrates the distance between the target car and the host car. The open-loop detection software can not detect the target car at the beginning. But once the target is recognized successfully, the detected distance output and the setting are almost matched. The offset might be the result of several reasons, such as model quality of the virtual target car, the simplified setting of the camera model in Unity or the virtual background. Furthermore, the test software can not get exactly accurate detection results, no matter how good the input data is. However, this offset is still tolerable due to VCC's standards, proving that the open-loop object detection software of the AD function is still working with the Unity virtual scene.

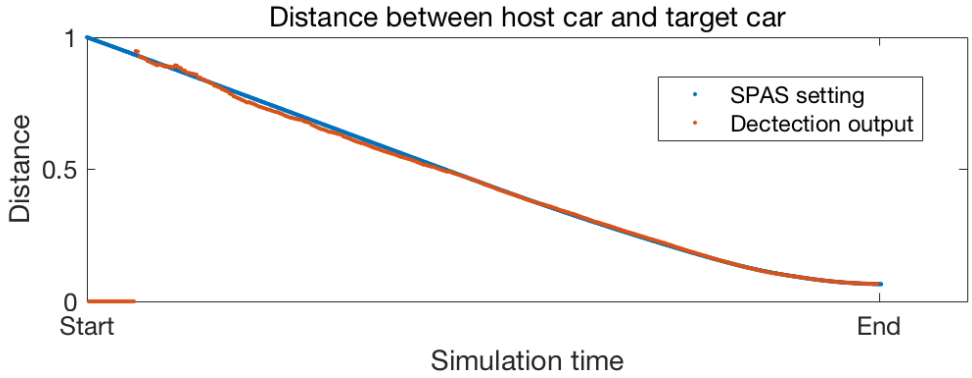

Figure 6.3: Detected distance between the host car and the target object car

- **Velocity & Acceleration**
  Velocity and acceleration information for the host car is already generated by SPAS, which are recorded as Ego Data. It is figured out that the host car movement data in both SPAS setting and the open-loop detection output are exactly the same, which indicates that the open-loop software does not do movement update for the host car, but only keeps the input sensor data as the output.

6.3 Influence of Radar Data

By comparing the detection result and setting parameters, a preliminary conclusion would be that the Unity virtual scene has the ability to offer the image data for the open-loop detection software, and a reasonable and reliable detection output could be obtained. The next step should be making variation in the Unity virtual scene to find out what elements may affect the detection result. Additionally, the performance of the open-loop detection software with more complicated situations, such as a bus driving on adjacent lane, will be considered.

As the virtual scenes in Unity and SPAS describe the same test, their settings should be matched. To adjust the vision components in Unity, it should be figured out which settings in SPAS should be modified correspondingly. Since there is only the road and the target car set in SPAS and the radar data is ideally generated, the radar information from SPAS only contains the distance to the target car. However, the radar data from the real field test would contain information about all the surroundings. Because the open-loop detection software still works with the lack of such surrounding information, we made a bold assumption that the open-loop software may not be influenced by the SPAS radar data.

As a verification, dummy radar data of all zeros are generated to replace the current radar data input of the open-loop software. The final detection result shows that the dummy radar data makes no difference in the open-loop test, as the output is exactly the same as before. Since the influence of the SPAS radar data could be ignored, only Ego Data and Image Data should be matched for the system in Fig. 2.2, which means the host car movement information should always be the same. To research the effects of vision components in Unity, all surroundings except the host car could be adjusted.

6.4 Result

It is validated that Unity virtual images with proper setting have the ability to produce credible output with the open-loop detection. This conclusion indicates the potential to make more virtual AD function tests with Unity, as Unity can render detailed enough virtual images in short time. Furthermore, it is figured out that the radar data does not influence the open-loop detection software. This result brings large convenience for future research, since there is no need to modify the radar data when there is variation in the Unity virtual scene.

7
Variables and Comparisons

The developed re-simulation tool-chain consists of building the virtual scenes, generating the required input data and testing with the open-loop test software. The feasibility of Unity virtual images for the open-loop detection test has already been validated, so the next goal is to make preliminary exploration about how the detection result will be affected by Unity virtual scene. Once the SPAS Ego Data is fixed, Unity image data will be the only influential factor for the detection process. A dummy SPAS sensor data could be used for all Unity virtual scenes, because the SPAS radar data will not affect the open-loop detection, as discussed in Chapter 6. Besides, the SPAS Ego Data is also fixed for all tests in this chapter, which means that the virtual host car's movement is the same for each detection test.

The virtual camera model used in Unity is fixed, so parameters such as focal length, sensor size and field-of-view are fixed. In the Unity virtual scene, the preliminary exploration only aims to figure out how the open-loop test result will be affected by different visual elements. For example, light conditions like weather, shadow, light intensity and light direction might be important influential factors for object recognition. Besides, the target car's color and other extra vehicles around may also affect the detection performance. Thus, in each group of tests, only one parameter will be adjusted to compare the detection outputs, based on the control variable method.

7.1 Output Comparison

To clearly compare the detection performance, several important parameters are extracted from the detection output.

- **Target Detection**
  For target car detection, the most important outputs are (1) the distance between the host car and the target car as well as (2) the maximal detectable distance — the maximal distance in which the target car could be recognized.

- **Lane Detection**
  The detection algorithm would pick the nearest segment of the road to calculate the two-dimensional position of lane marking (P_{Lane}) as a second-order polynomial expressed in (7.1). Since the road and the path of the host car are assumed to be straight in the setting, even though there is unavoidable

detection noise, the term with zero order in the output equation (a_0) can be interpreted as the latitude distance between the center of the host car and the lane markings.

$$y = a_0 + a_1 \cdot x + a_2 \cdot x^2 \tag{7.1}$$

- **Detection Error**
  In the basic virtual scene development, host car movement, target car position and lane marking position are set. Detection error could be calculated by utilizing the detection output ($P^{detection}$) and the setting value ($P^{setting}$). Where P stands for the distance between the host car and the target car.

$$Error_i = P_i^{detection} - P_i^{setting} \tag{7.2}$$

  Where i stands for the time index, indicating the frame number of camera image sampling.

- **Detection Deviation**
  The Detection error will change with time even though the target car and the surroundings are stationary. Thus a numerical factor should be introduced to assess the dispersion of error. Based on the definition of MSE (Mean Squared Error) in (7.3), detection deviation (D) is introduced in (7.4), where the offset from the mean is replaced by the detection error, where N is the total number of frames.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2 \tag{7.3}$$

$$D = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(P_i^{detection} - P_i^{setting}\right)^2} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} Error_i^2} \tag{7.4}$$

7.2 Light Intensity

Light intensity is a scale factor used in Unity to control light color, which could be adjusted to simulate over bright or dim light conditions. The light intensity setting can change the brightness of the extracted images, as shown in Fig. 7.1.

Figure 7.1: Extracted Unity images for the same frame with different light intensity settings

Fig. 7.1 (a) (b) and (c) are extracted Unity images for the same frame with light intensity factors at 0.3, 1 and 2 respectively. This group of tests uses five different light intensity at 0.3, 0.5, 1.0, 1.5 and 2.0.

7.2.1 Target Detection

The distance detection error is depicted in Fig. 7.2, where the red line marks the longitude position of the target car. It is obvious that further distance to the detected target will cause a larger error. But when the host car is close enough to the target, all these tests could work well to detect the target car with high accuracy. When the light intensity equals to 1, the absolute value of the detection error is minimized. Due to the confidentiality requirement, axes are normalized to hide the exact value and only the tendency of the curves is analyzed. For longitude, 0 is the origin and 1 stands for the target longitude position. The longitude normalization will be performed for all detection error figures in this chapter to illustrate the relative distance. Latitude is normalized with respect to the accepted tolerance, which is chosen manually for each test.

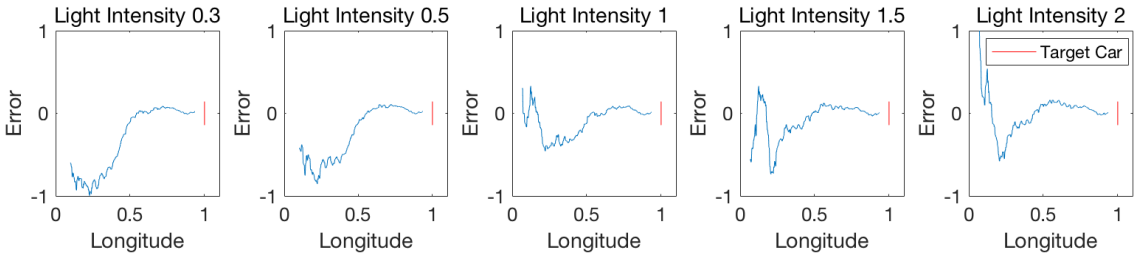

Figure 7.2: Detection error with different light intensity

Deviation of the target car detection is shown in Fig. 7.3. It is clear that the light intensity factor of 1 generates the smallest deviation. When the light intensity factor decreases below 1, the deviation will rise quickly. For long distance (at the beginning of the test), darker images will bring larger detection error than brighter images, which might happen due to the low contrast value that makes it harder to distinguish the edge of the target when it seems too small. For the light intensity factor larger than 1, the deviation will only increase very slightly. The last two figures in Fig. 7.2 and Fig. 7.1 (c) indicate that the edge of the white target car might not be so clear in condition of too bright surroundings and in very long distance.

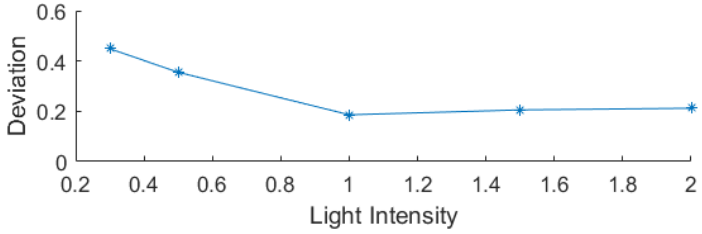

Figure 7.3: Deviation of the target distance detection with different light intensity

The maximal detectable distance for the same target car is illustrated in Fig.7.4, where the vertical ordinate is normalized with the longitude position of the target

car. This tendency validates the discussion before, that a lower light intensity makes it more difficult to detect the target car.

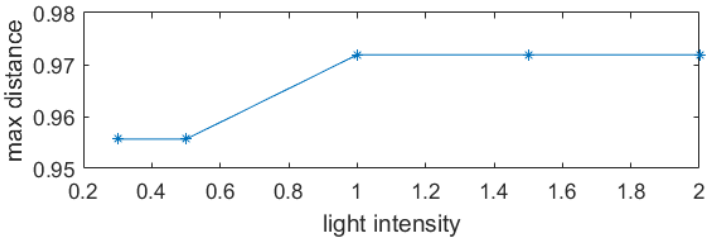

Figure 7.4: Maximal target detectable distance in presence of different light intensity values

7.2.2 Lane Detection

In this test, only the two lane markings on both sides of the host car's driving lane are detected. The detected positions of the two lane markings are shown in Fig. 7.5, where the light blue dash line is the setting position of the lane marking, and the red line is the target car longitude position. Fig. 7.5 is normalized with the same factor used in the target detection error of Fig. 7.2.

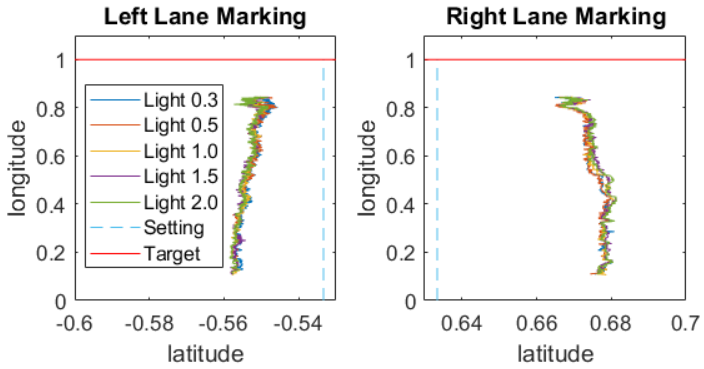

Figure 7.5: Lane Detection in presence of different light intensity values

It is obvious that no matter what is the value of light intensity, the detected lane markings' positions are approximately the same. However, there is a roughly constant offset from the setting value. This offset might be caused by the imperfect match between the camera models in Unity and in the open-loop detection software. For example, when different vehicle types and Ra-Cam systems are employed, focal length, sensor size and the relative position of the camera within the host car will be modified. In such situations, the relative position of the lane markings on the extracted Unity camera images will be changed, making the detected distance shift from the correct value.

Mean detection error is defined as the mean offset of the detected value from the setting value, which is different for adjacent lane markings. As shown in Fig. 7.6, there is an approximately constant mean detection error for each lane marking, but the right lane marking has a bit larger mean offset. The deviation of lane detection

in Fig. 7.7 illustrates a similar tendency. This might happen due to the difference between the surrounding on the left and on the right sides or maybe the driving path of the host car is not at the lane center but closer to the left side. Fig. 7.6 and Fig. 7.7 are both normalized with the acceptable tolerance.

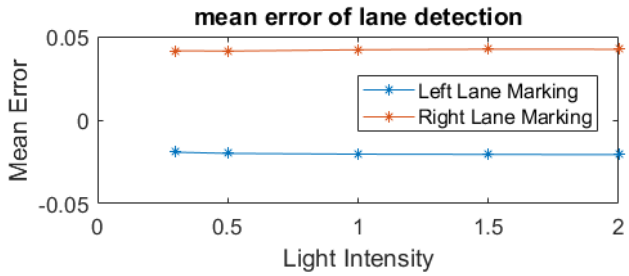

Figure 7.6: Mean error of lane detection for different light intensity values

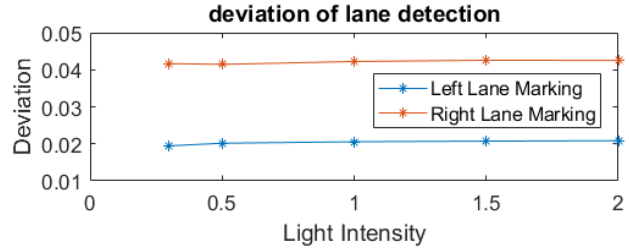

Figure 7.7: Deviation of lane detection for different light intensity values

To verify the influence of the driving path, a new test with the host car driven at lane center is carried out. As illustrated in Fig. 7.8, no matter the host car is driven on the central path or the original path, the distance between two detected lane markings is not affected noticeably, which means that the detected lane width is almost the same. It's remarkable that the detected lane markings' positions are more symmetric with central path, which indicates that the lane markings at two sides will produce similar detected distances.

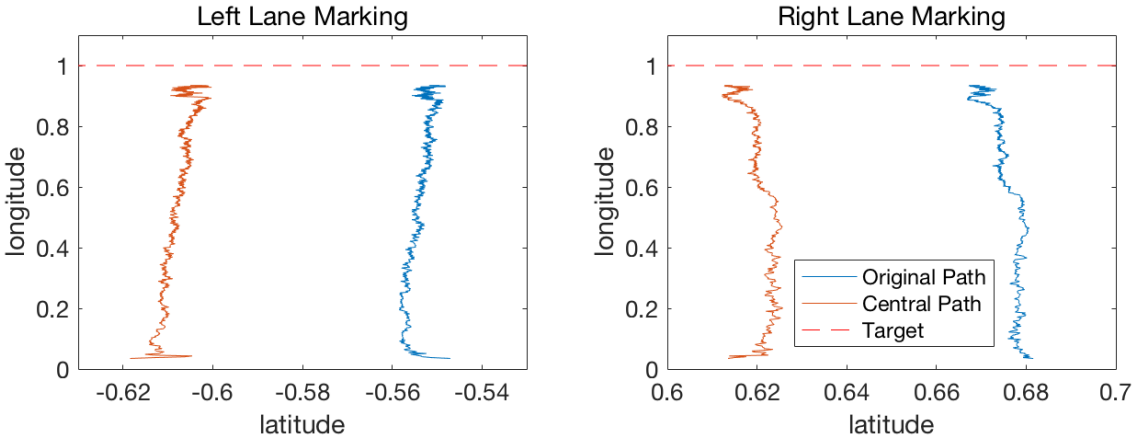

Figure 7.8: Lane detection when host car driven on different path

The mean detection error is plotted in Fig. 7.9. For the central driving path, the absolute value of mean error for the two adjacent lane markings is more similar than left path driving, but the right lane marking still has a bit larger offset. This might be caused by camera rotation and different surroundings on two sides.

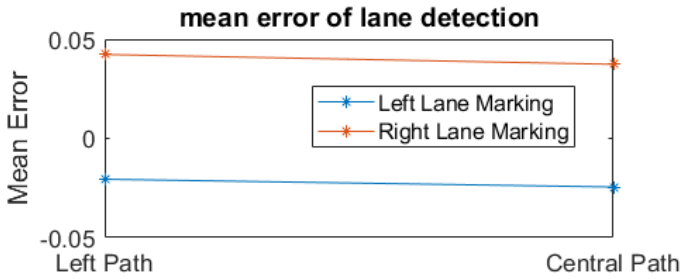

Figure 7.9: Mean lane detection error for different driving path

The distance between the two lane markings is larger than the setting, which might be due to the low height of the camera that expands the distance between lane markings. Furthermore, for the physical camera model in Unity, only limited parameters could be controlled, and these parameters have been set according to the default camera model used by VCC. While the embedded camera model in open-loop detection software is built and set in a different way, which makes it difficult to exactly match all the settings. Thus, it is difficult to erase the offset between detected value and the setting value of the lane marking position only based on Unity. Additionally, lane markings' color, contrast on the road, the sharpness of lane markings' edges and the rotation of the camera will all cause detection noise.

The camera rotation is extracted from the Real Log and the influence is difficult to be quantified. The lane detection outputs with camera rotation and without rotation are depicted in Fig. 7.10. After introducing camera rotation, the fluctuation increases a bit and the detected positions for both lane markings will be shifted slightly towards right. This effect might be caused by the curved structure of the road in the real field test. The camera rotation towards left side will cause the road to be shifted to right in extracted images. Consequently, both camera rotation and the driving path will variate detection output for the lane markings on two sides.

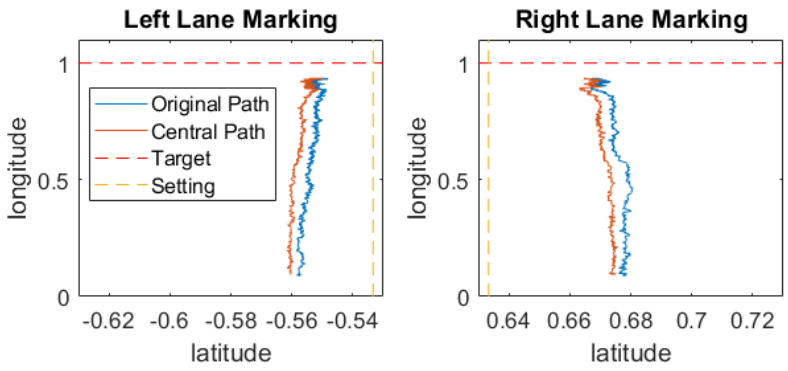

Figure 7.10: Target detection error with and without camera rotation

7.3 Shadow

In the Unity virtual environment, shadow could be hidden. This situation could also happen in real world, when there is no strong light. Fig. 7.11 depicts images extracted from Unity. For making a comparison between detection outputs, same analysis like what have been done for light intensity needs to be carried out.

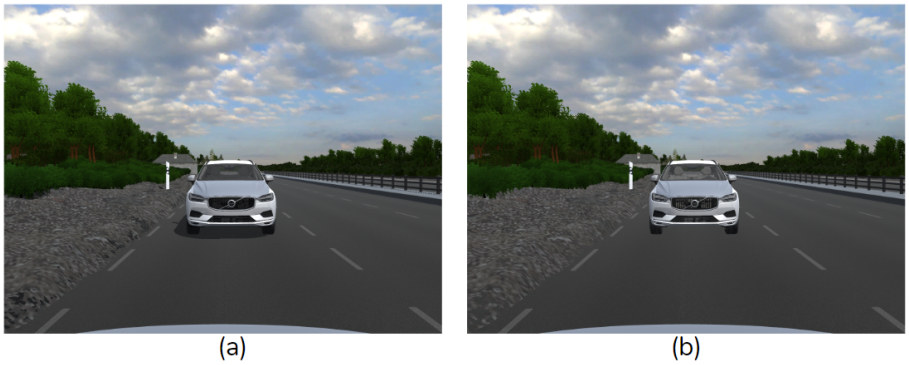

Figure 7.11: Extracted Unity Image with Shadow(a) and without Shadow(b)

Target detection error plots are depicted in Fig. 7.12. Based on the plots of Fig. 7.12, it can be declared that for different light intensity settings, tests with shadow and those without shadow will obtain approximately the same target detection error. Thus, the influence of shadow on target detection is not remarkable.

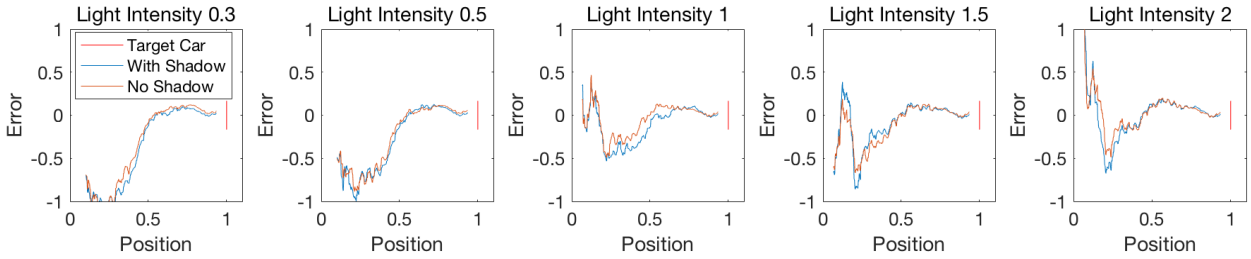

Figure 7.12: Target car detection error with shadow and without shadow

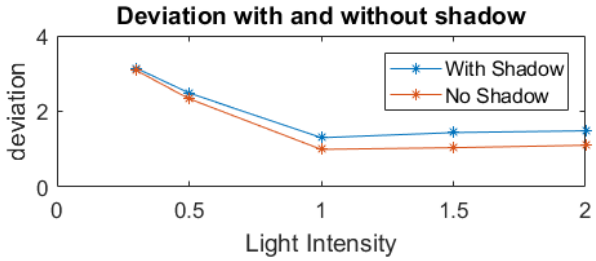

Figure 7.13: Target car detection deviation with shadow and without shadow

Target detection deviation is plotted in Fig. 7.13. For low light intensity, the deviations for these two groups are large and similar. Higher brightness brings a wider gap between tests with shadow and that without shadow. This might happen

due to the higher light intensity that will offer higher contrast of images, making the shadow more obvious. Furthermore, both of these two groups reach the lowest deviation when the light intensity factor is 1 due to the same reason as discussed in the last subsection.

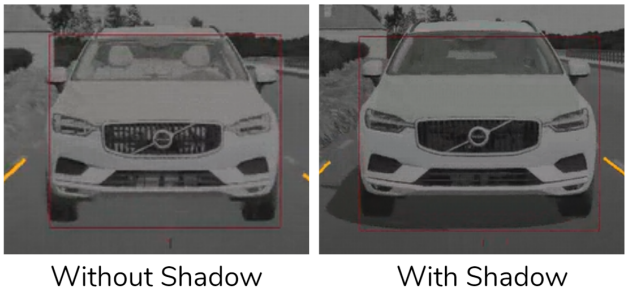

Figure 7.14: Screenshot of detection video output without shadow and with shadow

An interesting phenomenon is that a higher deviation occurs in presence of shadows, which means that the appearance of shadow will interfere target car detection. Fig. 7.14 is the screenshot of the detection video output, which shows that the detected object (marked by the red box) is a bit lower when there is a shadow. The reason of that might be the target detection algorithm aims to detect the bottom of the target vehicle. Thus, when the shadow appears, the algorithm will not follow the edge of the wheel but will track the edge of the shadow and will consider the shadow edge as the bottom of the target.

The maximal detectable distances for different light intensities are exactly the same as Fig. 7.4. This is due to the limitation of Unity that the shadow will only be rendered when the distance between the virtual camera and the object is close enough. The target car is the only vehicle in front of the host car and surrounding is not very complex, so shadow will not influence the beginning of detection remarkably.

Moreover, there is no shadow of surroundings covering lane markings, so lane detection are nearly the same as Fig. 7.5 and Fig. 7.7, which are not plotted here again to avoid redundancy.

7.4 Light Direction and Target Color

As discussed before, shadow will influence target car detection and light intensity will change the contrast of the shadow, while light direction could change the shape and position of the shadow. Thus, tests with different light directions are carried out in this section to explore the influence.

Fig. 7.15 illustrates the definition of the light direction, whose reference coordinate system is the Unity World Coordinate. The light direction pointing to the target car and parallel to the ground (Y-axis) is marked as zero degree, and the opposite direction is 180 degree. In this group of tests, light directions are only changing on the vertical plane.

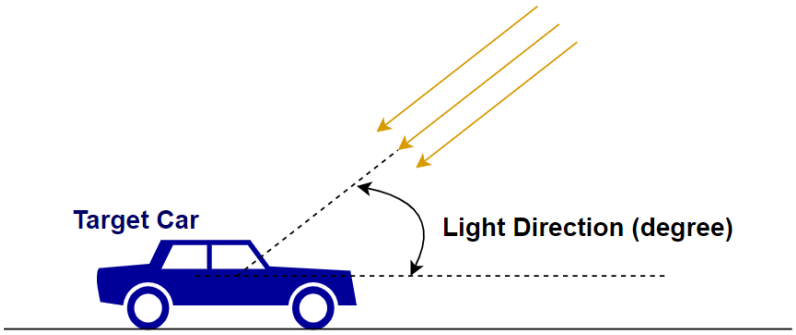

Figure 7.15: Definition of light direction

Fig.7.16, (a) to (c) are extracted Unity images with light direction angles of 150°, 90° and 30°. Large light direction means that light comes from forward, the remote end of the road is brighter, and the front side of the target car is darker.

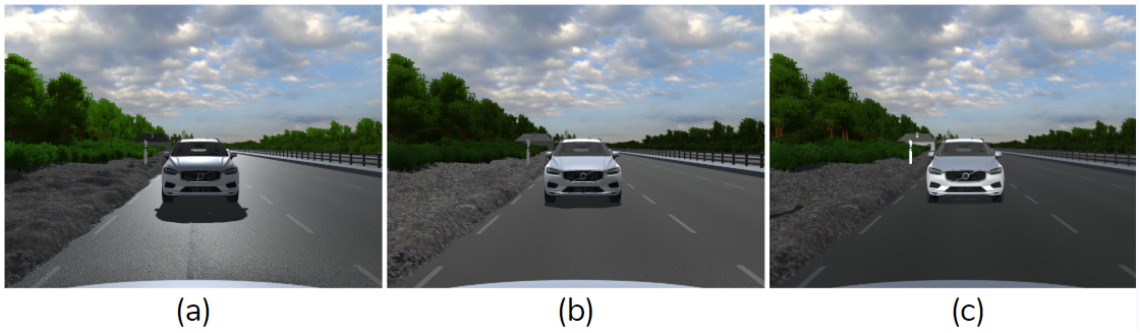

Figure 7.16: Unity images with different light directions

The target detection error is shown in Fig. 7.17. The high peak at 90° light direction angle might be caused by data loss and accidental error. The algorithm keeps tracking the target and calculating distance continuously, so it will catch the target soon again after losing track. It is obvious that when the light direction angle decreases, the beginning of the error curve moves down. It means that when the host car is far from the target car and light direction is moved from forward to backward, the detected distance to the target car will be smaller than the real value. To verify that whether the color of the vehicle will affect the simulation results or not, a group of tests with a black target car is carried out and the detection error is shown in Fig. 7.18.

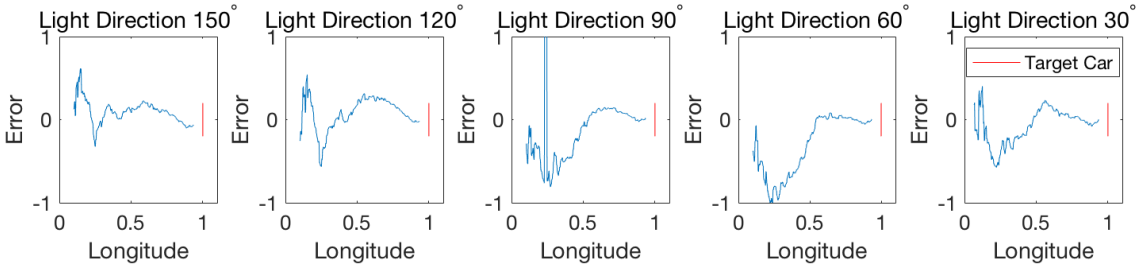

Figure 7.17: White target detection error for different light directions

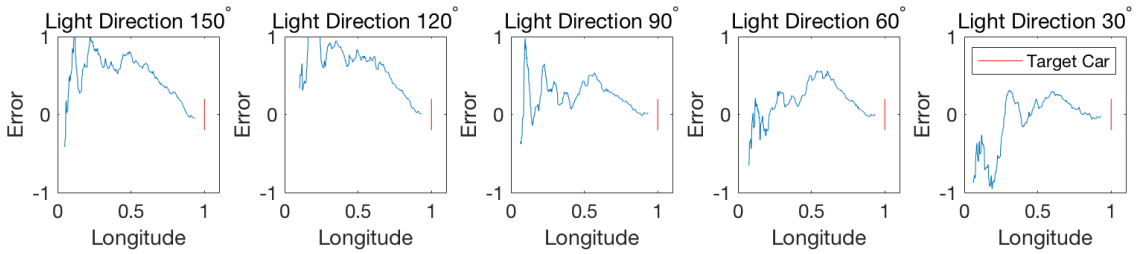

Figure 7.18: Black target detection error for different light directions

According to simulation results depicted in Figure. 7.18, the error curves of the black target will decreased by reducing the light direction angle. The black target car always has a higher detection error, which is clearly illustrated in Fig. 7.19. Even though the detection error is nearly zero when the host car is close to the target, detection offset is large when two cars are far from each other. Furthermore, the error is defined by Equation. According to (7.2), a higher value of error means a larger detected distance. It is obvious that the error curves of the white car are mainly below zero while for the black car it's mainly above zero, which means that the open-loop detection software will consider the white car and the black car at different sides of the setting position. Also considering Fig. 7.16, a larger light direction angle brings a darker target front surface and a longer detected distance. To sum up, darker target and larger light direction angle will make the detected distance shift towards the same direction.

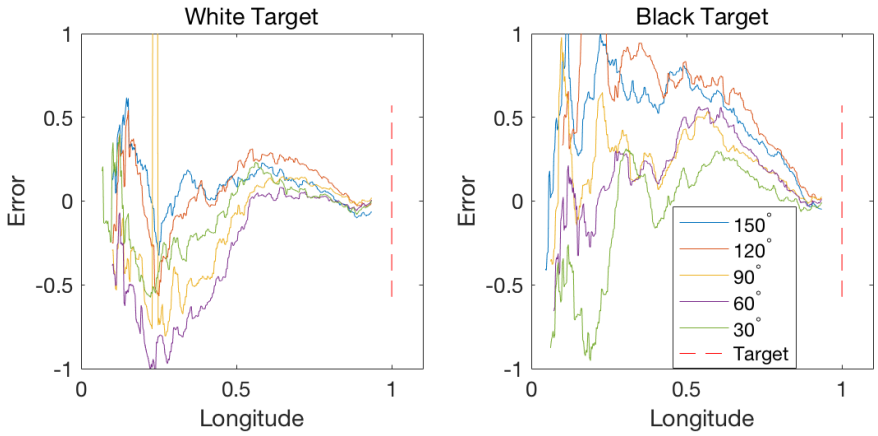

Figure 7.19: Target detection error for different light directions and different target colors

The maximal detectable distance for these two groups of tests is plotted in Fig. 7.20. It is clear that black target car could be detected at a longer distance and the distance can even be further when light direction angle increases. That might be due to a larger direction angle, that makes the front surface of target darker and makes the road around target car brighter, so the contrast between the black target and bright road is increased.

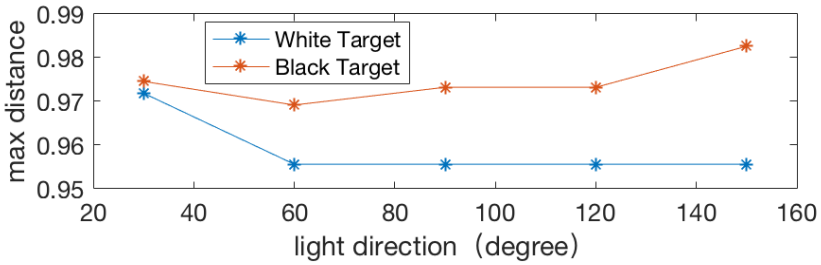

Figure 7.20: Max detectable distance for different target color and light direction

For the white target group, the output of lane detection and its main error and the deviation are shown in Fig. 7.21 and Fig. 7.22. There is no noticeable difference in the lane detection outputs when the light direction is changed, that might be due to the software algorithm only picks the road segment which locates ahead of the camera to calculate lane markings' position. Once that nearest part is not influenced too much, the detection output won't be affected significantly. The remote side of the road changes brightness a lot but it is not taken into account. Since the target color will not influence lane detection obviously, the test of a black target under different light directions is not shown here.

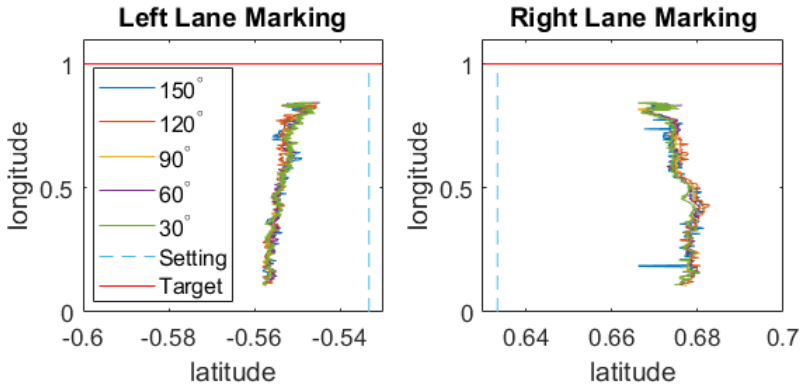

Figure 7.21: Lane detection output for different light directions

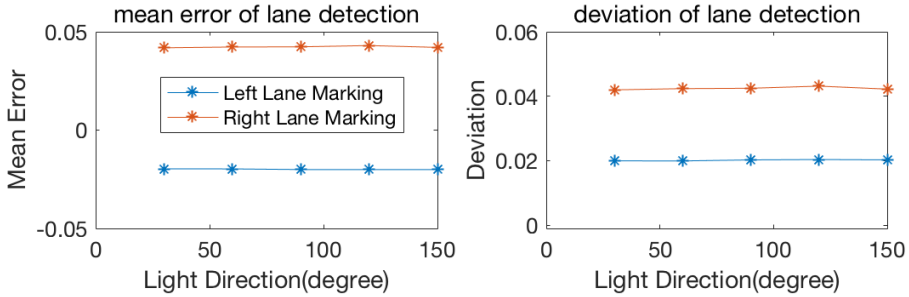

Figure 7.22: Error and deviation of lane detection output for different light directions

7.5 Extra Vehicle and Shadow

7.5.1 Shadow Covering Lane Marking

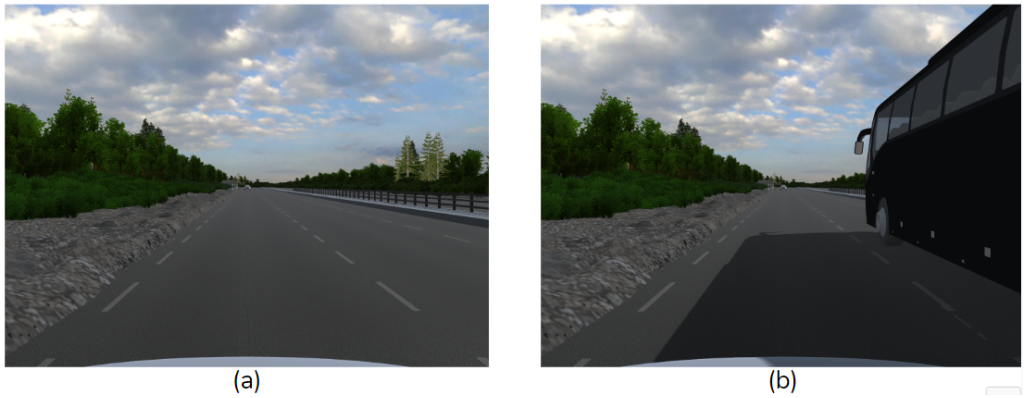

Figure 7.23: Extracted Unity image with and without a bus driven ahead

Situations mentioned in the last few subsections only influence target detection, so factors related to lane detection should be considered. Similar to the shadow below target car that affects the detection result, shadows covering lane marking will also have some effects on detection result. This situation is shown as Fig. 7.23. To be a realistic test, it's assumed that the large shadow is generated by a big bus, which is driving ahead of the host car at the adjacent lane with the same velocity as the host car.

The error in detecting the target car is shown in Fig. 7.24. The two curves with different shadow affection have a similar tendency and the one with bus shadow suffers a smaller error. That might happen due to the shadow makes the road darker, which leads to a higher contrast between the white target car and the black road, leading to an easier detection.

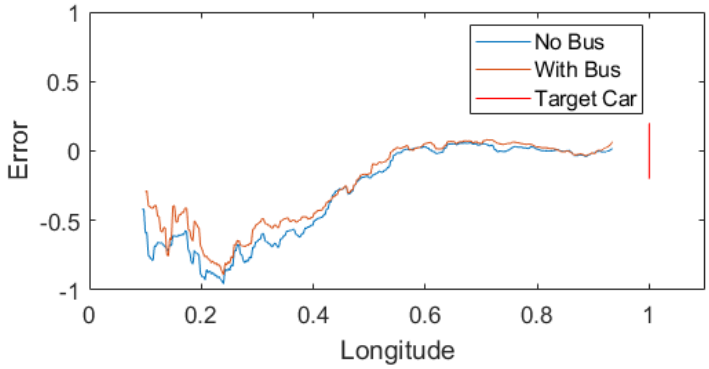

Figure 7.24: Target detection error with and without a bus driven ahead

The output of lane detection is shown in Fig. 7.25. At the beginning of the test, the detection of the right lane marking has obvious fluctuation, which disappears later.

Since the shadow only covers the right lane marking in Fig. 7.23 (b), the fluctuation should be introduced by that shadow. However, this problem is dealt automatically, which might carried out by the adaptive algorithm.

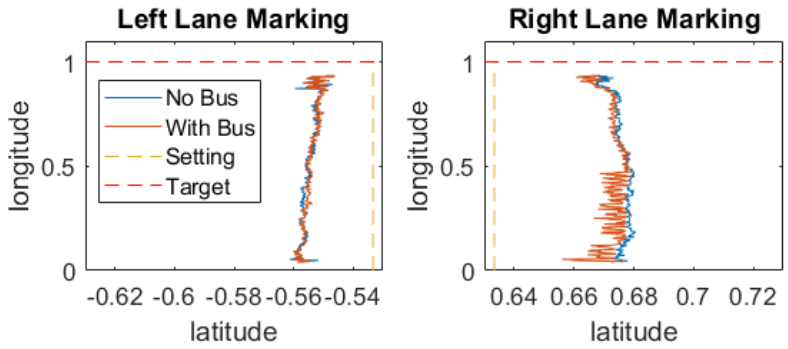

Figure 7.25: Lane detection with and without a bus driven ahead

AS plotted in Fig. 7.26, The normalized mean error and deviation of lane detection are very small and there is no significant difference between these two tests. It shows that the bus, which is driving next to the host cat with a shadow covering the lane marking will not have huge effects on the detected object.

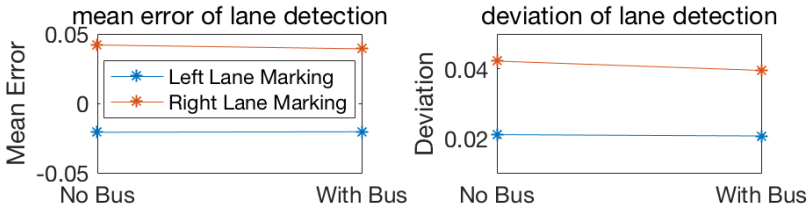

Figure 7.26: Lane detection error and deviation with and without a bus driven ahead

7.5.2 Vehicle Overlap

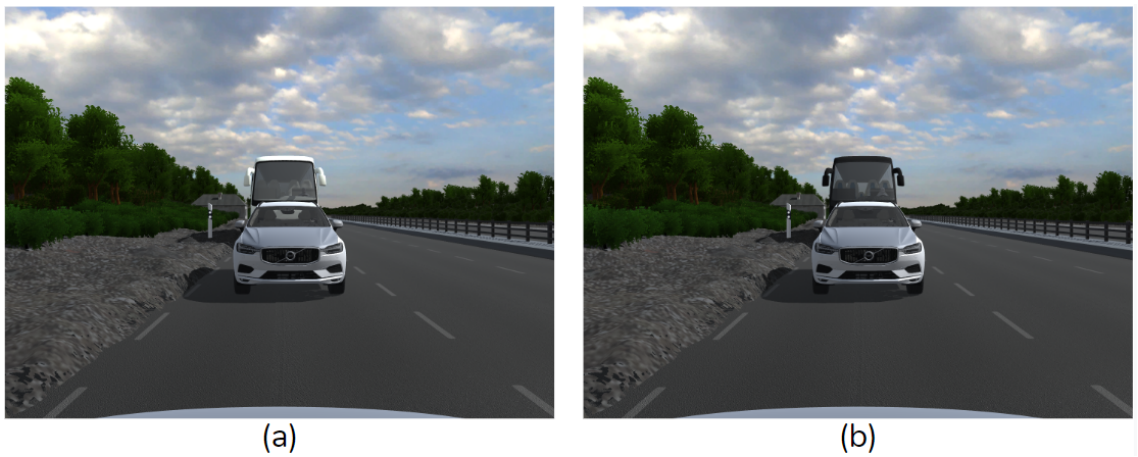

Figure 7.27: Extracted Unity image with vehicle overlap

In the real test, sometimes there will be several vehicles driving close to each other. To test the software behavior in this situation, a bus is placed behind the stationary target car and there is some overlap of vehicles in the extracted images, as shown in Fig. 7.27.

The detected lane position is similar to previous tests, but detection of target car will be affected significantly. The target detection error is shown in Fig. 7.28.

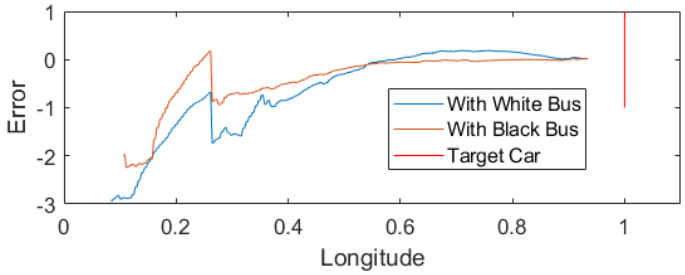

Figure 7.28: Target detection error with bus stopped behind

The detection error is normalized with respect to the maximal acceptable tolerance. At the beginning, there is a large detection error, that is around three times of the tolerance, then it decreases gradually to nearly zero.The main reason of this behavior of error is that the whole vehicle overlap part is detected as a truck when distance between the host car and the target car is very large, as shown in Fig. 7.29 (a). The algorithms to calculate the distance of target car and target truck are different. When the distance is close enough, the target car will be recognized like Fig. 7.29 (b), which causes a jump of detection error. The test with a black bus has larger detectable distances, which means that it could be distinguished earlier. Moreover, the combination of a white target car and a black bus has a smaller detection error. That is reasonable due to the fact that because the edge of the target will be clearer when overlapped vehicles are in different colors.

Figure 7.29: Screenshot of detection video output with bus stopped behind

7.6 Sky Box

In the Unity setting, the appearance of the sky could be changed by switching sky box. In Fig. 7.30, (a) to (c) stand for different weathers, which are named as 'sunny', 'cloudy' and 'overcast' respectively.

Figure 7.30: Extracted Unity image with different sky box

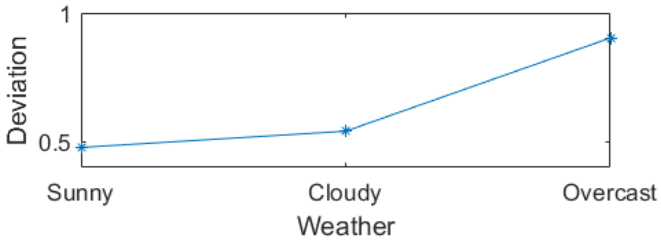

Figure 7.31: Target detection deviation with different sky boxes

Fig. 7.31 is the target car detection deviation, which illustrates that darker sky leads to a larger deviation. The main reason of this behavior might be the fact that darker sky has a closer color to the black target car, making the edge of target harder to be distinguished.

Fig. 7.32 shows the lane detection error and deviation, and the difference among these three weather conditions could be ignored. Thus, the sky box setting will not affect the lane detection significantly.

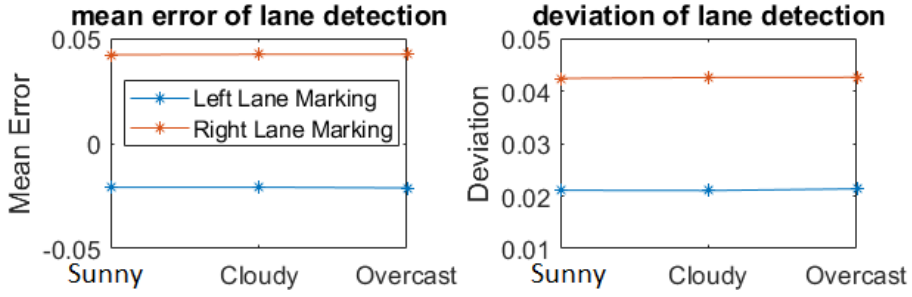

Figure 7.32: Lane detection error and deviation with different sky boxes

8

Conclusions and Future Work

8.1 Conclusions

It can be concluded that the virtual scenes developed by Unity and SPAS are feasible for the open-loop detection software, which shows the possibility to replace some real field tests with real-time and totally virtual tests. After matching the settings in both SPAS and Unity, the sensor data and the image data are generated respectively. A tool chain to extract and input data have been built and the process flow has been provided including virtual scene construction, parameter setting and analysis of detection behavior.

Apart from that, different parameters in the Unity software have been adjusted and the influence on the detection output has been analyzed. For lane detection, shadows covering could cause significant fluctuation at the beginning but it will be handled by the algorithm automatically. A longer distance to one lane marking will introduce larger lane detection deviation and mean error. Furthermore, camera posture will shift the relative lane markings' position on the extracted images, leading to a detection offset.

For target car detection, a light intensity factor at level 1 will always have the lowest deviation, i.e., the optimal combination of brightness and contrast. Larger light direction angle or darker target color will cause detection offset in the same direction. Vehicle overlap will cause a vehicle type detection error and the overlap for vehicles with different color could be accurately recognized at a longer distance.

Factors that have been tested and discussed are mainly related to light conditioning and might be the most important part of camera photography. The detection output analysis does not focus on the details of the performance of the open-loop detection software but instead tries to find out which parameters might be important for the future virtual test. In the virtual tests, all kinds of situations should be considered, especially extreme settings which might lead to remarkable detection errors. By developing algorithms and AD functions according to these extreme situations, safety and reliability could be improved effectively.

8.2 Future Work

In this thesis, the detection output is only compared with the initial settings of lane marking position and target car position, but more detection output parameters might be influenced by adjusting settings. Moreover, the controllable parameters in SPAS and Unity are limited, so not all factors in the real world could be taken into account. Thus, more analysis of the relationship between different settings and detected parameters could be done in the future.

Another way to assess the detection performance is to make comparison between tests with Unity image and V-Ray image. That is due to the fact that V-Ray could track every light ray as in the real world to generate more reliable and detailed images. Even though V-Ray might be too slow to achieve real-time test, it could still be utilized in the open-loop detection.

Besides, the next step to extend the achievement of this project could be performing tests with a more complex virtual environment. For example, a tilted and curved road can be considered, and extra target cars or pedestrians can be located on the road. If the detection error and deviation are still acceptable, this re-simulation tool chain could be tested with closed-loop AD functions, which means that the detection output could also influence the behavior of the host car. Unity has the potential to generate detailed enough images quickly for real-time closed-loop AD functions, which is likely to be achieved in the future.

Bibliography

[1] J. Liu, "Evaluation of closed-loop resimulation approach for virtual verification of active safety functions," Master's thesis, 2018.

[2] Wikipedia contributors, "Unity (game engine) — Wikipedia, the free encyclopedia," 2019, [Online; accessed 22-May-2019]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Unity_(game_engine)&oldid=898099090

[3] Y. Kuang and J. Jiang, "The designing of training simulation system based on unity 3d." *2011 International Conference on Intelligent Computation Technology Automation (ICICTA)*, p. 976.

[4] A. Morawiec, *Orientations and Rotations. [electronic resource] : Computations in Crystallographic Textures.* Springer Berlin Heidelberg.

[5] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis."

[6] Wikipedia contributors, "Focal length — Wikipedia, the free encyclopedia," 2019, [Online; accessed 21-May-2019]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Focal_length&oldid=893650138

[7] T. Norton, *Learning C by developing games with unity 3D beginner's guide. [electronic resource] : learn the fundamentals of C to create scripts for your GameObjects.* Packt Publishing.

[8] M. Maurer, J. C. Gerdes, B. Lenz, H. Winner *et al.*, "Autonomous driving," *Berlin, Germany: Springer Berlin Heidelberg*, vol. 10, pp. 978–3, 2016.

[9] E. J. Dasch, "instantaneous field of view." *A Dictionary of Space Exploration.*

[10] R. H. Parvin and W. E. Shephard, *Inertial navigation.*, ser. Principles of guided missile design. Van Nostrand.

[11] J. E. Stellet, M. R. Zofka, J. Schumacher, T. Schamm, F. Niewels, and J. M. Zollner, "Testing of advanced driver assistance towards automated driving: A survey and taxonomy on existing approaches and open questions." *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, 2015.

[12] Z.-W. Hong, C. Yu-Ming, S.-Y. Su, T.-Y. Shann, Y.-H. Chang, H.-K. Yang, B. Hsi-Lin Ho, C.-C. Tu, Y.-C. Chang, T.-C. Hsiao, H.-W. Hsiao, S.-P. Lai, and C.-Y. Lee, "Virtual-to-real: Learning to control in visual semantic segmentation," 02 2018.

[13] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator." 2017.

[14] S. Wang, Z. Mao, C. Zeng, H. Gong, S. Li, and B. Chen, "A new method of virtual reality based on unity3d," in *2010 18th International Conference on Geoinformatics*, June 2010.

[15] J. Gregory, *Game engine architecture.* Taylor Francis, CRC Press, 2018.

[16] C. Premebida, G. Monteiro, U. Nunes, and P. Peixoto, "A lidar and vision-based approach for pedestrian and vehicle detection and tracking," in *2007 IEEE Intelligent Transportation Systems Conference*, Sep. 2007, pp. 1044–1049.

[17] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, "3d traffic scene understanding from movable platforms." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 5, p. 1012.