HUMAN-VEHICLE COLLABORATIVE DRIVING TO IMPROVE TRANSPORTATION SAFETY

By

DUY TRAN

Bachelor of Science in Computer Engineering
Oklahoma State University
Stillwater, Oklahoma
2010

Master of Science in Electrical Engineering
Oklahoma State University
Stillwater, Oklahoma
2012

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
December, 2018

HUMAN-VEHICLE COLLABORATIVE DRIVING TO IMPROVE TRANSPORTATION SAFETY

Dissertation Approved:

Dr. Weihua Sheng

Dissertation Advisor

Dr. Yanmin Gong

Dr. Martin Hagan

Dr. He Bai

Name: Duy Tran

Institution: Oklahoma State University

Location: Stillwater, Oklahoma

Title of Study: HUMAN-VEHICLE COLLABORATIVE DRIVING TO IMPROVE TRANSPORTATION SAFETY

Pages in Study: 181

Major Field: Electrical Engineering

Abstract

This dissertation proposes a collaborative driving framework which is based on the assessments of both internal and external risks involved in vehicle driving. The internal risk analysis includes driver drowsiness detection, driver distraction detection, and driver intention recognition which help us better understand the human driver's behavior. Steering wheel data and facial expression are used to detect the drowsiness. Images from a camera observing the driver are used to detect various types of driver distraction by using the deep learning approach. Hidden Markov Models (HMM) is implemented to recognize the driver's intention using the vehicle's lane position, control and state data. For the external risk analysis, the co-pilot utilizes a Collision Avoidance System (CAS) to estimate the collision probability between the ego vehicle and other vehicles. Based on these two risk analyses, a novel collaborative driving scheme is proposed by fusing the control inputs from the human driver and the co-pilot to obtain the final control input for the vehicle under different circumstances. The proposed collaborative driving framework is validated in an Intelligent Transportation System (ITS) testbed which enables both autonomous and manual driving capabilities.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

In this work, a human-vehicle collaborative driving system, which combines the risk analysis of both the driver and the surrounding environment, is proposed to enhance the safety of transportation systems. This chapter gives an introduction to the motivation behind this work. The contributions of this work and the outline of the whole dissertation are provided at the end of this chapter.

I.1 Motivation

In recent years, the increasing number of vehicles on roads causes the increase of traffic accidents which leads to the rising number of fatalities. Globally, about 1.3 million people die in traffic accidents every year [1]. Human error is responsible in most traffic accidents. In 2013, according to the US National Highway Traffic Safety Administration, driving with drowsiness was responsible for 72,000 crashes, 800 fatalities and 44,000 injuries in the United States [2]. Moreover, in 2015, distracted driving was responsible for 391,000 injuries and 3,477 fatalities in the United States [3].

Intelligent Transportation Systems (ITS) have attracted more and more attention in recent years due to their great potential in enhancing traffic safety. As a part of ITS research, Google's self-driving cars [4] have been developed and have logged millions of miles on the roads in California [5]. Tesla tested its all-electric cars with autopilot capabilities under certain safety restrictions [6]. Many other car manufacturers are also interested in developing autonomous cars. Audi presented a prototype of a self-driving racing car using radars, lasers and cameras [7]. The car could finish a track lap

like a professional driver with a maximum speed of 150 mph. BMW partnered with Intel and Mobileye to create an open standards-based platform with a goal to bring self-driving cars to market by 2021 [8]. Ford has also been testing self-driving cars in less friendly environments, such as on snowy days or in poor lighting conditions [9]. Honda has received permission in California to test autonomous vehicles on public streets [10]. In July 2016, Jaguar Land Rover announced plans to deploy a fleet of at least 100 research vehicles over the next four years to test self-driving and connected car technology on roads in Britain [11]. Mercedes unveiled their concept F-015 autonomous vehicle, which the company said would be ready in 2030 [12]. Nissan has been testing an autonomous Nissan LEAF on the roads of Tokyo while promising that they would have vehicles with "significant autonomous functionalities" by 2020 [13]. In 2014, Toyota announced a $1B budget for autonomous driving research to establish its advanced Toyota Research Institute [14]. The company has targeted 2021 as a goal for deploying "AI car features". In 2015, Volkswagen announced that it would bring the autonomous driving technologies to market by 2025 [15]. Challenging BMW, Volvo also planned to sell a model that is able to pilot itself by 2021 [16].

With the potentials of self-driving vehicles, many non-vehicle-manufacturing companies have also joined the self-driving vehicle research. Apple has been testing its self-driving cars which are equipped with LIDARs, radars and cameras [17]. Nvidia introduced Nvidia Drive PX2 as a powerful computing platform for autonomous cars [18]. With 8 teraflops of processing power, the platform is robust enough to support deep learning and sensor fusion which are key elements of future self-driving vehicles. In May 2016, Uber, a taxi service provider, revealed its prototypes of autonomous vehicles [19]. Microsoft has also joined the self-driving vehicle research. Its deal with Volvo in November 2015 allowed Microsoft to develop autonomous vehicles by leveraging its HoloLens technology [20].

In 2016, the U.S. Department of Transportation (DoT) adopted the SAE (Soci-

ety of Automotive Engineers) International standard which uses six levels (0 - 5) of automation to categorize the autonomous vehicles [21]. While Google's self-driving cars are at the level 5 with fully autonomous capabilities in all driving modes (e.g expressway merging, high speed cruising, etc.), Tesla's auto-pilot cars [6] are at level 4 with fully autonomous capabilities in some driving modes. Similarly, car manufacturers like Toyota, BMW, Ford, Nissan, etc. [22] have also aimed to build up their own fully self-driving vehicles at either level 4 or 5.

Despite fully autonomous driving's great potentials for future transportation systems, mass deployment of such vehicles may still be years away. There are many impediments to the real-world deployment of fully autonomous driving which range from reliability to liability issues. Tesla cars also have reliability problems with their autopilot system. [23]. In June 2016, a Tesla car was involved in a fatal crash when the car was in auto-pilot mode [24]. The report indicated that the accident occurred when a tractor-trailer made a left turn in front of the Tesla car while the latter did not recognize the white side of the tractor-trailer against a bright sky and failed to apply the brakes. In addition, legal issues concerning the liability when such autonomous vehicles are involved in accidents have not been sorted out. This liability issue may make the automotive industry reluctant to manufacture driverless vehicles, even when the vehicles are reliable and robust.

It is believed that a collaborative driving system in which the vehicle runs under the control of both a human driver and a co-pilot system would be the best solution in the near future. Human drivers are more intelligent because they can assess the traffic situation precisely. However, they are inconsistent because of various distraction behaviors, including drowsiness. Co-pilot systems are more consistent in perceiving the surrounding environments than human drivers. However, co-pilot systems are not so intelligent in judgment, especially in situations that they have never seen before. Therefore, a human driver and a co-pilot complement each other. For a vehicle to

drive safely on the road, two risk factors must be considered: 1) internal risk factors which depend on the driver's behavior; 2) external risk factors which depend on the environment surrounding the vehicle. Most of the previous studies only address one of the risk factors. It is believed that taking both risk factors into consideration will make the vehicle drive more safely. Therefore, a driving assistance system that monitors and analyzes both internal and external risk factors is proposed. Such a new driving assistance system can help improve the safety of the transportation system. Here, we examines the two risk factors and find out how they are related to driving safety.

I.1.1 Internal risk factors

Internal risk factors are mainly caused by human driver's behavior. There are two main issues: driver intention and driver distraction. Driver drowsiness is a special type of distraction.

Driver intention

Vehicle crashes occur more often when driving maneuvers of one or more drivers are not announced properly to other vehicles. Therefore, driver intention recognition can provide useful information to nearby vehicles through vehicle-to-vehicle communication [25] to avoid traffic accidents. The built-in turning signal is a good source to determine other drivers' intentions. However, many drivers do not use the turning signal when they make turns or change lanes. A survey revealed that 57% of American drivers do not always use turning signals when changing lanes [26]. Thus, it is not sufficient to rely on turning signals for intention recognition. Several projects implemented driver monitoring systems inside the vehicle to observe the driver and infer his/her intention [27–29]. In this dissertation, a driver intention recognition system adopting Hidden Markov Models (HMM) [30] is presented. The author adopted

a multi-sensor fusion approach by using lane position, vehicle control data (steering wheel and pedal positions) and vehicle state data (velocity, acceleration and yaw rate) to obtain accurate predictions of the driver's intentions.

Driver distraction

In 2010, 18% of traffic accidents were distraction-related in the United States [31] with around 5000 fatalities and $40 billion in damage. There are many factors that may cause driver distractions. It is found that driver distractions are usually caused by sources from inside the vehicle [32]. As major motor companies like Toyota, Nissan, Ford, Mercedes-Benz, etc. gradually introduce advanced infotainment, control panel and display systems, adjusting those in-vehicle devices while driving could cause serious distractions that may lead to traffic accidents. Another source that can influence the driving performances is cellphone use. Driving and having conversations on cellphones each consumes a significant amount of the brain power. When doing both actions at the same time, it will consume 37% of the brain power [33]. It is not proved that using hands-free cellphones has safer driving performance than using hand-held cellphones. Text messaging while driving can raise even more distraction because it leads not only the driver's thought but also his/her hand and vision out of the driving tasks [33].

There are three types of distraction during driving [34]:

- Manual distraction: The driver takes his/her hands off the wheel, e.g., drinking, eating, adjusting music, etc.
- Visual distraction: The driver looks away from the road, e.g., reading, watching phones, etc.
- Cognitive distraction: The driver's mind is not fully focused on driving tasks, e.g., talking, texting, thinking, etc.

Drowsiness falls into both the second and the third types of distraction [35]. The many casualties and injuries caused by drowsiness call for an effective system that can detect drowsiness and take proper controls before accidents occur. The U.S. Department of Transportation has been promoting the development of intelligent vehicles in order to prevent such accidents [2]. In this dissertation, we aim to develop a driver drowsiness detection system which is used as an important component for the collaborative driving framework. Moreover, drowsiness detection is extended to distraction detection, which can be used in the collaborative driving framework in the future.

I.1.2 External risk factors

External risks to a vehicle can be attributed to many factors. Here, three main issues are focused on: lane departure, pedestrians and other vehicles.

Lane departure detection

Lane keeping is one of the essential skills that a human driver must have. To perform driving tasks, a human driver must analyze the road scene and choose suitable maneuvers. Therefore, the ability to analyze the road scene is a critical component in an Advanced Driver Assistance System (ADAS). The aim of lane detection is to ensure that the vehicle travels in the middle of the lane and raise alert in case there is any tendency of lane departure. Lane analysis includes the localization of the lane and the determination of the relative position between the vehicle and the lane.

In the real world, roads are categorized into two kinds: structured and unstructured. While structured roads usually have clear lane markings (solid or broken) and can be found on highways or urban streets, unstructured roads have vague or no painted lane markings. In this dissertation, only lane detection on structured roads, which reduces the complexity of the problem, is considered.

Pedestrian detection

Similar to lane detection, pedestrian detection is one of the most critical components in an ADAS. In urban areas, more pedestrians tend to cross the streets than in rural regions which may cause serious traffic accidents especially when vehicles are running at high speeds. In 2013 alone, 4,735 pedestrians were killed in traffic crashes in the United States [36]. Also in 2013, more than 150,000 pedestrians were treated in emergency departments for non-fatal-crash-related injuries [37]. Pedestrians are more vulnerable in vehicle-related accidents. It is found that pedestrians are 1.5 times more likely than passenger vehicle occupants to be killed in a vehicle crash [38]. Therefore, automated pedestrian detection may help the driver assistance system alert the driver, apply the brakes or perform necessary maneuvers to avoid collision with the pedestrian.

Similar to vision-based lane detection, vision-based pedestrian detection may encounter the following challenges. Pedestrian appearances may vary in clothes, height, poses, behaviors (walking, standing, running), carrying different objects, etc. Pedestrians may be partially occluded by other vehicles. Moreover, it is more challenging to detect moving pedestrians from a moving vehicle. In this dissertation, the pedestrian detection system could detect pedestrians with various appearances.

Vehicle detection

Vehicle detection is one of the core functions of any ADAS system. Other vehicles' behaviors play a major role in a human driver's decision-making process. Maneuvers of nearby vehicles, such as suddenly changing lanes, slowing down in the front or accelerating from behind may pose significant risks to a vehicle. Detecting other vehicles may provide an early warning or precaution about traffic situations. Therefore it is essential to explore techniques for vehicle detection and have appropriate actions to decrease the possibility of collisions.

Various sensing modalities have become available for on-road vehicle detection, including radars, lidars and cameras. Imaging technology has greatly progressed in recent years. Cameras are cheaper, smaller, and of higher quality than ever before. Concurrently, computing power has dramatically increased. Furthermore, in recent years, we have seen the emergence of computing platforms geared toward parallelization, such as multicore processing and graphical processing units (GPUs). Such hardware advantages make it possible to deploy computer vision approaches for real-time vehicle detection. In this dissertation, we aim to develop a vehicle detection system using cameras as in lane and pedestrian detection systems.

I.2 Objectives of this dissertation

In this dissertation, it is argued that an integrated manual and autonomous driving framework is more practical for real-world deployment compared to fully autonomous driving. First, a manual and autonomous switching framework based on driver drowsiness detection is developed. The driver's status is monitored by the drowsiness detection algorithm. If the driver is not drowsy, he/she can have full control of the vehicle. Otherwise, the vehicle can drive by itself. However, these conditions are only true if we are very sure of the driver's status such as drowsy or non-drowsy. The problem is how to choose a suitable control for the vehicle if we are not sure about the driver's status, or when the confidence regarding the status of the driver is not high. Therefore, to solve this problem, a collaborative driving framework which considers both internal and external risk factors is proposed. The proposed framework has two agents: a human driver and an autonomous agent (co-pilot). If the human intention agrees with the co-pilot's decision from the Collision Avoidance System (CAS) algorithm, the final decision is determined by fusing both control inputs from the human driver and the co-pilot. On the other hand, if the intention of the human driver and the co-pilot's decision do not match, the final de-

cision depends on the driver's status and the status of the ego vehicle. Besides the drowsiness detection system, a system which can detect driver distraction which is a general case of driver drowsiness is developed. This distraction detection system could be used for the collaborative driving framework in the future.

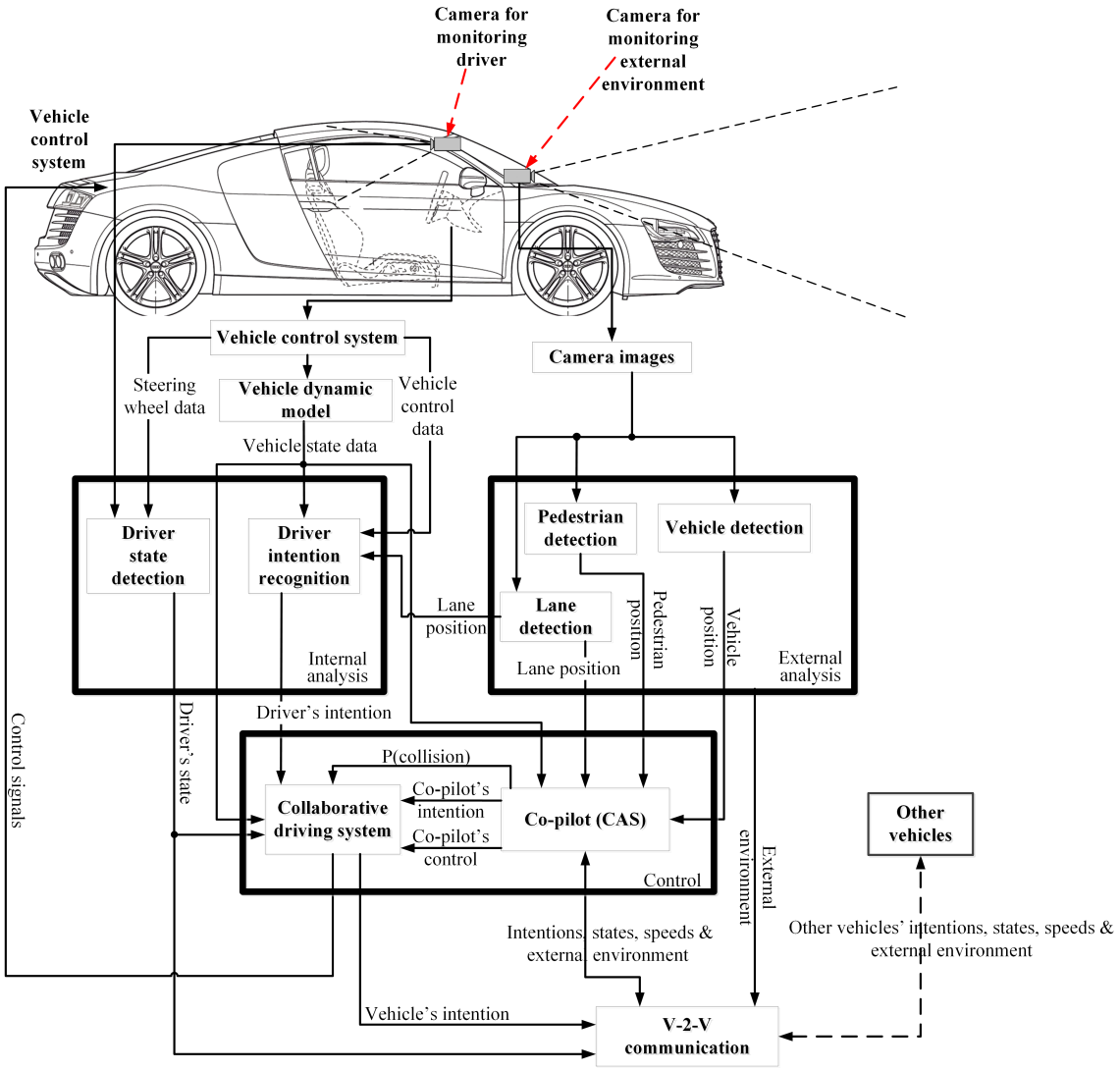


Figure I.1: The overall framework of the collaborative driving system.

I.3 Contributions

Figure I.1 shows the overall framework of the dissertation which can be divided into three main modules: internal risk analysis, external risk analysis, and control.

The internal risk analysis module aims to understand the driver's intention and state. The driver's intention is the potential action that the human driver may take in the near future. Typical intentions include changing lane left, changing lane right, speeding up, slowing down and keeping the vehicle's current state. The driver's states include drowsiness and distraction. The external analysis module aims to understand how the ego vehicle moves in the lane and how the nearby pedestrians and other vehicles move with respect to the ego vehicle. Such information is important to develop the collision avoidance algorithm. The control module consists of two parts: a co-pilot and a collaborative driving system (CDS).

The co-pilot utilizes the CAS which receives the collected information from the external analysis module and the intentions from other vehicles to make a decision regarding the best maneuver that the vehicle should take. The CDS module calculates the best action by integrating the outputs from the CAS module and the outputs from the internal analysis module. The human driver's intention and the co-pilot's intention are compared. If they agree, the final control for the ego vehicle is the fusion of the controls from the human driver and the co-pilot. Otherwise, the final control relies on the driver's state or the probability of drowsiness and the ego vehicle's probability of collision. The decision from the CDS module is given as the final command to the vehicle control system. The contributions of this work are summarized as follows.

1. A driver intention recognition system which uses the steering wheel, gas, brake and lane information was developed. Hidden Markov Models are adopted to classify five intentions (slowing down, speeding up, changing lane left, changing lane right and keeping the vehicle's current state). While most studies do not address curved roads in their recognition system, we found out that curved roads also play a major role in intention prediction. Unlike the studies of Doshi [29] and Henning [28], where the structure (curvature metric) of the road is treated

as input, in this dissertation, the proposed system does not take this feature as input but is still able to classify the changing lane maneuvers on curved roads.

2. An algorithm to detect driver drowsiness, which is a special type of distraction was developed. Tadasse's work [39] is adopted to achieve this task. However, in [39], the face detection relies on the CAMShift algorithm [40] which tracks the face based on the skin color. Therefore if the background has a similar color to the skin, the face would not be detected correctly which leads to miss-classifications. A new face detection algorithm is implemented. It is based on Histogram of Oriented Gradient (HOG) feature combined with a linear classifier, an image pyramid, and a sliding window detection scheme [41]. We also collected data and trained a new Support Vector Machine (SVM) model from more subjects with different ages, skin colors and genders to enhance the robustness of the proposed system. The drowsiness detection system was evaluated through a manual and autonomous driving switching control mechanism.

3. A driver distraction detection system which identifies various types of distractions through a camera observing the driver was proposed. An assisted-driving testbed is developed for the purposes of creating realistic driving experiences and validating the distraction detection algorithms. We collected our own dataset which consists of images of the drivers in both normal and distracted driving. Four deep Convolutional Neural Networks (CNNs) including VGG-16 [42], AlexNet [43], GoogleNet [44] and ResNet [45] are implemented and evaluated on an embedded GPU platform. In addition, a conversational warning system that alerts the driver in real time when he/she does not focus on the driving task was developed. The proposed distraction detection algorithm could be used for the collaborative driving framework in the future.

4. A new Collision Avoidance System (CAS) based on Osipychev's work [46] was

proposed. The new CAS takes the probability distributions of other drivers' intentions as inputs to cover all the possibilities of their intentions. Moreover, the vehicle's external environments such as lanes, pedestrians and other vehicles are also fed to the proposed CAS so that it can generate appropriate actions to assist the human driver.

5. A collaborative driving framework which considers both internal and external risk analyses was proposed. The framework takes inputs from the human driver and the co-pilot. The co-pilot always operates in parallel with the human driver but only intervenes in the control of the vehicle in certain situations based on the vehicle's status, the human driver drowsiness and his/her drowsiness intensity.

6. In order to conduct such research, it is necessary to have a testbed which can be used to verify the proposed methodologies. Since full scale experiments using real vehicles are expensive and risky, it is believed that a small scale physical testbed and a simulated testbed can be useful for preliminary study and feasibility test. The testbed can simulate real traffic environments, human driving experience and autonomous driving. The ITS testbed also provides functions that allow users to remotely access it.

I.4 Outline

The rest of this dissertation is organized as follows. In Chapter II, the literature review related to the proposed works is provided. Chapter III describes the development of the ITS testbed. Chapter IV presents the driver intention recognition system. Chapter V describes the driver drowsiness detection system. The distraction detection system is presented in Chapter VI. The CAS is presented in Chapter VII. Chapter VIII describes the proposed collaborative control framework. The conclusion and some future research directions are presented in Chapter IX.

CHAPTER II

RELATED WORKS

In this chapter, the existing works related to this dissertation are reviewed. This chapter is organized as follows. Section II.1 gives an introduction to internal risk analysis which includes driver intention recognition, driver drowsiness detection and driver distraction detection. Section II.2 presents the related works on the external risk analysis which includes lane and pedestrian detections. The literature review on CAS is discussed in Section II.3. In Section II.4, the related works on the collaborative control framework are presented.

II.1 Internal analysis

In this dissertation, the internal risk analysis includes the recognition of the human driver's intentions and the detection of his/her state (distraction or drowsiness). This sub-section gives a brief overview of the related studies in these areas.

II.1.1 Driver intention recognition

In recent years, many techniques and approaches have been implemented to recognize the human driver's intention. For example, Support Vector Machine (SVM) was used to classify the driver intention of lane changing [27]. Eye movement, steering angle, gas pedal pressure, and vehicle states including velocity, acceleration and yaw rate were used as inputs. The method could achieve an accuracy of 88.78% with 0.6 seconds of recognition time at a 5% false alarm rate. Moreover, SVM combined with Bayesian Filtering (BF) was also implemented for driver intention recognition [47].

This technique adopts distance between vehicles, heading angles, and speed as input features to achieve an accuracy of 77% and a 0% false alarm rate. The same authors also implemented a modified SVM-BF algorithm with discounted BF to increase the accuracy up to 90% with a 7% false alarm rate. Sparse Bayesian learning technique was also implemented for driver intention recognition in a research by McCall *et al.* [48]. The input features are the vehicle's control and state data from the CAN bus, lane position and head motion.

Driver intention recognition using facial data also attracted many researchers [27, 48]. Doshi *et al.* proposed a recognition method using the driver's eye gaze and head motion [29]. However, they adopted a different classification method called Relevance Vector Machine (RVM). Their method achieved an accuracy of 79.2%. Henning *et al.* also used facial data with adaptive interface technology to model driver's behavior without turning on the signal lights [28]. Their study shows that the glancing at the side-mirrors can yield earlier prediction than using signal lights. Overall, the facial expression is a good source of data for intention prediction. However, it is not reliable under weak lighting conditions [48].

The most widely used method in driver intention recognition is Hidden Markov Model (HMM). Liu and Pentland presented an HMM-based real-time recognition system [49] with 50% and 60% of accuracies in predictions of lane changing and turning respectively. Jin *et al.* introduced a lane changing recognition system using HMM [50] which takes both vehicle control and state data as observation inputs. However, their system does not handle curved roads. Ding *et al.* introduced an HMM method which uses Comprehensive Decision Index (CDI) as inputs [51]. This index reflects the influence of surrounding traffic to the driver's lane changing decisions. The results of their paper show that implementing the CDI index could yield better accuracy in prediction. While these papers mainly focus on lane changing actions and have some limitations in accuracy and recognition time, more maneuvers can be

predicted in this dissertation.

In this dissertation, maneuvers which include changing lane left, changing lane right, speeding up, slowing down and keeping the vehicle's current state are considered in the intention recognition system. The performance of the system is examined in three scenarios: using only the vehicle control data (steer, gas and brake pedal positions); using both vehicle control data and vehicle state data (velocity, acceleration and yaw rate); and using all the data including vehicle control data, vehicle state data and lane position. While most studies do not mention curved roads in their recognition system, we figured out that curved roads also play an important role in intention prediction. Unlike the studies of Doshi [29] and Henning [28], where the structure (curvature metric) of the road is taken as input, in this dissertation, the system does not take this feature as input but is still able to classify the lane change maneuvers on curved roads.

II.1.2 Driver state detection

The driver's state can be considered as his/her distraction status. As mentioned earlier, there are three types of distraction during driving: manual, visual and cognitive distractions. In this sub-section, the author gives literature reviews of driver distraction detection as well as its special example which is driver drowsiness detection.

Driver distraction detection systems

There are three main systems used to recognize driver distraction:

- Systems based on physiological data: (EEG, ECG). Brain activity or heart rate is used as a feature vector to measure driver inattention/distraction [52, 53]. This kind of systems can provide very accurate and fast results. However, it does not have any real practical application in the driving environment. Drivers have

to wear many physiological sensors around their bodies. This would interfere with their maneuvers while driving.

- Systems based on vehicle control data: (steering wheel movements and pedal positions). Jin *et al.* introduced a system that can detect cognitive distraction using only driver performance [54]. They claimed that distracted drivers usually steer the steering wheel and apply the pedals differently than normal drivers. They also mentioned that during distraction, drivers tend to increase the distance to the leading vehicle or tend to drive faster than normal with no leading vehicle. In 2008, Ranney stated that distraction would lead to a lack of control which may cause the vehicle to drift outside of the road [55]. This approach can provide an accurate detection. However, it has low detection speed since it requires long-term statistics.

- Systems based on computer vision: (driver's face, eye and body movements). This is the most popular approach in detecting driver distraction. The driver's face and eye images can help infer his/her state. Pooneh *et al.* used PERCLOS (Percentage of Eye Closure) to detect driver's drowsiness which is a kind of distraction [56]. Infrared (IR) cameras are used to detect the eyes using the difference between bright and dark pupils [57–59]. Driver expression such as yawning is also utilized to detect distraction [58, 60]. In addition, the driver's right hand can also be monitored for distraction detection [61].

Driver distraction detection methods

There are various methods for driver distraction detection. The simplest method for distraction detection is thresholding, in which a certain feature value is compared with a preset threshold. Tabrizi *et al.* detected drowsiness by applying a constant threshold to the PERCLOS value [62]. On the other hand, various machine learning

techniques have been implemented to detect driver distraction. Support Vector Machine (SVM) using vehicle control data as feature vectors was implemented by Jin *et al.* to detect cognitive distraction in [54]. In [63], the eye movements were utilized as features in the SVM classification of distraction. SVM was also used to detect distraction based on vision and lane tracking data [64]. Cray *et al.* proposed a Hidden Markov Model (HMM) based method to detect distracted driving activities. Their method requires the detection of the driver's face and right arm [61]. Considering the inherited uncertainty associated with the face features, Bayesian networks are used by Gu and Ji [60, 65] to model the uncertainty and determine the probability of distraction of the driver.

Deep learning has gained more attention recently in distraction detection. In 2016, State Farm started a competition on Kaggle.com with the goal to detect distracted driving based on a provided dataset of dashboard camera images that showed drivers either engaging in distracted behaviors or driving safely [66]. Convolutional Neural Networks (CNNs) have achieved top performance in the competition [66]. For example, Colbran *et al.* adopted a VGG-16 model that achieves an accuracy of 80% [67]. The AlexNet models were used in [68, 69]. Based on the State Farm dataset, Lőrincz *et al.* utilized region-based CNNs to detect the driver's head pose, hands and classify the object that the driver is holding and thus detect distraction behaviors [70]. Inspired by the State Farm competition, Abouelnag *et al.* provided a dataset and proposed a real-time distraction detection approach using a combination of 5 AlexNet and 5 GoogleNet models with hand, face, and skin features [69]. A weighted genetic algorithm (GA) ensemble of the CNNs is used to merge the CNN results to obtain the final result. The system achieved a 95.98 % driving posture classification accuracy. However, Abouelnag's approach requires a powerful computer to perform the calculations of the 10 CNNs in real time which is not feasible on embedded systems yet. Therefore, this dissertation proposes an approach of using a single CNN to ensure the

efficiency of computation on embedded systems. Choi *et al.* proposed a deep learning approach in real-time distraction detection based on eye gaze [71]. The driver's face is recorded and a Haar feature based face detector is combined with a correlation filter based MOSSE (Minimizing the Output Sum of Square Error) tracker for face tracking. Then the AlexNet model is utilized to classify the 9 gaze zones. In addition, deep learning techniques can also estimate the driver's head pose which is useful for distraction detection. Venturelli *et al.* proposed an approach which uses a simple CNN to estimate the head pose such as yaw, pitch and roll angle through a depth camera [72]. From the face image, a linear interpolation algorithm is used to remove the background. The result is fed to a CNN which has 5 convolutional layers and 3 fully connected layers to estimate the head pose. This network takes 64×64 images as input, which is relatively smaller than other deep learning architectures. Unlike the approaches in [71, 72] which focus on monitoring the face from a frontal camera, the proposed system focuses on monitoring the driving behaviors from the whole human body. Therefore, these approaches could be combined with the proposed one to enhance the robustness in the future. Hssayeni *et al.* compared the performance of deep learning approaches versus a traditional classification algorithm such as SVM with Histogram of Gradient features [73]. They also proposed an approach which combines both deep learning and traditional classification algorithms. In their approach, the outputs from the convolutional layers are used as features for an SVM classifier.

In this dissertation, the author aims to develop a real-time detection system of distracted driving which is achieved by utilizing four deep CNN architectures including VGG-16 [42], AlexNet [43], GoogleNet [44] and ResNet [45]. In addition to the existing public dataset, we collected our own dataset using an assisted-driving testbed. The performances of the four networks are evaluated and compared in terms of both accuracy and realtimeness to identify the algorithm that is best suited for real-time implementation. Distracted driving detection is performed in real time on

an embedded system which generates voice alerts to the driver if the distraction is detected.

Driver drowsiness detection

A special type of distraction is drowsiness in which the driver has fatigue and looses focus on the driving task. Similar to distraction detection, real-time drowsiness detection has been implemented through different detection techniques analyzing different types of input data e.g., physiological data, vehicle control data and driver's facial images. The first set of techniques makes use of the measurement of variations in the physiological activities of the human body such as brain wave (EEG), heart rate or pulse rate [74–78]. Even though the measurements and their correlation with the alertness of the driver are quite accurate, they are not practical because the driver is required to always wear the sensing devices and the hardware cost is too high to be used for commercial purposes. Moreover, wearing these kinds of sensors would interfere with the movements of the drivers' hands while driving. The second approach is analyzing the vehicle control data such as steering wheel and gas pedal position to detect the drowsiness of the driver [79–83]. The third approach, the most popular one, is making use of computer vision and image processing to detect the drowsiness of the driver. Researchers have mainly focused on the analysis of the driver's eye blinks and PERCLOS to determine his/her drowsiness [84–87]. In recent years, analyzing facial expression has attracted more attention. Vural *et al.* employed machine learning methods to analyze facial motions from a video [88]. The changes in the driver's facial expressions such as eyebrows, mouth, and wrinkles are utilized to detect his/her drowsiness [89, 90]. The different approaches mentioned above have their own advantages and limitations. The method using vehicle control information is non-intrusive but lacks reliability. The method utilizing measurements of physiological information is reliable and accurate but is intrusive. On the other hand,

the method using computer vision to analyze the driver's facial expression is more reliable than the one using vehicle control information. However, its performance is affected by ambient illumination. So, utilizing either of these methods may not be sufficient to determine the driver's state under different circumstances. Therefore, the fusing method is a good choice to improve detection reliability. Eskandarian *et al.* detected driver drowsiness by combining steering wheel movements and facial data using Neural Networks [91, 92]. In this dissertation, the author implements both the analyses of the steering wheel data and the vision-based facial expression which are then integrated to give a final decision on the state of the driver.

II.2 External analysis

The external risk analysis includes the detection of lanes, pedestrians and other vehicles. This sub-section summarizes the literature reviews of those systems.

II.2.1 Lane detection

There are many works on lane detection in recent years. They generally utilize different strategies to deal with certain kinds of surroundings and road conditions. In general, there are two types of sensors for lane detection: LIDARs and cameras. LIDAR-based lane detection attempts to detect the lane markings based on an increase in reflectivity of the lane markings when compared to the road surface [93]. LIDARs can be used to identify lane marks and road boundaries by their 3D extensions above the road surface [94, 95]. LIDARs can be used to estimate the ground roughness that is important for road and off-road segmentation [94, 96, 97]. LIDARs are also helpful in detecting road curbs and berms [95, 97]. The video-based method attempts to detect lane markings by detecting the features of the lane markings in a sequence of images. Color segmentation is used to detect the road [98]. The segmentation is performed by the ISODATA clustering algorithm on the hue and saturation

distribution (2D histogram) of the given image. Dahlkamp *et al.* introduced an approach in which the system identifies the drivable surface in desert terrain [99]. A mixture of Gaussians - MOG (Modified Gravity) model was used to describe the road surface. Gao *et al.* used HSV colors from road images as the main feature for detecting rough road surface [100]. In recent years, edge is one of the most common features used in road detection for structured roads. Edge-based methods use the edge information extracted from the road image to obtain road or lane-marking candidates. There are several famous techniques in edge detection such as Canny filters [101–104] and Sobel filters [105–108]. After the edge map of the image is obtained, the Hough transform is utilized to get line segments that possibly represent the lane markings.

II.2.2 Pedestrian detection

In their earlier work, Shashua *et al.* [109] proposed a sped-up method of HOG for characterizing spatially localized parts to model pedestrians. Since their introduction, the number of variants of HOG features has improved greatly with nearly all modern detectors utilizing them in some forms. The shape features of human body are also a frequent cue for detection. Boosting was used to learn head, torso, leg, and full body detectors. This approach was extended in [110] to handle multiple viewpoints. Similarly, Boosting was also used to combine multiple shapelets which are shape descriptors and discriminatingly learned from gradients in local patches, into an overall detector. Wojek and Schiele stated that using a combination of Haar-like features and shapelets to detect pedestrians was better than using an individual feature [111]. Wang *et al.* [112] combined HOG features with a texture descriptor based on local binary patterns (LBP) [113]. Additionally, a linear SVM classifier was modified to handle occlusions. Schwartz *et al.* [114] represented pedestrians by edges, texture, and color while applying partial least squares to project the features down to a lower dimensional space prior to the SVM training. Dollár *et al.* [115] proposed a method

of computing Haar-like features over multiple channels of visual data, including LUV color channels, grayscale, gradient magnitude, and gradient magnitude quantized by orientation. This approach provides a simple and uniform framework for integrating multiple feature types. Later this approach was extended for fast pedestrian detection [116]. In 2014, Dollár *et al.* [117] proposed a method that used Aggregated Channel Features (ACF) to form fast feature pyramids and detect pedestrians. Since this approach satisfies the fast processing requirement, it is implemented in the pedestrian detection system of this dissertation.

II.3 Collision Avoidance Systems (CAS)

Among many different approaches to driving assistance, one of the most popular approaches is reactive safety which warns the driver about risks on the road or even takes actions to avoid accidents [118, 119]. With the help of modern sensors and fast computers, these systems can surpass the human in reaction time and prevent up to 80% of simulated collisions [120, 121]. Pomerleau *et al.* introduced an advanced robotic system called ALVINN which utilizes the images from cameras and neural networks for reactive control [122]. Goodrich *et al.* proposed a collision avoidance system based on optical pattern growth rate. This system applies the brake when the size of the leading vehicle grows on the image from a front view camera [123]. Maile *et al.* proposed a system that could alert the driver and autonomously apply the brake when a high risk of collision at an intersection is detected [124]. Jimnez *et al.* introduced a CAS based on a laser-scanner to keep the vehicle away from the potential danger [125]. To improve the safety in reactive systems, we need to enhance their sensitivity. However, too much sensitivity may lead to the increase of false alarms and create annoying actions to the human driver when there is no danger.

Despite the reactive control, an autonomous vehicle can be equipped with a proactive control system which plans its action ahead with respect to the positions, be-

haviors and intentions of surrounding vehicles. This proactive approach helps achieve a higher sensitivity to a potentially dangerous situation while maintaining softer actions which would not annoy the human driver. However, it is still a challenge for the adoption of these methods in vehicles due to the following difficulties. First, a system that plans actions in advance requires the information of the human driver's intention. This task has been previously solved using machine learning algorithms which analyze human activity and output the intention or the warning about possible unintended actions due to drowsiness or distraction [39, 126]. The second challenge is the modeling of the human behavior which is utilized to predict the trajectory of the vehicle. This task can be solved by learning-based behavior models such as Gaussian model [127, 128]. The third challenge in developing a proactive algorithm is the evaluation of the corrective action based on the possible outcomes. This task could be performed as an optimization task using various single-step and sequential decision-making techniques such as tree search [129]. However, using a tree search requires many evaluating functions which may slow down the decision-making. Another common way of solving this task is using the potential fields [130]. The principle of this method is to represent the dynamic obstacles as objects with potential fields which force the autonomous vehicle to move away from them. However, this method assumes the use of the continuous model of the world which has large variations, especially in unknown areas. A better solution to collision avoidance is using a sequential Markov Decision Process (MDP) [131] and a Partially Observable MDP (POMDP) [132] in which hidden intentions affect obstacle's behaviors and transitions. The world is assumed to be partially observed or completely hidden, and the motivation and dynamics of the processes are not available while only the effects of certain actions can be observed [133, 134]. In theory, these methods would give an elegant solution. However, they are very hard to solve, require many samples to establish the hidden links, and are hard to check the correctness of the solution. In this dissertation, the

single-step cost function optimization is implemented to solve the collision avoidance problem. The system allows us to find the best actions given the full knowledge of the parameters of velocity, direction, position and intentions for all surrounding vehicles.

II.4 Collaborative control

Collaborative control has been studied in robotics for some time. A collaborative control framework allows the human and the robot to share the control to perform a certain task [135–138]. Fong et al. proposed a collaborative control system in which the robot is treated as a peer and could respond to human requests [139]. Carlson et al. presented a collaborative control approach to assist powered wheelchair users as they require assistance [140]. The system can predict the user's intentions through the control joystick on the wheelchair and adjust the signals if necessary to reach the goal safely. Macharet et al. proposed a collaborative driving framework for a telepresence robot in which the user's control input is treated as a general guidance for the robot to reach the target [141]. The robot processes this information by coupling it with a variable degree of automation depending on the task to obtain a suitable control input.

In recent years, collaborative driving has attracted great research attention. In the 2007 DARPA Urban Challenge, Wei et al. introduced a multi-level collaborative driving framework for their autonomous vehicle [142] which includes different levels of human driver involvement. An interface was created for the driver to control the vehicle by cooperating with the vehicle's intelligence system. Gillham et al. employed a framework to assist the human driver by generating the smoothest trajectory and applying the final control to the steering [143]. Their system tries to keep the human driver at full control as much as possible. Zimmermann et al. proposed multimodal interaction and a user interface for a cooperative lane-changing scenario in a highly automated driving environment [144]. Flemisch et al. proposed a framework of coop-

erative guidance and control for vehicles using the concepts of 'Conduct-by-Wire' and 'H-Mode' [145]. With 'Conduct-by-Wire', the driver can delegate maneuvers to the automation with a specialized maneuver interface. On the other hand, the 'H-Mode' makes use of haptic-multimodal interaction with highly automated vehicles based on the H(orse)-Metaphor. The cooperation is mainly carried out with a haptic interface. Similarly, haptic human-machine interfaces were also used in many other studies of collaborative driving of vehicles. Brandt et al. proposed an approach which combines steering wheel force-feedback with a potential field path-planner for lane-keeping and collision avoidance [130]. In their work, the human driver's control input must follow exactly the desired control from the path planner so that the control assistance would not be applied. Otherwise, the driver can only control the vehicle indirectly and his/her control input is always fused with the desired control from the path planner to follow the planned path. Similar to [130], Li et al. presented a cooperative driving paradigm in which the driver can only control the vehicle indirectly through the automation's input transformation [146]. The 'Steer-by-wire' technology is applied so that the vehicle can correct the driver's steering input. Unlike [130] and [146], the proposed system grants the driver more authority on controlling the vehicle, and the driver's control is only modified by the co-pilot in certain situations. An extended work by Li et al. utilizes adaption and trust models which could grant the steering control authority to the driver if he/she distrusts the automatic controller [147]. In [148] and [149], the H_2-optimization algorithm is utilized to share the steering wheel control with the driver in the best possible way. Soualmi et al. presented a Low-Level-Cooperation (LLC) system for the purpose of lane keeping. This system shares the vehicle's lateral control using the T-S controller with the human driver's steering wheel control, when obstacles on the road are not detected by the Electronic copilot (E-copilot) [150]. Unlike these studies which only focus on the shared control of the steering, this dissertation aims to share the control of both the steering and

the gas/throttle between the human driver and the co-pilot.

Most importantly, while these studies ([130, 143–150]) do not monitor the human driver, the proposed system achieves a high-level cooperation which not only monitors the driver but also utilizes his/her state as an input to the collaborative driving framework. In the proposed system, the human driver's intention is compared with the decision from the co-pilot running the CAS algorithm. If they match, the system assists the driving task by combining the control inputs from both the human driver and the co-pilot. If they do not match, additional conditions such as the vehicle's status and the driver's drowsiness status are taken into account to determine the control of the vehicle.

In this dissertation, the human driver's intention is compared with the decision generated by the CAS algorithm [46]. If they match, the system would assist the driving task by combining the control inputs from both the human driver and the CAS algorithm. If they do not match, additional conditions such as the vehicle's status, the driver drowsiness and his/her drowsiness intensity are taken into account to determine the control of the vehicle.

CHAPTER III

THE ITS TESTBED

Conducting full scale ITS research with real vehicles has many challenges. Modifying a real vehicle into a semi or fully autonomous one can cost a significant amount of money and time. A failed experiment can cause great risks to the drivers, vehicles or nearby people. Hence, a small scale ITS testbed is preferred for initial study and reliability tests before conducting the experiment in real traffic. This chapter presents the development of the small scale ITS testbed which comes in two types: physical and simulated testbeds.

III.1 The physical testbed

The physical testbed has two main parts: hardware and software. In this subsection, the physical testbed's hardware and software setup are described as well as its remote configuration capability.

III.1.1 Hardware setup

The ITS testbed includes an arena, an indoor localization system, automated radio-controlled (RC) cars, roadside monitoring facilities, a server and clients. To mimic typical traffic environments, an arena with a wooden floor, mock buildings and streets was built. An indoor localization system is developed to mimic the GPS tracking in the real world. Automated RC cars with both autonomous driving and human driving capabilities are also developed. For the roadside monitoring facilities, an overhead fish-eye camera is used and the associated advanced video processing

algorithms are developed. The server is the testbed's center which can collect data and control the RC cars. The client can access the server, configure and perform experiments remotely. The overall testbed is shown in Figure III.1.

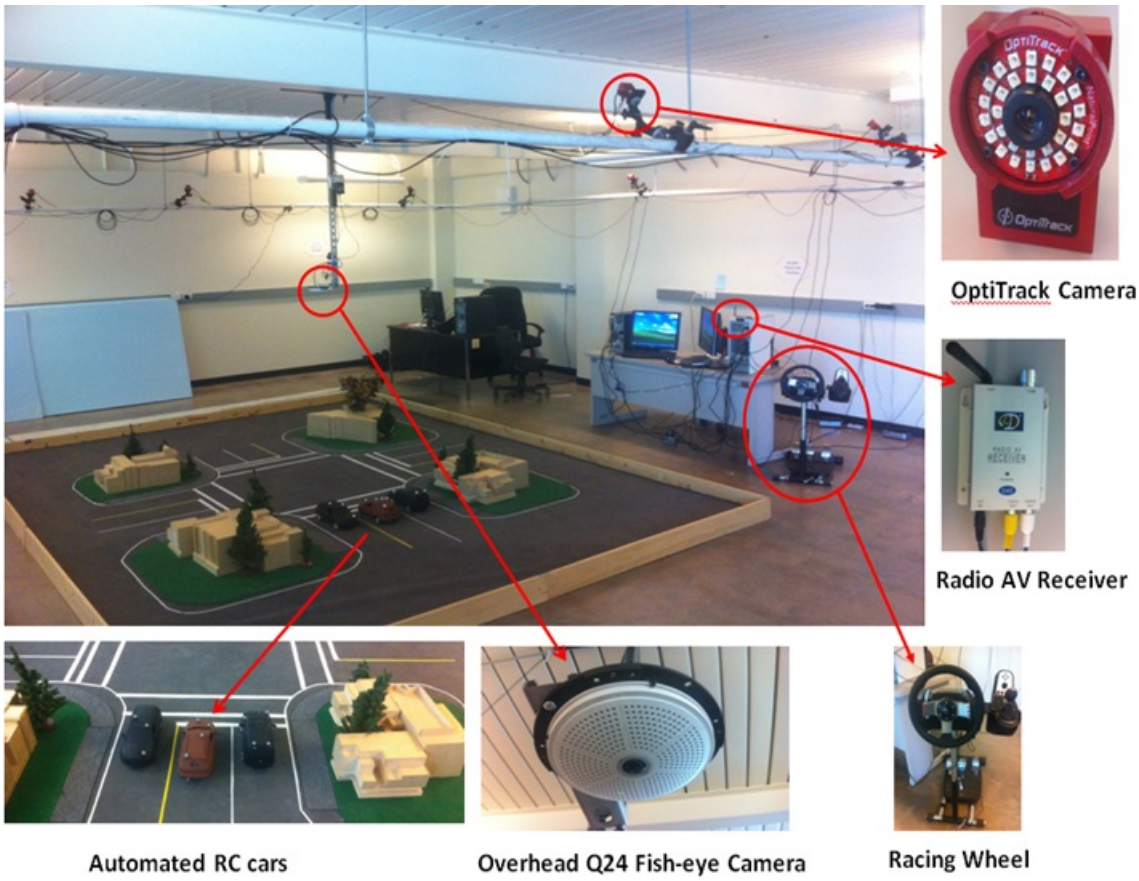


Figure III.1: Hardware setup of the physical ITS testbed.

Arena

The arena is a wooden board with a dimension of 16 feet by 12 feet. There is a 6-inch high wooden fence placed around the arena, in order to prevent the RC cars from running outside the arena. The arena is covered by a gray carpet which imitates the friction of real concrete or asphalt roads. A small scale intersection was developed at an approximate ratio of 1:14 (the ratio of an RC car to a real car). This intersection also has crosswalks, street lanes, sidewalks, and turning curves. It

has three driving lanes in each direction including two lanes crossing the intersection from both directions and one left-turning lane. The lane marks are made of yellow and white tapes. Building blocks, green carpets and trees are decorated to create a realistic driving environment. Building blocks do not have the 1:14 ratio since it would be too big and space consuming. The size information of crosswalks, lanes, sidewalks and curves is obtained from the website of the United State Department of Transportation [151–153].

Indoor localization system

The Opti-Track system manufactured by the Natural Point Inc. [154] is used to localize the RC cars. This system consists of twelve cameras placed around the arena to estimate the locations of the markers placed on top of each RC car. Each camera transmits IR beams and receives the signal reflected by the silver markers so that it can localize these markers. Each camera can detect a marker with an accuracy within sub-millimeters at a frame rate of 100 fps [154]. Each RC car, represented by a rigid body, has a specific placement of makers on its top so that it can be correctly identified. Each camera can localize markers within a range of 18 to 433 inches. The optimized area which these cameras can cover on the arena is approximately 12×12 feet of the arena.

Automated radio-controlled (RC) cars

The automated cars are developed using commercial off-the-shelf RC cars which have two DC motors: a front DC motor for steering control and a rear one for speed control. After testing the front DC motor, it has very poor steering performance and could not be used in this study. Therefore, the front DC motor is replaced by a servo motor for more accurate steering. The original control board was also replaced with another embedded board to control both motors. An XBee wireless

module is attached to this board to receive commands from the server computer. The RC cars are equipped with both autonomous and human driving capabilities. In the autonomous driving mode, the concept of "a virtual vehicle" [155] which is generated to run along a predefined trajectory is adopted. The server is programmed to wirelessly control the RC car to track this virtual vehicle. The human driving setup as shown in Figure III.2 can partially mimic the human driving experience. A miniature wireless camera is mounted on the hood of the RC car to provide the driving view. It is used to observe the environment in front of the car and send the video stream through the radio AV receiver connected to the server. The human driver controls the Logitech G27 steering wheel system and drives the RC car while he/she observes the video stream on the monitor. Information such as steering angle, gear, gas and brake pedal position is collected, encoded and sent by the server to the RC car so that it can drive following the control of the human driver.

A VN-100 sensor [156] was also mounted on the top of the RC car to monitor its motion. This is an integrated sensor package that combines multiple accelerometers and gyro-meters to produce three-dimensional measurements of both orientation, acceleration and angular rate with respect to an inertial reference frame [156]. The motion data is sent to the server through another XBee module for further processing. More details about the control of the RC car can be found in our previous works [39, 155].

Roadside monitoring facilities

A Mobotix Q24 fish-eye camera (Figure III.1) is mounted over the arena to monitor the intersection. This camera can support continuous images with a panoramic 360-degree view and no blind spot [157]. This camera can record videos with a 3MP resolution at 20 fps. It can also serve as an IP camera which lets other computers access its video stream. If the network connection is lost, the Q24 camera keeps

Figure III.2: Hardware setup of manual driving.

recording and storing videos to its internal memory. The camera view, frame rate, resolution, etc. can be adjusted through the software called MxControlCenter [158].

Server

The server is a computer that runs all software to control all components in the ITS testbed. The indoor localization system is connected to the server via USB ports. The server runs the ARENA software [154] to collect RC cars' locations. There are two XBee modules connected to the server. The RC car is controlled by the server in either autonomous or manual driving mode through one XBee module. The other one is used to receive data from the motion sensor attached on top of the RC car. A webcam is connected to the server to capture the driver's face. The server can then send the collected data and images to a remote client application once being requested.

Clients

The client can be any computer running the client program in any physical location. The client accesses the testbed by communicating with the server via TCP/IP. It can configure/request which data should be received and draw a trajectory along which the RC car in the testbed can track. The client can also utilize the received data to perform drowsiness detection (based on the data from the steering wheel and the webcam) or anomalous driving detection (based on the Q24 camera's video data).

III.1.2 Software setup

All the devices in the physical testbed are controlled by the server software running on the same computer. This server software can run the ARENA software to manage the Opti-Track system, control the RC cars, and perform data collection.

ARENA software

This software is provided by the Natural Point Inc. [154]. It is used to manage the Opti-Track system. The data obtained from the Opti-Track system can be a rigid body's unique ID, position (x, y, z), rotation in quaternions (qx, qy, qz, qw) and skeleton data which is a named hierarchical collection of rigid bodies. In this dissertation, rigid bodies are utilized to represent RC cars. Therefore, we just need the position and rotation data. The ARENA software also has an ability to stream its collected data to a client. There are many methods for collecting data from the ARENA server. We can use Direct Depacketization Client for Unix, Python or Java clients. We can also use the NatNet SDK library provided by the Natural Point for C++, C#, VB or .NET clients [154]. In this dissertation, the client is implemented in C++ using the NatNet SDK library. The server plays as a client of the ARENA software to collect the rigid body's data. Thus they operate on the same computer and use the local IP address (127.0.0.1) for communication. The data collected by the server enables

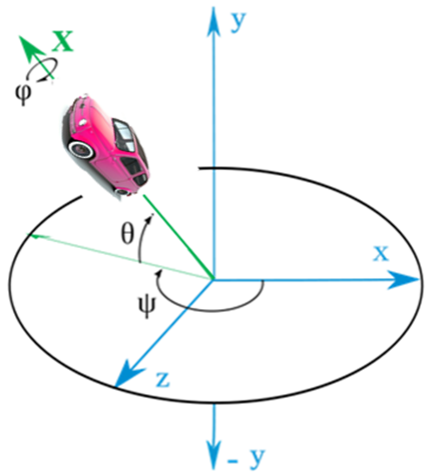


Figure III.3: Euler rotation orientation of a car.

it to control the RC cars. The location data is (x, z) since all RC cars are put on the ground. Their rigid locations vary with x and z values, and y values can be neglected. The quaternion data (qx, qy, qz, qw) collected from the ARENA software can be converted to Euler angles. Figure III.3 shows the Euler angles of a car in a 3-dimensional space. Since the RC car is on the ground, the value of θ and ϕ can be considered to be zero. The heading angle Ψ, used for tracking the virtual vehicle, can be computed using the following formula:

$$\Psi = arctan\left(\frac{2(qw \times qy + qx \times qz)}{1 - 2(qz^2 + qy^2)}\right) \tag{III.1}$$

Control of the RC cars

The RC car is controlled by a server implementing C++ programming. In order for an RC car to move according to specified paths, we need to know its steering angle, direction (forward/backward) and velocity. The RC car has two running modes: manual driving and autonomous driving. In the manual driving mode, the RC car's steering angle, direction and velocity are determined by the values generated from the steering wheel system which is manually controlled by the driver. In the autonomous

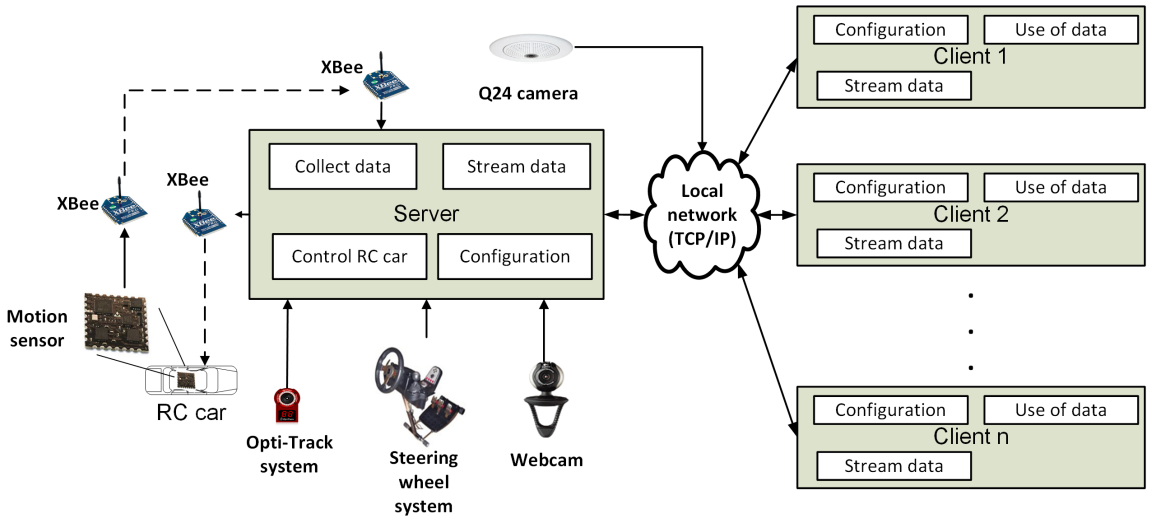


Figure III.4: The overall block diagram of the physical ITS testbed.

driving mode, the server generates a virtual vehicle that moves along a trajectory. Then, the necessary steering angle, direction and velocity based on the RC car's position and heading angle are computed so that the RC car can track the virtual vehicle. The detailed tracking algorithm is presented in my Master thesis [159].

Data collection

The overall data collection process of the physical ITS testbed is illustrated in Figure III.4. The dashed lines represent wireless communication and solid lines represent wired communication via USB or Ethernet cables. While the simulated testbed is presented in Section III.2.3, the physical testbed is discussed in this section. After the connection between the client and the server is established successfully, the client sends a request to ask the server to perform data collection. After receiving the request, the server lets the RC car be driven by the user and starts collecting data from the steering wheel system, the Opti-Track system and the motion sensor. Then it sends this collected data directly to the client program with a frequency of 20 Hz. The steering wheel set is controlled by the user to drive the RC car and its data is kept updating with a frequency of 20Hz. After receiving data successfully from the

server, the client displays on the screen the collected data and the video from the webcam in real time. It also saves the collected data to a file to record the history of the car. The client can use the collected data to perform driver analysis including intention recognition and drowsiness detection.

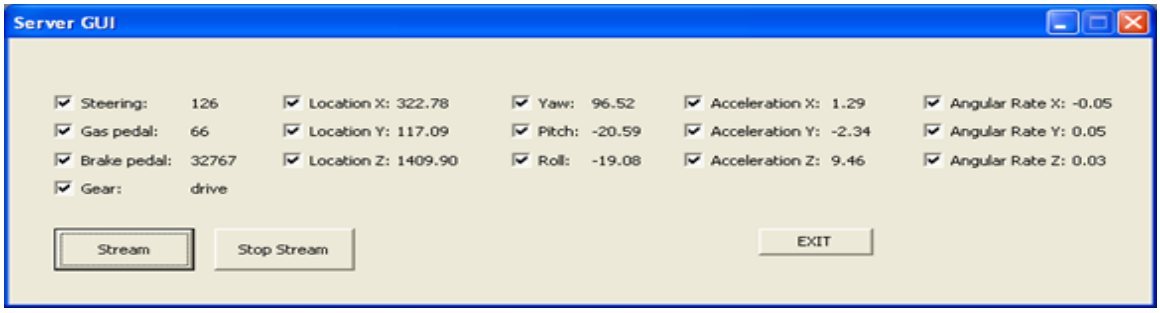
III.1.3 Remote configuration of the physical testbed

Besides the role of controlling the RC cars, the physical testbed helps us perform remote data collection and remote trajectory configuration which are presented in this sub-section.

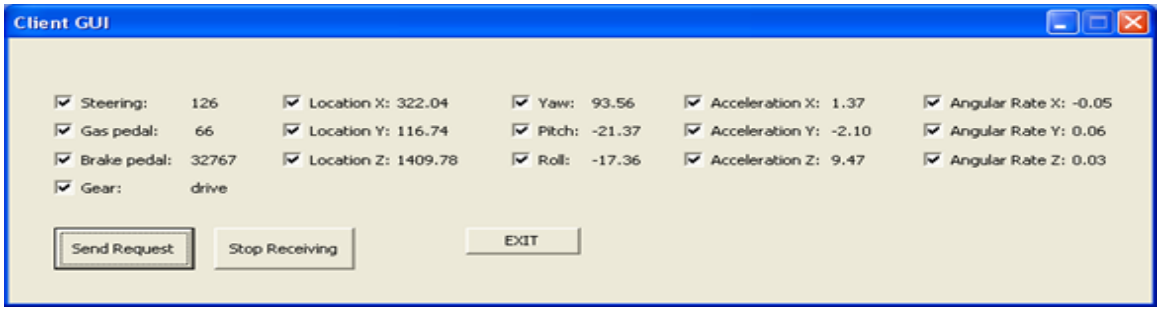
Remote data collection

The server and client user interfaces are created by implementing the Windows Application Programming Interface (Windows API) using the C++ programming language. There are sixteen data collected by the server:

- Steering wheel control data
  - Steering wheel: (0 to 250) from the most left to the most right.
  - Brake: (-32767 to 32767) from the maximum push to release.
  - Gas: (250 to 0) from the maximum push to release.
  - Gear: forward, neutral, backward (reverse).
- Location data in x, y, and z directions with a range of [-1829 mm, 1829 mm].
- Motion data in yaw, pitch, and roll with a range of [-180°, 180°].
- Acceleration data in x, y, and z directions with a unit of m/s^2.
- Angular rate data in x, y, and z directions with a unit of rad/s.



(a) Server user interface



(b) Client user interface

Figure III.5: Server and client user interfaces for data collection.

Check-boxes are created on both the server and the client user interfaces to select which data to be processed. Figure III.5 shows sixteen data selected and displayed on both the server and client user interfaces. Both the server and the client can select the data. Once the client and the server agree on the data selection, the client would receive the data via TCP/IP at the frequency of 20 Hz. The server can also collect the video of the driver's face via the webcam and send it to the client in real time. This video and steering wheel data are used in the case study of driver drowsiness detection in Chapter V. A session to plot the received data in real time is implemented for further analysis. Figure III.6 shows real-time plots of the received data at the client user interface with the positions corresponding to check-boxes in Figure III.5.

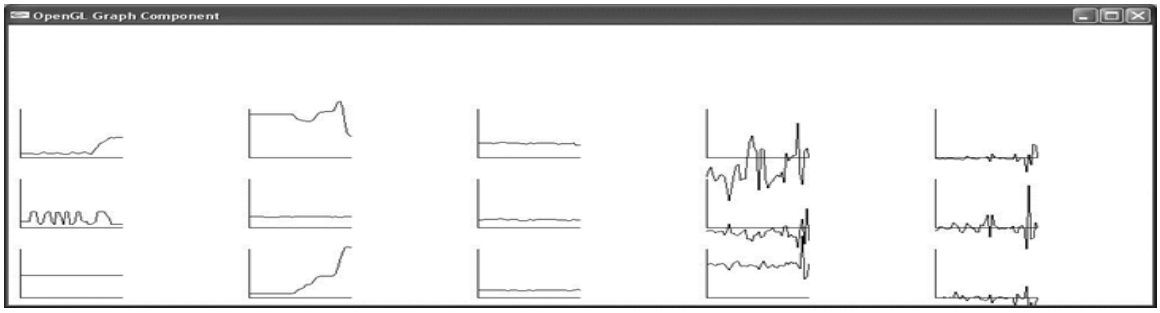


Figure III.6: The plotted graph of received data in the client application.

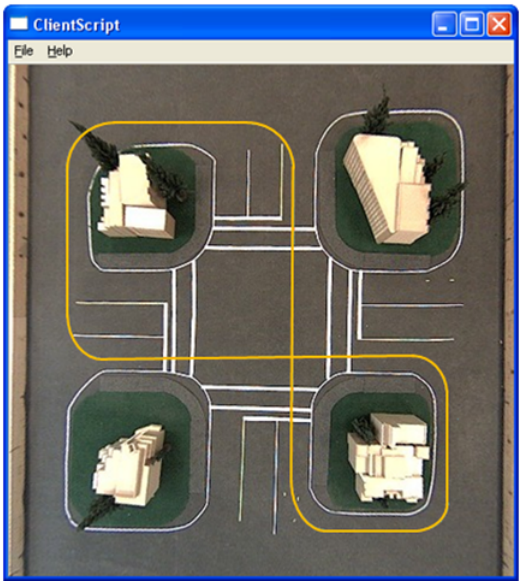


Figure III.7: The top view of the arena and a trajectory of lines and arcs.

Remote vehicle trajectory configuration

Before performing experiments, the client can design different trajectories for the RC car. Based on the designed trajectory, a script is encoded and sent to the server which would decode it and control the RC car to run on the designed trajectory after receiving the script.

1) Drawing vehicle trajectory: A client user interface for trajectory drawing and script encoding is created. This user interface has a background image of the arena's top view. The user interface provides the abilities of line and arc drawing. To draw a line, the user clicks on the image to specify the beginning point, then drags

the mouse to a new location to specify the ending point of the line. For drawing an arc, firstly, the user needs to specify the direction of the arc. The user also needs to specify the beginning and ending points of the arc using the mouse. Figure III.7 illustrates the top view image of the arena and a combination of lines and arcs drawn by the client user. Each line or arc is considered to be a numbered segment. When the RC car reaches the end of a segment, it continues tracking on the next one until the last segment is reached.

2) Mapping and Scripting location: The client program has to map the location of the line and arc from pixel coordinates to real-world coordinates of the Opti-Track system. The top view image has a resolution of 436 × 436 pixels. So, the center point of the image (218, 218) should correspond to the center point of the arena. The area that the Opti-Track system can cover is 3658 × 3658 millimeters. This area is measured on the image of 398 × 398 pixels. So, the location of a pixel $(image_X, image_Y)$ on the image is mapped to the location on the arena $(arena_X, arena_Z)$ through the following equations:

$$arena_X = (image_X - 218) \times \frac{3658}{398}$$
$$arena_Z = (image_Y - 218) \times \frac{3658}{398}$$

Figure III.8 shows the mapping result displayed on the client user interface. The intersection in the testbed is designed on a flat surface. Therefore, it is assumed that the coordinate value in the y direction to be zero and only the coordinate values in the x and z directions are considered.

For the RC car to track a line, we need to know its beginning point (x_b,y_b) and ending point (x_e,y_e). The ending point of a segment is the beginning point of the next segment. To track an arc, we need to specify the coordinate of the center of the arc (x_0, y_0), radius, angle and direction (clockwise or counter-clockwise). These

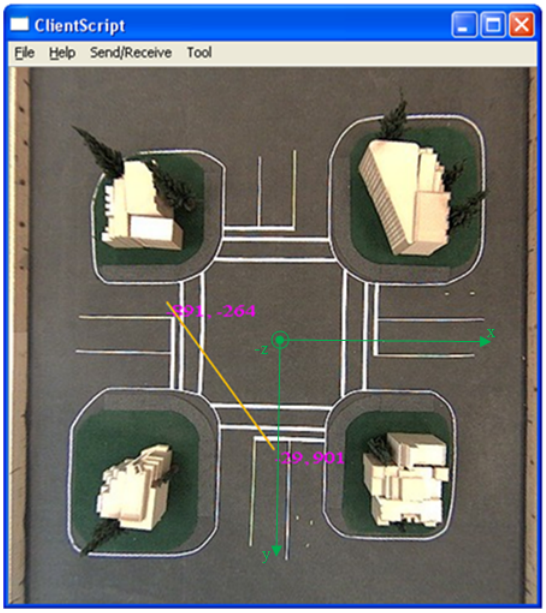


Figure III.8: Mapped locations from pixel to real-world coordinates.

parameters are calculated during the drawing of the arc and converted to real-world coordinates. Then we can combine the parameters of lines and arcs into a script and send it to the server, which can control the RC car based on the trajectory coded by the script. The script has the following format:

$$S, 1^{st}Line/Arc, 2^{nd}Line/Arc, ..., n^{th}Line/Arc, E$$

The formats of *Line* and *Arc* are as follows:

$$Line : L, x_b, y_b, x_e, y_e$$

$$Arc : A, x_0, y_0, radius, angle, direction$$

The letter 'S' and 'E' determine the start and end of the trajectory respectively. The letter 'L' and 'A' indicate the segment to be a Line or an Arc respectively.

To control the RC car, the virtual vehicle based tracking control algorithm is adopted following our previous works [160]. In previous projects, the trajectory of the RC car is predefined with a certain number of segments. In this dissertation, the trajectory is drawn by a client user. After receiving the encoded script from the

client, the server decodes it into different parameters that are used to control the RC car. Firstly, the virtual vehicle would run along a segment so that the RC car can follow it. Figure III.9 illustrates various trajectories drawn in yellow curves, and the corresponding real-world trajectories of the RC car are in red curves.

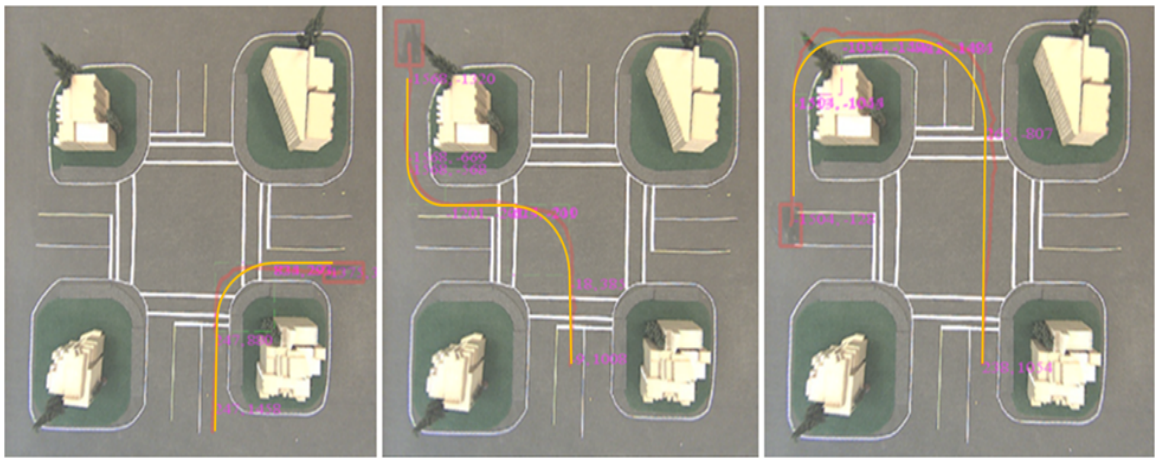


Figure III.9: Different drawn trajectories and corresponding RC car's movements.

III.2 The simulated testbed

The simulated testbed has three main components: 1) a driving simulator which simulates the driving environment, 2) a tool that helps user design road map database for the driving simulator, and 3) a script to control the simulator's behavior. The overall block diagram of the framework is shown in Figure III.10.

III.2.1 The driving simulator

The Carnetsoft driving simulator [161] is utilized to provide a realistic environment to collect data for training and local evaluation. Figure III.11 shows the hardware setup of the simulated testbed. A metal frame was built to mimic an inside cabin of a vehicle. Three monitors are mounted on the frame to display the driver's view. A Logitech G27 racing wheel and a webcam are used for this simulated testbed. An adjustable and reclinable driver seat is also included to ensure comfort for different

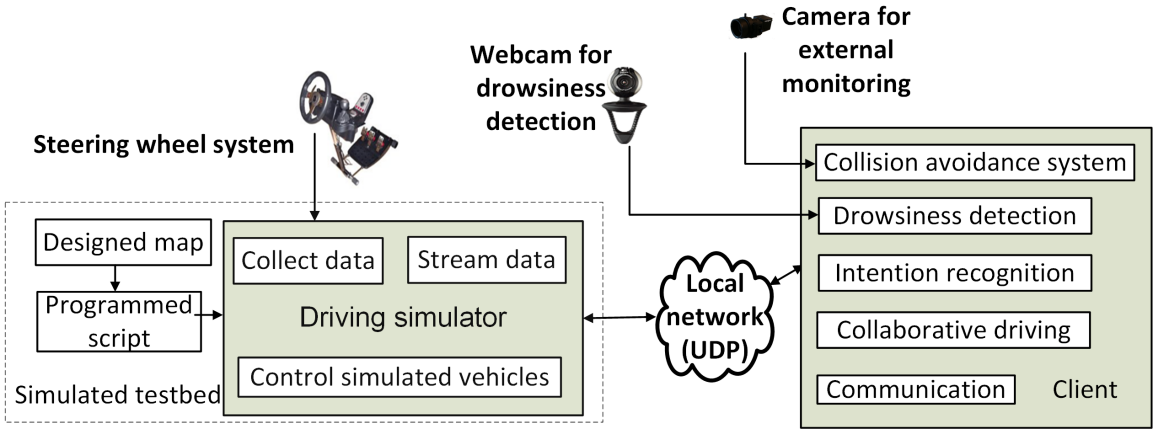


Figure III.10: The overall block diagram of the simulated testbed.

drivers. The simulator requires one more monitor to display its control windows while the driving environment occupies three other monitors. These control windows help the user manage the experiment, start/stop the simulator while monitoring the simulated vehicle's current position and the whole system's frame rate. In addition, the dashboard information such as speedometer, tachometer, fuel consumption, changing lane signal indicator, steering wheel angle, lighting switch and percentage of pressure on the clutch, brake and acceleration pedal positions are also monitored. The screenshots of four monitors are presented in Figure III.12. In the driving environment of this figure, we can also observe that there are glitches between the center monitor and two side ones. This is the result when placing the three monitors on the same plane. In practice, the two side monitors are mounted with certain angles like in Figure III.11, then the glitches disappear and the image transitions between the three monitors become smoother. In addition, a USB 3.0 PointGrey camera is mounted behind the driver seat to monitor the external environment.

III.2.2 Designing road map

The Carnetsoft driving simulator also gives us flexibilities in creating a driving simulation. It provides us a tool to design networks of roads [161]. Figure III.13

Figure III.11: The simulated testbed.

shows a screenshot of a designed road map of a two-lane highway with exit and merge roads. A road is described as a combination of multiple segments including intersection, straight and curved roads whose parameters such as starting points, length, curve angles, number of lanes, directions, traffic signs, etc. are inputted into the left panel shown in Figure III.13. Other objects like facades (blocking the horizon such as fences, tree lines, hills, etc.), underlying fields, trees, buildings, pedestrians, animals, etc. can also be inserted.

III.2.3 Script language

The driving simulator's behavior is controlled by a script language developed by the manufacturer [161]. The script is programmed by the user to specify the simulation scenarios. In a script, a pre-built road map database can be specified. The user can choose the driving environments such as weather condition (dry, rain or snow), lighting (daytime or nighttime), road frictions, etc. Also, the behaviors of all vehicles that participate in the traffic can be programmed. For example, a car can

Figure III.12: Screenshots from four monitors of the simulator.

be controlled to stop at an intersection for several seconds and then continue driving. Each scenario has a unique identification number. Various vehicles such as sedan, SUV, truck and bus can also specified. The script language also comes with many pre-built functions which let the user collect variables such as velocity, acceleration, heading angle, lateral position, etc. In this driving simulator, the driving mode of the ego vehicle can be either manual, semi-autonomous (via speed or steering only) or fully autonomous (via both speed and steering). A driving mode can be switched to another one easily through commands provided by the manufacturer. In this dissertation, only the manual (human) and fully autonomous (co-pilot) driving modes are considered. So, the behavior of the ego vehicle during the autonomous driving mode can be pre-programmed by setting its velocity and lateral position.

III.2.4 Data collection

The overall data collection process of the simulated driving testbed is illustrated in Figure III.10. The dashed lines represent wireless communication and solid lines represent wired communication via USB or Ethernet cables. Unlike in the physical testbed, in the simulated testbed, the cameras are connected directly to the client programs which also receive data sent from the driving simulator to perform both

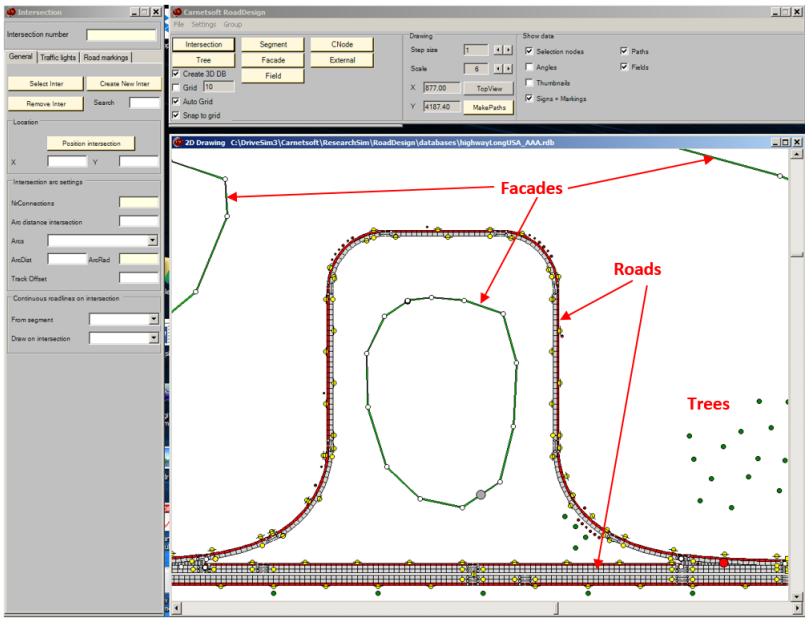


Figure III.13: A tool for designing road map.

the internal and external risk analyses. Then, the collaborative driving framework utilizes these analyses to generate an optimal control input for the ego vehicle. This control input is sent back to the driving simulator so that it could control the ego vehicle accordingly.

III.3 Summary

This chapter presents the development of the ITS testbed that includes both the physical and simulated testbed. The physical one has the remote configuration and remote data collection capabilities. It is used to validate the switching mechanism of manual/autonomous driving based on drowsiness detection. The simulated testbed provides us with the flexibility in designing road maps and traffic scenarios. It is used for validating the collaborative driving framework.

CHAPTER IV

DRIVER INTENTION RECOGNITION

Driver intention recognition can provide useful information for other human drivers or autonomous vehicles through vehicle-to-vehicle communication [25] to avoid traffic accidents. This chapter describes the driver intention recognition system which adopts the Hidden Markov Model (HMM) algorithm. In addition, the simulation of vehicle dynamics and experimental results of the system are also presented.

IV.1 Driver intention recognition using HMM

This section gives an introduction to the system overview and the HMM modeling.

IV.1.1 System overview

Figure IV.1 presents the overview of the proposed intention recognition system. A driver can control the steering wheel system to drive the vehicle in the driving simulator. The system consists of a server and a client. The server collects the car's front-view images from the camera so that the lane position with respect to the car is determined. The details of the lane detection system are presented in Section VII.1.1. Moreover, the server reads the steering wheel control data which drives the simulator. The server also sends the data to the client in which the vehicle states are obtained through the vehicle dynamic model. Based on the vehicle control data, state data and lane position, the intention of the driver can be inferred. In the real world implementation, the vehicle states such as velocity, acceleration and yaw rate can be obtained from the standard vehicle diagnostic interface. In this dissertation, to obtain

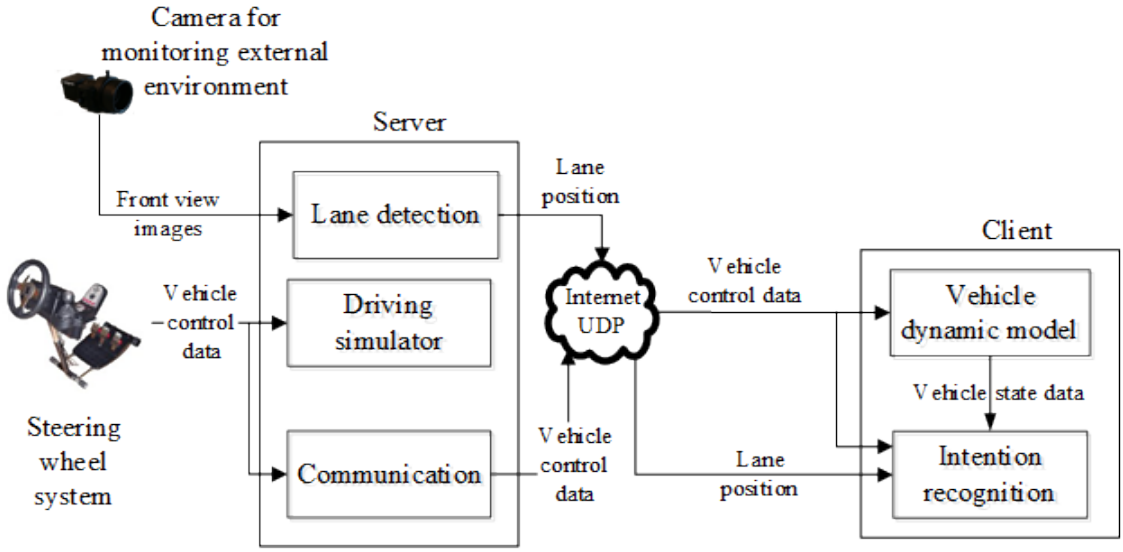


Figure IV.1: Overview of the intention recognition system.

the simulated vehicle's state data, a vehicle dynamic model is implemented. Figure IV.2 shows the overall approach of the driver intention classification. Lane position, steering wheel data and vehicle state derived from the vehicle dynamic model are used as feature vectors which are then segmented. The resulting segments are fed to a training process to find the HMM parameters that best suit the training data. In the testing stage, the segmented testing input is fed to each HMM model to evaluate how well the observation sequences fit that model. The model that has the best likelihood represents the current intention of the driver.

IV.1.2 HMM modeling

Hidden Markov Models have been used widely in the problem of driver intention recognition. In this dissertation, an HMM model, μ, includes hidden states, observations, initial state probabilities, state-transitional and observational probability matrices [30].

$$\mu = \{S, O, \pi, A, B\}$$

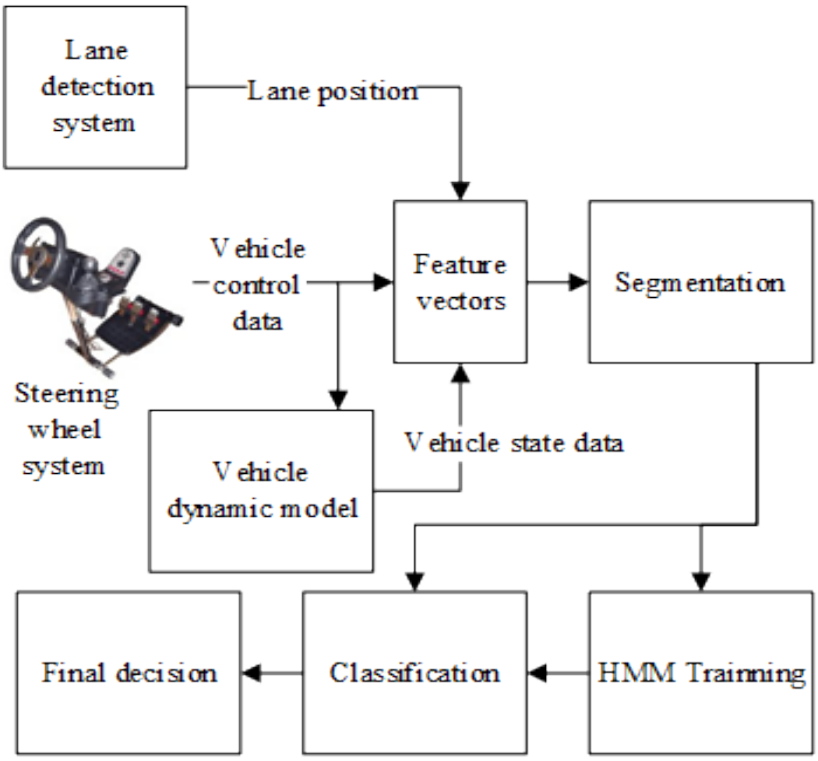


Figure IV.2: Overview of the intention recognition process.

S :$\{S_1, ..., S_N\}$: set of hidden states

O :$\{O_1, ..., O_M\}$: set of observation symbols

π :initial state probabilities

A :state probability matrix

B :state observation probability matrix

The input data to the HMM models is a multi-dimensional feature vector extracted from the lane position, the vehicle's control and vehicle state data. The lane position includes the distances from the left and right lines of the vehicle's current lane to its center. The vehicle's control data is collected by the Logitech G27 steering wheel system shown in Figure IV.3. It can provide the vehicle's current steering angle, acceleration and brake pedal position. After the matching process described in Section IV.2 is completed, the matched dynamic model can take vehicle control

Figure IV.3: The Logitech G27 racing wheel for manual driving.

data as input and derive the corresponding vehicle state. The unsupervised K-mean clustering technique is utilized to convert the multi-dimensional feature vectors to discrete symbols (observation sequences).

In order to optimize the HMM parameters (π, A, B) of certain training data, Baulm-Welch algorithm is implemented. This is a recursive forward-backward method to make the model parameters describe the observation sequence [30]. To handle the intentions at curved roads, a set of training data was also collected on different curved roads. In this dissertation, five driving intentions are aimed to be classified. They are slowing down, speeding up, changing lane left, changing lane right and keeping the vehicle's current state. According to Figure IV.2, the observation sequence of testing data is fed to five trained HMM models for evaluation using the Viterbi algorithm which can calculate the hidden state sequence s through the model μ that most likely produces the given testing observation sequence O [30].

$$s^* = \arg\max_s P(s|O, \mu)$$

This is the optimal segmentation of the testing observation sequence on a given model. Then, the likelihood of the corresponding maneuver, $P(O|s^*, \mu)$, can be obtained by

straightforward calculation based on the state transition matrix A and state observation matrix B of the model μ [162]. Then, the likelihoods of all trained HMM models with the given observation sequence can be compared. The HMM that has the maximum likelihood represents the best fitted maneuver.

IV.2 Simulation of vehicle dynamics

In real-world vehicles, the vehicle state data can be obtained easily through the CAN bus system. However, it is quite challenging to access the vehicle dynamics in simulated environments; especially in a commercial product like the Carnetsoft driving simulator [161]. Even if the Carnetsoft driving simulator can provide the vehicle state data, they do not let us access their vehicle's dynamics which would be later used to estimate the vehicle state data from the vehicle control data. This problem is presented in Section VIII.1. Therefore, from the ego vehicle's control data, the author aims to estimate its dynamics and state data.

In this section, a vehicle dynamic model is used to model the motion of the ego vehicle in the Carnetsoft driving simulator. The steering wheel data is fed to this vehicle dynamic model to yield the vehicle's current state such as velocity, acceleration and yaw rate. In order to obtain the vehicle state, the vehicle dynamic model's parameters must be adjusted so that they can yield the same vehicle state as in the simulator.

The vehicle dynamic model used in this simulation is derived by MathWorks [163]. Figure IV.4 shows the schematic of the vehicle dynamic model which takes the vehicle control input (steer (σ), gas(α) and brake(β)) and uses six parameters to describe

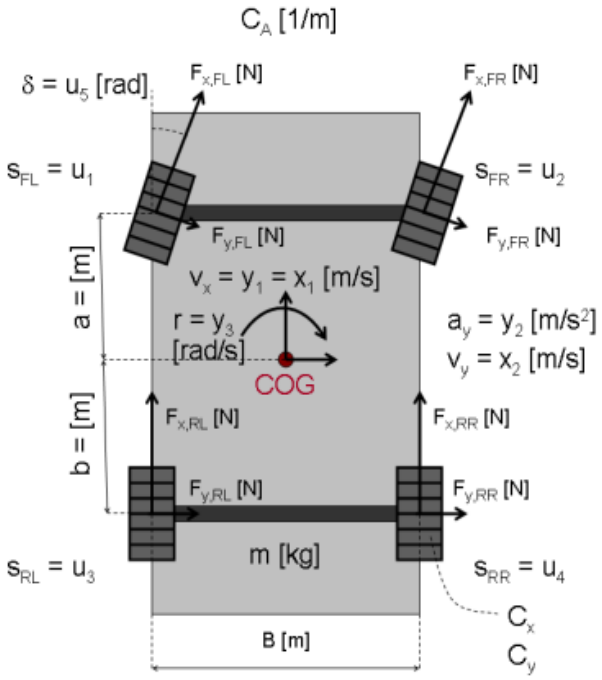


Figure IV.4: Schematic view of a vehicle dynamic system [163].

the real vehicle and the environment:

m : Mass of the vehicle [kg]

a : Distance from the front axle to Center of Gravity [m]

b : Distance from the rear axle to Center of Gravity [m]

C_x : Longitudinal tire stiffness [N]

C_y : Lateral tire stiffness [N/rad]

C_A : Air resistance coefficient [1/m]

The model takes five inputs:

$$u_k(t) = s\_i_k(t) = \text{Slip ratio of four tires} \tag{IV.1}$$

$$u_5(t) = \sigma(t) \text{ [rad]} \tag{IV.2}$$

where $k = 1, 2, 3, 4$; $i = \{FL, FR, RL, RR\}$ represents the Front Left, Front Right, Rear Left and Rear Right tire respectively. The Slip ratios of tires are derived from

the following equation:

$$s\_i(t) = F_{x,i}(t)/C_x \tag{IV.3}$$

where $F_{x,i}(t)$ denotes the longitudinal force from the engine. In this simulation, the vehicle is assumed to be front-wheel driven. So,

$$F_{x,FL}(t) = F_{x,FR}(t) \propto \alpha(t) - \beta(t) \tag{IV.4}$$

and $F_{x,RL} = F_{x,RR} = 0$. Steering angle is derived from the steering wheel value. So, the input of the vehicle dynamic model comes from the steering wheel system (Figure IV.3) which can provide us with steering angle, gas and bake pedal values. Three states of the model are described below:

$$x_1(t) = v_x(t) = \text{Longitudinal velocity [m/s]} \tag{IV.5}$$

$$x_2(t) = v_y(t) = \text{Lateral velocity [m/s]} \tag{IV.6}$$

$$x_3(t) = r(t) = \text{Yaw rate[rad/s]} \tag{IV.7}$$

The state-space model is illustrated by the following differential equations [163]:

$$\begin{aligned}\frac{dx_1(t)}{dt} =& x_2(t) \times x_3(t) \\ &+ m^{-1} \times [C_x \times (u_1(t) + u_2(t)) \times cos\,(u_5(t)) \\ &- 2 \times C_y \times \left(u_5(t) - \frac{x_2(t) + a \times x_3(t)}{x_1(t)}\right) \times sin\,(u_5(t)) \\ &+ C_x \times (u_3(t) + u_4(t)) - C_A \times x_1(t)^2]\end{aligned} \tag{IV.8}$$

$$\begin{aligned}\frac{dx_2(t)}{dt} =& - x_1(t) \times x_3(t) \\ &+ m^{-1} \times [C_x \times (u_1(t) + u_2(t)) \times sin\,(u_5(t)) \\ &+ 2 \times C_y \times \left(u_5(t) - \frac{x_2(t) + a \times x_3(t)}{x_1(t)}\right) \times cos\,(u_5(t)) \\ &+ 2 \times C_y \times \frac{b \times x_3(t) - x_2(t)}{x_1(t)}\Big]\end{aligned} \tag{IV.9}$$

$$\begin{aligned}\frac{dx_3(t)}{dt} =& \frac{1}{(0.5 \times (a+b))^2 \times m} \times \\ & \{a \times [C_x \times (u_1(t) + u_2(t)) \times sin\,(u_5(t)) \\ & + 2 \times C_y \times \left(u_5(t) - \frac{x_2 + a \times x_3(t)}{x_1(t)}\right) \times cos\,(u_5(t))\Big] \\ & - 2 \times b \times C_y \times \frac{b \times x_3(t) - x_2(t)}{x_1(t)}\Big\} \end{aligned} \tag{IV.10}$$

The output of the model is defined as:

$$y_1(t) = x_1(t) \tag{IV.11}$$

$$y_2(t) = x_2(t) \tag{IV.12}$$

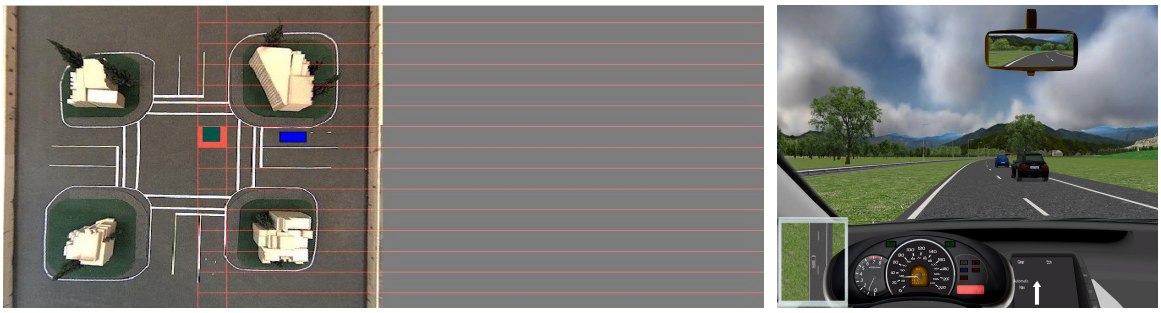
$$y_3(t) = x_3(t) \tag{IV.13}$$

$$y_4(t) = \text{longitudinal acceleration} = \frac{dx_1(t)}{dt} - x_2(t) \times x_3(t) \tag{IV.14}$$

$$y_5(t) = \text{lateral acceleration} = \frac{dx_2(t)}{dt} + x_1(t) \times x_3(t) \tag{IV.15}$$

Solving these ordinary differential equations (ODE), Equation (IV.8) – (IV.10), explicitly is difficult. However, the Runge-Kutta method [164] provides a numerical solution for this sort of ODE to estimate the dynamics of vehicle's behavior (velocity, acceleration and yaw rate) in every iteration. This information reveals the behavior of a real vehicle. The parameters of the vehicle dynamic model can be adjusted to match any real-world vehicle's dynamics.

Since the Carnetsoft driving simulator is a commercial product from a third-party company, we can only obtain the ego vehicle's state (velocity, acceleration, yaw rate) but not the dynamic model's parameters. However, they can be inferred by adjusting the parameters of the vehicle dynamic model so that it can match the outputs with those of the vehicle in the Carnetsoft driving simulator. Three criteria are used in the matching process. To match the velocity, the velocity of the dynamic model and that of the ego vehicle in the Carnetsoft driving simulator are compared. To match the acceleration, we measure how long it takes the vehicle to reach a certain velocity.



(a) Client simulation (b) Server simulation

Figure IV.5: The client simulation of matching the dynamic model of the simulated vehicle with the one in the Carnetsoft simulator.

From the yaw rate of the vehicle state, the orientation can be inferred. In order to match the orientation, a separate driving simulation is developed. In this simulation, a driver can manually drive a vehicle by controlling the Logitech G27 steering wheel system. This simulation environment gives a top view of the intersection where a car is driven manually [160]. Figure IV.5 illustrates the simulation of two vehicles passing the intersection. The blue rectangle represents the vehicle controlled by the Logitech steering wheel. To ensure the correct matching, the comparing process should collect information over a long distance of driving. This simulation is named as the client simulation and the Carnetsoft driving simulator is named as the server simulation. Due to computation burdens, the client and server simulations run in different computers and can share the vehicle control data using a communication program running in the background of the server simulation via UDP. The server simulation is assumed to simulate the US standard freeway lane width of 12 feet (3.6 m). So, to match the left steering, firstly, the distance from the position of the blue vehicle in the client simulation to the left center line can be proportionally set to 6 feet (1.8m). Then, the time when both vehicles in both simulations cross the center line can be matched. Repeating in the opposite direction would yield the matching in right-side steering. The perfect matching is that when turning, both vehicles can

Figure IV.6: Experimental setup with the client and server computers performing different tasks.

cross the center line at the same time with the same velocity.

IV.3 Experiments & results

Figure IV.6 illustrates the experimental setup on the simulated testbed. The Logitech steering wheel system is connected to the server. The lane position is determined by the lane detection program running in the background of the server. In addition, this server collects the vehicle control data and sends it to a client program running in MATLAB with a frequency of 20Hz. This client program can compute the corresponding vehicle state of the ego vehicle using the matched vehicle dynamic model. Both lane positions, vehicle control data (from steering wheel system) and vehicle state data are then used to form the feature vector and follow the process in Figure IV.2 to realize the classification. The program runs on a computer with 3.6 GHz Intel Core i7 CPU and 16 GB of RAM. The data collection was performed with

Table IV.1: Accuracy of the driver intention recognition system.

	Accuracy		
Maneuver	Control data only	Control & state data	Lane position, control & state data
Keeping current state	87%	93%	95%
Speeding up	89%	96%	96%
Slowing down	87%	95%	95%
Changing lane left	80%	88%	91%
Changing lane right	81%	89%	92%

five subjects. Each subject was asked to drive with five maneuvers in 5 minutes. 2/3 of the data was used for training. 1/3 of the data was used to evaluate the performance of the trained HMM models. The classification performance under three kinds of input combinations were compared. They include 1) vehicle control data only, 2) both vehicle control and state data, and 3) all inputs which include the vehicle's lane position, control and state data.

Table IV.1 illustrates the performance of the system. The longitudinal maneuvers (keeping current state, speeding up, and slowing down) could be recognized more accurately than the lateral maneuvers (changing lane left and changing lane right). Moreover, using all inputs has the best performance on the keeping and lane-changing maneuvers. This is reasonable because the lane position has the maximum contribution on the lateral movements. With longitudinal maneuvers, the lane position does not have much contribution, and thus using all inputs has the same performance as using the vehicle's control and state data.

Sixty driving tests were conducted to test the real-time classification. Figure IV.7 shows the classification results of detected maneuvers and the ground truth. All maneuvers/intentions could be classified before their completion. Table IV.2 shows

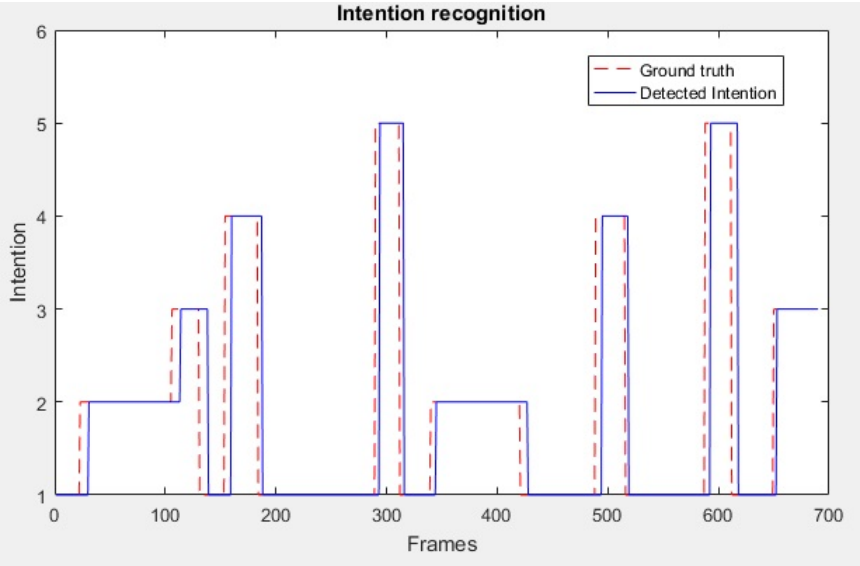


Figure IV.7: Graphs of intention recognition results. Intention values: 1 = keeping the vehicle's current state; 2 = speeding up; 3 = slowing down; 4 = changing lane left; 5 = changing lane right.

the recognition times of all intentions. On straight roads, the "Keeping" maneuvers can be quickly classified. However, at the beginning of curved roads when drivers steer along the curve, the classified maneuvers would be lane-changing. After a certain time (4.2 seconds with vehicle control data only; 3.5 seconds with vehicle control and state data; and 1.2 seconds with all inputs) when the trained model recognizes the curved patterns, the "Keeping" maneuvers would be classified accurately. After being recognized as running on a curved road, the recognition times of other maneuvers is similar to the recognition times of these maneuvers when not running on curved roads. In addition, on curved roads, using all inputs (lane position, vehicle control and state data) helps the system recognize the "Keeping" maneuver faster than using only vehicle data (control and state). This is because the lane position helps determine whether the driver tries to keep the lane or not. If the lane detection process fails e.g., no lane or wrong lane is detected, the pre-trained HMM models which only use the vehicle control and state data can be utilized.

Table IV.2: Average recognition time of the driver intention recognition system with different combinations of inputs.

	Recognition time		
Maneuver	Control data only	Control & state data	Lane position, control & state data
Keeping @ straight	0.5 s	0.3 s	0.3 s
Keeping @ curved	4.2 s	3.5 s	1.2 s
Speeding up	0.6 s	0.3 s	0.3 s
Slowing down	1.1 s	0.5 s	0.5 s
Changing lane left	1.1 s	0.9 s	**0.5 s**
Changing lane right	1.4 s	1.1 s	**0.7 s**

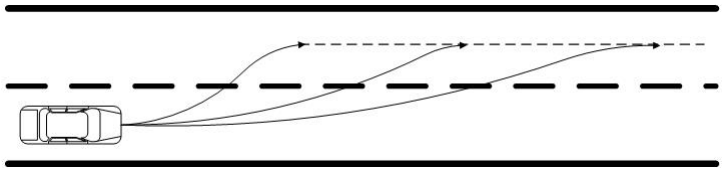


Figure IV.8: Lane changing styles.

As shown in Table IV.1 and IV.2, in terms of accuracy and recognition time, the approach that uses both vehicle control and state data has better performance than the one using only vehicle control data. This is because the vehicle state data can give more information about the status of the vehicle. For example, during a lane-changing maneuver, different drivers have different styles of changing lane, as illustrated in Figure IV.8. While some drivers merge slowly, others tend to merge more quickly. In this case, the vehicle state (velocity, acceleration, and yaw rate) would be very important to obtain an accurate prediction.

IV.4 Summary

This chapter presents a system that utilizes the ego vehicle's lane position, control and state data to predict the driver's intention. Five maneuvers which include changing lane left, changing lane right, speeding up, slowing down and keeping the vehicle's current state can be recognized. A dynamic model was used to model the motion of the ego vehicle in the Carnetsoft driving simulator.

CHAPTER V

DRIVER DROWSINESS DETECTION AND ITS APPLICATION IN ASSISTED DRIVING

This chapter presents the drowsiness detection system and its application on the simulated and physical testbeds.

V.1 Driver drowsiness detection

Drowsiness is one of the main causes of severe traffic accidents. In this dissertation, the drowsiness detection system takes two inputs: images of the driver's face and steering wheel data. Previous works have mainly focused on developing a drowsiness detection system using only one channel of information. In this dissertation, both of them are employed, pre-processed, and integrated at the feature level to obtain reliable drowsiness decisions. The drowsiness detection algorithm is implemented as a remote client application where the two sets of inputs transmitted from the server are processed and the final decision is sent back to the server.

As can be seen from the system diagram in Figure V.1, there are three main components of the system: facial expression feature extraction, steering wheel feature extraction, and feature level integration.

V.1.1 Facial expression feature extraction

Using facial expressions to determine the drowsiness condition involves the following steps: accepting the stream of images in real time from the data server through the TCP/IP network, detecting the face of the driver from the image frame, and

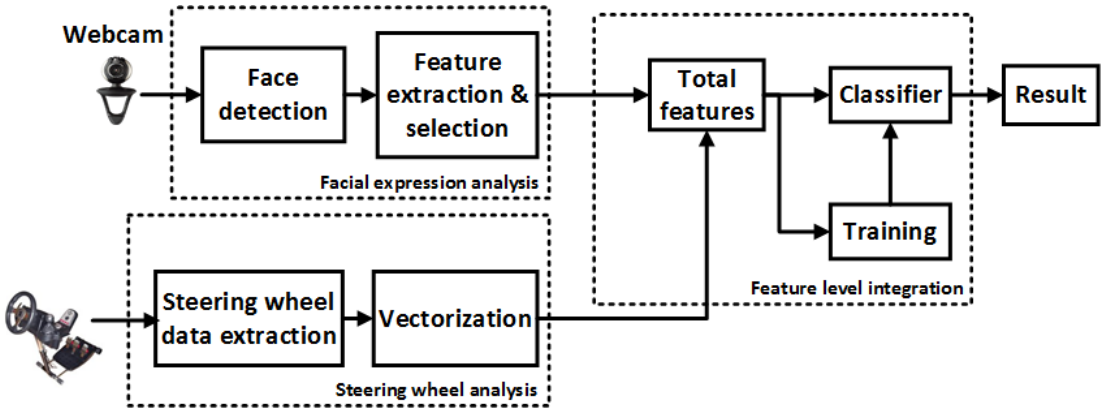


Figure V.1: The overall system diagram of drowsiness detection.

processing the image to determine the state of the driver.

The system accepts a stream of images from the server at a rate of 20 frames per second. When a frame is captured, it is first converted to grayscale and histogram equalization is utilized to increase the contrast of the image for better face detection. In our previous works [39, 165], the Viola-Jones robust real-time face detection algorithm [166] is implemented in OpenCV [167] to detect the driver's face and the CAMShift algorithm [40] is utilized to track the face's movement under different circumstances where the face detector fails to detect. However, there were still some portions of the background in the face area. The features in different backgrounds may affect the detection performance even if the driver's behaviors in testing and training data are similar. Moreover, different light conditions (sunny, shadows or dark) may affect the face detection especially when it relies on the color information (the CAMShift algorithm). Therefore, an open-source Dlib C++ library's face detection was implemented. The new face detection uses the classic HOG feature combined with a linear classifier, an image pyramid, and a sliding window detection scheme [41]. The new face detector overcomes the dynamic background issue while providing fast face tracking.

Once the face is detected, features are extracted. To do that, two different areas

of interest are used: the face and the eye regions. The area of interest is fed to Gabor wavelet decomposition of 2 scales and 4 orientations to extract the facial features. From the Gabor decomposition, there are so many facial features which contain redundant information. Therefore, the Adaboost weak learning algorithm [168] is employed to select the most important features for classification. The weak classifiers are based on a single facial feature. Then the features with the minimum classification errors are selected. For the weak classifiers, two different thresholds are used: 1) Averaging: the average of the values of the facial feature of all training images is taken. 2) Searching maximum: each value of the facial feature of all training images is utilized as a threshold of the weak classifier and choose the one that gives the maximum separation between the drowsy and non-drowsy training images.

V.1.2 Steering wheel feature extraction

According to previous studies in drowsiness detection, there is a good correlation between the steering wheel movement and the drop in the state of vigilance while driving [79, 80]. The authors claimed that classifying driver drowsiness is possible by using steering wheel data as an input for artificial neural networks. In an alert state, the driver tends to make small adjustments to the steering wheel angle and hence there are only small variations in the steering wheel angle. When the driver is in a drowsy state, the movement of steering wheel becomes unpredictable resulting in a large change in trajectory (zigzag driving) and there will be a larger amplitude of movement to keep the vehicle in the center of the lane. However, zigzag driving in a short time does not mean that the driver is drowsy. It is possible that the driver may make sudden zigzag movements during some urgent situations in order to avoid accidents. Therefore, the drowsiness detection using steering wheel information is based on a 10-second sliding window to achieve reliability. The step size of the sliding window is 0.05 seconds, which can achieve realtimeness.

V.1.3 Feature level integration

By using feature extraction and selection methods, a single vector of the most important features from the input image is obtained. Then, the vector of steering wheel angles is appended to that vector as shown in Figure V.1. This feature vector is the input to an SVM classifier with Gaussian Radial Basis Function (RBF) kernels which provides a nonlinear decision hyper-plane between drowsy and non-drowsy feature vectors. The stream of images with the detected face, face locations, the steering wheel angle vector and their corresponding labels are saved. Using these data, the classifier is trained offline. In the real-time testing stage, the final decision from the classifier is sent to the collaborative driving algorithm to find the proper control action.

Figure V.2: The experimental setup of the drowsiness detection's application in the simulated testbed.

V.2 Evaluation of the drowsiness detection system

The experimental setup is shown in Figure V.2. A driver sits behind the steering wheel to control the vehicle in the simulator which sends steering wheel information to a C++ program to combine with the images of the driver's face into total feature vectors which are used for training and evaluating the drowsiness detection algorithm. A total of 11 human subjects with different races, ages and genders were recruited to conduct the tests. Seven subjects were asked to drive with drowsy and non-drowsy status for 3 minutes. They were asked to drive the vehicle in both drowsiness and non-drowsiness states. Each state lasted 3 minutes. Four subjects were asked to drive for 10 minutes in both drowsy and non-drowsy states. In the laboratory-based experiments, the subjects were asked to mimic drowsiness. They were asked to push a button on the steering wheel when they started to mimic the drowsiness state and push another button when they finished mimicking the drowsiness state. In this way, the ground truth of drowsiness could be collected. The subjects' videos and data were labeled. While 2/3 of the data was used for training, the remaining 1/3 of the data was used to evaluate the trained model. The evaluation with individual source was performed separately, and then their combination (feature level integration) was also evaluated to investigate their performances.

V.2.1 Drowsiness detection with facial expression data

The detection accuracy was evaluated based on the different numbers of facial features, different regions of interest (eye only or face), different thresholds for Adaboost classification (Averaging or Searching Maximum) and different techniques (i.e., Adaboost or SVM). For a better understanding, the performance evaluation scheme is categorized into *Averaging* and *Searching Maximum Adaboost threshold computation* approaches. In each approach, the number of features selected by the Adaboost is increased from 10 to 300 with an interval of 10 and observed the variation in perfor-

mance.

Averaging threshold calculation - approach #1

The accuracy values are shown in Figure V.3 and V.4 for different system parameter settings. It can be observed that when the number of facial features selected for classification increases, the performance converges to the maximum accuracy values. In Figure V.3, the accuracy elevates to the saturation point quickly until the number of features reaches 100. For averaging threshold computation, using the Adaboost classification is significantly less accurate than using the SVM classification with RBF kernel for the same chosen region of interest (either eye or face region). In addition, for the same chosen classification technique (either Adaboost or SVM), the classification accuracy of the system using face region as the ROI is better than using the eye region as shown in Figure V.3. This result proves the fact that detecting drowsiness using other facial expressions in combination with eye closure gives more reliable performance than using eye closure alone. A maximum accuracy of 95.36% for 150 facial features with the detected face selected as the ROI and using SVM classification was obtained.

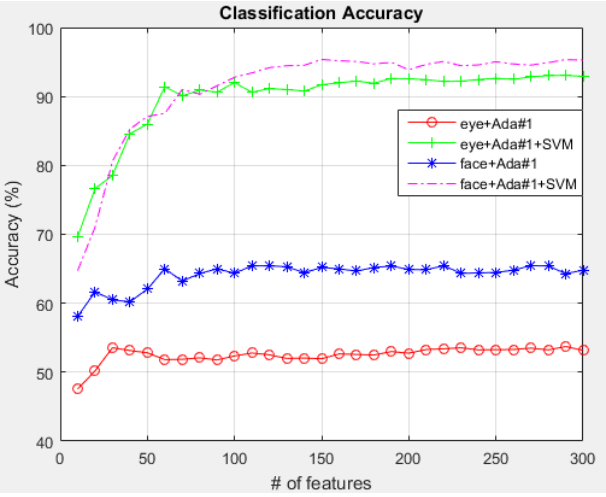


Figure V.3: Classification accuracy for Averaging threshold computation.

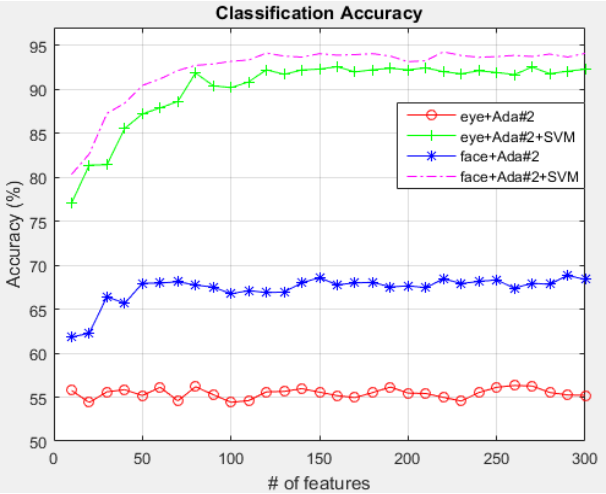


Figure V.4: Classification accuracy for Searching maximum threshold computation.

Searching-maximum threshold calculation - approach #2

Figure V.4 illustrates an interesting relationship between the different methods of classification and different choices of regions of interest (ROIs) for searching-maximum threshold computation. By using the Adaboost classification method, the obtained classification accuracy almost remains constant. This result implies that increasing the number of features being added to the cascaded combination would not improve the detection performance too much. Moreover, its classification accuracy is significantly lower than using SVM classification. Similar to the case of averaging threshold computation, using face region as the ROI also gives better results in classification accuracy than using eye region only. A maximum accuracy of 94.28% for 220 facial features with face region selected as the ROI and using SVM classification was obtained.

V.2.2 Steering wheel data analysis

The steering wheel data analysis is independently performed to evaluate the classification accuracy. Similar to the facial expression case, different feature vector sizes can also be used from 10 to 300 in an interval of 10. A Gaussian radial basis function

(RBF) kernel is utilized for the SVM classifier and obtained a maximum accuracy of 81.28% by using a vector size of 140 steering wheel features as can be seen in Figure V.5.

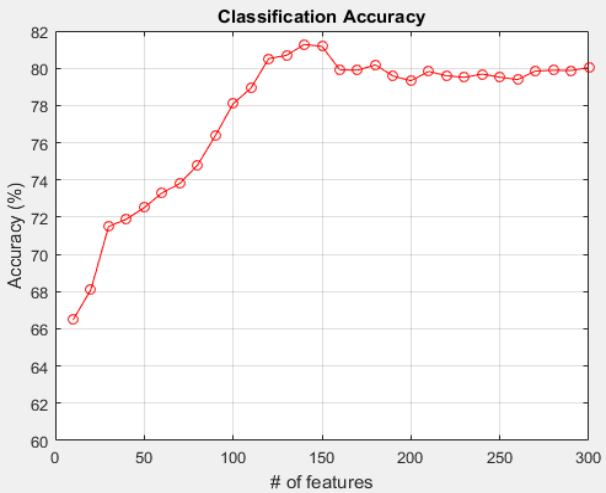


Figure V.5: Classification accuracy for steering wheel data analysis.

V.2.3 Feature level integration

For the feature level integration, the vector of steering wheel angles is added to the vector of facial features. The maximum classification accuracy is already obtained by using a vector size of 140 features for the steering wheel data analysis. Hence, the same vector size of steering wheel angles is retained. Then, the number of facial features was varied from 10 to 300 with the interval of 10, and the classification parameter settings was also varied to evaluate the classification accuracy of the system illustrated in Figure V.6. The result shows that classification accuracy of all parameter settings increases rapidly before the number of facial features reaches 110. After that, it does not have many changes. The classification of feature level integration using facial features as the ROI along with an SVM classifier also provides better results than that using eye region as the ROI.

In Table V.1, the maximum accuracies achieved by the three different approaches

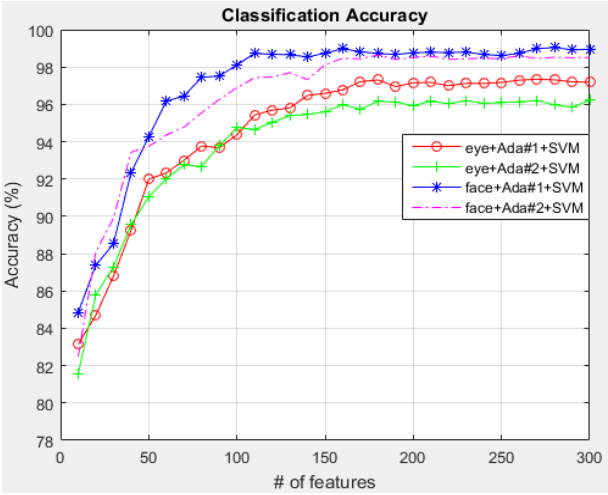


Figure V.6: Classification accuracy for feature level integration.

are compared. It can be seen that integrating the two input sources (driver's face and steering wheel) at the feature level helps us obtain the best accuracy. A maximum accuracy of 99.05% for 280 facial features selected with the averaging threshold computation, face region used as the ROI, and an SVM used for classification was achieved.

Table V.1: Maximum classification accuracy with different approaches.

Types of systems	**Accuracy**
Facial expression analysis	95.36%
Steering wheel analysis	81.28%
Feature level integration	99.05%

The labeled testing data from two subjects was fed to the trained model to output the total number of correct and incorrect classifications. Then, the true/false positive and true/false negative rates were calculated. Table V.2 shows the classification results of the actual and predicted data in two categories: drowsy and non-drowsy status.

Table V.2: Confusion matrix for feature level integration.

Actual Data	**Predicted Data**	
	Non-drowsy	Drowsy
Non-drowsy	98.92%	1.08%
Drowsy	0.84%	99.16%

V.3 Application of drowsiness detection in the simulation testbed

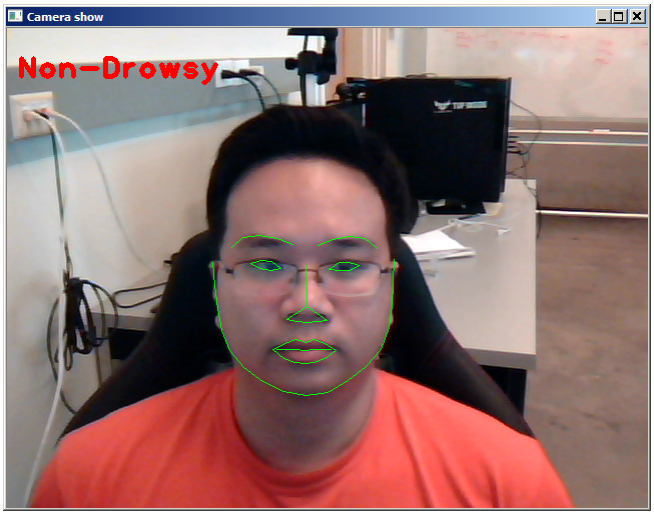
The experimental setup is shown in Figure V.2. A driver sits behind the steering wheel to control the vehicle in the simulator which sends the steering wheel data to a C++ program to combine with the images of the driver's face into the total feature vector which is utilized to detect his state by the C++ program shown in Figure V.7. After the driver's status is determined, it is sent to the driving simulator. If the driver is non-drowsy, the simulator continues letting the driver control the vehicle. Otherwise, the simulator switches the control mechanism into the autonomous driving mode. This manual/autonomous switching mechanism is one of the pre-built functionalities of the simulator.

V.3.1 A simple collision avoidance system in the simulation testbed

During the autonomous driving mode, a simple collision avoidance algorithm is implemented by adjusting the ego vehicle's velocity, v_h, when there is another vehicle blocking it in the front or approaching it from behind. If the ego vehicle is not in the dangerous range, R, to the other vehicle, its velocity is determined by a predefined constant. Otherwise, v_h must be adjusted to maximize the time to collision (TTC)

$$v_h^* = \underset{v_h}{\operatorname{argmax}} TTC = \underset{v_h}{\operatorname{argmax}} \frac{d}{|v_h - v_a|} \tag{V.1}$$

where d is the distance between the ego vehicle and the other vehicle with velocity v_a; v_h^* is the optimal velocity for the ego vehicle. The optimal solution for Equation



(a) Non-Drowsy

(b) Drowsy

Figure V.7: The driver states while driving.

(V.1) could be found easily as $v_h^* = v_a$. So, when the ego vehicle enters the dangerous range, R, the algorithm adjusts its velocity according to the other vehicle. Hence, this simple collision avoidance system can be formulated as:

$$v_h(t) = \begin{cases} v_h(t-1) \text{ if } d(t-1) > R \\ v_a(t-1) \text{ if } d(t-1) \leq R \end{cases} \tag{V.2}$$

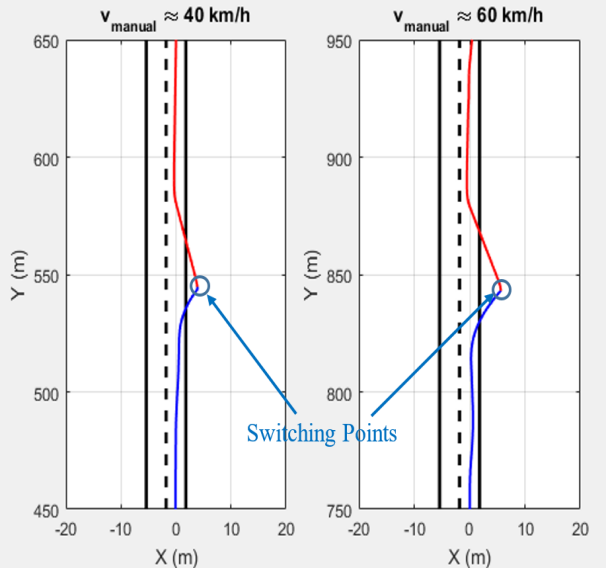


Figure V.8: The trajectories of the ego vehicle at the switching moment from manual to autonomous driving with different speeds of manual driving.

This simple collision avoidance algorithm just reflects the interaction between the ego vehicle and another vehicle. For interactions with multiple other vehicles, a more complex collision avoidance algorithm presented in Chapter VII is employed.

V.3.2 Switching between manual and autonomous driving in the simulated testbed

In this experiment, only the switching between manual and fully autonomous driving when the driver gets drowsy is considered. So, the behavior of the ego vehicle during the autonomous driving mode can be pre-programmed by specifying its velocity and lane position. Note that the steering of the ego vehicle is autonomously controlled to follow the preferred lane. So, at the switching moments from manual to autonomous driving, if the ego vehicle is already outside of the preferred lane position, the simulator would steer the vehicle to drive back to the desired lane based on its own dynamics.

It is noticed that the speed of manual driving may affect the switching behavior.

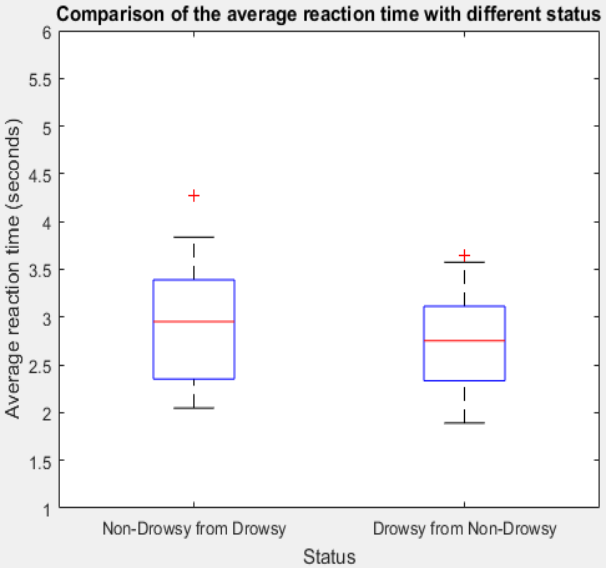
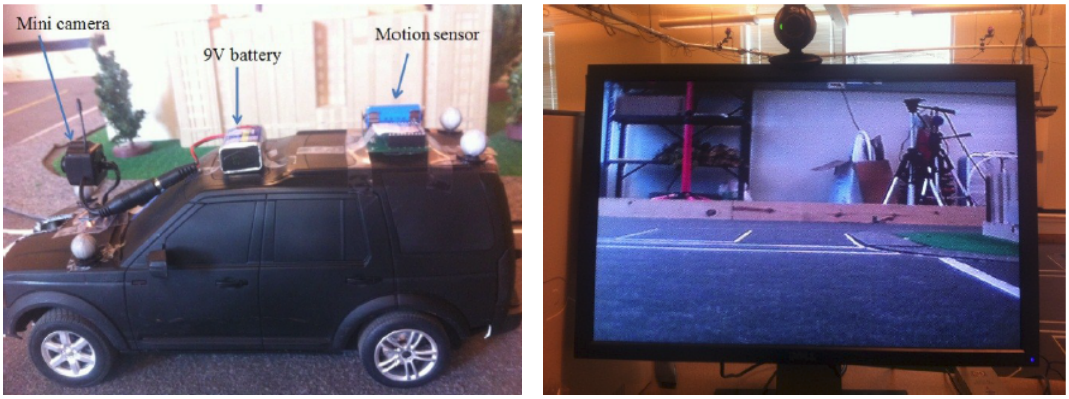


Figure V.9: Recognition time when driver's status changes.

Figure V.8 shows the trajectories of two experiments in which the simulated vehicle switched from manual to autonomous driving with different manual driving speeds. The blue curves represent the manual driving trajectories. At the end of the blue curves, the driver got drowsy and the vehicle switched to the autonomous driving mode whose trajectories are represented by the red curves. The results show that when the driver is drowsy and driving with higher speed, the vehicle tends to go off the road further. This makes the vehicle take longer to recover back to a preferred trajectory.

The recognition time was also investigated. In this dissertation, this time is defined as the elapsed time from the moment when the driver changes his/her status (from drowsy to non-drowsy or vice versa) until when the program detects the change of his/her status. Figure V.9 shows the recognition time of the system when the driver's status changes from the other one. It can be seen that the average recognition time is approximately 3 seconds. This is true because the driver's facial expressions at the transition moments are sometimes similar to each other. For example, at the moment a driver's begins getting drowsy, some of his/her facial expressions are



(a) (b)

Figure V.10: Manual driving experiment setup. (a) The miniature camera setup, (b) An image streamed back from the miniature camera.

similar to that when he is non-drowsy. Therefore, the drowsiness detection system takes approximately 3 seconds to overcome this moment completely to obtain correct classifications.

V.4 Application of drowsiness detection in the physical testbed

To validate the proposed algorithms, experiments on the physical testbed were conducted. Since the proposed system consists of many sub-systems, the functionalities of all sub-systems must be verified before assembling them into the overall system. First, the manual driving control is tested. Second, the autonomous control algorithm is tested with an RC car tracking a figure-eight trajectory. Finally, all subsystems are put together and the overall system is tested with the switching control based on drowsiness detection.

V.4.1 Manual driving

The signal from the miniature camera which was placed on the RC car's hood was tested. This camera was mounted so that it can provide the front driving view of the RC car. The miniature camera setup is displayed in Figure V.10a. Figure V.10b

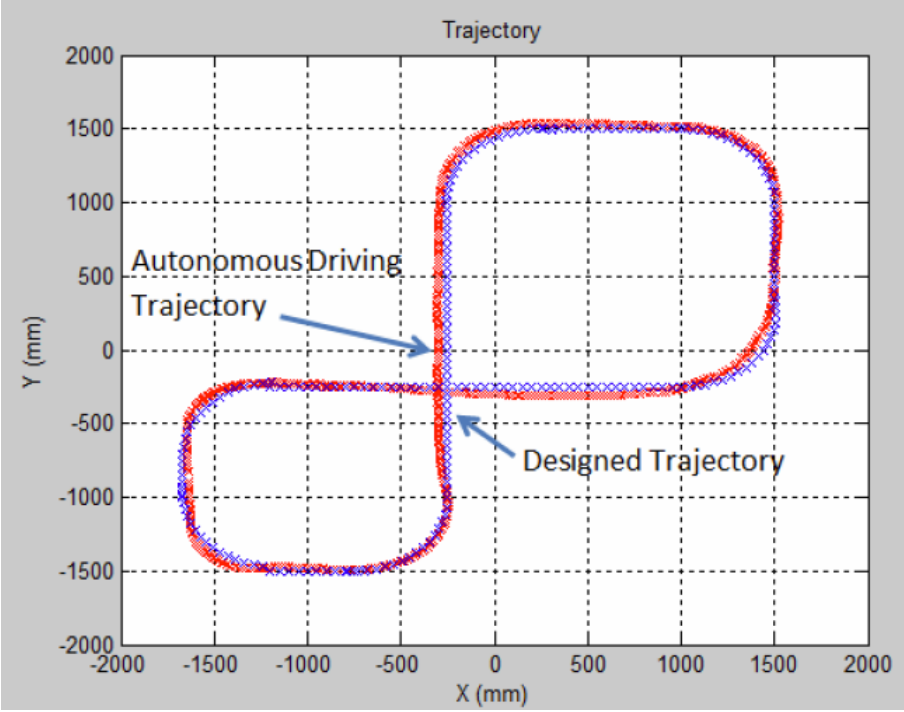


Figure V.11: Trajectories of the autonomous RC car and the virtual vehicle.

shows an image from the mini camera. In manual driving, a human driver controls the G27 Logitech Racing controller to drive the RC car manually. While looking at the video streamed from the miniature camera, the driver can control the RC car to run in the arena. After running several rounds, it noticed that the run time for the mini camera to function correctly (before the battery runs out) is around 15 – 20 minutes. During this time, the experiments must be executed so that the driver can see the images and control the car correctly. The reason, a 9V battery is utilized is that it is small and light. Other higher power supplies would make the RC car heavier, create more frictions to the ground and slow down the RC car's movement.

V.4.2 Autonomous driving

In this experiment, one RC car is used to test the tracking algorithm in the designed figure-eight trajectory. Before the testing, some parameters were set as follows: the desired distance between the RC car and the virtual vehicle d_ρ is 300 mm;

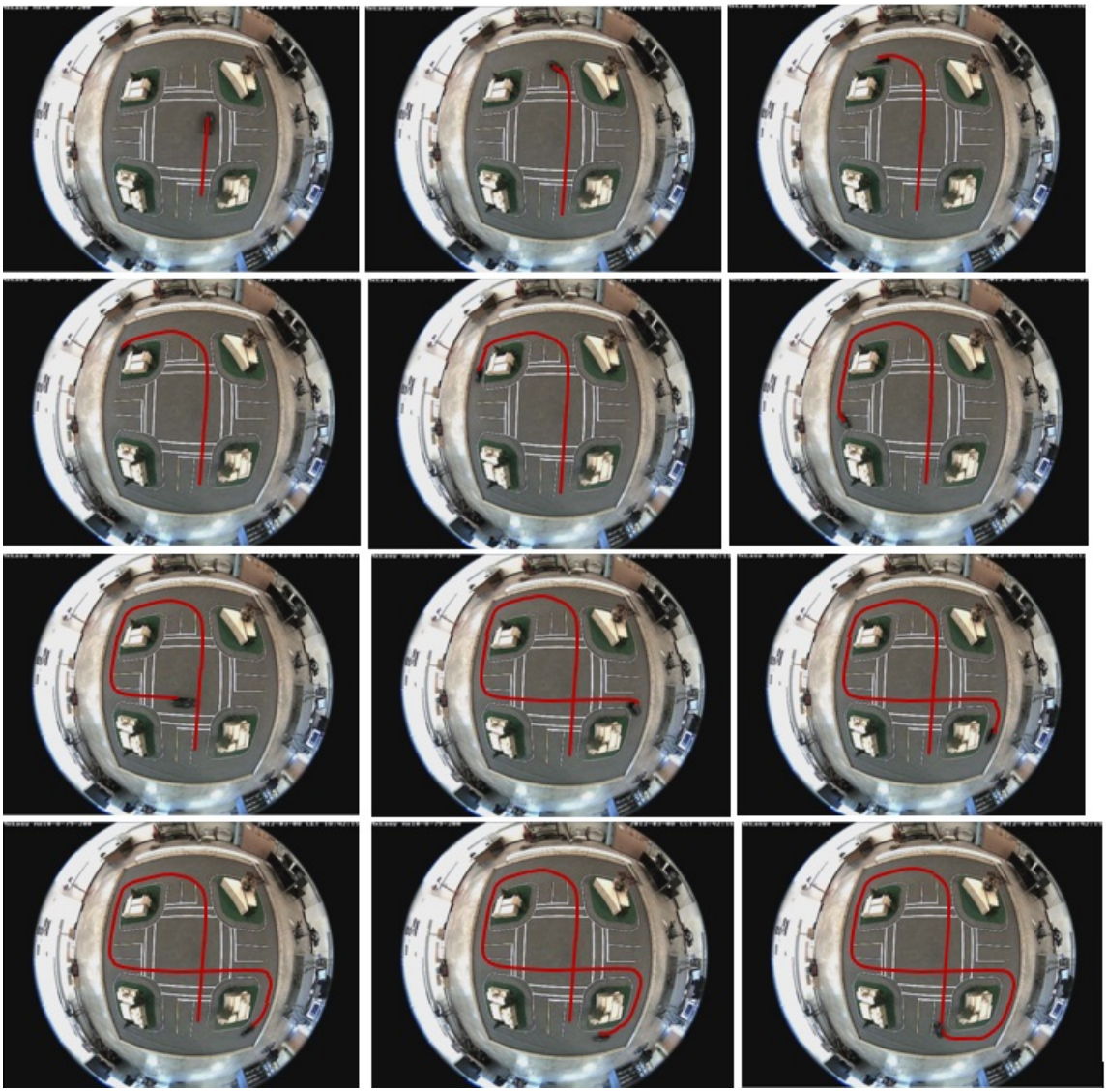


Figure V.12: Image of tracking the figure-eight trajectory from the roadside facility.

the constant, μ, for computing the virtual vehicle's velocity is 2; the initial velocity of the virtual vehicle is 0; the constants for the steering controller $k_p = 1, k_d = 0.8$; the desired velocity parameter used to send to the RC car is 80; the desired velocity of the RC car is 250 mm/s; the constant for controlling the RC car's velocity, $K = 2$ (in straight segments) and $K = 2.5$ (in arc segments). The definitions of μ, k_p, k_d and K can be found in our previous work [169]. The tracking result of the autonomous control algorithm is shown in Figure V.11. The RC car can complete the figure-eight trajectory by tracking the virtual vehicle that runs along the designed trajectory; even

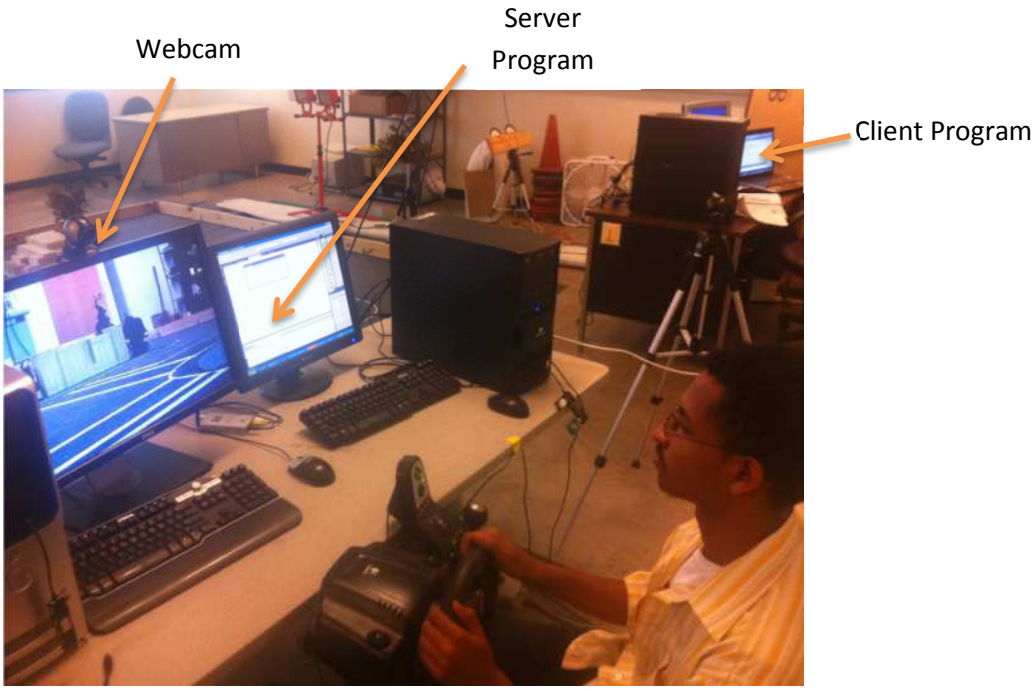


Figure V.13: Manual/autonomous switching experimental setup.

though at some points, the tracking is not precise. However, in overall, the RC car's trajectory and the designed trajectory along which the virtual vehicle runs almost overlap with each other. Figure V.12 shows the images captured from the roadside monitoring facility (Q24 camera).

V.4.3 Switching between manual and autonomous driving in the physical testbed

The switching control is determined based on the driver drowsiness conditions. The driver's status is continuously monitored by the client which runs the drowsiness detection by integrating the facial and steering wheel data. The output, a Boolean variable, from the drowsiness detection system is sent to the server to let it determine the suitable control of the RC car based on the driver's status. The switching process in the server is an infinite loop which always checks the Boolean variable sent from the client application. When the server receives a non-drowsy state from the client,

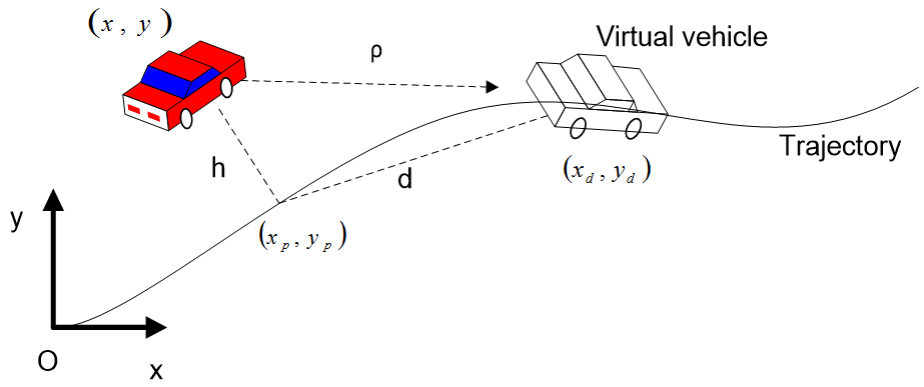


Figure V.14: Determining the virtual vehicle's starting location during the switching from manual driving to autonomous driving.

the server lets the driver manually control the RC car via the Logitech steering wheel controller. When the driver gets drowsy, the Boolean variable asserts that the server would control the RC car by tracking a predefined figure-eight trajectory. Before the switching moment, the RC car might go off the trajectory. Therefore, the point $(x_p,\ y_p)$ that has the shortest distance h from the RC car's current location to the trajectory must be determined firstly. Then, an offset distance d can be added to find the starting location $(x_d,\ y_d)$ for the virtual vehicle. The virtual vehicle's starting location is illustrated in Figure V.14. To autonomously drive the RC car, its speed and steering angle are computed according to the autonomous driving control algorithm developed in [155].

Scale-down experiments were conducted to validate the proposed manual/autonomous driving framework. The experimental setup is shown in Figure V.13. The webcam is mounted on the monitor to watch the driver's face. By integrating the facial expression and steering wheel data, the client program determines if the driver is awake or drowsy as shown in Figure V.7. The drowsiness detection runs on the client computer and the result is sent back to the server to trigger the switching from manual driving to autonomous driving. At the beginning, the control is set to be manual driving. Once a "drowsy" state is detected, the server switches to autonomous driving along

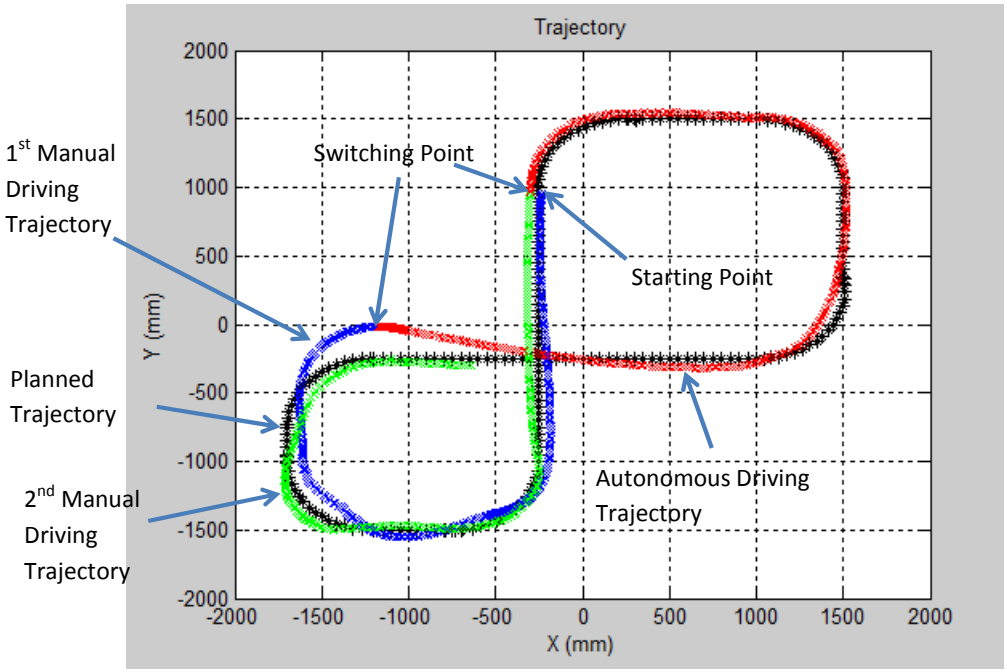


Figure V.15: The trajectory of the RC car.

the predefined trajectory. When the driver is awake again and the "non-drowsy" state is detected, the server lets the driver gain the control again. It is worth to note that at the time of experiments, the driver mimicked drowsiness instead of making the driver really sleepy.

The steering control data which captures the moment of the switching is plotted in Figure V.16. At the beginning, the steering and gas values varied. This shows that the driver was manually controlling the RC car at this time. After 300 samples (15 seconds), the steering data stayed unchanged since the driver got drowsy and he did not do anything to control the RC car. Then, the RC car was controlled autonomously by the server to track the predefined trajectory. After 400 samples (20 seconds), the steering and gas data changed again which means the driver was detected to be non-drowsy and could manually control the RC car. During the experiment, the brake pedal was not pressed, so its value remained unchanged.

Figure V.15 illustrates the trajectories that the RC car traveled by manual driving (blue curve and green curves) and autonomous driving (red curve) modes. The black

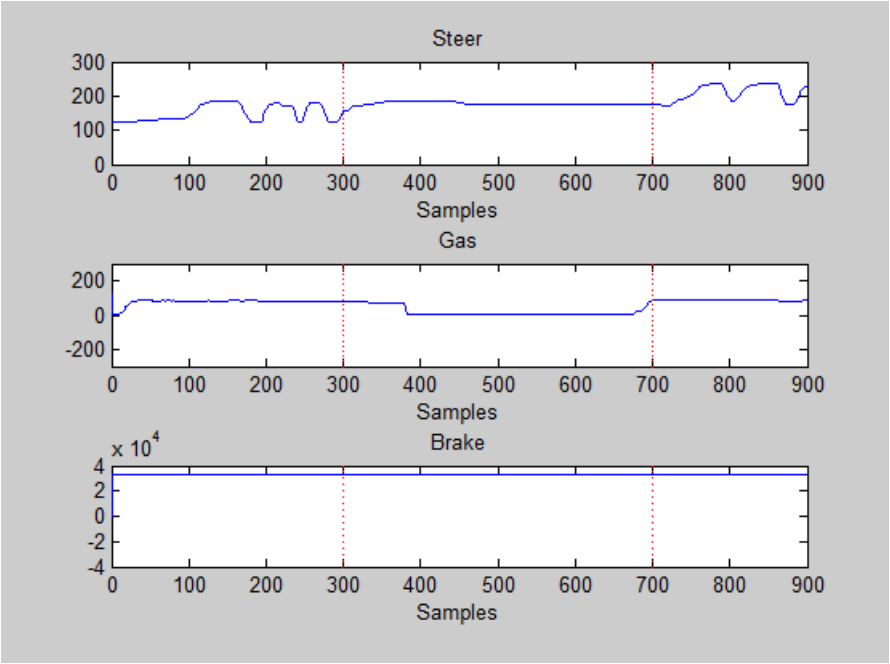


Figure V.16: The racing wheel data. The left red dotted lines indicate the moment of switching from manual to autonomous driving. The right red dotted lines indicate the moment of switching from autonomous to manual driving.

curve represents the planned trajectory. The RC car was first started at the location (-250, 1000) and manually controlled by the human driver to run downward along the blue curve. The end of the blue curve indicates the point where the driver got drowsy and drove the RC car outside of the lane. At this moment the server autonomously controlled the RC car and started tracking the predefined trajectory along the red curve. At the end of the red curve, the driver woke up. This is the point that indicates the switching moment from autonomous back to manual driving. Then the driver could control the RC car to run along the green curve. The recovering time depends on how far the RC car runs away from the predefined trajectory. In this experiment, the recovering time was around 4 seconds.

V.5 Summary

This chapter presents a driver assistance framework which monitors the driver's status. If the driver is non-drowsy, he/she can manually control the vehicle. Otherwise, it is switched to autonomous driving mode to run along a predefined trajectory. This framework demonstrates that intermittent autonomous driving can be adopted as a mechanism to prevent accidents in certain abnormal situations. Experiments were conducted on both the simulated and physical testbed to evaluate the proposed framework.

CHAPTER VI

REAL-TIME DETECTION OF DISTRACTED DRIVING BASED ON DEEP LEARNING

Driver distraction is a general case of driver drowsiness. This chapter presents the distracted driving detection system and its application on the assisted-driving testbed. The distraction detection system could be implemented to the collaborative driving framework in the future.

VI.1 Distraction detection

Four different Convolutional Neural Networks (CNNs) are adopted to recognize the various distracted driving behaviors which consist of the following 10 types:

- Safe driving
- Texting on the phone using the right hand
- Talking on the phone using the right hand
- Texting on the phone using the left hand
- Talking on the phone using the left hand
- Operating the radio
- Drinking
- Reaching behind
- Doing hair and makeup

Figure VI.1: An example image of distracted driving.

- Talking to passenger

In this section, the CNN models are described firstly. Then the training techniques are discussed.

VI.1.1 CNN models

CNN models are adopted to classify driving activities. The input to each CNN model contains the pixel values of the input image. Full-color images with dimensions of $640 \times 480 \times 3$ are utilized as inputs. This means that the neurons are structured as a 3D volume for R, G and B channels. Figure VI.1 shows an example input image of a distracted driving activity. The following subsections describe the four CNN models in more detail.

VGG-16 model

VGG-16 is a 16-layer CNN developed by Simon *et al.* for image recognition in the 2014 ImageNet large-scale visual recognition challenge (ILSVRC) [42]. 3×3 filters are used for all convolutional layers. The network accepts the input image with a dimension of 224×224. The image is passed through a sequence of 16 convolutional

layers. A multilayer perceptron (MLP) classifier including 3 fully-connected (FC) layers is used in addition to the convolutional layers to perform the classification. Rectified linear unit (ReLU) layers and max-pooling layers are used in the whole network to prevent the overfitting problem.

AlexNet model

The AlexNet architecture designed by Krizhevsky was the winner of the 2012 ILSVRC with an error rate (ϵ) of 15.3% [43]. The network includes 2 streams of 5 convolutional layers, 3 max-pooling layers, 3 dropout layers, and 3 fully-connected layers. The network was designed to classify 1000 categories of objects. The receptive field sizes are 11×11 and 5×5 for the first and second convolutional layers, respectively, and 3×3 for the last three convolutional layers. In addition, it consists of local response normalization layers, ReLU layers, and overlapping max-pooling layers.

GoogleNet model

GoogleNet is a 22-layer CNN developed by Google, one of the winning teams in the 2014 ILSVRC [44]. It was designed based on a local network topology. This new model pays close attention to memory and power usage. GoogleNet is designed to be very deep with 22 layers when counting only the layers with parameters. Another characteristic of GoogleNet is that a new local Inception module was introduced into the CNN. The basic idea of this module is to find the optimal local construction and to repeat it spatially. One of the main beneficial aspects of this architecture is that it allows the number of units to increase at each stage significantly without any uncontrolled blow-up in computational complexity. Therefore, the CNN can be very deep but still efficient in training. Each Inception module is a combination of 1×1, 3×3, and 5×5 convolutional layers along with 3×3 max-pooling layers. Moreover, 1×1 convolutional layers are applied to computation reductions because they include

Table VI.1: The architectures of the four CNN models. The (*) notation represents the layers on the parallel stream of the AlexNet model.

	VGG	**AlexNet**	**GoogleNet**	**ResNet**
Input	Image	Image	Image	Image
Convolutional part	conv3-64 conv3-64	conv3-48 conv3-48*	conv7-64	conv7-64
	Max pooling layer	Max pooling layer	Max pooling layer	Max pooling layer
	conv3-128 conv3-128	conv3-128 conv3-128*	conv3-192	3 blocks of [conv1-64 conv3-64 conv1-256]
	Max pooling layer	Max pooling layer	Max pooling layer	[conv1-128 conv3-128 conv1-512] with a stride of 2
	conv3-256 conv3-256 conv3-256	conv3-192 conv3-192*	Inception3-256 Inception3-480	7 blocks of [conv1-128 conv3-128 conv1-512]
	Max-pooling layer	conv3-192 conv3-192*	Max-pooling layer	[conv1-256 conv3-256 conv1-1024] with a stride of 2
	conv3-512 conv3-512 conv3-512	conv3-128 conv3-128*	Inception4-512 Inception4-512 Inception4-512 Inception4-528 Inception4-832	35 blocks of [conv1-256 conv3-256 conv1-1024]
	Max-pooling layer	Max-pooling layer	Max-pooling layer	[conv1-512 conv3-512 conv1-2048] with a stride of 2
	conv3-512 conv3-512 conv3-512		Inception5-832 Inception5-1024	2 blocks of [conv1-512 conv3-512 conv1-2048]
	Max-pooling layer		Average-pooling layer Dropout (40%)	Average-pooling layer
MLP classifier	Fully-connected layer-2048 Fully-connected layer-1024	Fully-connected layer-2048 Fully-connected layer-2048* Fully-connected layer-1024 Fully-connected layer-1024*	Fully-connected layer-2048 Fully-connected layer-1024	Fully-connected layer-2048 Fully-connected layer-1024
	Fully-connected layer-10	Fully-connected layer-10	Fully-connected layer-10	Fully-connected layer-10

less parameters as well as rectified activations before the expensive 3×3 and 5×5 convolutional layers. From the Inception module, local feature representations can be extracted using flexible convolutional kernel filter sizes with a layer-by-layer structure, which was proved to be robust and effective for the large scale high-resolution images. Furthermore, the padding strategy and precise designs after the Inception module operation help obtain a number of feature maps of the same sizes in terms of different scale convolutions and poolings. The feature maps are concatenated together by a concatenate-layer followed by each Inception module.

ResNet model

He *et al.* [45] from Microsoft Research proposed a new training framework called residual network (ResNet) for very deep CNN training. The network is adopted from the VGG network. The input image size for these networks is 224×224. The convolutional layers have mostly 3×3 filters and the design follows two rules: 1) For the same output feature map size, the layers have the same number of filters. 2) If the feature map size is halved, the number of filters is doubled in order to preserve the time complexity per layer. The downsampling operation is performed by the convolutional layers that have a stride of 2, hence no pooling layers. The network ends with a global average pooling layer and a 1000-class fully-connected layer with the softmax function. In this network, a shortcut connection is added to each building block (two or three consecutive convolutional layers). Because of this new training framework, they are able to train very deep networks, with 18, 34, 50, 101, and 152-weight layers, without encountering the degradation problem.

VI.1.2 Model training

Training the CNN models for distracted driving detection is not a trivial task. Transfer learning is a commonly used technique in training deep neural networks

[170]. The networks which were pre-trained with millions of images can be re-used. To this end, the last FC layers were replaced with a new Multi-layer perceptron (MLP) classifier. It is a type of ordinary neural networks which includes several fully-connected layers where each neuron is connected to all the outputs from the previous layer. The last FC layer computes the probability for each class. For multi-class classification, softmax regression is a popular choice. The new classifier consists of activation layers with 1024 and 2048 neurons as well as a 10-neuron FC layer which corresponds to the 10 driving behaviors. Transfer learning helps save a significant amount of time because we do not need to train the whole model from scratch. Table VI.1 shows the architectures of the four CNN models with the modified MLP classifier as the last layers.

One of the problems that transfer learning usually encounters is that it makes the model prone to over-fitting when the nature of the learned data is rather different than the training data. In this case, the model can be fine-tuned by unfreezing some pre-trained layers and train them along with the fully-connected layers. Fine-tuning may increase the training time but it could solve the over-fitting problem. The following hyper parameters were adjusted for each model: learning rate (α), weight decay (γ) and momentum (μ).

VI.1.3 Embedded computers for real-time implementation in the assisted-driving testbed

It is apparent that placing a powerful computer machine in a car is unrealistic due to the cost, power and size constraints. Embedded computers are cheaper, smaller and power efficient. Two embedded computers were utilized to set up the distraction detection system. Figure VI.2 presents the assisted-driving testbed which was developed based on the simulated testbed. The assisted-driving testbed was equipped with the two embedded boards that are utilized for the real-time distraction detection and

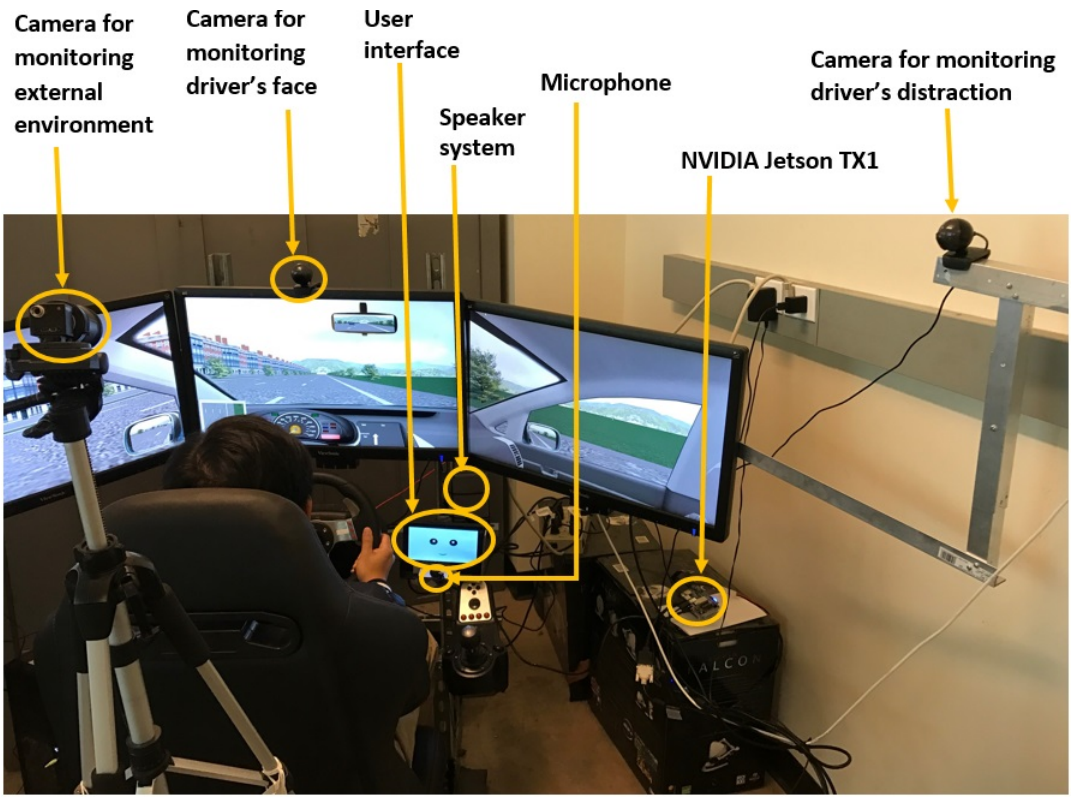


Figure VI.2: The assisted-driving testbed with embedded boards for real-time distraction detection.

warning system.

NVIDIA Jetson TX1

An NVIDIA Jetson TX1 board was employed for the distraction detection task. The board has a Quad ARM A57/2 1.73GHz CPU, 4 GB 64-bit LPDDR4 and a GPU (Graphic Processing Unit) with 256 Maxwell CUDA cores [171]. The GPU helps accelerate the distraction detection task which requires a signficant amount of computation as the input images are passed through the layers of the deep CNNs.

NanoPi M3

The NanoPi M3 is an ARM board developed by FriendlyARM. It uses the Samsung Octa-Core Cortex-A53 S5P6818 SoC at 1.4 GHz and 1GB 32bit DDR3 RAM

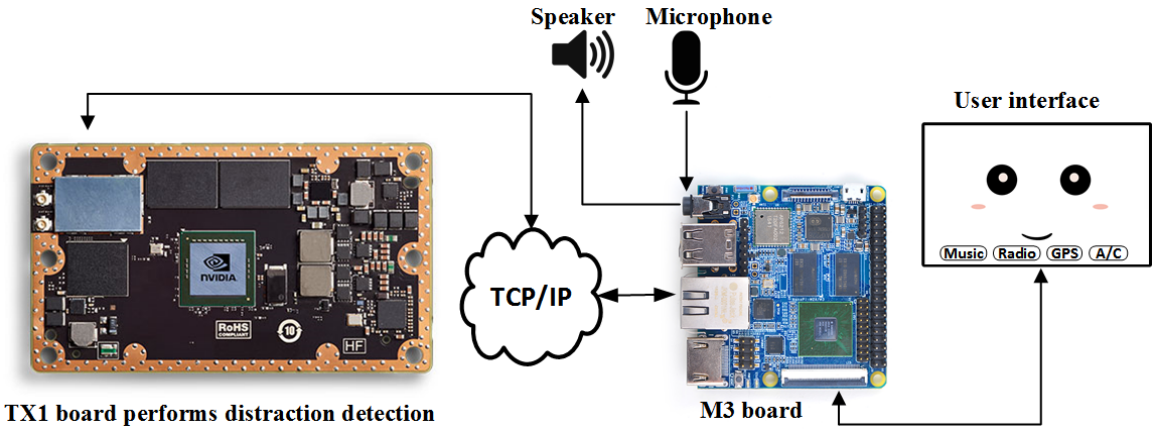


Figure VI.3: The hardware setup of the embedded computing system for distraction detection.

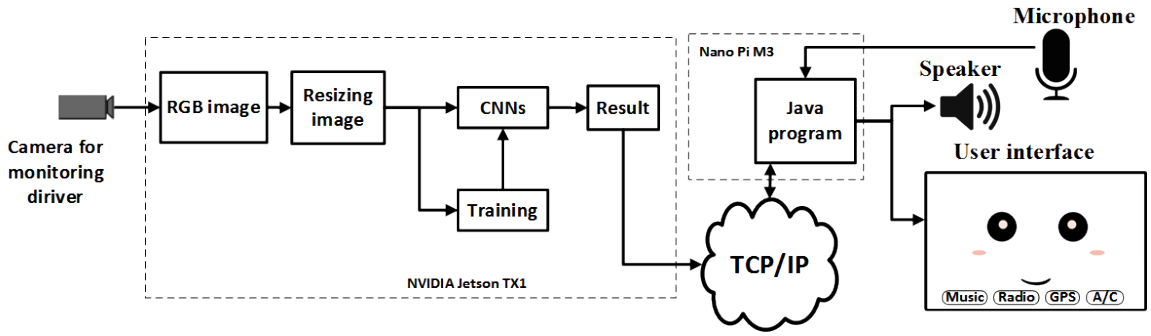


Figure VI.4: The software structure for distraction detection.

[172]. A touch screen display is connected to the NanoPi M3 board which runs the Android operating system as an interface to the human driver. The M3 board is placed behind the user interface so that it would not create any obtrusion while using the driving simulator. The NanoPi M3 board is connected to the NVIDIA Jetson TX1 board through an Ethernet cable so that they can communicate and share information. The detection result is transferred from the TX1 board to the M3 board through TCP/IP protocol. Then the M3 board can display the message on the screen or alert the driver about his/her driving status through the speakers. Figure VI.3 shows the hardware setup of the embedded computer system for distraction detection. Figure VI.4 illustrates the software structure of the system. The images captured by

Table VI.2: List of parameters used to train the CNNs.

Model	Learning rate	Weight decay	Momentum
VGG	5×10^{-5}	5×10^{-5}	9×10^{-1}
AlexNet	9×10^{-4}	1×10^{-5}	9×10^{-1}
GoogleNet	1×10^{-4}	1×10^{-5}	9×10^{-1}
ResNet	1×10^{-3}	5×10^{-5}	9.5×10^{-1}

the camera are firstly resized to fulfill the input size requirements of each CNN model. The resized images are then trained with each of the four CNN models. In the testing phase, the resized image is fed directly to each CNN model to classify the driver's activity which is sent to a Java program on the NanoPi M3 board. In normal conditions, the user interface operates as a control dashboard. The driver can use it to change the music, adjust the volume, choose the station, etc. However, when a distraction event is detected, the user interface can provide a reminder to the human driver to ask him/her to focus on the driving task.

VI.2 Experiments & results

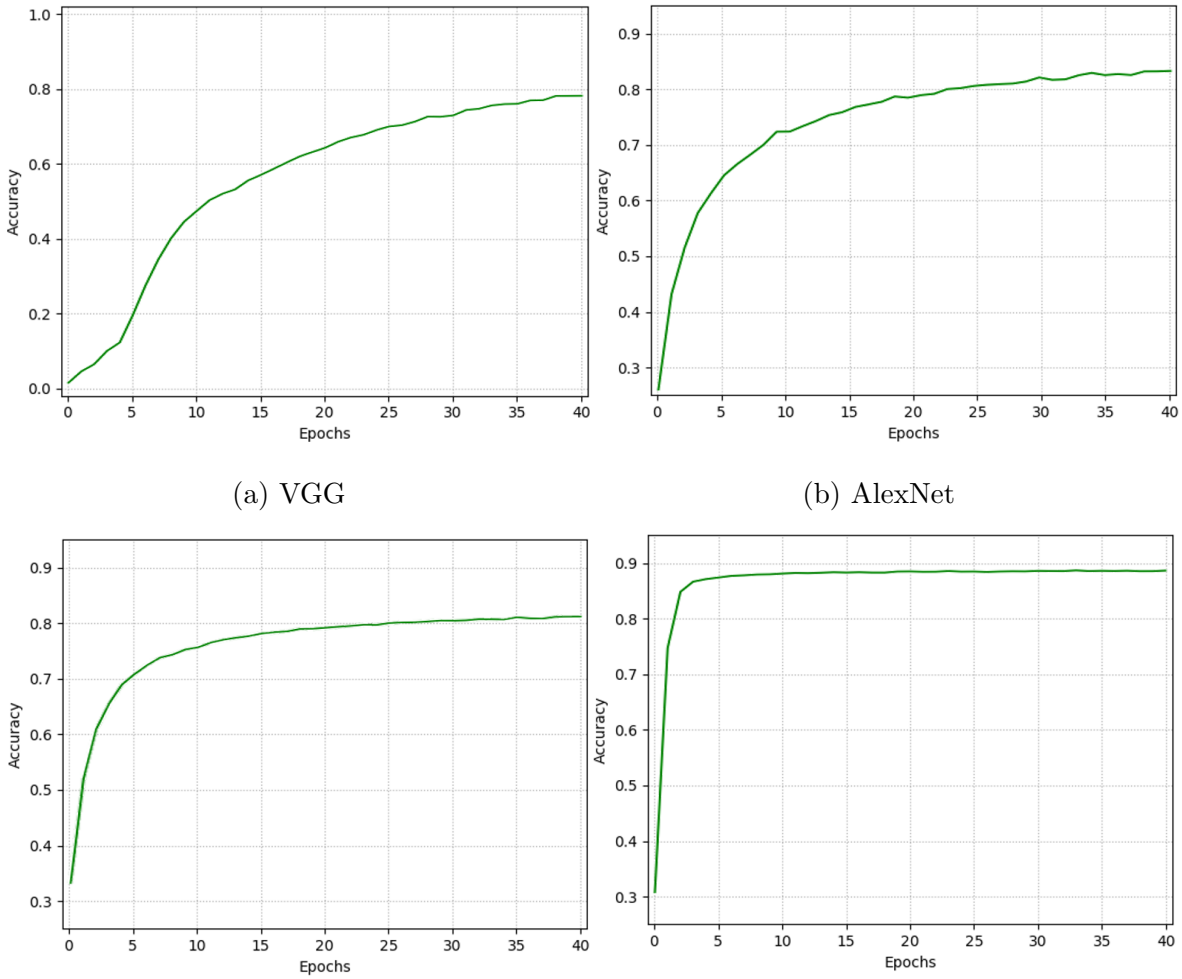
In order to validate and evaluate the CNN models in distraction detection, experiments were conducted on the assisted driving simulator. Figure VI.2 shows the complete setup of the simulator, which consists of multiple cameras for driver monitoring and environment monitoring. In the experiments, for the purpose of distraction detection, only the side camera was utilized to observe the driver's hand and body movement. Ten subjects were asked to drive the car and conduct the 10 activities mentioned in Section VI.1. For each activity, a five-minute video was recorded. Since the images are taken from the video clip, there exist similar postures that belong to the same activity. Therefore, only one image per second from the videos was selected. Hence, about 35,000 images were extracted. Then, data augmentation was performed

Table VI.3: Validation accuracy using the baseline approach.

Model	Accuracy
VGG	79%
AlexNet	81%
GoogleNet	83%
ResNet	88%

by shearing, rotating, width shifting, height shifting, and zooming the original images. Then, the dataset has about 200,000 images. 70 % of the data was used for training. 15 % of the data was used for validation and 15 % of the data was used for testing. The four CNN models were trained on a computer with the following configuration: Intel i7 4790 CPU, 16 GB RAM and NVIDIA GeForce GTX 970 GPU. To reduce overfitting on the training data, the following methods are applied during the training stage: data augmentation by cropping and flipping the images horizontally; applying the dropout technique [173] with a probability of 50% to the first two FC layers; using batch normalization with a size of 128; and then applying Stochastic Gradient Descent Learning (SGD) [174] with learning rate (α), momentum (μ) and weight decay (γ). Table VI.2 shows the training parameter values corresponding to each model.

For comparison purposes, a baseline approach was established. In this baseline approach, each of the four CNN models has 256 neurons in the fully-connected layers, as commonly adopted by most teams in the State Farm competition. Figure VI.5 and Table VI.3 show the validation results of the baseline approach. Compared to the baseline approach, the proposed approach have more neurons in the fully-connected layers as shown in Table VI.1. Table VI.3 shows the performance of each of the four CNN models.



(a) VGG

(b) AlexNet

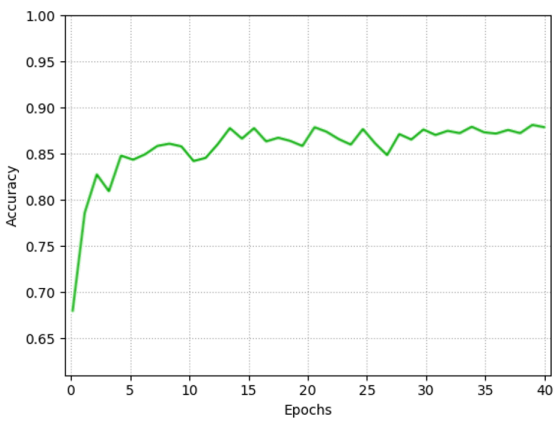
(c) GoogleNet

(d) ResNet

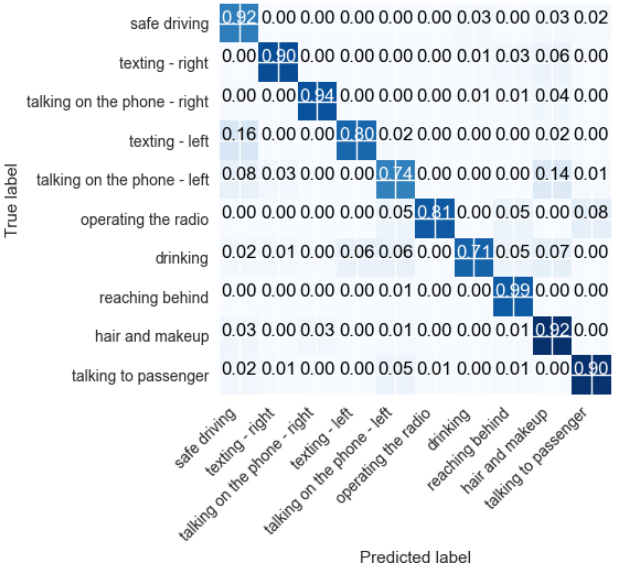
Figure VI.5: The validation results of the 4 CNN models using the baseline approach with 256 neurons at the fully-connected layers.

VI.2.1 VGG-16 model

Figure VI.6a illustrates the accuracy of the VGG-16 model over the validation dataset. Figure VI.6b presents the confusion matrix of the VGG-16 model. Most of the time, the activities were recognized correctly. There were some misclassifications between the class "talking on the phone - left" and "hair and makeup". This is because these two activities have many common postures especially when the phone is occluded by the driver's face completely. On the other hand, "texting left" was



(a) The validation accuracy.



(b) The confusion matrix of the results.

Figure VI.6: The evaluation results of the VGG-16 model.

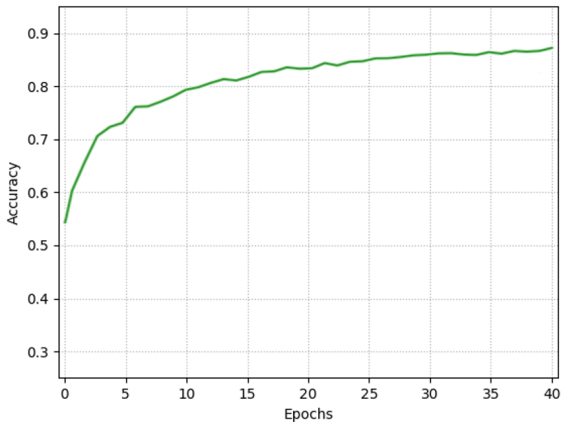
misclassified with "safe driving" in some scenarios when the steering wheel blocked the left hand. The evaluation of the trained model achieves an accuracy of 86%.

VI.2.2 AlexNet model

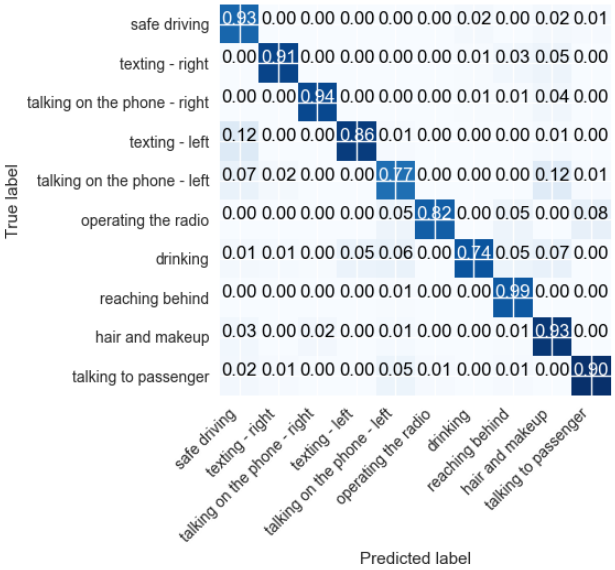
Figure VI.7a presents the validation accuracy of the AlexNet model after 41 epochs with a maximum accuracy of 88%. Similar to the VGG-16 model, the trained AlexNet model also had the most misclassifications in two scenarios such as between "talking with the phone - left" and "hair and makeup", and between "texting left" and "safe driving". Other than that, the behavior "drinking" was also misclassified with others such as "texting left", "talking to the phone - left", or "hair and makeup".

VI.2.3 GoogleNet model

Figure VI.8a shows the validation accuracy while Figure VI.8b illustrates the confusion matrix of the validation result. The maximum validation accuracy is 89%. The model is well-fitted. Similar to the VGG-16 and AlexNet models, the GoogleNet



(a) The validation accuracy.



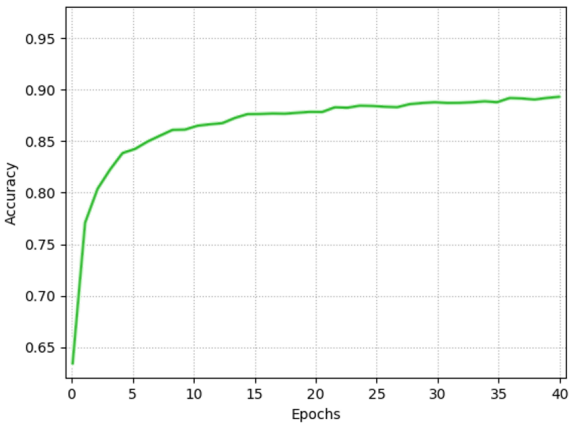
(b) The confusion matrix of the results.

Figure VI.7: The evaluation results of the AlexNet model.

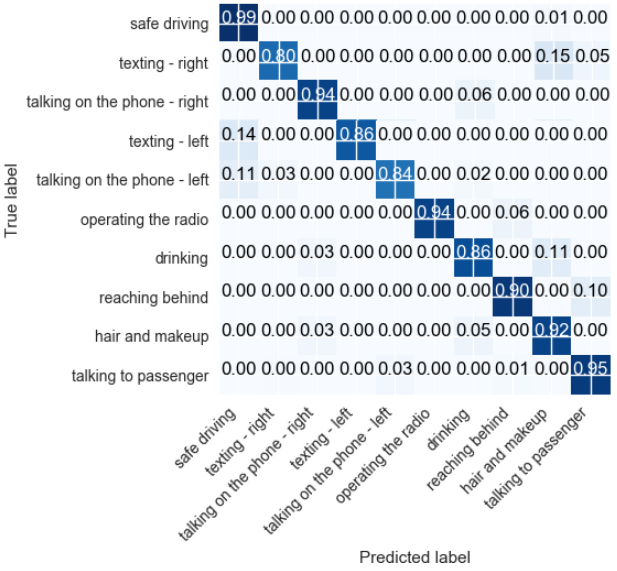
model had the most misclassifications between the classes of "texting left" and "safe-driving" because of their similarity in postures especially when the phone was covered by the steering wheel. On the other hand, the behavior "texting right" was misclassified to "hair and makeup". It is because the driver in some images put the phone too close to his/her face. This reason leads to the wrong prediction. In addition, "drinking" was also misclassified to "hair and makeup" because both behaviors had some similar postures.

VI.2.4 ResNet model

Figure VI.9a and VI.9b show the validation accuracy and the confusion matrix respectively. The maximum validation accuracy is 92%. It can be observed that the ResNet model converged faster than the other three models. This means that the deeper the model is, the faster it fits the dataset. The ResNet model has the most misclassifications of "texting right" with "hair and makeup" because they also have similar postures. Overall, for all of the four CNN models, the proposed approach

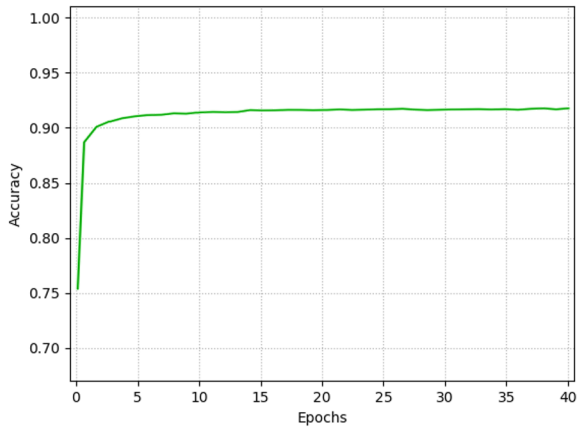


(a) The validation accuracy.

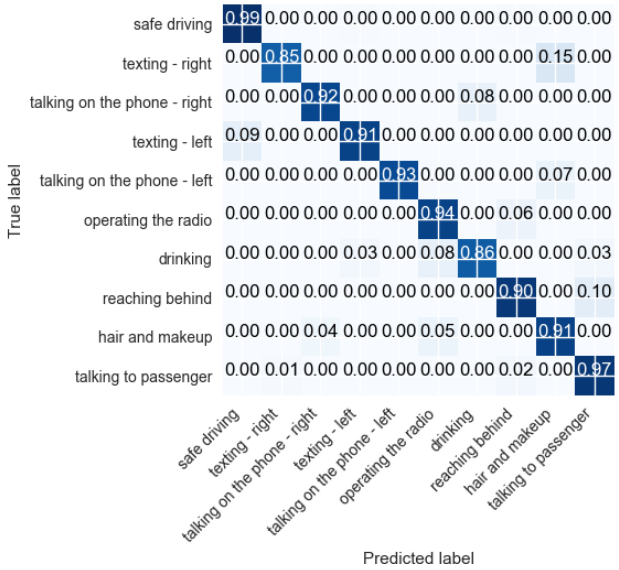


(b) The confusion matrix of the results.

Figure VI.8: The evaluation results of the GoogleNet model.



(a) The validation accuracy.



(b) The confusion matrix of the results.

Figure VI.9: The evaluation results of the ResNet model.

Table VI.4: Validation accuracy and maximum frequency of each model running on the TX1 board.

Model	Accuracy	Max frequency
VGG	86%	14 Hz
AlexNet	88%	12 Hz
GoogleNet	89%	11 Hz
ResNet	92%	8 Hz

achieves better performances than the baseline one.

VI.2.5 Performance on the embedded computer

The trained models were executed on the Jetson TX1 board to evaluate the real-timeness. Table VI.4 shows the validation accuracy and the maximum frequency of each model running on the TX1 board. Overall, all models can operate in real-time but with different calculation times. When the human driver gets distracted, the TX1 board sends the request immediately to the M3 board to trigger the voice alert which reminds the driver to focus on the driving task.

In terms of performance, the VGG-16 achieves the fastest frequency (14 Hz) but has the least accuracy (86%). This is because the VGG-16 has the simplest architecture which is a sequential model. Other models with more complex architectures achieve higher accuracies. The ResNet model obtains the highest accuracy of 92%. However, it is the slowest among the four models with a frequency of only 8 Hz and a delay of $\approx$1 second. We have noticed that, for a model to operate smoothly without any noticeable delay in the assisted-driving tested, it must obtain at least 11 Hz. The GoogleNet model can obtain such a processing frequency but with a slight loss of accuracy (3% reduction). Even though the GoogleNet model is a little slower than the AlexNet model (11 Hz vs. 12 Hz), it has higher accuracy than the AlexNet

model. Therefore, to balance accuracy and speed, the GoogleNet model is the best for distraction detection in the proposed driving simulation testbed.

VI.3 Summary

This chapter presents a distraction detection system which is based on different deep learning architectures including VGG-16,GoogleNet, AlexNet, and ResNet. An assisted-driving testbed was developed so that the distracted driving dataset could be collected. Experiments were conducted on the assisted-driving testbed to evaluate the trained models. The proposed approach achieves better performance than the baseline one. The distraction detection system operates on a Jetson TX1 embedded computer board in real-time with a frequency in the range of 8 Hz and 14 Hz and an accuracy in the range of 86% and 92%. To balance the performance in terms of both accuracy and frequency, the GoogleNet model which achieves a frequency of 11 Hz at an accuracy of 89% is selected. This system has the potential to be implemented in real cars, which can significantly reduce the traffic accidents caused by driving distractions.

CHAPTER VII

COLLISION AVOIDANCE SYSTEM (CAS)

This chapter discusses the video-based external risk assessment, the structure of behavior models, the CAS, and its evaluation on the simulated testbed. The Collision Avoidance System (CAS) is proposed based on Osipychev's work [46]. The CAS makes use of the external risk assessment to perform two general tasks. In the first task, we detect the behavior/intention of the other drivers and predict their trajectories in the near future. For that reason, behavior models which store the activities of both human-driven and autonomous vehicles are utilized. In the second task, the proposed CAS takes the probability distributions of intentions from other vehicles as inputs and chooses the best action through an optimization algorithm. This desired action is then applied to the vehicle dynamics using the vehicle's actuators such as steer, throttle and brake.

VII.1 Video-based external risk assessment

For most automated vehicles, monitoring the outside environment is an important component to enhance safety. In the simulated testbed system, this information can be extracted easily from the simulator. In order to enhance the proposed system's robustness, a video-based detection system is developed. This system relies on the sequences of images from a camera to detect lanes and pedestrians.

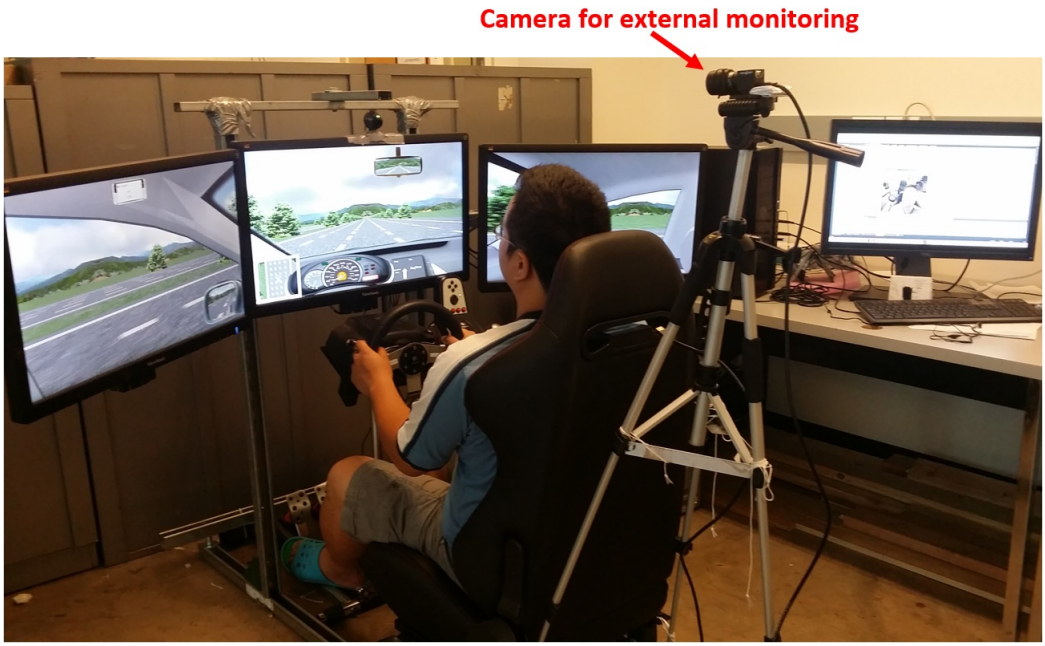


Figure VII.1: The overall experimental setup of the video-based external risk assessment system.

VII.1.1 Lane departure warning system

To determine a lane departure off a road, the lane and its type (solid or broken) are needed to be detected firstly. In this section, the lane detection system, the lane type detection, and the departure warning system are discussed.

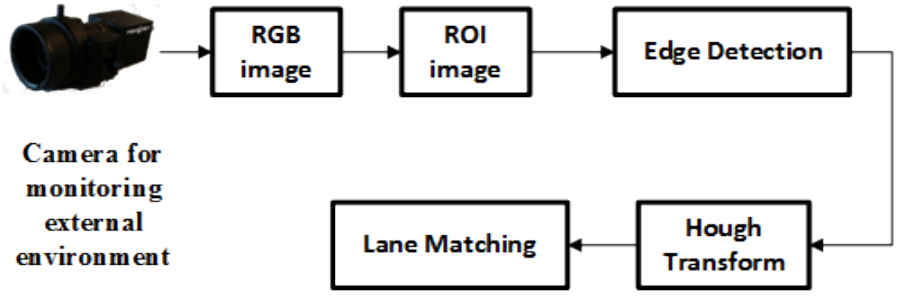


Figure VII.2: The overall system diagram of lane detection.

Lane detection

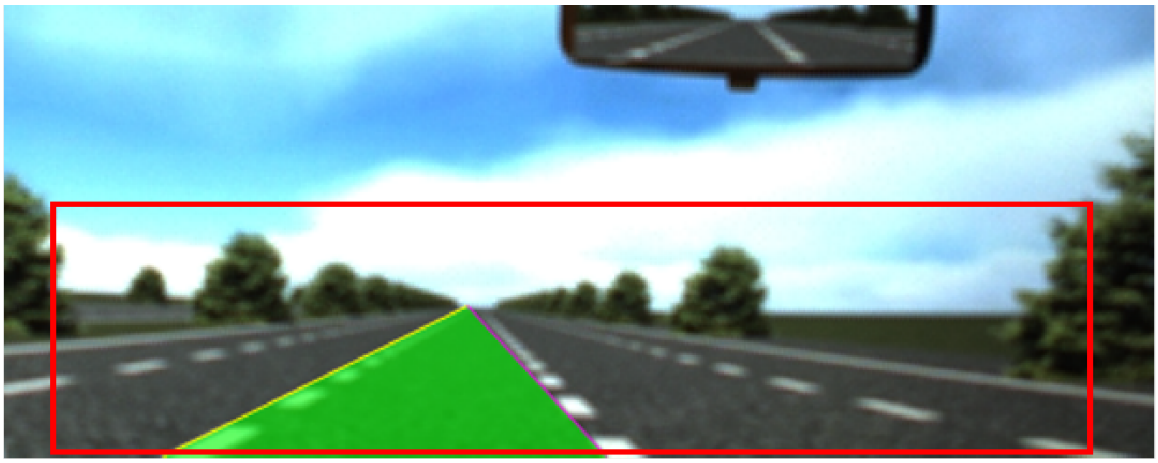
Figure VII.1 illustrates the experimental setup of the lane detection system. In most of on-board lane detection systems, the cameras are mounted on the windshield area and focus on the road region. However, in the simulated environment, mounting the camera on the windshield (too close to the center monitor) would make the camera out-of-focus and unable to cover the road region. Therefore, in this dissertation, a USB 3.0 PointGrey camera is placed behind the driver's seat. This camera has zooming functionalities and focuses on the road region only to enhance the detection performance.

In this dissertation, the lane detection system is inspired by MathWorks [175]. The overall system diagram is presented in Figure VII.2. After receiving the image stream from the camera, the system pre-processes the image by extracting the ROI where the algorithm should search for the lanes. Then the system detects the edges on the image which is converted to a binary image. The Standard Hough Transform is utilized to detect the straight lines created by the detected edges. In the lane matching block, the system counts the number of times each lane is detected and compares with the previously-detected lanes. If they are similar enough, the lane is updated and get ready for the next iteration.

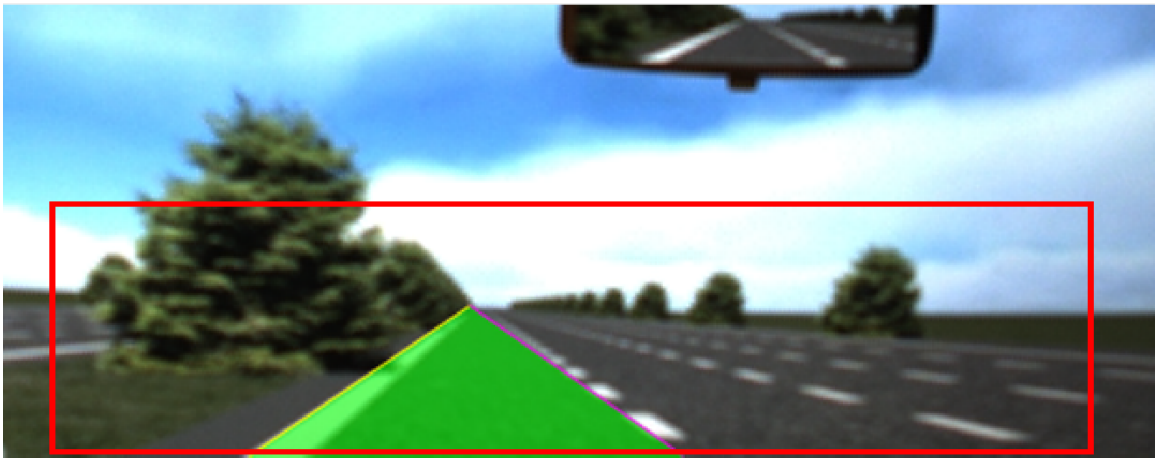
The lane detection results are illustrated in Figure VII.3 with the scenarios of broken and solid lines. The red bounding box represents the region of interest where the system searches for the lanes. The left and right lines are plotted in yellow and purple colors respectively. The area in the middle of the left and right lines is painted in green color to determine the current lane of the vehicle.

Lane type detection

In this dissertation, only road maps with white lane markings are considered to ease the detection of lane types. After detecting two lines, the total number of white



(a) Broken lines

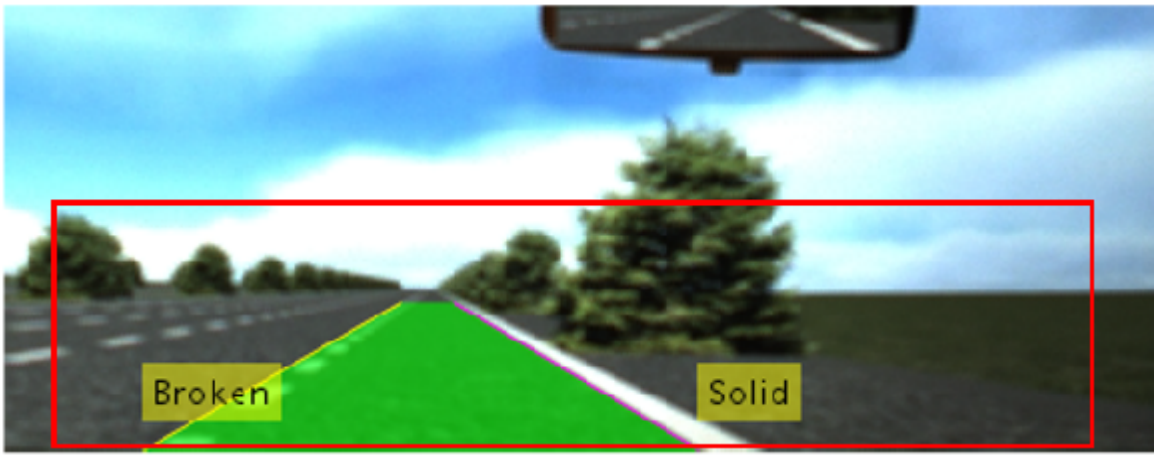


(b) Solid and broken lines

Figure VII.3: Lane is detected regardless of different line types.

pixels along each line can be counted to determine the type of that line. Note that, any pixel that has an intensity value greater than 200 can be considered to be a white pixel. If the total number of white pixels along a line is large enough, this line is considered to be solid. Otherwise, this line presents a broken line. Figure VII.4 shows the detected lane types.

(a) Detected broken lines



(b) Detected broken and solid lines

Figure VII.4: Lane type detection.

The overall lane departure warning system

The lane detection can help us determine the relative location of the vehicle to the lane so the system can determine if the ego vehicle departs from its current lane. Figure VII.5 shows the reference location of the vehicle and the lines. L_b and R_b are the locations where the left and right detected lines meet the bottom edge of the image. dl, dr and dc are the distances from the left margin of the image to L_b, R_b and the center of the vehicle, respectively. The lane departure warning system is

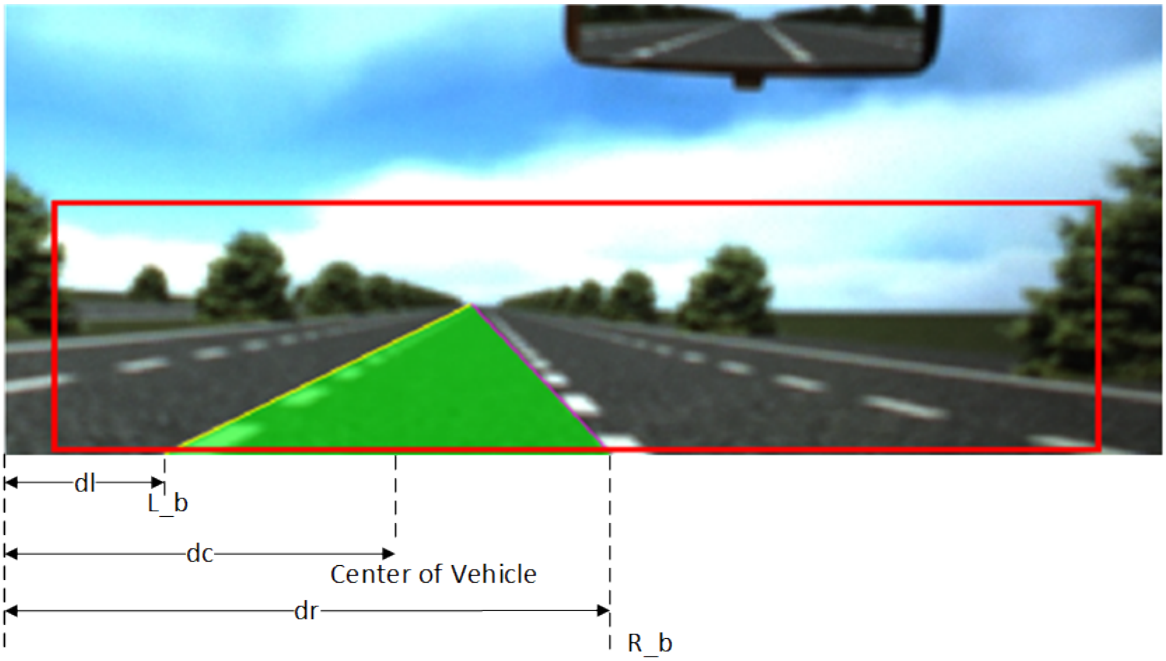


Figure VII.5: The reference location of the vehicle and the lines.

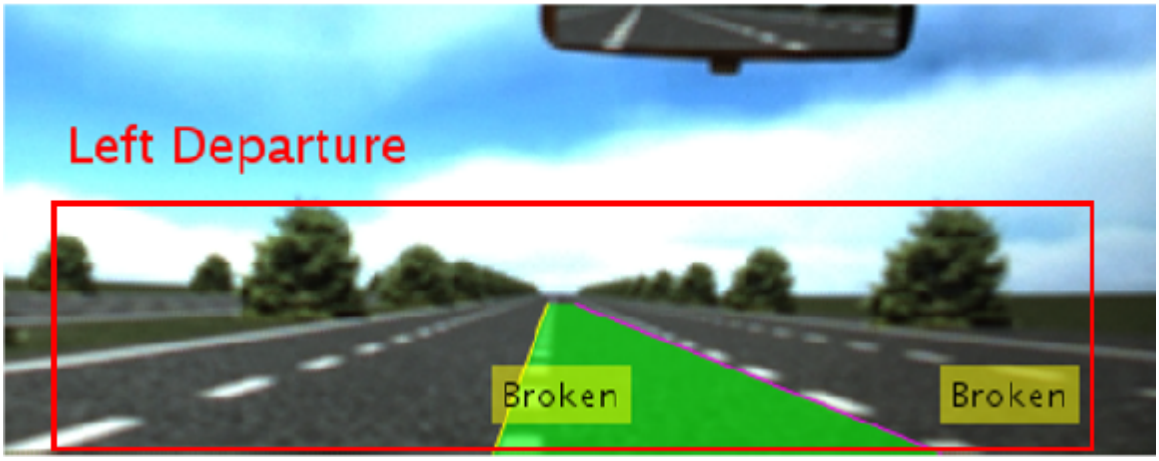
determined by the following equation:

$$Warning = \begin{cases} \text{Left Departure ,} & |dc - dl| < |dc - dr|, |dc - dl| < T_{warn} \\ \text{Right Departure ,} & |dc - dl| > |dc - dr|, |dc - dr| < T_{warn} \\ \text{Nothing ,} & \text{Otherwise} \end{cases} \quad \text{(VII.1)}$$

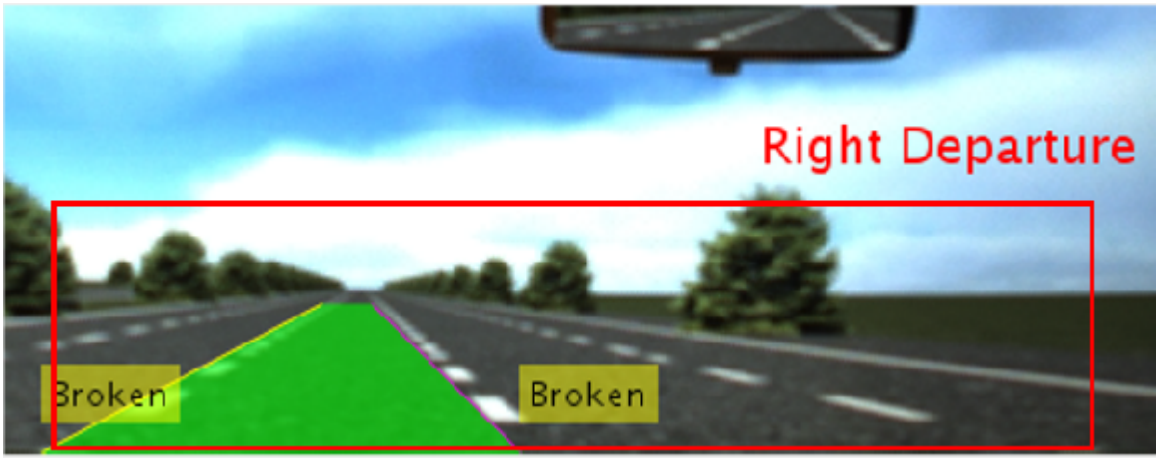
T_{warn} is the threshold distance to determine if the vehicle is too close to the left or right line. If the vehicle departs from its current lane, the system would display the warning message ("Left Departure" or "Right Departure"). Otherwise, the system would not display any warning. The results of the lane departure system are presented in Figure VII.6.

VII.1.2 Pedestrian detection system

Accident statistics indicate that 70 percent of the pedestrians involved in vehicle accidents were crossing the road in front of forward-moving vehicles [176]. Therefore, a pedestrian detection system typically uses a forward-facing camera. In this project,



(a) Left departure



(b) Right departure

Figure VII.6: Lane departure detection.

the lane and pedestrian detection systems use the same image source from the USB 3.0 PointGrey camera placed behind the human driver. The camera captures the video from the center monitor and sends it to an on-board computer for pedestrian detection. The experimental setup is presented in Figure VII.1. The camera is mounted so that the image can cover the road and pedestrians.

The overall system diagram is presented in Figure VII.7. After receiving the image stream from the camera, the system pre-processes the image by extracting the ROI where the algorithm should search for pedestrians using the Aggregated Channel

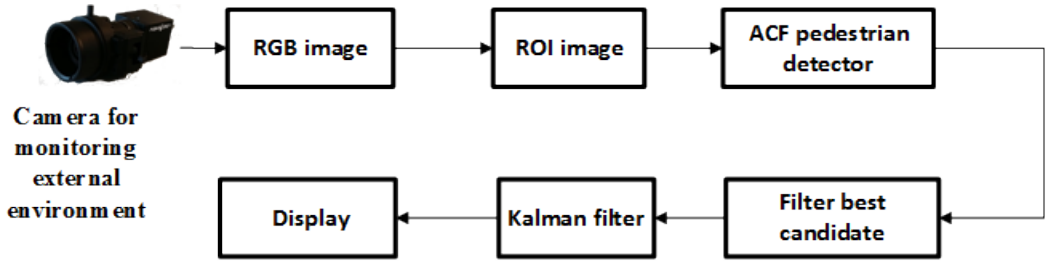


Figure VII.7: The overall diagram of the pedestrian detection system.

Features (ACF) pedestrian detection. Non-maximum suppression is used to find the best candidates in the image. Then a Kalman filter is utilized to track the detected pedestrian. The detail of the ACF pedestrian detection and the results are presented in the next sub-section.

The ACF pedestrian detection

The ACF pedestrian detection was proposed by Dollár *et al.* [117]. In this detection algorithm, the features of channels are aggregated for better detection. The channels include normalized gradient magnitude, histograms of oriented gradients (six channels), and LUV color channels. The detailed ACF pedestrian detector is displayed in Figure VII.8. Given an input image I, the 10 channels $C = \Omega(I)$ are divided into 4×4 blocks; and pixels in each block are summed. A feature pyramid is a multi-scale representation of an image I where channels C_s are computed at every scale s. Scales are sampled evenly in log-space, starting at $s = 1$, with typically four to 12 scales per octave. Note that an octave is the interval between one scale and another with half or double its value. The feature pyramid is proposed and proven to be fast constructed in [117]. Then a Boosting detector is used to train and combine decision trees over these features to distinguish objects from the background by using the multiscale sliding-window approach.

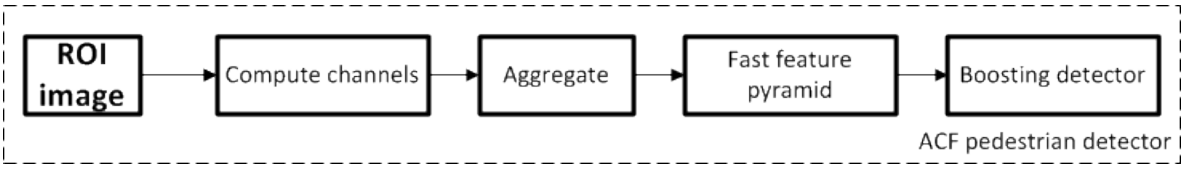
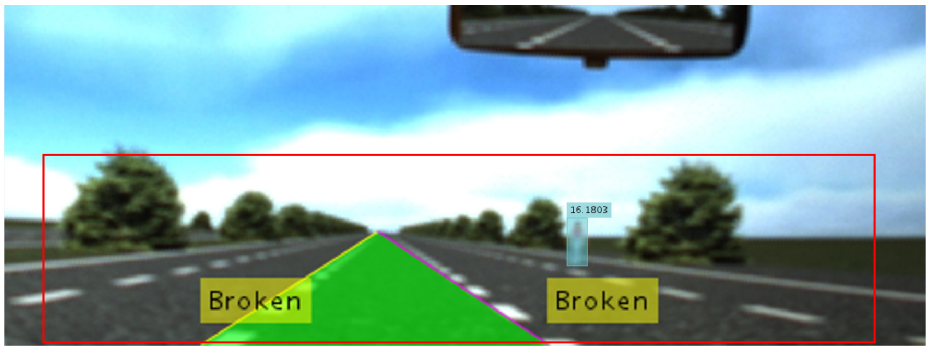


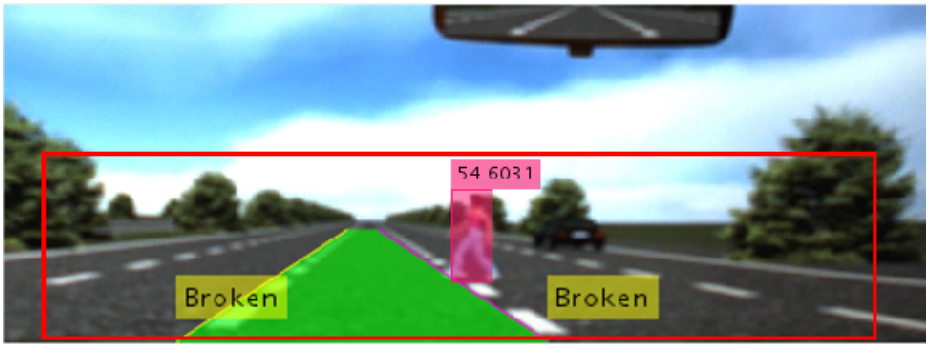
Figure VII.8: The overall diagram of the ACF pedestrian detector.

Results

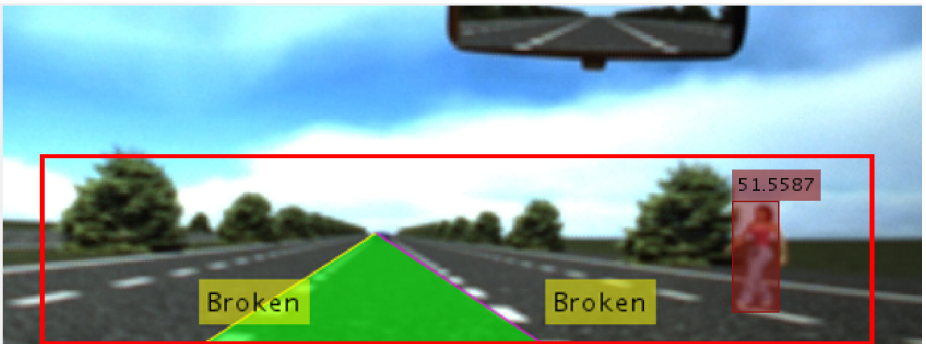
Figure VII.9 shows the detection results at different locations on the simulated testbed. Multiple-pedestrian detection results are presented in Figure VII.10. The region of interest is limited to 420 × 100 pixels. In this dissertation, the pedestrian and lane detections are executed in the same MATLAB code. With the given size of ROI, the system can detect pedestrians correctly within 15 m with a frequency of ≈20 fps. If the pedestrian is too far, the occupied pixels are too small, the system does not have enough information for detection. To solve the long-distance detection problem, the up-sampling technique can be performed by doubling the size of the ROI. Therefore, the detection distance can increase up to 30 m with a trade-off speed decreasing to ≈15 fps.



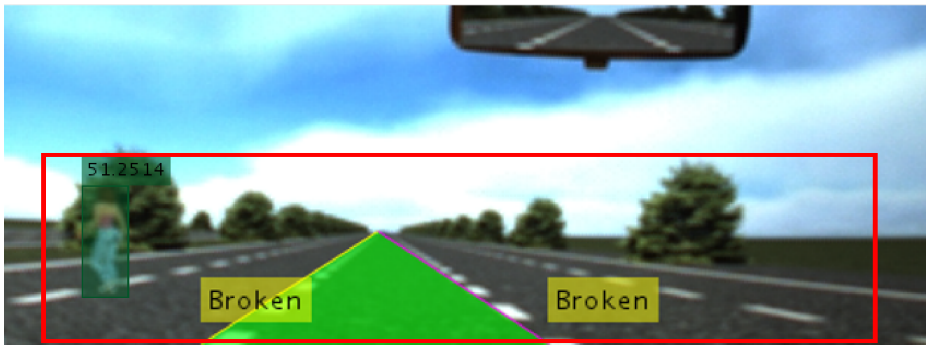
(a)



(b)

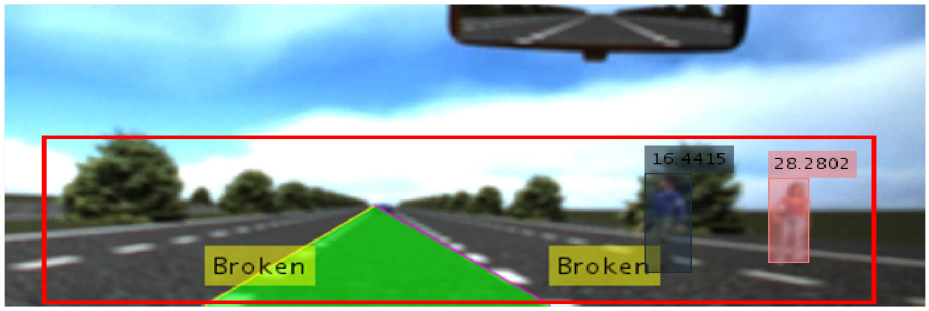


(c)

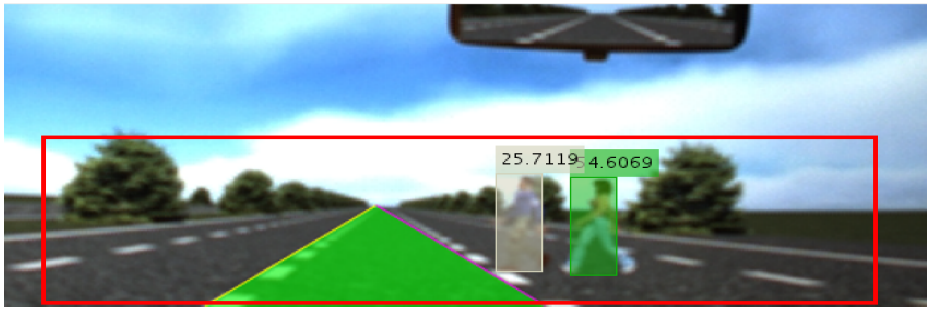


(d)

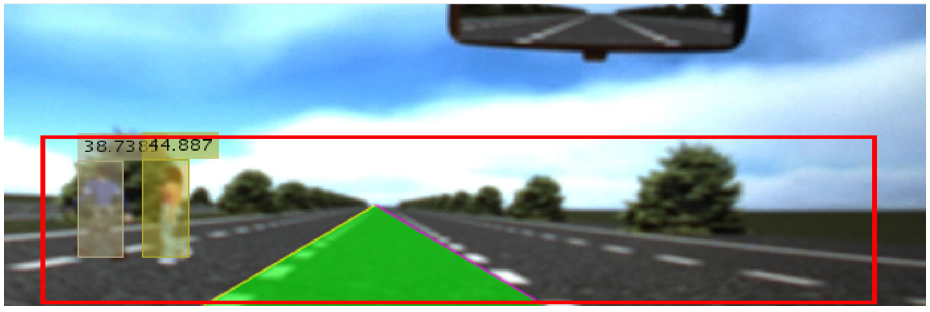
Figure VII.9: Pedestrian detected at different locations and sizes.



(a)



(b)



(c)

Figure VII.10: Multiple pedestrians detected at different locations.

VII.2 Vehicle behavior models used by the CAS

The collision avoidance system (CAS) calculates the vehicle actions based on the external risk analysis. In the CAS, the ego vehicle is treated as an autonomous vehicle surrounded by other human-driven vehicles. The CAS is based on our's previous work [46] with two general tasks shown in Figure VII.11. In the first task, the vehicle's future state is predicted. Similar to [177], the proposed CAS relies on the intention of the human-driven vehicles to predict their trajectories in the near future (5 seconds ahead; and two consecutive predictions are done in 0.5 seconds apart). For that reason, behavior models are utilized. These models store the position transitions of both human-driven and autonomous vehicles with respect to the selected intention/action. The intentions of other vehicles are inferred by the intention recognition described in Chapter IV. Both the Human-driven vehicle Dynamic Model (HDM) and Autonomous vehicle Dynamic Model (ADM) have the same structure which is a Gaussian process model [127]. In the second task, a control algorithm is used to determine the optimized action that the co-pilot needs to take to have safe and smooth driving.

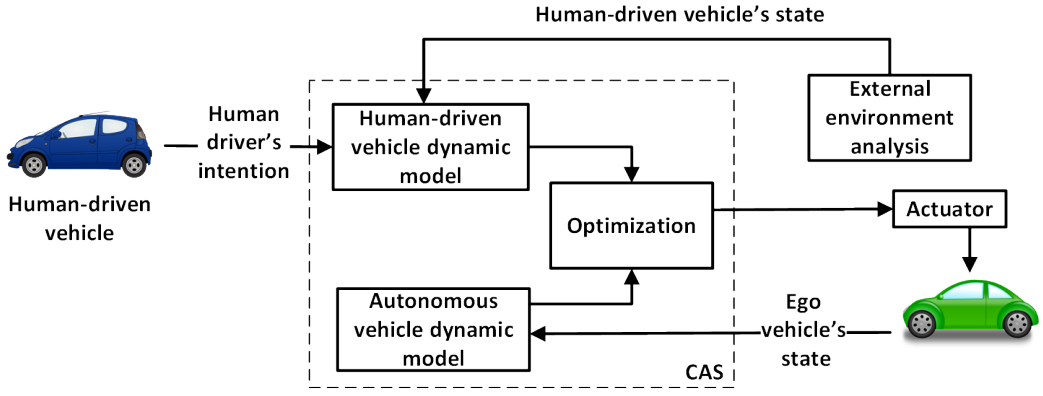


Figure VII.11: General overview of the proposed CAS system.

The following sub-sections detail how behavior is represented. First, the terms

and assumptions that are used in the formulation of the CAS are presented. Second, the HDM and the ADM are discussed. Next, both behavior models are presented in the Gaussian Models and trained by the Gaussian Process.

VII.2.1 Term definition and assumptions

The behavior models are utilized to store the information about the transitions of vehicles from one location to another in time. To explicitly describe this transition, we have to know the physical state of the vehicle. However, it is impossible to keep track of many parameters of the vehicle at the same time and learn their effect on the transition itself. Therefore, the Markov assumption, which only considers important parameters seems to be a reasonable assumption. Hence, it is assumed that at any certain time, all information regarding the physical state of the vehicle is completed and enough to predict the next state. To reproduce the motion of a vehicle, we must obtain its previous location on the road, steering wheel angle, orientation, velocity, and acceleration of the vehicle. By acquiring all of this data, the new state of the vehicle can be predicted after some finite time, assuming that the acceleration and steering wheel have not been changed in this time.

However, the use of this data would make the model very complex. So, to make it simple, this work only considers the location (X, Y) and velocity V as a state of the vehicle, with an assumption that all other readings are approximately small enough or negligible. For example, the vehicle is assumed to be located in the location (x, y) with velocity v at time t, then its state is defined as $s(x, y, v, t)$. The behavior model stores the transition to the next state $s'(x', y', v', t + 1)$. Note that the longitudinal position y and the lateral position x of the vehicle are chosen as the location parameters.

The probabilistic transition from one state to another stores the probability distribution of the next location of the vehicle over the whole state space if the initial state is given. The transition probability may be unique for each input vector, which

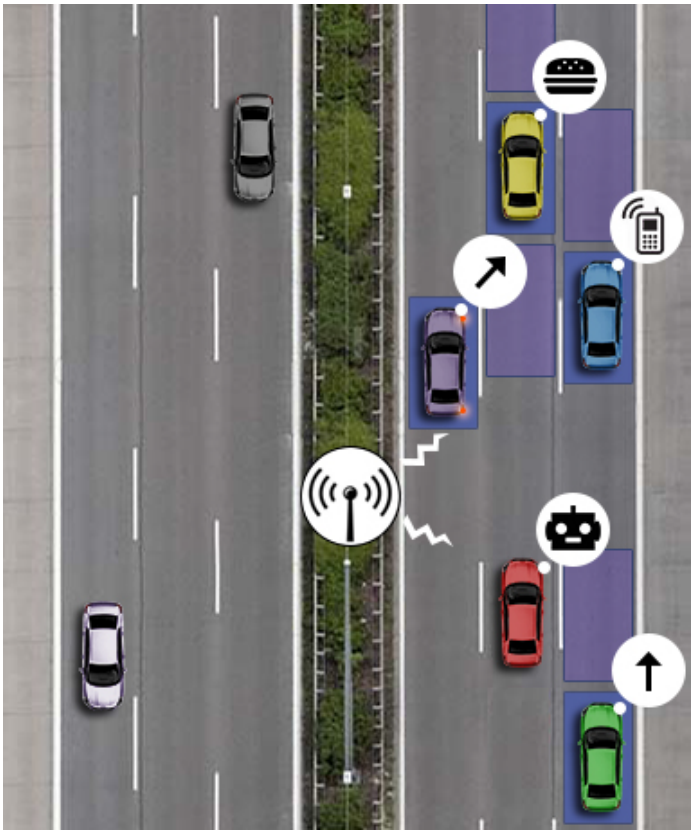


Figure VII.12: The prediction of future occupied locations is made based on the shared intention.

is given by the tuple (s, a). The action a might be given by any intentional and unintentional policy-action from the action set A.

VII.2.2 Human-driven vehicle Dynamic Models (HDM)

In this sub-section, the relationship between human behaviors and maneuvers is discussed. In his paper, Ortiz defined a driver's behaviors to be the set of actions caused by the aim of the person [128]. This can be interpreted that the human driver's behaviors can be effectively separated from his/her intentions and resultant actions. Let us consider the Human-driven vehicle Dynamic Model (HDM) first.

Normally, human drivers use visual signals to notify others about the intention which they are going to perform. Such knowledge is important to plan a trajectory

with respect to the future changes in the environment and to increase the safety as well as making the driving comfort. However, when using only visible communication, it is hard to share intentions between drivers within a group of vehicles. Today, the advanced ITS provides us a framework which supports communication between vehicles such as Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), or through radio transmitters equipped in vehicles. Therefore, we can share the intentions between drivers and thus estimate the possible future locations of all surrounding vehicles as in the example shown in Figure VII.12. To obtain the driver's intention, the intention recognition system presented in Chapter IV can be utilized.

VII.2.3 Autonomous (co-pilot) vehicle Dynamic Models (ADMs)

Similar to an HDM, an ADM stores the transitions of the autonomous vehicle with respect to the selected action. The intelligent vehicle utilizes low-level control signals to control the vehicle such as steering, gas, and brake, but they are not sufficiently effective for high-level decision making. For this reason, the driving task has been decomposed into two levels. At the low level, a control algorithm allows the vehicle to follow the lane and keep the chosen speed. This task can be easily done by using proportional-derivative PD controller minimizing the difference between the target values and the actual lateral location and velocity of the vehicle. At a higher level, a decision-making algorithm chooses an action from the set of actions available to the vehicle such as changing lane and speed. In the CAS, the algorithm may know that the vehicle should change lane, but it may not know the way it happens, how long it takes, or the interim states of the vehicle between the actual state and the target state. To learn these rules, the decision-making algorithm requires a probabilistic transition model connecting the initial state with the desired state via all temporal states.

It is found out that the ADM model may be learned as effective as the HDM

model. Since the action spaces are different, it should be trained separately from the HDM, but the internal structures of both models are absolutely the same which are the Gaussian models and learned by the Gaussian Process.

VII.2.4 Gaussian models

Figure VII.13 shows an example of a Gaussian distribution of traveling between the two points. There is an infinite number of routes that can be built from the point (x, y) to the point (x', y'). However, it can be assumed that the routes between these two points will be a Gaussian distribution against the single most probable route between these two points. This assumption is based on the assumption that the road path is free between two points, and there is no impossible locations on the road at any limited time.

To store the transitions, it might be possible to find a separate Gaussian distribution over X and Y separately. However, this probability distribution will be symmetric against the axes while the actual trajectory of changing lane is diagonal. For this reason, Vectored Gaussian Processes (VGPs) are used. The VGPs store the covariance between X and Y states which may make the trajectory diagonal.

VII.2.5 Gaussian Process

The Gaussian Process (GP) is a supervised learning method widely used to learn a relation between input/output pairs from a training dataset in such a way to be able to predict the output when given system inputs. Generally, there are two types of supervised learning methods which use a parametric or a non-parametric model. While the parametric model considers that the nature of the function is known with pre-specified complexity, the non-parametric model is used when the nature of the function is unknown. In the CAS, the dataset is represented by the transition from one point to another, mainly due to the intent of the human performing this transition.

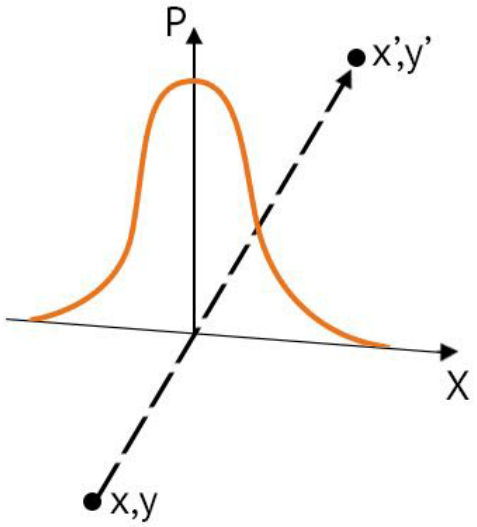


Figure VII.13: An example of Gaussian distribution when traveling between two points.

The GP is a Bayesian nonparametric method operating in Reproducing Kernel Hilbert Space widely used for signal estimation in control systems [178] and can be written as:

$$f(j) \sim GP(m(j), k(j, j')) \tag{VII.2}$$

where $m(j)$ is the mean of a given dataset $D = \{j_1, j_2, ...j_N\}$; and $k(j, j')$ is the covariance kernel used for approximating the covariance of the dataset $j \in D$ and other values j'. The Radial Basis Function kernel (RBF) is utilized for $k(j, j')$ as follows:

$$k(j, j') = exp\left(-\frac{||j - j'||^2}{2\sigma^2}\right) \tag{VII.3}$$

Since the choice of the trajectory depends on the human driver, it is therefore natural to use the RBF kernel shown in Equation (VII.3). This kernel function allows to exponentially weigh the error during the training.

In order to budget the number of kernels, Csato's sparsification method [179] is employed. This method implements an upper bound on the cardinality of the basis vector and allocates RBFs to reduce the regression error. If the novelty of

information for the new incoming data exceeds the threshold, the basis vector set is updated. Otherwise, only the weights and covariance are updated.

$$\triangle v(t) \sim GP_i(v_0, t) \tag{VII.4}$$

$$v(t) = v_0 + \triangle v(t) \tag{VII.5}$$

$$(\triangle x(t), \triangle y(t)) \sim GP_i(x_0, y_0, v(t), t) \tag{VII.6}$$

$$x(t) = x_0 + \triangle x(t) \tag{VII.7}$$

$$y(t) = y_0 + \triangle y(t) \tag{VII.8}$$

In the proposed system, five independent GPs were built - one for each intention. Each GP has two stages. Equation (VII.4) shows the first stage in which the change in the velocity $\triangle v$ at each time for that particular intention i is estimated by the GP. In the second stage as shown in Equation (VII.6), the system predicts the change in the location $(\triangle x, \triangle y)$ when GP is given time t, velocity $v(t)$ and the vehicle's current location (x_0, y_0). These GP estimators were used to develop HDM/ADM blocks as discussed in greater detail in [180].

VII.3 Control algorithm of the CAS

In the real world, it is not easy to exactly predict the intention of a driver. For example, under a specific situation, the predicted probability of the keep-current-state intention is 90%. However, there are still small probabilities of other intentions that could occur such as 4% for changing lane left, 2% for changing lane right, 1% for decreasing speed, and 3% for increasing speed. So, implementing the collision avoidance system with only one maximum probability intention may not address all the possibilities which can be covered by a probability distribution of intentions shown in Figure VII.14. This section discusses the implementation of the probability distribution of intentions on the CAS.

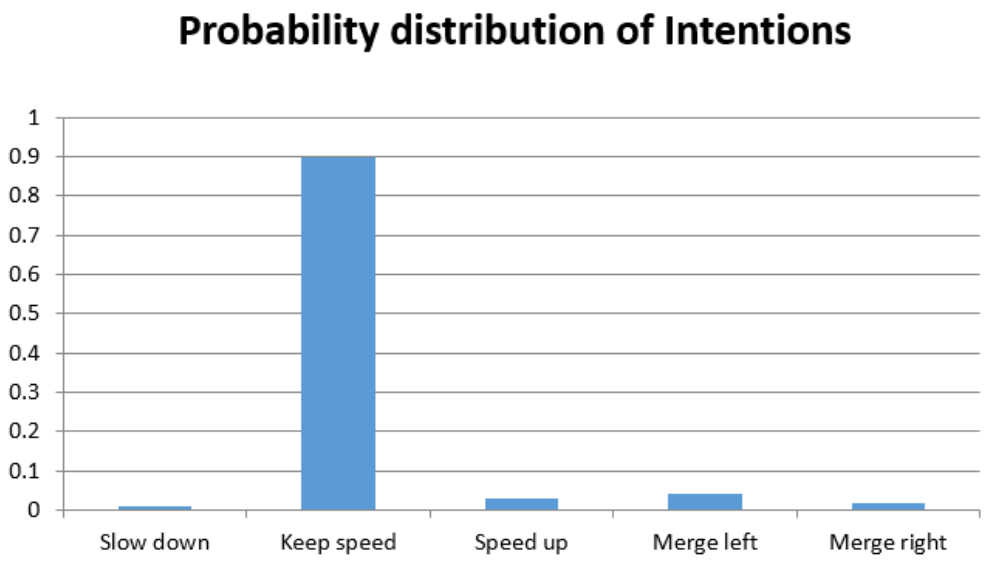


Figure VII.14: An example of the probability distribution of intentions.

In this dissertation, the ego vehicle is assumed to be controlled by the CAS when it is in the autonomous driving mode. It takes action based on the probability distribution of intentions from other human-driven vehicles. The CAS algorithm was developed originally in MATLAB with trajectory prediction (5 seconds ahead) based on one single intention from the human driver. The new goal is to improve the algorithm with the probability distribution of five intentions. So, five trajectories must be predicted at each time step. If doing this task sequentially, the computation time would be increased. Hence, five trajectories must be predicted in parallel with the help of the C++ multiple thread functionality which is not available in MATLAB. Figure VII.15 illustrates the overall system's flowchart. On its main thread, the C++ program makes a connection through UDP to the Carnetsoft simulator which can send out information of all vehicles on road.

The driver intention recognition is also done in the main thread with the probability distribution of five intentions. The C++ program then creates five threads which would call the MATLAB function to predict trajectories of five intentions. Each

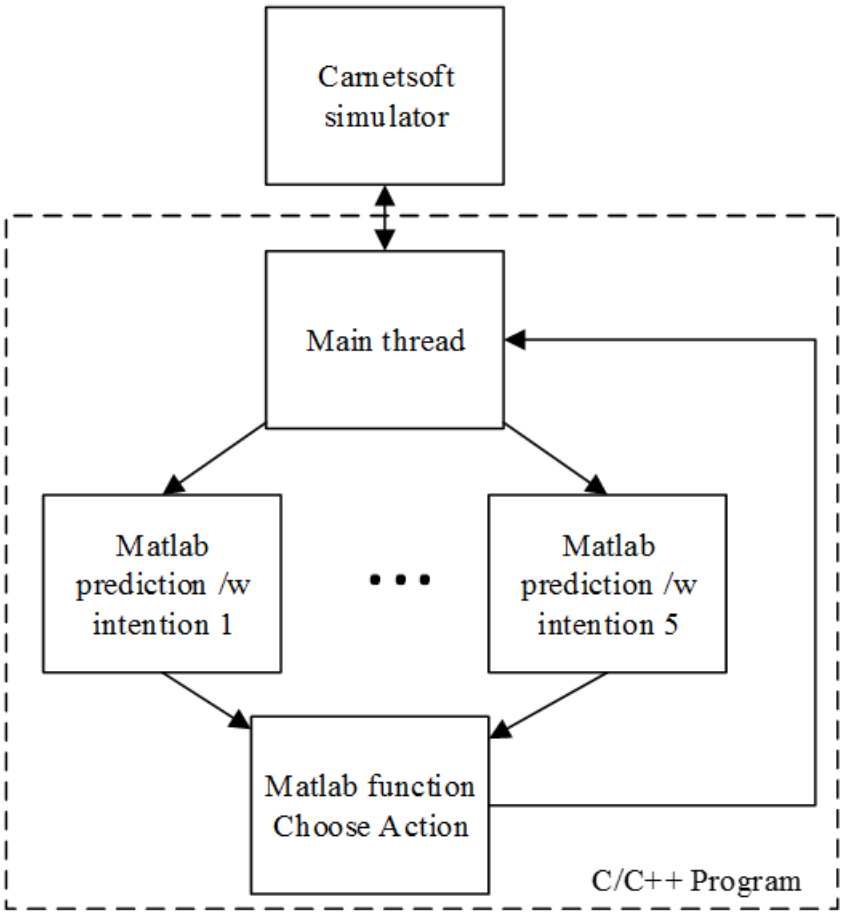


Figure VII.15: Flowchart of implementing the probability distribution of intentions with C++ and MATLAB.

thread would take information from the human-driven vehicle and one intention as inputs. Then the output is generated as a predicted trajectory (T) which includes an array of the mean (x_{mean}, y_{mean}) and the variance (x_{var}, y_{var}) in 5 seconds with 0.5 second time-steps.

The ego vehicle is named as c_1, and the nearby vehicles are named as $c_j, j = 2, ..., n$. A collision between vehicles happens when the ego vehicle c_1 and one or more vehicles reach the same location at the same time. The probability of the ego vehicle c_1 being in state $s \in S(x, y, t)$ is defined as $P_{c_1}(s)$. The probability of vehicle c_2 being in the state s is defined as $P_{c_2}(s)$. Then, the probability of collision is a probability when

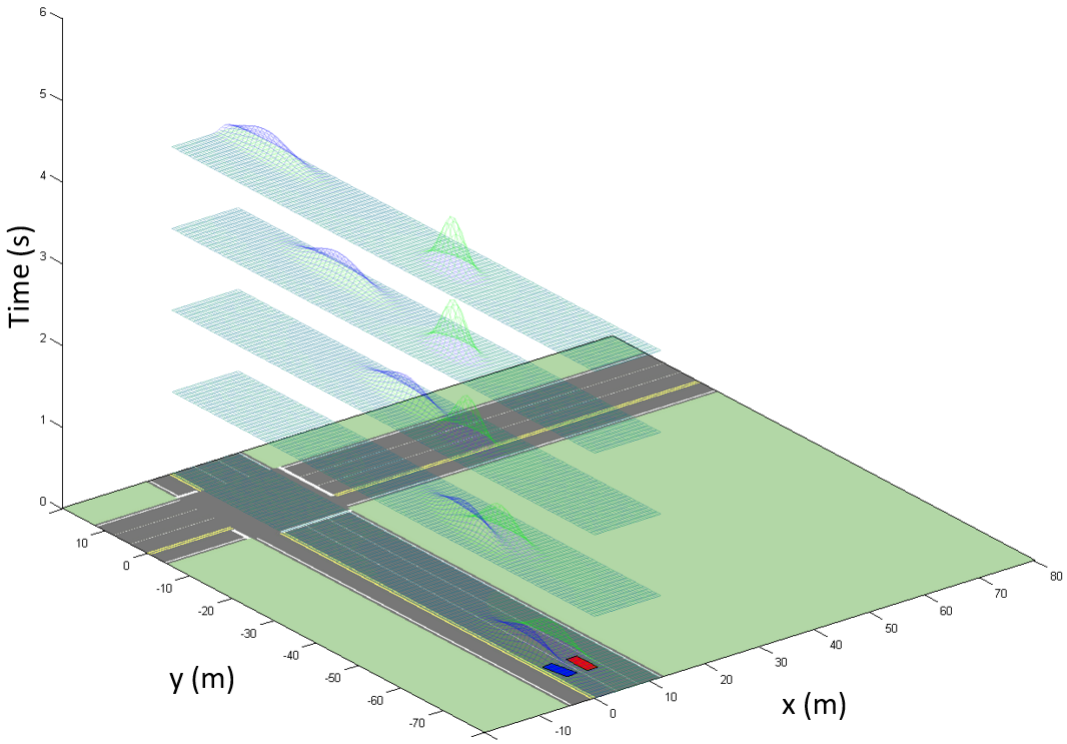


Figure VII.16: Predictions in 5 seconds ahead of the distributions of possible locations of the ego vehicle (red) and another vehicle (blue). The scenario is shown when the human driver intends to merge right.

both vehicles are in the same state:

$$\begin{aligned} P(\text{collision of } c_1 \text{with } c_2 \text{ in } s) &= P(c_1 = s, c_2 = s) \\ &= P_{c_1}(s) P_{c_2}(s) \end{aligned} \tag{VII.9}$$

For a nearby vehicle c_j, its probability in the state $s \in S(x, y, t)$ with each intention $i = 1, ..., 5$ is estimated as $P_{c_j i}(s)$. Then, the total probability of c_j in the state s with five intentions is:

$$P_{c_j}(s) = P(c_j = s) = \sum_{i=1}^{5} K_i P_{c_j i}(s) \tag{VII.10}$$

where K_i is the probability of intention i. We also need to predict the trajectory and the probability of the ego vehicle c_1. Figure VII.16 shows the distributions of possible

locations of the ego vehicle (c_1) and another vehicle (c_2) in an example scenario. Then, the total probability of collision between them becomes:

$$\begin{aligned} &P(\text{collision of } c_1 \text{with } c_2 \text{ in } s) \\ &= \sum_{S} P((c_1 = s) \text{ AND } (c_2 = s)) \\ &= \sum_{0}^{t_{max}} \sum_{y_{min}}^{y_{max}} \sum_{x_{min}}^{x_{max}} (P_{c_1}(x, y, t) P_{c_2}(x, y, t)) \end{aligned} \tag{VII.11}$$

where t_{max} is the time horizon; x_{min}, x_{max}, y_{min} and y_{max} determine the road space.

In general, the collision happens when the ego vehicle (c_1) is in the same location (x, y) with other vehicles $(c_j, j = 2, ..., n)$ at the same time t. According to this assumption, the total probability of collision between the ego vehicle and other nearby ones is defined as:

$$\begin{aligned} &P(\text{collision of } c_1 \text{ with other vehicles in } s) \\ &= \sum_{S} P\left((c_1 = s) \text{ AND } (c_2 = s \text{ OR } ... \text{ OR } c_n = s)\right) \\ &= \sum_{t_0}^{t_{max}} \sum_{y_{min}}^{y_{max}} \sum_{x_{min}}^{x_{max}} \left(P_{c_1}(x, y, t) \sum_{j=2}^{n} P_{c_j}(x, y, t) \right) \end{aligned} \tag{VII.12}$$

Then, the cost J for each action $a \in A = \{1, ..., 5\}$ corresponding to {keeping the vehicle's current state, changing lane left, changing lane right, speeding up, slowing down} can be calculated as follows.

$$J = P(\text{collision}, a) + \text{Cost}(a) + \text{Penalty}(v') + \text{Penalty}(x', y') \tag{VII.13}$$

where Cost(a) is the cost of the action a according to the rank of preferences (less annoying actions have less cost), Penalty(v') is a penalty for driving with the speed different from the speed limit defined on the digital map, Penalty(x', y') is a penalty

for being off-road to motivate the vehicle to follow the road.

$$\text{Cost}(a) = \begin{cases} C_{a_1} & \text{if } a = 1 \\ C_{a_2} & \text{if } a = 2 \\ \vdots & \\ C_{a_n} & \text{if } a = n \end{cases} \tag{VII.14}$$

$$\text{Penalty}(v') = ||V_{desired} - v'||P_v \tag{VII.15}$$

$$\text{Penalty}(x', y') = \begin{cases} P_{out} & \text{if } x', y' \notin \text{Road} \\ 0 & \text{if } x', y' \in \text{Road} \end{cases} \tag{VII.16}$$

where C_{a_i}, P_v and P_{out} are manually defined penalty coefficients for the constrained optimization problem. They are selected so that the values of Cost(a), Penalty(v'), and Penalty(x', y') are in the range of [0,1]. Generating the optimal action to avoid collision is a necessary process in a typical CAS [181]. Unlike [181] which utilizes the "minimal future distance" (MFD), the proposed optimization process relies on estimating the least cost, J^*.

$$J^* = \underset{a \in A}{\text{argmin}}\, J \tag{VII.17}$$

The action/intention that generates the least cost is the best action/intention to be chosen by the co-pilot. A control signal corresponding to this action is also generated. Then the co-pilot's intention along with its control signal and the probability of collision are sent to the collaborative driving block to generate the suitable control for the ego vehicle.

VII.4 Evaluation

In this dissertation, the human driver's intentions and their probability distribution are predicted as presented in Chapter IV. Ten driving tests from five subjects

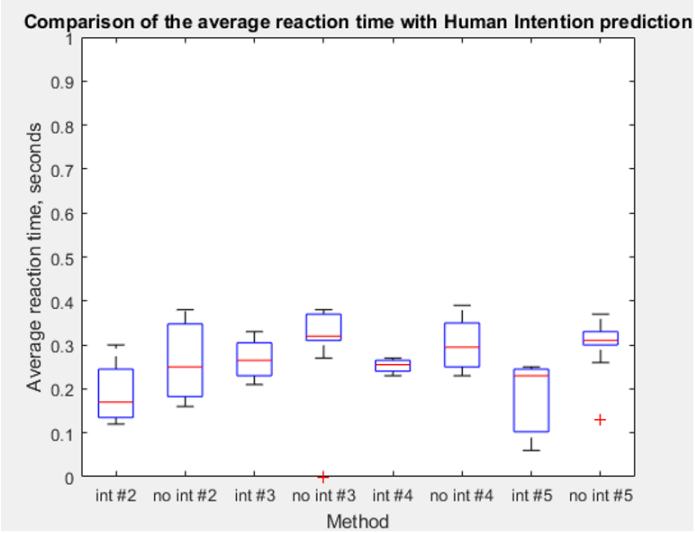


Figure VII.17: Reaction times of the ego vehicle with human intentions. int #2: changing lane left, #3: changing lane right, #4: speeding up, #5: slowing down. no int: always using intention #1, do not consider other intentions.

(five minutes for each subjects) were conducted to evaluate the performance of the CAS. The performance of the system using a single intention only was evaluated first by calculating the reaction time of the autonomous vehicle. This is the time from when the human driver in another vehicle change his intention until when the ego vehicle changes its behavior. Figure VII.17 illustrates the average reaction time of both using and not using intentions. The intention #1 (keeping the vehicle's current state) is the default intention. So, the reaction time of this intention is not estimated. In this experiment, "no int" means that the driver's current intention is unknown. The results show that using predicted human intentions can let the ego vehicle have faster reactions to avoid collisions than not using predicted intentions.

The performance of the CAS using the probability distribution was also evaluated via the ego vehicle's reaction time shown in Figure VII.18. In this experiment, "no int" means that the driver's probability distribution of intentions is unknown but the prior knowledge of the probability distribution of intentions is given. This information can be collected by recording the behavior of a random traffic in a certain location on

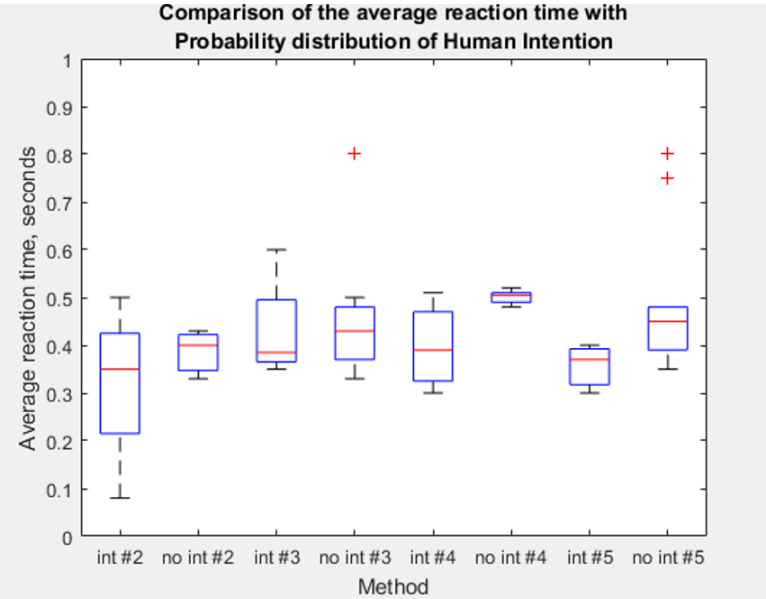


Figure VII.18: Reaction times of the ego vehicle with the probability distribution of human intentions. int #2: changing lane left, #3: changing lane right, #4: speeding up, #5: slowing down.

the road (a bride, a tree, etc.) for a long time. The results from Figure VII.17 and VII.18 show that the average reaction time of using the probability distribution of intentions (≈ 0.4 seconds) is slower than using only one intention (≈ 0.25 seconds). This is true because of the longer computation time of five different trajectories at the same time. Even with parallel computing, the computation is still slower than using only one intention because the computer resource must be shared between the five threads. Moreover, the communications between C++ and MATLAB applications also take longer than those between MATLAB applications only. However, the delay in reaction time of this method is not significant, and it can help the ego vehicle cover all the possibilities that the human-driven vehicle can take to make a suitable action. In addition, using the multi-thread method is faster than sequentially using a single thread of five intentions (with an average reaction time of ≈ 0.8 seconds). So, the computation time is sacrificed to gain an advantage which covers all intention possibilities. However, it is also proved that despite the delay, the collision avoidance

algorithm can react quickly enough for the ego vehicle to avoid collisions.

VII.5 Summary

This chapter presents a collision avoidance system (CAS) which utilizes the information from the video-based external risk assessment. The external risk analysis includes a lane departure warning system and a pedestrian detection system. Based on the external risk analysis and the probability distributions of intentions from other drivers, the CAS can determine the best action the ego vehicle should take in the near future.

CHAPTER VIII

COLLABORATIVE DRIVING FRAMEWORK

A collaborative driving framework, which considers conditions such as the ego vehicle's status, the driver's drowsiness, and his/her drowsiness intensity, is proposed. The framework takes inputs from two agents which are the human driver and the autonomous driver (co-pilot) whose action is generated by the CAS. This chapter presents the ideas behind the collaborative driving framework, the experimental results, and discussions regarding the overall system's behaviors.

VIII.1 Collaborative driving framework

The control from the human driver and the control from the co-pilot are integrated to generate the final control of the vehicle. The human driver's control is obtained from the vehicle control system through the steering wheel, and the gas and brake pedals while the co-pilot's control is generated by the CAS algorithm, which includes the preferred lateral position and velocity. In order to fuse the two control inputs, the CDS converts the human driver's control input to the preferred lateral position and velocity. The overall collaborative driving framework is shown in Figure VIII.1.

In the first stage, the prediction is estimated by the posterior probability $p(a|O_h, O_c)$. O_h is the ego vehicle's control data (steering angle, gas and brake pedal positions). O_c is the vehicle state data (positions, heading angles, and speeds) of all vehicles. Equation (VIII.1) presents the posterior probability calculated using the Bayesian theorem.

$$p(a|O_h, O_c) = \frac{p(O_h, O_c|a)p(a)}{p(O_h, O_c)} \tag{VIII.1}$$

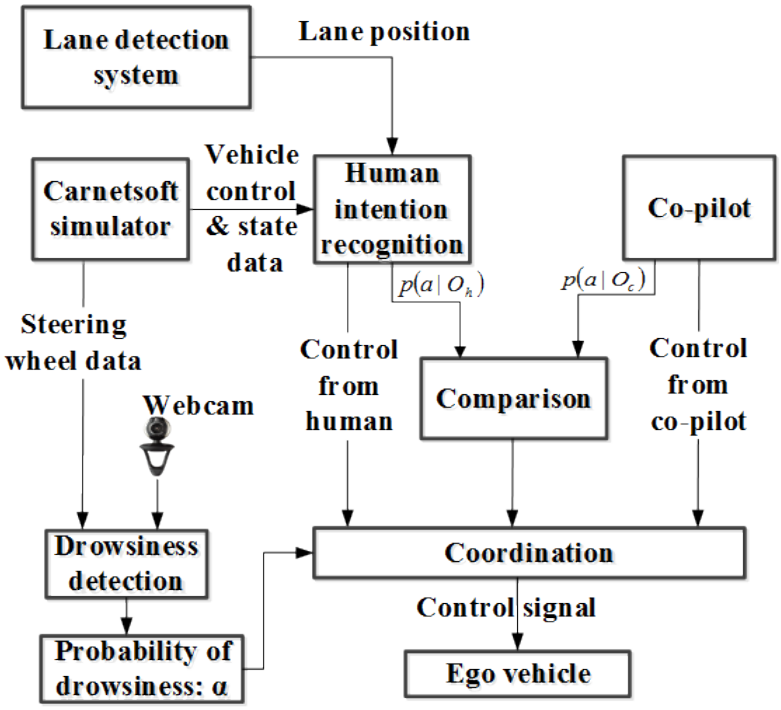


Figure VIII.1: The collaborative control framework.

Here, O_h and O_c are assumed to be independent. Then the posterior probability can be inferred by fusing the likelihood functions as in Equation (VIII.2).

$$p(a|O_h, O_c) = \frac{p(O_h|a)p(O_c|a)p(a)}{p(O_h)p(O_c)} \tag{VIII.2}$$

The human driver's intention/action can be inferred by using the Hidden Markov Model classification as described in Chapter IV, from which the likelihood of each action a from the human, $p(a|O_h)$ can be obtained. On the other hand, the CAS algorithm can estimate the likelihood of all five actions, $p(a|O_c)$. Then, the difference of entropy, d_E, between the human driver's intended actions and the co-pilot's actions is calculated. The better the prediction is, the larger the difference d_E is.

$$\begin{aligned} d_E = &\sum_a p(a|O_h) \log p(a|O_h) \\ &- \sum_a p(a|O_h, O_c) \log \left[p(a|O_h, O_c)\right] \end{aligned} \tag{VIII.3}$$

The ego vehicle's final control U is calculated as follows:

$$U = u(d_E)F + (1 - u(d_E))G \tag{VIII.4}$$

$$F = U_h(1 - \beta) + \beta U_{c_t} \tag{VIII.5}$$

$$\begin{aligned} G =& u(\alpha - \alpha_T)[(1 - u(d_l - D))U_{c_t} + u(d_l - D)U_{c_s}] \\ &+ [1 - u(\alpha - \alpha_T)] \\ &\times [u(\lambda - \lambda_T)U_{c_t} + (1 - u(\lambda - \lambda_T))U_h] \end{aligned} \tag{VIII.6}$$

$$u(n) = \begin{cases} 0, \, n < 0 \\ 1, \, n \geq 0 \end{cases} \tag{VIII.7}$$

$$\beta = \frac{1}{1 + e^{-c_1 \times d_E + c_2}} \tag{VIII.8}$$

U_h is the control input from the human driver; U_{c_t} and U_{c_s} are the co-pilot's control inputs to drive the ego vehicle temporarily and stop the ego vehicle at the roadside, respectively; $u(n)$ is a unit step function. The reason that the unit step function is chosen to represent the control formulation is to have a complete separation of controls under each scenario. α is the human driver's drowsiness probability; α_T is a threshold to determine if the driver is drowsy; λ is the probability of collision estimated by Equation (VII.12); λ_T is the threshold to determine if the ego vehicle is in a dangerous state; d_l is the drowsiness level, which is accumulated when the driver gets drowsy multiple times; D is the drowsiness level threshold to determine how deep the driver is getting into the drowsy state; β is a sigmoid function with the range of [0,1]. If β approaches 1, the co-pilot has more control of the ego vehicle. If β approaches 0, the control input relies mostly on the human driver. c_1 and c_2 are the characteristic coefficients of the sigmoid functions.

In the proposed collaborative driving framework, two predicted actions (from the human driver and the co-pilot) are compared. If they agree with each other ($d_E \geq 0$),

it means that the human driver is controlling the ego vehicle safely no matter his/her status. So, the final control is F as in Equation (VIII.5) to assist the human to drive safely. For example, if the co-pilot's best action is changing to the left lane and the driver also wants to do that task, the control input is a fusion of controls from both the driver and the co-pilot. Then the ego vehicle can change to the left lane more quickly. When d_E is a small positive value, and the two predicted actions agree but not much, the fused control input relies more on the human driver. When d_E is a large positive value, the co-pilot and the human driver have a strong agreement. Therefore the fused control input relies more on the co-pilot, and the human does not need to apply much control on the ego vehicle in this situation.

If the driver's and the co-pilot's predicted actions do not agree with each other ($d_E < 0$), it means that the driver does not drive safely according to the co-pilot. In this case, the final control is G. As shown in Equation (VIII.6), G is determined based on three conditions which are the driver's probability of drowsiness, drowsiness level, and the probability of collision between the ego vehicle and other nearby vehicles. If the driver is not drowsy ($\alpha < \alpha_T$) and the probability of collision is low ($\lambda < \lambda_T$), he/she is free to manually drive the ego vehicle with the control input U_h. If the driver is not drowsy, but the probability of collision with another vehicle is high ($\lambda \geq \lambda_T$), it is possible that he/she is distracted and not focusing on the driving task. At this moment, the co-pilot should take over the control to drive the ego vehicle temporarily with the control input U_{c_t}.

To ensure safe driving, it is not wise to let the driver control the vehicle if he/she keeps getting drowsy. Therefore, some thresholds of drowsiness level are set. When the driver's drowsiness level (d_l) increases beyond the threshold (D), it should be safer for him/her to stop the vehicle. This is also the time for the co-pilot to kick in and control the vehicle to stop at a safe place at the roadside while triggering an alarm to wake up the driver. After the vehicle stops, it is better to turn off the engine to

avoid any unnecessary control from the driver while he/she is drowsy. After waking up, the human driver has to press the deactivate button (or ignition key) to control the vehicle again.

In the fusion part, Equation (VIII.5), the control of the ego vehicle relies on how much the human and the co-pilot agree, which is estimated by the value β in Equation (VIII.8). Because β is a smooth sigmoid function, the transition between the controls of the human and the co-pilot system will be smooth. Moreover, in the driving simulator, the ego vehicle can be programmed to run by specifying its desired velocity and lateral position which is described in Chapter III. Hence, when the newly-desired velocity and lateral position are estimated by the system, the driving simulator will gradually control the ego vehicle from the old status to the new one. Therefore, the transition will be smooth and safe.

VIII.2 Experiments & results

Figure I.1 presents the external risk analysis, which is done through a camera placed inside the ego vehicle. However, due to the lack of hardware facilities and the safety concerns associated with real road tests, the research is conducted on the assisted-driving testbed, which is useful for preliminary studies. In the assisted-driving testbed, it is only possible to detect and localize vehicles which are in front of the ego vehicle. It is impossible to use a camera to detect vehicles behind the ego vehicle. Therefore, their precise locations obtained from the driving simulator are utilized for the collaborative driving framework. In the real-world implementation, LiDAR sensors, sonar sensors, or a set of multiple cameras can be used to detect and localize moving objects surrounding the ego vehicle.

The collaborative driving algorithm was evaluated on the testbed in different scenarios with the following parameters: $\alpha_T = 0.7, \lambda_T = 0.7, D = 2, c_1 = 10$ and $c_2 = 5$. The control input as in Figure VIII.2 and VIII.3 has five possible values as:

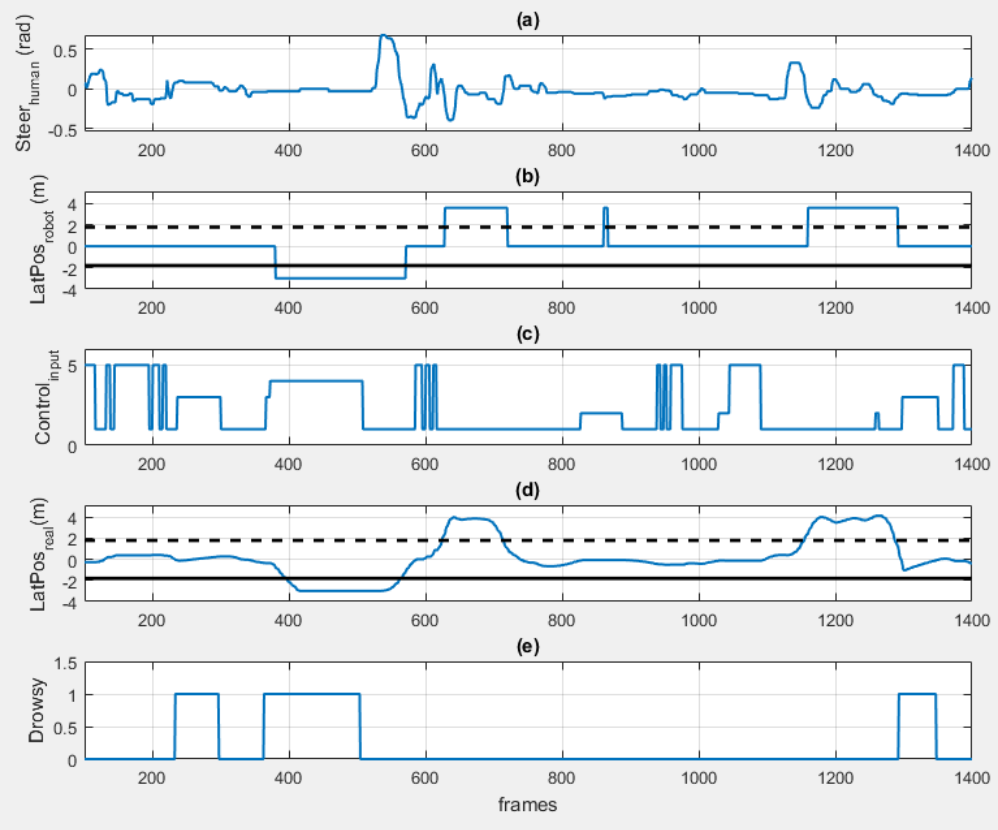


Figure VIII.2: Results of the ego vehicle's lateral position. (a): the steering angle controlled by the driver; (b): predicted lateral position from the co-pilot; (c): control input; (d): real lateral position; (e): the driver's state: 0 = non-drowsy, 1 = drowsy.

- 1: U_h. If the prediction is not good ($d_E < 0$), the human driver is non-drowsy ($\alpha < \alpha_T$), and the ego vehicle is not in any dangerous situation ($\lambda < \lambda_T$), then he/she can control the ego vehicle at this moment.

- 2: U_{c_t}. If the prediction is not good ($d_E < 0$), and the human driver is non-drowsy ($\alpha < \alpha_T$), but the ego vehicle is in a dangerous situation ($\lambda \geq \lambda_T$), then the co-pilot controls the ego vehicle temporarily.

- 3: U_{c_t}. If the prediction is not good ($d_E < 0$), and the human driver is drowsy ($\alpha \geq \alpha_T$) for the first time ($d_l < D$), then the co-pilot controls the ego vehicle temporarily.

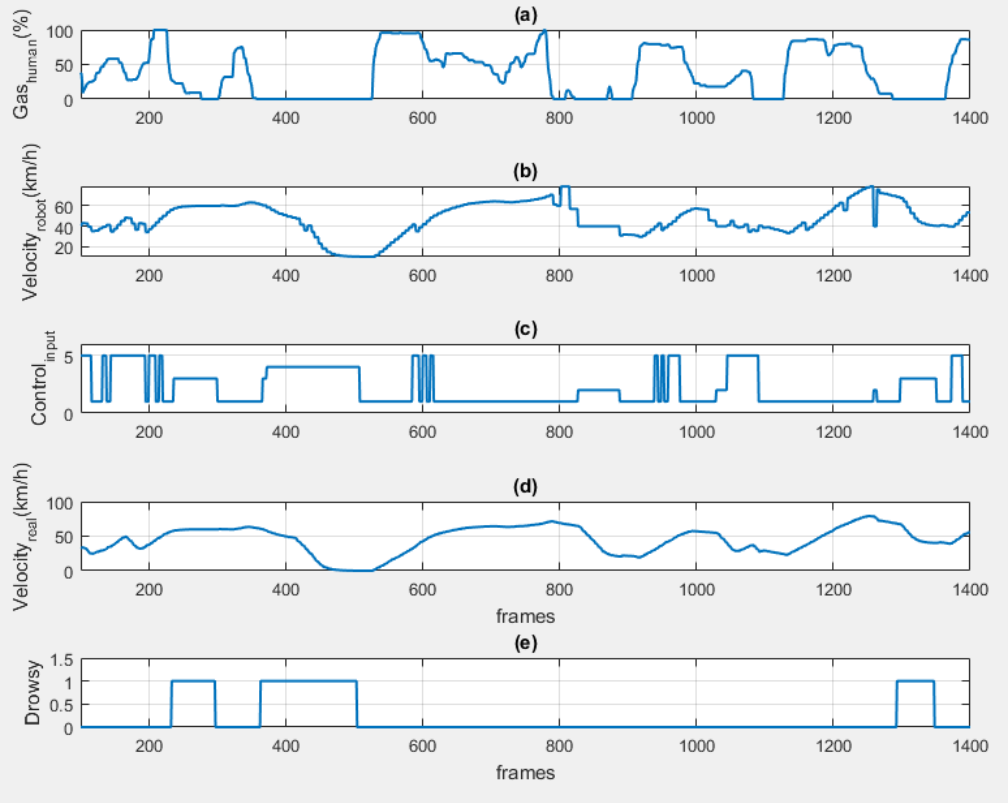


Figure VIII.3: Results of the ego vehicle's velocity. (a): the gas value controlled by the driver; (b): predicted velocity from the co-pilot; (c): control input; (d): the vehicle's real velocity; (e): the driver's state: 0 = non-drowsy, 1 = drowsy.

- 4: U_{c_s}. If the prediction is not good ($d_E < 0$), and the human driver is drowsy ($\alpha \geq \alpha_T$) for the second time ($d_l \geq D$), then the co-pilot controls the ego vehicle to stop at the roadside.

- 5: F. If the prediction is good ($d_E \geq 0$), the co-pilot can assist the human driver by fusing the control inputs from both the human driver and the co-pilot.

Note that the control inputs 2 and 3 are similar in two different conditions. That is to let us keep track of the conditions and display the results easily.

Figure VIII.2 and VIII.3 present the results with the lateral position input and velocity input, respectively of an experiment as shown in Figure III.11. In Figure VIII.2 (b) and (d), the black dashed lines indicate the road centerlines, while the

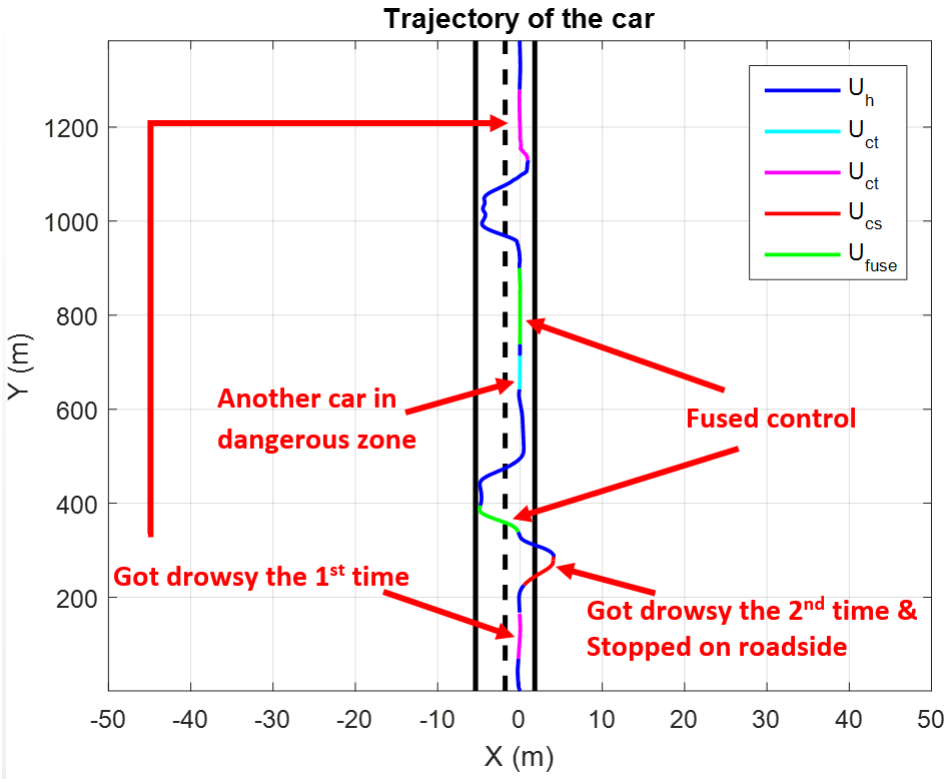


Figure VIII.4: The trajectory of the ego vehicle with different control inputs.

black solid lines represent the solid road lines beyond which is the roadside. From the beginning, the co-pilot assists the human driver to increase the speed. In Figure VIII.3(a) and (b), from the frame 220, the co-pilot and the human driver do not agree with each other while the driver becomes drowsy and the ego vehicle is in the safe zone, so the co-pilot temporarily controls the ego vehicle. In the meantime, an alarm signal is triggered to wake up the driver. From the frame 300 - 370, the human driver wakes up so he can manually control the ego vehicle. After that, he becomes drowsy again, and the drowsiness level is leveraged. Therefore the co-pilot takes over the control and drives the ego vehicle to stop at the roadside (the frame 400 - 550 in Figure VIII.2(d) and VIII.3(d)). When the driver wakes up, he presses the deactivation button to manually drive the ego vehicle back to the road. At the frame 850, the co-pilot and the human driver do not agree (Figure VIII.3(a) and (b)), and the ego vehicle is too close to a leading vehicle while the human driver is not drowsy.

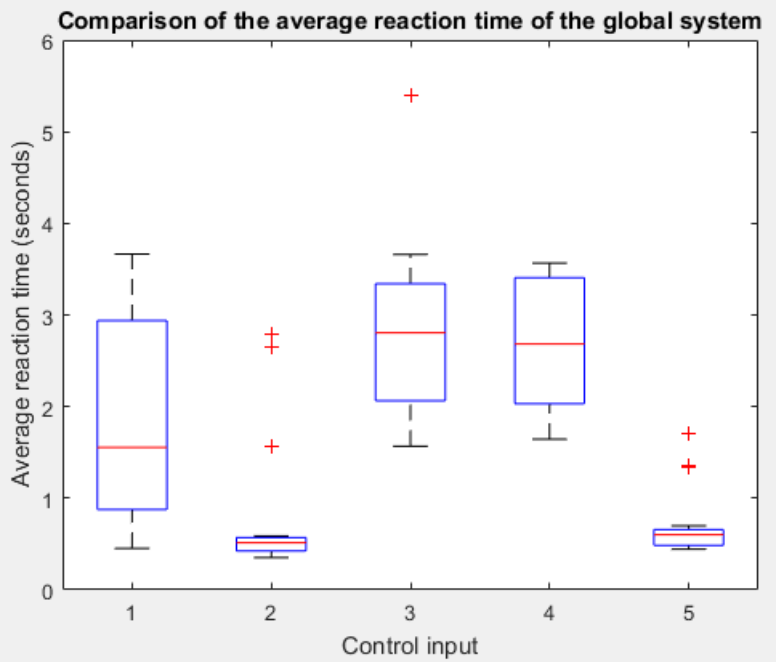


Figure VIII.5: Average reaction times of the collaborative driving system with different control inputs.

At this moment, the human driver is deduced to be distracted, and the co-pilot kicks in to control the ego vehicle temporarily and triggers a voice alarm to alert the driver. Overall, the system responded as expected with smooth transitions in trajectory as shown in Figure VIII.4.

In addition, the reaction time of the collaborative driving system with different control inputs is evaluated. Ten-minute driving tests from three subjects were conducted. Because different control inputs correspond to different states of the driver and the ego vehicle, the reaction time of each control input is defined as follows.

- The reaction time of the control input 2 is the elapsed time from the moment the driver takes an action differently from that of the co-pilot, and the ego vehicle is in a dangerous situation, until it takes action.

- The reaction time of the control inputs 3 and 4 is the elapsed time from the moment the driver becomes drowsy and takes an action differently from that of

the co-pilot, until the ego vehicle reacts.

- The reaction time of the control input 5 is the elapsed time from the moment the predicted actions of the human driver and the co-pilot agree with each other, until the ego vehicle takes action.

- The reaction time of the control input 1 is the elapsed time from the moment the states of the control inputs 2, 3, 4, and 5 change back to that of the control input 1, until the ego vehicle reacts.

The results are shown in Figure VIII.5. It can be observed that the average reaction times of the system with control inputs 3 and 4 are approximately 2.8 seconds. This is reasonable because these controls rely on the drowsiness detection, which needs about 2.5 seconds to detect the status change from non-drowsy to drowsy (Figure V.9). However, the control inputs 2 and 5 do not require the drowsiness detection. Therefore, the system just needs around 0.6 seconds to react to these changes of these control inputs. On the other hand, the reaction time of the control input 1 has a large variation because it is estimated based on the changes from all other control inputs. Its average time is approximately 1.5 seconds.

VIII.3 Discussions

It is worthwhile to discuss the impact of the detection failures to the system performance. The proposed collaborative driving framework has two conditional layers to determine the vehicle's final control. The first layer is to check if the control inputs from the driver and the co-pilot agree. Then, the second layer checks the driver's state and the vehicle's state (the possibility of collision). In the first layer, if the two control inputs agree, the driver's control input is safe no matter the driver's state. Therefore, false detections in this situation are not needed to be worried about. If the two control inputs do not agree. In this case, there might be false positives and

false negatives in the driver drowsiness detection. For false negatives (i.e., the driver is actually drowsy, but he/she is detected to be non-drowsy), the system lets the driver control the vehicle by himself/herself. However, when the vehicle is in a dangerous state, e.g., the vehicle is about to collide with other vehicles due to the driver's drowsiness, the co-pilot takes over the control to ensure the traffic safety. For false positives (i.e., the driver is actually non-drowsy, but he/she is detected to be drowsy), the co-pilot takes over the control and triggers a voice alarm to alert the driver about the switching in the driving mode. In this case, the input from the human driver is ignored. The duration of such a case is usually short. However, if this lasts for an extended time, a push-button on the dashboard can be implemented for the driver to override the co-pilot.

The proposed collaborative driving framework will also bring some changes to the normal driving experience. When the human driver applies a control input (e.g., steering, accelerating, or braking), it is analyzed by the collaborative driving system to determine if it is safe or not. If the system predicts that it is safe, this control input is sent to the vehicle's engine. If the system asserts that it is dangerous (when the probability of collision is high or when the driver is drowsy), the co-pilot's control input is used to control the vehicle. Therefore, the system can ensure the traffic safety. There are three scenarios when the co-pilot involves in the vehicle's responses:

- When the driver takes an action which is different from that of the co-pilot, and the ego vehicle is in a dangerous situation.
- When the driver becomes drowsy and takes an action which is different from that of the co-pilot.
- When the predicted actions of the human driver and the co-pilot agree with each other.

In the first two cases, the driver may observe that the co-pilot's involvement is un-

predictable. However, the involvement is necessary to make sure that the ego vehicle does not collide with other vehicles. So the unpredictable responses of the vehicle in these two cases are necessary because they ensure the traffic safety. In the last case, the final control input is a fusion of the driver's control input and the co-pilot's control input. Hence, there might be some unpredicted responses. To address this issue, a voice interface is implemented to alert the driver. In this way, the driver can be prepared when the system starts the collaboration, which can improve the driver's satisfaction.

VIII.4 Summary

This chapter presents the development of a collaborative driving framework which aims to improve transportation safety. The framework integrates two agents: an automated co-pilot and a human driver. The co-pilot calculates control actions to avoid collision with nearby vehicles and runs in parallel with the human driver. The co-pilot's decisions can be fused with the human driver's decisions or override his/her decisions depending on the driver's status and environmental conditions. This framework improves the safety of driving in different dangerous scenarios and ensures seamless transitions between driving modes.

CHAPTER IX

CONCLUSIONS & FUTURE WORKS

This chapter concludes this dissertation and discusses some works that can be improved in the future.

IX.1 Conclusions

This dissertation proposes a driver assistance system to improve safety for future ITS systems. The system analyzes both internal and external risk factors. The internal risk analysis includes the driver's intention, drowsiness, and distraction. The intention detection system can detect intentions including slowing down, speeding up, changing lane left, changing lane right and keeping the vehicle's current state. In addition, a driver drowsiness detection system is presented. It takes two input types: steering wheel angles and driver's facial images. Also, a system which could detect distracted-driving behaviors and alert the human driver is proposed. In terms of the external risk analysis, the driver assistance system is developed with both lane and pedestrian detection capabilities. The system can detect different line types (broken or solid) and trigger a warning signal during departure. On the other hand, the pedestrian detection system works correctly within 30 m. The CAS system utilizes the external risk analysis and the probability distribution of intentions from other human-driven vehicles to generate the best action that the ego vehicle should take. A collaborative driving framework which integrates two agents: a co-pilot and a human driver is proposed. The co-pilot always runs the CAS in parallel with the human driver and gives assistance if necessary. This framework enables safe driving in a

variety of dangerous scenarios while ensuring seamless transitions between driving modes.

To validate the proposed framework, an ITS testbed which includes both physical and simulated setups is developed. The testbed can perform remote configuration including remote data selection and remote trajectory design so that the client can design his/her own experiments. Moreover, the ITS testbed can help the client perform driver analysis such as intention prediction and drowsiness detection to enhance safety. In addition, an assisted-driving testbed is developed by adding two embedded boards to the simulated testbed. The assisted-driving testbed is utilized for the real-time distraction detection system.

The results of this dissertation have been partially published in [39, 165, 180, 182–188].

IX.2 Future works

IX.2.1 Internal risk analysis

Currently, the collaborative driving framework only utilizes the driver's intention and drowsiness from the internal risk analysis. However, as mentioned above, driver drowsiness is just one type of driver distraction. Therefore, it is believed that the distraction detection system could replace the drowsiness detection system in the collaborative driving framework. There are many ways to enhance the distraction detection system. Currently, the similarities in postures of different behaviors result in incorrect classifications. The behaviors that have more misclassifications are "drinking", "hair and makeup" and "texting on the phone - left". To improve the accuracy, the proposed approach can be combined with the face-based approach as in [71, 72] or replace the last fully-connected layers by the traditional classifiers such as SVM or HMM to classify the distracted behaviors [73]. In addition, the distraction detection system can be enhanced by using other sensing modalities. For example, microphones

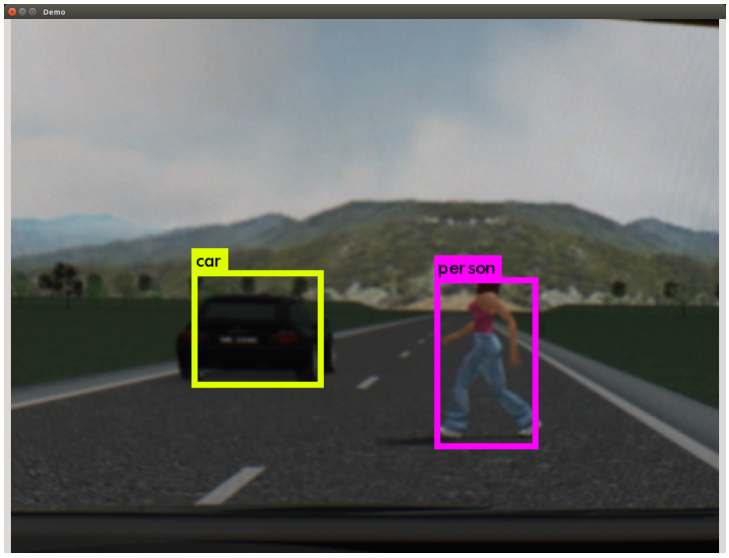


Figure IX.1: Pedestrian and vehicle detection in YOLO.

can be utilized to capture the sounds and voices in the car, which offer rich clues to detect different distracted driving behaviors. Moreover, the distraction detection system performance could be enhanced by using the image from the driver's face as an additional input because it provides useful information about his/her status.

IX.2.2 External risk analysis

The video-based external risk assessment can detect lanes and pedestrians. Vehicle detection, which can be considered as a special type of object detection, is the next step to complete the external risk analysis. Similar to pedestrian detection, there are several challenges in video-based vehicle detection. There are a wide variety of vehicle types and colors, and other vehicles can approach from any direction, which may make their shapes change especially when the ego vehicle is moving as well.

With the rapid evolution of parallel computing and deep learning, robust tools which detect vehicles are available. Figure IX.1 shows the pedestrian and vehicle detections with YOLO, an open source deep CNN object detection algorithm [189]. The program runs on an NVIDIA Jetson TK1 board with a frame rate of 10-12 fps and 1-2 second delay. The TX1 board has only 4 GB of RAM and a Quad ARM A57

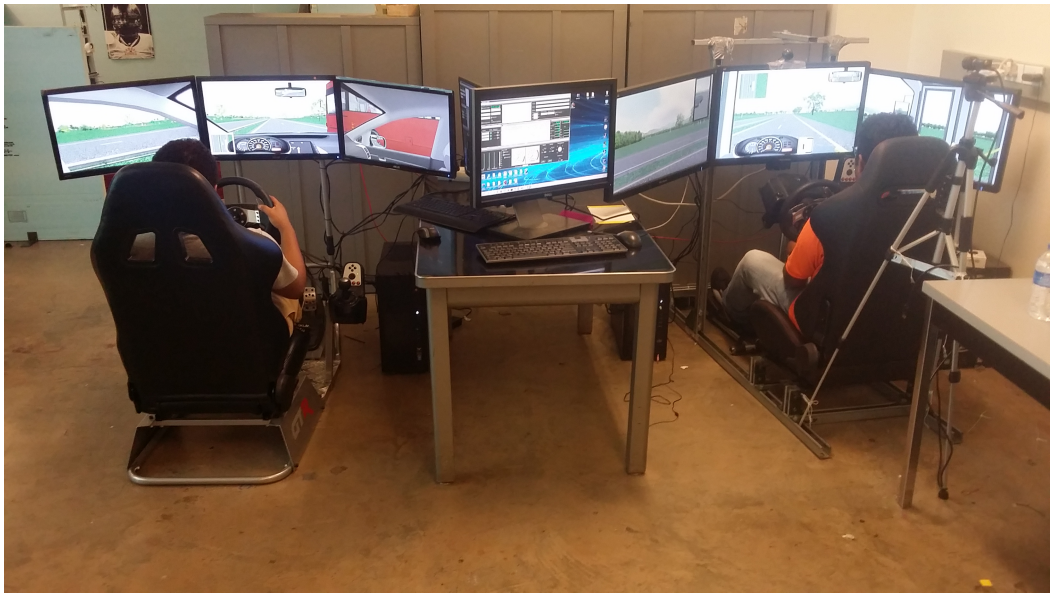


Figure IX.2: The two simulator setup.

L2 CPU. If YOLO is implemented on a more powerful computer, the performance would be improved.

IX.2.3 Vehicle-to-Vehicle (V2V) communication

In order to create more realistic scenarios, an additional driving simulator that is similar to the current one can be utilized. Both simulators can share data through the V2V communication. Another human driver can control this new simulator and create any desired interference. The setup of two simulators for V2V communication is presented in Figure IX.2. In the proposed system, the left-side vehicle is programmed to create some behaviors such as blocking, pushing from behind and changing lane to disrupt the ego vehicle's driving.

A more complicated scenario can also be assumed as shown in Figure IX.3. The driver in the right simulator is driving a bus which is stopping on the road while the driver in the left simulator is driving a car whose view is blocked by the bus. The bus can detect a pedestrian crossing the road in front of it. In this situation, a system that shares the pedestrian's current location and displays it on the simulator screen

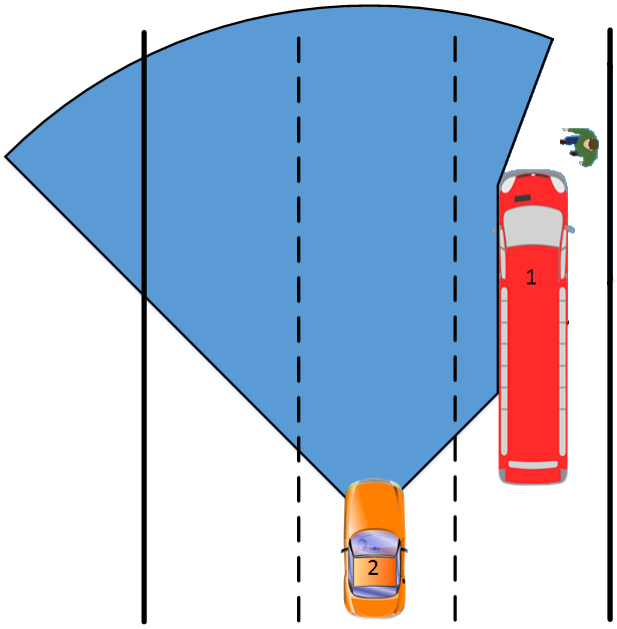


Figure IX.3: A view-blocking scenario.

of the car can be implemented. This can alert the driver to take precautions. The pedestrian detection algorithm can detect the distance between the pedestrian and the bus and use the camera calibration process to map 2D-pixel coordinates into 3D real-world coordinates. Then, 3D real-world coordinates can be shifted from the bus to the car based on the relative positions of the two vehicles. Then, they are mapped back to 2D-pixel coordinates and displayed on the screen of the car.

One of the most common errors in automated systems is that sensors fail to work. We can also consider scenarios where the sensor has noisy output. By considering the noisy environment, the system can be made more realistic and robust. Moreover, it can also request the information from surrounding vehicles to obtain more accurate decisions. For implementing this approach in the proposed system, noisy sensor readings such as distance to other vehicles, pedestrians or lanes can be assumed by adding a Gaussian noise to the original readings. Then, sensor fusion algorithms such as Kalman filter or particle filter can be used to fuse the sensor readings from both vehicles to have better estimations and predictions of the vehicle state.

IX.2.4 Driver-vehicle communication protocol

A protocol for communication between the co-pilot and the human driver can be developed. The system detects the driver's intention which is considered to have the highest priority. Then the co-pilot compares the human driver's intention with its generated best action. If they do not match, the co-pilot can inform the driver whether he/she wants to wait or process his/her current intention. For example, the driver wants to change lane but the co-pilot detects it is not safe to do that. It would inform the driver that it is not safe and recommend the human driver to wait for a certain time to change lanes.

IX.2.5 Security

Security is also a challenging problem. When the vehicles share their information with each other, hackers can remotely hack the vehicles, steal the driver's information or even paralyze the control. This can cause serious traffic accidents because the human driver totally loses the control of the vehicle. Therefore developing a protocol that can secure the self-driving vehicles from cyber-attacks is also a potential research area.

BIBLIOGRAPHY

[1] Association for Safe International Road Travel. Annual global road crash statistics. Online : http://asirt.org/initiatives/informing-road-users/road-safety-facts/road-crash-statistics.

[2] Center of Disease Control and Prevention. Drowsy Driving: Asleep at the Wheel. *Center of Disease Control and Prevention*, November 2015. Online: http://www.cdc.gov/features/dsdrowsydriving/.

[3] US Department of Transportation - National Highway Traffic Safety Administratio. Distracted driving.

[4] Google. Autonomous driving. *The New York Times, Oct 10*, 2010. Online: http://www.nytimes.com/2010/10/10/science/10google.html.

[5] Mike Murphy. Google's self-driving cars are now on the street of California. *Quartz*, June 2015. Online: http://www.wired.com/2015/10/tesla-self-driving-over-air-update-live/.

[6] Todd C. Frankel. What it feels like to drive a Tesla on autopilot. *The Washington Post*, February 2016. Online: https://www.washingtonpost.com/news/the-switch/wp/2016/02/01/what-it-feels-like-to-drive-a-tesla-on-autopilot/.

[7] Marco della Cava. Audi's screaming-hot race car drives itself. *USA Today*, 2015. Online: http://www.usatoday.com/story/tech/2015/07/11/audi-rs7-drives-itself-around-racetrack/29965135/.

[8] Sarah Sloat. BMW, Intel, Mobileye Link Up in Self-Driving Tech Alliance. *The Wall Street Journal*, 2016. Online: http://www.wsj.com/articles/bmw-intel-mobileye-link-up-in-self-driving-tech-alliance-1467379145.

[9] Graham Rapier. Ford is going all-in on self-driving cars. *Business Insider*, 2015. Online: http://www.businessinsider.com/ford-joins-rivals-developing-self-driving-cars-2015-6.

[10] Riley McDermid. Concord snags another self-driving carmaker, as Honda receives DMV permit. *San Francisco Business Times*, 2015. Online: http://www.bizjournals.com/sanfrancisco/morning_call/2015/09/honda-apple-to-test-selfdriving-car-concord-aapl.html.

[11] Nick Gibbs. Jaguar Land Rover to test over 100 autonomous cars in Britain by 2020. *Reuters*, 2016. Online: http://www.reuters.com/article/us-jaguarlandrover-driverless-idUSKCN0ZS2V2.

[12] Alex Davies. The Mercedes Robo-car that made we want to stop driving. *Wired*, 2015. Online: https://www.wired.com/2015/03/mercedes-benz-f-015-autonomous-car/.

[13] Kukil Bora. Nissan Gets Into Self-Driving Mode, Says Its Autonomous Cars Will Be Ready By 2020. *International Business Times*, May 2015. Online: http://www.ibtimes.com/nissan-gets-self-driving-mode-says-its-autonomous-cars-will-be-ready-2020-1926447.

[14] Alex Davies. Toyota finally gets serious about self-driving cars. *Wired*, 2015. Online: https://www.wired.com/2015/09/toyota-enters-self-driving-car-race/.

[15] Alistair Charlton. Meet Volkswagen's electric, self-driving Golf of 2025. *International Business Times*, 2016. Online: http://www.ibtimes.co.uk/meet-volkswagens-electric-self-driving-golf-2025-1583988.

[16] Elisabeth Behrmann. Volvo plans self-driving car by 2021 to challenge BMW. *Autonomtive News*, 2016. Online: http://www.autonews.com/article/20160722/COPY01/307229945/volvo-plans-self-driving-car-by-2021-to-challenge-bmw.

[17] Jimi Beckwith. Apple driverless car project killed by lack of direction, say employees. *Autocar*, 2017. Online: https://www.theverge.com/2018/3/20/17143708/apple-self-driving-google-waymo-california-uber.

[18] Napier Lopez. Nvidia announces a 'supercomputer' GPU and deep-learning platform for self-driving cars. *The Next Web*, 2015. Online: http://thenextweb.com/gadgets/2016/01/05/nvidia-announces-a-supercomputer-gpu-and-deep-learning-platform-for-self-driving-cars/.

[19] Molly. Steel Citys New Wheels. *Uber Newsroom*, May 2016. Online: https://newsroom.uber.com/us-pennsylvania/new-wheels/.

[20] Arjun Kharpal. Microsoft, Volvo strike deal to make driverless cars. *CNBC*, November 2015. Online: http://www.cnbc.com/2015/11/20/microsoft-volvo-strike-deal-to-make-driverless-cars.html.

[21] Warrendale Pa. Nhtsa adopts sae international standard defining autonomous vehicles; sae releases new version for free - j3016 states and defines six levels of automation in on-road motor vehicles.

[22] Danielle Muoio. These 19 companies are racing to build self-driving cars in the next 5 years. *Business Insider*, 2017.

[23] Ellie Zolfaghrifard. When Tesla's autopilot goes wrong: Owners post terrifying footage showing what happens when brand new autonomous driving software fails. *Dailymail.com*, October 2015. Online: http://www.dailymail.co.uk/sciencetech/article-3281562/Tesla-autopilot-fail-videos-emerge-Terrifying-footage-shows-happens-autonomous-driving-goes-wrong.html.

[24] Neal Boudette Bill Vlasic. Self-Driving Tesla Was Involved in Fatal Crash, U.S. Says. *The New York Time*, 2016. Online: https://www.washingtonpost.com/news/the-switch/wp/2016/02/01/what-it-feels-like-to-drive-a-tesla-on-autopilot/.

[25] National Highway Traffic Safety Administration. Vehicle to vehicle communication. http://www.safercar.gov/v2v/index.html.

[26] PR Newswire. National survey reveals why drivers dont use turn signals. http://www.prnewswire.com/newsreleases/national-survey-reveals-why-drivers-dont-use-turn-signals-55355907.

[27] X. Huang. Driver lane change intention recognition by using entropy-based fusion techniques and support vector machine learning strategy. Master's thesis, Northeastern University, Massachusetts, December 2012.

[28] Matthias J Henning, Olivier Georgeon, Tony Wynn, and Josef F Krems. Modelling driver behaviour in order to infer the intention to change lanes. In *Proceedings of European Conference on Human Centred Design for Intelligent Transport Systems*, page 113, 2008.

[29] A. Doshi and M. Trivedi. A comparative exploration of eye gaze and head motion cues for lane change intent prediction. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 49–54, June 2008.

[30] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb 1989.

[31] US Department of Transportation - National Highway Traffic Safety Administration. Distraction.gov, official us government website for distracted driving. May 2010. Online : `http://www.distraction.gov`.

[32] US Department of Transportation - National Highway Traffic Safety Administratio. Distractions: In and out of the vehicle.

[33] Marcel Adam Just, Timothy A Keller, and Jacquelyn Cynkar. A decrease in brain activation associated with driving when listening to someone speak. *Brain research*, 1205:70–80, 2008.

[34] Esurance. 3 types of distracted driving. 2016. Online: `https://www.esurance.com/info/car/3-types-of-distracted-driving`.

[35] Karen Bowman. Did You Know That Fatigue Causes Distracted Driving? A.K.A. Drowsy Driving. *Dropitanddrive*, November 2016. Online: `http://dropitanddrive.com/2016/11/14/did-you-know-that-fatigue-causes-distracted-driving-a-k-a-drowsy-driving/`.

[36] US Department of Transportation - National Highway Traffic Safety Administration. Traffic safety facts 2013 data - pedestrians, Feb 2015. `http://www-nrd.nhtsa.dot.gov/Pubs/812124.pdf`.

[37] US Department of Health and Human Services. Wisqars (web-based injury statistics query and reporting system), Feb 2015. `http://www.cdc.gov/injury/wisqars`.

[38] Laurie F Beck, Ann M Dellinger, and Mary E O'neil. Motor vehicle crash injury rates by mode of travel, united states: using exposure-based methods to quantify differences. *American Journal of Epidemiology*, 166(2):212–218, 2007.

[39] Weihua Sheng, Yongsheng Ou, Duke Tran, Eyosiyas Tadesse, Meiqin Liu, and Gangfeng Yan. An integrated manual and autonomous driving framework based on driver drowsiness detection. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 4376–4381. IEEE, 2013.

[40] David Exner, Erich Bruns, Daniel Kurz, Anselm Grundhöfer, and Oliver Bimber. Fast and robust camshift tracking. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, pages 9–16. IEEE, 2010.

[41] Davis King. Dlib C++ library. October 2015. Online: `http://dlib.net/intro.html/`.

[42] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[43] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[44] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[45] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[46] Denis Osipychev. Collision avoidance for autonomous cars based on human intention. Master's thesis, Oklahoma State University, 2015.

[47] Georges S Aoude and Jonathan P How. Using support vector machines and bayesian filtering for classifying agent intentions at road intersections, 2009.

[48] J.C. McCall, D.P. Wipf, M.M. Trivedi, and B.D. Rao. Lane change intent analysis using robust operators and sparse bayesian learning. *Intelligent Transportation Systems, IEEE Transactions on*, 8(3):431–440, Sept 2007.

[49] A. Liu and A. Pentland. Towards real-time recognition of driver intentions. In *Intelligent Transportation System, 1997. ITSC '97., IEEE Conference on*, pages 236–241, Nov 1997.

[50] Jin, L. and Hou, H.and Jiang Y. Driver intention recognition based on continuous hidden markov model, 2011.

[51] Jieyun Ding, Ruina Dang, Jianqiang Wang, and Keqiang Li. Driver intention recognition method based on comprehensive lane-change environment assessment. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pages 214–220, June 2014.

[52] Li Shiwu, Wang Linhong, Yang Zhifa, Ji Bingkui, Qiao Feiyan, and Yang Zhongkai. An active driver fatigue identification technique using multiple physiological features. In *Mechatronic Science, Electric Engineering and Computer (MEC), 2011 International Conference on*, pages 733–737, Aug 2011.

[53] Saroj KL Lal and Ashley Craig. Driver fatigue: electroencephalography and psychological assessment. *Psychophysiology*, 39(3):313–321, 2002.

[54] Lisheng Jin, Qingning Niu, Haijing Hou, Huacai Xian, Yali Wang, and Dongdong Shi. Driver cognitive distraction detection using driving performance measures. *Discrete Dynamics in Nature and Society*, 2012, 2012.

[55] Thomas A Ranney. Driver distraction: A review of the current state-of-knowledge. Technical report, 2008.

[56] Pooneh. R. Tabrizi and Reza. A. Zoroofi. Drowsiness detection based on brightness and numeral features of eye image, September 2009.

[57] L.M. Bergasa, J. Nuevo, M.A. Sotelo, R. Barea, and M.E. Lopez. Real-time system for monitoring driver vigilance. *Intelligent Transportation Systems, IEEE Transactions on*, 7(1):63–77, March 2006.

[58] Qiang Ji, P. Lan, and C. Looney. A probabilistic framework for modeling and real-time monitoring human fatigue. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 36(5):862–875, Sept 2006.

[59] C. Craye and F. Karray. Multi-distributions particle filter for eye tracking inside a vehicle, 2013.

[60] Qiang Ji, Zhiwei Zhu, and P. Lan. Real-time nonintrusive monitoring and prediction of driver fatigue. *Vehicular Technology, IEEE Transactions on*, 53(4):1052–1068, July 2004.

[61] C. Craye and F. Karray. Driver distraction detection and recognition using rgb-d sensor, 2015.

[62] P. R. Tabrizi and R. A. Zoroofi. Drowsiness detection based on brightness and numeral features of eye image. In *2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 1310–1313, Sept 2009.

[63] Yulan Liang, M.L. Reyes, and J.D. Lee. Real-time detection of driver cognitive distraction using support vector machines. *Intelligent Transportation Systems, IEEE Transactions on*, 8(2):340–350, June 2007.

[64] M. Kutila, M. Jokela, G. Markkula, and M.R. Rue. Driver distraction detection with a camera vision system. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 6, pages VI – 201–VI – 204, Sept 2007.

[65] H. Gu and Q. Ji. Facial event classification with task oriented dynamic bayesian network, 2004.

[66] State Farm Corporate. State farm distracted driver detection.

[67] Samuel Colbran, Kaiqi Cen, and Danni Luo. Classification of driver distraction.

[68] Ofonime Dominic Okon and Li Meng. Detecting distracted driving with deep learning. In *International Conference on Interactive Collaborative Robotics*, pages 170–179. Springer, 2017.

[69] Yehya Abouelnaga, Hesham M Eraqi, and Mohamed N Moustafa. Real-time distracted driver posture classification. *arXiv preprint arXiv:1706.09498*, 2017.

[70] András Lőrincz, Máté Csákvári, Áron Fóthi, Zoltán Ádám Milacski, András Sárkány, and Zoltán Tősér. Cognitive deep machine can train itself. *arXiv preprint arXiv:1612.00745*, 2016.

[71] In-Ho Choi, Sung Kyung Hong, and Yong-Guk Kim. Real-time categorization of driver's gaze zone using the deep learning techniques. In *Big Data and Smart Computing (BigComp), 2016 International Conference on*, pages 143–148. IEEE, 2016.

[72] Marco Venturelli, Guido Borghi, Roberto Vezzani, and Rita Cucchiara. Deep head pose estimation from depth data for in-car automotive applications. *arXiv preprint arXiv:1703.01883*, 2017.

[73] Murtadha D Hssayeni, Sagar Saxena, Raymond Ptucha, and Andreas Savakis. Distracted driver detection: Deep learning vs handcrafted features. *Electronic Imaging*, 2017(10):20–26, 2017.

[74] T. Akerstedt and M. Gillberg. Subjective and objective sleepiness in the active individual. *International Journal of Neuroscience*, 52:29–37, 1990.

[75] Y.Lin, H.Leng, and et al. G.Yang. An intelligent noninvasive sensor for driver pulse wave measurement. *IEEE Sensor Journal*, 7:790–799, 2007.

[76] Chin-Teng Lin, Ruei-Cheng Wu, and Sheng-Fu Liang. Eeg-based drowsiness estimation for safety driving using independent component analysis. *IEEE Transactions on Circuits and Systems*, 52:2726–2738, 2005.

[77] Agustina Garcés Correa, Lorena Orosco, and Eric Laciar. Automatic detection of drowsiness in eeg records based on multimodal analysis. *Medical engineering & physics*, 36(2):244–249, 2014.

[78] Rui Wang, Yang Wang, and Chunheng Luo. Eeg-based real-time drowsiness detection using hilbert-huang transform. In *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2015 7th International Conference on*, volume 1, pages 195–198. IEEE, 2015.

[79] R. Sayed and A. Eskandarian. Unobtrusive drowsiness detection by neural network learning of driver steering. *Proceedings of the Institution of Mechanical Engineers. Part D, Journal of Automobile Engineering*, 215:969–975, 2001.

[80] A. Eskandarian and A. Mortazavi. Evaluation of smart algorithm for commercial vehicle driver drowsiness detection. *Proceedings of the 2007 IEEE Intelligent Vehicles Symposium*, pages 553–559, 2007.

[81] Betsy Thomas and Ashutosh Gupta. Wireless sensor embedded steering wheel for real time monitoring of driver fatigue detection. *Advances in CSEE*, 2012.

[82] Anthony D McDonald, John D Lee, Chris Schwarz, and Timothy L Brown. Steering in a random forest ensemble learning for detecting drowsiness-related lane departures. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 56(5):986–998, 2014.

[83] Samuel Lawoyin, Ding-Yu Fei, and Ou Bai. Accelerometer-based steering-wheel movement monitoring for drowsy-driving detection. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of automobile engineering*, 229(2):163–173, 2015.

[84] Danghui Liu, Peng Sun, YanQing Xiao, and Yunxia Yin. Drowsiness detection based on eyelid movement. In *Education Technology and Computer Science (ETCS), 2010 Second International Workshop on*, volume 2, pages 49–52. IEEE, 2010.

[85] R Andy McKinley, Lindsey K McIntire, Regina Schmidt, Daniel W Repperger, and John A Caldwell. Evaluation of eye metrics as a detector of fatigue. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 53(4):403–414, 2011.

[86] Sheng Tong Lin, Ying Ying Tan, Pei Ying Chua, Lian Kheng Tey, and Chie Hui Ang. Perclos threshold for drowsiness detection during real driving. *Journal of Vision*, 12(9):546–546, 2012.

[87] Shreya P Patel, Bhumika P Patel, Madhu Sharma, Nisha Shukla, and Hinaxi M Patel. Detection of drowsiness and fatigue level of driver. *International Journal for Innovative Research in Science and Technology*, 1(11):133–138, 2015.

[88] Esra Vural, Mujdat Cetin, Aytul Ercil, Gwen Littlewort, Marian Bartlett, and Javier Movellan. Machine learning systems for detecting driver drowsiness. *IN-VEHICLE CORPUS AND SIGNAL PROCESSING FOR DRIVER BEHAV-IORs*, pages 97–110, 2009.

[89] Satori Hachisuka. Human and vehicle-driver drowsiness detection by facial expression. In *Biometrics and Kansei Engineering (ICBAKE), 2013 International Conference on*, pages 320–326. IEEE, 2013.

[90] Taro Nakamura, Akinobu Maejima, and Satoru Morishima. Detection of driver's drowsy facial expression. In *Pattern Recognition (ACPR), 2013 2nd IAPR Asian Conference on*, pages 749–753. IEEE, 2013.

[91] A. Eskandarian and R. A. Sayed. Detecting driver fatigue by monitoring eye and steering activity. In *Proceeding of Annual Intelligent Vehicles Systems Symposium*, 2003.

[92] Azim Eskandarian and Riaz A Sayed. Analysis of driver impairment, fatigue, and drowsiness and an unobtrusive vehicle-based detection scheme. In *Proc. 1st Int. Conf. Traffic Accidents*, pages 35–49, 2005.

[93] Alexander von Reyher, A Joos, and H Winner. A lidar-based approach for near range lane detection. In *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.*, pages 147–152. IEEE, 2005.

[94] Albert S Huang, David Moore, Matthew Antone, Edwin Olson, and Seth Teller. Finding multiple lanes in urban road networks with vision and lidar. *Autonomous Robots*, 26(2-3):103–122, 2009.

[95] Jorge Hernández and Beatriz Marcotegui. Filtering of artifacts and pavement segmentation from mobile lidar data. In *ISPRS Workshop Laserscanning 2009*, 2009.

[96] Soren Kammel and Benjamin Pitzer. Lidar-based lane marker detection and mapping. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 1137–1142. IEEE, 2008.

[97] Lars B Cremean and Richard M Murray. Model-based estimation of off-highway road geometry using single-axis ladar and inertial sensing. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 1661–1666. IEEE, 2006.

[98] Nicolas Soquet, Didier Aubert, and Nicolas Hautiere. Road segmentation supervised by an extended v-disparity algorithm for autonomous navigation. In *2007 IEEE Intelligent Vehicles Symposium*, pages 160–165. IEEE, 2007.

[99] Hendrik Dahlkamp, Adrian Kaehler, David Stavens, Sebastian Thrun, and Gary R Bradski. Self-supervised monocular road detection in desert terrain. In *Robotics: science and systems*, volume 38. Philadelphia, 2006.

[100] Qingji Gao, Qijun Luo, and Sun Moli. Rough set based unstructured road detection through feature learning. In *2007 IEEE International Conference on Automation and Logistics*, pages 101–106. IEEE, 2007.

[101] M Boumediene, A Ouamri, and N Dahnoun. Lane boundary detection and tracking using nnf and hmm approaches. In *2007 IEEE Intelligent Vehicles Symposium*, pages 1107–1111. IEEE, 2007.

[102] Qiang He and Chee-hung Henry Chu. Lane detection and tracking through affine rectification. In *MVA*, pages 536–539, 2007.

[103] Abdulhakam AM Assidiq, Othman O Khalifa, Md Rafiqul Islam, and Sheroz Khan. Real time lane detection for autonomous vehicles. In *Computer and Communication Engineering, 2008. ICCCE 2008. International Conference on*, pages 82–88. IEEE, 2008.

[104] Quoc-Bao Truong and Byung-Ryong Lee. New lane detection algorithm for autonomous vehicles using computer vision. In *Control, Automation and Systems, 2008. ICCAS 2008. International Conference on*, pages 1208–1213. IEEE, 2008.

[105] You Feng, Wang Rong-ben, and Zhang Rong-hui. Research on road recognition algorithm based on structure environment for its. In *2008 ISECS International Colloquium on Computing, Communication, Control, and Management*, volume 1, pages 84–87. IEEE, 2008.

[106] Banggui Zheng, Bingxiang Tian, Jianmin Duan, and Dezhi Gao. Automatic detection technique of preceding lane and vehicle. In *2008 IEEE International Conference on Automation and Logistics*, pages 1370–1375. IEEE, 2008.

[107] Wenhong Zhu, Fuqiang Liu, Zhipeng Li, Xinhong Wang, and Shanshan Zhang. A vision based lane detection and tracking algorithm in automatic drive. In *Computational Intelligence and Industrial Application, 2008. PACIIA'08. Pacific-Asia Workshop on*, volume 1, pages 799–803. IEEE, 2008.

[108] Yifei Wang, Naim Dahnoun, and Alin Achim. A novel lane feature extraction algorithm implemented on the tms320dm6437 dsp platform. In *2009 16th International Conference on Digital Signal Processing*, pages 1–6. IEEE, 2009.

[109] Amnon Shashua, Yoram Gdalyahu, and Gaby Hayun. Pedestrian detection for driving assistance systems: Single-frame classification and system level performance. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 1–6. IEEE, 2004.

[110] Bo Wu and Ram Nevatia. Cluster boosted tree classifier for multi-view, multi-pose object detection. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[111] Christian Wojek and Bernt Schiele. A performance evaluation of single and multi-feature people detection. In *Joint Pattern Recognition Symposium*, pages 82–91. Springer, 2008.

[112] Xiaoyu Wang, Tony X Han, and Shuicheng Yan. An hog-lbp human detector with partial occlusion handling. In *2009 IEEE 12th International Conference on Computer Vision*, pages 32–39. IEEE, 2009.

[113] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):971–987, 2002.

[114] William Robson Schwartz, Aniruddha Kembhavi, David Harwood, and Larry S Davis. Human detection using partial least squares analysis. In *2009 IEEE 12th international conference on computer vision*, pages 24–31. IEEE, 2009.

[115] Piotr Dollár, Zhuowen Tu, Pietro Perona, and Serge Belongie. Integral channel features. 2009.

[116] Piotr Dollár, Serge Belongie, and Pietro Perona. The fastest pedestrian detector in the west. In *BMVC*, volume 2, page 7. Citeseer, 2010.

[117] P. Dollr, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1532–1545, Aug 2014.

[118] Sayanan Sivaraman and Mohan Manubhai Trivedi. Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior

analysis. *IEEE Transactions on Intelligent Transportation Systems*, 14(4):1773–1795, 2013.

[119] Wei Liu, XueZhi Wen, Bobo Duan, Huai Yuan, and Nan Wang. Rear vehicle detection and tracking for lane change assist. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 252–257. IEEE, 2007.

[120] T-HS Li, Shih-Jie Chang, and Yi-Xiang Chen. Implementation of human-like driving skills by autonomous fuzzy behavior control on an fpga-based car-like mobile robot. *IEEE Transactions on Industrial Electronics*, 50(5):867–880, 2003.

[121] Rahul Sukthankar. Raccoon: A real-time autonomous car chaser operating optimally at night. Technical report, DTIC Document, 1992.

[122] Dean A Pomerleau. Alvinn, an autonomous land vehicle in a neural network. Technical report, Carnegie Mellon University, Computer Science Department, 1989.

[123] George W Goodrich. Collision avoidance using optical pattern growth rate, March 24 1981. US Patent 4,257,703.

[124] Michael Maile, Qi Chen, G Brown, and Luca Delgrossi. Intersection collision avoidance: From driver alerts to vehicle control. In *Vehicular Technology Conference (VTC Spring), 2015 IEEE 81st*, pages 1–5. IEEE, 2015.

[125] Felipe Jiménez, José Eugenio Naranjo, and Óscar Gómez. Autonomous collision avoidance system based on accurate knowledge of the vehicle surroundings. *IET intelligent transport systems*, 9(1):105–117, 2014.

[126] Jake K Aggarwal and Michael S Ryoo. Human activity analysis: A review. *ACM Computing Surveys (CSUR)*, 43(3):16, 2011.

[127] Jason Hardy, Frank Havlak, and Mark Campbell. Multiple-step prediction using a two stage gaussian process model. In *American Control Conference (ACC), 2014*, pages 3443–3449. IEEE, 2014.

[128] Michael Garcia Ortiz. Prediction of driver behavior. 2014.

[129] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[130] Thorsten Brandt, Thomas Sattel, and Michael Bohm. Combining haptic human-machine interaction with predictive path planning for lane-keeping and collision avoidance systems. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 582–587. IEEE, 2007.

[131] Selim Temizer, Mykel Kochenderfer, Leslie Kaelbling, Tomas Lozano-Pérez, and James Kuchar. Collision avoidance for unmanned aircraft using markov decision processes. In *AIAA guidance, navigation, and control conference*, page 8040, 2010.

[132] Shankarachary Ragi and Edwin KP Chong. Uav path planning in a dynamic environment via partially observable markov decision process. *IEEE Transactions on Aerospace and Electronic Systems*, 49(4):2397–2412, 2013.

[133] Tirthankar Bandyopadhyay, Chong Zhuang Jie, David Hsu, Marcelo H Ang Jr, Daniela Rus, and Emilio Frazzoli. Intention-aware pedestrian avoidance. In *Experimental Robotics*, pages 963–977. Springer, 2013.

[134] Sebastian Brechtel, Tobias Gindele, and Rüdiger Dillmann. Probabilistic mdp-behavior planning for cars. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pages 1537–1542. IEEE, 2011.

[135] Ha M Do, Craig J Mouser, Ye Gu, Weihua Sheng, Sam Honarvar, and Tingting Chen. An open platform telepresence robot with natural human interface. In *Cyber Technology in Automation, Control and Intelligent Systems (CYBER), 2013 IEEE 3rd Annual International Conference on*, pages 81–86. IEEE, 2013.

[136] Ha Manh Do, Craig Mouser, Meiqin Liu, and Weihua Sheng. Human-robot collaboration in a mobile visual sensor network. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 2203–2208. IEEE, 2014.

[137] Ha M Do, Craig J Mouser, and Weihua Sheng. Building a telepresence robot based on an open-source robot operating system and android. In *Third Conference on Theoretical and Applied Computer Science (TACS 2012)*, 2012.

[138] Ha Manh Do, Weihua Sheng, and Meiqin Liu. Human-assisted sound event recognition for home service robots. *Robotics and Biomimetics*, 3(1):7, June 2016.

[139] Terrence Fong, Charles Thorpe, and Charles Baur. *Collaborative control: A robot-centric model for vehicle teleoperation*, volume 1. Carnegie Mellon University, The Robotics Institute, 2001.

[140] T. Carlson and Y. Demiris. Collaborative control for a robotic wheelchair: Evaluation of performance, attention, and workload. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(3):876–888, June 2012.

[141] Douglas G Macharet and Dinei A Florencio. A collaborative control system for telepresence robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5105–5111. IEEE, 2012.

[142] Junqing Wei and John M Dolan. A multi-level collaborative driving framework for autonomous vehicles. In *The 18th IEEE International Symposium on Robot and Human Interactive Communication*, pages 40–45. IEEE, 2009.

[143] Michael Gillham, Gareth Howells, and Stephen Kelly. Assistive trajectories for human-in-the-loop mobile robotic platforms. In *The 6th International Conference on Emerging Security Technologies (EST)*, pages 56–61. IEEE, 2015.

[144] Markus Zimmermann and Klaus Bengler. A multimodal interaction concept for cooperative driving. In *Intelligent Vehicles Symposium (IV)*, pages 1285–1290. IEEE, 2013.

[145] Frank Ole Flemisch, Klaus Bengler, Heiner Bubb, Hermann Winner, and Ralph Bruder. Towards cooperative guidance and control of highly automated vehicles: H-mode and conduct-by-wire. *Ergonomics*, 57(3):343–360, 2014.

[146] Renjie Li, Yanan Li, Shengbo Li, Etienne Burdet, and Bo Cheng. Driver-automation indirect shared control of highly automated vehicles with intention-aware authority transition. In *2017 IEEE Intelligent Vehicles Symposium (IV)*. Institute of Electrical and Electronics Engineers, 2017.

[147] Renjie Li, Shengbo Li, Hongbo Gao, Keqiang Li, Bo Cheng, and Deyi Li. Effects of human adaptation and trust on shared control for driver-automation cooperative driving. Technical report, SAE Technical Paper, 2017.

[148] F. Mars, M. Deroo, and J. Hoc. Analysis of human-machine cooperation when driving with different degrees of haptic shared control. *IEEE Transactions on Haptics*, 7(3):324–333, July 2014.

[149] L. Saleh, P. Chevrel, F. Claveau, J. Lafay, and F. Mars. Shared steering control between a driver and an automation: Stability in the presence of driver be-

havior uncertainty. *IEEE Transactions on Intelligent Transportation Systems*, 14(2):974–983, June 2013.

[150] B Soualmi, C Sentouh, JC Popieul, and S Debernard. Automation-driver cooperative driving in presence of undetected obstacles. *Control engineering practice*, 24:106–119, 2014.

[151] US Department of Transportation - Federal Highway Administration. Manual on uniform traffic control devices (mutcd), chapter 6h: Typical applications, 2003. Online: `http://mutcd.fhwa.dot.gov/htm/2003/part6/part6h1.htm`.

[152] US Department of Transportation Federal Highway Administration. Manual on uniform traffic control devices (mutcd), chapter 3a: General, 2009. Online :`http://mutcd.fhwa.dot.gov/htm/2009/part3/part3a.htm`.

[153] US Department of Transportation Federal Highway Administration. Chapter 4: Sidewalk corridors, 2009. Online : `http://www.fhwa.dot.gov/environment/sidewalk2/sidewalks204.htm`.

[154] Natural Point. Optitrack-v100:r2. Online : `http://www.naturalpoint.com/optitrack/\\products/v100-r2/specs.html`.

[155] Hung Manh La, R.S. Lim, Jianhao Du, Sijian Zhang, Gangfeng Yan, and Weihua Sheng. Development of a small-scale research platform for intelligent transportation systems. *Intelligent Transportation Systems, IEEE Transactions on*, 13(4):1753–1762, Dec 2012.

[156] VECTORNAV . VN-100 IMU/AHRS.

[157] Mobotix. Q24 hemispheric. Online : `http://www.mobotix.com/engUS/Products/Cameras/Hemispheric-Q24`.

[158] Mobotix. Mx control center. Online : http://www.abptech.com/products/Mobotix/mxcontrol.html.

[159] D. Tran. A small-scaled testbed for integrated manual and autonomous driving. Master's thesis, Oklahoma State University, Stillwater, December 2012.

[160] Weihua Sheng, Yongsheng Ou, Duy Tran, Eyosiyas Tadesse, Meiqin Liu, and Gangfeng Yan. An integrated manual and autonomous driving framework based on driver drowsiness detection. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 4376–4381. IEEE, 2013.

[161] Carnetsoft Inc. Research driving simulator. Online : http://www.carnetsoft.com/research-simulator.html.

[162] H. Berndt and K. Dietmayer. Driver intention inference with vehicle onboard sensors. In *Vehicular Electronics and Safety (ICVES), 2009 IEEE International Conference on*, pages 102–107, Nov 2009.

[163] MathWorks. Modeling a vehicle dynamics system. Online : http://www.mathworks.com/help/ident/examples/modeling-a-vehicledynamics-system.html.

[164] Hairer, E.; Lubich, C.and Roche, M. The numerical solution of differential-algebraic systems by runge-kutta method, 1989. Online : http://archiveouverte.unige.ch/unige12339.

[165] Duy Tran, Eyosiyas Tadesse, Weihua Sheng, Meqin Liu, and Senlin Zhang. A driver assistance framework based on driver drowsiness detection. In *The 6th Annual IEEE International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (IEEE-CYBER 2016)*. IEEE, 2016.

[166] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 511–518, 2001.

[167] http://opencv.willowgarage.com/wiki/. Open Source Computer Vision Library.

[168] L Shen and Li Bai. Adaboost gabor feature selection for classification. In *Proc. of Image and Vision Computing NewZealand*, pages 77–83. Citeseer, 2004.

[169] Hung Manh La, Ronny Salim Lim, Jianhao Du, Sijian Zhang, Gangfeng Yan, and Weihua Sheng. Development of a small-scale research platform for intelligent transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1753–1762, 2012.

[170] Hoo-Chang Shin, Holger R Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, 35(5):1285–1298, 2016.

[171] NVIDIA. Embedded systems.

[172] FriendlyARM. NanoPi M3.

[173] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[174] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[175] MathWorks. Lane departure warning system. `http://www.mathworks.com/help/vision/examples/lane-departure-warning-system-1.html`.

[176] Yasuhiro Matsui, Kunio Takahashi, Ryoko Imaizumi, Kenichi Ando, et al. Car-to-pedestrian contact situations in near-miss incidents and real-world accidents in japan. In *22nd International Technical Conference on the Enhanced Safety of Vehicles*, number 110164, 2011.

[177] Pin Wang, Junhua Wang, Ching-Yao Chan, and Shouen Fang. Trajectory prediction for turning vehicles at intersections by fusing vehicle dynamics and driver's future input estimation. *Transportation Research Record: Journal of the Transportation Research Board*, (2602):68–77, 2016.

[178] Girish Chowdhary, Hassan A Kingravi, Jonathan P How, and Patricio A Vela. Bayesian nonparametric adaptive control of time-varying systems using gaussian processes. In *American Control Conference (ACC), 2013*, pages 2655–2661. IEEE, 2013.

[179] Lehel Csató and Manfred Opper. Sparse on-line gaussian processes. *Neural computation*, 14(3):641–668, 2002.

[180] Denis Osipychev, Duy Tran, Weihua Sheng, and Girish Chowdhary. Human intention-based collision avoidance for autonomous cars. In *American Control Conference (ACC), 2017*, pages 2974–2979. IEEE, 2017.

[181] Pin Wang and Ching-Yao Chan. Vehicle collision prediction at intersections based on comparison of minimal distance between vehicles and dynamic thresholds. *IET Intelligent Transport Systems*, 11(10):676–684, 2017.

[182] Praveen Batapati, Duy Tran, Weihua Sheng, Meiqin Liu, and Ruili Zeng. Video analysis for traffic anomaly detection using support vector machines. In *Intelligent Control and Automation (WCICA), 2014 11th World Congress on*, pages 5500–5505. IEEE, 2014.

[183] Duy Tran, Eyosiyas Tadesse, Praveen Batapati, Weihua Sheng, and Li Liu. A cloud based testbed for research and education in intelligent transportation system. In *Robotics and Biomimetics (ROBIO), 2015 IEEE International Conference on*, pages 452–457. IEEE, 2015.

[184] Duy Tran, Weihua Sheng, Li Liu, and Meiqin Liu. A hidden markov model based driver intention prediction system. In *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2015 IEEE International Conference on*, pages 115–120. IEEE, 2015.

[185] Denis Osipychev, Duy Tran, Weihua Sheng, Girish Chowdhary, and Ruili Zeng. Proactive mdp-based collision avoidance algorithm for autonomous cars. In *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2015 IEEE International Conference on*, pages 983–988. IEEE, 2015.

[186] Duy Tran, Eyosiyas Tadesse, Denis Osipychev, Jianhao Du, Weihua Sheng, Yuge Sun, and Heping Chen. A collaborative control framework for driver assistance systems. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 6038–6043. IEEE, 2017.

[187] Duy Tran, Ha Manh Do, Weihua Sheng, He Bai, and Girish Chowdhary. Real-time detection of distracted driving based on deep learning. *IET Intelligent Transport Systems*, 2018.

[188] Duy Tran, Jianhao Du, Weihua Sheng, Eyosias Tadesse, Denis Osipychev, Yuge Sun, and He Bai. A human-vehicle collaborative driving framework for driver assistance. (in press).

[189] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.

[190] Seong-Woo Kim, Baoxing Qin, Zhuang Jie Chong, Xiaotong Shen, Wei Liu, Marcelo H Ang, Emilio Frazzoli, and Daniela Rus. Multivehicle cooperative driving using cooperative perception: Design and experimental validation. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):663–680, 2015.

[191] Masato Ito, Daniel Mcgibney, Kosuke Sekiyama, Hiro Mukai, and Toshio Fukuda. A multiple robot cognitive sharing system using audio and video sensor. In *Micro-NanoMechatronics and Human Science (MHS), 2013 International Symposium on*, pages 1–4. IEEE, 2013.

[192] Richard Wang, Manuela Veloso, and Srinivasan Seshan. Multi-robot information sharing for complementing limited perception: A case study of moving ball interception. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1884–1889. IEEE, 2013.

[193] Ashley W Stroupe, Martin C Martin, and Tucker Balch. Distributed sensor fusion for object position estimation by multi-robot systems. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1092–1098. IEEE, 2001.

[194] Manuel Mazo, Alberto Speranzon, Karl Henrik Johansson, and Xiaoming Hu. Multi-robot tracking of a moving object using directional sensors. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 2, pages 1103–1108. IEEE, 2004.

[195] Derek Caveney and William B Dunbar. Cooperative driving: beyond v2v as an adas sensor. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 529–534. IEEE, 2012.

[196] Sae Fujii, Atsushi Fujita, Takaaki Umedu, Shigeru Kaneda, Hirozumi Yamaguchi, Teruo Higashino, and Mineo Takai. Cooperative vehicle positioning via

v2v communications and onboard sensors. In *Vehicular Technology Conference (VTC Fall), 2011 IEEE*, pages 1–5. IEEE, 2011.

[197] Ryan Parker and Shahrokh Valaee. Cooperative vehicle position estimation. In *Communications, 2007. ICC'07. IEEE International Conference on*, pages 5837–5842. IEEE, 2007.

[198] E. Belyaev, A. Vinel, K. Egiazarian, and Y. Koucheryavy. Power control in see-through overtaking assistance system. *IEEE Communications Letters*, 17(3):612–615, March 2013.

[199] Cristina Olaverri-Monreal, Pedro Gomes, Ricardo Fernandes, Fausto Vieira, and Michel Ferreira. The see-through system: A vanet-enabled assistant for overtaking maneuvers. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 123–128. IEEE, 2010.

[200] Evgeny Belyaev, Alexey Vinel, Karen Egiazarian, and Yevgeni Koucheryavy. Power control in see-through overtaking assistance system. *IEEE communications letters*, 17(3):612–615, 2013.

[201] Pedro Gomes, Fausto Vieira, and Michel Ferreira. The see-through system: From implementation to test-drive. In *Vehicular Networking Conference (VNC), 2012 IEEE*, pages 40–47. IEEE, 2012.

[202] Mathworks. What is Camera Calibration.

[203] J.V. Bouguet. Camera Calibration toolbox for MATLAB.

[204] Gwyn Topham. Volvo to test self-driving cars on London's roads next year. *Autonomtive News*, 2016. Online: `https://www.theguardian.com/technology/2016/apr/27/volvo-test-self-driving-cars-london-2017`.

[205] E Farber, J Foley, and S Scott. Visual attention design limits for its in-vehicle systems: The society of automotive engineers standard for limiting visual distraction while driving. In *Transportation Research Board Annual General Meeting*, pages 2–3. Washington DC USA, 2000.

[206] Matti Kutila, Maria Jokela, Gustav Markkula, and Maria Romera Rué. Driver distraction detection with a camera vision system. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 6, pages VI–201. IEEE, 2007.

[207] L Fletcher and A Zelinsky. Driver state monitoring to mitigate distraction. In *Proceedings of the Internal Conference on the Distractions in Driving*, pages 487–523, 2007.

[208] Afizan Azman, Qinggang Meng, and Eran Edirisinghe. Non intrusive physiological measurement for driver cognitive distraction detection: Eye and mouth movements. In *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, volume 3, pages V3–595. IEEE, 2010.

[209] T Victor, O Blomberg, and A Zelinsky. Automating the measurement of driver visual behaviours using passive stereo vision. In *Proc. Int. Conf. Series Vision Vehicles (VIV9)*, 2001.

[210] Sangho Park and Mohan Trivedi. Driver activity analysis for intelligent vehicles: issues and development framework. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pages 644–649. IEEE, 2005.

[211] Jochen Pohl, Wolfgang Birk, and Lena Westervall. A driver-distraction-based lane-keeping assistance system. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 221(4):541–552, 2007.

[212] Katja Kircher, Christer Ahlstrom, and Albert Kircher. Comparison of two eye-gaze based real-time driver distraction detection algorithms in a small-scale field operational test. In *Proc. 5th Int. Symposium on Human Factors in Driver Assessment, Training and Vehicle Design*, pages 16–23, 2009.

[213] Erik Murphy-Chutorian, Anup Doshi, and Mohan Manubhai Trivedi. Head pose estimation for driver assistance systems: A robust algorithm and experimental evaluation. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 709–714. IEEE, 2007.

[214] Shraddha Thummar and Vaishali Kalariya. A real time driver fatigue system based on eye gaze detection. *Int J Eng Res Gen Sci*, 3(1):105–110, 2015.

[215] Luke Ameen. The 25 scariest texting and driving accident statistics.

[216] Lisheng Jin, Qingning Niu, Haijing Hou, Huacai Xian, Yali Wang, and Dongdong Shi. Driver cognitive distraction detection using driving performance measures. *Discrete Dynamics in Nature and Society*, 2012, 2012.

[217] Chao LI, Tie-gen LIU, Hong-li LIU, Jun-feng JIANG, and Xiao-tian YAO. Face tracking based on haar detection and improved camshift algorithm. *Journal of Optoelectronics. Laser*, 12:021, 2011.

[218] US Department of Transportation - National Highway Traffic Safety Administration. Traffic safety facts 2012. May 2012. Online: `http://www-nrd.nhtsa.dot.gov/Pubs/812032.pdf`.

[219] National Sleep Foundation. 1.9 million drivers have fatigue-related car crashes or near misses each year. 15, 2012. Online: `http://www.sleepfoundation.org/article/press-release/19-million\-drivers-have-fatigue-related-car-crashes-or-near-misses-each-year`.

[220] T. Cowen. Can I See Your License, Registration and C.P.U? 2011.

[221] Y. Ono. Improving road safety using intelligent transportation systems (ITS) in the peoples republic of china. Online: http://cleanairinitiative.org/portal/system/files/presentations/ADB_Yuji_Ono.pdf.

[222] V. Vijayenthiran. 2013 mercedes-benz s-class to debut autonomous driving system. 2013. Online: http://www.motorauthority.com/news/1068584_2013-mercedes-benz-s-class\\-to-debut-autonomous-driving-system.

[223] V. Vijayenthiran. Autonomous driving traffic jam assistant coming to audi r8. 2013. Online: http://www.motorauthority.com/news/1071685_autonomous-driving-traffic\\-jam-assistant-coming-to-audi-a8.

[224] H. Kelly. Self-driving cars now legal in california. 2012. Online: http://www.cnn.com/2012/09/25/tech/innovation/self-driving-car-\\california/index.html.

[225] O. Thomas. Googles self-driving cars may cost more than a ferrari. 2012. Online: http://www.businessinsider.com/google-self-driving-car-sensor-cost-2012-9.

[226] AlertDriving Magazine. Human error accounts for 90 2012. Online :http://www.alertdriving.com/home/fleet-alertmagazine/.

[227] Tec. Googles self-driving car logs 300, 000 miles without an accident. 2012. Online: http://news.yahoo.com/blogs/technology-blog/google-self-driving-car-logs-300-000-miles-014608510.html.

[228] J Joseph Antony and M Suchetha. Vision based vehicle detection: A literature review. *International Journal of Applied Engineering Research*, 11(5):3128–3133, 2016.

[229] Zhiwei Tang, Ziwei Lin, Bin Li, and Longhu Chen. The embedded real-time detection system of moving object based on improved gaussian mixture model. *International Journal of Embedded Systems*, 8(2-3):119–124, 2016.

[230] Yunsheng Zhang, Chihang Zhao, Jie He, and Aiwei Chen. Vehicles detection in complex urban traffic scenes using gaussian mixture model with confidence measurement. *IET Intelligent Transport Systems*, 2016.

[231] W. Yuan and J. Wang. Gaussian mixture model based on the number of moving mehicle detection algorithm. In *2012 IEEE International Conference on Intelligent Control, Automatic Detection and High-End Equipment*, pages 94–97, July 2012.

[232] Margrit Betke, Esin Haritaoglu, and Larry S Davis. Highway scene analysis in hard real-time. In *Intelligent Transportation System, 1997. ITSC'97., IEEE Conference on*, pages 812–817. IEEE, 1997.

[233] Ben Kröse, Ben Krose, Patrick van der Smagt, and Patrick Smagt. An introduction to neural networks. 1993.

[234] Vladimir Vapnik. *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[235] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.

[236] Prasanta Chandra Mahalanobis. On the generalized distance in statistics. *Proceedings of the National Institute of Sciences (Calcutta)*, 2:49–55, 1936.

[237] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

[238] Paul E Rybski, Daniel Huber, Daniel D Morris, and Regis Hoffman. Visual classification of coarse vehicle orientation using histogram of oriented gradients features. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 921–928. IEEE, 2010.

[239] Zehang Sun, George Bebis, and Ronald Miller. On-road vehicle detection using gabor filters and support vector machines. In *Digital Signal Processing, 2002. DSP 2002. 2002 14th International Conference on*, volume 2, pages 1019–1022. IEEE, 2002.

[240] ND Matthews, PE An, D Charnley, and CJ Harris. Vehicle detection and recognition in greyscale imagery. *Control Engineering Practice*, 4(4):473–479, 1996.

[241] Quoc Bao Truong and Byung Ryong Lee. Vehicle detection algorithm using hypothesis generation and verification. In *International Conference on Intelligent Computing*, pages 534–543. Springer, 2009.

[242] Constantine Papageorgiou and Tomaso Poggio. A trainable system for object detection. *International Journal of Computer Vision*, 38(1):15–33, 2000.

[243] Chunpeng Wu, Lijuan Duan, Jun Miao, Faming Fang, and Xuebin Wang. Detection of front-view vehicle with occlusions using adaboost. In *2009 International Conference on Information Engineering and Computer Science*, pages 1–4. IEEE, 2009.

[244] Ling Mao, Mei Xie, Yi Huang, and Yuefei Zhang. Preceding vehicle detection using histograms of oriented gradients. In *Communications, Circuits and*

Systems (ICCCAS), 2010 International Conference on, pages 354–358. IEEE, 2010.

[245] Zehang Sun, George Bebis, and Ronald Miller. On-road vehicle detection using evolutionary gabor filter optimization. *IEEE Transactions on Intelligent Transportation Systems*, 6(2):125–137, 2005.

[246] Hong Cheng, Nanning Zheng, and Chong Sun. Boosted gabor features applied to vehicle detection. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 1, pages 662–666. IEEE, 2006.

[247] Yan Zhang, Stephen J Kiselewich, and William A Bauson. Legendre and gabor moments for vehicle recognition in forward collision warning. In *2006 IEEE Intelligent Transportation Systems Conference*, pages 1185–1190. IEEE, 2006.

[248] Mingxiu Lin and Xinhe Xu. Multiple vehicle visual tracking from a moving vehicle. In *Sixth International Conference on Intelligent Systems Design and Applications*, volume 2, pages 373–378. IEEE, 2006.

[249] Uwe Handmann, Thomas Kalinke, Christos Tzomakas, Martin Werner, and WV Seelen. An image processing system for driver assistance. *Image and Vision Computing*, 18(5):367–376, 2000.

[250] José Santa, Antonio F Gómez-Skarmeta, and Marc Sánchez-Artigas. Architecture and evaluation of a unified v2v and v2i communication system based on cellular networks. *Computer Communications*, 31(12):2850–2861, 2008.

[251] NTS Board. Special investigation report. the use of forward collision avoidance systems to prevent and mitigate rear-end crashes. *Washington, DC, May*, 2015.

[252] Gabriel Leen and Donal Heffernan. Expanding automotive electronic systems. *Computer*, 35(1):88–93, 2002.

[253] Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Soeren Kammel, J Zico Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, et al. Towards fully autonomous driving: Systems and algorithms. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 163–168. IEEE, 2011.

[254] Matthias Heesen, Marc Dziennus, Tobias Hesse, Anna Schieben, Claas Brunken, Christian Löper, Johann Kelsch, and Martin Baumann. Interaction design of automatic steering for collision avoidance: challenges and potentials of driver decoupling. *IET intelligent transport systems*, 9(1):95–104, 2014.

[255] German Federal Statistical Office. Unfallgeschehen im strabenvekehr 2006 (road traffic accidents 2006). 2007.

[256] Intel Inc. Developing for the intel realsense 3d camera (f200). Online : `https://software.intel.com/en-us/RealSense/F200Camera`.

[257] APlusB Software. Simuride. Online : `http://www.aplusbsoftware.com/simuride-he.html`.

[258] Lei He, Chang fu Zong, and Chang Wang. Driving intention recognition and behaviour prediction based on a double-layer hidden markov model Zhejiang University - Science C, 2012.

[259] A. Sathyanarayana, P. Boyraz, and J.H.L. Hansen. Driver behavior analysis and route recognition by hidden markov models. In *Vehicular Electronics and Safety, 2008. ICVES 2008. IEEE International Conference on*, pages 276–281, Sept 2008.

[260] Tiziana DOrazio, Marco Leo, Cataldo Guaragnella, and Arcangelo Distante. A visual approach for driver inattention detection. *Pattern Recognition*, 40(8):2341–2355, 2007.

[261] Chris Woodyard. Study: Self-driving cars have higher accident rate. *USA Today*, October 2015. Online: `http://www.usatoday.com/story/money/cars/2015/10/31/study-self-driving-cars-accidents/74946614/`.

[262] Phil LeBeau. Crash data for self-driving cars may not tell whole story. *CNBC*, October 2015. Online: `http://www.cnbc.com/2015/10/29/crash-data-for-self-driving-cars-may-not-tell-whole-story.html`.

[263] M. Barth and M. Todd. Intelligent transportation system architecture for a multi-station shared vehicle system. In *Intelligent Transportation Systems, 2000. Proceedings. 2000 IEEE*, pages 240–245, 2000.

[264] R. Verma, D. Del Vecchio, and H.K. Fathy. Development of a scaled vehicle with longitudinal dynamics of an hmmwv for an its testbed. *Mechatronics, IEEE/ASME Transactions on*, 13(1):46–57, Feb 2008.

[265] Weidong Xiang, Yue Huang, and S. Majhi. The design of a wireless access for vehicular environment (wave) prototype for intelligent transportation system (its) and vehicular infrastructure integration (vii). In *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th*, pages 1–2, Sept 2008.

[266] P. Papadimitratos, A. La Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza. Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation. *Communications Magazine, IEEE*, 47(11):84–95, November 2009.

[267] Ara V Nefian and Gary R Bradski. Detection of drivable corridors for off-road autonomous navigation. In *2006 International Conference on Image Processing*, pages 3025–3028. IEEE, 2006.

[268] D. Tran, E. Tadesse, D. Osipychev, J. Du, W. Sheng, Y. Sun, and H. Chen. A collaborative control framework for driver assistance systems. In *2017 IEEE*

International Conference on Robotics and Automation (ICRA), pages 6038–6043, May 2017.

[269] Luke Ng, Chris Clark, Jan Huissoon, and Gabriele D'Eleuterio. A decentralized reinforcement learning controller for collaborative driving. *IFAC Proceedings Volumes*, 39(20):83–88, 2006.

[270] L. Ng, C. M. Clark, and J. P. Huissoon. Reinforcement learning of adaptive longitudinal vehicle control for dynamic collaborative driving. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 907–912, June 2008.

[271] Frank Flemisch, Fawzi Nashashibi, Nadja Rauch, Anna Schieben, Sebastien Glaser, Gerald Temme, Paulo Resende, Benoit Vanholme, Christian Löper, George Thomaidis, et al. Towards highly automated driving: Intermediate report on the haveit-joint system. In *3rd European Road Transport Research Arena, TRA 2010*, 2010.

[272] Hermann Winner, Stephan Hakuli, Ralph Bruder, Ulrich Konigorski, and Bernt Schiele. Conduct-by-wire-ein neues paradigma für die weiterentwicklung der fahrerassistenz. In *Workshop Fahrerassistenzsysteme*, volume 2006, pages 112–125, 2006.

[273] Frank Flemisch, Julian Schindler, Johann Kelsch, Anna Schieben, and Daniel Damböck. Some bridging methods towards a balanced design of human-machine systems, applied to highly automated vehicles. 2008.

[274] Dipl-Psych Lars Biester. *CooperativeAutomation in Automobiles*. PhD thesis, Humboldt-Universität zu Berlin, 2008.

[275] Mark Mulder, David A Abbink, and Erwin R Boer. Sharing control with haptics seamless driver support from manual to automatic control. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 54(5):786–798, 2012.

[276] Varaiya, P. California's performance measurement system: Improving freeway efficiency through transportation intelligence. `http://.techtransfer.berkeley.edu/newsletter/02-4/pems.php`.

[277] T. F. Junge and C. Schmid. Web-based remote experimentation using a laboratory-scale optical tracker, June 2000.

[278] C.C. Ko, B.M. Chen, Jianping Chen, Y. Zhuang, and K. Chen Tan. Development of a web-based laboratory for control experiments on a coupled tank apparatus. *Education, IEEE Transactions on*, 44(1):76–86, Feb 2001.

[279] J.A. Asumadu, R. Tanner, J. Fitzmaurice, M. Kelly, H. Ogunleye, J. Belter, and Song Chin Koh. A web-based electrical and electronics remote wiring and measurement laboratory (rwmlab) instrument. *Instrumentation and Measurement, IEEE Transactions on*, 54(1):38–44, Feb 2005.

[280] S.L.T. Marin, F.J.B. Garcia, R.M. Torres, S.G. Vazquez, and A.J.L. Moreno. Implementation of a web-based educational tool for digital signal processing teaching using the technological acceptance model. *Education, IEEE Transactions on*, 48(4):632–641, Nov 2005.

[281] Z. Aydogmus and O. Aydogmus. A web-based remote access laboratory using scada. *Education, IEEE Transactions on*, 52(1):126–132, Feb 2009.

[282] A. Yazidi, H. Henao, G.-A. Capolino, F. Betin, and F. Filippetti. A web-based remote laboratory for monitoring and diagnosis of ac electrical machines. *Industrial Electronics, IEEE Transactions on*, 58(10):4950–4959, Oct 2011.

[283] J. Sanchez, S. Dormido, R. Pastor, and F. Morilla. A java/matlab-based environment for remote control system laboratories: illustrated with an inverted pendulum. *Education, IEEE Transactions on*, 47(3):321–329, Aug 2004.

[284] A. Ferrero, S. Salicone, C. Bonora, and M. Parmigiani. Remlab: a java-based remote, didactic measurement laboratory. In *Virtual and Intelligent Measurement Systems, 2002. VIMS '02. 2002 IEEE International Symposium on*, pages 117–122, 2002.

[285] M. Stefanovic, V. Cvijetkovic, M. Matijevic and V. Simic. A labviewbased remote laboratory experiments for control engineering education, June 2011.

[286] S. H. Chen, R. Chen, V. Ramakrishnan, S. Y. Hu,Y. K. Zhuang, C. C., Ko and B. M. Chen. Development of remote laboratory experimentation through internet, July 1999.

[287] J.L. Hardison, K. DeLong, P.H. Bailey, and V.J. Harward. Deploying interactive remote labs using the ilab shared architecture. In *Frontiers in Education Conference, 2008. FIE 2008. 38th Annual*, pages S2A–1–S2A–6, Oct 2008.

[288] V. Marozas, R. Jurkonis and A. Lukoeviius. Development of virtual and remote lab experimentation system for electronics engineering, 2008.

[289] M. Abdulwahed and Z. K. Nagy. Developing the TriLab, a triple access mode (handson, virtual, remote) laboratory, of a process control rig using LabVIEW and Joomla, 2010.

[290] Joanne L Harbluk, Y Ian Noy, and Moshe Eizenman. The impact of cognitive distraction on driver visual behaviour and vehicle control, 2002.

[291] Dario D Salvucci. Modeling driver behavior in a cognitive architecture. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 48(2):362–380, 2006.

[292] Yosef Sheffi and Hani Mahmassani. A model of driver behavior at high speed signalized intersections. *Transportation Science*, 15(1):50–61, 1981.

[293] L. Malta, C. Miyajima, and K. Takeda. A study of driver behavior under potential threats in vehicle traffic. *Intelligent Transportation Systems, IEEE Transactions on*, 10(2):201–210, June 2009.

[294] Yimin Wei, Huadong Meng, Hao Zhang, and Xiqin Wang. Vehicle frontal collision warning system based on improved target tracking and threat assessment. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 167–172. IEEE, 2007.

[295] Yizhen Zhang, Erik K Antonsson, and Karl Grote. A new threat assessment measure for collision avoidance systems. In *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*, pages 968–975. IEEE, 2006.

[296] Andreas Eidehall and Lars Petersson. Statistical threat assessment for general road scenes using monte carlo sampling, 2008.

[297] Mattias Brannstrom, Erik Coelingh, and Jonas Sjoberg. Model-based threat assessment for avoiding arbitrary vehicle collisions. *Intelligent Transportation Systems, IEEE Transactions on*, 11(3):658–669, 2010.

[298] Paolo Falcone, Mohammad Ali, and Jonas Sjoberg. Predictive threat assessment via reachability analysis and set invariance theory. *Intelligent Transportation Systems, IEEE Transactions on*, 12(4):1352–1361, 2011.

[299] Andreas Eidehall. Tracking and threat assessment for automotive collision avoidance, 2007.

[300] Nickens Okello and Gavin Thoms. Threat assessment using bayesian networks. In *Proceedings of the 6th International Conference on Information fusion*, pages 1102–1109, 2003.

[301] X Thong Nguyen. Threat assessment in tactical airborne environments. In *Proceedings of the Fifth International Conference on Information Fusion; Volume II*, volume 2, pages 1300–1307, 2002.

[302] Fredrik Johansson and Göran Falkman. A bayesian network approach to threat evaluation with application to an air defense scenario. In *Information Fusion, 2008 11th International Conference on*, pages 1–7. IEEE, 2008.

[303] Aurelio Locsin, Demand Media . Is air travel safer than car travel? `http://traveltips.usatoday.com/air-travel-safer-car-travel-1581.html`.

[304] Stéphanie Lefèvre, Christian Laugier, and Javier Ibañez-Guzmàn. Evaluating risk at road intersections by detecting conflicting intentions. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4841–4846. IEEE, 2012.

[305] Stéphanie Lefevre. Risk estimation at road intersections for connected vehicle safety applications, 2012.

[306] Georges S Aoude, Brandon D Luders, Kenneth KH Lee, Daniel S Levine, and Jonathan P How. Threat assessment design for driver assistance system at intersections. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 1855–1862. IEEE, 2010.

[307] Jonathan P How, Brett Bethke, Adrian Frank, Daniel Dale, and John Vian. Real-time indoor autonomous vehicle test environment. *Control Systems, IEEE*, 28(2):51–64, 2008.

[308] Velodyne. Hdl-64e. `http://velodynelidar.com/lidar/hdlproducts/hdl64e.aspx`.

[309] PointGrey. Blackfly 0.5 mp color usb3 vision (sony icx693). `http://www.ptgrey.com/blackfly-05-mp-color-usb3-vision-sony-icx693`.

[310] RobotShop. Hokuyo urg-04lx-ug01 scanning laser rangefinder. http://www.robotshop.com/en/hokuyo-urg-04lx-ug01-scanning-laser-rangefinder.html.

[311] Yue Wang, Eam Khwang Teoh, and Dinggang Shen. Lane detection and tracking using b-snake. *Image and Vision computing*, 22(4):269–280, 2004.

[312] Yong Zhou, Rong Xu, Xiaofeng Hu, and Qingtai Ye. A robust lane detection and tracking method based on computer vision. *Measurement science and technology*, 17(4):736, 2006.

[313] Ha Manh Do, Minh Pham, Weihua Sheng, Dan Yang, and Meiqin Liu. Rish: A robot-integrated smart home for elderly care. *Robotics and Autonomous Systems*, 101:74–92, 2018.

[314] Minh Pham, Yehenew Mengistu, Ha Do, and Weihua Sheng. Delivering home healthcare through a cloud-based smart home environment (coshe). *Future Generation Computer Systems*, 81:129–140, 2018.

[315] G. McCall, E. Boevers, E. Harrington, A.J. Bishop, H. Do, and W. Sheng. Emergent themes in the likes and dislikes of social robots expressed by older adults. *Gerontechnology*, 17(suppl):123, 2018.

[316] E. Harrington, H.M. Do, G. McCall, H. Boevers, A.J. Bishop, and W. Sheng. Older adult interaction with social robots: implications for socio-emotional well-being. *Gerontechnology*, 17(suppl):121, 2018.

[317] E. Boevers, G. McCall, E. Harrington, H. Do, A.J. Bishop, and W. Sheng. Exploring older adult concerns regarding acceptance and use of social companion robots. *Gerontechnology*, 17(suppl):120, 2018.

[318] Francisco Erivaldo Femandes, Ha Manh Do, Kiran Muniraju, Weihua Sheng, and Alex J Bishop. Cognitive orientation assessment for older adults using social robots. In *Robotics and Biomimetics (ROBIO), 2017 IEEE International Conference on*, pages 196–201. IEEE, 2017.

[319] Ye Gu, Ha Do, Yongsheng Ou, and Weihua Sheng. Human gesture recognition through a kinect sensor. In *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*, pages 1379–1384. IEEE, 2012.

[320] Ha Manh Do, Weihua Sheng, and Meiqin Liu. An open platform of auditory perception for home service robots. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 6161–6166. IEEE, 2015.

[321] Yehenew Mengistu, Minh Pham, Ha Manh Do, and Weihua Sheng. Autohydrate: A wearable hydration monitoring system. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 1857–1862. IEEE, 2016.

[322] Minh Pham, Yehenew Mengistu, Ha Manh Do, and Weihua Sheng. Cloud-based smart home environment (coshe) for home healthcare. In *Automation Science and Engineering (CASE), 2016 IEEE International Conference on*, pages 483–488. IEEE, 2016.

[323] Francisco Erivaldo Fernandes, Guanci Yang, Ha Manh Do, and Weihua Sheng. Detection of privacy-sensitive situations for social robots in smart homes. In *Automation Science and Engineering (CASE), 2016 IEEE International Conference on*, pages 727–732. IEEE, 2016.

[324] Ha Manh Do, Weihua Sheng, Meiqin Liu, and Senlin Zhang. Context-aware sound event recognition for home service robots. In *Automation Science and*

Engineering (CASE), 2016 IEEE International Conference on, pages 739–744. IEEE, 2016.

[325] Ha Manh Do. Developing a home service robot platform for smart homes, 2015.

VITA

Duy Tran

Candidate for the Degree of

Doctor of Philosophy

Dissertation: HUMAN-VEHICLE COLLABORATIVE DRIVING TO IMPROVE TRANSPORTATION SAFETY

Major Field: Electrical Engineering

Biographical:

Education:

Completed the requirements for the degree of Doctor of Philosophy in Electrical Engineering at Oklahoma State University in December, 2018

Received the M.S. degree in Electrical Engineering from Oklahoma State University, Stillwater, Oklahoma, 2012

Received the B.S. degree in Computer Engineering from Oklahoma State University, Stillwater, Oklahoma, 2010