TOWARDS SUSTAINABLE AUTONOMOUS VEHICLES

XINXIN DU

(B.Eng (1st Class Hons), NTU)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2016

DECLARATION

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

[illegible]

Xinxin Du
5 July 2016

Acknowledgments

First of all, I would like to express my special and great appreciation to my supervisor, Prof. Tan Kok Kiong. He has been supportive since the first day I joint his research group. I am very grateful for the trust he has on me and the freedom he gives to me to work on the things that are really interested to me. I still remember the meetings and discussions in the early mornings and emails in the late nights. Without his guidance, trust and support, I would not be able to achieve what I have done today. His encouragement and praise have brought me through the difficulties and harsh times in my Ph.D journey. I hope I did not disappoint him. He is not only my PhD supervisor, but also my life mentor.

I also would like to thank Mr. Edmund Tang and Mr. Harry Setiawan Lim from Toyota Tsusho Asia Pacific PTE. LTD (TTAP). Without their support and coordination in building and maintaining the testing vehicle, my Ph.D project would not be as smooth as it was. My special thanks also go to Mr. Sekiguchi and Mr. Ishikawa from ZMP INC. Japan for their help in debugging the low level control firmware for the autonomous vehicle.

Many thanks go to Mr. Tan Chee Siong in the Control & Simulation Lab, for his help in coordinating the project demonstrations and in all other administration matters. He saved me from a lot of troubles. I also would like to thank my lab mates. Thank you all for the joy times we have spent together. You guys are wonderful and have made my Ph.D journey more meaningful and enjoyable.

Last but not least, I will always owe my gratitude to my family, my wife Ms. Wang Mengmeng, my father Mr. Du Hongjie and my mother Ms. Han Zhimei, for their love, encouragement and support. Without them as my backup, I would fail half way.

Contents

Summary

Various entities, including traditional auto makers and new rising IT giants, have been investing huge amount of resources into the development of autonomous vehicles. The momentum and expectation have been brought up to such a high level that people have generally acknowledged it is just a matter of time before technologies take control of the steering wheels over human drivers.

However, in the current typical setups, the autonomous vehicles are always equipped with dozens of ranging sensors for environment perception, resulting in overly-complex systems. These setups attract high cost and slow down the commercialization process even though the sensor costs have dropped significantly. The power consumption from these sensors is another big concern for long-distance travelling, especially for electric vehicles.

This thesis aims to propose a more sustainable solution towards autonomous vehicles by hybridizing a minimum viable vision-based autonomous vehicle with remote human intervention. The remote human driver takes over the control of the vehicle when the limits of autonomous technology are stretched. Human touch and technology co-exist in the solution, allowing a fine balance in cost, safety and efficiency to be achieved without the need to push for 100% fail-proof solutions which would otherwise be extremely costly and complex.

Due to the huge amount of research involved, this thesis only elaborates on the minimum viable vision-based autonomous vehicle development, while remote human intervention is covered in my teammate's thesis. The minimum viable solution consists of a vision-based vehicle lane-level localization system, a lane following control system based on nonlinear model predictive control (NMPC) scheme and a vision-based fully autonomous parking system.

First of all, two vision systems were proposed for the vehicle lane-level localization. In both systems, the lane line markings can be detected and tracked. The vehicle's location and orientation with respect to the detected lane lines can be estimated as well. This information serves as the feedback to the NMPC system which controls the vehicle to follow the detected lane.

In the first system, it uses one single camera to capture road images. Four key steps are involved: Lane line pixels are first pooled with a ridge detector. An effective noise filtering mechanism next removes noise pixels to a large extent. A modified version of sequential RANSAC (RANdom Sample Consensus) is then adopted in a model fitting procedure to capture all the lane lines in the image. Finally, if lane lines on both sides of the road exist, a parallelism reinforcement technique is imposed to improve the model accuracy.

Built upon this mono-camera system, a stereovision system is proposed to further improve the detection accuracy and consistency. It integrates a new lane line detection algorithm with other lane marking detectors (e.g. zebra crossing/hump/warning letters/arrows etc.) to effectively identify the correct lane line markings and remove noise pixels. It also fits multiple road models to improve accuracy. An effective stereo 3D reconstruction method is proposed to estimate vehicle localization. The estimation consistency is further guaranteed by a new particle filter framework, which takes vehicle dynamics into account.

Based on the feedback of the stereovision system, an NMPC scheme is proposed to control the vehicle velocity and steering simultaneously to follow the detected lane. The optimization solver for the NMPC is based on genetic algorithms (GA). As compared to other solvers, using GA in the optimization enables a more flexible structure for MPC formulation. The cost function and constraints can be designed in a more accurate, meaningful and direct way. Moreover, passengers safety and comfort are well taken care of under the proposed NMPC scheme as both the vehicle movement acceleration and steering acceleration are well confined within a safety range.

Another fundamental task for the autonomous vehicle is self-parking. In this thesis, a low-cost vision-based reverse parking system is proposed. It consists of four key modules: a novel path planning module ensures that a feasible path is available under any given initial poses, which frees human intervention completely; a modified sliding mode control (SMC) module on the steering wheel is designed for path following; the image processing module with Kalman state prediction provides consistent and real-time estimation on the vehicle pose; and a robust overall control module ensures that the vehicle can park along the slot center line accurately without intrusion into adjacent slots.

In conclusion, a minimum viable vision-based approach for autonomous vehicles is proposed in this thesis. Integrating it with the remote human intervention leads to a more sustainable autonomous vehicle solution with a fine balance in cost, safety and efficiency. Currently, obstacle avoidance, together with other exceptions, are under the responsibilities of remote human drivers. As the technology development advances, they will be eventually embedded into the minimum viable vision solution and remote drivers will be purely dedicated to emergencies and contingencies.

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

With the recent advance and development in technologies, autonomous driving vehicles have drawn a lot of interest from researchers, media and general public. It has been generally acknowledged that it is just a matter of time before the technology takes over the control of the wheels from human drivers, with the vision to reduce car accident, traffic jam, carbon dioxide emission and so on.

Recently, more and more car models are equipped with some kinds of Advanced Driver Assistance Systems (ADAS), such as autonomous cruise control (ACC), autonomous emergency braking, lane keeping and so on. As shown in Fig. 1.1, ADAS can be considered as the transition between traditional and autonomous vehicles.

The successful implementation of ADAS has led to promising forecasts on the fully autonomous driving. In the report by KPMG and Center for Automotive Research [1], they predict that the sufficient built-in and after-market penetration to support self-driving applications will be ready by the year 2025. In [8], IEEE predicts that up to 75 percent of cars on the road will be autonomous by the year 2040. China Association of Automobile Manufactures (CAAM) also tries to commercialize autonomous vehicles by the year 2030.

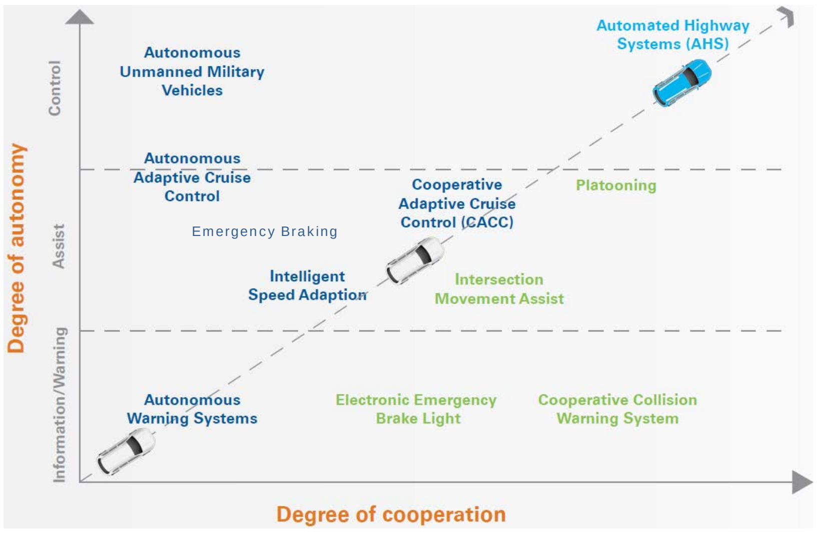

Degree of autonomy
Control
Assist
Information/Warning
Degree of cooperation
Automated Highway Systems (AHS)
Autonomous Unmanned Military Vehicles
Autonomous Adaptive Cruise Control
Cooperative Adaptive Cruise Control (CACC)
Platooning
Emergency Braking
Intelligent Speed Adaption
Intersection Movement Assist
Autonomous Warning Systems
Electronic Emergency Brake Light
Cooperative Collision Warning System

Figure 1.1: The degree of autonomy in autonomous driving vehicles [1]

According to [9], by adding various ADAS to existing car models, the automakers have already gained additional 30 billion USD revenue in the year 2014. By 2030, with the increase in the degree of autonomy, autonomous vehicles are expected to bring additional 250 billion USD revenue yearly.

Besides this, the autonomous vehicle technology will create new opportunities for the automotive value chain and bring in new players to join incumbents looking to capitalize on a new market. Prospective players along the value chain should expect significant changes over time, inevitably creating opportunities for new entrants.

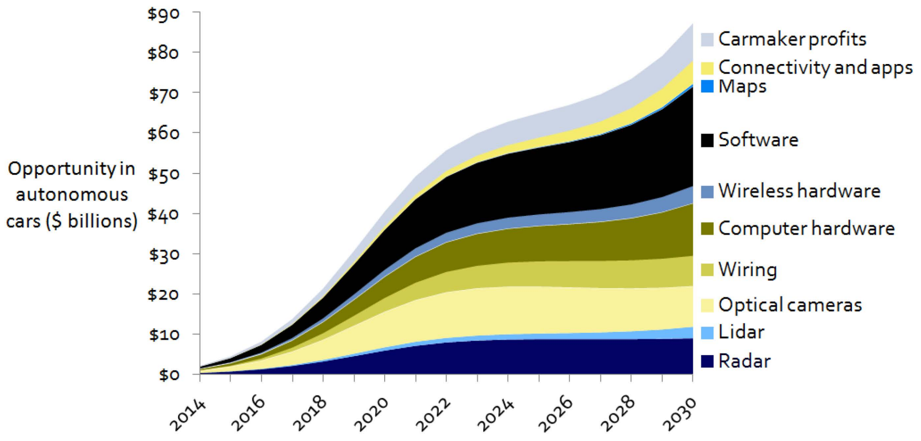


Figure 1.2: Opportunities created by autonomous vehicles in different fields [2]

As forecasted in the investigation report by Lux Research, Inc.[2], the new opportunity will reach 87 billion USD by the year 2030, which is shared among players from different fields as shown in Fig. 1.2. Software will capture the largest slice in the market pi, followed by optical cameras.

The Society of Automotive Engineers (SAE) US and Verband der Automobilindustrie (VDA) Germany have set up the standards for autonomous vehicles to guide the development. CAAM has also started to define their own standards in preparation for the autonomous vehicle commercialization.

To further support the development, some countries, including Swizerland, US and Japan, have established virtual towns for autonomous vehicle tests. Fig. 1.3 shows the virtual town Mcity Test Facility deployed at the University of Michigan, US. So far, they have established collaborations with Ford, GM, Honda, Nissan and so on.

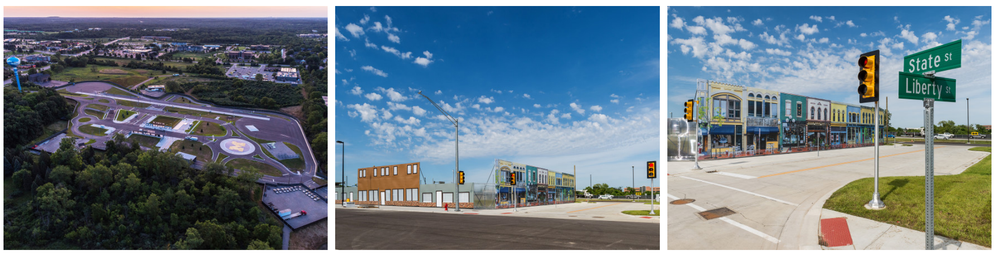


Figure 1.3: Virtual town for autonomous vehicle testing at University of Michigan [3]

Currently, almost every traditional auto-maker has their own road map towards autonomous vehicles and has started their own research and development in order to capture the booming market in the near future. Collaborations with IT giants and virtual capitals have also been initiated and reported.

Ford announced its Smart Mobility blueprint at CES 2015 (Consumer Electronics Show), aiming to take the company to the next level in connectivity, mobility, autonomous vehicles, customer experience and big data [10]. Ford also highlighted the semi-autonomous vehicles it has on the road today (Fig. 1.4) and the fully autonomous vehicles now in development for the future.

BMW has teamed up with Baidu, a Chinese search giant, to test its own autonomous vehicles (Fig. 1.4) in Beijing, joining the race with Google to get the first autonomous vehicle to the market [11]. This kind of joint venture seems to become more and more popular as auto-makers have the advantages in car design, manufacture, brand and retail channel, while IT companies have the advantages in big data analysis, user database and software development.

Figure 1.4: Autonomous vehicles by Ford (left) and BMW (right)

Tesla, a rising giant in the electric vehicle industry, has launched Model S (Fig. 1.5) as its showcase towards autonomous electric vehicle. The autopilot function is designed to learn from human drivers, improve its driving skills automatically and share its driving knowledge with other cars [12]. The technology to make a fully autonomous vehicle in Tesla will be ready in five or six years as stated by its CEO Elon Musk in his interview with The Wall Street Journal [13].

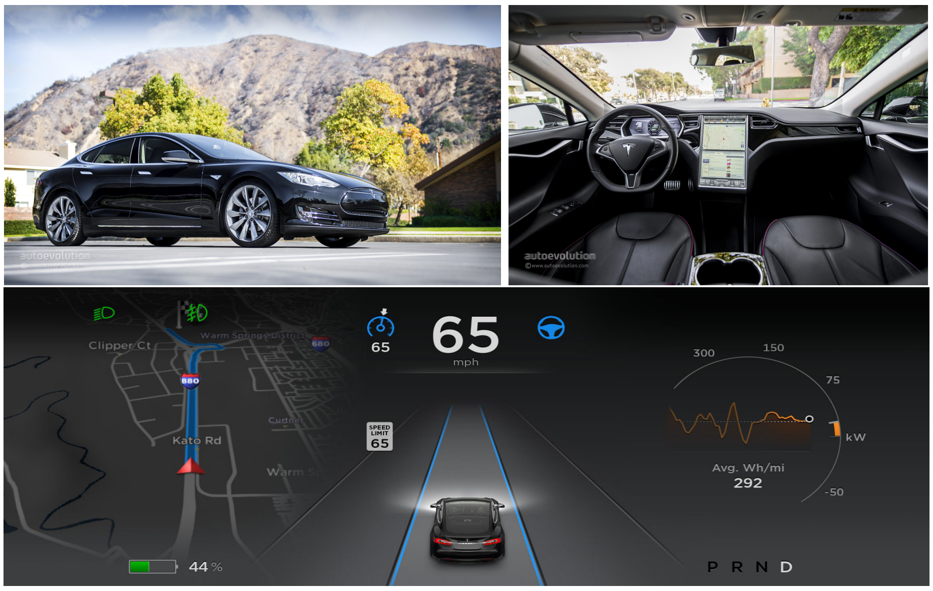


Figure 1.5: Tesla Model S, its interior and the autopilot system

Furthermore, some giants from other industries especially IT, also sensed the huge potential in the autonomous vehicle market and stepped into it in a more aggressive manner. The traditional auto-makers tend to gradually enhance existing cars with self-driving features [9], taking the "improvement" path. But the new entrants plan to achieve self-driving at one go through artificial intelligence and computer vision, taking the "revolution" path.

The most famous one is the search giant Google. 53 Google driverless cars are being tested at the city of Mountain View of California and Austin of Texas. Out of the 53 cars, 23 are Lexus RX450h SUV (Fig. 1.6 left) and 30 are Google's own two-seater electric vehicle (Fig. 1.6 right). Since 2009, the travelling distance of these driverless cars has accumulated to 2.3 million miles with 60% of the miles under autonomous driving. Google has also shifted its focus from highway-oriented autonomous driving to local-street driving, which is more realistic but more challenging.

Google is the biggest thread to the traditional auto-makers, not only from the self-driving technologies, but also from its advantages on the Android platform which can seamlessly connect the car system to the smart phone users.

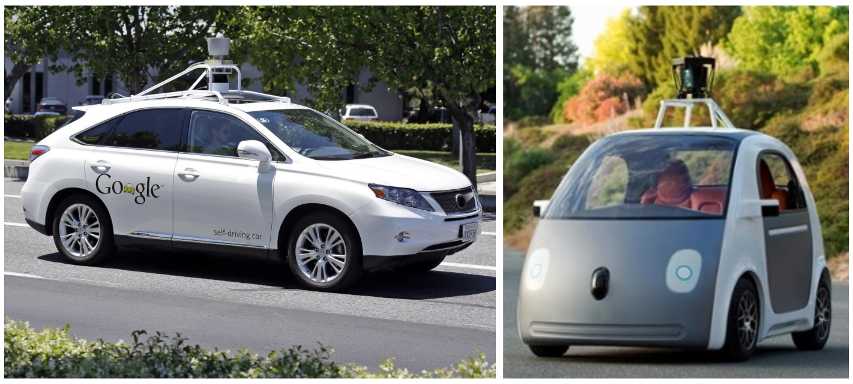


Figure 1.6: Google driverless cars

Fellow tech giant Apple is also widely rumoured to be plotting its move into this industry. The Apple Car is reportedly to be fully electrical with in-car connectivity like the CarPlay system. It is expected to be the next break-through product for the company after iPhone and iPad. According to the domain name service provider "who.is", in December 2015, Apple registered several domain names related to cars, including *apple.car*, *apple.cars* and *apple.auto*.

Uber, the biggest car sharing company, after seeing the potential in autonomous vehicles to change the way of car sharing, established its own robotics center called Advanced Technologies Center in Pittsburgh and poached a number of researchers from the nearby Carnegie Mellon University, who are specialized in autonomous vehicle research.

TomTom, a navigation system provider from Holland, has started to establish high resolution map specifically for autonomous driving vehicles. The map will contain the information on road line markings, traffic lights position and height, road signboards and etc.. So far, they have finished the map for the Interstate 280 at California US and A81 high way at Germany. By the year 2017, the map will cover North America and most of Europe.

Apart from industries, universities and research institutes also play unreplaceable roles to bring forward the autonomous technologies. The embryo of the Google driverless car was built upon the robotic vehicle "Stanley" (Fig. 1.8 left image) designed by Thrun's team at Stanford University [14] in 2005.

Globally, almost every top university has dedicated research resources working on autonomous driving related technologies. Locally at Singapore, both National University of Singapore (NUS) and Nanyang Technological University (NTU) have established research teams to look into different perspectives of autonomous vehicles.

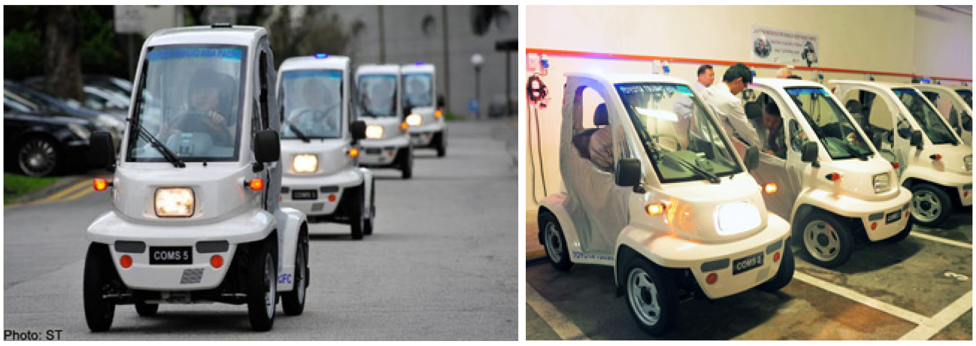


Figure 1.7: Micro electric vehicle project commenced by NUS and TTAP

The work described in this thesis is part of Micro Electric Vehicles Project jointly commenced by NUS and Toyota Tsusho Asia Pacific PTE. LTD (TTAP), attempting to solve the last mile challenges using autonomous vehicles. In this joint project, TTAP provided test vehicles as shown in Fig. 1.7 while NUS put in research resources to study viable and sustainable solutions.

To push forward the development and bring awareness to the public, different kinds of conferences, forums and competitions on autonomous vehicles are being organized every year all over the world. The DARPA (Defence Advanced Research Projects Agency) Challenge is one of the famous, with the vision to spur the development of technologies needed for fully autonomous vehicles. The aforementioned "Stanley" vehicle was the winner in 2005 (Fig. 1.8).

The CES was the show for electronics products initially, like hand phone, play stations and so on. But for the recent years, more and more auto-makers use this platform to release their new prototypes and products that are related to autonomous vehicles. In this year, more than 500 companies in automotive industry participated in this show and occupied one quarter of the show area. Fig. 1.8 (right) shows Toyota's conceptual design for the autonomous electric vehicle at CES 2016 Las Vegas.

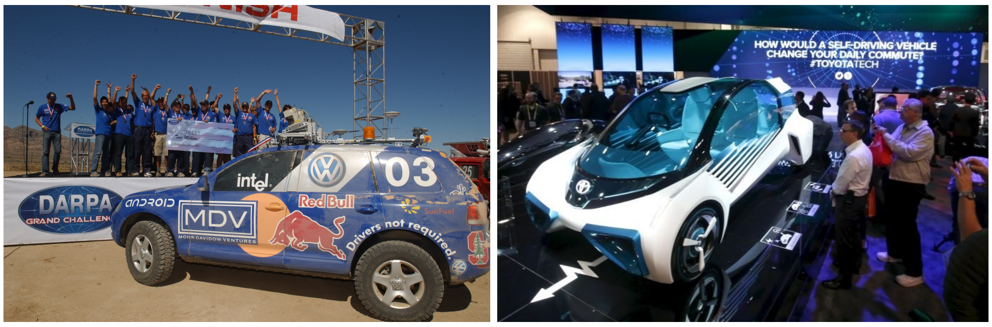


Figure 1.8: The "Stanley" vehicle at DARPA 2005 and Toyota booth at CES 2016

With the autonomous driving technologies becoming more and more mature and pervasive, some significant paradigm changes are expected for the automotive ecosystem and our daily commutes, some of which will offer enormous benefits economically and socially.

The autonomous vehicles are expected to improve the driving safety and eliminate car crashes ultimately. Because their driving patterns are more predictable, controllable and systematic as compared to human drivers. They can adhere the traffic rules and regulations as explicitly as being programmed. Globally, this will save more than 1.24 million lives a year and save additional 20-50 million people from being injured or disabled [15].

In a recent study carried out by Virginia Tech Transportation Institute (VT-TI), at all the crash severity levels (level 1 is the most severe level) defined by SHRP 2 NDS (Strategic Highway Research Program 2 Naturalistic Driving S-tudy), the crash rate per million miles is lower in the autonomous vehicles than the normal vehicles [4]. The detailed comparison is shown in Fig. 1.9, where the crash rate for normal vehicles is referred as SHRP 2 Age-Adjusted. This data is adjusted based on SHRP 2 NDS for a fair comparison.

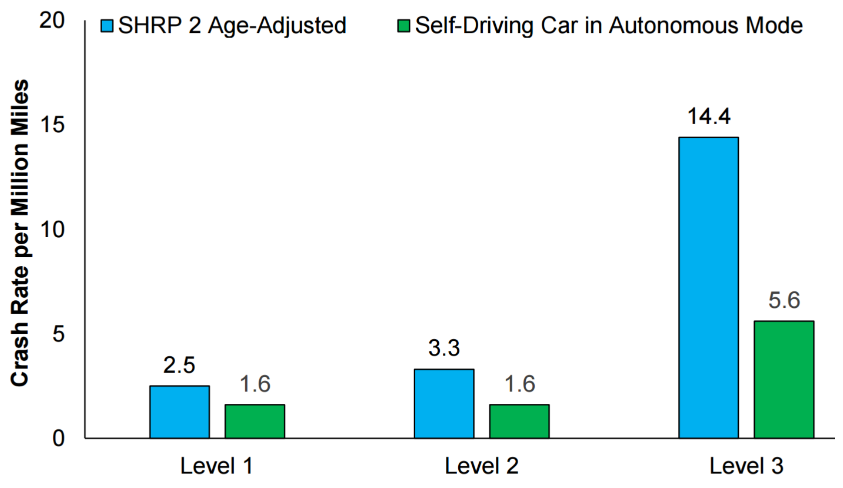


Figure 1.9: Comparison on crash rate between autonomous vehicles and normal vehicles (SHRP 2 Age-Adjusted) [4]

Uncertainties in anticipation of the travel time will be eliminated or at least reduced substantially. This will bring great convenience to the travellers and they can plan their trips more precisely without wasting waiting time. At the same time, the vehicle can plan a more reliable and efficient path from its origin to its destination. Consequently, traffic jam can be reduced as well. Furthermore, with this, the delivery service companies are able to predict and schedule their delivery more precisely and provide better service to their customers.

Autonomous vehicles could redefine the vehicle ownership models significantly and expand opportunities for vehicle sharing. If vehicles can drive automatically, they can be summoned when needed and redistributed to other duties when the trip is over. Thus, travellers would no longer need to own the vehicles but could instead purchase mobility services on demand [1]. This is also the main reason why Uber is rushing into this market as aforementioned. Consequently, it will lead to a car-lite society, which benefits the environment and further reduces traffic congestions. Moreover, autonomous vehicles can be used more efficiently than private cars throughout the day instead of being parked most of the day and night.

The time we spend inside the autonomous vehicle will become less boring and more productive. The autonomous vehicle will save people from being focused on driving, instead, they will have more time for entertainment, reading news, watching movies, online chatting and so on. This kind of information ecosystem is one of the major interests to those IT giants, out of which they can make profit, similar to their websites, smart TV sets and smart phone applications. Fig. 1.10 left image illustrates this concept where the passengers can lie down and enjoy a movie without worrying about the driving.

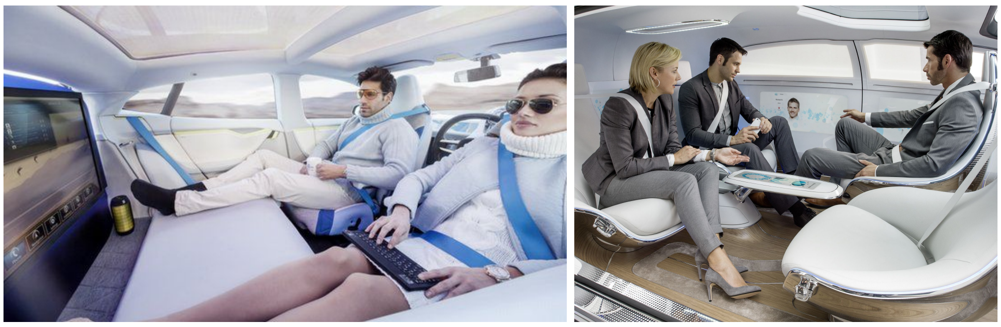


Figure 1.10: Autonomous vehicles redefines driving: entertainment and working

In the right image, it shows another concept where the autonomous vehicle is turned into an office where people can conduct meetings and discussions inside. The car doors themselves are display screens. This concept can be found in the Mercedes-Bens futuristic model F015.

In summary, autonomous vehicles are the future for automotive industry. It is a huge market that everyone wants to share a piece of it, varying from traditional auto-makers, IT giants to universities. The time to reshuffle the automotive industry is coming, with the opportunities for new entrants to capture the market and with the risks for traditional players to be eliminated.

1.2 Current gaps and problems

With the inputs and efforts from auto-makers, IT giants and even governments, the future of autonomous vehicles is prosperous. But before they finally come into our daily life, three major challenges have to be addressed in advance, including laws (and regulations), information security and technology.

The technologies involved in autonomous vehicles can be roughly divided into two categories: external and internal. External ones mainly refer to the vehicle to vehicle (V2V) communication and vehicle to infrastructure (V2X) communication. Internal ones include the vehicle's ability to perceive the environment and its ability to control its movement in response to the environment. This thesis aims to tackle the challenges related to the internal technologies of autonomous vehicles, especially for level 5 autonomous driving. At this level, the vehicle is supposed to make all decisions on its own, instead of the driver or passengers. Therefore, the requirement on the perception accuracy is much higher and more challenging.

The environment perception is the most basic aspect to enable autonomous vehicles and this is the main focus in autonomous vehicle research. The common setup adopted in most of the autonomous vehicles is shown in Fig. 1.11.

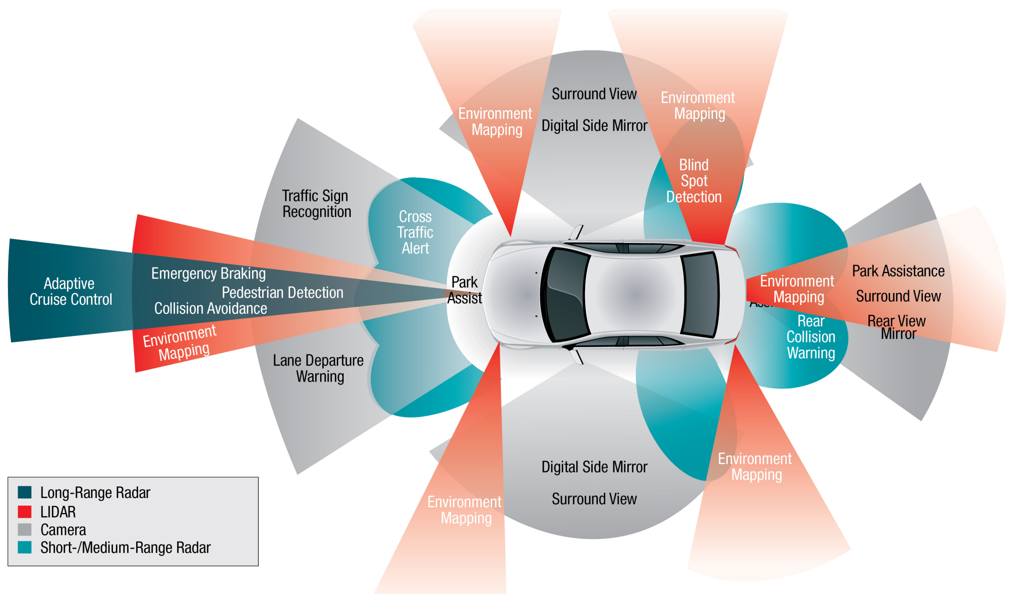


Figure 1.11: Illustration on the environment perception from on-board sensors[5]

The long-range radar is for the detection of obstacles in the far distance, at around 100 to 200 meters while the short range radar (including ultrasonic sensors and sonar) is for nearby obstacles, including pedestrians and road curbs. The working distance is within about 6 meters. In the current commercial vehicles, the short-range radar is mainly used in reversing alert system.

LIDAR (Lighting Detection and Ranging) is mainly for mid-range obstacle detection within 100 meters. It sends millions of light pulses per second in a well-designed pattern. With its rotating axis, it is able to create a dynamic, three-dimensional map of the environment. LIDAR is the heart for object detection for most of the existing autonomous vehicles. Fig. 1.12 shows the ideal detection results from LIDAR, with all the moving objects being detected.

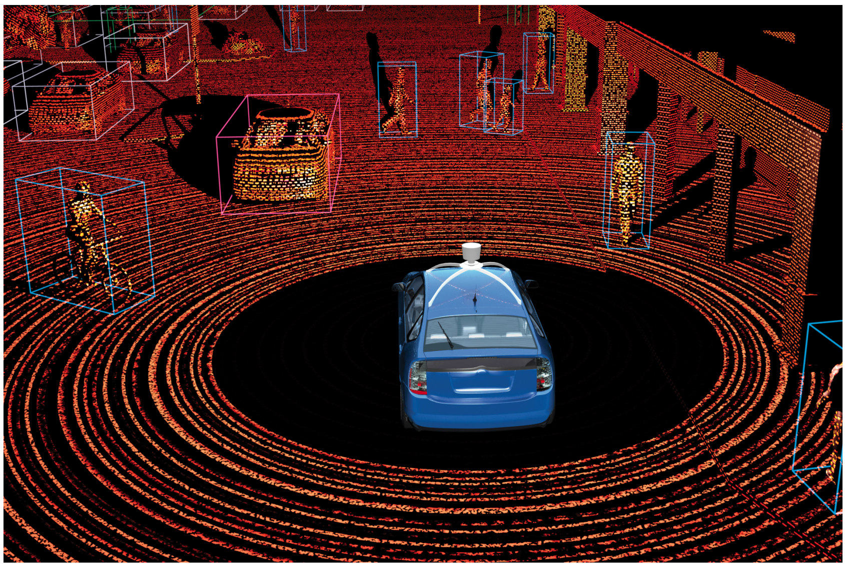


Figure 1.12: The ideal detection result from LIDAR with all moving objects detected

Cameras are also heavily involved in the perception of the environment. With the development of image processing technologies, they will play more and more roles in autonomous vehicles since they are able to present direct visual results to human.

In the current setup, the detection results from these four kinds of sensors are fused through different technologies in order to achieve reliable results. However, the number of sensors involved is much more than four, resulting in a very complicated system. The amount of data to be analysed in unit time is numerous. For example, the "Autonomos" car from University of Nevada has 7 LIDARs, 7 Radars and 9 cameras; the "Junior" car from Stanford University has 5 LIDARs and 5 BOSCH Radars [6].

Due to the huge amount of data to be processed in real time, it requires several powerful computing units working in parallel. The "Junior" car aforementioned contains 30 units to process the data collected by the sensors. Fig. 1.13 shows the sensor setup and its computing units in the trunk.

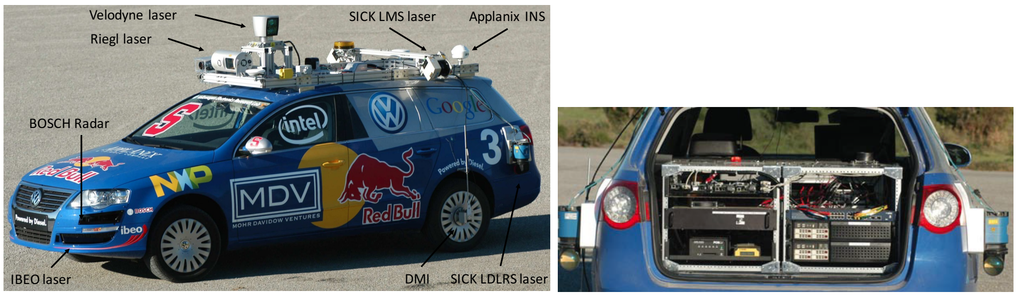


Figure 1.13: The sensor setup and computing units installed on "Junior" from Stanford University [6]

This over-complex setup results in very expensive autonomous vehicles, which is the main factor slowing down the commercialization process. Even though the price of LIDAR sensors has dropped significantly, the cheapest one from Velodyne still costs 8,000USD, which has only 16 channels and covers only 2D. For a typical 3D LIDAR with 64 channels, the price is still more than 70,000USD. If the total cost of autonomous vehicle is not able to drop down to an acceptable level, they will become the "big toys" for the rich only. The power consumption from LIDARs, Radars and computers is another big issue for long-distance travelling, especially for electric vehicles.

Therefore, the current approach is not a sustainable and scalable way for the development of autonomous vehicles, unless the cost of LIDAR can further drop down to the level of 1,000USD in the near future, which is very unlikely. A more sustainable solution is highly desirable.

Cameras seem to have the potential to replace LIDARs and Radars, especially with the development of image processing technologies and with the increase in computation power. They are much cheaper, more compact and less power consuming than LIDARs and Radars. Human can drive a car with eyes, why cannot the computer drive a car with cameras? Due to its huge potential, cameras are projected to be the second largest share in the market opportunities created by autonomous vehicles as shown previously in Fig. 1.2 by Lux Research, Inc. [2].

However, with the current technologies, cameras are not practical and reliable enough. For algorithms that can achieve high performance, they cannot work in real time. For example, for the top three state-of-the-art algorithms (in terms of detection rate) on road detection in the KITTI database [16], all of them take about 2 seconds to finish one iteration. This is not fast enough as compared to the speed of a moving vehicle. The detection results may be subject to other noise conditions as well, such as low visibility, poor weather conditions, worn-off lane lines and etc..

In summary, the autonomous vehicles, equipped with dozens of ranging sensors for environment perception, are overly-complex, attract very high cost and slow down commercialization efforts. The power consumption by these sensors is another big concern, especially for electric vehicles. Therefore, the current autonomous vehicle solutions, while impressive in the use of technologies, may not be sustainable and scalable. An alternate one through vision system is highly desirable to achieve a fine balance in cost, efficiency and safety.

1.3 Research objectives

With cost, safety, scalability at the forefront of considerations, this thesis aims to propose a more sustainable solution towards autonomous vehicles, especially for level 5 autonomous driving. This can be achieved through a hybrid platform by integrating minimum viable vision-based autonomous vehicles with remote human intervention.

The vehicles are to be installed with cameras only (no other kinds of perceiving sensors) and work under a nominal and controlled environment. Some of the responsibilities especially under exceptional conditions will be transferred from hard core autonomous technology to technology-assisted remote human drivers at the top and contingency layer of the overall system hierarchy. The remote driver working in a central hub, oversees a fleet of vehicles and seamlessly takes over the steering wheel when the limits of autonomous technology are stretched, thus allowing the cost factor to be reduced and yet achieving a better safety net as a consequence. Technology and the human touch co-exist in this solution which allows a fine balance in cost, safety and efficiency to be achieved without the need to push for 100% fail-proof solutions in 'fully′ autonomous vehicles which will be accompanied by extreme costs and complexity.

However, due to the huge amount of research work required in this solution, this thesis will focus on the development of the vision-based autonomous vehicle part, while the remote human intervention part is concurrently developed by my teammate, Mr. Kyaw Ko Ko Htet. In his development, the challenges to be addressed include enabling a 360-degree view with minimum blind spots, sending real-time video streaming to the remote control station, minimizing latency during wireless transmission and ensuring reliability of the transmitted data. Interested readers may refer to [17] for more details.

For the vision-based autonomous vehicle part, to achieve a minimum viable solution, the following sub-systems are to be developed:

(i) A vision system that is able to identify the road lane line markings and road signs, and then work out the vehicle pose, including vehicle lateral distance to the road boundary and vehicle heading orientation deviation to the road tangent. The system must work in real time and reliably under different road conditions.

(ii) A control system that is able to control the vehicle to follow the road based on the vision detection results. It must be able to control the velocity and steering simultaneously. At the same time, passengers' safety and comfort must be guaranteed.

(iii) A fully self-reverse parking system that is able to detect the available parking slot from vision and control the vehicle to park into the slot accurately.

With these sub-systems in completion, the vehicle will be able to understand the road structure by itself without referring to high resolution digital maps, localize itself in the lane without using high-accuracy GPS (up to the level of centimeters), drive along the road under different conditions and park itself at the car park accurately.

1.4 Thesis outline

The remaining parts of the thesis are organized as: in Chapter 2, a vision system based on single camera is proposed for lane line detection and vehicle localization. In Chapter 3, a more advanced vision system based on stereo cameras is presented, which works more robustly and accurately than the system in Chapter 2. Chapter 4 proposes an NMPC scheme which controls the autonomous vehicles to follow the detected road; while Chapter 5 describes a vision-based autonomous reverse parking system. Conclusions and future works are provided in Chapter 6. Below are the summaries of each chapter.

Chapter 2: In this chapter, we propose a robust mono-camera system for vehicle lane-level localization. The approach incorporates four key steps. Lane line pixels are first pooled with a ridge detector. Next, an effective noise filtering mechanism removes noise pixels to a large extent. Then a modified version of sequential RANSAC is adopted in a model fitting procedure to capture each lane line in the image. Finally, if lane lines on both sides of the road exist, a parallelism reinforcement technique is imposed to improve the model accuracy.

The results show that the proposed approach is able to detect the lane lines accurately and at a high success rate compared to other approaches. The road model derived from the lane line detection is capable of generating precise and consistent vehicle localization information with respect to road lane lines, including road geometry, vehicle position and orientation.

Chapter 3: Built upon the previous mono-camera system, this chapter proposes a stereovision system, which is able to achieve higher accuracy and consistency. It integrates a new lane line detection algorithm with other lane marking detectors to effectively identify the correct lane line markings. It also fits multiple road models to improve accuracy. An effective stereo 3D reconstruction method is proposed to estimate vehicle localization. The estimation consistency is further guaranteed by a new particle filter framework, which takes vehicle dynamics into account.

Experiment results based on image sequences taken under different illumination conditions showed that the proposed system can identify the lane line markings with 98.6% accuracy. The maximum estimation error of the vehicle distance to lane lines is $16cm$ in daytime and $26cm$ at night, and the maximum estimation error of its moving direction respected to the road tangent is $0.06rad$ in daytime and $0.12rad$ at night.

Chapter 4: An NMPC scheme is proposed which controls the vehicle velocity and steering simultaneously to follow the detected lane line from the stereovision system. The optimization solver for the MPC is based on genetic algorithms (GA). As compared to other solvers in the literature, using GA in the optimization enables a more flexible structure for MPC formulation. The cost function and constraints can be designed in a more accurate, meaningful and direct way. Otherwise, they have to be formulated into certain formats and meet all prerequisites in order to implement the optimization solvers.

Both simulation and on-field test results showed that the vehicle under the control of the proposed nonlinear MPC is able to follow the road center line accurately and consistently, even at sharp corners. Moreover, the results also showed that passengers' safety and comfort can be well taken care of under the proposed MPC scheme as both the vehicle movement acceleration and steering acceleration are well confined within a safety range.

Chapter 5: This chapter proposes a low-cost vision-based fully self-reverse parking system. It consists of four key modules: a novel path planning module ensures that a feasible path is available under any initial poses, which frees human intervention completely; a modified sliding mode control (SMC) module on the steering wheel is designed for path following; the image processing module with Kalman state prediction provides consistent and real-time estimation on the vehicle pose; and a robust overall control module ensures that the vehicle can park along the slot center line accurately without intrusion into adjacent slots.

Experiment results based on 216 on-field tests under different illumination conditions showed that the proposed system is able to park the vehicle accurately and consistently in all cases with a $4.71cm$ RMS offset distance from center line and 1.24° RMS orientation deviation.

Chapter 6: This chapter concludes the entire thesis and proposes future works. In this thesis, a minimum viable vision-based autonomous vehicle solution is proposed, in which the vehicle is able to follow the road lane lines accurately and consistently based on the stereo vision system and the NMPC control system. It is also able to automatically park itself into the slot based on the vision parking system. Hybridizing it with remote human intervention leads to a more sustainable autonomous vehicle solution as compared to the current approaches, with a better balance in cost, safety and efficiency.

However, in the current hybrid system, some other routine tasks, such as obstacle detection through vision (which itself can be a PhD research topic), are handled by the remote driver. In the future, all of them will be eventually transferred back to the autonomous vehicles and the human efforts will be dedicated for contingencies and emergencies only.

1.5 Hardware platform

The testing vehicle used throughout the thesis is a one-seater electric vehicle, COMS from Toyota. COMS stands for "Chotto Odekake Machimade Suisui" in Japanese, meaning a little smooth driving into town. The detailed dimensions are illustrated in Fig. 1.14. It can travel up to $40km/hr$.

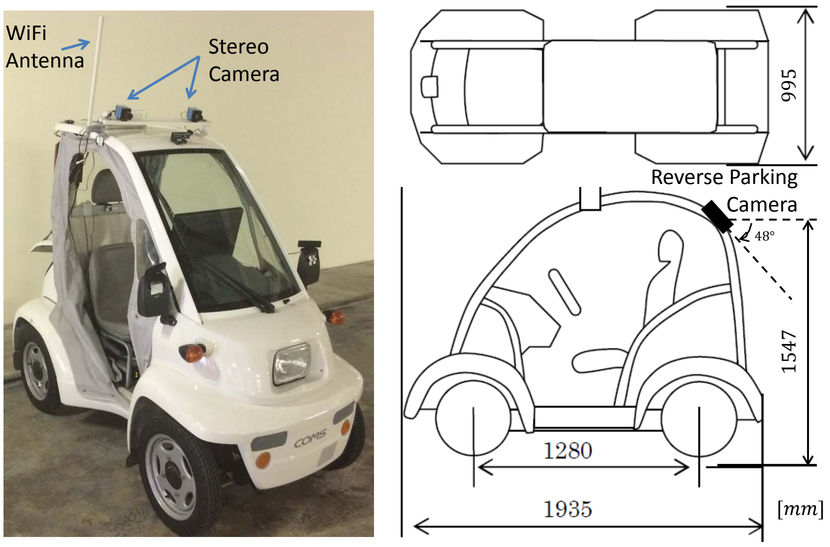


Figure 1.14: Toyota COMS EV and its dimensions

The sensors involved in this platform include a driving speed sensor, a s-teering wheel position sensor, a pair of stereo cameras and a reverse parking camera. The super WiFi antenna and the remaining cameras (wide angle) as shown in Fig. 1.14 are for the remote control purpose, which is out of the scope of the thesis. The stereo cameras are mainly for road lane line and road markings detection while the reverse parking camera is for parking slot detection.

As shown in the communication diagram Fig. 1.15, the computer system consists of three hierarchies. The communications between different hierarchies are through different protocols for the ease of accessibility.

The micro-computer system is the lowest level, running Linux. It directly communicates to the driving speed sensor and steering wheel position sensor and broadcasts the sensor readings (v, ϕ) to the higher level at $100Hz$. When it receives commands (v_t, ϕ_t) from the higher level, it can control the movement

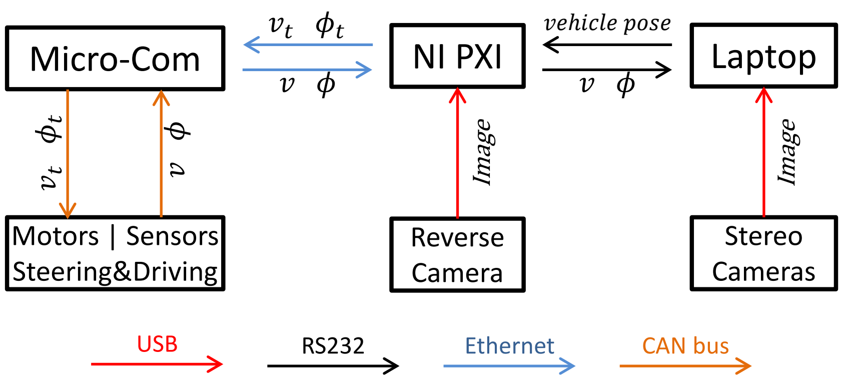


Figure 1.15: Communication diagram of the computer system

of the steering wheel and driving motors through the steer-by-wire system and drive-by-wire system.

The NI (National Instruments) PXI computer, which runs Window7, is the middle level. It is in charge of the control algorithms for both NMPC and reverse parking. The image process for reverse parking is also carried out by this computer.

The top level is a laptop which directly connects to the stereo cameras. Its main focus is lane line detection. Based on the detection results, it calculates and sends the vehicle pose with respect to the road to NI PXI, from which NI PXI will work out the control law to follow the detected road.

Table 1.1: Computation unit specifications

	OS	*CPU*	*Memory*
Micro-Com	Ubuntu	i586	-
NI PXI	Window7	Intel i5	4G
Laptop	Window7	Intel i5	2G

The detailed specifications related to the computation power of each computing unit are listed in TABLE 1.1. Note that there is no GPU (Graphic Processing Unit) involved in any of the image processing algorithms.

Chapter 2

Mono-Camera-Based Lane Line Detection and Vehicle Localization

2.1 Introduction and literature review

Vehicle localization with respect to road lanes is the fundamental function to be enabled in autonomous vehicles. Vision-based solution is a low-cost approach to address such problems. The vehicle position and orientation can be derived from the detection results. A vast amount of research work has been done in this domain since a few decades ago [18]. However, it is yet to be completely solved and has remained as a challenging problem due to the wide range of uncertainties in real traffic road conditions, which may include shadows from cars and trees, variation of lighting conditions, worn-out lane markings and other markings such as directional arrows, warning words and zebra crossings.

A survey on existing lane detection approaches is provided by Hillel AB et al. in [19]. Most of them share three common steps: 1) Lane line candidate extraction using different image features such as edges [20] [21] and color [22] [23], or using machine learning methods such as support vector machine [24], boost classification [25] [26]. 2) Model fitting to straight lines [27][28], parabolas [29][30] and hyperbolas [31][32][33]. 3) Vehicle pose estimation from the fitted model.

Some algorithms may also include another tracking step to impose temporal continuity, where the detection result in the current frame is used to guide the next search through filter mechanisms, such as Kalman filter [20] [34] and particle filter [24] [35].

A common disadvantage of a feature-based lane line candidate extraction is the high sensitivity to noise because it is mainly done through thresholding gradient magnitude. However, edges of shadows and surrounding objects also tend to have higher gradient values while poor lighting conditions result in lower gradient values at the lane line boundaries. Thus, gradient thresholding alone is not feasible without other complex adaptation mechanisms. Another disadvantage is that it is limited to a local view [36]. A feature point is detected with only two pixels in the image without considering its connectivity and similarity to the neighbouring pixels.

The machine learning-based approach is subject to the selection of training data and off line training results. To fulfil a good detection, the training set has to contain enough samples under a variety of scenarios. If new situations occur during the test, the approach may fail.

To resolve these issues, *A.López* in [37] introduced a newly defined feature 'ridge', which is claimed to be more suitable than other features (such as edge or color) to this problem. The noise pixels are then filtered based on ridge magnitude and ridge orientation. In model fitting, the author adopted RANSAC (RANdom Sample Consensus) to fit a pair of lane lines (right and left lane lines) simultaneously based on the given camera height and pitch angle.

However, the algorithm suffers from the following shortcomings:

(i) The noise filtering mechanism is not robust. The author assumed ridge orientation was always along lane line direction and he used this feature to filter out some pixels. But this is not true in the presence of shadows, uneven distribution of lane line painting, worn-out lane lines and etc.

(ii) The filtering based on a fixed ridgeness threshold is not suitable, which leads to under-filtering and over-filtering issues in some situations.

(iii) Due to the nature of the model fitting method in [37], when the lane exists on only one side, the algorithm will not work properly.

(iv) The fitted model accuracy is sensitive to the camera viewing angle. Braking/acceleration and non-flat road surface changes the pitch angel. To compensate the variation, the author assigned a series of discrete values within a certain range to pitch angle. But the result due to this variation is still clearly lingering especially when the actual pitch angle is outside the prescribed range.

(v) The hyperbola road model itself is not able to provide an accurate fitting for a straight road.

Motivated by these unresolved yet important shortcomings, a more robust approach is proposed in this chapter to solve the vehicle lane-level localization challenge. It is based on ridge identification and incorporates a modified sequential RANSAC algorithm in the model fitting process. The main contributions of the chapter include:

(i) Implementation of an effective noise filtering mechanism based on adaptive threshold after ridge detection, thus increasing the processing speed and improving the detection results.

(ii) Removal of the effects from camera pitch angle variation on model fitting. In the proposed approach, the model fitting does not rely on having the pitch angle fixed a priori, instead the pitch angle is back calculated from the fitted model.

(iii) Lane detection on one side of the road even if the other side is missing.

(iv) Incorporation of a modified version of sequential RANSAC algorithm in model fitting to capture every single lane line in the image independently.

(v) Fitting multiple road models simultaneously, including straight line and hyperbola, in the quest for the best matching results.

The chapter is organized as follows: Section 2.2 provides a brief introduction to the concept of a ridge. Section 2.3 will focus on the filtering of noise pixels after the ridgeness thresholding. The modified version of sequential RANSAC for model fitting is elaborated in Section 2.4. Section 2.5 will illustrate the experimental validation results and the conclusions are drawn in Section 2.6.

2.2 Ridgeness

The ridge of a grey-scale road image refers to the center axis of the elongated and bright lane lines. The concept can be visualized by considering the image as a landscape with intensity represented along the z axis or height [37]. The intensity increases as it gets closer to the center axis of lane lines, which forms the shape of a ridge as illustrated in Fig. 2.1. Moreover, *ridgeness* quantifies how well the pixel neighbourhood resembles a ridge. At the center axis of the lane line, its neighborhood at both sides contributes to the formation of the ridge; therefore it will have a higher ridgeness value. This observation can enable the detection on lane lines by a simple thresholding method. This is also one of the root factors explaining why ridgeness is a more robust feature than edge or color as it takes all its neighborhood pixels into account instead of just two.

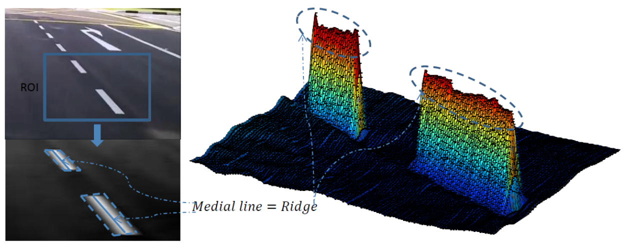


Figure 2.1: Concept visualization of ridge

Ridgeness is a loosely defined terminology and there exist different versions of mathematical definitions. In this chapter, we adopt the version of *A.López* in [37] since it has been proven to be largely effective.

First, the original grey-level image $L(x)$ is convoluted (*) by a 2D Gaussian filter G_{σ_d}. $L(x)$ is taken as the intensity value (I) in HSI (Hue, Saturation, Intensity) space because it has clear advantages than H or S or other spaces (e.g. RGB) as concluded in [23].

$$L_{\sigma_d}(\mathbf{x}) = G_{\sigma_d}(\mathbf{x}) * L(\mathbf{x}) \tag{2.1}$$

G_{σ_d} is anisotropic Gaussian kernel with covariance matrix $\sum = diag(\sigma_{dx}, \sigma_{dy})$ where σ_{dy} is constant (set as 1) and σ_{dx} increases with row number which equals to half of the lane line width. It depends on the camera focal length and pitch angle φ. In our setup, it varies from 1 to 12.

The gradient vector field at each pixel along row (u) and column (v) is:

$$\mathbf{w}_{\sigma_d}(\mathbf{x}) = (\partial_u L_{\sigma_d}(\mathbf{x}), \partial_v L_{\sigma_d}(\mathbf{x}))^T \tag{2.2}$$

A 2×2 matrix $s_{\sigma_d}(\mathbf{x})$, similar to Hessian matrix, is computed by dot product $(\cdot)$ of gradient vector for each pixel:

$$s_{\sigma_d}(\mathbf{x}) = \mathbf{w}_{\sigma_d}(\mathbf{x}) \cdot \mathbf{w}^T_{\sigma_d}(\mathbf{x}) \tag{2.3}$$

The structure tensor field $S_{\sigma_d \sigma_i}(\mathbf{x})$ is computed by convoluting each $s_{\sigma_d}(\mathbf{x})$ matrix with another Gaussian filter G_{σ_i} (σ_i is set as 1):

$$S_{\sigma_d \sigma_i}(\mathbf{x}) = G_{\sigma_i}(\mathbf{x}) * s_{\sigma_d}(\mathbf{x}) \tag{2.4}$$

The eigenvector $\mathbf{w}'_{\sigma_d \sigma_i}(\mathbf{x})$ is obtained corresponding to the highest eigenvalue of $S_{\sigma_d \sigma_i}$. The projection of $\mathbf{w}'_{\sigma_d \sigma_i}(\mathbf{x})$ onto $\mathbf{w}_{\sigma_d}(\mathbf{x})$ is defined as:

$$p_{\sigma_d \sigma_i}(\mathbf{x}) = \mathbf{w}'_{\sigma_d \sigma_i}{}^T(\mathbf{x}) \cdot \mathbf{w}_{\sigma_d}(\mathbf{x}) \tag{2.5}$$

A new vector field $\widetilde{\mathbf{w}}_{\sigma_d \sigma_i}(\mathbf{x})$ is defined below. It is along $\mathbf{w}'_{\sigma_d \sigma_i}(\mathbf{x})$ but its direction is determined by the sign of $p_{\sigma_d \sigma_i}(\mathbf{x})$.

$$\widetilde{\mathbf{w}}_{\sigma_d \sigma_i}(\mathbf{x}) = sign(p_{\sigma_d \sigma_i}(\mathbf{x}))\mathbf{w}'_{\sigma_d \sigma_i}{}^T(\mathbf{x}) \tag{2.6}$$

The ridgeness is then defined by the positive value of divergence of $\widetilde{\mathbf{w}}_{\sigma_d \sigma_i}(\mathbf{x})$.

$$\widetilde{k}_{\sigma_d \sigma_i}(\mathbf{x}) = -div(\widetilde{\mathbf{w}}_{\sigma_d \sigma_i}(\mathbf{x})) \tag{2.7}$$

Fig. 2.2 illustrates the $\widetilde{\mathbf{w}}_{\sigma_d\sigma_i}(\mathbf{x})$ field orientation of the ROI (region of interest) at the original image. The green box highlights the corresponding lane line boundaries. It shows that due to the tree shadows, $\widetilde{\mathbf{w}}_{\sigma_d\sigma_i}(\mathbf{x})$ orientation deviates from lane line direction, especially for the pixels along the lane line medial axis.

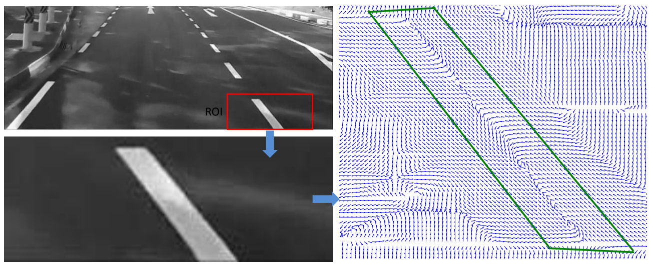


Figure 2.2: Illustration on ridge orientation $\widetilde{\mathbf{w}}_{\sigma_d\sigma_i}(\mathbf{x})$

The detailed dimensions of the variables are shown in TABLE 2.1.

Table 2.1: Ridgeness variable dimensions

L_{σ_d}	image size	$\mathbf{w}_{\sigma_d}(\mathbf{x})$	2x1	$s_{\sigma_d}(\mathbf{x})$	2x2	$S_{\sigma_d\sigma_i}(\mathbf{x})$	2x2
G_{σ_i}	3x3	$p_{\sigma_d\sigma_i}(\mathbf{x})$	scalar	$\widetilde{\mathbf{w}}_{\sigma_d\sigma_i}(\mathbf{x})$	2x1	$\widetilde{k}_{\sigma_d\sigma_i}(\mathbf{x})$	scalar
G_{σ_d}	3xn, n varies from 3 to 25 with increasing row number						

Fig. 2.3 provides an example of the grey-scale image based on ridgeness value. It is clear that the medial axis of the lane line is brighter than the rest, which enables the following processing algorithms.

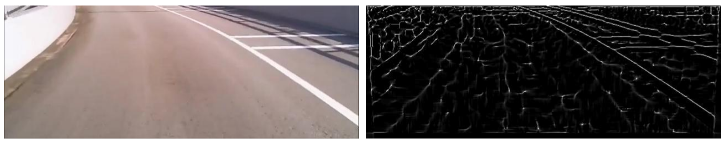


Figure 2.3: Original image and grey-scale image based on ridgeness value

2.3 Noise filtering mechanism

As mentioned previously, the original approach in removing noise pixels is not robust and effective since a fixed threshold is implemented and $\widetilde{\mathbf{w}}_{\sigma_d\sigma_i}(\mathbf{x})$ is not always along lane line direction. Here, we propose an adaptive thresholding mechanism based on ridgeness value, which avoids checking $\widetilde{\mathbf{w}}_{\sigma_d\sigma_i}(\mathbf{x})$ direction.

Based on the fact that lane line medial axis always has higher ridgeness value, it can be selected by thresholding. To have an adaptive threshold, the ridgeness histogram is used.

The image size is 480×640, but the image processing takes effect only on the bottom half. Because based on the camera setup, only the bottom half contains useful information for lane line detection while the top half mainly consists of sky and road portions that are too far to see. The number of pixels consisting of the longest lane medial axis for one line must be less than $240 + 640 = 880$. Since lane lines exist on both sides and sometimes, double lane lines may be used on both sides, the number of pixels for lane medal axis must be less than $4 \times 880 = 3520$. Therefore, 3520 can be a conservative estimation of the maximum number of lane line medial axis pixels. In other words, the number of pixels resulting from lane line detection must be less than 3520.

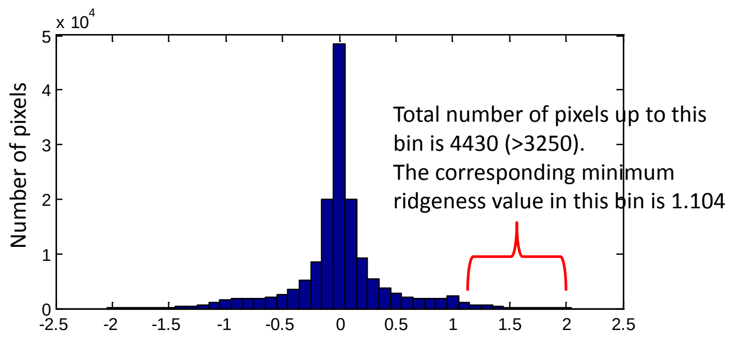
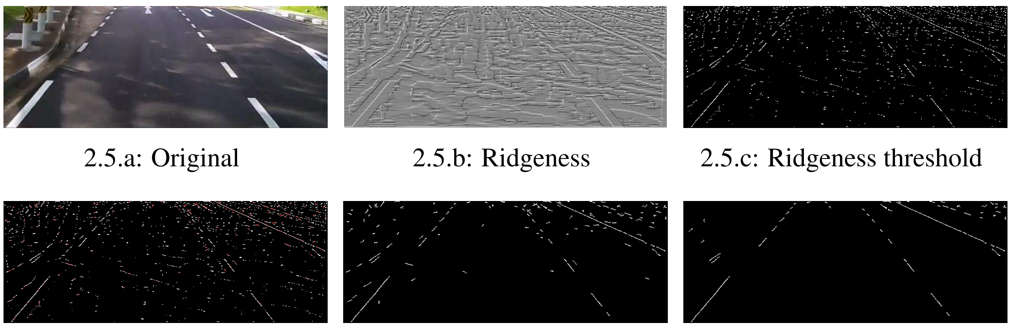


Figure 2.4: Histogram based on the ridgeness image

After the ridgeness calculation, the histogram of the ridgeness grey-scale image can be extracted, varying from -2 to 2 with a bin size of 0.1. Each bin is summed in descending order until the number of pixels exceeds 3520. The corresponding minimum ridgeness value in that bin will be set as the threshold. Fig. 2.4 shows the histogram plot based on Fig. 2.3(right).

After thresholding, as shown in Fig. 2.5(c), almost all the lane line medial axis pixels are extracted. But some are disconnected and a lot of noise pixels still exist, resulting from shadows and irregularities on the pavements.

To compensate for these, first a morphographic 'bridge' operation to connect pixels with gap of one pixel is applied then followed by a connected component labelling operation. The corresponding component is removed if its number of pixels is less than a prescribed threshold number. This threshold can be worked out based on the minimum number of pixels required to form a lane line segment medial axis. For breaking lane lines, according to our on-field measurement, the shortest segment is about 1 meter. Based on the camera nominal pitch angle, its intrinsic parameters and assuming a segment at the furthest distance in the camera view (bottom half), the minimum number can be calculated approximately. In our setup, the threshold number is 6.


2.5.a: Original 2.5.b: Ridgeness 2.5.c: Ridgeness threshold

2.5.d: Bridge connection 2.5.e: Min. structure removal 2.5.f: Intensity check

Figure 2.5: Proposed noise filtering mechanism

Components are further removed if the average intensity is less than a threshold as highlighted in [23]. The threshold is set conservatively low to cater for the case when lane lines are obscured by strong shadows or the illumination condition is poor. The number implemented here is 150.

Fig. 2.5 illustrates how the lane line candidate pixels are selected from the original image systematically using the proposed noise filtering mechanism. The red pixels in Fig. 2.5(d) indicate the effects from bridge connection. Both quantitative and qualitative comparisons between the original approach in [37] and our proposed one are presented in Section 2.5. They show that the proposed approach can outperform the original one most of the time.

2.4 Model fitting with modified sequential RANSAC

2.4.1 Lane line model construction

Many lane line models have been proposed in the literature, ranging from s-traight lines to spline and conical curves. There is no clear conclusion on which one is the best. However intuitively, assuming the road surface to be flat and lane lines to be parallel, there should exist an explicit relationship between the real road lane line geometry and the projected lane line in the image subject to the camera position and orientation.

Leveraging on this consideration, we adopt the road model proposed by Guiducci [38]. The simplified version provided in [37] and [39] is shown in (2.8) and (2.9) for the left and right lane respectively.

$$u_l = E_u\left(\frac{\theta_e}{\cos\varphi} + \frac{\cos\varphi}{HE_v}d_e(v_l + E_v\tan\varphi) + \frac{E_vHC_0/\cos^3\varphi}{4(v_l+E_v\tan\varphi)}\right) \tag{2.8}$$

$$u_r = E_u\left(\frac{\theta_e}{\cos\varphi} - \frac{\cos\varphi}{HE_v}d_r(v_r + E_v\tan\varphi) + \frac{E_vHC_0/\cos^3\varphi}{4(v_r+E_v\tan\varphi)}\right) \tag{2.9}$$

To fully understand the model, let's define three coordinate systems attached to global (or road), car and camera respectively, with the same origin at the camera principal point. θ_e and φ are camera yaw and pitch angle with respect to global system.

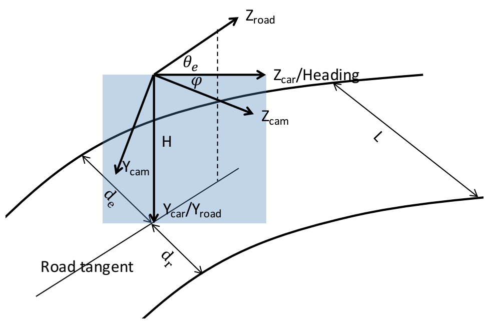


Figure 2.6: Road model parameters illustration

(u, v) defines the horizontal and vertical pixel count of one pixel to the image principal point. E_u and E_v refer to the camera horizontal and vertical focal lengths in the unit of $pixel/meter$. All these parameters ($principal\ point$, E_u and E_v) are camera dependant and can be obtained uniquely through calibration. Interested readers may refer to Bouguet's toolbox on the calibration [40].

φ is the camera pitch angle. Ideally, this is a fixed value and can be measured in advance as well. However, in reality, it varies when the car is slowed/stopped with the brake, accelerating, running on uneven road and etc. The value is so critical to the final fitting result that it cannot be treated as a constant. To compensate for the variation, in [37], the author assigned a series of discrete values within a certain range. However, this approach is sensitive to quantization noises. In this chapter, we will consider φ as an unknown and show that it can be back-estimated instead accurately.

H is the height of the camera to the road surface, measured in advance. θ_e defines the angle between car heading direction and road tangent. C_0 is the lateral curvature of real road with unit of m^{-1}. If the road is straight, $C_0 = 0$ and (2.8) and (2.9) represent a line. d_e and d_r represent the distances from the car to the left and right road lane lines. $L = d_e + d_r$ is the lane width.

For each side of the road model, there are four unknown parameters to be determined through model fitting, namely φ, θ_e, C_0 and $d_e(or\ d_r)$. Although there are parallelism relationships between left and right lanes, we will ignore this relationship and take them to be independent for the first pass model fitting and only merge/fuse the results based on this relationship in a latter process when both left and right lanes are confirmed to exist. This is to cater for the case when the lane line only exists on one side.

$$u = A/(v-D) + Bv + C \tag{2.10}$$

To facilitate the model fitting process, the road model can be further simplified to the form of (2.10) with A, B, C and D as unknowns.

The matrix form is shown in (2.11), where Cr is symmetrical, $E = C - BD$, $F = A - CD$, $(Cr \cdot P)$ represents the tangent of hyperbola at point P.

$$P^T \cdot Cr \cdot P = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}^T \begin{bmatrix} 0 & -0.5 & 0.5D \\ -0.5 & B & 0.5E \\ 0.5D & 0.5E & F \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0 \tag{2.11}$$

2.4.2 Model fitting

As mentioned, we will fit model independently for left and right lanes. Therefore, the image is deliberately separated into right and left parts. For simplicity, we choose the vertical center axis of the image as the separation line. If the connected component is across the center axis, it will be classified according to the number of pixels on each side. If it has more pixels on the left, then it belongs to the left part and vice versa.

To determine one model, four pixels are required. However, most of the time, there will be much more than four candidate pixels. Moreover, some are outliers and multiple models may exist. For such a multiple model fitting problem with outliers, there are several popular techniques in the literature, such as sequential RANSAC, multiRANSAC, residual histogram analysis, J-linkage, k-ernel fitting and energy minimization PEARL. As concluded by Fouhey [41] in his study, sequential RANSAC is a strong first choice due to its effectiveness.

As the name implies, sequential RANSAC implements the RANSAC algorithm a number of times until all models are found or a certain number of iterations has been reached. Once one model is determined, all its supporting data points will be eliminated from the data set and RANSAC is run through the remaining data points to look for another model. However, when the previous model is not correctly selected, the subsequent models will be affected as some of the data points are eliminated wrongly. This becomes more often when double lane lines are used at road bending. Fig. 2.7 illustrates examples of inaccurately fitted model (highlighted as red) using sequential RANSAC.

To resolve this critical issue and achieve better lane line detection results, we propose the following modified sequential RANSAC algorithm by adding a fusion step:

(i) Randomly select four points from data set

(ii) Create the model and find all its supporting data points

(iii) Fuse the new model with previous models. If there are common data points in the new model and the previous ones, only the model with more data points will be kept while the other is discharged. If no common data points exist, the model will be taken as new. This is valid because there should not be any intersections between lane lines.

(iv) Repeat step i) to iii) for a certain number of iterations. No data is eliminated as we do not want this model to affect the consequent fitting results.

(v) Till this step, we have obtained models without any intersection. To expedite the process, eliminate all the supporting points to these models and repeat step i) to iii) with the remaining data for a certain number of times.

For example in Fig 2.7(a), in one iteration, the inadequate model represented by the red pixels is identified. In another subsequent iteration, the accurate model represented by the inner line is also identified. Then it will be fused with the inadequate model since they share some data points (the red points at the top of the image). Only the accurate model is kept and the inadequate model is discharged because it has less data points.

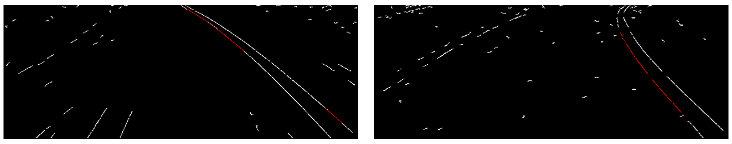

2.7.a: Inaccurate Fitting 1 2.7.b: Inaccurate Fitting 2

Figure 2.7: Inaccurate fitting (red pixels) results from conventional sequential RANSAC

But if using conventional sequential RANSAC, once the inadequate model is identified before the accurate model, all the red pixels will be removed from the data pool. Almost half of the inner line data points are removed wrongly, even the accurate model can be identified later, it won't be kept if the number of remaining inner line data points is less than that of red data points.

The results from this modified sequential RANSAC are several non-intersecting models from both sides. The next step is to pair up models and select the best one from all the possible pairs.

Before explaining the pairing mechanism, we would like to elaborate more on step ii), which is the core of RANSAC. Although any four arbitrary points (not on the same line) can generate one unique hyperbola, not every result can describe the road properly. Before searching for supporting data points in step ii), we need to validate the model first. The explicit form of A, B, C and D is shown in (2.12)-(2.15).

The first validation is on D. Its value varies within the range determined by the physical limits of φ. The change of φ is a direct result from the car suspension systems. When its front suspension system is fully compressed and the rear one is fully released, the camera points most downwards and φ is maximum. When the front suspension system is fully released and the rear one is fully compressed, the camera points least downwards or even upwards and φ is minimum. The nominal pitch angle φ_n can be calibrated when the car is not moving and not loaded. By referring to the vehicle catalogue, we can calculate the maximum angle that the car frame can be tilted, which corresponds to the two extreme conditions. For the testing vehicle used, the maximum angle is $\sim 3^\circ$ or $\sim 0.035 rad$. Therefore, the range of φ is approximated as $[\varphi_n - 0.035,\ \varphi_n + 0.035]$.

$$A = C_0 H E_u E_v / (4\cos^3(\varphi)) \tag{2.12}$$

$$B = d_e E_u \cos(\varphi)/(H E_v) \tag{2.13}$$

$$C = \theta_e E_u / \cos(\varphi) + d_e E_u \sin(\varphi)/H \tag{2.14}$$

$$D = -tan(\varphi) E_v \tag{2.15}$$

With the valid D, A is further validated based on the road curvature. The absolute value of A has to be less than the value derived from the maximum allowable $|C_0|$ for a feasible and safe turning. For a normal sedan, the minimum feasible turning radius is $\sim 10m$ limited by its steering angle. Therefore, in this chapter, the maximum allowable $|C_0|$ is set as $0.1m^{-1}$.

B and C are related to the vehicle pose. Since the vehicle can be at any locations on the road, they should not be constrained. In summary, we have defined a subclass of valid hyperbolas based on camera pitch angle limits and road curvature limit.

If it is not able to pass the validation on D or A, the next iteration will begin. Some weird fitting results (highlighted in black) without adding these constraints are shown in Fig. 2.8.

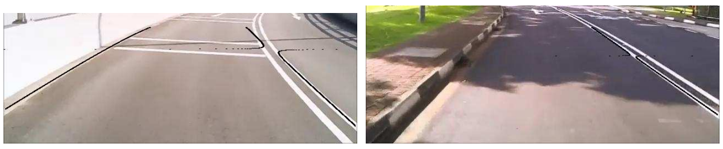


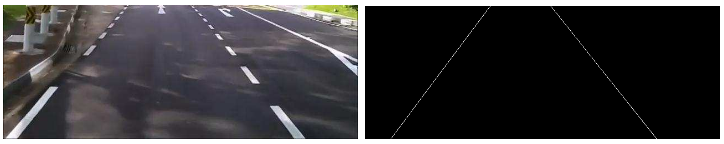
Figure 2.8: Wrong fitting results without constraint on hyperbola center

As mentioned, two base models, hyperbola and straight line, will be fitted simultaneously. This is done in step ii) as well. The four randomly selected points can generate one unique hyperbola and six lines. Out of these seven models, only the valid one with most supporting points will be kept. Supporting data points will be determined purely based on Sampson's distance d_s [42]. If its $d_s < a\ threshold$, the point P will be classified as a supporting point.

$$d_s = \frac{(P^T \cdot Cr \cdot P)^2}{4((Cr \cdot P)_1^2 + (Cr \cdot P)_2^2)} \tag{2.16}$$

where Cr is defined in (2.11) and $(Cr \cdot P)_n$ refers to the nth element of the vector. For straight lines, $A = 0$.

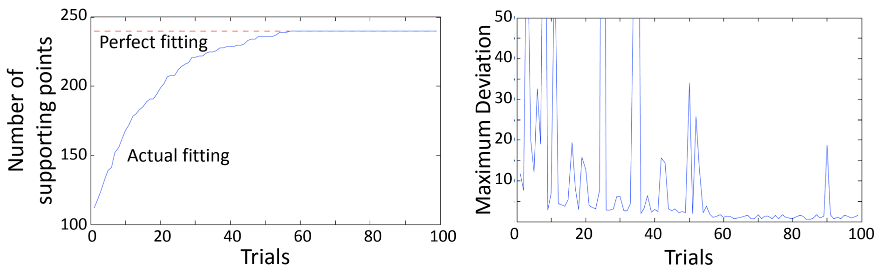
The straight line model is a necessary complement to the hyperbola model in fitting a straight road. Firstly, it increases the successful rate in finding a valid model. Fig. 2.9 illustrates a perfect lane line extraction for the straight road with



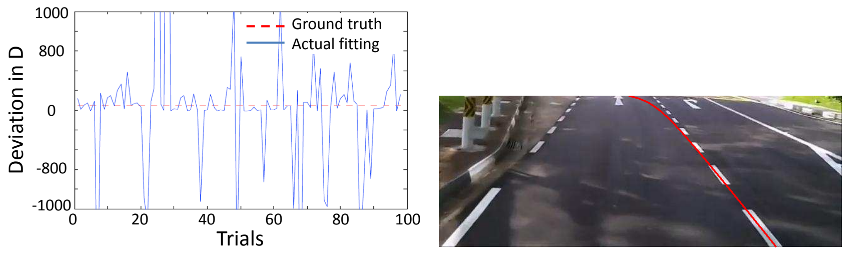
2.9.a: Original 2.9.b: Perfect line extraction

Figure 2.9: Perfect model for a straight road

width of 3.4m. By randomly selecting 4 points on one side, only about 9% of the 1000 trials are able to provide a valid hyperbola model. It indicates that the rate of fitting is very low by using hyperbola only even under the perfect situation, not to mention the situations when there exist noise pixels from the lane line extraction. The original approach [37] suffers the same issue.



2.10.a: Supporting points 2.10.b: Row-wise maximum deviation



2.10.c: Derived D from fitted models 2.10.d: Example of row-wise deviation

Figure 2.10: Hyperbola model accuracy for a straight line

Secondly, it increases the accuracy of the model. Fig. 2.10(a) plots the number of supporting points for each of the successful fitted models in ascending order. Fig. 2.10(b) illustrates their corresponding row-wise maximum deviations. These two figures indicate that out of the 9% successful trials, only half provide acceptable results in terms of number of supporting points and maximum deviation. The maximum deviation mainly occurs at the top part of the image as shown in Fig. 2.10(d).

Even among those successful trials with small row-wise maximum deviations (from trial 60 onwards), the derived parameters from fitted models are not consistent. Fig. 2.10(c) plots the D value derived from the corresponding trials. Among the successful trials, it varies from -1000 to 1000 while the ground truth value is 42.8. As shown in (2.18)-(2.22), all the localization information is directly related to D. If this value is not accurate enough, then the whole set of information becomes inaccurate.

The numbers presented above may vary from trial to trial and image to image, but it provides insights on how a straight line model helps in fitting a straight road and increasing the accuracy.

2.4.3 Pairing and parallelism reinforcement

In the case that lane lines on both sides exist, a pairing step is implemented right after all single models are captured. The best pair is determined based on the number of supporting data points and whether the lane width estimated from the model is within a typical road lane width ($\sim 3.4m$).

Equation (2.8)-(2.9) will give the solution for lane width L. However, most of the time, the results from independent fitting will not follow the parallelism relationship exactly. In turn, we will get a non-constant L. To compensate this and to get the model as accurate as possible, the two paired models need to be fine tuned and adjusted according to the parallelism relationship defined by (2.8) and (2.9). This fine-tuning process is called $parallelism\ reinforcement$.

It can be derived from (2.8) and (2.9) that if one lane model follows (2.10), then the other lane model will be

$$u = A/(v-D) + B'v + C + D(B-B') \tag{2.17}$$

Equations (2.10) and (2.17) indicate that to fine tune a pair of lane models, five points are required to determine the five unknowns (A, B, B', C and D). However, the pair of lane models is derived from eight points in model fitting step (four for each side). To balance the contribution from both sides, three out of the four points from one side and two out of the other four points from the other side will be chosen. In total, there will be 48 combinations to fine tune this pair of lane lines. Out of these 48 combinations, the one with the most supporting data points will be the final model for this pair.

Just to highlight, the five unknowns cannot be solved in the form of linear algebras. The set of equations consists of high order polynomials with multiple variables. Fortunately after tedious conversions and substitutions, it can be downgraded to one third order polynomial equation with variable D only. Close form solutions are available in the literature to get the three roots explicitly [43]. The real root with the most supporting points will be selected.

After finalizing all pair models, the one which has the most supporting points with L within the prescribed range will be used to describe the detected lane. The corresponding localization information can be calculated as following:

$$\varphi = \arctan(-D/E_v) \tag{2.18}$$

$$d_e \ or \ d_r = BHE_v/(E_u \cos\varphi) \tag{2.19}$$

$$C_0 = 4A\cos^3\varphi/(HE_uE_v) \tag{2.20}$$

$$\theta_e = (C - E_u d_e \sin\varphi/H)\cos\varphi/E_u \tag{2.21}$$

$$L = d_e + d_r \tag{2.22}$$

Note that even the lane line exists on one side only, the localization information is still able to be calculated using the same formula above. The only missing information is lane width L.

2.5 Experiment validation and results

The following tests were carried out on the NI PXI shown in TABLE 1.1. The algorithm was programmed in MATLAB with mex-C functions and did not optimize to run parallel threads. The average processing time for one image is

$0.12s$, of which 20% is for ridgeness calculation, 17% for noise filtering, 57% for model fitting and 6% for pairing and parallelism reinforcement. The processing speed is sufficient for a vehicle moving at normal speed ($\sim 70km/hr$). Because the look-ahead distance in the image is approximately $30m$, but the vehicle only moves $2 \sim 3m$ during one image processing time.

2.5.1 Noise filtering

The filter results in Section 2.3 using the proposed method and the original method are compared both qualitatively and quantitatively. For the quantitative comparison, only 350 images from 3 video clips are used due to the difficulties in obtaining the ground truth. In each video, the images are sampled consecutively at $10Hz$. The 3 video clips contain different challenging scenarios, such as breaking lines, dense traffic and worn-off lane lines. The detailed challenges or noise sources are tabulated in TABLE 2.2.

Table 2.2: Noise sources contained in each video clip

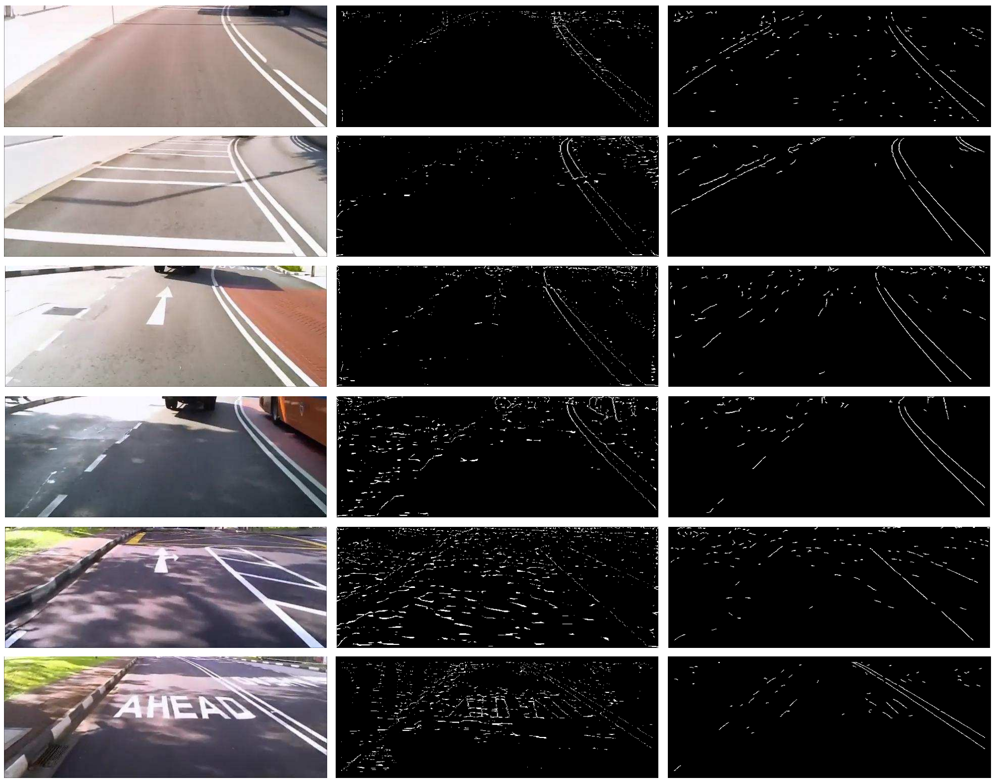
Video	Noise Sources
1	Other markings, horizontal speed regulation lines
2	Other markings, tree/car shadow, breaking lines, dense traffic, worn-off markings
3	Other markings, tree/car shadow, breaking lines, horizontal speed regulation lines

Two images from each video clip are shown in Fig. 2.11 as a qualitative comparison. Column (b) contains the results from the original approach [37] and Column (c) contains results from the proposed one.

The first row shows an example on simple images in which few challenging scenarios exist. Both approaches are able to identify the right double while lines. For the left boundary, it is debatable whether it can be treated as lane marking or not. The original approach cannot identify it but the proposed one can.

The second row indicates that both approaches are able to remove the noise from the horizontal speed regulation strip.

The third row depicts the results on worn-off lane markings (left lane line). The proposed approach is able to identify the three left lane line segments count-



2.11.a: Original images 2.11.b: Approach in [37] 2.11.c: Proposed approach

Figure 2.11: Noise filtering mechanism comparison

ing from the image bottom but the original one almost fails.

In the fourth row, the nearby vehicle leads to false detection in the original approach as shown on the right top corner of the image. The non-uniform road color on left side of the image leads to more false detections in the original approach.

The last two rows illustrate the performances when there are strong tree shadows and letters in the image. The original approach is not able to remove the noise pixels effectively.

From the qualitative comparison, we can conclude that the proposed approach performs similarly as the original approach on simple road scenarios (e.g. speed regulation strips). But when the road surface becomes more erratic, non-uniform and complex (e.g. tree shadows), the proposed approach is more capable of removing noise pixels.

In the quantitative comparison, the ground truth is labelled manually according to the 350 images. One example is shown in Fig. 2.12(c). Three ratios, commonly used in ROC (Receiver operating characteristic) curve, are introduced for the quantitative evaluation.

$$True\ positive\ rate\ (TPR) = TP/P \tag{2.23}$$

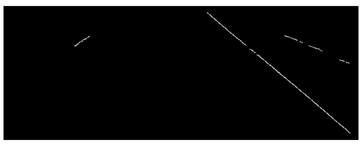
$$False\ positive\ rate\ (FPR) = FP/N \tag{2.24}$$

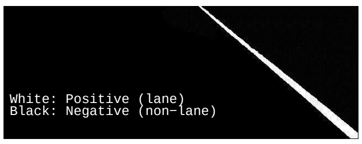
$$Accuracy\ (ACC) = (TP+TN)/(P+N) \tag{2.25}$$

where TP is the number of lane pixels labelled correctly as lane line (true positive); FP is non-lane pixels erroneously labelled as lane line (false positive); TN is the non-lane pixels correctly labelled as non-lane line (true negative); P and N are the number of lane line and non-lane line pixels.

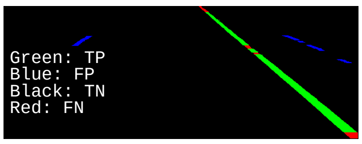
A lower FPR and a higher TPR indicate better detection results as it is closer to the perfect result with $FPR = 0$ and $TPR = 1$.

2.12.a: Original image


2.12.b: Ridge detector



2.12.c: Ground truth

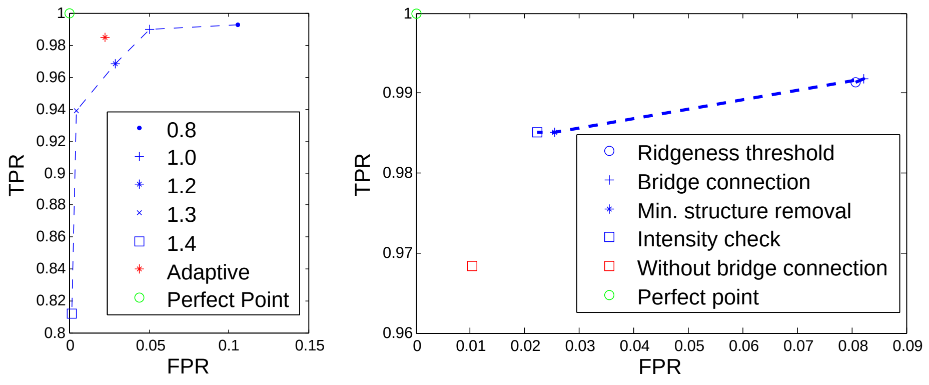


2.12.d: ROC

Figure 2.12: Illustration on ROC counting (TP: True Positive, FP: False Positive, TN: True Negative, FN: False Negative)

Note the way of counting TP, FP and TN. Due to the fact that the result from ridge detector is the lane line medial axis instead of the whole lane area, so if one pixel after the ridge detector is classified as TP, all its connected pixels forming the width of the lane on the same row in the ground truth image will be counted as TP. If one pixel is classified as FP in the ridge image, it will be expanded along row direction first. The width of expansion equals to 2 times of its corresponding σ_{dx} (defined in (2.1)). For example, if the FP pixel is at the last row and its corresponding σ_{dx} is 12, then the number of FP pixels is approximated to be $2 \times 12 = 24$. TN number equals to the number of N minus FP pixels. Fig. 2.12 illustrates this process in details. In Fig. 2.12(d), Green represents TP pixels, Blue for FP, Black for TN and Red for FN.

As mentioned before, the first filtering step is an adaptive threshold on ridgeness values. To evaluate its performance, we compare its TPR and FPR values with those resulted from a series of fixed threshold as shown in Fig. 2.13.a. The blue dots are obtained by setting the ridgeness threshold to the corresponding fixed values and keeping the remaining filtering steps the same.



2.13.a: 2.13.b:

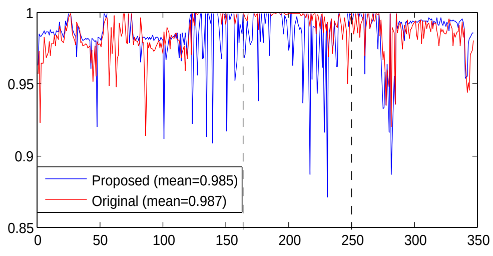
Figure 2.13: a) Average TPR and FPR comparison between ridgeness adaptive threshold and fixed-value threshold. b) Average TPR and FPR after each filtering step.

The figure shows that the resulted point defined by $(FPR,\ TPR)$ is closer to the perfect point $(0,\ 1)$ when using the proposed adaptive threshold. This indicates that adaptive threshold is able to improve the noise filtering performance as compared to fixed-value threshold.

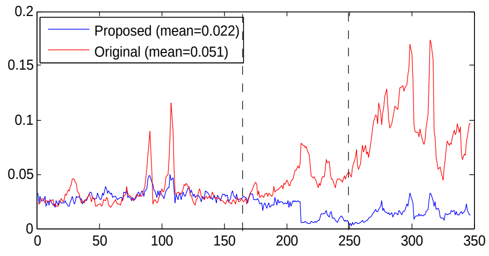
To further analyze the impact from each individual filtering step, after running each step, the corresponding ROC values of each image are logged and the average FPR and TPR values over the 350 images after each step are plotted in Fig. 2.13.b (blue indicators).

As can be seen from the figure, the minimum structure removal has the greatest impact in reducing the FPR, which means the noise pixels are largely removed by this step. The intensity check step can further improve FPR without sacrificing TPR.

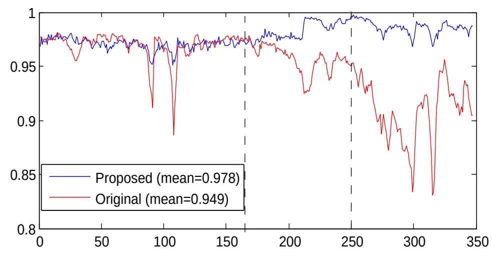
The bridge connection step seems to have little impact on FPR and TPR. But it is necessary. To verify this, another filtering test on the same set of images is carried out by removing this step but keeping the rest unchanged. The resulted (FPR, TPR) is shown as the red square in Fig. 2.13.b. The FPR is further reduced as compared to the blue square, but the TPR is reduced more significantly, resulting in a longer distance to the perfect point. The bridge connection step prevents some true positive points from being removed by the minimum structure removal step.



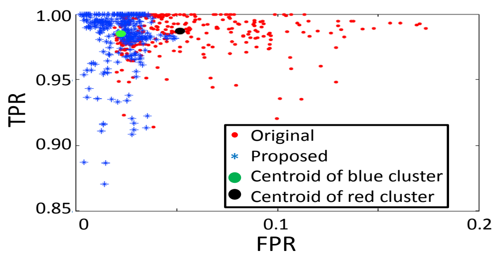
2.14.a: TPR (True Positive Rate)



2.14.b: FPR (False Positive Rate)



2.14.c: ACC (Accuracy)



2.14.d: TPR vs. FPR distribution

Figure 2.14: ROC comparison between the proposed and original algorithms

To further evaluate the performance of the proposed noise filtering mechanism, we also compare its ROC values with those obtained from the original approach [37]. Fig. 2.14(a)-(c) illustrates the comparison results frame by frame, where the black dashed lines mark the separation of different video clips.

For the first video, both approaches achieve similar performances as the road conditions are relatively simple. But when more challenging scenarios (e.g. tree shadows) come into the image as shown in the second and third video, the proposed algorithm is still able to maintain a low FPR as compared to the original approach, yet without scarifying TPR too much. This trade-off strategy yields higher accuracy as illustrated in Fig. 2.14(c) and improves the model fitting efficiency significantly as shown later in this section. In other words, the proposed algorithm is able to remove the noise pixels more effectively. This observation is similar to that from the quantitative comparison.

In overall, both methods achieve a similar TPR (the proposed is slightly worse by 0.2% only), but the proposed method has much better performance in FPR (lower by 3%) and ACC (better by 3%).

Fig. 2.14(d) presents the $(FPR,\ TPR)$ distribution for each image. It is clear that the result cluster based on the proposed method is closer to the perfect point $(0,1)$ than that from the original method.

To further analyze the impact of the noise filtering mechanism on the model fitting process, define the following variables: ω is the probability of choosing an inlier (TP) each time a single point is selected from the lane line pixel candidates ($TP+FP$), p is the probability that the RANSAC process produces a useful result, n is the number of points needed to estimate model parameters and k is the number of iterations for the RANSAC process. The following relationships hold:

$$\omega = TP/(TP+FP) \tag{2.26}$$

$$1-p = (1-\omega^n)^k \tag{2.27}$$

Let the subscript $_p$ refer to the proposed method and o refer to the original method, based on (2.23-2.25) and (2.26-2.27), we can derive

$$\frac{k_p}{k_o}\cdot\frac{\lg(1-p_o)}{lg(1-p_p)} = \frac{\lg(1-[\frac{TPR_o\cdot P}{TPR_o\cdot P+FPR_o\cdot N}]^{n_o})}{\lg(1-[\frac{TPR_p\cdot P}{TPR_p\cdot P+FPR_p\cdot N}]^{n_p})} \tag{2.28}$$

To facilitate the analysis, we will use average values based on the 350 images to substitute to the right hand side of (2.28). If assuming hyperbola, we can have

$$\frac{k_p}{k_o}\cdot\frac{\lg(1-p_o)}{lg(1-p_p)} = 0.1382 \tag{2.29}$$

where on average, $P = 3672$, $N = 149928$, $TPR_o = 0.987$, $FPR_o = 0.051$, $TPR_p = 0.985$, $FPR_p = 0.022$, $n_o = n_p = 4$.

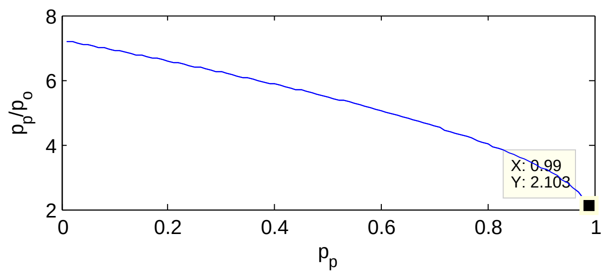


Figure 2.15: RANSAC probability analysis of the proposed (p_p) and the original (p_o) approach when running same number of iterations

From (2.29), statistically, to achieve the same probability p from RANSAC fitting ($p_o = p_p$), the number of iterations required in the proposed approach is only 13.8% of the original one. Furthermore, if running the same number of iterations ($k_p = k_o$), the probability of the proposed approach to get a useful model is always higher than that of the original one as shown from Fig. 2.15, where x-axis is p_p, varying from 0.01 to 0.99 and y-axis is p_p/p_o. When the proposed approach has the probability of 99% to find a useful model, the original approach only has 47%.

From the ROC analysis above, we can conclude that without sacrificing the true positive detection rate by much (worse by 0.2% only), the proposed filtering mechanism removes the noise pixels much more effective than the original one and thus it improves the detection accuracy and RANSAC fitting efficiency.

2.5.2 Modified sequential RANSAC

To evaluate the modified RANSAC algorithm, it is compared with the conventional one based on the first video clip. The first half of the video contains single line marking and the second contains double-line marking.

The fitting results from both algorithms were then compared with the ground truth. Their row-wise maximum pixel deviation from the ground truth are depict frame by frame in Fig. 2.16.

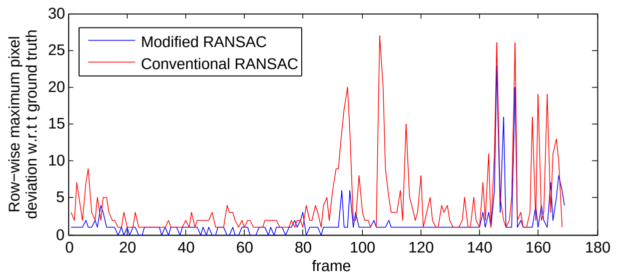
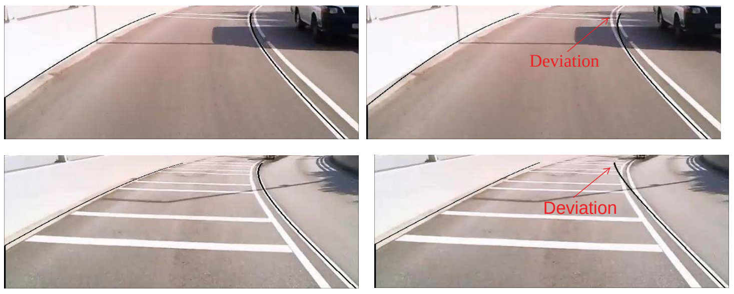


Figure 2.16: Line fitting comparison between modified and conventional RANSAC in terms of maximum deviations



2.17.a: Modified 2.17.b: Conventional

Figure 2.17: Comparison of fitting results between modified (column a) and conventional (column b) RANSAC

For the first half of the video, both algorithms perform similarly in fitting single line markings. But for the second half, the modified RANSAC algorithm obtains more accurate results when fitting double line markings. The average maximum deviation is only 2.26 pixels for the modified algorithm and 5.63 pixels for the conventional one. It is more often that the conventional algorithm results in large fitting deviations. Two examples are provided in Fig. 2.17. The major deviations from the conventional algorithm occur at the top of the images.

2.5.3 Simulation

As aforementioned, the main purpose of this mono-camera system is to generate vehicle localization information with respect to the road lane. This information can be calculated from the fitted model based on equations (2.18)-(2.22).

To verify the model fitting accuracy, we first tested the algorithm on the simulator proposed in [37]. The simulator simulates a sequence of road images (resolution 480×640) captured by an on-board camera when changing its ori-

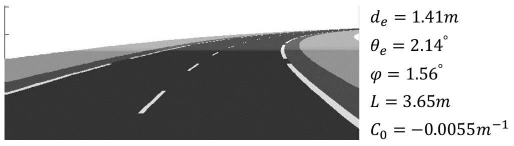


Figure 2.18: Simulated road with corresponding parameters

entation (θ_e and *pitch angle*), position (d_e) and road geometry (L and C_0). The total number of images is 2000. Fig. 2.18 is one of the images generated by the simulator. The corresponding parameters are given beside the figure. The proposed approach is tested with the simulated road images and all the unknown parameters for each image are back estimated from the fitted model. At the end, the estimated parameters are compared with the corresponding ones set up in the simulator to evaluate the accuracy.

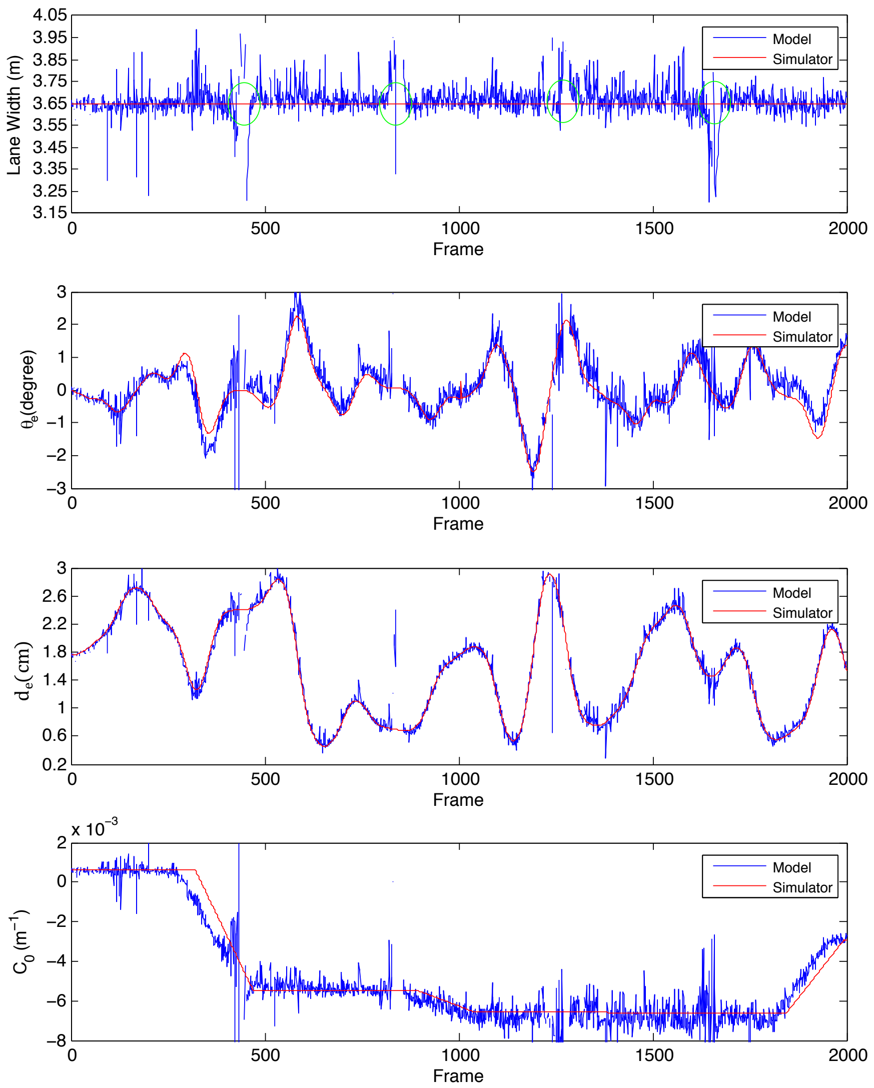


Figure 2.19: Model accuracy evaluation

Fig. 2.19 illustrates the comparison results of L, θ_e, d_e and C_0 between the estimated model value (*blue*) and the simulator ground truth (*red*). Most of the time, all the estimated values are quite close to their true values. The most prominent errors occur at the locations where road surface is not flat (highlighted by green circles), which violates the assumptions on model construction in Section 2.4.1. The lane line markings in the image don't follow hyperbolas or straight lines. Fig. 2.20 shows two fitting examples at location 417 and 827, representing down-hill and up-hill situations respectively.

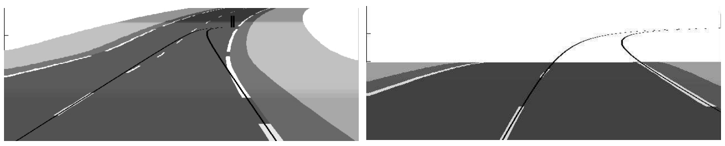


Figure 2.20: Inaccurate fitting examples at non-flat road surface

The absolute error distributions for the corresponding parameters are shown in Fig. 2.21. Most of the time ($\sim 90\%$), the absolute error for all the parameters is very small.

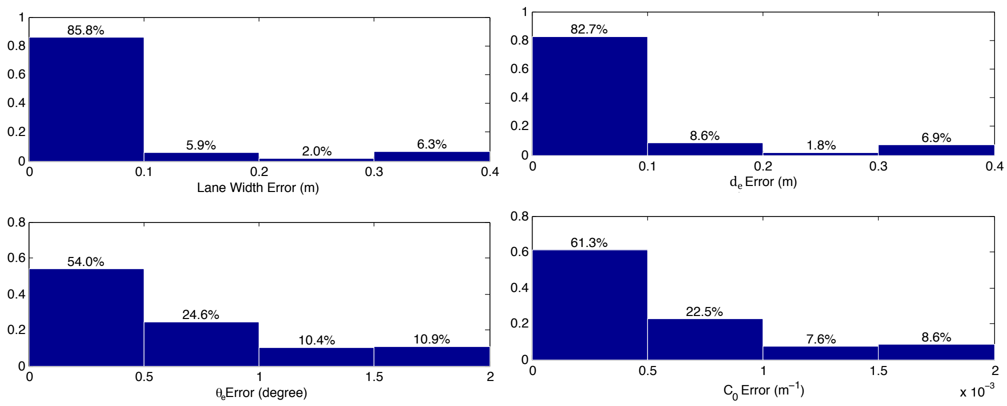


Figure 2.21: Model absolute error distribution

The comparison results with the original approach in terms of root mean square errors are shown in TABLE 2.3. n refers to the number of discrete values assigned to the pitch angle in the original approach as aforementioned. L, θ_e and d_e are always better. C_0 is a little bit worse than the best performance of the original approach with $n = 1$, but better than the remaining cases.

Another important parameter related to localization is the camera pitch an-

Table 2.3: Lane model parameters comparison between proposed and original approach (root-mean square error)

Parameter	Proposed	Original			
		n=1	n=3	n=7	n=41
$L\ (m)$	0.070	0.215	0.222	0.210	0.215
$\theta_e\ (degree)$	0.94	1.09	1.75	1.54	1.56
$d_e\ (m)$	0.116	0.25	0.26	0.26	0.26
$C_0\ (m^{-1})$	0.0029	0.0027	0.0050	0.0045	0.0046

gle. The simulation results for both approaches are illustrated in Fig. 2.22. Top one is the proposed method while bottom is original with $n = 41$ [37]. Zooming into the figures, we can see that the pitch angle from the proposed method is closer to ground truth despite several spikes arising while the original approach provides coarser results around the true values due to the discrete noise. This indicates that pitch angle can be back-calculated/estimated more accurately than assigning a series of discrete values in advance and searching for the best matching. The root mean square error for the proposed approach is 0.1052 while it is 0.1249 for the original approach.

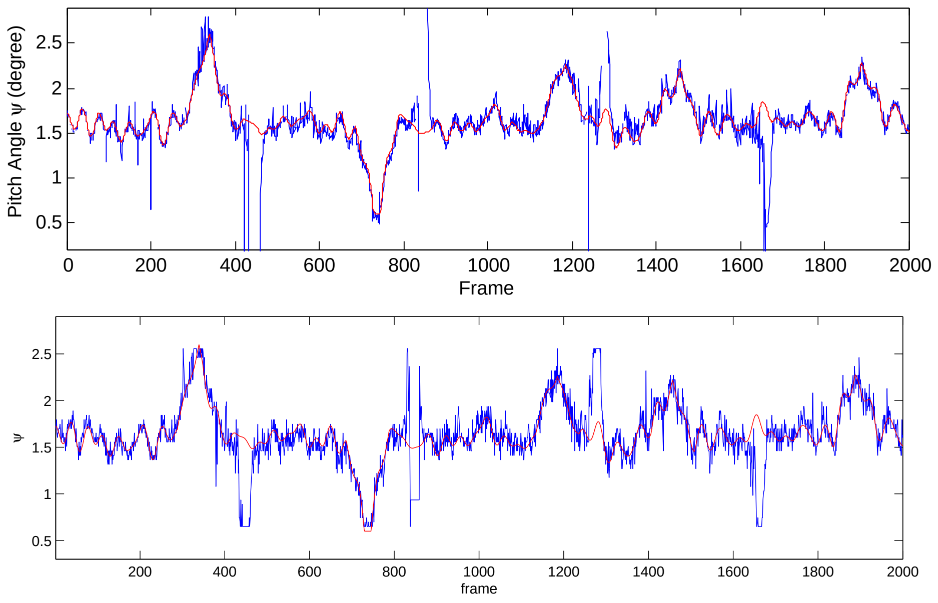


Figure 2.22: Pitch angle comparison between proposed (top) and original approach (bottom). Red line represents simulator ground truth and blue is estimation results

In summary, the simulation results show that the proposed method, even without a priori accurate information of the camera pitch angle, is still able to find a suitable model and generate more accurate geometrical information related to vehicle localization with respect to road lane. In addition, we also show that pitch angle can be back estimated from the model and the value is more precise than the original approach.

2.5.4 Real world environment performance

In the literature, most authors tested their algorithms on highways, where the road structure is well constructed most of the time and the environment is more confined and more predictable. To further challenge the capabilities of our algorithm, we extended the test of the proposed approach on normal roads around NUS (National University of Singapore) instead of highways. The situation is more challenging and erratic as shown in Fig. 2.25.

However, similar to other works, here we face the main difficulties in obtaining precise ground truth. Theoretically, to generate the ground truth, all the five parameters (L, C_0, θ_e, d_e and φ) need to be measured directly at each frame. However, this is infeasible by just using any manual measuring tools. In [44], the ground truth was generated based on high precision GPS and accelerometer-s. But it also required a high-resolution digital map (lane level) in place which is still a research issue in itself and not available widely.

Inspired by [31] and [45], we used a dedicated side camera, mounted on the quarter panel of the vehicle and pointing downwards at about 45°, to measure d_e and θ_e. The measurements from this side camera can be used as the ground truth for the front camera. The reasons are:

(i) The image quality is less affected by vehicle vibrations as the side camera is mounted at a very low position of the vehicle. This also implies that the camera height and pitch angle can be assumed as constant.

(ii) The image is less contaminated by noise (e.g. trees, shadows, buildings etc.) because it consists of road pavement and lane line markings only due to its narrow and limited field of view (FOV).

(iii) The camera is able to capture clearer marking boundaries as it is much closer to the road surface.

Therefore, the results from the side camera are supposed to be more accurate than that from the stereo and thus can be served as ground truth.

However, the side camera is not as reliable or effective as the front camera. It works only when the lane line markings are continuous and within its limited FOV. And it is not able to estimate the road profile in a long range. This is also why the estimations on L and C_0 are not analyzed in this study. Therefore, it can only be a customized method to generate ground truth for this particular application but cannot be implemented as a practical system for autonomous vehicle navigation.

In the following analysis, if the ground truth was not available from the side camera (e.g. at dashed lines), the vehicle dynamics, the prior and posterior ground truth were used to interpolate the missing ground truth.

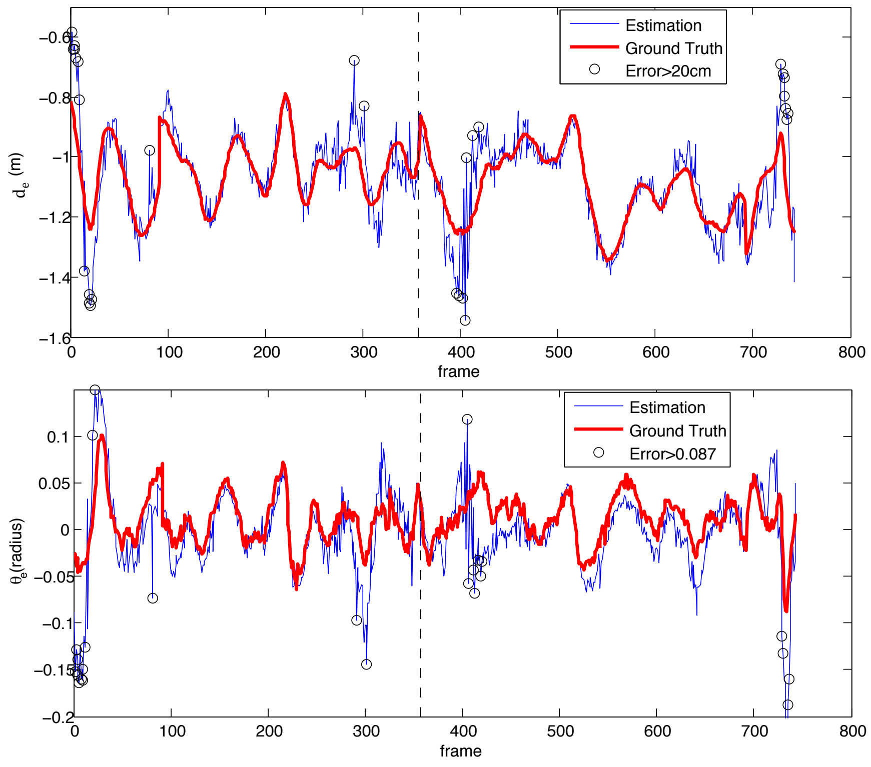


Figure 2.23: Evaluation on model accuracy for real world testing

The intrinsic and extrinsic parameters of both the front and side cameras are listed in TABLE2.4.

Table 2.4: Parameter setup for front and side cameras

	E_u	E_f	Principle point	nominal H	nominal φ
Front camera	839	839	(371, 237)	1.55m	0.349
Side camera	717	717	(307, 257)	0.50m	0.785

Fig. 2.23 illustrates the comparison results based on two video clips (black vertical dash line marks the separation of the two videos). Both videos contain solid/dash single line markings on one side of the road. The noise sources include non-flat surface, poor illumination, warning letters and glare from the head lights of incoming vehicles. TABLE 2.5 shows the corresponding mean error, absolute mean error and the error standard deviation for both parameters.

Table 2.5: Fitting results error statistics

	Mean	Abs Mean	STD
$\theta_e\ (rad)$	-0.0131	0.0247	0.032
$d_e\ (m)$	0.0018	0.0559	0.0811

From the comparison, it is clear that most of the time, the proposed algorithm can provide accurate and consistent estimation on vehicle lateral distance d_e to the road boundary and its moving direction θ_e with respect to road. The average absolute error in d_e is only $5.6cm$, even less than lane marking width.

The most prominent errors for both parameters, as highlighted by black circles in Fig. 2.23, occur at the locations where the road surface is not flat and the road is curving. This observation tallies with that from the simulation. Two examples are illustrated in Fig. 2.24, where the fitted line (red) slightly miss-aligns with the medial axis of the lane line markings at the bottom of the image.

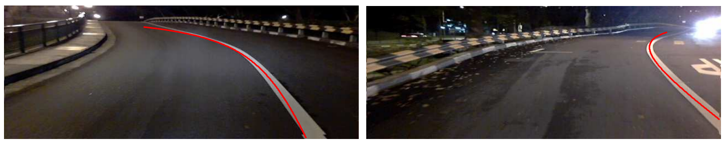
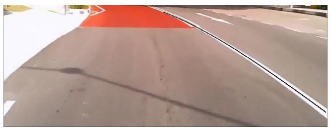
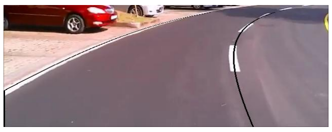
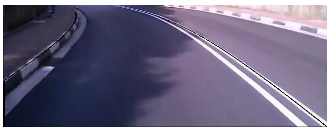


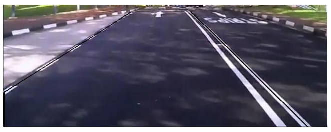
Figure 2.24: Inadequate model fitting examples (red lines) at non-flat road surface

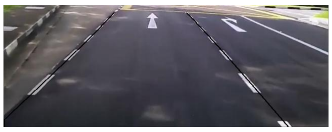
Another observation, similar to the simulation as well, is the non-smoothing estimation, which fluctuates around the ground truth values with small errors. From the statistical analysis in TABLE 2.5, the estimation error can be approximated as zero-mean white noise since their mean values are close to zero. This implies that a Particle Filter can be implemented to improve the consistency.

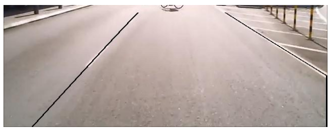

2.25.a: Line on one side only

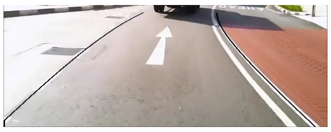

2.25.b: Sharp Turn

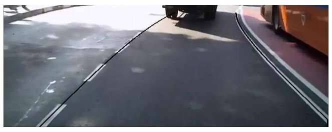

2.25.c: Different Lighting

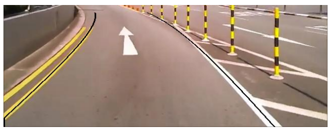

2.25.d: Shadow

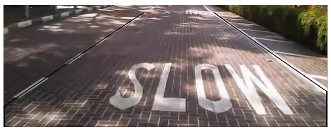

2.25.e: Breaking Line 1

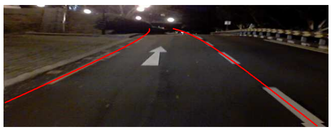

2.25.f: Breaking Line 2

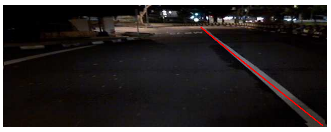

2.25.g: Worn-out of lane line

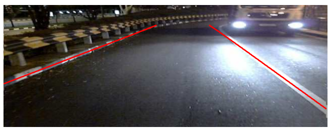

2.25.h: With other vehicles


2.25.i: Different line colors


2.25.j: Different Surface


2.25.k: Breaking lines


2.25.l: Poor illumination


2.25.m: Glare

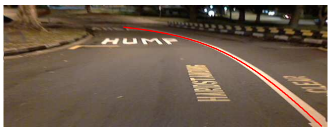
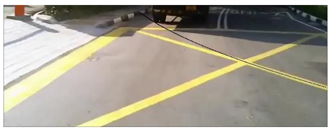

2.25.n: Bending

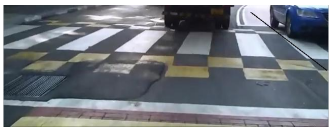
Figure 2.25: Fitting results under different situations. $a \sim j$ daytime vision (black line) and $k \sim n$ night vision (red line)

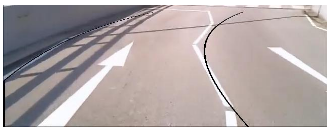
In Fig. 2.25, we present quantitative fitting results sampled from the testing sequence under different situations as indicated under each image. Black (daytime vision $a \sim j$) and red lines (night vision $k \sim n$) represent the fitted results.

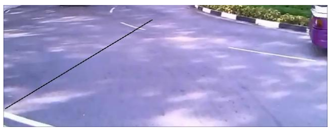
In this section, we demonstrate both qualitatively and quantitatively that the proposed approach works well even with strong disturbances coming into the images. It is capable to provide estimation on vehicle localization with respect to road lane lines, which can be used as a feedback to control an autonomous vehicle to follow the lane.

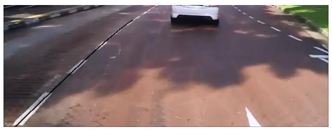
In addition, when lane line exists on one side only (Fig. 2.25(a,l,n)), the original approach in [37] will fail due to the nature of the algorithm but the proposed one is still able to detect the lane line. Under situations with small irregularities on road surfaces (such as *d* and *j*), the original approach is not able to remove noise pixels effectively and consequently it will have a higher probability to fail as shown by the probability analysis in Section 2.5.1.


2.26.a: Non-stop yellow box


2.26.b: Zebra crossing


2.26.c: Before zebra crossing


2.26.d: Round-about


2.26.e: Non-parallel lines

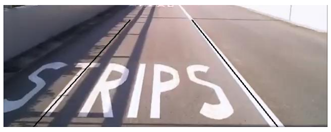

2.26.f: Warning letters

Figure 2.26: Situations where original and proposed approaches not working properly

However, there are still extreme situations under which both the proposed and original approaches may not work satisfactorily. These situations include:

(i) Other road markings, which are similar to lane line markings, may result in false detection, for example, the non-stop yellow box (Fig. 2.26(a)) and zebra crossing (Fig. 2.26(b)).

(ii) Lane lines do not follow the hyperbola or straight line assumptions. For example, before the zebra crossings, the lane line is in zigzag shape (Fig. 2.26(c)). At round-about, it forms a circle (Fig. 2.26(d))

(iii) Lane lines are not parallel. For example, at lanes merging location, the lane line width shrinks (Fig. 2.26(e)).

(iv) Sometimes, warning letters on the road will lead to false detection as well (Fig. 2.26(f)), especially when lane line exists on one side only.

To address these cases, additional special detectors should be implemented to indicate to the algorithms that the vehicle is approaching these locations, so that the results can be interpreted accordingly.

2.6 Chapter summary

In this chapter, we proposed a robust and reliable vision-based lane line detection approach which works even under challenging road situations. We demonstrated that the fitted model is capable of providing accurate estimation on vehicle localization information with respect to road lane lines, including the camera pitch angle φ, vehicle heading direction θ_e, vehicle position to lane line d_e *or* d_r, road width L and road curvature C_0.

However, this system is still not practical enough to be implemented on autonomous vehicles. Firstly, it is not able to handle the extreme conditions as shown in Fig. 2.26. Although they occur intermittently, they can cause fatal errors. Secondly, the estimation results are too sensitive to the accuracy of φ. Lastly, its estimation consistency should be further improved by including a filtering step.

Due to these drawbacks, the single camera system is not implemented on the AV platform eventually, but it proves that the ridge detector is effective for lane line detection and vehicle pose can be estimated from the road model. These findings and lessons enable and motivate us to come out with the more advanced stereovision system in Chapter 3. Without these findings in this chapter, the stereovision system would not be as accurate and consistent as it is, and consequently the NMPC system in Chapter 4 would not control the vehicle properly. Furthermore, the parking slot detection and tracking algorithm in the self-parking system in Chapter 5 is also inherited from this chapter.

Chapter 3

Stereovision-Based Lane Line Detection and Vehicle Localization

3.1 Introduction and literature review

As highlighted in Chapter 2, the mono-camera system has limitations and drawbacks, but it provides us the insights and experiences in carrying out the research in this field. With the lessons learnt from the mono-camera system, we re-visited the literature and proposed a more advanced vehicle localization system based on stereovision instead of mono camera. A more comprehensive and updated literature review is provided below to illustrate how we arrived at the stereovision system.

As aforementioned in Chapter 2, lane-level vehicle localization includes the estimation on vehicle's lateral distance to the lane boundaries and vehicle moving direction with respect to the lane tangent. However, as raised in both the recent survey [19] and the previous chapter, research gaps still exist to apply vision system to achieve accurate lane-level localization for autonomous vehicles. The main difficulties lie in understanding complex roads, achieving high reliability and catering road singularities (such as low visibility, zebra crossing and various illuminations as defined in [46]).

The objective of this chapter is to propose a more reliable and comprehensive vision-based system to achieve high precision lane-level localization for autonomous vehicles. As summarized in [19] and Chapter 2, four major steps are required generally, including lane line feature extraction, lane line model fitting, time integration (or tracking or filtering) and vehicle pose estimation from the fitted model.

Lane line feature extraction is to identify the pixels that belong to lane line markings and eliminate non-lane line marking pixels. Most approaches in the literature are based on the observations that lane markings have large contrast compared to road pavement.

Some gradient based algorithms can be commonly found in the literature. Besides those reviewed in Chapter 2, some other examples are Sobel edge detector with symmetrical local threshold [47], adaptive thresholding [34] and gradient-enhancing conversion [48]. But as pointed out before, these algorithms are sensitive to noise and can result in a large number of outliers from clutter and shadows. Furthermore, they are limited to local view and ignore the shape feature of lane line markings (long and thin bright structures).

Some other more advanced variants based on image gradient have been proposed in the literature, which are less sensitive to noise. For example, the steerable filter ([45][29]) is based on second order derivatives of 2-D Gaussians and ridge detector ([37][49] and Chapter 2) is based on tensor field construction of first order derivative. Both methods are able to get the response of gradient directions which facilitates to remove outliers if their directions deviate too much from the presumed lane line direction.

Another set of algorithm attempts to detect lane line markings from a different perspective, searching for low-high-low intensity pattern along image rows. The most common one is box filter (also known as top-hat filter) or other forms of variants, e.g. [50], [51], [52], and [53]. They are considered as more reliable than the aforementioned algorithms. In brief, it convolutes the image with certain form of step filters and selects the high response pixels as the lane line candidates at each image row. Normally, it is capable of extracting the medial axis of lane line markings instead of edges.

For this kind of algorithms to work properly, its scale or step width must be tuned accurately according to the lane line marking width in the image, to prevent under/over filtering. Otherwise, the original image has to be transformed through inverse-perspective mapping (IPM) to compensate the camera perspective effect (e.g. [54] [55]). But this also requires a good estimation of camera pitch angle (or viewing angle). At the same time, interpolation is needed to make up for the missing pixels in the IPM image. As the viewing distance becomes larger, the interpolation becomes more and more inaccurate.

Another shortcoming that is common to the aforementioned lane line extraction algorithms, including the one proposed in Chapter 2, is that they cannot distinguish lane line markings with other on-road markings, such as warning letters, humps and so on. These on-road markings may result in severe estimation errors from time to time.

The second step is model fitting. It is the process to extract a compact high-level representation of the lane from the lane line detection results. Depending on the model used, the vehicle pose can be derived from the fitted model as shown in [37] and Chapter 2. The model can also be used to guide the lane line detection in the next frame to improve continuity (e.g. [56] [57]).

Different road models have been proposed in the literature. Those reviewed in Chapter 2 are parametric models. The other group is semi-parametric and mainly consists of splines, such as B-Snake [58], Cubic splines [24], active contours [59] and so on. The advantage of these models is that they are more flexible and can cover various road shapes. But they are more computationally demanding and complex. They also require a good selection of control points. As concluded in [19], there is no single model that can cover all kinds of road shapes, on-line model selection should be considered.

The time integration step, which is not implemented in Chapter 2, is to make use of the previous information to guide the search in the current image. It imposes smoothness and continuity between consecutive images. It can improve vehicle localization accuracy and prevent erroneous detection failures.

Most of the proposed approaches are stochastic. For example, the Kalman filter can be found in [45], [50] and [60] and particle filter is applied in [24], [29], [35] and [57]. As pointed out in [19], the particle filter is more reliable especially under abrupt changes in between consecutive images induced by vehicle vibrations or non-flat road surfaces.

In general, the particle filter can be implemented directly to image (or pixels), lane line model and vehicle. For example, in [24], each particle contains the locations of control points in the image for the cubic spline fitting. In [35] and [55], each particle represents lane line model parameters. The change of the parameters is simply assumed to follow a Gaussian distribution. But they did not mention how the covariance matrix was obtained. In [29], each particle represents the location of the vehicle in real world coordinate but again the motion of vehicle is simply assumed to be Gaussian.

Since the change between consecutive images is purely due to the vehicle motion, a more intuitive and straightforward approach is to apply the particle filter on the moving vehicle and take its explicit dynamic model into account, especially when the vehicle is moving fast.

The last step in the lane-level localization is to estimate the vehicle lateral position and moving orientation based on the lane line model. To recover this information from 2D image to 3D real world, depth is required. In most approaches, depth is derived from the camera viewing angle or pitch angle by assuming constant camera height and flat road surface. One typical example is the IPM. The approach in Chapter 2 also follows the same principle but in a more compact and hidden form as shown from (2.18) to (2.22). Therefore, localization accuracy depends strongly on the pitch angle and it is sensitive to pitch angle estimation noise.

A more reliable and direct way to recover the depth is through stereo cameras, given that the disparity image can be constructed effectively and accurately. Another advantage to use stereo is that on-road vehicle or obstacle detection can be integrated easily into the system. However, as mentioned in [19], the low texture of road surface poses processing challenge to obtain the disparity image. This is the main reason why stereo is not widely adopted in this research field. In [61], the author used dense mapping to obtain disparity while in [62], Maximum A Posteriori - Markov Random Field (MAP-MRF) approach was applied. But both are not effective and subject to smoothing noise.

Leveraging on the problems listed above and the unresolved issues in Chapter 2, we propose a more comprehensive vision-based solution to achieve high precision lane-level vehicle localization. The system makes use of stereo for 3D information reconstruction and the particle filter for time integration. The novelties lie in four fold:

(i) A more reliable lane line detection algorithm with adaptive Gaussian-box filter and special detectors to remove other road markings.

(ii) Simultaneous multi-model fitting which covers straight lines, curves and even zigzag lines. We proposed the first fitting algorithm for zigzag line.

(iii) New particle filter framework which takes vehicle dynamics and road shape into account explicitly.

(iv) Effective way of recovering 3D information from stereo vision based on road model. It avoids the conventional time-consuming dense correspondence mapping algorithms.

The chapter is organized as follows: Section 3.2 provides detailed elaboration on the aforementioned four steps. The experiment setup and results are illustrated in Section 3.3, and conclusions are drawn in Section 3.4.

3.2 Image processing

3.2.1 Pre-processing

As aforementioned, the images are from a pair of stereo cameras. Before they are processed, several pre-process steps are required.

First, we design a customized mechanism to determine camera exposure time to prevent over/under exposure. Normally, the exposure time is adjusted based on different metering mechanisms on image overall brightness, e.g. partial area metering, center weighted metering [63]. Here in our approach, the exposure time is calculated based on the road surface (ROI) brightness. The road surface is approximately derived from the previous lane line detection. This is helpful especially when the vehicle moves from a shaded area to an unshaded or the other way round.

Secondly, the images from left and right cameras need to be rectified so that the corresponding pixels in the two images lie on the same row [64]. The rectification parameters can be calibrated accurately according to [40].

Thirdly, both images need to be converted to grey images. We choose intensity value from HSI color space as the grey value, same as Chapter 2.

3.2.2 Lane line detection

Both images I are smoothen or convoluted by a normalized 1D Gaussian kernel G with variable step width w_G.

$$G(v) = \exp\left(\frac{-0.5x^2}{(0.5w_G)^2}\right) \Bigg/ \int_{-0.5w_G}^{0.5w_G} \exp\left(\frac{-0.5x^2}{(0.5w_G)^2}\right) dx \qquad (3.1)$$

where v is image row, w_G is the width of Gaussian kernel and x is the integration variable varying from $-0.5w_G$ to $0.5w_G$.

w_G has to be adaptive to prevent over/under filtering. It is adjusted according to the image row (v), the real lane line marking width (w_m), the camera pitch angle (φ), height (H) and focal length (f).

The left and right edges of lane line markings follow (3.18) and (3.19). Then w_G can be derived as (3.2) where φ and H are from the previous estimation. f is camera focal length. w_m normally follows government standards (e.g.[65]) and have few possible values ($10cm$, $15cm$ and $20cm$). At the end of each iteration, several templates based on these values are generated. A template matching process with the lane line markings in the current image is carried out to check which value is more accurate and should be used in the next iteration.

$$w_G(v) = u_l - u_r = w_m \cos\varphi(v + f\tan\varphi)/H \tag{3.2}$$

The smoothed image L and corresponding gradient image L_u along u are defined in (3.3) and (3.4), where $\star$ denotes convolution and u is column number.

$$L(u,v) = I(u,v) \star G(v) \tag{3.3}$$

$$L_u(u,v) = \partial L(u,v)/\partial u \tag{3.4}$$

A lane line marking pixel $I(u_0, v_0)$ is detected if it satisfies

$$\arg\max_u \left(L(u,v_0) | u \in [b_l,\ b_r]\right) = u_0 \tag{3.5}$$

$$L_u(u,v_0) \geqslant -s,\ \forall u \in [b_l,\ u_0] \tag{3.6}$$

$$L_u(u,v_0) \leqslant s,\ \forall u \in [u_0,\ b_r] \tag{3.7}$$

$$\int_{b_l}^{u_0} L_u(u,v_0)\, du \geqslant I_t \tag{3.8}$$

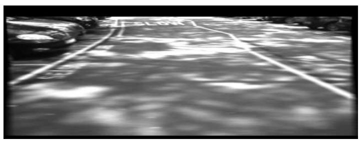
$$\int_{u_0}^{b_r} L_u(u,v_0)\, du \leqslant -I_t \tag{3.9}$$

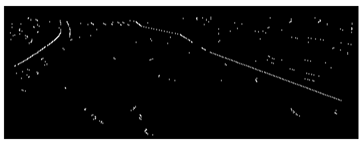
where $b_l = u_0 - w_G(v_0)$, $b_r = u_0 + w_G(v_0)$, s (set at 3) is a small positive value to add in tolerance in the increasing or decreasing trend and I_t (set at 30) is the intensity threshold.

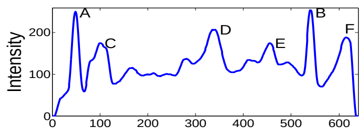
The set of constraints defines a hill structure in L along its rows, with specifications on the hill height and steepness. The detection algorithm is insensitive to illumination changes as the threshold is not based on absolute intensity but the difference between hill peak and valley. It is not subject to local view problems [36] because it takes its neighbourhood pixels into account.

Fig. 3.1 illustrates the lane line detection process. Fig. 3.1(d) depicts the intensity plot at *Row* 125 of the smoothed image. *A* and *B* are detected as the lane line marking pixels. The rest are not because their hill structures are not steep enough.

3.1.a: Original image


3.1.b: Smoothed image


3.1.c: Detection result



3.1.d: Row125 intensity

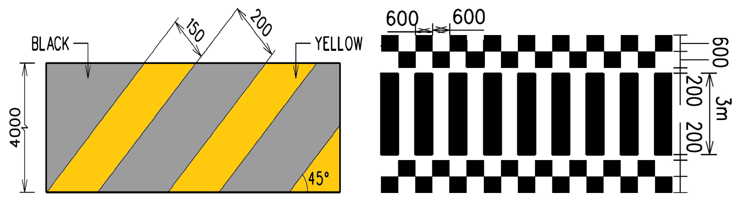
Figure 3.1: Illustration on lane line detection algorithm

The proposed algorithm is capable of detecting bright long structures in the images. However, besides lane line markings, humps, zebra crossings, warning letters and arrows also exhibit similar features and may be false detected as lane line markings as shown in Fig. 3.5(b), Fig. 3.6(f) and Fig. 3.7(e). This is a common issue to other detection algorithms as well. To distinguish them from lane line markings, special detectors have to be implemented.

3.2.2.1 Hump Detector

As show in Fig. 3.2(a) and 3.5(a), hump markings consist of equally spaced parallel lines that cover the entire lane width and are slanted to lane tangent. They follow the government standards as shown in Fig. 3.2(a).

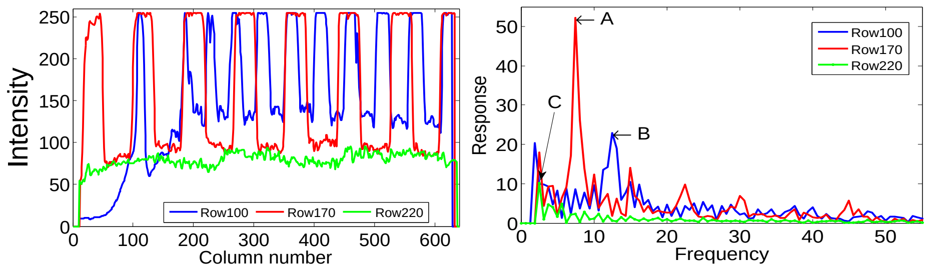
Row wise intensity change of the hump portion in the image follows a periodic pattern as shown in Fig. 3.3(a) *Row* 100 and 170. Fourier transform [66] is



3.2.a: Hump 3.2.b: Zebra crossing

Figure 3.2: Government standards for hump and zebra markings (in *mm*) [7]

implemented to convert the row-wise signal to its frequency domain. As indicated by A and B in Fig. 3.3(b), they have obvious dominant frequencies.



3.3.a: Row wise intensity 3.3.b: Frequency response

Figure 3.3: The frequency response at different rows of hump image shown in Fig. 3.5

However, due to the image perspective effect, their dominant frequencies at different rows are different. At *Row* v, the distance d_y between two yellow marking lines can be derived similarly as w_G.

$$d_y(v) = 0.35(v + f\tan\varphi)\cos\varphi/H/\cos(\theta_e - \pi/4) \tag{3.10}$$

where φ, H and θ_e are all from the previous detection results. Then the signal frequency f_d at *Row* v is defined as (3.11) with W_I being the image width.

$$f_d(v) = W_I/d_y(v) \tag{3.11}$$

Based on (3.10) and (3.11), an upper and lower limits for the dominant frequency at *Row* v can be derived empirically if assuming the estimation error in θ_e varies from $-\theta_E$ to θ_E. An example based on the hump image (Fig. 3.5) is shown in Fig. 3.4. The actual dominant frequency is derived from the Fourier transform at each image row.

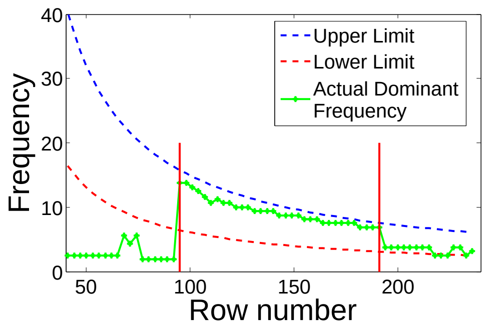
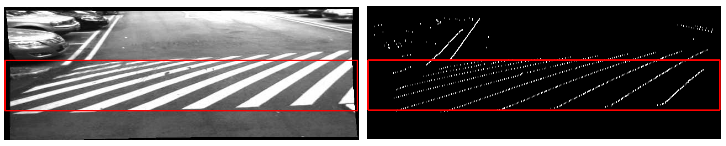


Figure 3.4: Illustration on frequency limits at different image rows and the actual dominant frequencies derived from Fourier transform

If *Row* v belongs to hump, its dominant frequency must be within the upper and lower limits. At the same time, its response at the dominant frequency has to be larger than a prescribed threshold (set at 18.5). For example, for *Row* 220, its dominant frequency is very small and within the limits (Fig. 3.4). But its response is very low at this frequency as indicated at C in Fig. 3.3. Therefore, *Row* 220 is not considered as hump.

One more step is required to further remove the false detections. Similarly to the lane line detection, the image is smoothed by G and all hill structures are detected as shown in Fig. 3.5(b). At each row, the number of structures has to be greater than a prescribed threshold (set at 4) and the distance between two consecutive hill structures must be within the range defined in d_y (by varying θ_e between $\theta_e - \theta_E$ and $\theta_e + \theta_E$).



3.5.a: Hump image 3.5.b: Lane line detection

Figure 3.5: Hump and the lane line detection results. (The red box indicates the hump detection result)

The hump detection results are shown by the red box in Fig. 3.5. Although every single line of the hump is identified as the lane line markings, they will be removed effectively after the hump detection.

3.2.2.2 Zebra Crossing Detector

Similar to the hump detector, zebra crossing can be identified through Fourier transform as well. The dominant frequencies f_d and the smoothing kernel G need to be adjusted based on the dimensions in Fig. 3.2(b).

3.2.2.3 Warning Letter Detector

The types of warning letters painted on the road surface are very limited. Common ones may include "*SLOW*", "*HUMP*", "*AHEAD*", "*BUS*" and so on. They follow the standards as listed in [7] (different countries may follow different standards), from which the templates can be obtained.

However, the letters in the images are subject to camera perspective, orientation and scaling effects. For this kind of template matching, the advanced approaches in the literature [67] include Scale-Invariant Feature Transform (SIFT), Principal Component Analysis (PCA)-SIFT [68], Speeded Up Robust Features (SURF) and Features from Accelerated Segment Test (FAST, [69] and [70]). But they are not applicable here due to their relatively slow speed. The low-texture feature of the warning letters also makes it difficult to identify corresponding feature points effectively.

Here we propose a fast convolution based template matching algorithm for warning letter detection.

First, image I is transformed through IPM to remove the perspective and scaling effect. Since we only target to identify the rough position of the warning letters, all the shortcomings with IPM as mentioned in Section 3.1 are tolerable. Then threshold the IPM image to obtain the binary image I_{inv} (Fig. 3.6(b)).

The second step is to remove its orientation effect. Every single warning word has line components that is perpendicular to its orientation. 13 s-traight line masks are predefined, representing 13 equally spaced orientations in $[-\pi/3,\ \pi/3]$. Their lengths are set approximately as the height of the letters.

Convolute I_{inv} with every mask. At each pixel, its maximum response I_{max}

(Fig. 3.6(c)) and the corresponding orientation I_{ori} can be obtained.

$$I_{mask_i} = I_{inv} \star mask_i \ , \quad i = 1, 2, \cdots, 13 \tag{3.12}$$

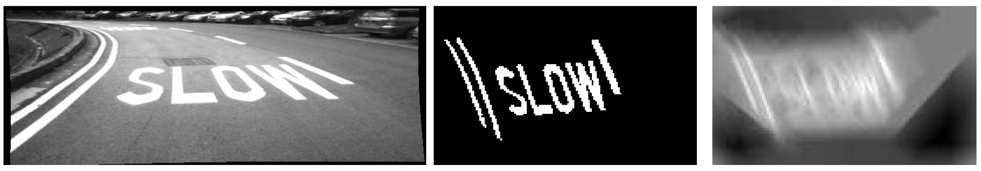
$$I_{max}(u,v) = \max\left(I_{mask_i}(u,v) \middle|\ i = 1, 2, \cdots, 13\right) \tag{3.13}$$

$$I_{ori}(u,v) = \arg\max_i \left(I_{mask_i}(u,v) \middle|\ i = 1, 2, \cdots, 13\right) \tag{3.14}$$

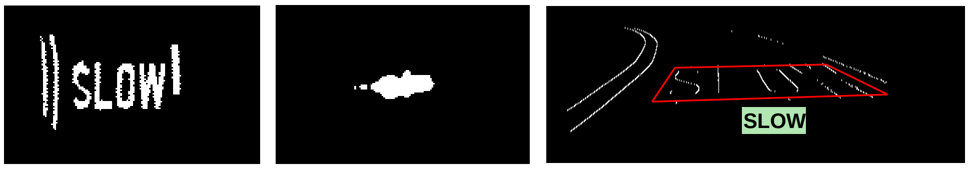
For each orientation i, count the number of pixels (N_i) that have the maximum response at orientation i and match $> 75\%$ to the corresponding mask.

$$N_i = \left|\left|(u,v)\right| I_{max}(u,v) > 0.75,\ I_{ori}(u,v) = i,\ \forall u,v\right|\Big|_0 \tag{3.15}$$

The orientation of the warning letters can be approximated by the angle that corresponds to the maximum of N_i. Then I_{inv} can be rotated accordingly as shown in 3.6(d) I_{rot}.



3.6.a: Original image I 3.6.b: IPM (threshold) 3.6.c: Direction response



3.6.d: Rotation 3.6.e: ROI 3.6.f: Result

Figure 3.6: Warning letter detection algorithm

The next step is template matching which is the most time consuming part. To reduce computation effort, regions of interest (ROI) will be identified first.

Based on the statistical analysis on all the templates, the coverage ratio, defined as the number of **1**s over the total number of pixels in the template, falls in the interval of $[0.3,\ 0.5]$. Taking the pixel (u,v) in I_{rot} as the center of a rectangular, which has similar size as templates, if the coverage ratio of the rectangular is in the interval, the pixel (u,v) belongs to ROI. This process can be realized through convolution as shown below:

$$I_{conv} = (I_{rot} \star O)/n_O \tag{3.16}$$

$$ROI = \{(u,v) : I_{conv}(u,v) \in [0.3, 0.5]\} \tag{3.17}$$

where O is a matrix containing **1** only and its size is similar to the templates. n_O is the total number of entries in O.

Each pixel in ROI will go through template matching. The difference measurement index is $L1$ norm due to its efficiency and robustness as concluded in [71]. Once the index is less than 0.3, the corresponding pixel will be taken as the center of the particular warning letter. Its four vertices in I can be back calculated. Any pixel from lane line detection will be removed if they fall into the area enclosed by the four vertices. The final result is shown in Fig. 3.6(f).

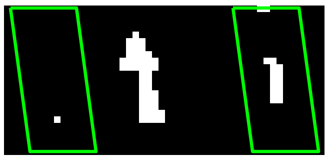
3.2.2.4 Arrow Detector

Since the arrow has very clear and unique features, PCA is implemented for arrow detection. To remove the perspective effect, IPM image is used, same as the warning letter detection. The major components of PCA is generated by rotating the standard template from -30° to 30° with a step size of 1°. The standard template is again from [7]. There are a total of 61 templates and based on SVD decomposition, the first 10 components are more dominant.

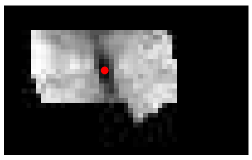
After the IPM and threshold, similar ROI selection process as defined by (3.16) and (3.17) is carried out. In this case, the size of O is the same as the arrow template size and the coverage ratio interval is $[0.07\ 0.10]$.

Each pixel in the ROI will be expressed by the 10 major PCA components and its corresponding coefficients are compared with the coefficients of the 61 templates. The one with minimum difference will be taken as the template for that pixel. The difference image I_d is shown in Fig. 3.7. In I_d, the minimum value is identified (marked as red) and if its corresponding difference is less than a threshold (set at 5.5), the pixel will be taken as the center of the arrow and the four vertices that correspond to the particular template can be obtained and then they can be back calculated in I. Any pixel in I that is within the area enclosed by these four vertices will be removed.

3.7.a: Original image I


3.7.b: IPM (threshold)

3.7.c: ROI


3.7.d: Difference image

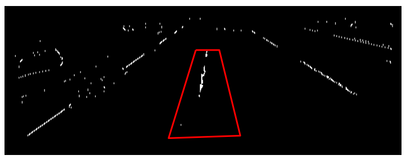

3.7.e: Result

Figure 3.7: Arrow detection algorithm

However this algorithm may lead to false detections for those wide lane line markings. To eliminate this, the predicted lane line model based on the vehicle motion and the previous lane line detection is worked out. After projecting the predicted model on the IPM, a range, with fixed width and centered at the predicted model, is generated as shown by the green boxes in Fig. 3.7(b). Any pixel in this range will not be considered for arrow detection.

As known, PCA is not robust for partial object detection. For example, for the image in Fig. 3.7(a), when the vehicle moves forward, the arrow tail will disappear from the bottom of the image. The PCA algorithm will fail to identify the remaining part of the arrow. If this happens, a similar PCA detection algorithm based on arrow head will be carried out.

3.2.3 Model fitting

The basic road model adopted here is hyperbola, similar to (2.8) and (2.9), but with the assumption of $E_u = E_v = f$. The straight line and zigzag line can be derived from the hyperbola. For the stereo camera pair, if the left road lane line follows (3.18) in the left camera, then its correspondence in the right camera follows (3.20) with d_{lr} the distance between left and right cameras.

$$u_l = \frac{f\theta_e}{\cos\varphi} + \frac{\cos\varphi d_e(v_l + f\tan\varphi)}{H} + \frac{f^2HC_0/\cos^3\varphi}{4(v_l + f\tan\varphi)} \tag{3.18}$$

$$u_r = \frac{f\theta_e}{\cos\varphi} - \frac{\cos\varphi d_r(v_r + f\tan\varphi)}{H} + \frac{f^2HC_0/\cos^3\varphi}{4(v_r + f\tan\varphi)} \tag{3.19}$$

$$u_l = \frac{f\theta_e}{\cos\varphi} + \frac{f^2HC_0/\cos^3\varphi}{4(v_l + f\tan\varphi)} + \frac{\cos\varphi(d_e + d_{lr})(v_l + f\tan\varphi)}{H} \tag{3.20}$$

Theoretically, this also means that if we can solve the unknowns in (3.18) and (3.20), we are able to find the correspondence of the lane line in left and right images. Then, the lane line disparity can be calculated and followed by 3D road reconstruction.

These two equations can be solved independently but there is no guarantee that they result in the same H, θ_e and d_e and thus no guarantee on correspondence. They have to be solved jointly. Define the pixels in left and right image to be (u_L, v_L) and (u_R, v_R) respectively. Subtract (3.18) with (3.20), we can have

$$\begin{aligned} u_L - u_R = &\frac{f^2HC_0}{4\cos^3\varphi}\left(\frac{1}{v_L + f\tan\varphi} - \frac{1}{v_R + f\tan\varphi}\right) \\ &+ \frac{\cos\varphi}{H}d_e(v_L - v_R) - \frac{\cos\varphi}{H}d_{lr}(v_R + f\tan\varphi) \end{aligned} \tag{3.21}$$

To solve (3.21), we take φ as known from the previous detection and H, C_0 and d_e as unknown, so that (3.21) becomes LIP (linear in parameter). Taking two pixels each from left and right images is sufficient to solve the unknowns in simple linear algebra. Substituting the unknowns in (3.18) or (3.20), θ_e can be obtained as well.

It seems we have solved the vehicle lane-level localization; however, the results (d_e, θ_e) are not accurate due to the various assumptions as listed in [37] and there is no mechanism to update φ.

Therefore, this model, defined as primary model or road-in-image model, is just used to guide the search for lane line marking pixels in the images. A more accurate model, defined as ultimate model or road-in-real-world model will be derived based on stereo 3D reconstruction to solve the vehicle localization.

Random sample consensus (RANSAC) algorithm is implemented to find the inliers to the primary model. The detailed algorithms are as following:

(i) Randomly select two pixels each from left and right images. Based on (3.21), (3.20) and (3.18), work out the lane line model in both images.

(ii) Verify the derived models by checking C_0 and H. C_0 cannot be very large for road as mentioned in Chapter 2; otherwise the vehicle cannot turn. H cannot be less than 0. If verification fails, go back to i).

(iii) For each lane line candidate pixel in left and right images, calculate its distance to the corresponding models. If its distance is less than a prescribed threshold (set as 2 pixels), it is an inlier to the model.

(iv) Repeat i)-iii) for a certain number of iterations (set as 1600).

(v) The model with maximum number of inliers is selected as the primary hyperbola model.

Note that this RANSAC algorithm based on (3.21) avoids the conventional time-consuming algorithms to find stereo dense correspondence. This innovative approach saves so much computation power that the system can run in real time. This is the main novelty that distinguishes the system from other systems which use stereo images.

For straight road, its primary model can be derived by setting $C_0 = 0$ in (3.21), (3.20) and (3.18). The similar RANSAC procedure can be carried out to fit the primary line model.

For zigzag markings, they can be identified based on the straight road primary model. If the straight model is part of the zigzag line, it must be continuous and a DBSCAN (Density-Based Spatial Clustering of Applications with Noise) clustering algorithm [72] is implemented to remove its discontinuous part. The next step is to search for the next segment of the zigzag line.

Define the inliers to the first straight line as set S_A. Taking the pixels above the first straight line as an example, define them as set S_B. Similarly, to determine the second straight line model, two points each from left and right images are to be selected. The first point from each image can be selected from S_B randomly. The second point from each image is selected on the first straight line, which means it is the intersection of the zigzag line. To select this point, select its *row* value first and determine its *column* value based on the first straight line model. Since the intersection point has high probability to lie in the vertical gap between S_A and S_B, the *row* value can be selected based on the Gaussian distribution with $\mu = 0.5(\min_v(S_A) + \max_v(S_B))$ and $\sigma = 0.5(\min_v(S_A) - \max_v(S_B))$. There are no other fitting methods available in the literature for zigzag line.

After the three RANSAC processes, three primary models, hyperbola, straight line and zigzag line, are derived. The one with maximum inliers is selected as the primary model for further operations.

Since all the results are based on rectified images, in theory, the disparity map of the lane line markings can be obtained straightforwardly by setting $v_L = v_R$ in (3.21). However, due to the following reasons, a refine process is required.

(i) Small offset exists between the fitted model and the actual pixel position

(ii) The fitted model is based on previous estimated φ, which might not be accurate for the current iteration.

(iii) When double lines exist on one side of the road, e.g. Fig. 3.6(a) left lane line markings, since they are very close ($15cm$), the model fitting may match the outer line in one image to the inner line in the other.

The refine process is described as follows. Taking left image as the base, the template at row v_0 is defined as the pixels between $u_L - w_G$ and $u_L + w_G$. The searching range in right image is from $u_R - 2w_G$ to $u_R + 2w_G$ on the same row. The pixel with the maximum matching results in the right image is the correspondence to the base pixel in the left image. By doing this, we are able to map lane line correspondence accurately. And their disparities *dis* at each row can be worked out accordingly.

In stereo coordinate system (by translating origin of left camera coordinate to the midpoint of the left and right cameras), the position of the detected lane line markings follows

$$\begin{cases} X_{cam} = u_L Z_{cam}/f + d_{lr} \\ Y_{cam} = v Z_{cam}/f \\ Z_{cam} = d_{lr} f/dis \end{cases} \tag{3.22}$$

The camera height H and pitch angle φ can be calculated accurately by assuming the short road segment ($3 \sim 6m$) in front of the car is flat. Based on geometry, for these lane line points, they follow (3.23). $\tan\varphi$ and $H/\cos\varphi$ can be solved by least squares, and thus H and φ can be determined.

$$\begin{bmatrix} Z_{cam} & -1 \end{bmatrix} \begin{bmatrix} \tan\varphi \\ H/\cos\varphi \end{bmatrix} = -Y_{cam} \tag{3.23}$$

With these two values, the lane line 3D coordinates in the car coordinate system can be directly obtained by coordinate rotation. In the car coordinate XZ plane, if the primary model is a hyperbola, a parabola model is used to fit the lane line points as shown in (3.24). The resulted model is defined as the ultimate model. Since all the lane line points in car coordinate are transformed from the inliers of the primary model, they are inliers to the ultimate model. Therefore, least squares fitting is enough to work out the model parameters.

$$X_{car} = aZ_{car}^2 + bZ_{car} + c \tag{3.24}$$

Similarly, if the primary model is a straight line, in the car coordinate system, a line model will be implemented accordingly as the ultimate model. However, if the primary model is zigzag line, we need to transform it into one single line model which is parallel to the road boundary.

To achieve this, first, two straight line models can be fitted based on the zigzag line marking points in the car coordinate system. Then calculate their intersection. According to [65], each line segment in zigzag markings is $4m$ long. The midpoint of the corresponding line segment can be located by finding a point on the line segment with a distance of $2m$ to the intersection point. The line determined by two midpoints is the model to describe the zigzag line.

With the ultimate model (3.24), the vehicle localization information (d_e and θ_e) can be calculated easily based on geometry.

To further improve the robustness, after getting the primary model on one side of the road, the algorithm tries to search for lane line on the other side of the road. We define this model as primary conjugate model. But it won't be used for any further calculation. It just checks whether the other side of the road has clearer lane line markings (more inliers). If yes, in the next iteration, the primary model will be fitted by this side.

If the primary model follows (3.18) in the left image, the conjugate model in the left image follows (3.19) if they are parallel. By varying lane width L, we can get an ROI for the right lane. Normally, L is within the range of $[2.5m, 4.0m]$.

When fitting the conjugate model, we will drop the parallel constraint. That means the primary model and its conjugate share the same φ, H and C_0, but have different θ_e and the conjugate has one more variable L. Equation (3.21) becomes

$$\begin{aligned} u_L - u_R = {} & \frac{f^2 H C_0}{4\cos^3\varphi}\left(\frac{1}{v_L + f\tan\varphi} - \frac{1}{v_R + f\tan\varphi}\right) \\ & + \frac{\cos\varphi}{H}(d_e - L)(v_L - v_R) - \frac{\cos\varphi}{H} d_{lr}(v_R + f\tan\varphi) \end{aligned} \tag{3.25}$$

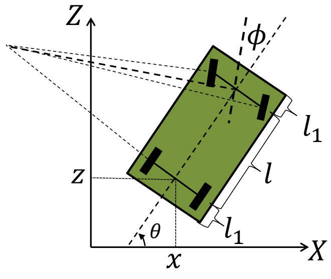
where the only unknown is L and other parameters can be obtained from primary model. Again similar RANSAC procedures can be carried out. But here, we only need to choose one pixel each from left and right images to calculate L. If L is out of the range mentioned above, new pixels need to be selected. Substitute L into the conjugate of (3.18) and (3.20), we can obtain the new θ_e for the conjugate lane line, and thus the conjugate model.

3.2.4 Particle filter

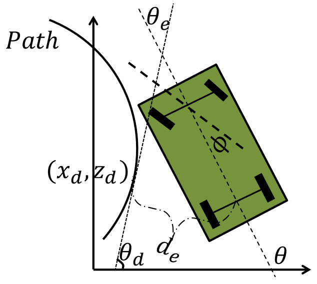
The particle filter has been widely used in this research field for tracking purpose to improve the consistence and smoothness. Here, as aforementioned, we propose a new particle filter scheme which takes the vehicle dynamic model and the road shape into account explicitly. While in other works, vehicle dynamics are always ignored or simplified as Gaussian movements.

The model of the vehicle motion follows Ackermann equations as defined in (3.26), where x, z and θ refer to the vehicle location and orientation in the global coordinate as shown in Fig. 3.8(a). T_s is the sampling time. l is the distance between the front and rear wheel axis. v_k and ϕ_k are velocity and steering direction. d_e and θ_e defined in Fig. 3.8(b) are the same as those in Fig. 2.6.

$$\begin{cases} x_{k+1} = x_k + \dot{x}_k T_s & \dot{x}_{k+1} = \cos(\theta_k) v_k \\ z_{k+1} = z_k + \dot{z}_k T_s & \dot{z}_{k+1} = \sin(\theta_k) v_k \\ \theta_{k+1} = \theta_k + \dot{\theta}_k T_s & \dot{\theta}_{k+1} = \tan(\phi_k) v_k / l \end{cases} \tag{3.26}$$



3.8.a: Ackermann model



3.8.b: Vehicle pose error

Figure 3.8: Ackermann vehicle model and pose error definition

Each particle represents the location of the vehicle. The only measurable variables in this setup are d_e and θ_e. By comparing the measured values and the predicted values associated to each particle, an importance factor of each particle can be calculated by approximating the measurement difference as Gaussian distribution. The SIR (sampling importance resampling) algorithm is carried out to re-draw particles from the current particle pool.

$$\min_{a_m,b_m,c_m} \Big(\sum_{i=1}^{N_{car}} \left(a_m z_{car}^2 + b_m z_{car} + c_m - x_{car}\right)^2\Big) \tag{3.27}$$

$$subject\ to:$$

$$x_t = a_m z_t^2 + b_m z_t + c_m \quad \tan(\theta_e) = 2a_m z_t + b_m$$

$$x_t = d_e \cos(-\theta_e) \qquad z_t = d_e \sin(-\theta_e)$$

The updated d_e and θ_e are taken as the mean of those of the re-drawn particles. The states of re-drawn particles in (3.26) are translated from their original coordinate system to a new one whose origin is at the gravity point of the re-drawn particles and whose z axis aligns with the mean θ of the re-drawn particles. In this new coordinate system, the origin represents the updated vehicle location and its z axis represents the vehicle moving direction. If the updated d_e and θ_e are the same as the measured ones, (3.24) can be used directly for the next prediction step. Otherwise, it has to be refined by solving the constrained optimization problem in (3.27), where N_{car} is the number of detected 3D points, d_e and θ_e refers to the updated ones, x_t and z_t are transition variables.

Equation (3.27) ensures that the vehicle localization with respect to the refined model is exactly d_e and θ_e. At the same time, it ensures the refined model has minimum deviation to the detected 3D points.

Equation (3.26) is used to predict the re-drawn particle based on the input v_k and ϕ_k. To prevent sample impoverishment or increase the particle diversities, an empirical Gaussian covariance matrix is appended to the right hand side of (3.26). The predicted d_e and θ_e associated to each particle can be calculated based on their predicted states and the refined model $[a_m, b_m, c_m]$.

3.2.5 Algorithm summary

For better illustration, the algorithm is summarized in the following flow diagram (Fig. 3.9), where the italics indicate the output of the previous process.

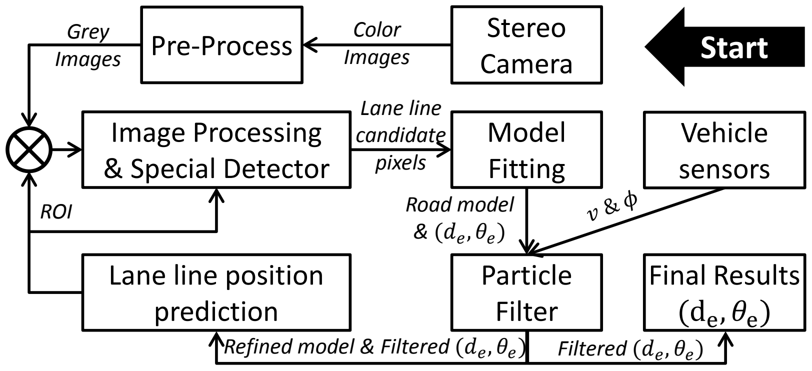

Start
Stereo Camera
Color Images
Pre-Process
Grey Images
Image Processing & Special Detector
ROI
Lane line candidate pixels
Model Fitting
Vehicle sensors
Road model & (d_e, θ_e)
v & ϕ
Particle Filter
Lane line position prediction
Final Results (d_e, θ_e)
Refined model & Filtered (d_e, θ_e)
Filtered (d_e, θ_e)

Figure 3.9: Lane line detection algorithm flow diagram

The algorithm starts from stereo camera and ends with vehicle lane level localization information $(d_e,\ \theta_e)$. The lane line position prediction module makes use of the current estimation results to predict the lane line position in the consequent image to improve the image processing consistency.

To further improve the smoothness of the detection, two more steps are carried out. First, the change of road curvature C_0 cannot be abrupt and can be approximated as logistic distribution. Therefore, the number of inliers in RANSAC model fitting step v) refers to the normalized number by the corresponding logistic distribution. Second, in determining (3.24), the points from previous iteration are taken into account, but with a weighting factor of 50%. Therefore, the lease squares fitting is indeed the weighted lease squares fitting. 50% is determined empirically. The main reason to include the previous detection results is that the lane line portion near the vehicle falls behind the camera scene and thus cannot be seen in the image. The previous detected points are used to estimate this lane line portion.

3.3 Experiment results

The proposed algorithm is implemented in Matlab running on the Asus laptop in TABLE 1.1. On average, it takes only $0.11s$ for one iteration.

The test rout was shown as red in Fig. 3.10 between point A and B. The algorithm has been tested extensively under different situations as shown in Fig. 3.12 and Fig. 3.14. The challenging situations include sharp turns, tree shadows, camera saturation, worn-off lane lines, poor visibility and etc..

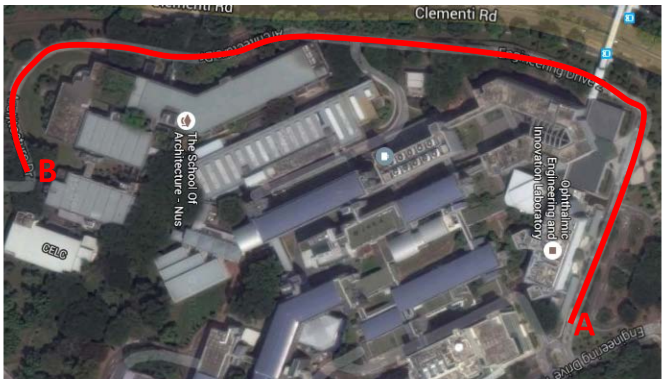

Clementi Rd
Engineering Drive
The School Of Architecture - Nus
Ophthalmic Engineering and Innovation Laboratory
CELC
A
B

Figure 3.10: Test bed area and the test rout

The results shown from Section 3.3.1 to 3.3.4 are based on image sequences taken under normal daytime. To further evaluate the algorithm, it has been tested at different times of the day and under different weather conditions, including dusk, night and after rain. The results are presented in Section 3.3.5. Two illustration video links (Link i and ii) are provided in Appendix B.

3.3.1 Results on special detectors

The results are based on 1145 image sequences. They were captured continuously when driving the vehicle from point A to point B, then back to point A in Fig. 3.10. TPR (true positive rate), FPR (false positive rate) and ACC (accuracy) are used as defined from (2.23) to (2.25).

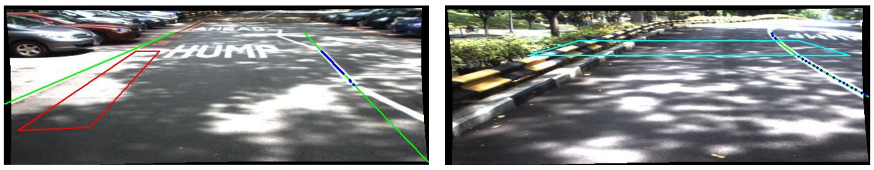
To classify TP and FP when the detection is misaligned or partially overlapped with ground truth, we adopted the index proposed in [73] for car and pedestrian detection. If the overlap ratio is more than 70%, the result is considered as TP; otherwise, it is FP.

The detailed results are tabulated in TABLE 3.1. In general, all the detectors are able to achieve a very high TPR, ACC and low FPR, indicating that most of the time the detectors are able to identify those markings on the road and when they do not appear, the detectors seldom alert.

Table 3.1: Quantitative evaluation of special detectors

	P	N	TP	FP	TPR	FPR	ACC
Arrow	174	971	167	12	0.960	0.012	0.983
Hump	61	1084	58	6	0.951	0.006	0.992
Letter	155	990	140	0	0.903	0.000	0.987
Zebra	44	1101	41	4	0.932	0.004	0.994

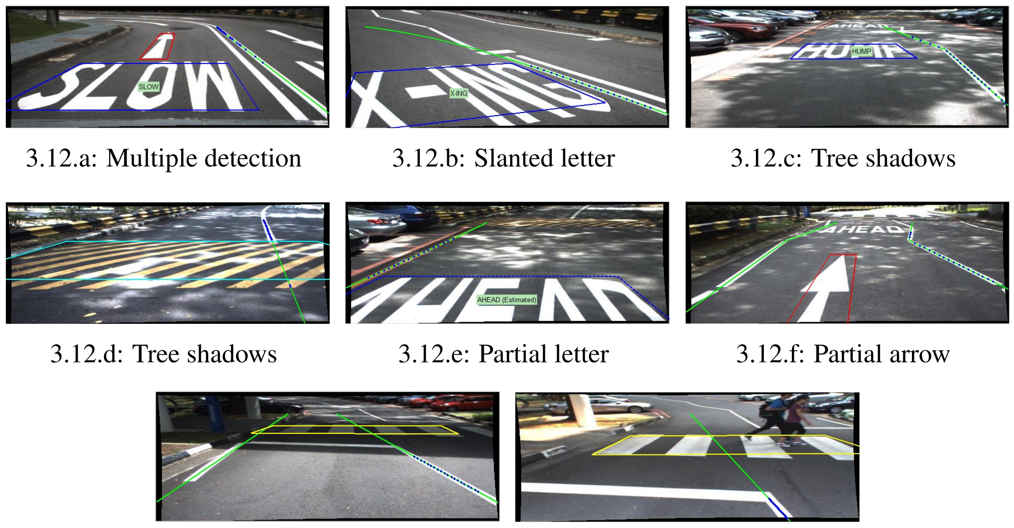
The defects in TPR are mainly due to the long distance between the object and the camera. The object becomes either too small to detect or distorted too much in the IPM image due to the interpolation, e.g., the "HUMP" in Fig. 3.11(a). The small and intermittent FP are mainly induced by strong irregular shadows as shown in Fig. 3.11 where the detected arrow and hump are false.



3.11.a: False arrow detection (red box) 3.11.b: False hump detection (cyan box)

Figure 3.11: False positive detection from strong tree shadows

Fig. 3.12 shows the detection examples under some challenging scenarios. More examples can be found in Fig. 3.14. (a) shows the case when multiple special markings exist in the same image. (b) demonstrates even the letter is slanted in the image, the algorithm is still able to detect. (c) and (d) illustrate the situation under strong irregular tree shadows. In (e) and (f), only half of the special markings are visible in the image. In (g), the visibility of the zebra crossing is poor due to the shadows of the cover roof. In the last image, the zebra crossing is partially occluded by pedestrians.



3.12.a: Multiple detection 3.12.b: Slanted letter 3.12.c: Tree shadows

3.12.d: Tree shadows 3.12.e: Partial letter 3.12.f: Partial arrow

3.12.g: Low visibility 3.12.h: Occluded by pedestrians

Figure 3.12: Special markings detection in challenging situations (Red box: Arrow. Blue box: Warning letter. Cyan box: Hump. Yellow box: Zebra crossing)

3.3.2 Results on lane line detection

The results in this section are based on the same set of images as in Section 3.3.1. As mentioned in Chapter 2, the major difficulty to evaluate the lane line detection performance is the lack of ground truth, especially on lane line level. Some established databases, such as KITTI benchmark [16], are not applicable here since they only include ground truth on road surface but not the lane line.

In Chapter 2, we implemented ROC for evaluation. But this method is very tedious as the lane line pixels of the ground truth were labelled manually image by image. It is not practical in this chapter due to the huge number of images to be evaluated. Therefore, in this chapter, we adopted a simpler method proposed in [34] and [24], which divided the fitting results into three categories: correct detection, misaligned one and incorrect one. We followed the guidelines in [24] to determine them. Although this method is less accurate and less objective than the one in Chapter 2, it is still good enough to provide insights on the lane line detection performance.

For benchmark, the results from the proposed algorithm are compared with *PG* (probabilistic grouping [24]), *ALD* (advanced lane detector [34]) and the one (MC, mono-camera) in Chapter 2. Since none of these algorithms is designed to detect zigzag lines, their detection results on zigzag lines are not counted.

As shown in TABLE 3.2, among these four methods, the proposed one achieves the highest correct fitting rate and lowest incorrect fitting rate. This indicates that the proposed lane line detection algorithm, based on adaptive Gaussian kernel filter with special detectors, is more effective in removing noise pixels under various conditions. Furthermore, the fitting mechanism also reduces its sensitivity to noise, because it fits two correspondence images simultaneously and imposes stereo disparity constraints.

Table 3.2: Lane line detection results comparison

	PG	*ALD*	*MC*	*Stereo*
Correct fitting rate	82%	87%	86.7%	93%
Misaligned fitting rate	4.0%	6.0%	7.0%	5.6%
Incorrect fitting rate	14%	7.0%	6.3%	1.4%

Some incorrect fittings in *PG*, *ALD* and *MC* are from those special road markings, especially from the humps. Some are from sharp turns (Fig. 3.14(i)(j)). The curve penalty in *PG* is too restrict to allow such large curvatures while the logistic distribution used in the proposed algorithm is a more relax assumption and proved to be more reliable and stable.

Other incorrect fittings in *PG* and *ALD* are due to fast vehicle lateral movement. In *PG*, the motions of the control points between consecutive images are assumed as Gaussian. However the fast vehicle lateral movement induces unusual movement to the control points. Consequently, some correct lane-boundary hypotheses with weak lane-marking supports are rejected. In *ALD*, the temporal blurring assumes small ego dynamics. But the fast lateral movement dictates such assumption and results in poor average images and thus high rate of incorrect fittings. Therefore, it is an obvious improvement to consider the explicit vehicle dynamic model in the prediction of the particle filter.

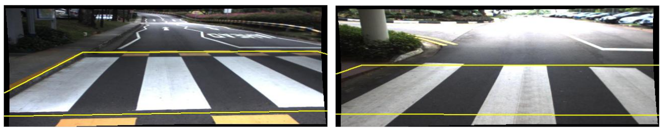


Figure 3.13: Incorrect fitting on lane lines due to far distance

The incorrect fittings from the proposed algorithm mainly happen when the visible line markings are far away from the camera. For the examples shown in Fig. 3.13, the line segments after the zebra crossing are very short in the image, and the algorithm fails to identify them. Under such incorrect fitting conditions, the system will make use of the previous road profile, vehicle dynamics and the particle filter to estimate the vehicle position.

3.3.3 Qualitative results on lane line detection

To further illustrate the capabilities of the proposed algorithm, some lane line detection examples are shown in Fig. 3.14 under different conditions. The green lines represent the detection results and blue dots represent the points used to work out the ultimate road model.

Fig. 3.14(a) to (d) show that even the road surface intensity is not uniformly distributed (strong contrasts), the algorithm is able to remove noise pixels and fit the correct line, regardless of its shape (straight, curve or zigzag). Fig. 3.14(e) to (h) indicate that although the number of lane line marking pixels under those situations are very few, the algorithm manages to identify them correctly. Fig. 3.14(i) and (j) demonstrate its capability to fit sharp turns, even when the line markings are not continuous. The last two images show the fitting performance on non-flat road surfaces.

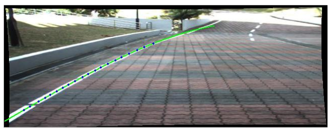
3.3.4 Results on lane-level localization

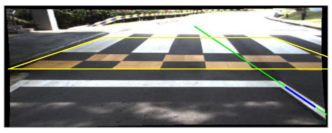
In this section, we used the same side camera method as describe in Section 2.5.4 Chapter 2 to generate the ground truth for d_e and θ_e. However, due to

3.14.a: Zigzag line fitting

3.14.b: Irregular shadows

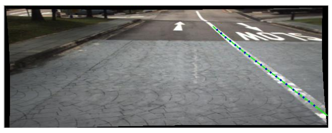
3.14.c: Camera saturation

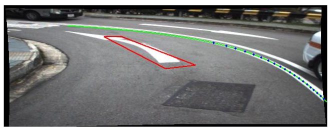

3.14.d: Different surfaces

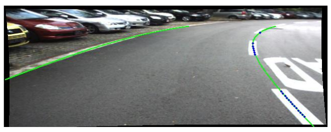

3.14.e: Limited visible line

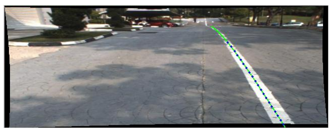
3.14.f: Fitting over hump

3.14.g: Occlusion


3.14.h: Worn-off line


3.14.i: Turn (continuous line)


3.14.j: Turn (dotted line)


3.14.k: Uphill

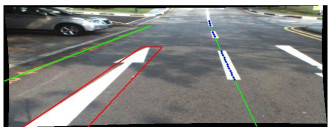

3.14.l: Down slope

Figure 3.14: Lane line detection in challenging situations

the inconsistent feature of zigzag lines, their ground truth cannot be generated. Therefore, the quantitative analysis does not cover the performance on zigzag lines. A new set of image sequences were used in this section. It consists of 749 images in total, out of which 499 have ground truth. They were taken on the same road as shown in Fig. 3.10 when driving from point A to point B during normal daytime.

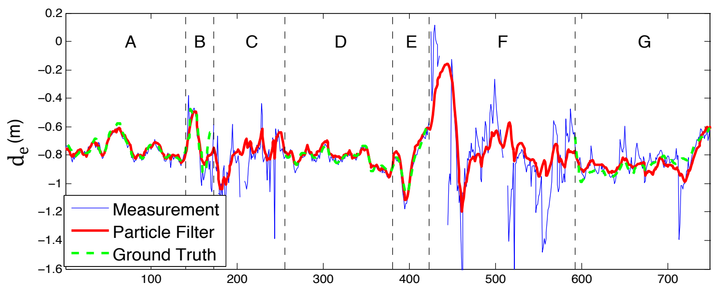


Figure 3.15: Estimation results on d_e. A & D: straight road. B: sharp turn. C & F: zigzag. E & G: moderate turn.

The testing results are illustrated in Fig. 3.15 and 3.16. In both plots, the green line refers to the ground truth from the side camera, the blue line refers to

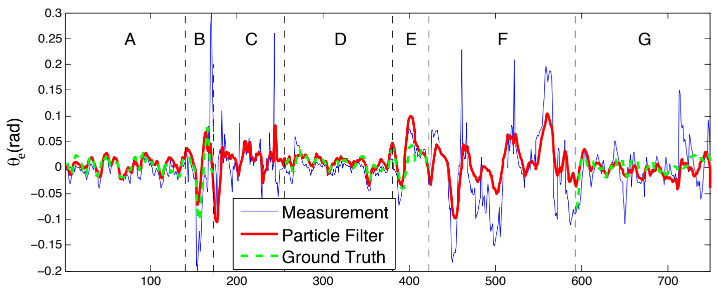


Figure 3.16: Estimation results on θ_e.

the results measured/derived directly from the image and the red line refers to the results from the particle filter.

For straight road (A & D), the direct measurement (DM) and the particle filter (PF) achieve similar results in both d_e and θ_e. They are very close to the ground truth with average abs d_e error $2cm$ and average abs θ_e error $0.01rad$.

For curve road (B ,E & G), most of the time, the particle filter performs better than direct measurement. The direct measurement exhibits inconsistency in estimation as intermittent spikes occur, especially in the θ_e estimation. The d_e standard deviation (STD) is $9cm$ for DM and $5cm$ for PF. The θ_e STD is $0.043rad$ for DM and $0.022rad$ for PF. The maximum abs d_e error from PF is only $16cm$, which is similar to the width of lane line markings and the maximum abs θ_e error is only $0.06rad$.

For zigzag lines (C & F), although there is no ground truth, the particle filter achieves smoother and more consistent estimation than direct measurement.

In overall, the estimation error in d_e has an absolute mean of $0.029m$, an STD of $0.039m$ and a 95% confidence interval between $-0.024m$ to $0.084m$. The estimation error in θ_e has an absolute mean of $0.011rad$, an STD of $0.016rad$ and a confidence interval between $-0.013rad$ to $0.035rad$.

To further verify the algorithm, it is compared with the ALD algorithm in [34]. The d_e and θ_e can be derived based on its line fitting parameters (ρ & θ in the referred paper) and IPM image. Before applying ALD, the images were pre-processed by the special detectors to remove the impact from those non-lane line markings. The comparison results are illustrated in Fig. 3.17 and 3.18.

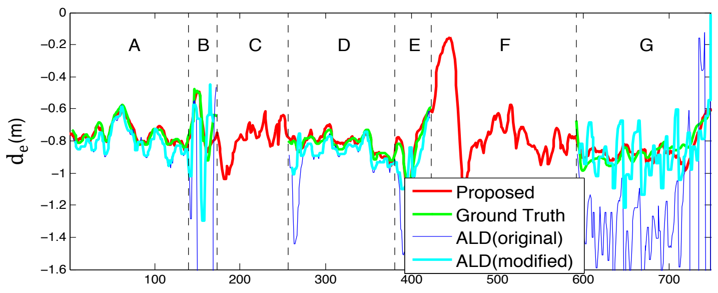


Figure 3.17: Comparison results between the proposed and *ALD* algorithms on d_e.

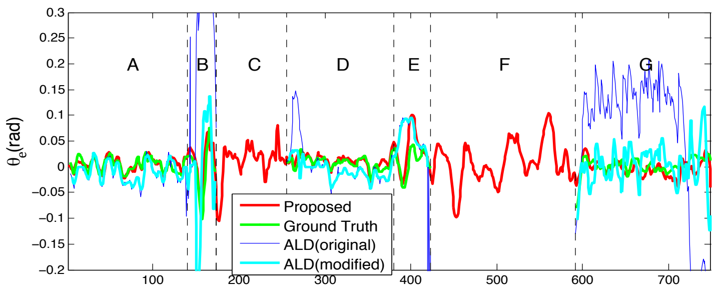


Figure 3.18: Comparison results between the proposed and *ALD* algorithms on θ_e.

However, the original *ALD* algorithm uses straight line model only and when fitting the curve road, it has obvious deviations as shown by the blue line ($ALD(original)$) in both figures. Therefore, improvements are made so that the original *ALD* algorithm is able to fit parabola model for curve road and the results are depict as cyan lines ($ALD(modified)$).

As shown in both d_e and θ_e plots (Fig. 3.17 and 3.18), the proposed algorithm is able to achieve more accurate and smoother results. The estimation from the proposed algorithm is more consistent and reliable. The main reason for the spikes in *ALD* is due to constant camera pitch angle assumption in obtaining the IPM. However, this assumption is not valid as the camera pitch angle varies with vehicle acceleration and uneven road surface (e.g. at humps).

Table 3.3: Accuracy and consistency comparison between MC and stereovision

	$d_e(m)$		$\theta_e(rad)$	
	MC	*Stereo*	*MC*	*Stereo*
Mean	0.002	-0.001	-0.013	-0.002
Mean Abs	0.056	0.029	0.025	0.011
STD	0.081	0.039	0.032	0.016

As compared with the MC method in Chapter 2, again, the stereovision system proposed here has clearer advantages in both accuracy and consistency. The comparison results are tabulated in TABLE 3.3. The stereovision system has smaller absolute errors and smaller standard deviations for both d_e and θ_e.

In summary, the proposed algorithm is able to provide accurate estimation of the vehicle lane-level localization. The particle filter is necessary to improve its estimation consistency and accuracy, especially for curve road.

3.3.5 Results at different times of the day and weather

To further evaluate the competencies of the system, it has been tested at different times of the day and under different weather conditions.

For the special detectors, the performances at dusk are similar to that during normal daytime. At some locations, they perform better because tree shadows become obscure at dusk.

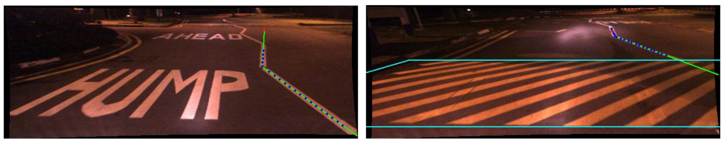


Figure 3.19: Road markings detection results at night

At night, the visibility becomes very poor. Due to the fixed intensity threshold used in the detection algorithms, warning letters and arrows cannot be detected at most of the time as shown in Fig. 3.19 left. (The word "HUMP" was not detected.) This is the limitation of the proposed algorithm. While the hump and zebra crossing detections are much less affected by the poor visibility as their detections are based on intensity patterns instead of values (Fig. 3.19 right).

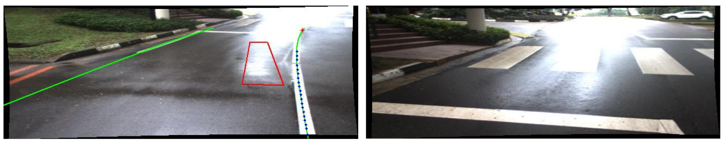


Figure 3.20: Incorrect road markings detection due to wet surface

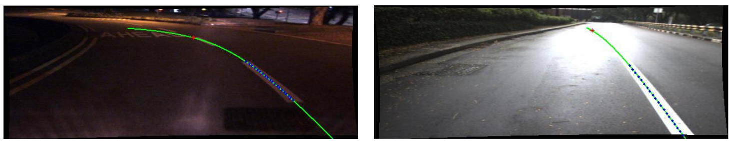
After rain, the wet road surface may induce glare effects in the image and cause some intermittent incorrect detections. For example, in Fig. 3.20 left image, the algorithm misclassified the glare to the arrow sign, while in Fig. 3.20 right image, the zebra crossing is not detectable due to the glare.

Table 3.4: Localization accuracy at different times of the day and weather

		Normal	Dusk	Night	After Rain
$d_e(m)$	Mean	-0.001	-0.004	-0.002	-0.006
	Mean Abs	0.029	0.053	0.051	0.043
	STD	0.039	0.074	0.072	0.059
	Max Abs	0.160	0.258	0.231	0.233
$\theta_e(rad)$	Mean	-0.002	-0.002	-0.003	-0.003
	Mean Abs	0.011	0.022	0.022	0.019
	STD	0.016	0.030	0.030	0.027
	Max Abs	0.063	0.122	0.120	0.122

For the lane-level localization accuracy, the comparisons of d_e and θ_e are summarized in TABLE 3.4. The ratio of images with ground truth to the total number of images are 590/859, 474/727, and 488/712 for the case of dusk, night and after rain respectively.

Under normal daytime condition, the system achieves the best performance in both accuracy and stability. For the cases at dusk and night, the lighting condition is the main reason causing the deteriorations as some parts of the lane line markings at far distance become invisible (Fig. 3.21(a)) at dusk and night. For the case after rain, the glare from wet surface may cause inadequate fitting as illustrated in Fig. 3.21(b).

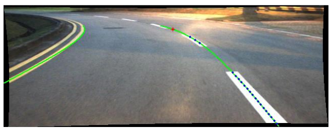


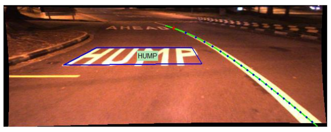
3.21.a: Poor illumination 3.21.b: Glare from wet surface

Figure 3.21: Inadequate lane line fitting results

However, the impacts from poor lighting condition and wet surface are not severe. The proposed algorithm is still able to function adequately. The errors in both d_e and θ_e are small and tolerable. The largest error in d_e is only $26cm$ and largest error in θ_e is $0.122rad$. From the standard deviations, it can be concluded that the estimation is able to maintain its stability and continuity.

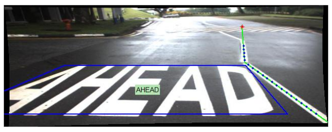
3.22.a: At dusk 1


3.22.b: At dusk 2



3.22.c: At night (street lights)

3.22.d: At night (head lights)



3.22.e: After rain 1

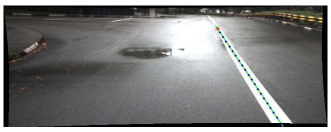

3.22.f: After rain 2

Figure 3.22: Lane line fitting results at dusk, night and after rain

Some sample images are depicted in Fig. 3.22 to demonstrate the lane line fitting results qualitatively. Image a and b illustrate the illumination conditions at dusk. c and d show the cases at night when the road was lit up by street lights and car head lights. The last two images show the cases after rain, when the road surface is wet and bright.

3.4 Chapter summary

In this chapter, built upon the mono-camera system in Chapter 2, a more advanced vehicle lane-level localization system is proposed. The system works based on stereovision. Through extensive on-field tests, it has been proven that the proposed system is able to estimate the vehicle localization information more accurately and robustly. More importantly, the system works in real time and has very few limitations in practice. It even works in dark nights.

Its high level of accuracy and consistent good performances under different conditions enable its implementation on the navigation of autonomous vehicles. The outputs (d_e, θ_e and road profile) from this stereovision system are served as the feedback to the NMPC control system, which is to be elaborated in the next chapter, to control the vehicle to move along the detected lane.

Without this high-performance stereovision system, the NMPC will suffer from the inaccurate feedback and not be able to control the vehicle properly regardless of how adaptive and advanced the controller is.

Chapter 4

Nonlinear Model Predictive Control on Autonomous Vehicles

4.1 Introduction and literature review

This chapter attempts to address the challenges lying in the field of autonomous vehicle control. The purpose is to control the vehicle to move along the detected lane accurately and smoothly. However, this control problem is nontrivial as a result of the nonlinear vehicle dynamics, highly coupled system inputs and non-differentiable constraints related to physical limitations and safety.

A number of control schemes have been proposed in the literature for such complex systems, varying from conventional PID control [74] to advanced adaptive ones, like back stepping control, sliding mode control (SMC) [75][49], fuzzy logic control [76], model predictive control (MPC) and etc. Among them, with the increment in computation power of computers, MPC has been shown to be a promising control scheme for such applications [77].

MPC is capable of systematically handling model nonlinearities, uncertainties and constraints [78] and it is able to optimize the current step while keeping the future steps and future trajectories into account. Its predictive ability distinguishes MPC from the rest of the control schemes aforementioned. However, the major concern in applying MPC is its computational burden to solve the optimization problem in real time, especially for nonlinear systems.

The common approach to compensate for its high computation demand in nonlinear systems is to linearize the nonlinear model around an operating point. Then the linearized MPC problem is fitted to certain formats, so that some fast and well developed convex optimization solvers can be implemented directly. For example, in [79], the vehicle model was linearized iteratively and customized to fit into the sequential quadratic programming (SQP) algorithm. In [80], a linearized time-state control form was adopted and the optimization was recast into SQP as well. One of the popular MPC optimization toolboxes is the multi-parametric toolbox (MPT) by ETH [81].

Common linearization techniques include linear time-varying (LTV) system [79][82] and piecewise affine (PWA) system [83]. Switching-based MPC is a variant of PWA [77], in which the piecewise region index is updated only at the initial step of the prediction horizon. Some examples can be found in [84] (Ford's active front steering) and [85].

However, a linearized model is only accurate within a small region around the operating point (a constant velocity and fixed steering angle). This leads to other limitations in applying MPC to vehicle control, for example, the approach in [86] is limited to low curvature roads. Moreover, the linearization is only applicable to single input system, either the velocity [83] or the steering.

To control both velocity and steering, several nonlinear MPC solvers were proposed in the literature, such as NPSOL [78], FP-SQP [87], OPTNOV [88] and so on. However, they have their own limitations. For example, NPSOL works only for polynomial nonlinearities. Moreover, the computation complexity in these nonlinear solvers significantly limits the number of maximum prediction cycles in MPC, which is critical for high speed applications.

Some attempts have been taken in the literature to reduce the computation time. The one in [89] separated the optimization into off-line and online components, where the optimization was carried out on the on-line part only, and thus reduced computation time. But the effectiveness of this approach depends on the separability of the constraints and reduces its flexibility for a wider range of applications [87]. In [90] and [91], the real time iteration algorithm, built upon direct multiple shooting method, was implemented. It reduces computation time dramatically to $\sim 20ms$ per iteration. But it requires the cost function and constraints to be at least twice continuously differentiable [92].

Furthermore, for both linear and nonlinear solvers, they are only useful when the model and constraints can be converted into particular formats and satisfy all the prerequisites (e.g. KKT condition). This may compromise or even prohibit the implementation of some complicated but important constraints.

The other set of algorithms to solve constrained optimization problem is the Genetic Algorithms (GA) and its variants. Over decades of development, GA has demonstrated its potential in solving real and practical applications in control systems, such as PID parameter fine tuning [93], fuzzy logic [94] and so on. However, as mentioned by P.J. Fleming [95] and F. Manenti [88], the computation complexity of GA is the main impediment to real-time applications, same for the nonlinear solvers.

Due to this, only a few applications of GA to nonlinear MPC are available in the literature. Most of them are applicable to slow processes and the results are based on simulations only. For example, in [96], the GA-based MPC is implemented in the autopilot design of an underwater vehicle (UV), but the UV runs at $2knots$ $(1.03m/s)$ only.

As compared with other nonlinear solvers, GA has its considerable advantages. It is not limited by the typical control problem attributes (e.g. convexity, polynomial nonlinearity) and has a very flexible structure [95]. The searching range of input variable constraints can be easily incorporated in the search space of GA during optimization [97]. Furthermore, it can be parallelized to further reduce its computation time.

Leveraging on these gaps in current MPC schemes, in this chapter, we propose a fast nonlinear MPC scheme to control the vehicle velocity and steering simultaneously to follow a reference path at high speed. The optimization is carried out through customized GA, which is designed to run at real time.

The implementation of GA gives the flexibility to design and formulate the MPC scheme in a more accurate, meaningful and direct manner. One need not worry about the convexity, linearity and continuity of the formulation. Benefitting from this flexible structure, the control signal smoothness is explicitly enforced by setting constraints in the MPC scheme. More importantly and novelly, the passengers' safety and their comfort are specifically taken into considerations in the MPC formulation while they are not found in other state-of-the-art MPC approaches. Safety and comfort are two most important indices in autonomous driving vehicles. Taking them into account is the main contribution of this chapter. The results show that the vehicle exhibits certain kinds of behaviours that mimic a human driver's.

For the remaining parts of the chapter, it is organized as follows: the formulation of the nonlinear MPC, with all physical and safety constraints, is elaborated in Section 4.2 in details. The GA design is depicted in Section 4.3, together with some specific improvements on the computation efficiency to make it real time. The results and discussions based on both simulations and on-field tests are illustrated in Section 4.4 and conclusions are drawn in Section 4.5.

4.2 Nonlinear MPC formulation

To implement nonlinear MPC, the non-holonomic vehicle kinetic model (Ackermann model) is adopted as shown in Chapter 3 Fig. 3.8(a). In some other research works [78][79][86][98], a more complicated dynamic model (Pacejka tire model) was implemented and some even utilized both [82]. We did not follow this because we wanted to keep the model less complex so that we could have more computation resource to implement safety and comfort constraints. Furthermore, IMU (Initial Measurement Unit) in addition to velocity and steering encoders is required in the Pacejka model. The Ackermann kinetic model given in (3.26) can be expressed compactly as:

$$\xi_{k+1} = f(\xi_k, T_s, U_k) \tag{4.1}$$

where $\xi_k = [x_k,\ z_k,\ \theta_k]^T$ defines the vehicle pose in the global coordinate system; $U_k = [v_k,\ \phi_k]^T$ refers to the inputs of the system and is defined as:

$$\begin{cases} v_k = v_{k-1} + \delta v_k \\ \phi_k = \phi_{k-1} + \delta \phi_k \end{cases} \quad or \;\; U_k = U_{k-1} + \delta U_k \tag{4.2}$$

For path following, both the distance and moving direction errors (d_e and θ_e) respected to the reference path (Fig. 3.8(b)) should be minimized. Same as Chapter 2 and 3, the reference path within the prediction horizon is approximated as a parabola in (4.3), where $(a,\ b,\ c)$ are the parabola parameters.

$$X = aZ^2 + bZ + c \tag{4.3}$$

Given the vehicle pose $(x_k,\ z_k,\ \theta_k)$ and the shape of the reference path in the global coordinate, the corresponding distance and moving direction errors can be worked out analytically by solving a third order polynomial based on geometry. This process is denoted as $g(\cdot)$ as shown in (4.5d) with $E_k = [d_{e_k},\ \theta_{e_k}]^T$.

This process will not be possible without the high-performance stereovision system proposed in Chapter 3. Other approaches mainly use IMU and odometry to calculate E_k, suffering from accumulative errors. The stereovision guarantees the accuracy in E_k, which is prerequisite for NMPC to work properly.

$$\begin{aligned} J = &\sum_{i=1}^{H_p} \left(w_1 ||d_{e_{k+i}}||^2 + w_2 ||\theta_{e_{k+i}}||^2 \right) + \sum_{i=0}^{H_c-1} \left(w_3 ||\delta\phi_{k+i}||^2 + w_4 ||\delta v_{k+i}||^2 \right) \\ &+ \sum_{i=1}^{H_p} \left(w_{acc_{k+i}} ||acc_{k+i}||^2 + w_{in_{k+i}} ||\delta v_{k+i}||^2 + w_{v_{k+i}} ||v_{max} - v_{k+i}||^2 \right) \end{aligned} \tag{4.4}$$

The L2-norm cost function at time instant k is constructed as (4.4), where

w_1, w_2, w_3 and w_4 are the prescribed weighting factors to reflect the importance of individual terms, while $w_{acc_{k+i}}$,$w_{in_{k+i}}$ and $w_{v_{k+i}}$ are on-off weighting factors varying with the vehicle states as defined in (4.5m), (4.5n) and (4.5o). H_c and H_p are control horizon and prediction horizon respectively and $H_c \leq H_p$.

$$\min_{\delta U_k,...,\delta U_{k+H_c-1}} (J) \tag{4.5a}$$

$subject\ to:$

$$\xi_{k+1} = f(\xi_k, T_s, U_k) \tag{4.5b}$$

$$U_k = U_{k-1} + \delta U_k \tag{4.5c}$$

$$E_k = g(\xi_k, a, b, c) \tag{4.5d}$$

$$\delta U_{k+m} = 0,\ \ for\ \ m = H_c, ..., H_p \tag{4.5e}$$

$$acc_{cen} = v_{k+i}^2 \tan(\phi_{k+i})/l \tag{4.5f}$$

$$acc_{tan} = \delta v_{k+i}/T_s \tag{4.5g}$$

$$acc_{k+i} = \sqrt{acc_{cen}^2 + acc_{tan}^2} \tag{4.5h}$$

$$-\delta v_{max} \leq \delta v_{k+i} \leq \delta v_{max} \tag{4.5i}$$

$$-\phi_{max} \leq \phi_{k+i} \leq \phi_{max} \tag{4.5j}$$

$$-\delta\phi_{max} \leq \delta\phi_{k+i} \leq \delta\phi_{max} \tag{4.5k}$$

$$-\delta\omega_{max} \leq \delta\phi_{k+i} - \delta\phi_{k+i-1} \leq \delta\omega_{max} \tag{4.5l}$$

$$w_{acc_{k+i}} = \begin{cases} 0, & \text{if } acc_{k+i} \leq acc_{max} \\ w_{acc_0}, & \text{otherwise} \end{cases} \tag{4.5m}$$

$$w_{in_{k+i}} = \begin{cases} w_{in_0}, & \text{if } condition1 \\ 0, & \text{otherwise} \end{cases} \tag{4.5n}$$

$$w_{v_{k+i}} = \begin{cases} 0, & \text{if } condition1 \\ w_{v_0}, & \text{otherwise} \end{cases} \tag{4.5o}$$

$$\begin{aligned} condition1 = {} & (\delta v_{k+i} > 0)\ \&\ \big((acc > acc_{max}) \\ & \mid (|d_{e_{k+i}}| > 0.3m) \mid (|\theta_{e_{k+i}}| > 0.09rad)\big) \end{aligned} \tag{4.5p}$$

The first summand in (4.4) indicates the penalties on path tracking errors and the second emphasizes control input smoothness. The *acc* term (4.5h) in the last summand, which refers to the vehicle acceleration, penalizes high acceleration for safety purpose; the δv term prevents the velocity from increasing when dangerous conditions occur (as defined in (4.5p)) and the last term controls the vehicle to travel at a prescribed cruise speed v_{max} under the safe condition.

The optimization problem can be formulated as (4.5), where $[\delta U_k,\ \delta U_{k+1}...,\ \delta U_{k+H_c-1}]$ is a sequence of control actions that minimize J. The control actions from H_c to H_p are assumed to be 0 (4.5e). A longer H_p than H_c is desired to yield smoother control signals as the portion from H_c to H_p servers as a kind of stabilizer. It forces the vehicle to reach a position with small errors before H_c and/or move in the direction of reducing errors. If the vehicle is to the wrong direction at H_c, the cost function will be inflated from H_c to H_p.

The vehicle centrifugal and tangential accelerations are defined as acc_{cen} and acc_{tan} respectively while acc_{k+i} is the resulted overall acceleration. δv_{max} and $\delta\phi_{max}$ are the maximum achievable velocity change and steering change in T_s. ϕ_{max} is steering physical limit corresponding to its full lock position. $\delta\omega_{max}$ is the maximum allowable change in consecutive $\delta\phi$, equivalently it reflects the maximum angular acceleration of the steering wheel.

The safety and comfort of human passengers are specifically taken into account in this formulation:

(i) High vehicle acceleration (both centrifugal and tangential) is penalized heavily so that the resulted acceleration is less than the max acceleration defined in [99]. This can prevent drifting of the vehicle and motion sick of passengers. It is realized by (4.5m).

(ii) It tends to slow down the vehicle when the acceleration is too large or the deviations to the reference path are large. Because under this condition, $w_{in_{k+i}}$ will be turned on to penalize positive δv_{k+i}; and $w_{v_{k+i}}$ will be turned off so that J will not be penalized even the current speed is less than the cruise speed. This is a precaution action to keep the vehicle on track and it is realized by (4.5n), (4.5o) and (4.5p).

(iii) The angular acceleration of the steering wheel is well confined by (4.5l) to prevent sudden and large changes in the steering, which may otherwise induce strong shaking.

From the set of constraints in (4.5), we manage to implement and realize the safety and comfort concepts without any compromise or approximation.

4.3 GA design

The nonlinearities of the scheme include the trigonometric functions, squares, square root, on-off weights and the combinations of them. To the best of our knowledge, no other solvers except GA are able to solve this complex, high dimension and discontinues optimization problem. As reviewed in Section 4.1, the design of GA is well established in the literature, but the challenge here is to make it run in real time. The overall design flow chart is illustrated in Fig. 4.1, which involves typical processes for standard GA.

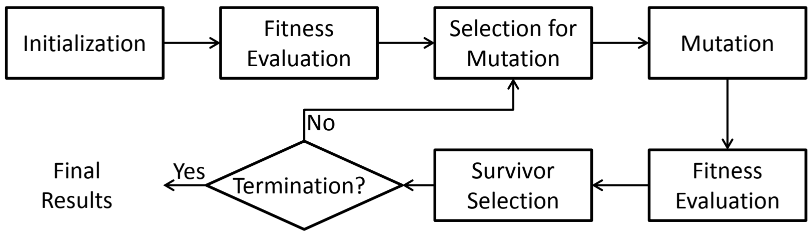


Figure 4.1: GA design flow chart

To make the GA suitable for real time application, new features are designed:

(i) No crossover operation is implemented in the design since it is not a necessary step.

(ii) The constraints (4.5j) and (4.5l) are enforced in the Fitness Evaluation process instead of initialization or mutation processes. Otherwise, additional loops of condition checking have to be carried out.

(iii) Steering direction alignment algorithm is proposed and embedded in the Fitness Evaluation process. This algorithm facilitates the GA to obtain the optimal faster and makes the results more stable.

To further improve the computation speed, all the processes in the flow chart are carried out in parallel. For example, in Fitness Evaluation, *n* chromosomes are evaluated simultaneously, where *n* refers to the number of CPU cores.

4.3.1 Encoding

Each chromosome (or individual) o_p is a $2 \times H_c$ matrix, with the first row representing δv and the second row representing $\delta \phi$ in the control horizon. The total number of chromosome in the population is N_{pop}.

$$o_p = \begin{bmatrix} \delta v_1 & \delta v_2 & ... & \delta v_{H_c} \\ \delta \phi_1 & \delta \phi_2 & ... & \delta \phi_{H_c} \end{bmatrix}, \quad p = 1, 2, ..., N_{pop} \tag{4.6}$$

Each gene (δv_i or $\delta \phi_i$) in all the chromosome population follows the constraints defined in (4.5i), (4.5k) and (4.5l).

4.3.2 Initialization

A suitable initialization procedure at the beginning of each MPC cycle is essential for a better and faster optimization result. At first, each gene in every chromosome is assigned by randomly picking a floating number within the intervals defined in (4.5i) and (4.5k). (4.5j) and (4.5l) are only enforced in Fitness Evaluation process as aforementioned.

The best chromosome from the previous optimization cycle also plays a role in the initialization process of the current cycle. Each gene in the previous best chromosome is shifted to the left by one position and the last position is patched randomly within the intervals in (4.5i) and (4.5k). 20% of the initialization population is generated from this chromosome with the last position being different. This facilitates the GA to exploit based on the previous best knowledge so that it may reach at the optimum faster. This also enforces the stability of the GA results. Whereas the rest of the population, which is generated randomly, creates diversity and possibility in the searching space to prevent the GA from being trapped at local optimum.

4.3.3 Selection

Two stages of selection are required, one is the selection for mutation and the other is survivor selection.

The selection for mutation selects individuals or chromosomes which go to the mutation process. To prevent the good individuals from taking over the entire population rapidly and maintain a suitable selection pressure, deterministic q-tournament selection is implemented.

First, q individuals are selected at random from the current population (parent pool). Then their fitness is compared and the best out of the q individuals is selected and copied to the intermediate pool, on which the mutation will act. All the q individuals are then returned to the parent pool and may be selected again. The process is repeated N_{pop} times until the population in the intermediate pool reaches N_{pop}. The mutation will act on this intermediate pool and the results are copied into the child pool.

The survivor selection is to select N_{pop} chromosomes out of the $2N_{pop}$ after mutation. The $2N_{pop}$ consist of the parent and child pool. In order to maintain the diversity in the subsequent generations, all parents are replaced by children.

4.3.4 Mutation

Mutation acts on the gene of individuals in the intermediate pool. At each gene, a random number between 0 and 1 is picked uniformly and compared with the mutation rate p_m that is associated with the particular individual/chromosome. If it is less than p_m, that gene undergoes mutation, meaning the gene will be replaced by a new random number picked uniformly within the range of (4.5i) or (4.5k) accordingly. If the mutated gene is $\delta\phi$, only its value will change but its sign remains. The reason will be explained later in the Fitness Evaluation.

To prevent good individuals from being destroyed by mutation and expedite the optimization process, we adopt the self-adaptive mutation rate. p_m is associated to each individual. It self-adapts each time when the corresponding individual is selected to mutate or the mutation rate itself undergoes mutation process. The new mutation rate p'_m is given by

$$p'_m = (1 + (1/p_m - 1)e^{-\gamma N(0,1)})^{-1} \tag{4.7}$$

where e is the exponential operator, γ is the learning rate (set at 0.2) and $N(0,1)$ represents a random number picked from normal distribution.

4.3.5 Fitness evaluation

The fitness F of an individual is defined based on the cost function J. This is the most time consuming part in GA.

$$F = 1/(1+J) \tag{4.8}$$

As aforementioned, constraints (4.5j) and (4.5l) are enforced here to save computation time. But before checking these constraints, steering direction alignment is carried out first. This process facilitates the GA to arrive at the optimal faster and makes the results stabler. However, to maintain the diversity, only half of the population is randomly selected and undergoes this process.

When the vehicle is on the left side of the road and moving towards to the road left boundary (Fig. 4.2(left)), in order to bring the vehicle back to the center and align it to the tangent direction of the road, the steering angle has to be smaller than the desired one ϕ_d. Based on the sign convention in Fig. 3.8, smaller means to steer more towards the right. ϕ_d corresponds to the curvature k of the road, $\phi_d = \arctan(l/k)$, at which the vehicle can follow the road arc exactly if starting from the correct direction.

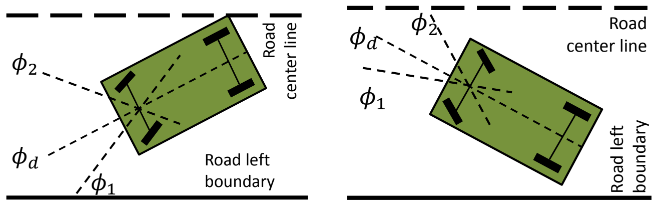


Figure 4.2: Vehicle pose illustration for steering angle alignment

For the instance shown in Fig. 4.2(left), if the current steering position is at ϕ_1, $\delta\phi$ needs to be negative. If $\delta\phi$ is positive, its sign is changed but its magnitude remains. If the current steering position is at ϕ_2, $\delta\phi$ can be positive or negative. However, if $\delta\phi$ is positive and results in a steering position larger than ϕ_d, its sign is kept but its magnitude is changed to a uniformly picked number from $[0,\ \phi_d - \phi_2]$ to keep the resulted steering position less than ϕ_d.

For the instance shown in Fig. 4.2(right), to bring the vehicle to the center and align it to the tangent direction of the road, the steering angle needs to be larger than or equal to ϕ_d. Similar alignment procedures can be implemented. If the vehicle is on the right side of the road, the same concept applies as well.

The next step in Fitness Evaluation is to ensure the constraint (4.51) is fulfilled. If not, $\delta\phi_{k+i}$ needs to be updated as following based on its sign.

If $\delta\phi_{k+i}$ is less than or equal to zero

$$\delta\phi_{k+i} = \begin{cases} R(-\delta\omega_{max}, \delta\omega_{max}) + \delta\phi_{k+i-1}, & \text{if } \delta\phi_{k+i-1} + \delta\omega_{max} < 0 \\ \delta\phi_{k+i-1} - \delta\omega_{max}, & \text{else if } \delta\phi_{k+i-1} - \delta\omega_{max} > 0 \\ R(\delta\phi_{k+i-1} - \delta\omega_{max}, 0), & \text{else} \end{cases} \tag{4.9}$$

where $R(x,y)$ defines a uniformly picked number in the interval $[x,\ y]$. The new $\delta\phi_{k+i}$ under the first and last conditions satisfies (4.51) and maintains its original sign. But under the second condition, we choose to sacrifice the sign of $\delta\phi_{k+i}$ in order to fulfil the more important constraint (4.5l). The new value for $\delta\phi_{k+i}$ is the nearest possible value from the interval constrained by (4.5l) to its original sign. If $\delta\phi_{k+i}$ is greater than zero, similar update mechanism can be derived.

When applying (4.9), the new $\delta\phi_{k+i}$ is counter checked with (4.5k) and capped by the limits in (4.5k). If the resulted steering position ϕ violates constraint (4.5j), the limits ($\pm\phi_{max}$) is used in the calculation instead of ϕ.

4.3.6 Termination

This determines when the GA should stop and return the best individual leading to the minimum of J. The termination condition is a trade-off between the best optimal solution and computation time. In order to make sure the MPC works properly, a sub-optimal solution will be returned if the time taken is too long. In particular, the GA will terminate when the optimal solution is found or the time elapsed is reaching $1.1T_s$ whichever is earlier.

4.4 Results and discussions

In order to evaluate the performance of the proposed GA-based nonlinear MPC control scheme, both simulations and on-field tests have been carried out. The control scheme is implemented on the NI PXI as listed in TABLE 1.1.

Table 4.1: Parameters in MPC setup

w_1	0.8	H_c	15	δv_{max}	$0.05m/s$	w_{acc_0}	0.5
w_2	1.5	H_p	20	$\delta\phi_{max}$	$0.02rad/s$	w_{v_0}	5.0
w_3	2.0	l	$1.28m$	$\delta\omega_{max}$	$0.015rad/s^2$	N_{pop}	40
w_4	2.0	T_s	$0.1s$	ϕ_{max}	$0.40rad/s$	p_{m_0}	0.1
V_{max}	$20m/s$	acc_{max}	$1.5m/s^2$	w_{in_0}	2.0	λ	2.0

The MPC parameters are summarized in TABLE 4.1. w_1 to w_4 were tuned based on the following guidelines and simulation trial and error. First w_1 should not be larger than the rest as d_e generally has much larger value than θ_e, $\delta\phi_{max}$ and δv_{max}; Otherwise, the optimization will be biased by the d_e term. Second, when the vehicle is off from the lane center or moves towards to the boundaries ($|d_e| > 0.3m$ or $|\theta_e| > 0.09rad$), the pose of the vehicle (the first summand in (4.4)) is more important than the control smoothness (second summand), so $0.3^2 w_1 + 0.09^2 w_2$ needs to be larger than $w_3 \delta\phi_{max}{}^2 + w_4 \delta v_{max}{}^2$. Third, when the vehicle moves along the lane center with small d_e and θ_e, we need to ensure $w_1 d_e^2 + w_2 \theta_e^2$ can be smaller than $w_3 \delta\phi_{max}{}^2 + w_4 \delta v_{max}{}^2$ so that (4.4) focuses more on control smoothness.

4.4.1 Simulation

The simulation platform consists of three major parts, namely road map, the vehicle simulator and the MPC controller. The road map is built according to the real road around the university engineering buildings with a total length of $1.8km$. The lane width is assumed to be constant, $3.5m$. It consists of straight line segments, bends with different turning radiuses and even a double bend at position C as shown in Fig. 4.5.

The road curvatures are not polished or smoothed as shown in Fig. 4.4(left). For example, at one particular corner, its turning radius may be varying instead of being smoothed to a constant arc. The turning radius indicated in Fig. 4.5 refers to the minimum value inside the turning corner. They show how sharp the turns are. All these varying curvatures and sharp turns induce challenges to the path following control, but they reflect the real world scenarios and make the simulation more realistic. While in some other research works, only simple paths are used. For example, lane change path, obstacle avoidance path and overtaking path are used in [100] [101] and [98] respectively on straight road. They only require at most four manoeuvres to finish the task.

The vehicle simulator is designed according to (4.1) and (4.2). The dimensions are adopted from the Toyota COMS EV as introduced in Section 1.5. The steering motor and driving motor dynamics are also taken into account. Given the desired input, the output from the steering motor and driving motor can be approximated by two second order systems. These two systems are identified based on the real data collected from the Toyota COMS EV.

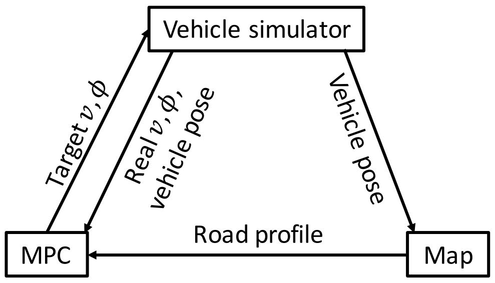


Figure 4.3: Simulation flow diagram

The detailed flow diagram is illustrated in Fig. 4.3. The simulator takes the target velocity and steering from MPC as the vehicle inputs. Then it works out the vehicle new real velocity, steering and vehicle pose (position and orientation). The sampling rate of the vehicle simulator is $100Hz$, which is much faster than MPC frequency ($1/T_s = 10Hz$). The map takes the vehicle pose as input, then it locates the vehicle in the map and outputs a parabola which approximates the road profile in front of the vehicle up to the prediction horizon. The MPC takes the road profile, the vehicle velocity, steering and pose as the inputs, and outputs the target velocity and steering to the vehicle simulator.

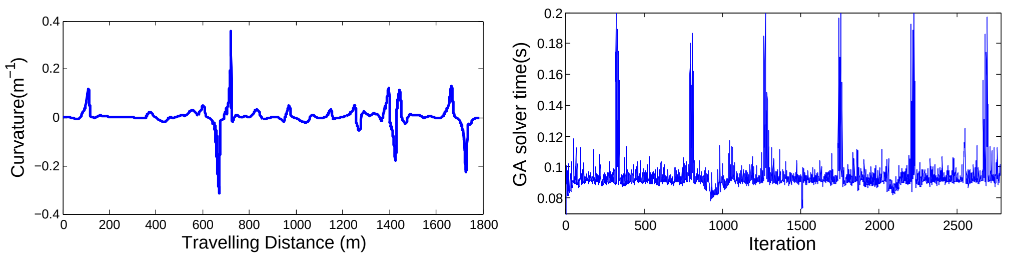


Figure 4.4: Road curvature and time taken for each MPC iteration

Fig. 4.4(right) shows the time taken for each MPC iteration during a typical run of the simulation. The average time per iteration is $0.098s$ which is less than T_s. Out of all the iterations, 95% is able to return the optimal solution before the time termination condition is triggered. For the remaining 5%, only sub-optimal solution is returned, but the control is able to work properly as shown later. Several spikes occur occasionally in the plot. This happens when the GA solver is stuck at the fitness evaluation process due to computer hang. Under such situations, the second control command in the previous optimal control sequence is passed to the vehicle.

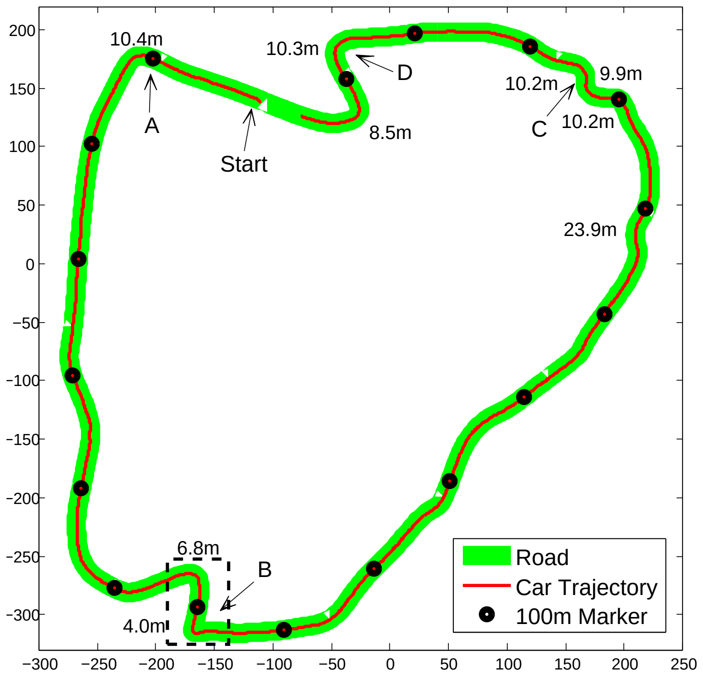


Figure 4.5: Vehicle travelling trajectory under the control of MPC

The red line in Fig. 4.5 illustrates the car travelling trajectory under the control of MPC. In overall, the controller is able to maintain the vehicle to travel along the center line of the road approximately. Even at those sharp corners, the vehicle remains travelling within its ego lane.

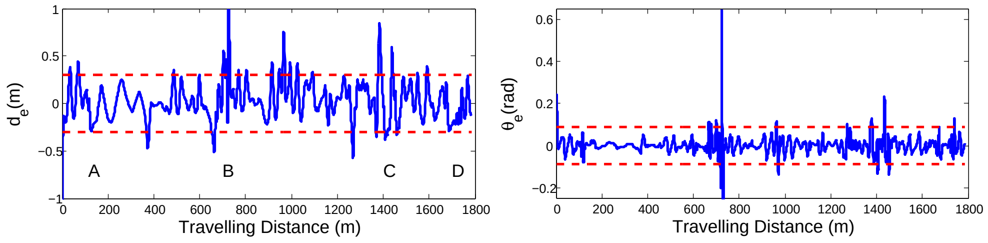
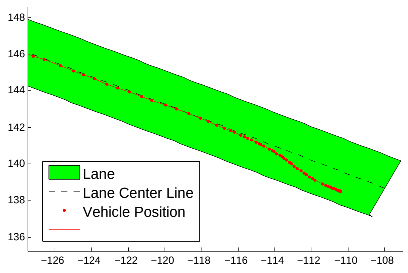


Figure 4.6: Tracking error (red dashed lines: safety boundaries)

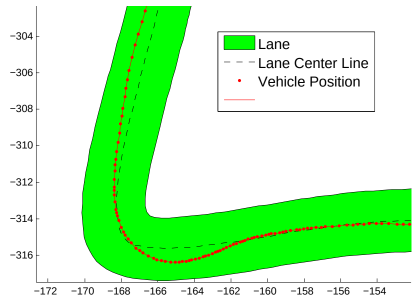
Fig. 4.6 demonstrates the tracking errors d_e and θ_e with respect to the center line of the ego lane. Most of the time (82%), the errors are within the safety bounds (red dashed lines, $|d_e| < 0.3m \textit{ and } |\theta_e| < 0.09$) regardless the change of road curvature. When it is outside of the safety bounds, the MPC can quickly adjust the vehicle back. On average, the absolute distance tracking error is $19.5cm$ and absolute moving direction error is $0.039rad$. The standard deviation in d_e is $27cm$ and standard deviation in θ_e is $0.072rad$. The 95% confidence intervals for d_e and θ_e are $[-0.35m,\ 0.48m]$ and $[-0.13rad,\ 0.14rad]$ respectively. These small values indicate that the algorithm is able to control the vehicle to follow the path consistently under different road curvatures and vehicle velocities.

Note that the safety boundaries are the designed criteria we want the system to achieve. They are set empirically.

The most prominent errors occur at the starting point and those sharp turning corners. For the starting point, we purposely set it at an off-center position to see whether the MPC is able to adjust the vehicle to the lane center as the vehicle moves. The trajectory in Fig. 4.7(a) (zooming in Fig. 4.5 at the starting point) shows how the vehicle is controlled to move to the lane center gradually.



4.7.a: Starting point



4.7.b: Corner B

Figure 4.7: Zooming in the trajectory at starting point & corner B

The errors at the sharp turning corners are mainly caused by two reasons. The first is that the road shape at the sharp corners cannot be adequately approximated by parabola. The second is that the vehicle overshoots and misses the turning point before it is fully slowed down. Fig. 4.7(b) (zooming in at corner B of Fig. 4.5) depicts these two reasons clearly, the error before the corner is due to inaccurate parabola model and the error after the corner is due to overshooting.

Fig. 4.8 illustrates the velocity and steering outputs under the control of MPC in each iteration. The behaviour is similar to that of a human driver, for

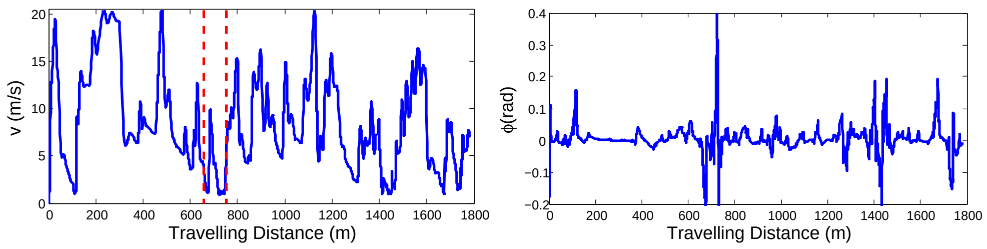


Figure 4.8: Vehicle velocity and steering output

example, the portion in between the dotted line which corresponds to the dotted area in Fig. 4.5. Before entering the first corner, the velocity starts to reduce and concurrently, the steering wheel turns to its right. Once passing through the corner, the steering wheel retrieves back to the center and the velocity increases. However, before the velocity reaches cruise speed ($20m/s$), the vehicle starts to decelerate again and turns to its left in order to move around the second corner. Once moving out of the corner, the steering wheel is quickly turned to its right and gradually returns to its center. Meanwhile, the velocity does not increase because the steering is at full lock position and the tracking errors are big. But when it returns to the lane center, the vehicle picks up its velocity quickly.

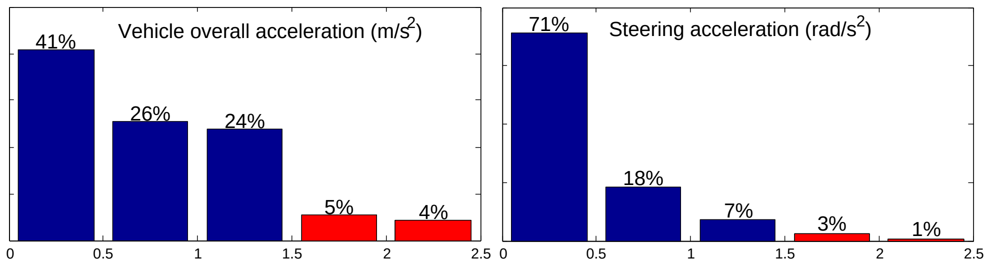


Figure 4.9: Vehicle moving acceleration and steering acceleration

The distribution of the vehicle overall acceleration (absolute value, resulted from centrifugal and tangential accelerations as shown in (4.5h)) and steering wheel acceleration (absolute) is shown in Fig. 4.9, where red bars refer to unpleasant accelerations. For the vehicle acceleration, 90.6% of the running time (blue bars) is less than acc_{max}, which is set at $1.5m/s^2$ according to [99]. Similarly, for the steering acceleration, when it is less than $1.5rad/s^2$ (96.4% of the time), the passengers will not feel the shaking effects in general.

For the red bars, they will induce certain level of discomfort to the passengers, but they are necessary in order to keep the vehicle within the safety bounds. The MPC chooses to sacrifice the vehicle/steering acceleration. It is the direct result from the weight assignment in the cost function J (4.4).

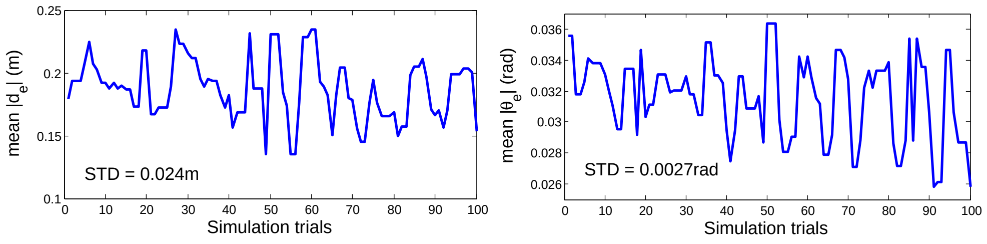


Figure 4.10: GA solver consistency analysis

As stated in Section 4.3, the GA solver contains several random processes. 100 simulations with the same initial conditions for a straight road segment ($\sim 80m$ long and $3m$ wide) were carried out to demonstrate the consistency of the proposed solver. The mean $|d_e|$ and $|\theta_e|$ for each simulation are depicted in Fig. 4.10. The largest difference among the 100 mean $|d_e|$ is $0.1m$, which is small compared to the road width and car size. The largest difference among the 100 mean $|\theta_e|$ is only $0.01rad$. The STD of the 100 mean $|d_e|$ is $0.024m$ and $|\theta_e|$ is $0.0027rad$. Both values are very small, and thus the optimization from the GA solver is consistent.

4.4.2 On-field tests

In the on-field tests, the stereovision system described in Chapter 3 was used to provide the road profile and the vehicle pose information to the nonlinear MPC controller. We did not test the vehicle on the same road as the simulation since parts of the road do not have lane line markings. The on-field tests were carried out at a different location. The mock up of the test field is shown in Fig. 4.11, with a total length of $140m$. It consists of bends with different turning radiuses. The road width is $3.0m$ approximately.

As shown in Fig. 1.15, the on-field test system comprises of three computing units. The first one is the NI (National Instruments) PXI, which runs Windows7

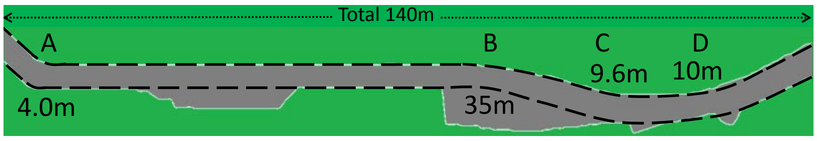
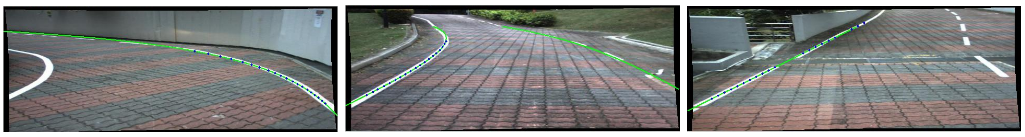


Figure 4.11: On-field test road profile

with Intel i5 CPU (same as the one for simulation) and carries out the nonlinear MPC related calculations, including the GA optimization. The second one is an Asus laptop, dedicated to perform the image processing for the stereovision. The last one is a low-level micro-computer, which runs Linux with i586 CPU and serves as an information exchange hub between NI PXI and low-level steering/driving motor controllers and sensors.

Fig. 4.12 illustrates the conditions of the testing field and the lane line detection results from the stereovision system. Fig. 4.12(a) corresponds to the bend with 4.0m turning radius. Fig. 4.12(b) is 10m turning radius and on an incline slope of 5°. Furthermore, part of the road surface is not flat as shown in Fig. 4.12(c). All these conditions add more challenges to the control scheme. Two illustration videos (link iii and iv) are provided in Appendix B.



4.12.a: Sharp turn 4.12.b: Moderate turn 4.12.c: Uneven road surface

Figure 4.12: Illustration on road conditions of test field (Green lines and blue dots indicate lane line detection results)

The tracking errors (d_e and θ_e) measured from the stereovision system are depicted in Fig. 4.13. They illustrate the vehicle performance when it travelled from left to right on the test field. As shown by the large errors at the beginning of the plot, the vehicle was purposely started from an off-center location and pointing to the road boundary.

The letters (A, B, C and D) correspond to the locations marked in Fig. 4.11. Before entering corner A, the vehicle managed to move into the safety boundaries from its initial position. Although the minimum turning radius at A is only

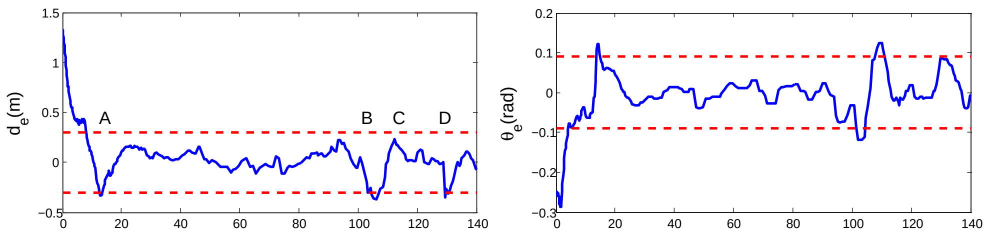


Figure 4.13: Test tracking errors (red dashed lines represent safety boundaries)

$4.0m$, the vehicle remained in the safety boundaries when manoeuvring around the corner. It also maintained a low speed as shown in Fig. 4.14, otherwise the acceleration (mainly from centrifugal acceleration) would exceed $1.5m/s^2$.

For the straight road between A and B, the vehicle picked up its speed at a constant acceleration and maintained at its cruise speed ($5.0m/s$). It made only minor changes on steering wheel to refine its travelling direction. The maximum cruise speed is limited by the speed regulation at the test field ($15km/hr$).

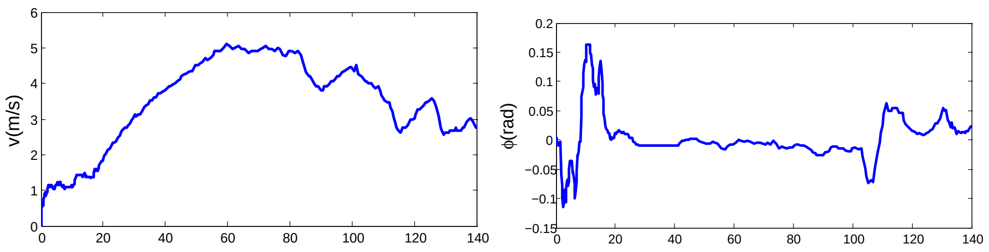


Figure 4.14: Vehicle velocity and steering output in an on-field test

The most prominent error occurs at the location around corner C. But it quickly adjusted the vehicle back to the safety boundaries. At the same time, it reduced the vehicle speed immediately for precaution. Without counting the initial part when the vehicle is moving from its initial position to safety boundaries, the average absolution distance tracking error is $11.4cm$ with STD of $13.4cm$ and confidence interval of $[-0.30,\ 0.25]$. The average absolute moving direction error is $0.027rad$ with STD of $0.039rad$ and confidence interval of $[-0.079,\ 0.082]$.

The vehicle overall acceleration and steering acceleration are illustrated in Fig. 4.15. For all the time, both the vehicle acceleration and steering acceleration are confined within the comfort regions. The passengers will not feel any

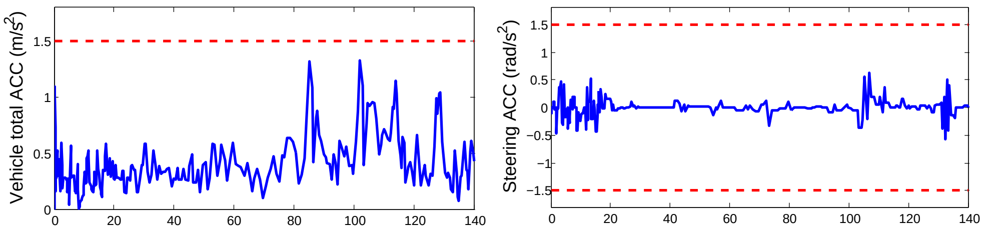


Figure 4.15: Vehicle & steering acceleration in an on-field test

shaking and jerking effect from the nonlinear MPC controller.

4.5 Chapter summary

In this chapter, we have proposed a nonlinear MPC to control a vehicle to follow a prescribe reference path at relatively high speed ($\sim 20m/s$). The velocity and steering direction are controlled simultaneously. We have also proposed a fast genetic-algorithm-based solver to solve the optimization problem, which allows more meaningful and direct design of MPC without any compromise. Benefitting from the flexible structure of the nonlinear MPC, human passengers' comfort and safety are able to be taken into account explicitly.

We have also built a realistic simulation platform to evaluate the proposed nonlinear MPC scheme. The results clearly show the competencies of the proposed MPC in controlling the vehicle to travel along the road center line. The vehicle is even able to maintain its position in its ego lane at sharp corners.

Extensive on-field tests (about 300) have been carried out with the help of stereovision system which provides road profile and tracking errors. The results tally with that from the simulation. The vehicle is able to manoeuvre to travel along the road center line and maintain its position within the safety boundaries. The results show that the control signal is smooth enough not to cause any jerking effect to the passengers.

Chapter 5

Vision-Based Autonomous Reverse Parking System

5.1 Introduction and literature review

Parking is a fundamental function for the autonomous vehicle. Only with the stereovision and NMPC systems proposed in the previous chapters, the AV is not able to park properly as there is no vision feedback from the back of the AV. This chapter proposes a new vision-based self-parking system. The detection of the parking slot is inherited from the ridge detector in Chapter 2 since it has been proven to be largely effective under normal conditions.

As mentioned in Chapter 1, some commercial car models have adopted, to a certain extent, various types of self-parking systems. However, most still require human intervention. For example, in the Toyota Prius [102] and BMW 7 series [103], the driver needs to regulate the speed of the vehicle by pressing and releasing the brake pedal. The system only takes control of the steering wheel. Thus, gaps still exist in the attempt to render a fully self-parking system.

Parking can be categorized into two major groups based on the parking slot orientation with respect to the road, namely reverse parking and parallel parking, while the focus of the chapter is on reverse parking. All concepts except path generation can still be applied directly to parallel parking. To realize autonomous parking, typically three basic steps are needed, including target position designation, path planning and path following/tracking.

The target position designation is to identify the available parking slot and track the desired slot during the parking process. Based on the sensors utilized, it can be roughly divided into three categories, including active ultrasonic or laser sensor based, infrastructure-based and vision-based methods.

The active sensor based method is the most common one for parallel parking. The system collects range data as vehicle passing by a free parking space and registers the range data using odometry to create depth map. Some typical examples can be found in [104], [105] and [106]. However, as pointed out in [107], this approach usually fails in reverse parking because the incident angle between the sensor and the side facets of nearby vehicles is too large. Moreover, the final parking accuracy depends on the nearby vehicle parking orientation.

The infrastructure based method requires modifications on existing car park systems. In [108], local GPS, car park digital map and communication with the parking administration systems are required. However, these approaches may not be practical in a short time due to the requirement of additional hardware installation on current car parks.

Vision-based methods provide the simplest and cheapest solution among these three categories. In [109] and [110], the vision system was used to recognize adjacent vehicles as the boundary of the slot while in [111], the author used fish-eye cameras and Canny edge detector to identify the slot markings.

However, conventional vision-based methods may not be robust enough to tackle various illumination situations in different types of car parks [106]. To deal with poorly lit environment which is common for indoor/underground car parks, a new system was proposed in [112] to recognize 3D information by analyzing the light stripe from a light plane projector. But it requires external references to work properly, such as vehicles in the adjacent slots.

Taking all these into consideration and based on successful experience gained from Chapter 2, we propose to use mono camera with ridge detector for the target position designation. As proven in Chapter 2, ridge detector is more robust than conventional edge detectors in identifying bright long structures in the image. More importantly, it functions well in poor illumination conditions. And since the parking slot image is less complex and less contaminated by noise as compared with the road image, the ridge detector can work properly enough to detect the slot markings. The detector proposed in Chapter 3 cannot be applied here directly without further adjustment because it is not designed to detect lines that are near horizontal orientation.

Similarly as Chapter 3, to improve the tracking consistency, filtering mechanism, like Kalman filter or particle filter, between consecutive images is desired. For this particular application, the vehicle moves slowly and on flat surface, thus there are no abrupt changes between consecutive images. This indicates that Kalman filter will work as robust as the particle filter. But it is less computation demanding; therefore, Kalman filter is preferable here.

In the domain of path planning for reverse parking, many continuous-curvature path generators based on geometrical information have been proposed in the literature. Pioneering works by Dubins [113] and Reeds [114] proved that the shortest paths for a car to move from one pose to another consist of arcs of circles (minimum turning radius) and straight line segments. Based on this concept, Massaki [108] and Daobin [115] proposed several algorithms to address this problem in their respective works. Some other researchers [116] adopted concepts of way points. Wang et al. [111] proposed double circular trajectory in which the first arc follows the minimum turning arc and the second one depends on the available space. But the key idea is still based on arc-line type of path.

A common disadvantage of their works is that a feasible path is not always guaranteed under the constraint of minimum turning radius. The success in finding a path depends strongly on the vehicle initial pose relative to the car park slot, for example, the double circular trajectory [111] requires the initial pose to be parallel to the road. M. B. Oetiker et al. proposed a different approach from navigation field point of view [117]. But the critical problem is to determine the way point and the available paths depend on a pre-calculated vector field.

Leveraging on these considerations, we propose an arc-line based path planning algorithm which guarantees to find a feasible path under any initial pose.

For the path following controller design, as reviewed in Chapter 4 there are many designs available varying from PID to more advanced adaptive ones, including sliding mode control (SMC) [118], input-output linearization [119], fuzzy-neural network [120], MPC [80], back-stepping [121] and so on.

Although MPC has the aforementioned advantages among these controllers, we did not use it here because it requires too much computation resources and is overly-complex for this parking system. The vehicle moves at low speed and follows a simpler path as compared with the real road.

For this parking system, we propose to apply SMC since it has the lowest computation complexity among the advanced controller, yet it is still able to achieve fast response and good transient tracking [122].

However, the conventional SMC also has drawbacks. As shown later, the control signal is not smooth when it is applied to non-holonomic vehicles. Several spikes arise which induce strong vibrations to the vehicle. To mitigate this, we improved the SMC design and managed to suppress these spikes.

In this chapter, a complete set of solutions to self-reverse parking system is proposed. The system utilizes a mono camera with Kalman filter to generate real-time feedback on vehicle pose. A supervision control scheme is designed as well to prevent the vehicle from entering adjacent parking slots and ensure the vehicle parks along the slot center line accurately. The system is proven, through experiment data, to be robust, reliable and practical. The main contributions of the work include:

(i) A robust path planning algorithm. It is proven that a feasible path can be generated regardless of the vehicle initial pose with respect to the slot.

(ii) A smooth SMC path following controller. It is able to eliminate the control signal spikes induced by conventional SMC and provide high level of human comfort without jerking or shaking.

The chapter is organized as follows: Section 5.2 and 5.3 provide detailed explanation on the main contributions of the chapter, including path generation and SMC. Section 5.4 explains the image processing and Section 5.5 elaborates the interactions between individual modules. The experiment setup and results are illustrated in Section 5.6, and conclusions are drawn in Section 5.7.

5.2 Path generation

It has been proven in [113] and [114] that the shortest path between two vehicle poses consists of straight line segments connected with circular arcs of minimum turning radius. Based on this result, we propose a novel path planning algorithm which guarantees the existence of a feasible path at any vehicle pose. Although the curvature profile at the transition points between arcs and lines is discontinuous [120], it is not critical as far as this application is concerned since the vehicle moves at slow speed $(1-2km/hr)$.

To better illustrate the algorithm, the coordinate system in Fig. 5.6 is defined with its origin at the center of bottom boundary. The last segment of all the paths must be a straight line along Z axis, otherwise, part of the vehicle body will have to pass through the forbidden areas before reaching the destination point.

For a path consisting of two segments, the first segment must be an arc. Depending on the vehicle moving direction and steering direction, there are four cases. Similarly, for a path consisting of three segments, the first segment has four cases as well and thus the total number of paths in this case is $4 \times 4 = 16$. Analogously, for a path consisting of n segments, each segment except the last has four cases and the number of possible paths is 4^{n-1}.

Given a vehicle pose, since the number of path segments is unknown, the program has to check from the path with one segment until n segments. Then the total number of paths to be examined equals $1+4+4^2+\cdots+4^{n-1}=(4^n-1)/3$. Based on our implementation, when the check is up to $n=4$, there will be at least one feasible path for 90.5% of the possible poses. Further increasing n, the total number of paths to check will increase by 256 and no explicit solution is available for a 5-segment path even the arc-line combination and sequence are fixed. The computation requirement increases dramatically. Therefore, it is not worth of trying $n \geq 5$ for the remaining 9.5% poses. A compensate solution will be proposed later to cover these remaining poses.

Table 5.1: Basic path models

Segment	Path	Number
n=1	SB	1
n=2	(RB/LB/RF/LF)-SB	4
n=3	(SB/SF)-(All cases with n=2), (RB/RF)-LF-SB, (LB/LF)-RF-SB	12
n=4	(SB/SF)-LB-RB-SB, (SB/SF)-RB-LB-SB	4

When $n=4$, the total number of paths is 85, out of which some are redundant especially for those with four segments. Therefore, instead of checking all 85 paths, we used 21 non-redundant paths as the basic set. The details are tabulated in TABLE 5.1. The first letter refers to steering direction (**S**traight, **L**eft, **R**ight) and second refers to moving direction (**F**orward, **B**ackward).

$$x_c = x_0 \pm R\sin\theta \tag{5.1}$$

$$z_c = z_0 \mp R\cos\theta \tag{5.2}$$

To facilitate the illustration on how to check paths, the following parameters are defined: the vehicle position $(x_0,\ z_0)$ is represented by the center point of its rear axel, its orientation θ is the angle from $+X$ direction to its body center axis (Fig. 5.1(b)), the car width is W_{car} and the parking slot width is W_{slot}. For a given car pose $(x_0,\ z_0,\ \theta)$, its turning center $(x_c,\ z_c)$ with radius of R is (5.1) and (5.2) according to its steering direction (Top signs are for right steering).

5.2.1 Paths with two segments

To check a path with two segments for $RB-SB$ as shown in Fig. 5.1(a), the path must satisfy the following conditions in order to avoid passing through the adjacent parking slots.

(i) $|x_1|$ needs to be less than $tolerance$, where $(x_1,\ z_1)$ is the transition point between the arc and the line. $tolerance$ defines the allowable offset from the parking slot center line. From a geometrical relationship, $x_1 = x_c + R$.

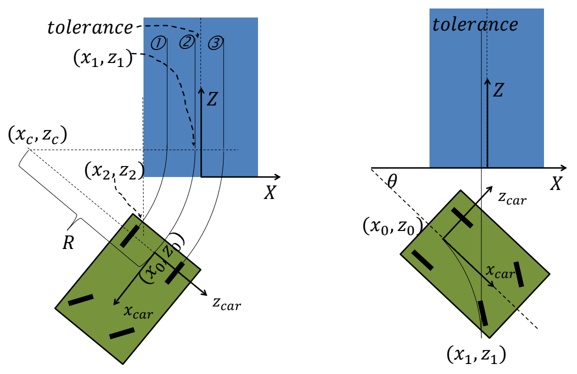
(ii) Arc① needs to intersect with left boundary or $x_1 - W_{car}/2 \geqslant -W_{slot}/2$

(iii) The lower intersection point $(x_2,\ z_2)$ needs to be below the bottom boundary or z_2 needs to be less than 0. Arc① is given in (5.3). By substituting $x = -W_{slot}/2,\ z = z_2$ and picking the smaller value, z_2 is solved as (5.4).

$$(x-x_c)^2 + (z-z_c)^2 = (R - W_{car}/2)^2 \tag{5.3}$$

$$z_2 = z_c - \sqrt{(R-W_{car}/2)^2 - (x_c + W_{slot}/2)^2} \tag{5.4}$$

Once these validations are passed, the path can be determined uniquely by point $(x_0,\ z_0)$ and $(x_1,\ z_1)$.



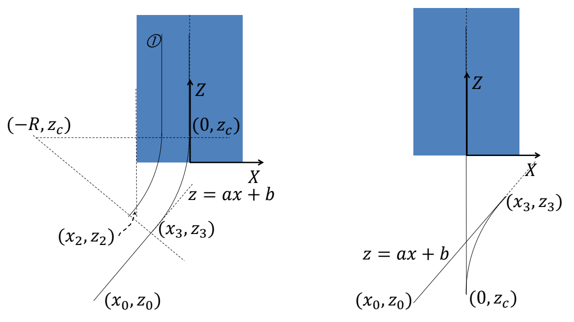
5.1.a: 5.1.b:

Figure 5.1: Paths with two segments

For the path in Fig. 5.1(b) $(RF-SB)$, the only thing to ensure is the first criteria as none of the vehicle body will enter the forbidden areas along the path. The remaining two paths for $n = 2$ $(LB - SB \text{ and } LF - SB)$ can be worked out similarly since they are symmetrical with the above two cases.

5.2.2 Paths with three segments

For paths with $n = 3$, they can be divided into two groups according to the first segment, $Line - Arc - Line$ (Fig. 5.2) and $Arc - Arc - Line$ (Fig. 5.3).



5.2.a: 5.2.b:

Figure 5.2: Paths with three segments – $Line - Arc - Line$

For the first group, the straight line $z = ax + b$ can be determined based on the vehicle initial pose with $a = \tan\theta$ and $b = z_0 - x_0 \tan\theta$. The first transition point $(x_3,\ z_3)$ satisfies

$$z_3 = ax_3 + b \tag{5.5}$$

$$(x_3 - (-R))^2 + (z_3 - z_c)^2 = R^2 \tag{5.6}$$

$$a(z_3 - z_c)/(x_3 - (-R)) = -1 \tag{5.7}$$

From the three equations above, all transition points $(x_3,\ z_3)$ and $(0,\ z_c)$ can be derived as (5.8)-(5.10). By comparing the relative position between $(x_0,\ z_0)$ and $(x_3,\ z_3)$, the moving direction for the first segment can be determined.

$$x_3 = -R + aR/\sqrt{1+a^2} \tag{5.8}$$

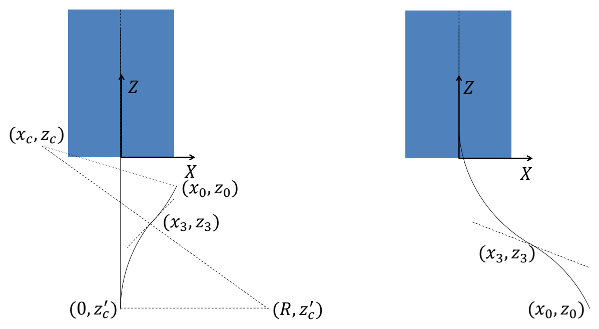
$$z_3 = -aR + b + a^2R/\sqrt{1+a^2} \tag{5.9}$$

$$z_c = aR + b + R/\sqrt{1+a^2} \tag{5.10}$$

Similarly, a collision check on the left boundary is necessary to ensure the vehicle does not enter forbidden areas. The condition is $z_2 < 0$, where z_2 is defined in (5.4) with $x_c = -R$.

For the path in Fig. 5.2(b), the transition points can be derived similarly with the arc center at $(R,\ z_c)$ instead of $(-R,\ z_c)$. For collision check, as long as the vehicle is outside the forbidden areas at $(x_3,\ z_3)$, it will be safe along the whole path. However, there is no short form solution for this case. Instead of using the analytical approach, we implement a more general collision checking method which is illustrated at the end of this section.

All the remaining paths in $Line-Arc-Line$ group can be worked out similarly based on the vehicle initial pose and the corresponding turning centers.



5.3.a: 5.3.b:

Figure 5.3: Paths with three segments – $Arc-Arc-Line$

If the path is $Arc-Arc-Line$ (Fig. 5.3), the transition points for Fig. 5.3(a) can be derived as following: The center for the first arc $(x_c,\ z_c)$ is given by (5.1) and (5.2). Since the two arcs are tangent, the distance between the two centers satisfies (5.11), from which $z_c{}'$ can be solved by selecting the smaller value. Then the transition point $(x_3,\ z_3)$ can be solved as (5.12) and (5.13).

$$(x_c - R)^2 + (z_c - z_c{}')^2 = (2R)^2 \tag{5.11}$$

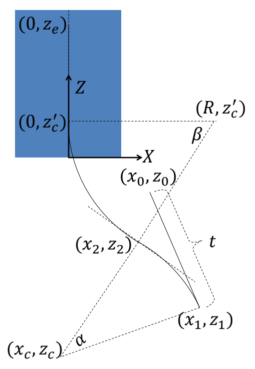
$$x_3 = 0.5(x_c + R) \tag{5.12}$$

$$z_3 = 0.5(z_c + z_c{}') \tag{5.13}$$

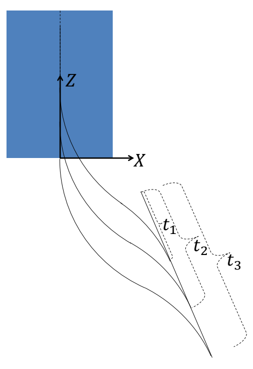
Fig. 5.3(b) path is not covered as it is redundant to the path with $n = 4$.

5.2.3 Paths with four segments

For the path with four segments, only the type *Line* − *Arc* − *Arc* − *Line* are checked. Even for this single type, the number of solutions given an initial vehicle pose is infinity. For example in Fig. 5.4(a), the path is related to the length of the first straight line t. As shown in Fig. 5.4(b), three different t result in three different paths.



5.4.a:



5.4.b:

Figure 5.4: Paths with four segments and its non-unique solutions with different t

From geometry, the transition points $(x_1,\ z_1)$ $(x_2,\ z_2)$ $(0,\ z_c{}')$ and corresponding turning centers $(x_c,\ z_c)$ $(R,\ z_c{}')$ in Fig. 5.4(a) can be specified as

$$x_1 = x_0 + t\cos\theta \tag{5.14}$$

$$z_1 = z_0 + t\sin\theta \tag{5.15}$$

$$x_c = x_1 + R\sin\theta \tag{5.16}$$

$$z_c = z_1 - R\cos\theta \tag{5.17}$$

$$z_c{}' = z_c + \sqrt{(2R)^2 - (x_c - R)^2} \tag{5.18}$$

$$x_2 = 0.5(x_c + R) \tag{5.19}$$

$$z_2 = 0.5(z_c + z_c{}') \tag{5.20}$$

Theoretically, t should be selected according to the shortest travelling distance d as shown in (5.21). However, the explicit form of d is too complicated to get its minimum analytically as it consists of terms with squares under square root, square root inside arcsine etc. It also depends on the vehicle initial pose $(x_0,\ z_0,\ \theta)$. A numerical method is preferred to solve this minimization problem since the range of t can be estimated roughly.

$$d = t + R\alpha + R\beta + (z_e - z_c{}') \tag{5.21}$$

$$\alpha = 2\arcsin\left(\sqrt{(x_1 - x_2)^2 + (z_1 - z_2)^2}/(2R)\right) \tag{5.22}$$

$$\beta = 2\arcsin\left(\sqrt{x_2^2 + (z_2 - z_c{}')^2}/(2R)\right) \tag{5.23}$$

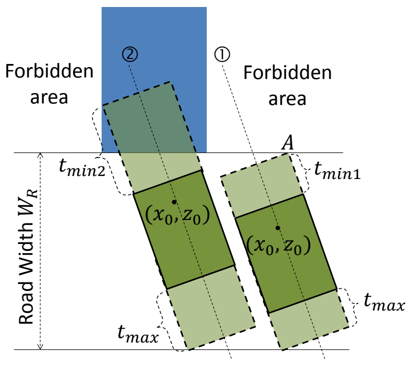


Figure 5.5: Range of t for paths with four segments

The range of t is defined in Fig. 5.5, where the solid dark areas mark the vehicle initial location and dotted light areas marks the extreme locations it might end when moving straight forward and backward. The upper limit of t, t_{max} is determined by the road boundary in the front of the parking slot as shown in Fig. 5.5. From geometry, t_{max} can be expressed in (5.24), where l and l_1 are defined in Fig. 3.8(a) and W_R is $\sim 8m$ for a typical two-lane road.

$$t_{max} = (z_0 + W_R)/(\cos\theta) - (l + l_1) + W_{car}/(2\tan\theta) \tag{5.24}$$

For its lower limit, two cases may occur as indicated by path ① and ② depending on the x location of point A or x_A. On path ①, the maximum distance the vehicle is able to move backward from location $(x_0,\ z_0)$ is t_{min1}, otherwise it will enter the forbidden area. While on path ②, it is able to move back further into the slot before it enters the forbidden area. From geometrical relationship, the lower limit t_{min} can be shown in (5.25) with $x_A = x_0 + \frac{W_{car}}{2\sin\theta} + \frac{z_0}{\tan\theta}$.

$$t_{min} = \begin{cases} \frac{z_0}{\sin\theta} + l_1 + \frac{W_{car}}{2\tan\theta} & |x_A| > \frac{W_{slot}}{2} \\ \\ -\frac{x_0+0.5W_{slot}}{\cos\theta} + l_1 + \frac{W_{car}\tan\theta}{2} & |x_A| \leqslant \frac{W_{slot}}{2} \end{cases}, \tag{5.25}$$

Since the scale of parking range is several meters, a discrete series of t with step size of $0.01m$ from t_{min} to t_{max} will provide an accurate enough estimation in locating the minimum value of d. However, each element in t series needs to meet the following criteria before being passed to d.

(i) Equation(5.18) must be real

(ii) Similar to (5.4), The intersection point between the vehicle inner wheel path and parking slot right boundary has to be below X axis.

The two criteria lead to the following inequalities respectively

$$(x_0 + R\sin\theta - R + \cos\theta t)^2 \leqslant 4R^2 \tag{5.26}$$

$$\begin{aligned} &\sin\theta t + \sqrt{4R^2 - (x_0 + R\sin\theta - R + \cos\theta t)^2} \\ &\leqslant R\cos\theta + \sqrt{(R - 0.5W_{car})^2 - (R - 0.5W_{slot})^2} - z_0 \end{aligned} \tag{5.27}$$

In summary, for a path with 4 segments, first, get the range of t based on (5.24) and (5.25). Discretize t with step size of $0.01m$ from t_{min} to t_{max}. Select the valid elements in t which satisfy (5.26) and (5.27). Pass all the valid elements to (5.14)-(5.23) to get the minimum of d and its corresponding transition points.

5.2.4 Refined paths

As mentioned before, the feasible path is not available at 9.5% of the poses just based on these 21 basic paths. To compensate this shortfall, we propose the following path generation algorithm incorporated with these basic paths.

(i) For a given initial vehicle pose $(x_0,\ z_0,\ \theta)$, get the shortest feasible path out of the basic ones if such a path exists.

(ii) Assume the vehicle moves LF (or LB, RF, RB) by a small angle $\delta\theta$.

(iii) Based on the new vehicle pose, get the shortest feasible path out of the basic ones if such a path exists.

(iv) Assume the vehicle further moves in the same direction and steering angle by another $\delta\theta$.

(v) Repeat iii)-iv) until a feasible path is found or the vehicle hits the forbidden area

(vi) Change to another combination of moving and steering directions (LB, RF or RB) and repeat ii)-v).

(vii) The optimal path is selected as the one with the shortest traveling distance.

If the path is still not available, move the vehicle forward or backward along a straight line by a small distance and repeat the steps above. The time complexity of the algorithm is approximately $O(n)$, linear to the number of basic paths.

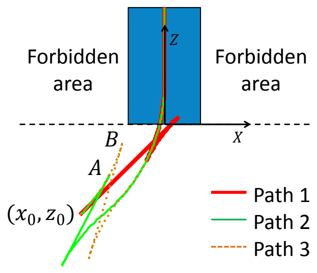


Figure 5.6: Illustration on path refining process

Fig. 5.6 illustrates this process briefly. Path1 is from the basic path while Path2 and Path3 are generated by assuming the vehicle steers to the right and moves backwards to Point A and Point B respectively, then follow the basic four-segment paths. As shown in the simulation in Section IV, the proposed path generation algorithm can provide a feasible path at each single pose, even when the vehicle is inside the parking slot.

5.2.5 Collision check

As aforementioned, due to the complexity in checking collision analytically under some cases, a general method is proposed, which can not only validate a generated path but also can check the vehicle collision status during the real run as long as the vehicle pose is known.

First, define a new coordinate system $x_{car} - z_{car}$ attached to the vehicle as shown in Fig. 5.1(a). Every point on the vehicle boundaries can be expressed easily in this coordinate system. By taking points on the boundaries with a step size of $0.01m$, we can get a set of points representing the vehicle boundaries.

$$C_{car} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \\ z_1 & z_2 & \dots & x_n \end{bmatrix} \tag{5.28}$$

The translation matrix T from $x_{car} - z_{car}$ coordinate to $X - Z$ coordinate is defined below, where $(x_0,\ z_0,\ \theta)$ is a given vehicle pose.

$$T = \begin{bmatrix} \cos\theta & -\sin\theta & x_0 \\ \sin\theta & \cos\theta & z_0 \end{bmatrix} \tag{5.29}$$

The vehicle boundary points expressed in $X - Z$ coordinate are

$$C = T \begin{bmatrix} C_{car} \\ 1_{1\times n} \end{bmatrix} \tag{5.30}$$

If any of the boundary points in C fall in the forbidden area, collision occurs.

5.3 Sliding mode controller design

To control the vehicle to follow the generated path, we apply SMC on the vehicle steering wheel. The controller calculates a target steering angle based on the vehicle position and orientation errors (d_e and θ_e) with respect to the target path. Vehicle velocity is not under the control of SMC. To implement SMC, the non-holonomic vehicle model defined previously is adopted (Fig. 3.8(a)).

d_e and θ_e are defined similarly as previous. However, since the target path is either an arc or a line instead of a parabola, the expressions are different from (4.5d). For the particular relative position shown in Fig. 3.8(b), if the path is an arc with the arc center at $(x_c,\ z_c)$, d_e and θ_e can be expressed as

$$d_e = -\sin\theta_d(x - x_d) + \cos\theta_d(z - z_d) \tag{5.31}$$

$$\theta_e = \theta - \theta_d \tag{5.32}$$

where $(x_d,\ z_d,\ \theta_d)$ defines the target position and orientation as shown below

$$\begin{cases} x_d = x_c + R|x - x_c|/\sqrt{(x - x_c)^2 + (z - z_c)^2} \\ z_d = z_c - R|z - z_c|/\sqrt{(x - x_c)^2 + (z - z_c)^2} \\ \theta_d = -\arctan\big((z - z_c)/(x - x_c)\big) \end{cases} \tag{5.33}$$

The position error dynamics can be calculated from (3.26) and (5.31).

$$\begin{aligned} \dot{d}_e &= v\sin\theta_e - \dot{\theta}_d[\cos\theta_d(x - x_d) + \sin\theta_d(z - z_d)] \\ &= v\sin\theta_e \qquad (from\ geometry) \end{aligned} \tag{5.34}$$

The following sliding surface is proposed so that both position and orientation errors converge to zero. k is non-negative for a stable sliding surface [123].

$$s = \dot{d}_e + kd_e = v\sin\theta_e + kd_e \tag{5.35}$$

The sliding surface convergence rate can be shown as

$$\dot{s} = v^2\cos\theta_e(\tan\phi - \tan\phi_d)/l + kv\sin\theta_e \tag{5.36}$$

Define $\dot{s} = -f(s)$ where $f(s)$ could be any non-decreasing odd function or combination of them, such as *sign*, *saturation* etc. To suppress the chattering effect and prevent a sudden change in control signal due to $f(s)$, we choose the *arctan* function which is non-decreasing odd and more importantly, continuous.

$$\dot{s} = -Q\arctan(s/P) \tag{5.37}$$

Equation(5.36) and (5.37) provide the control law on ϕ

$$\phi = \arctan\Big(-\frac{lQ\arctan(s/P)}{v^2\cos\theta_e} - \frac{lk\tan\theta_e}{v} + \tan\phi_d\Big) \tag{5.38}$$

A simple Lyapunov stability check using $V = s^2/2$ implies that P and Q must be positive for a stable system.

Conventionally, we should go on to calibrate k, P and Q based on their constraints and the corresponding control performance. However, for this particular application, a fixed value of k and Q is not sufficient to provide a smooth control signal even though some precautions have been taken in the choice of $f(s)$.

The main reason is due to the v terms in the denominator in (5.38). $|v|$ may drop close to zero occasionally, which magnifies the control signal in ϕ unnecessarily even when both s and θ_e are small. Such fast switching in steering directions induces strong jerking experience to the driver.

To compensate for this effect, we propose the following designs for Q and k, where Q' and k' are the new positive parameters to be tuned. Q is ensured to be non-negative as $-\pi/2 < \theta_e \leqslant \pi/2$. The resultant control law is shown in (5.40).

$$Q = (v^2\cos\theta_e/l)Q' \qquad\qquad k = |v/l|k' \tag{5.39}$$

$$\phi = \arctan\Big(-Q'\arctan(s/P) - sg(v)k'\tan\theta_e + \tan\phi_d\Big) \tag{5.40}$$

To summarize, (5.31)-(5.33), (5.35) and (5.40) form the full set of SMC design on the steering wheel, which can generate a smooth and fast response for path following.

5.4 Image processing

It has been shown in Chapter 2 that ridge detector is effective in extracting bright long structures in the image. Therefore, it is also implemented here to extract slot markings. The top row in Fig. 5.7 illustrates the results from ridge calculation. It directly extracts the medial axis of the slot boundary markings. Their ridge values are larger than the rest of the image.

A simple fixed value threshold is applied on the ridge image. Then, a noise filtering step, consisting of connected components labeling operation and removal of components with small number of pixels (< 25 *pixels*), is applied to remove the noise pixels. A sequential RANSAC line fitting algorithm is implemented to detect all the lines representing the visible slot boundaries, followed by a line assignment process to determine which boundary the lines belong to. Other robust line fitting algorithms are also applicable to replace RANSAC, e.g. Radon Transform [111]. Fig. 5.7 illustrates the whole process.

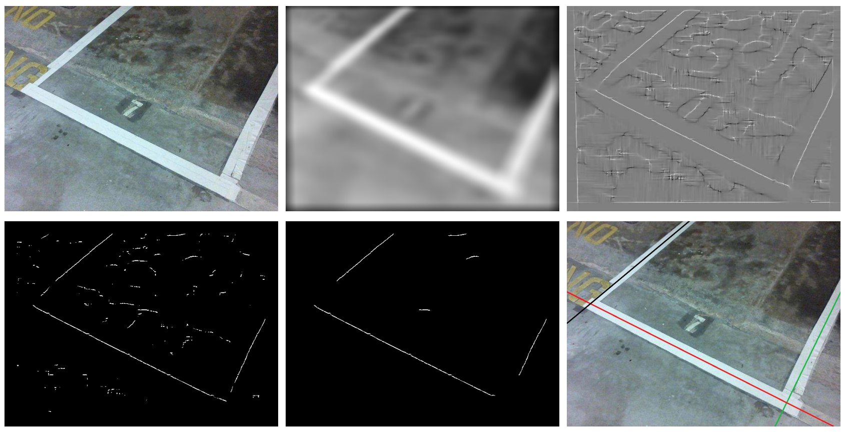


Figure 5.7: Illustration on image processing (row-wise=> original, smoothed, ridgeness, threshold, filtering, line assignment)

This line assignment process is based on Kalman filter prediction. From the predicted vehicle pose, a simulated parking slot image can be realized and thus, all its boundary medial axis can be expressed in the image coordinate system. The fitted lines in the real image will be compared with each of the simulated medial axis in terms of gradient and average pixel distance d_{ave} (5.41).

$$d_{ave} = (\sum_{i=1}^{N} |Cx_i + Ey_i + F/2|)/(N\sqrt{C^2 + E^2}) \tag{5.41}$$

where (x_i, y_i) represents one pixel on the fitted line, N is the total number of pixels on the fitted line and $Cx + Ey + F/2 = 0$ refers to the simulated slot boundary medial axis. In [124], the author also used the predicted image to fuse with actual images to improve the tracking accuracy. But the predicted image was estimated directly from vehicle odometry sensors without any filtering mechanisms, thus the results may be sensitive to noise.

The vehicle pose with respect to each parking slot boundary can be calculated by making use of the fitted lines and the camera intrinsic and extrinsic transformation matrix.

5.5 System integration

In this section, an overall picture of the algorithm is illustrated to demonstrate how each individual module interacts with each other.

When the driver stops the car at any pose near the intended car park slot, the fully automated reverse parking process can be initiated if part of the parking slot is within the camera view. The steps of the process are stated as follows:

(i) Take a picture of the parking slot and start the image processing to calculate vehicle pose.

(ii) Initialize the path generator with the above information and start to work out a feasible path. The associated moving velocity and steering angle at each point of the path are generated simultaneously.

(iii) Initialize Kalman filter and sliding mode controller.

(iv) Start the iteration by taking a new picture at the new vehicle pose.

(v) Derive the new vehicle pose based on image processing and Kalman filter state prediction.

(vi) Read in vehicle moving speed and steering angle.

(vii) Pass the new vehicle pose, speed and steering angle to Kalman filter for state estimation and prediction. A check on re-initialization of Kalman filter is carried out at the same time.

(viii) Check collision based on the estimated states. If collision occurs, stop the vehicle, re-generate a feasible path and resume from step iii) onwards.

If not, work out the target position $(x_d,\ z_d)$ and orientation (θ_d) based on the path and the estimated states.

(ix) Check whether the vehicle passes beyond the end point. If not, proceed to the next step. Otherwise, stop the vehicle and further check its alignment.

If its misalignment is small (within $\pm 2^\circ$ and $\pm 7cm$ from Z axis), the parking is completed. If not, re-generate a path and resume from step iii).

(x) Generate the error signals and pass them to SMC if no collision and not beyond the end point

(xi) Work out a steering angle command through SMC and send it to the vehicle steering motor.

(xii) Sent the target speed to an independent PID controller.

(xiii) Repeat step iv) to xii) until the parking is completed.

The flow chart on the whole algorithm is illustrated in Fig. 5.8. The Kalman filter re-initialization prevents the initial error, which is induced from the difference between the pre-assumed and actual parking slot size, from inflating as iteration continues. It also makes the system adaptive to different slot sizes.

The entire scheme guarantees a collision-free parking process since it checks collision at every iteration. The final parking accuracy is imposed as well by a misalignment check. The path planning step only carries out at the beginning and when collision or end-point misalignment occurs. The robust path planning frees human intervention from the whole process as there are no dead spots requiring the human driver to adjust the car pose to generate a feasible path.

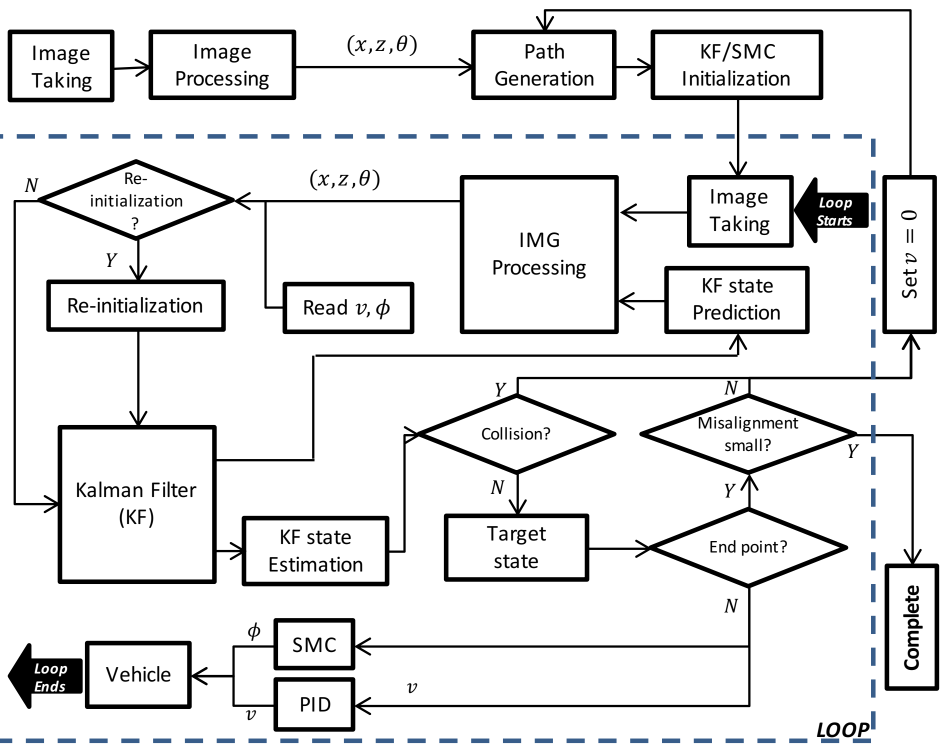


Figure 5.8: Self reverse parking algorithm flow chart

5.6 Experiment results

To validate the proposed algorithm, we implemented it on Toyota COMS EV as shown in Fig. 1.14. The camera is mounted at the center of the vehicle rear roof, pointing downwards at an angle of 48° from horizon and height of $1.55m$ from the ground. Other related dimensions are shown in the schematic drawing (Fig. 1.14 right). Note that the minimum turning radius measured from the control point is about $3.6m$.

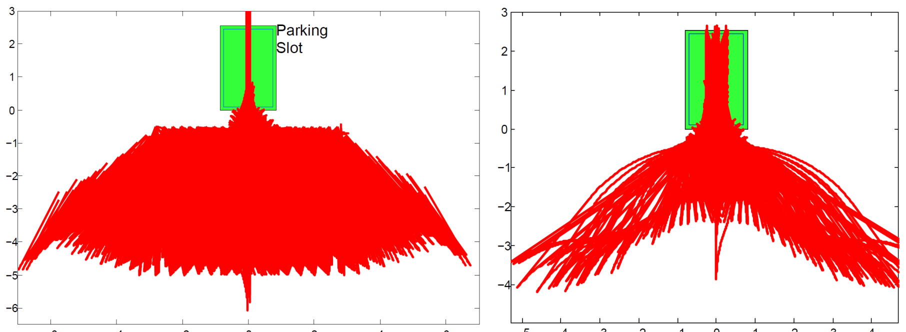
All the function modules mentioned above are realized by MATLAB running in the on-board NI PXI with Intel Core i5 CPU. The average time taken for one iteration is about $0.18s$ which is satisfactory for this low speed application. 85% of the time is spent on image processing.

5.6.1 Results on path generation

To validate the robustness of the path generation, the following simulation is designed: for the space in front of the parking slot, assume the vehicle position

X varies from $-2.8m$ to $2.8m$ with an interval of $0.2m$ and Z varies from $-0.5m$ to $-2.5m$ with an interval of $0.2m$ as well. For each position $(X,\ Z)$, assume the vehicle orientation varies from $0rad$ to $-\pi rad$ with an interval of $0.1rad$. Based on this mesh grid, we generate 10208 vehicle poses to test.

Fig. 5.9(a) shows the distribution of the path at each pose generated by the proposed algorithm with the turning radius $R = 3.6m$. The green box refers to the parking slot. At each of the 10208 poses, at least one feasible path can be generated. The free space required varies from $-7m$ to $7m$ for X and up to $-6m$ for Z. The widespread of the distribution is mainly caused by the poses which are close to horizontal direction and far away from parking center line. If the turning radius can be reduced, the free space required can be shrunk accordingly.

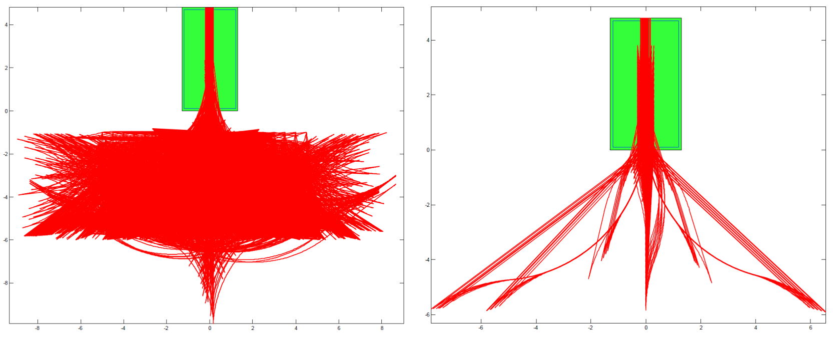


5.9.a: 5.9.b:

Figure 5.9: Path distribution for the testing vehicle with $R = 3.6m$

When the vehicle is inside the parking slot, similar grids are generated. The path distribution is shown in Fig. 5.9(b). The paths for the poses where the vehicle is at collision are not shown. An example is when the vehicle is near the left or right boundary; however, it seldom ends up at those poses as the collision check is carried out in every control cycle. The widespread wings are due to the poses near the slot bottom boundary with near horizontal orientations.

Since the test bed is smaller than normal sedans, we carry out further simulations to verify the path generation algorithms with normal sedan size and normal parking slot size. The sedan size is based on Toyota Corolla [125], 1.8m wide and 4.6m long with minimum turning radius of 5.4m. The size for a standard parking slot is $2.4m \times 4.8m$ [126]. The testing area is enlarged accordingly. The range in X direction is from $-5m$ to $5m$ and Z is from $-1m$ to $-5m$.



5.10.a: 5.10.b:

Figure 5.10: Path distribution for a normal sedan with $R = 5.4m$

For all the simulated poses, we obtained the path distribution as shown in Fig. 5.10. Again, we confirmed that for a normal sedan, at least one feasible path is available at any given vehicle pose, even when it is inside the parking slot (Fig. 5.10(b)). However, as compared to the test EV, the free space required in front of the parking slot is larger for a normal sedan since both its size and turning radius R are larger. For the same reason, under certain poses, a normal sedan requires paths with more segments.

To further evaluate the proposed algorithm, we also compared it with other two state-of-the-art algorithms proposed in [127] and [128] which also try to generate a feasible path regardless of vehicle initial poses.

In [127], the path consists of tangent arcs only except for the last segment which is a straight line (we refer this as Arc Algorithm). One of its disadvantages is that the final parking orientation is not unique. For example in Fig. 5.11, if the initial pose faces to the left (red vehicle), the final vehicle orientation faces outwards. If the initial pose faces to the right, it generates the same path (blue dot-dash line) and ends up with an inwards-faced orientation. If uniqueness is to be preserved, there may be no feasible path under some poses. But our algorithm guarantees an outwards-faced final orientation.

Moreover, no path is available when the vehicle initial pose is inside the slot.

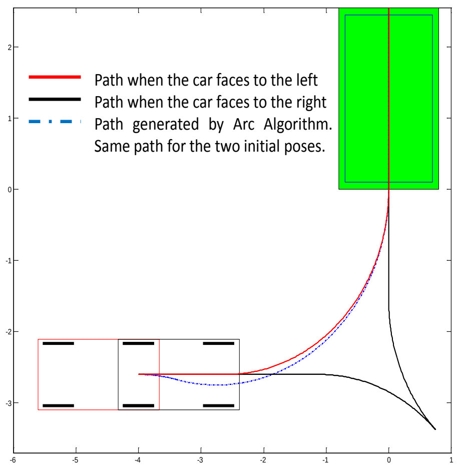


Figure 5.11: Non-uniqueness of final orientation by Arc Algorithm

This situation happens when the vehicle deviates from the planned path due to some disturbances and ends up in collision with adjacent slot. The path needs to be re-generated and the Arc Algorithm will fail.

For the algorithm proposed in [128] (KPP), it generates a multi-folded path, of which each section consists of translation, followed by rotation (minimum radius) and another translation. It is also able to generate a feasible path under any vehicle initial poses. Fig. 5.12 illustrates the comparison between the paths generated by KPP algorithm and our one for 3 initial poses. The 3 poses were used to compare KPP with another algorithm RRT in [129]. For a fair comparison, we adopted all the dimensions in that article.

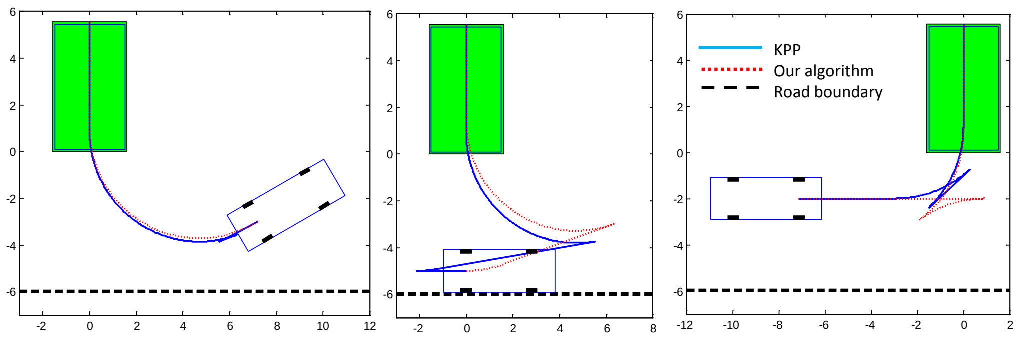


Figure 5.12: Path generation comparison between KPP algorithm and our proposed one

TABLE 5.2 shows the comparison of path length, number of maneuvers and computing time to generate the path. Both algorithms generate similar paths in terms of path length and no clear advantages of one algorithm over the other.

But due to the multi-folded property of KPP, it tends to generate paths with more maneuvers, which is more difficult to follow. And our algorithm is able to generate a feasible path much faster than KPP. The average time of generating one path shown in Fig. 5.9 is 0.098 seconds.

Table 5.2: Comparison between KPP and the proposed algorithm

Initial Pose	(1)		(2)		(3)	
Algorithm	Ours	KPP	Ours	KPP	Ours	KPP
Length(m)	14.6	15.5	19.8	22.64	20.11	18.56
Maneuvers	2	4	3	4	3	4
Com. time(s)	0.132	0.875	0.138	0.812	0.111	0.813

Therefore, the proposed algorithm outperforms KPP in the domain of reverse parking. But under a cluttered environment as shown in [128], a multi-folded approach is preferred at the expense of computing time. Note that our algorithm can be extended to a multi-folded version by adding straight lines and minimum turning arcs alternatively at the beginning of the proposed algorithm.

In summary, the robustness of the path generation algorithm has been validated and a feasible path is guaranteed under any initial poses.

5.6.2 Results on SMC

Fig. 5.13 illustrates how the vehicle position error d_e and orientation error θ_e vary from frame to frame during one typical parking trial. The errors are calculated based on (5.31) and (5.32). The left y axis and blue solid line are for θ_e while right y axis and red dotted line for d_e. Both values vary within a small range around 0. Frame 65 to 75 show the fast response from the controller to bring large deviations back to 0. The revised SMC is capable to control the vehicle to follow the planned path with very small position and orientation errors.

To validate the improvement in human driver's comfort level, the proposed path following controller was compared with the conventional SMC in [122] and fuzzy logic control. Three sets of on-field tests using these three controllers were carried out. Each set consist of 50 trials with random initial poses.

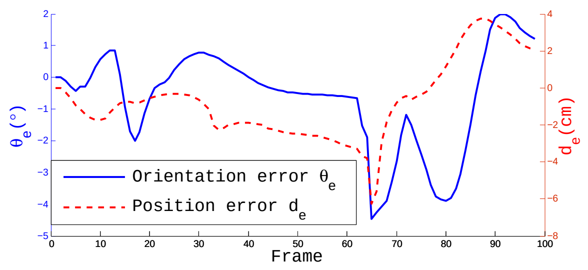


Figure 5.13: Position and orientation error trend during one trial

Similar to SMC, the fuzzy logic takes d_e and θ_e as the input and ϕ as the output. The membership functions are illustrated in Fig. 5.14. Based on the vehicle moving direction, 30 rules are designed in total. (*NB* stands for Negative Big, *N* for Negative, *Z* for Zero, *P* for Positive and *PB* for Positive Big.)

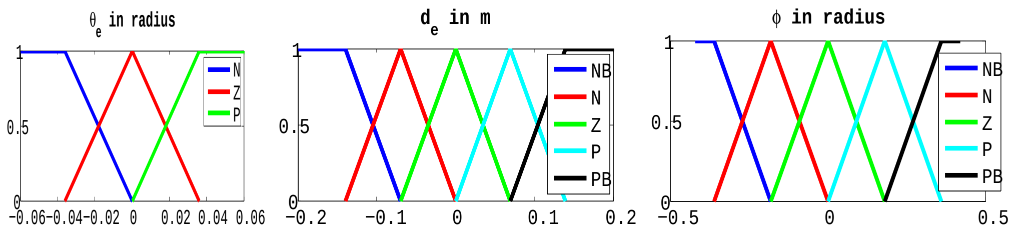


Figure 5.14: Fuzzy logic membership function design

To measure the comfort level, the following indices, unexpected steering swings (USS) and jerk, are proposed.

USS: one steering swing (SS) is defined as turning the steering wheel from center or zero position to its left or right limit. The number of expected SS is counted from the generated path. The difference between the expected SS and SS counted from the actual run is USS. Ideally USS equals to 0. The smaller the USS is, the smoother the controller is.

$$J_L = \dot{a_L} = d(\dot{\theta}v)/dt = d(tan(\phi)v^2/l)/dt$$
$$= (v^2 \sec^2(\phi)\dot{\phi} + 2\tan(\phi)v\dot{v})/l \tag{5.42}$$

$$J_T = \dot{a_T} = d(\dot{v})/dt = \ddot{v} \tag{5.43}$$

$$J_v = \sqrt{J_L^2 + J_T^2} \tag{5.44}$$

$$J_\phi = d^3(\phi)/dt^3 \tag{5.45}$$

Jerk [130]: It is the time derivative of the acceleration. Both vehicle motion (J_v) and steering (J_ϕ) induce jerk to the system. J_v can be decomposed into two components, longitudinal J_L and lateral J_T as shown in (5.42) and (5.43). Both J_v and J_ϕ average absolute values based on all the measurements in one trial are used as indices to indicate the jerking effect of that trial.

TABLE 5.3 shows the comparison between the proposed, conventional SM-C and fuzzy logic performance. Data presented are mean values over the 50 trials for each controller. As compared to the conventional SMC, since there are less unexpected steering swings, the proposed controller output is more capable of following the required steering profile while making necessary minor adjustments. The induced jerking effects can be suppressed significantly, especially the ones from steering wheel.

Table 5.3: Comparison between the proposed, conventional SMC and fuzzy logic

Controller	USS	$J_v(ms^{-3})$	$J_\phi(s^{-3})$
Proposed SMC	1.02	0.985	8.218
Conventional SMC	3.06	1.669	20.812
Fuzzy Logic	0.03	1.212	10.850

As compared to fuzzy logic control, although the proposed SMC yields higher USS due to the fast transient response, it is still able to reduce jerking effects in both J_v and J_ϕ and provide smoother control actions. Moreover, the tuning of the proposed SMC is much simpler than the fuzzy logic control since SMC involves two parameters only while fuzzy logic control involves more than 10.

From all the experiments above, we can conclude that the revised SMC is able to generate smooth control output and eliminate jerking to a large extent. It results in a higher level of human driver's comfort.

5.6.3 Results on final parking accuracy

Extensive experiments were conducted under different illumination situations to evaluate the robustness of the proposed algorithm. The situations include an indoor car park and an open air car park during daytime, sunset and night time. Sample images are shown in Fig. 5.15 to demonstrate illumination intensities. During daytime, the sunshine is strongest and visibility is the best. But some pixels may be saturated. During sunset, the sunshine and visibility are moderate. During night, the visibility is very poor and the light source is from the vehicle tail lights. In indoor car park, visibility is also moderate and fluorescent lamp is the main light source. Two illustration videos (link v and vi) are provided in Appendix B to show the parking process.

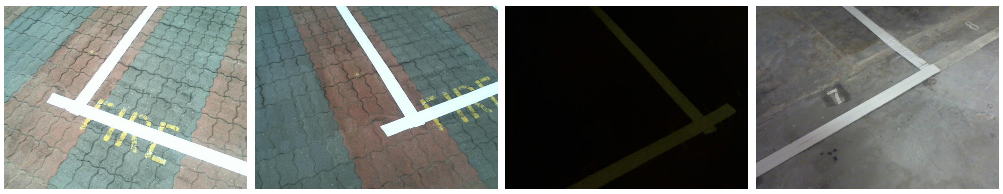


Figure 5.15: Illumination during daytime, sunset, night and indoor

54 trials were carried out under each situation with a variety of initial poses as shown in Fig. 5.16 to better reflect real parking exercises. Parking accuracy is evaluated based on two indices, namely the vehicle offset distance from the slot center line and its orientation deviation. These two values were measured manually after finishing each trial.

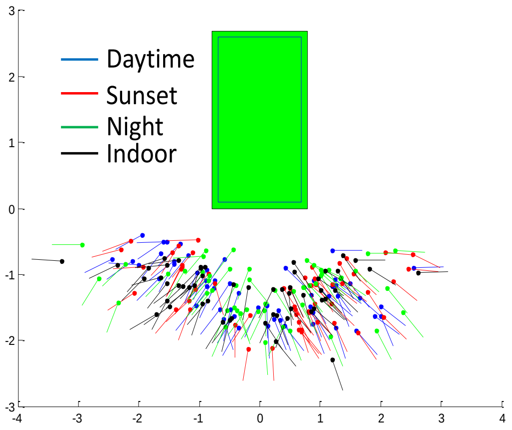


Figure 5.16: Initial pose distribution for all the trials

Fig. 5.17 plots the absolute offset and orientation error for each trial under different illumination conditions. From both offset and orientation error plots, the accuracy is similar under different conditions. There is no clear advantage of one condition over the others. The same observation is made on RMS as shown in TABLE 5.4. This infers that the proposed vision approach is robust to illumination variation, even the image quality taken during the night is poor.

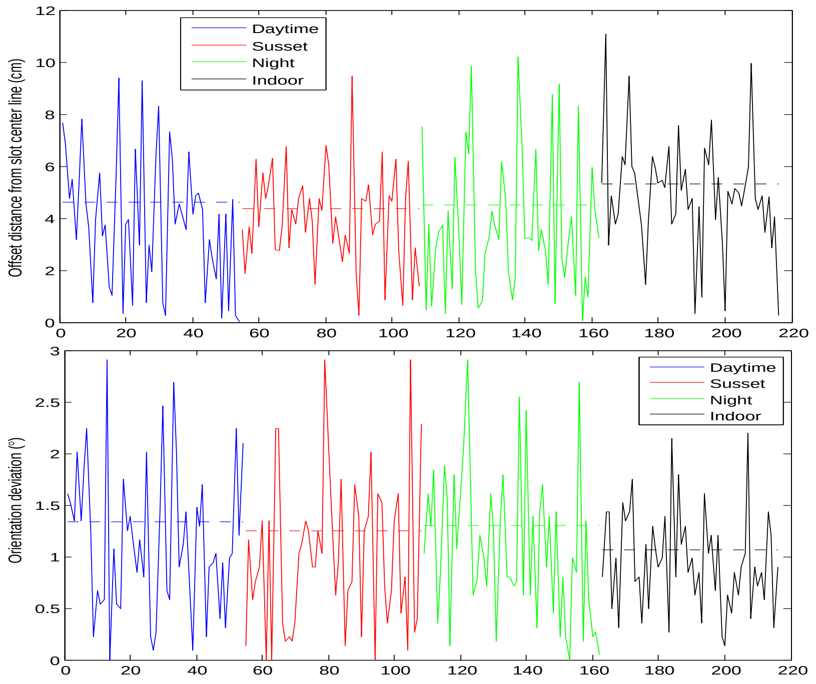


Figure 5.17: Absolute offset distance and orientation deviation for each trial (dashed line represents RMS)

The overall RMS distance offset is $4.71cm$ and orientation deviation is $1.24°$. The overall standard deviation (STD) for offset is $2.31cm$ and $0.66°$ for orientation deviation. The overall 95% confidence interval for offset is $[-0.46cm,\ 8.7cm]$ and $[-0.25°,\ 2.4°]$ for orientation deviation. This shows that the proposed approach is capable of providing consistently satisfactory parking accuracy under environment changes. The approach is therefore reliable.

Table 5.4: Offset and orientation RMS and STD

		Daytime	Sunset	Night	Indoor	Overall
Offset(cm)	RMS	4.60	4.36	4.50	5.31	4.71
	STD	2.50	1.81	2.64	2.09	2.31
Orientation Deviation(°)	RMS	1.34	1.25	1.30	1.07	1.24
	STD	0.70	0.73	0.71	0.47	0.66

It is important to note that the results obtained are subject to other disturbances, including 1) inaccurate conversion from steering motor turning angle to vehicle turning radius; 2) un-modeled slippage between ground and wheels, between wheels and driving motors; 3) deficits in camera calibration. All these factors can lead to inaccuracy in Kalman filter and image processing. But the result shows that the algorithm handles these disturbances well.

The parking accuracy was also compared with manual parking results. Except for the autonomous COMS mentioned above, there were 8 more manual COMS shared among NUS staff. The sharing scheme has been carried out for 1.5 years and the COMS users are considered experienced. The parking accuracy was measured at the end of the day after the users returned all the COMS to the dedicated indoor car park. 50 sets of data were collected. The RMS distance offset is $8.12cm$ and orientation deviation is $1.73°$. This shows that the proposed autonomous parking approach can achieve a slightly better accuracy than experienced human drivers.

5.7 Chapter summary

In this chapter, a vision-based autonomous reverse parking system is proposed, which is a supplementary but essential system to the AV platform. The low cost system is proven to be practical and reliable through extensive on-field experiments under different illumination conditions.

The implementation of a ridge detector and a Kalman filter ensures the accuracy and consistency of vehicle pose estimation while the revised SMC reduces the jerking effects to a large extent. The robust overall control scheme makes sure that there is no collision with adjacent parking slots and the vehicle parks accurately along the slot center line.

The novel path planning algorithm guarantees a feasible path at any given pose. This is the key function to enable the fully autonomous feature because it doesn't require human driver's intervention to re-adjust the vehicle to a certain pose in order to get a feasible path.

Chapter 6

Conclusions

6.1 Main contributions

In this thesis, a vision-based minimum viable solution towards sustainable autonomous vehicles is proposed, which substantially reduces both the cost and power consumption as compared to typical AV setups. It utilizes vision to perceive environment instead of Lidar or Radar. Special controllers are designed to control the vehicle based on the vision feedback. The solution consists of a vision-based vehicle lane-level localization system, an NMPC-based path following control system and a fully vision-based reverse parking system.

First of all, a vehicle lane-level localization system based on mono camera is proposed. It identifies lane line pixels in the images through the customized ridge detector and then removes noise pixels to a large extent based on an adaptive filtering mechanism. The road model, from which the vehicle pose respected to the road can be calculated, is fitted through a modified RANSAC algorithm and then followed by a parallelism reinforcement technique. It is able to detect lane lines in the images at a high success rate and estimate the vehicle localization information with small errors.

To further improve the vehicle pose estimation accuracy and consistency, at the same time, to obtain the depth information, a more advanced stereovision system is constructed based on the lessons learnt from the mono camera system. It is more robust to noise as it incorporates special detectors to distinguish lane line markings from warning letters, directional arrows, humps and zebra lines. It fits multiple road models to cater for different road conditions, even including zigzag lines. The depth information is obtained from an effective stereo 3D reconstruction algorithm without carrying out the conventional time-consuming correspondence mapping algorithms. A new particle filter framework, which takes vehicle dynamics into account, is embedded into the solution to further improve the estimation consistency. More importantly the system works in real time and has very few limitations in practice, even working in dark nights.

By taking the output from the stereovision system as feedback, an NMPC scheme is designed to control the vehicle to follow the detected lane. It controls the velocity and steering direction of the vehicle simultaneously. The GA is customized to optimize the NMPC in real time. As compared to other existing MPC optimizers, using GA enables a more flexible structure for MPC formulation. The cost function and constraints can be designed in a more accurate, meaningful and direct way, which would otherwise have to be approximated or compromised in order to fit into other optimizers. Benefitting from this flexible structure, passengers' safety and comfort are taken into account explicitly.

Both simulations and on-field tests show that the NMPC system based on the stereovision feedback is able to control the vehicle to travel along the center line of the detected road accurately and consistently. The accelerations induced by the steering and vehicle movements are well confined within a certain range, thus allowing a pleasant and safe journey.

Furthermore, a vision-based reverse parking system is proposed as well to park the car into the parking slot autonomously. It uses a single camera, pointing backwards, to identify and track the parking slot markings on the floor. A novel path planning module ensures that a feasible path is always available under any initial poses, which frees human intervention completely. A modified sliding mode controller controls the vehicle to follow the prescribed path to park the car into the slot. Intensive on-field tests under different illumination conditions demonstrate its robustness, consistency and accuracy.

In conclusion, all these systems mentioned above, working together, formulate the minimum viable vision-based autonomous vehicle solution. Although the mono-vision system is not applied on the AV platform eventually, it proves us the feasibility of the frame work for lane line detection and vehicle pose estimation and motivates us to come out with the more advanced stereovision system and the self parking vision system as well. The stereovision system serves as the feedback to the NMPC control system. Benefitting from its high detection accuracy and consistency, the NMPC is able to achieve high performance in controlling the vehicle to follow the detected lane line. Otherwise, the NMPC will suffer from the detection error. And regardless of how advanced or adaptive the controller is, it will not achieve good performance if the feedback is wrong. The vision parking system is a supplementary but essential component to the AV platform. Without this system, the AV will not park properly no matter how accurate the stereovision system and NMPC system can perform.

Hybridizing the minimum viable vision-based platform with remote human intervention leads to a more sustainable autonomous vehicle solution with a fine balance in cost, safety and efficiency. The hybrid system has been tested and demonstrated for a number of times over the years to illustrate its viability. The five awards, won both regionally and internationally as listed in Appendix A, are self-evident testimonies of the excellence and potential of the proposed system.

6.2 Future works

The next milestone to be achieved in this solution is to minimize the level of remote human intervention and dedicate it purely to emergencies and contingencies so that one remote driver can oversee more vehicles at one time, and thus further reducing the total cost. In the current hybrid system, the responsibilities under the remote human drivers are still considered as heavy to reach at a lower ratio between the number of remote drivers and autonomous vehicles. To achieve this milestone, the following future works are proposed.

First of all, vision-based obstacle detection is the most essential key to reduce the remote human intervention substantially. The anticipated results from the obstacle detection include the obstacle category (car, cyclist, pedestrian), it-s moving speed and moving direction. The main challenges in achieving this lie in the requirements on real time operation and high detection rate, which is expected to be more than 99% for a reliable and safe driving. But the current benchmark for vision-based car and pedestrian detection in complex environment is only 83.8% and 64.8% respectively as listed in the KITTI database [16]. Therefore, there exist big gaps for improvement and huge potential for further and deeper research in this area. With the development of machine learning technologies and the increment in computation power, it is anticipated that the vision-based obstacle detection is highly achievable.

With this information from the obstacle detection, the autonomous vehicle will be able to identify dynamic models for each obstacle and predict their future dynamic states to understand their intentions. An intelligent decision-making s-trategy is to be developed to determine whether the autonomous vehicle should give way to, overtake or follow the obstacles. Machine learning will again play an important role in achieving this as we can try to teach the autonomous vehicles to learn from a human driver.

Global path planning is also required to guide the autonomous vehicle to move from one point to another. The technology is quite mature and has been applied widely in the current commercial GPS navigators and smart phones. But the technology was developed based on how it can be understood and used easily by human. Therefore how to integrate and tailor it to fulfill the need of autonomous vehicles is another challenge, especially for vehicle navigation at junctions with multiple branches.

With all these future works in completion, the vehicle will be able to automatically drive the passengers around under normal conditions. The remote human driver will only be activated when emergencies or unforeseen scenarios occur, such as temporary road closure, car sensor failure and etc.. The commuters, after setting their destinations, can enjoy their journey to the fullest without any concerns.

References

[1] Gary Silberg, Richard Wallace, G Matuszak, J Plessers, C Brower, and D Subramanian. Self-driving cars: The next revolution. *KPMG LLP & Center of Automotive Research*, 2012.

[2] Carole Jacques. Self-driving cars an $87 billion opportunity in 2030, though none reach full autonomy. `https://portal.luxresearchinc.com/research/report_excerpt/16874`, 2014-05-20.

[3] Mobility Transformation Center University of Michigan. Mcity test facility. `http://www.mtc.umich.edu/test-facility`, 2014-05-20.

[4] Myra Blanco, Jon Atwood, Sheldon Russell, Tammy Trimble, Julie McClafferty, and Miguel Perez. Automated vehicle crash rate comparison using naturalistic data. Technical report, Virginia Tech Transportation Institute, 2016.

[5] Kristen Parrish. Astonishing advances that may eliminate accidents and put valets out of business. `http://sharpheels.com/2015/04/self-driving-cars-of-tomorrow/`, 2015-04. [Online; accessed 12-April-2016].

[6] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, et al. Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597, 2008.

[7] Land Transport Authority. Standard details of road elements. `https://www.lta.gov.sg/content/dam/ltaweb/corp/Industry/files/SDRE14-0content1-7.pdf`, 2014-04. [Online; accessed 23-June-2015].

[8] JFrancine Tardo and Monika Stickel. Look ma, no hands! `https://www.ieee.org/about/news/2012/5september_2_2012.html`, 2012-09-05.

[9] Tao Jiang, Srdjan Petrovic, Uma Ayyer, Anand Tolani, and Sajid Husain. Self-driving cars: Disruptive or incremental? *Applied Innovation Review*, 1:3–22, 2015.

[10] The Ford Motor Company. Ford at ces announces s-mart mobility plan. `https://media.ford.com/content/fordmedia/fna/us/en/news/2015/01/06/ford-at-ces-announces-smart-mobility-plan.html`, 2015-01-06.

[11] Rhiannon Williams. Bmw and baidu partner to build driverless cars in china. `http://www.telegraph.co.uk/technology/news/12044117/BMW-and-Baidu-partner-to-build-driverless-cars-/in-China.html`, 2015-10-10.

[12] The Tesla Motors Team. Your autopilot has arrived. `https://www.teslamotors.com/blog/your-autopilot-has-arrived`, 2015-10-14.

[13] Kpg81. Tesla's musk sees fully autonomous car ready in 5 years. `https://my.teslamotors.com/fr_CH/forum/forums/tesla's-musk-sees-fully-autonomous-car-ready-5-years`, 2014-09-17.

[14] Mohr Davidow Ventures. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006.

[15] World Health Organization. Global status report on road safety 2015: summary. page 4, 2015.

[16] Jannik Fritsch, Tobias Kuehnl, and Andreas Geiger. A new performance measure and evaluation benchmark for road detection algorithms. In *International Conference on Intelligent Transportation Systems (ITSC)*, pages 1693–1700. IEEE, 2013.

[17] Kyaw Ko Ko Htet. *Development and Control of Personal Mobility Platform for Multi-Hierarchy Last Mile Transportation*. PhD thesis, National University of Singapore, Singapore, 2016.

[18] Charles Thorpe, Martial H Hebert, Takeo Kanade, and Steven A Shafer. Vision and navigation for the carnegie-mellon navlab. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 10(3):362–373, 1988.

[19] Aharon Bar Hillel, Ronen Lerner, Dan Levi, and Guy Raz. Recent progress in road and lane detection: a survey. *Machine Vision and Applications*, 25(3):727–745, 2014.

[20] Sergiu Nedevschi, Rolf Schmidt, Thorsten Graf, Radu Danescu, Dan Frentiu, Tiberiu Marita, Florin Oniga, and Ciprian Pocol. 3d lane detection system based on stereovision. In *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, pages 161–166. IEEE, 2004.

[21] Qing Li, Nanning Zheng, and Hong Cheng. An adaptive approach to lane markings detection. In *Intelligent Transportation Systems, 2003. Proceedings.*, volume 1, pages 510–514. IEEE, 2003.

[22] Rodolfo Tapia-Espinoza and Miguel Torres-Torriti. A comparison of gradient versus color and texture analysis for lane detection and tracking. In *Robotics Symposium (LARS), 2009 6th Latin American*, pages 1–6. IEEE, 2009.

[23] Tsung-Ying Sun, Shang-Jeng Tsai, and Vincent Chan. Hsi color model based lane-marking detection. In *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*, pages 1168–1172. IEEE, 2006.

[24] ZuWhan Kim. Robust lane detection and tracking in challenging scenarios. *Intelligent Transportation Systems, IEEE Transactions on*, 9(1):16–26, 2008.

[25] Raghuraman Gopalan, Tsai Hong, Michael Shneier, and Ramalingam Chellappa. A learning approach towards detection and tracking of lane markings. *Intelligent Transportation Systems, IEEE Transactions on*, 13(3):1088–1098, 2012.

[26] Joerg Fritsch, Tobias Kuhnl, and Franz Kummert. Monocular road terrain detection by combining visual and spatial information. *Intelligent Transportation Systems, IEEE Transactions on*, 15(4):1586–1596, 2014.

[27] Dong-Joong Kang and Mun-Ho Jung. Road lane segmentation using dynamic programming for active safety vehicles. *Pattern Recognition Letters*, 24(16):3177–3185, 2003.

[28] Chang-Hoon Kum, Dong-Chan Cho, Moon-Soo Ra, and Whoi-Yul Kim. Lane detection system with around view monitoring for intelligent vehicle. In *SoC Design Conference (ISOCC), 2013 International*, pages 215–218. IEEE, 2013.

[29] Guangtao Cui, Junzheng Wang, and Jing Li. Robust multilane detection and tracking in urban scenarios based on lidar and mono-vision. *Image Processing, IET*, 8(5):269–279, 2014.

[30] Wenjie Lu, Emmanuel Seignez, F Sergio A Rodriguez, and Roger Reynaud. Lane marking based vehicle localization using particle filter and multi-kernel estimation. In *Control Automation Robotics & Vision (ICARCV), 2014 13th International Conference on*, pages 601–606. IEEE, 2014.

[31] Sayanan Sivaraman and Mohan Manubhai Trivedi. Integrated lane and vehicle detection, localization, and tracking: A synergistic approach. *Intelligent Transportation Systems, IEEE Transactions on*, 14(2):906–917, 2013.

[32] Xinxin Du and Kok Kiong Tan. Vision-based approach towards lane line detection and vehicle localization. *Machine Vision and Applications*, pages 1–17, 2015.

[33] Xinxin Du, Kok Kiong Tan, and Kyaw Ko Ko Htet. Vision-based lane line detection for autonomous vehicle navigation and guidance. In *Control Conference (ASCC), 2015 10th Asian*, pages 3139–3143. IEEE, 2015.

[34] Amol Borkar, Monson Hayes, and Mark T Smith. A novel lane detection system with efficient ground truth generation. *Intelligent Transportation Systems, IEEE Transactions on*, 13(1):365–374, 2012.

[35] Radu Danescu and Sergiu Nedevschi. Probabilistic lane tracking in difficult road scenarios using stereovision. *Intelligent Transportation Systems, IEEE Transactions on*, 10(2):272–282, 2009.

[36] Hao Li and Fawzi Nashashibi. Robust real-time lane detection based on lane mark segment features and general a priori knowledge. In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pages 812–817. IEEE, 2011.

[37] A López, J Serrat, C Canero, F Lumbreras, and T Graf. Robust lane markings detection and road geometry computation. *International Journal of Automotive Technology*, 11(3):395–407, 2010.

[38] Antonio Guiducci. Parametric model of the perspective projection of a road with applications to lane keeping and 3d road reconstruction. *Computer Vision and Image Understanding*, 73(3):414–427, 1999.

[39] Romuald Aufrere, Roland Chapuis, and Frederic Chausse. A model-driven approach for real-time road recognition. *Machine Vision and Applications*, 13(2):95–107, 2001.

[40] Jean-Yves Bouguet. Camera calibration toolbox for matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example5.html`, 2013-12-02. [Online; accessed 23-June-2015].

[41] F David and Daniel Scharstein. Multi-model estimation in the presence of outliers. Master's thesis, Middlebury College, may 2011.

[42] Paul D Sampson. Fitting conic sections to very scattered data: An iterative refinement of the bookstein algorithm. *Computer graphics and image processing*, 18(1):97–108, 1982.

[43] William H Press. *Numerical recipes in Fortran 77: the art of scientific computing*, volume 1. Cambridge university press, 1992.

[44] Jin Wang, Stefan Schroedl, Klaus Mezger, Roland Ortloff, Armin Joos, and Thomas Passegger. Lane keeping based on location technology. *Intelligent Transportation Systems, IEEE Transactions on*, 6(3):351–356, 2005.

[45] Joel C McCall and Mohan M Trivedi. Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation. *Intelligent Transportation Systems, IEEE Transactions on*, 7(1):20–37, 2006.

[46] Raphael Labayrade, Jerome Douret, and Didier Aubert. A multi-model lane detector that handles road singularities. In *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*, pages 1143–1148. IEEE, 2006.

[47] Daniel Topfer, Jens Spehr, Jan Effertz, and Christoph Stiller. Efficient road scene understanding for intelligent vehicles using compositional hierarchical models. *Intelligent Transportation Systems, IEEE Transactions on*, 16(1):441–451, 2015.

[48] Hunjae Yoo, Ukil Yang, and Kwanghoon Sohn. Gradient-enhancing conversion for illumination-robust lane detection. *Intelligent Transportation Systems, IEEE Transactions on*, 14(3):1083–1094, 2013.

[49] Xin Du and Kok Kiong Tan. Autonomous reverse parking system based on robust path generation and improved sliding mode control. *Intelligent Transportation Systems, IEEE Transactions on*, 16(3):1225–1237, 2015.

[50] Raphaël Labayrade, Jerome Douret, Jean Laneurit, and Roland Chapuis. A reliable and robust lane detection system based on the parallel use of three algorithms for driving safety assistance. *IEICE transactions on information and systems*, 89(7):2092–2100, 2006.

[51] Shinq-Jen Wu, Hsin-Han Chiang, Jau-Woei Perng, Chao-Jung Chen, Bing-Fei Wu, and Tsu-Tian Lee. The heterogeneous systems integration design and implementation for lane keeping on a vehicle. *Intelligent Transportation Systems, IEEE Transactions on*, 9(2):246–263, 2008.

[52] 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.

[53] Dajun Ding, Jongsu Yoo, Jekyo Jung, Sungho Jin, and Soon Kwon. Various lane marking detection and classification for vision-based navigation system? In *Consumer Electronics (ICCE), 2015 IEEE International Conference on*, pages 491–492. IEEE, 2015.

[54] Yan Jiang, Feng Gao, and Guoyan Xu. Computer vision-based multiple-lane detection on straight road and in a curve. In *Image Analysis and Signal Processing (IASP), 2010 International Conference on*, pages 114–117. IEEE, 2010.

[55] Ruyi Jiang, Reinhard Klette, Tobi Vaudrey, and Shigang Wang. New lane model and distance transform for lane detection and tracking. In *Computer Analysis of Images and Patterns*, pages 1044–1052. Springer, 2009.

[56] Jiang Ruyi, Klette Reinhard, Vaudrey Tobi, and Wang Shigang. Lane detection and tracking using a new lane model and distance transform. *Machine vision and applications*, 22(4):721–737, 2011.

[57] Guoliang Liu, Florentin Worgotter, and Irene Markelic. Stochastic lane shape estimation using local image descriptors. *Intelligent Transportation Systems, IEEE Transactions on*, 14(1):13–21, 2013.

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

[59] Hiroaki Sawano and Minoru Okada. A road extraction method by an active contour model with inertia and differential features. *IEICE transactions on information and systems*, 89(7):2257–2267, 2006.

[60] Amol Borkar, Monson Hayes, and Mark T Smith. Robust lane detection and tracking with ransac and kalman filter. In *ICIP*, pages 3261–3264, 2009.

[61] Le Thanh Sach, Kiyoaki Atsuta, Kazuhiko Hamamoto, and Shozo Kondo. A robust road profile estimation method for low texture stereo images. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 4273–4276. IEEE, 2009.

[62] Chunzhao Guo, Seiichi Mita, and David McAllester. Stereovision-based road boundary detection for intelligent vehicles in challenging scenarios. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1723–1728. IEEE, 2009.

[63] Sebastiano Battiato, Giuseppe Messina, and Alfio Castorina. Exposure correction for imaging devices: An overview. *Single-Sensor Imaging: Methods and Applications for Digital Cameras*, pages 323–349, 2008.

[64] Yun-Suk Kang and Yo-Sung Ho. An efficient image rectification method for parallel multi-camera arrangement. *Consumer Electronics, IEEE Transactions on*, 57(3):1041–1048, 2011.

[65] Ministry of Transportation and Highways. Manual of standard traffic signs & pavement markings. `http://www.th.gov.bc.ca/publications/eng_publications/electrical/most_pm.pdf`, 2000-09. [Online; accessed 23-June-2015].

[66] Ronald N Bracewell. *The Fourier transform and its applications*. McGraw-Hill, 1978.

[67] Luo Juan and Oubong Gwun. A comparison of sift, pca-sift and surf. *International Journal of Image Processing (IJIP)*, 3(4):143–152, 2009.

[68] Yan Ke and Rahul Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–506. IEEE, 2004.

[69] Tao Wu and Ananth Ranganathan. A practical system for road marking detection and recognition. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 25–30. IEEE, 2012.

[70] Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(1):105–119, 2010.

[71] A Ardeshir Goshtasby. *Image registration: Principles, tools and methods*, chapter Chapter 2: Similarity and Dissimilarity Measures, pages 57–58. Springer Science & Business Media, 2012.

[72] M Ester, HP Kriegel, J Sander, and X Xu. Density-based spatial clustering of applications with noise. In *Int. Conf. Knowledge Discovery and Data Mining*, volume 240, 1996.

[73] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012.

[74] Riccardo Marino, Stefano Scalzi, Giuseppe Orlando, and Mariana Netto. A nested pid steering control for lane keeping in vision based autonomous vehicles. In *American Control Conference*, pages 2885–2890. IEEE, 2009.

[75] Xinxin Du, Kok Kiong Tan, and Kyaw Ko Ko Htet. Vision approach towards fully self-reverse parking system. In *Mechatronics and Automation (ICMA), 2014 IEEE International Conference on*, pages 186–191. IEEE, 2014.

[76] Ying-Chieh Yeh, Tzuu-Hseng S Li, and Chih-Yang Chen. Adaptive fuzzy sliding-mode control of dynamic model based car-like mobile robot. *International Journal of Fuzzy Systems*, 11(4):272–286, 2009.

[77] Davor Hrovat, Stefano Di Cairano, H Eric Tseng, and Ilya V Kolmanovsky. The development of model predictive control in automotive industry: A survey. In *Control Applications (CCA), 2012 IEEE International Conference on*, pages 295–302. IEEE, 2012.

[78] Yiqi Gao, Andrew Gray, H Eric Tseng, and Francesco Borrelli. A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles. *Vehicle System Dynamics*, 52(6):802–823, 2014.

[79] Yiqi Gao, Theresa Lin, Francesco Borrelli, Eric Tseng, and Davor Hrovat. Predictive control of an autonomous ground vehicle using an iterative linearization approach. In *ASME 2010 Dynamic Systems and Control Conference*, volume 1, pages 265–272. ASME, 2010.

[80] Kentaro Oyama and Kenichiro Nonaka. Model predictive parking control for nonholonomic vehicles using time-state control form. In *Control Conference (ECC), 2013 European*, pages 458–465. IEEE, 2013.

[81] Multi-parametric toolbox 3. `http://control.ee.ethz.ch/~mpt/3/Main/HomePage`, 2014-07. [Online; accessed 29-July-2015].

[82] Muhammad Aizzat Zakaria, Hairi Zamzuri, Rosbi Mamat, Saiful Amri Mazlan, et al. A path tracking algorithm using future prediction control with spike detection for an autonomous vehicle robot. *International Journal of Advanced Robotic Systems*, 10:309–317, 2013.

[83] Daniele Corona and Bart De Schutter. Adaptive cruise control for a smart car: A comparison benchmark for mpc-pwa control methods. *Control Systems Technology, IEEE Trans on*, 16(2):365–372, 2008.

[84] S Di Cairano and H Eric Tseng. Driver-assist steering by active front steering and differential braking: design, implementation and experimental evaluation of a switched model predictive control approach. In *Decision and Control (CDC), 49th IEEE Conference on*, pages 2886–2891. IEEE, 2010.

[85] Ting Qu, Hong Chen, Dongpu Cao, Hongyan Guo, and Bingzhao Gao. Switching-based stochastic model predictive control approach for modeling driver steering skill. *Intelligent Transportation Systems, IEEE Transactions on*, 16(1):365–375, 2015.

[86] Valerio Turri, Ashwin Carvalho, Hongtei Tseng, Karl Henrik Johansson, and Francesco Borrelli. Linear model predictive control for lane keeping and obstacle avoidance on low curvature roads. In *Intelligent Transportation Systems (ITSC), 2013 IEEE 16th International Conference on*, volume 1, pages 378–383. IEEE, 2013.

[87] Eelco Scholte and Mark E Campbell. Robust nonlinear model predictive control with partial state information. *Control Systems Technology, IEEE Transactions on*, 16(4):636–651, 2008.

[88] Flavio Manenti. Considerations on nonlinear model predictive control techniques. *Computers & Chemical Engineering*, 35(11):2491–2509, 2011.

[89] Zhaoyang Wan and Mayuresh V Kothare. Efficient scheduled stabilizing model predictive control for constrained nonlinear systems. *International Journal of Robust and Nonlinear Control*, 13(3-4):331–346, 2003.

[90] Janick V Frasch, Alison Gray, Mario Zanon, Hans Joachim Ferreau, Sebastian Sager, Francesco Borrelli, and Moritz Diehl. An auto-generated nonlinear mpc algorithm for real-time obstacle avoidance of ground vehicles. In *Control Conference (ECC), 2013 European*, pages 4136–4141. IEEE, 2013.

[91] Mario Zanon, Janick V Frasch, Milan Vukov, Sebastian Sager, and Moritz Diehl. Model predictive control of autonomous vehicles. In *Optimization and Optimal Control in Automotive Systems*, pages 41–57. Springer, 2014.

[92] D. Ariens, B. Houska, H. Ferreau, and F. Logist. *ACADO for Matlab User's Manual*. Optimization in Engineering Center (OPTEC), 1.0beta edition, May 2010. `http://www.acadotoolkit.org/`.

[93] Ayman A Aly et al. Pid parameters optimization using genetic algorithm technique for electrohydraulic servo control system. *Intelligent Control and Automation*, 2(02):69, 2011.

[94] M Nasir Uddin, MA Abido, and MA Rahman. Real-time performance evaluation of a genetic-algorithm-based fuzzy logic controller for ipm motor drives. *Industry Applications, IEEE Transactions on*, 41(1):246–252, 2005.

[95] Peter J Fleming and RC Purshouse. Genetic algorithms in control systems engineering. Technical report, University of Sheffield, 2001.

[96] W Naeem, R Sutton, J Chudley, FR Dalgleish, and S Tetlow. A genetic algorithm-based model predictive control autopilot design and its implementation in an autonomous underwater vehicle. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, 218(3):175–188, 2004.

[97] SK Sharma and R Sutton. A genetic algorithm based nonlinear guidance and control system for an uninhabited surface vehicle. *Journal of Marine Engineering & Technology*, 12(2):29–40, 2013.

[98] Ching-Fu Lin, Jyh-Ching Juang, and Kun-Rui Li. Active collision avoidance system for steering control of autonomous vehicles. *Intelligent Transport Systems, IET*, 8(6):550–557, 2014.

[99] Lars Svensson and Jenny Eriksson. Tuning for ride quality in autonomous vehicle: Application to linear quadratic path planning algorithm. 2015.

[100] Paolo Falcone, Francesco Borrelli, H Eric Tseng, Jahan Asgari, and Davor Hrovat. Linear time-varying model predictive control and its application to active steering systems: Stability analysis and experimental validation. *International journal of robust and nonlinear control*, 18(8):862–875, 2008.

[101] Taehyun Shim, Ganesh Adireddy, and Hongliang Yuan. Autonomous vehicle collision avoidance system using path planning and model-predictive-control-based active front steering and wheel torque control. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of automobile engineering*, page 0954407011430275, 2012.

[102] The hybrid that started it all. `http://www.toyota.com/prius/#!/Welcome`, 2014-03.

[103] Bwm 7 series – park assist. `http://www.bmw.com/com/en/newvehicles/7series/sedan/2012/showroom/driver_assistance/park-assistant.html#t=l`, 2013-03.

[104] Carsten Heilenkötter, Norbert Höver, Peter Magyar, Thomas Ottenhues, Tilmann Seubert, and Joachim Wassmuth. The consistent use of engineering methods and tools. *AutoTechnology*, 6(6):52–55, 2006.

[105] Jochen Pohl, M Sethsson, Pär Degerman, and Jonas Larsson. A semi-automated parallel parking system for passenger cars. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 220(1):53–65, 2006.

[106] Ho Gi Jung, Young Ha Cho, Pal Joo Yoon, and Jaihie Kim. Scanning laser radar-based target position designation for parking aid system. *Intelligent Transportation Systems, IEEE Transactions on*, 9(3):406–424, 2008.

[107] Paír Degerman, Jochen Pohl, and Magnus Sethson. Ultrasonic sensor modeling for automatic parallel parking systems in passenger cars. In *SAE World Congress (2007: Detroit, Mich.)*, 2007.

[108] Massaki Wada, Kang Sup Yoon, and Hideki Hashimoto. Developmen-t of advanced parking assistance system. *Industrial Electronics, IEEE Transactions on*, 50(1):4–17, 2003.

[109] Katia Fintzel, Reny Bendahan, C Vestri, Sylvian Bougnoux, and T Kak-inami. 3d parking assistant system. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 881–886. IEEE, 2004.

[110] Nico Kaempchen, Uwe Franke, and Rainer Ott. Stereo vision based pose estimation of parking lots using 3d vehicle models. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 2, pages 459–464. IEEE, 2002.

[111] Chunxiang Wang, Hengrun Zhang, Ming Yang, Xudong Wang, Lei Ye, and Chunzhao Guo. Automatic parking based on a bird's eye view vision system. *Advances in Mechanical Engineering*, 6:847406, 2014.

[112] Ho Gi Jung, Dong Seok Kim, and Jaihie Kim. Light-stripe-projection-based target position designation for intelligent parking-assist system. *Intelligent Transportation Systems, IEEE Transactions on*, 11(4):942–953, 2010.

[113] Lester E Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, 79(3):497–516, 1957.

[114] JA Reeds and LA Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.

[115] Daobin Wang, Huawei Liang, Tao Mei, and Hui Zhu. Research on self-parking path planning algorithms. In *Vehicular Electronics and Safety (ICVES), 2011 IEEE International Conference on*, pages 258–262. IEEE, 2011.

[116] T-HS Li and Shih-Jie Chang. Autonomous fuzzy parking control of a car-like mobile robot. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 33(4):451–465, 2003.

[117] Moritz B Oetiker, Gion P Baker, and Lino Guzzella. A navigation-field-based semi-autonomous nonholonomic vehicle-parking assistant. *Vehicular Technology, IEEE Transactions on*, 58(3):1106–1118, 2009.

[118] Kristijan Macek, Roland Philippsen, and Roland Siegwart. Path following for autonomous vehicle navigation with inherent safety and dynamics margin. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 108–113. IEEE, 2008.

[119] Brigitte d'Andréa Novel, Gianni Campion, and Georges Bastin. Control of nonholonomic wheeled mobile robots by state feedback linearization. *The International journal of robotics research*, 14(6):543–559, 1995.

[120] M Khoshnejad and K Demirli. Autonomous parallel parking of a car-like mobile robot by a neuro-fuzzy behavior-based controller. In *Fuzzy Information Processing Society, 2005. NAFIPS 2005. Annual Meeting of the North American*, pages 814–819. IEEE, 2005.

[121] Wenjie Dong and K-D Kuhnert. Robust adaptive control of nonholonomic mobile robot with parameter and nonparameter uncertainties. *Robotics, IEEE Transactions on*, 21(2):261–266, 2005.

[122] Razvan Solea and Urbano Nunes. Trajectory planning with velocity planner for fully-automated passenger vehicles. In *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*, pages 474–480. IEEE, 2006.

[123] Hassan K Khalil. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, 2002.

[124] Jae Kyu Suhr and Ho Gi Jung. Sensor fusion-based vacant parking slot detection and tracking. *Intelligent Transportation Systems, IEEE Transactions on*, 15(1):21–36, Feb 2014.

[125] Elevate your drive. `http://www.toyota.com/corolla/#!/Welcome`, 2014-03.

[126] Land Transport Authority Singapore. Code of practice on vehicle parking provision in development proposals. `http://www.lta.gov.sg/content/dam/ltaweb/corp/Industry/files/VPCOP2011.pdf`, 2011-03.

[127] Ming Feng Hsieh and Umit Ozguner. A parking algorithm for an autonomous vehicle. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 1155–1160, 2008.

[128] Dalhyung Kim, Woojin Chung, and Shinsuk Park. Practical motion planning for car-parking control in narrow environment. *IET control theory & applications*, 4(1):129–139, 2010.

[129] Hyunki Kwon and Woojin Chung. Comparative analysis of path planners for a car-like mobile robot in a cluttered environment. In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pages 602–607. IEEE, 2011.

[130] Julien Clinton Sprott and Julien C Sprott. *Chaos and time-series analysis*, volume 69. Oxford University Press Oxford, 2003.

Appendix A: Publication and Award List

Journal Papers

[1] Xinxin Du, and Kok Kiong Tan, Development of Genetic-Algorithm-based Nonlinear Model Predictive Control Scheme on Velocity and Steering of Autonomous Driving Vehicles, *Industrial Electronics, IEEE Transactions on*, accepted, 2016.

[2] Kyaw Ko Ko Htet, Xinxin Du, and Kok Kiong Tan, Comprehensive Lane Keeping System with Mono Wide-Angle Camera, *Intelligent Transportation Systems, IEEE Transactions on*, submitted under review, 2016.

[3] Xinxin Du, and Kok Kiong Tan, Comprehensive and Practical Vision System for Self-Driving Vehicle Lane-Level Localization, *Image Processing, IEEE Transactions on*, 25(5):2075-2088, 2016.

[4] Xinxin Du, and Kok Kiong Tan, Vision-Based Approach Towards Lane Line Detection and Vehicle Localization, *Machine Vision and Applications*, 27(2):175-191, 2015.

[5] Xinxin Du, and Kok Kiong Tan, Autonomous Reverse Parking System Based on Robust Path Generation and Improved Sliding Mode Control, *Intelligent Transportation Systems, IEEE Transactions on*, 16(3):1225-1237, 2015.

[6] Xinxin Du, Kok Kiong Tan and Kyaw Ko Ko Htet, Autonomous reverse parking system-vision approach through ridge detector and Kalman filter, *International Journal of Mechatronics and Automation*, 5(1):22-33, 2015.

Conference Papers

[1] Xinxin Du, and Kok Kiong Tan. Autonomous vehicle velocity and steering control through nonlinear model predictive control scheme. In *Transportation Electrification Conference and Expo Asia-Pacific, 2016 IEEE*, Accepted. IEEE, 2016.

[2] Xinxin Du, Kok Kiong Tan, and Kyaw Ko Ko Htet. Vision-based lane line detection for autonomous vehicle navigation and guidance. In *Control Conference (ASCC), 2015 10th Asian*, pages 3139-3143. IEEE, 2015.

[3] Xinxin Du, Kok Kiong Tan, and Kyaw Ko Ko Htet. Development of a genetic algorithm based nonlinear model predictive scheme. In *Control Conference (ASCC), 2015 10th Asian*, pages 1102-1107. IEEE, 2015.

[4] Kyaw Ko Ko Htet, Kok Kiong Tan, and Xinxin Du. Comprehensive Lane Keeping System with Mono Camera. In *Control Conference (ASCC), 2015 10th Asian*, pages 808-813. IEEE, 2015.

[5] Xinxin Du, Kok Kiong Tan, and Kyaw Ko Ko Htet. Vision approach towards fully self-reverse parking system, *Mechatronics and Automation (ICMA), 2014 IEEE International Conference on*, pages 186-191. IEEE, 2014.

Awards

[1] Won Academic Honorary Mention at 2015 National Instruments (NI) Engineering Impact Awards ASEAN Regional Contest

[2] Won Advanced Research Award at 2015 National Instruments (NI) Engineering Impact Awards ASEAN Regional Contest

[3] Won Editor's Choice at 2015 National Instruments (NI) Engineering Impact Awards ASEAN Regional Contest

[4] Won Best Presentation Award at 2015 NUS ECE 5th Graduate Student Symposium

[5] Won Best Paper Award in Automation at The 2014 IEEE International Conference on Mechatronics and Automation

Appendix B: Illustration Video Links

(i) Lane line detection through stereovision during daytime

`https://www.dropbox.com/s/4edblzmixprfemn/LLD1.mp4?dl=0`

(ii) Lane line detection through stereovision at dusk, at night and after rain

`https://www.dropbox.com/s/kkcvb9rgqs9m7j5/LLD2.mp4?dl=0`

(iii) Lane line following under nonlinear MPC 1

`https://www.dropbox.com/s/kcdplw29pge0zqn/MPC1.mp4?dl=0`

(iv) Lane line following under nonlinear MPC 2

`https://www.dropbox.com/s/a8ah16ouq4lc7nw/MPC2.mp4?dl=0`

(v) Autonomous reverse parking 1

`https://www.dropbox.com/s/j07si2cuyc2f8xc/Parking1.mp4?dl=0`

(vi) Autonomous reverse parking 2

`https://www.dropbox.com/s/l34afznpw1w3jqo/Parking2.mp4?dl=0`