Technische Universiteit
Eindhoven
University of Technology

/ Department of Mathematics and Computer Science
/ PDEng Automotive Systems Design

Next-Generation Lane Centering Assist System

Design and Implementation of a Lane Centering Assist System, using NXP-BlueBox

October 2017

Rameez Ismail

Where innovation starts

Next-Generation Lane Centering Assist System

Design and Implementation of a Lane Centering Assist System, using NXP BlueBox

Eindhoven University of Technology
Stan Ackermans Instiiute - Automotive Systems Design

PDEng Report: 2017/092

The design that is described in this report has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct.

Partners

NXP Semiconductors	Eindhoven University of Technology

Submitted By	Rameez Ismail
Steering Group	Dr. P.S.C. (Peter) Heuberger Dr. Gerardo Daalderop Dr. Gijs Dubbelman Ing. Han Raaijmakers
Date	October 2017

Contact Address	Eindhoven University of Technology Department of Mathematics and Computer Science MF 5.072, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands +31 402743908
Partnership	This project was supported by Eindhoven University of Technology and NXP.
Published by	Eindhoven University of Technology Stan Ackermans Institiute
Printed by	Eindhoven University of Technology *UniversiteitsDrukkerij*
PDEng-report	2017/092
Preferred reference	Next Generation Lane Centering Assist System. Eindhoven University of Technology, PDEng Technical Report, October, 2017.092
Keywords	BlueBox, Lane Tracking, Probabilistic Classification, System Design, ISO26262, LCAS
Disclaimer Endorsement	Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the Eindhoven University of Technology or NXP. The views and opinions of authors expressed herein do not necessarily state or reflect those of the Eindhoven University of Technology or NXP, and shall not be used for advertising or product endorsement purposes.
Disclaimer Liability	While every effort will be made to ensure that the information contained in this report is accurate and up to date, Eindhoven University of Technology makes no warranty, representation or undertaking whether expressed or implied, nor does it assume any legal liability, whether direct or indirect, or responsibility for the accuracy, completeness, or usefulness of any information.
Trademarks	Product and company names mentioned herein may be trademarks and/or service marks of their respective owners. We use these names without any particular endorsement or with the intent to infringe the copyright of the respective owners.
Copyright	Copyright © 2016. The Eindhoven University of Technology. All rights reserved. No part of the material protected by this copyright notice may be reproduced, modified, or redistributed in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the Eindhoven University of Technology and NXP Semiconductors.

Foreword

Currently, a tremendous growth in embedded automated systems and robotics is foreseen for the next decades. The many opportunities in diverse application areas will drive solutions at affordable-cost to every household and industry.

The growth is spurred by accelerating innovation in sensor technologies, in reliable high-performance networking and in low power general and application-specific processing with ever-increasing performance. Taken together, this leads to a symbiosis between big data, cloud computing and embedded computing, allowing, for example, the efficient inference of neural networks on embedded systems and novel algorithms for object detection and classification, world modeling and path-finding. Applied to automotive, this enables a big trend towards safe driving ("zero accidents") and highly automated driving. Business-wise, this leads to the expectation that the value of electronics in automobiles will increase roughly threefold over the next decade.

In this project, Rameez has developed a vision algorithm on NXP vision processor and optimized the implementation to enable a very fast and low power performance. This application he then used to automate the lateral driving function of a car. For this, Rameez had to understand the system architecture of the automated driving system of the car ('the driver-replacement domain'), perform the safety case analysis, perform safe system integration and guide a team of 10 students during 3 months. Rameez did this in a very nice, independent, well-structured and high-quality way, for which I want to thank him warmly.

Dr. G.H.O Daalderop MBA
September 2017

Preface

This report describes my final assignment for the Professional Doctorate in Engineering (PDEng) program at the Eindhoven University of Technology (TU/e). The degree program is provided by the TU/e automotive groups and offers specialization in Automotive Systems Design (ASD). The focus of the ASD program is on providing training on the systems approach to solve automotive design problems. The trainees work on several multidisciplinary projects from automotive companies, where state-of-the-art systems engineering approach is followed. The program is divided in to two phases, each with a length of one year. During the first phase, the focus is on the professional and personal development of the trainees through extensive industrial workshops. Also, short design projects are carried out in this phase for the industrial partners of the TU/e. In these projects, ASD trainees work in teams to learn leadership skills and practice the art of team work. The second phase consists of a final design assignment of twelve month period, carried out at a company. The final assignment, however, is an individual assignment which provides every trainee an opportunity to prove and establish himself as a systems designer.

In accordance with this, I carried out my final design assignment at NXP with the goal to design and develop an advanced Lane Centering Assist System (LCAS) using the NXP BlueBox. The main task of the system is to prevent a car from inadvertently straying from the lane it is driving in. This report describes the design and realization of the LCAS and is aimed for an audience that has basic knowledge of the systems engineering, physics and signal processing. Additionally, in the first part of the report, a socioeconomic need for the LCAS and self-driving cars is explained along with their relationship in context of this assignment.

Rameez Ismail
September, 2017

Acknowledgements

Firstly, I would like to express my sincere gratitude to my TU/e supervisor Dr. Dubbelman for the knowledge and support he offered during my PDEng assignment and for his motivation and dedication for mobile perception systems. His research and guidance helped me to achieve the results of my PDEng assignment, I could not have imagined having a better advisor for my PDEng assignment.

Besides my TU/e advisor, I would like to express my deepest appreciation for my supervisor from NXP, Dr. Gerardo Daalderop, who provided me the possibility to work on this exciting assignment. I am immensely thankful for his contribution in stimulating suggestions and encouragement. I have learned a lot from him and have always praised his ability to multitask and manage different projects at the same time.

My special thanks also go to my second supervisor and mentor from NXP, Ing. Han Raaijmakers, who has very patiently taught me various tools and technologies associated with the NXP BlueBox. He's the liveliest person in the laboratory and one of the smartest. I am highly indebted to him for his guidance and constant supervision as well as for providing necessary technical assistance regarding the project.

Above all, my sincere gratitude for the ASD program manager Dr. P.S.C. (Peter) Heuberger, I am grateful to him for giving me this opportunity to be a part of this prestigious program and guiding me from time to time with his precious advice. He has always reminded me to take breaks from work.

A very special gratitude goes out to Dr. Andrei Terechko, Andrei has always been very resourceful in providing advice and help on technical issues and has always inspired me from his enthusiasm for work. I also thanks, Dr. Terechko, for the stimulating discussions on the topics of technology.

I thank my fellow PDEng colleagues, for their moral support and for all the fun we have had in the last two years. Also, I thank all my friends and colleagues at the Technical University of Eindhoven as well as at NXP Semiconductors. In particular, I would like to thank Sapfo Tsoutsou for her efforts in reviewing this report.

Last but not the least, I would like to thank my family: my parents, Farzana Ismail and Chaudhary Ismail, as well as to my brothers and sisters for supporting me throughout my life.

Rameez Ismail
September, 2017

Executive Summary

The technology for self-driving cars is on the verge of emergence. It is driven by the ever-increasing demand for convenient transportation and growing awareness on the safety and suitability issues of the current transport and mobility infrastructure. However, there are huge technical challenges that need attention to fulfill the envisioned future of sustainable mobility. The biggest challenge is to robustly sense the immediate surrounding while driving. Self-driving cars need to reliably perceive the environment, in real-time. The underlying algorithms are rapidly evolving but one practical limiting factor is the amount of processing power available in the vehicles. To bridge this technological gap, NXP has introduced a high-end embedded platform, which they named as NXP BlueBox.

The goal of this design assignment is to make a step forward in vehicle automation while showcasing the real-time perception capabilities of the BlueBox. To this purpose, a vehicle automation concept is formulated for the test vehicle, Toyota-Prius, The concept is derived from a bottom-up approach which advocates piecing together of various subsystems to give rise to a more complex system. In this project design and realization of one such subsystem, namely 'Lane Centering Assist System (LCAS)' was carried out using the BlueBox. The main function of the system is to perceive its immediate environment through a forward-facing camera and then steer the vehicle to keep it centered in the driving lane. The core enabler of the system is a state-of-the-art lane detection and tracking algorithm, which is under research at the Technical University of Eindhoven (TU/e). In this assignment, various functional and non-functional improvements were introduced to the algorithm to enhance its reliability and robustness while lowering the computational cost. A real-time implementation of the algorithm for the BlueBox is realized along with an open-source implementation, which can be deployed on an x86-64 or ARM-based processor.

The BlueBox implementation makes use of the heterogeneous computing units, for example, Image Signal Processor (ISP) and APEX cores, of the platform to accelerate the processing. Besides providing a real-time lane detection and tracking, the system is also capable of providing an active steering to keep the vehicle, automatically, centered in the current driving lane. The functional performance of the lane tracker is evaluated by visual means, according to which the algorithm performs far superior compared to the conventional approaches. This is more evident when the vehicle is taking sharp turns, changing lanes or when the lane markings are not clearly visible. The results from the algorithm can be viewed by visiting the open source repository for the project[1].

Furthermore, a thorough system design for the LCAS is also proposed in this report which follows an advanced system design process. The systems engineering approach employed, to carry out the design task, is a combination of the CAFCR architectural reasoning and the design guidelines from the recent automotive safety standard, ISO26262. As the LCAS is a safety critical system, various safety risks and hazards were analyzed before proposing the system architecture. The motive behind this is to ensure that not only the product but also the process followed in the assignment is the technical state of the art.

[1] https://github.com/RameezI/TUeLaneTracker

Table of Contents

List of Figures

List of Tables

1.Introduction

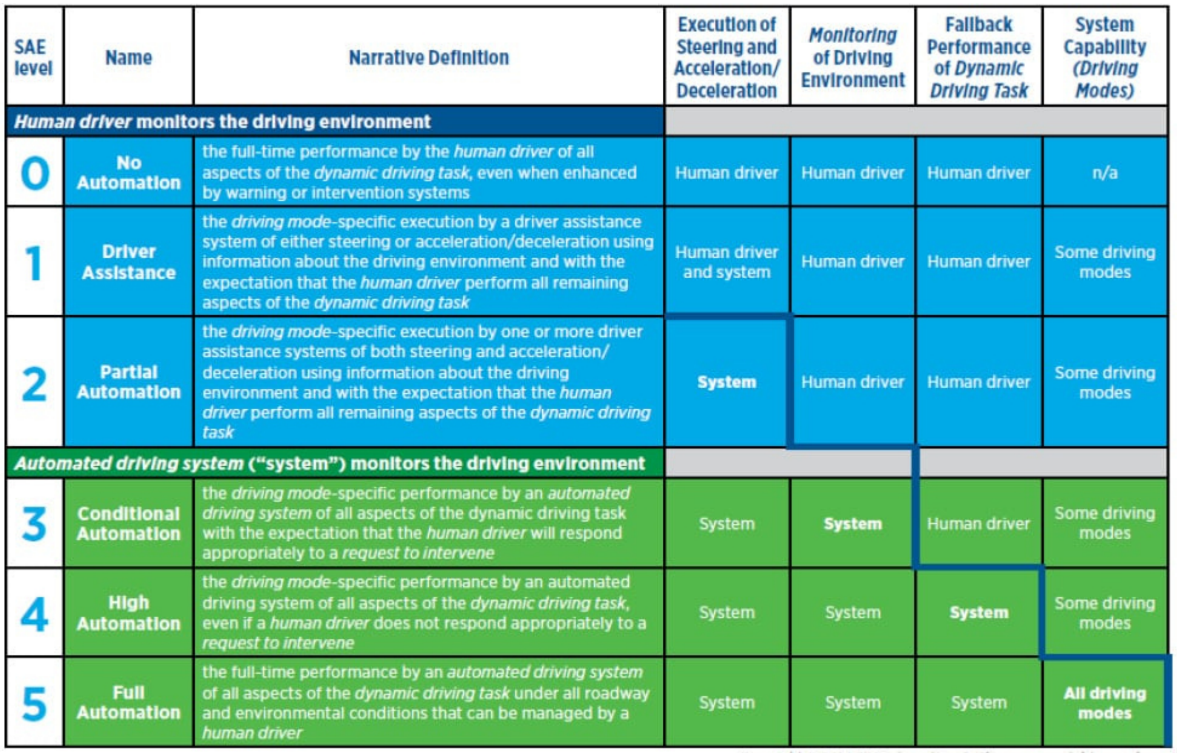
Every systems designer, I suppose, has a motivational context in mind within which his or her design can be fully understood and explained. This chapter is mainly dedicated to establishing this context for the reader. A self-driving car is no longer confined to science fiction movies but is the fact of today and will become a consumer reality in coming decades. It is now time that we get past our initial impulsive reactions of excitement as well as fear and critically analyze various benefits, challenges, risks and opportunities that self-driving cars could present. The benefits are manifold from safety to convenience; self-driving cars have the potential to completely transform the transport infrastructure, which could lead to mobility as well as economic and sustainability gains. However, we need to overcome huge challenges to fulfill this envisioned future of smart mobility. Among various challenges, one major technological challenge is the reliable sensing of the immediate surrounding in real-time.

1.1 Motivation

According to the World Health Organization (WHO), around 1.25 million people die each year in road accidents, making it the leading cause of death among people aged between 15 and 29. The major causes include over speeding, driver's distraction, driving under influence of alcohol as well as unsafe road infrastructure and inadequate enforcement of the traffic laws. Among these causes, human error and in particular distracted driving is found to be the number one reason for the road accidents [1]. In a Tri-Level study of the causes of traffic accidents [2] it was found that "human errors and deficiencies" were a definite or probable cause in 90-93% of all the incidents examined.

The fact is, evolution has not equipped the human mind with the ability to function safely at high speeds. At a speed higher than just 50 km/h, our senses are overloaded and we struggle with prioritizing and maintaining the needed focus. Furthermore, our mind has a limited budget of attention that we can allocate to certain activities. If we spend too much of this budget on one activity, for example driving carefully for long hours, the next activity will become more challenging. Several psychological studies have shown that if you have had to force yourself to do something, you are less willing or less able to exert self-control when the next challenge comes around. This well-known phenomenon has been named 'ego depletion' [3]. Therefore, driving long hours on busy roads results not only in wasted time but also compromises your productivity.

Self-driving cars could mean several things to humanity. They have the potential to transform the entire mobile ecosystem and the transport infrastructure as we know today. Today, every major city around the world recognizes the fact that it would be better off with fewer cars [4]. However, we are also aware that fewer cars cannot meet the ever-growing demand for convenient transportation. Hence, the mission is to design a mobility infrastructure which is sustainable and yet able to meet the growing transport needs. In a recent research report [5], it was illustrated that fully automated vehicles could lead to rapid growth of autonomous taxi networks. Consequently, the traditional automotive industry could be subsumed by various Mobility-as-a-Service (MaaS) platforms. This will not just revolutionize the transport industry but also the urban lifestyle. The residents would no longer rely on their personal cars but on public transport, shared cars and real-time data on their smartphones. The cost and inconvenience of a point to point mobility will dramatically decrease leading to a boost in economic as well a mental productivity. Above all, the safety aspect of mobility could improve substantially as driverless cars will rule out 90-93% of the accidents, which are a direct result of human error. However, it is true that driverless cars will not automatically result in safer roads as they only shift the responsibility of ensuring safety from drivers to designers and developers.



SAE level	Name	Narrative Definition	Execution of Steering and Acceleration/ Deceleration	*Monitoring* of Driving Environment	Fallback Performance of *Dynamic Driving Task*	System Capability *(Driving Modes)*
***Human driver* monitors the driving environment**						
0	No Automation	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a
1	Driver Assistance	the *driving mode*-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	Human driver and system	Human driver	Human driver	Some driving modes
2	Partial Automation	the *driving mode*-specific execution by one or more driver assistance systems of both steering and acceleration/ deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	**System**	Human driver	Human driver	Some driving modes
***Automated driving system* ("system") monitors the driving environment**						
3	Conditional Automation	the *driving mode*-specific performance by an *automated driving system* of all aspects of the dynamic driving task with the expectation that the *human driver* will respond appropriately to a *request to intervene*	System	**System**	Human driver	Some driving modes
4	High Automation	the *driving mode*-specific performance by an automated driving system of all aspects of the *dynamic driving task*, even if a *human driver* does not respond appropriately to a *request to intervene*	System	System	**System**	Some driving modes
5	Full Automation	the full-time performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	System	**All driving modes**

Figure 1 SAE Levels of Vehicle Automation

The topic of self-driving cars has been in research for a couple of decades and has always been received with fierce skepticism. In the last few years though, as perception technology has evolved considerably, this skepticism has by far vanished. The quickly evolving perception technology has led to the introduction of various Advanced Driver Assistance Systems (ADAS) in commercial cars thus paving the way for self-driving cars. It is now widely accepted that self-driving cars will be available commercially in just a decade. Germany, for example, has therefore passed the world's first law for automated driving to accommodate self-driving cars on the roads. It is believed though that vehicle automation will happen in steps. In Figure 1, various levels of automation, as defined by the Society of Automotive Engineers (SAE), are presented. At present, cars with partial automation (SAE Level-2), for example, Tesla-S, are commercially available. It is quite likely that in less than a decade vehicles with, SAE-Level-4, will also make their way to the market. However, to get there we still need to overcome huge legislative, ethical, technical and scientific challenges.

The biggest challenge, of all technical and scientific challenges, is to robustly sense the immediate surrounding while driving. Self-driving cars need to reliably perceive the environment, in real-time, while driving in highly dynamic situations regardless of weather or lighting. The perception technology is still in its primitive stage and computer vision remains a big challenge. The underlying algorithms are rapidly evolving but one practical limiting factor is the amount of processing power available in the vehicles. Embedded processors and computation units, normally deployed in a car, are simply not able to cope with the massive amount of data generated by various sensors, such as cameras, of a self-driving car. A number of semiconductor companies recognize this technological need of automotive companies and therefore are developing high-end computing platforms for self-driving cars. NVidia Drive-PX, Intel-GO and NXP BlueBox are examples of such programmable platforms. These platforms are, however, still in their development or early release phase with limited availability and an incomplete set of tools. General availability of such high-performance platforms will definitely accelerate the pace of a paradigm shift towards self-driving cars.

The goal of this design assignment is to make a step forward in vehicle automation while showcasing the real-time perception capabilities of NXP-BlueBox. To this purpose, a vehicle automation concept is formulated for the test vehicle, Toyota-Prius, which allows deployment of various automated driving functionalities. The concept is derived from a bottom-up approach which advocates piecing together of various subsystems to give rise to a more complex system. This assignment mainly focused on the design and realization of only one such subsystem namely 'Vision based Lane Centering Assist System' using NXP BlueBox.

1.2 Lane Centering Assist System

One severe consequence of driver's distraction is the drifting of the ego vehicle from the current lane. A brief diversion of the driver's attention from the road is enough to cause a fatal accident. According to a recent study on causes of lane drift accidents, by Insurance Institute for Highway Safety (IIHS), two most common factors in such accidents were a distracted driver and an incapacitated driver at the time of the crash. A total of 631 of such accidents were analyzed in which 34% of the accidents resulted from drivers being incapacitated while 22% of the accidents were the direct result of driver distraction. In some cases, the incapacity of the driver was completely unavoidable. In nearly one-half of the cases, the driver fell asleep while in other cases the driver had become incapacitated due to a medical emergency or due to intoxication by drugs or alcohol.

Most new vehicles these days offer automated assist systems to prevent lane-drift accidents. Many of these systems, Lane Departure Warning Systems (LDWS), are passive systems which keep an eye on the road and alert the driver about a possible lane drift through audio-visual warnings. Some more advanced systems in the market, Lane Keeping Assist Systems (LKAS), even provide corrective actions, for example, corrective steering or braking the opposite wheel, to course correct you in case lane drift is detected. Although such systems are quite useful in preventing accidents caused by a momentary distraction of the driver, they could not help in situations where the driver is incapacitated or is unable to take control of the vehicle for a longer time.

A Lane Centering Assist System (LCAS) is a proactive system that continuously steers the vehicle to keep it in the center of the current lane. It is the most advanced lane drift prevention system in the market but is not as well established as the two mentioned before. In case, the LCAS is combined with an Adaptive Cruise Control System (ACCS), an SAE-level 2 automation system will be in place. Thus, the driver can disengage, from physically operating the vehicle, for a while. He can even relax by taking his hands off the wheel from time to time. It is important to mention here that, as this is only SAE-level-2 system, the driver needs to be mentally present behind the wheel at all times. Nevertheless, in case of driver's incapacity or a medical emergency, the vehicle has more chances to safely maneuver and bring itself to a safe stop, but this is still not guaranteed by the current level of automation.

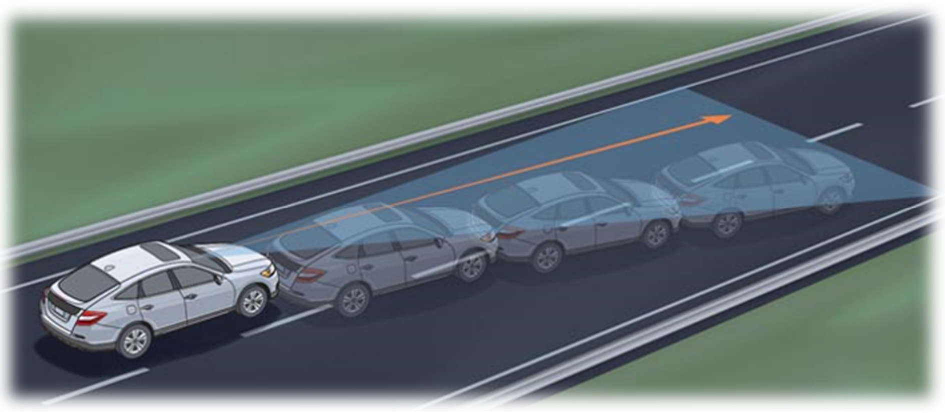


Figure 2: Lane Centering System [24]

1.3 Outline

In chapter 2, various stakeholders involved in this design assignment are introduced, along with their concerns and interests. Subsequently, in chapter 3, an analysis of various problems that need to be solved, in order to achieve an active lane centering system, is presented followed by definition of the system context. A functional safety view of the system is presented in chapter 1, establishes various safety goals that must be fulfilled by the system design, in order to ensure that the system performs in a safe way. Based on these goals, a functional architecture of the system is proposed decomposing the system into various functions. To this point, the complete system at vehicle level is considered but there is a need to limit the scope of the assignment and therefore, in chapter 5, various delimitations presented and a use case is derived. This limited scope is then used to draft various requirements for the system in chapter 6 followed by a complete system design in chapter 7. The lane detection and tracking algorithm and software application design are discussed in chapter 8 and chapter 9 respectively. Chapter 10 is dedicated to various realization related topics such as the adopted workflow and the mapping of the algorithm onto various computing units inside the BlueBox. Chapter 10 also provides details on verification and validation methods employed. Finally, in chapter 11 the report is concluded by presenting the results and recommendation on future ideas.

2.Stakeholder Analysis

This analysis was carried out to ensure that project activities will have a lasting impact on project partner organizations and the stakeholders. The key stakeholders of this project are Eindhoven University of Technology (TU/e) and NXP Semiconductors. The main interest of TU/e, in this assignment, is to set up a test vehicle with an LCAS automated system, while formalizing the concept of a self-driving car. Concurrently, NXP is providing various technical expertise as well as the prototyping platform for automated driving, which they named as 'BlueBox'. The key expectation of NXP from this assignment is to promote the usage of the BlueBox in the automotive sector and to showcase its computing capabilities. Besides the key stakeholders, concerns of developers and designers, who will further develop or interact with the LCAS, are highlighted. Moreover, various concerns of the car manufacturers are also important in order to derive recommendations for the BlueBox.

2.1 *Introduction*

In this chapter, a comprehensive analysis of different stakeholders, directly or indirectly involved in this project, is presented. This analysis was first carried out in the project initiation phase and afterward was regularly updated to ensure that each stakeholder is constructively involved in contributing towards the envisioned future. The analysis provided significant assistance in planning the design goals and in constructing a communication strategy.

In particular, this analysis tried to answer questions like what financial and emotional interests each stakeholder has in the outcome of this doctorate assignment, What are various concerns and challenges that project stakeholders hope to solve, what expectations they have from the outcome of this project and how would it affect them in a long run. Another major objective of stakeholder analysis is to build consensus, between different parties, for project priorities and deliverables. Fortunately, there were no major conflicts of interest and therefore no additional effort was required for alignment of the stakeholders.

From a reader's perspective, it is important to get an insight into the situation and conditions guiding the strategic objectives and goals of this assignment. This insight will enable the reader to understand various rationales behind design choices made. The analysis is split into four different groups which are discussed in the following sections of this chapter. The first two groups, namely the Eindhoven University of Technology and NXP Semiconductors, address organizational and business level objectives and interests. The third group, Designers, and Developers address these interests and concerns on a more personal level. Here, current developers and designers were considered, but also those who will be responsible for the continuation of the project. Finally, in the last group, various concerns and reservations from automotive OEMs are presented. These concerns and reservations will definitely help NXP in planning the future releases of the BlueBox.

2.2 *Eindhoven University of Technology*

Eindhoven University of Technology (TU/e) is committed towards a world without mobility problems this is why 'Smart mobility' is designated as one of the university's three strategic areas. TU/e holds a great deal of expertise in the fields of Intelligent Transport Systems, Automotive Technology, and ICT/ Embedded Systems. Such expertise are prerequisites for making a transition towards a smart and sustainable mobility [6]. The main goal of TU/e is to make use of its vast knowledge base to design and develop a prototype for a self-driving car, hence contributing towards its smart mobility mission.

Among various research clusters under the flag of smart mobility, 'mobile perception' is a research cluster that focuses on technologies, which will enable a self-driving car to perceive its environment. The key research areas of this cluster include computer vision, pattern recognition, supervised deep learning as well as sensor fusion. This research cluster is headed by assistant professor Dr. Dubbelman who, besides providing domain knowledge in visual perception, is also supervising this project as a stakeholder from TU/e.

In the context of this assignment, the main technological interest of TU/e is to set up a test vehicle for deploying and testing various automated driving functionalities. To this purpose, TU/e has acquired a test vehicle, Toyota Prius, from TASS international, which is one of the technology partners of TU/e. The vehicle is equipped with various advanced sensors and prototyping platforms with a vision to achieve SAE Level-4 automation in the next five years.

Dr. Dubbelman has been working on a research algorithm for vision-based lane detection and tracking, the preliminary results from the algorithm are promising. Therefore, as the first step towards a next-generation LCAS, enhancements of the lane tracking algorithm and a real-time implementation on the BlueBox was proposed. The main questions that Dr. Dubbelman is trying to answer from this assignment are:

- How to set up the test vehicle with required sensors and computing devices?
- How does the current algorithm compare with other lane tracking approaches, in context of LCAS?
- Can we generate a real-time implementation of the research algorithm for NXP BlueBox?
- How to make this algorithm scalable in order to extend it to various scenarios and use cases?

Answers to these questions will help TU/e in identifying practical issues concerning the deployment of advanced perception algorithms in an actual vehicle. The main expectation, from this assignment, is to achieve a real-time implementation of the algorithm, which can be deployed in the test vehicle for achieving an automatic lane centering functionality.

2.3 NXP Semiconductors

NXP is the leading vendor of semiconductors to the automotive sector. The rising popularity of ADAS and self-driving cars has created a huge demand for high bandwidth communication and high-performance computing in cars. NXP recognizes this need and has thus developed the BlueBox, a prototyping platform designed to provide the required performance, reliability and functional safety. The platform is, however, still under heavy development and needs in-field evaluation along with rigorous verification. In this assignment, the main interest of NXP is to fully understand various functional and performance capabilities of the BlueBox along with its limitations in the context of automated driving. Another objective of NXP is to promote the BlueBox technology in the automotive sector, through advanced demonstrations, as the most versatile enabler of self-driving cars.

An increasing demand for autonomy in cars has raged a race among semiconductor companies, to define the next generation car platform. The stakes are high and various companies, even some that have never participated in the automotive sector before, are now offering ADAS platforms. This is because any ADAS platform that could gain early acceptance will have a huge advantage if self-driving cars reach the market. The goal of NXP is, of course, to maintain its leading position and therefore it is important for NXP to understand various factors that can accelerate acceptability of the BlueBox technology in ADAS and automated driving sector. Furthermore, it is also important to highlight the performance metrics of the BlueBox, together with various capabilities and distinguishing characteristics, through real-world applications. Key performance metrics include computing performance, energy efficiency and compliance with the safety standards. However, evaluating the true computing performance of a heterogeneous system, such as the vision subsystem of the BlueBox, is an overwhelming task in itself.

In the context of this assignment, NXP would like to demonstrate true compute performance of the vision system-on-chip (SoC) in the BlueBox. The vision subsystem of the BlueBox is a heterogeneous system comprised of an Image Signal Processor (ISP), specialized computer vision cores named as APEX, graphical processing unit (GPU) as well as Arm cores for general purpose computing. In order to demonstrate its true compute power, proper balancing of the compute tasks across multiple compute devices is needed. This, in turn, requires expert knowledge of individual computing units as well as of the heterogeneous architecture as a whole. The main questions that NXP would like to answer through this assignment are:

- What is an effective workflow for developing ADAS applications using NXP-BlueBox?
- How to achieve an optimal utilization of the compute resources in the BlueBox vision processor?
- How to determine true compute power of the vision subsystem?
- How to demonstrate various competencies of vision accelerators in the BlueBox?
- How to showcase various abilities of the BlueBox and how to overcome the limitations?

Answers to these questions will help NXP in identifying various practical issues concerning the deployment of complex vision algorithms on the BlueBox. The main expectation of NXP, from this assignment, is to showcase compute capabilities of the vision subsystem in case of a lane centering assist application.

2.4 Designers and Developers

This group is comprised of developers and designers who are working on different aspects of automated driving, at TU/e and NXP, and have stakes in the system under development. It is important to note that the designer and developers are also the end users of this system as the target is to develop a prototype rather a consumer product. As each subsystem share a common context, automotive, it is essential to consider concerns from the designers and developers involved in other subsystems.

To be specific, in this group we analyzed concerns and constraints from embedded platform developers, functional safety experts, vision algorithm developers as well as system architects. For example, Dr. Andrei Terechko, a senior principal architect at NXP, is responsible for designing a fault tolerant/safe system in the context of self-driving cars using NXP BlueBox. Dr. Terechko is also supervising a Ph.D. candidate to investigate various software isolation and redundancy schemes for the BlueBox. Their goal is to ensure a fault resilient execution environment for automotive applications. Similarly, Ing. Han Raaijmakers, principle senior architect at NXP, is mainly interested in establishing an effective work-flow for developing ADAS applications using the BlueBox. One of his main expectation, from this assignment, is to establish a setup that allows to effectively code, test and profile various computing units in the BlueBox. Another major expectation is to determine a systemic approach to balancing and optimizing compute load among these units.

Furthermore, from an algorithm developer point of view, major interests include provision for adapta-bility of the algorithm as well as a quick way to profile the effect of modifications or upgrades on the functional or non-functional performance of the system. As the lane tracking algorithm is still in the research phase, there is a need for various improvements and various extensions are already planned. Therefore, one major expectation is to carry out feasibility analysis on implementing these extensions on the original algorithm. Also, this process to upgrade the algorithm is likely to continue in future, therefore a setup is expected in which new algorithmic approaches and fixes could be rapidly tested along with their effects on the system. Additionally, Anweshan Das, a Ph.D. candidate at TU/e, is working on synchronizing and logging various real-time data streams, generated from the vehicle, for later analysis. He is using RTMAPS to capture and record in-vehicle data which will be played back and analyzed for validation and benchmarking of various systems. In case of LCAS, the data will be logged and quantitatively analyzed for verification.

2.5 Original Equipment Manufacturers (OEMs)

Although car manufacturers and suppliers are not directly involved in this project, it is important to understand their viewpoint in order to derive various recommendations for further development of the BlueBox. This will also enable us to understand a number of practical challenges that car industry is facing towards commercializing self-driving cars. The main issues, in commercializing self-driving cars, involve the cost of automated driving enablers, safety, and security verification as well as current regulations and technology.

Regarding sensor technology, automotive companies are mainly concerned with determining a right set of sensors to enable a reliable and robust perception for self-driving cars. Cameras are low-cost sensors that provide high-resolution visual information, but they are quite susceptible to lighting conditions. This is why most automotive companies are skeptical about solely relying on cameras. The camera technology, however, is evolving quickly and today, we have various automotive grade High Dynamic Range (HDR) cameras can see even in very challenging lighting conditions. An alternate approach is to make use of cameras fused with the radar data, for example, Tesla auto pilot relies on a combination of cameras and radars for it autopilot system. Fusion of LIDAR, an active sensor that constructs 3D representation of the world, data with cameras and radars is also common. The fusion of various sensors could improve the reliability of the system and also provides a world model, which is easier to comprehend for the self-driving cars. Similarly, a combination of camera, High Definition (HD) maps, RTK-GNSS, which is a highly precise positioning service and an onboard inertial measurement unit (IMU) can be employed to achieve a highly automated car. It is still an open question for automotive companies that what combination of sensors could enable safe and reliable sensing for the self-driving cars.

The major technological concern of the automotive industry, when it comes to computing platforms, is how fast a platform can compute and how easy is it to develop novel features with the platform. Besides performance and power consumption, automakers are also keen about various soft factors while selecting a development platform, such as relationship, roadmap alignment, and provision of a broader software ecosystem [7]. For example, NVIDIA already has a big software ecosystem from its gaming sector and is using it to its advantage in automotive. A bigger and pre-defined software ecosystem implies an easy deployment of new ADAS applications as well as easier access to developers. This is perhaps, one of the reasons Audi and Tesla both have established partnerships with NVIDIA. Likewise, BMW has recently announced its partnership with Intel.

Once the automaker or a major Tier-1 supplier chooses a particular computing platform, the transition to another hardware platform will be a quite effortful task for the company. Furthermore, automotive companies are putting a lot of focus in verifying safety aspects of the systems that are to be installed in the vehicle. This is because, in case of an accident, caused by an automated system, the car manufacturer will be held liable. Consequently, a system that adheres to the highest safety standard will naturally receive more acceptance and will be easier to commercialize.

3.Problem Analysis

Based on the motivation and stakeholder analysis, presented in previous chapters, a problem statement is drafted along with high-level design goals. The main objective of this assignment is to design and develop a Lane Centering Assist System (LCAS) in the context of a self-driving car. It is very important to view the design problem in a bigger context of self-driving cars, as various non-functional aspects of the system owe their origin to this context. The main function of the system is to perceive its immediate environment through a forward-facing camera and then steer the vehicle to keep it centered in the lane. The system is to be realized on an embedded platform, NXP BlueBox.

3.1 Introduction

The design approach adopted at TU/e, for achieving a self-driving car, is a state-of-the-art bottom-up approach, where various automated subsystems will be integrated into a larger highly automated system. This demands acute attention on various non-functional aspects of the subsystems for example scalability, upgradability, modularity as well as maintainability. Otherwise, the bottom-up approach has fair chances to collapse because of integration failures. Regarding functional aspects of LCAS, there are various challenges involved in automatically steering a car to stay in the center of the current lane. One of the main challenges is to accurately determine the lane boundaries and laterally localize the ego vehicle.

If a car needs to position itself laterally, it needs to know where exactly in the lane it is driving, with cm-level accuracy, along with an accurate knowledge of the boundaries. In terms of localization, GPS information alone is not adequate as the precision is of the order of meters with low reliability. On the other hand, Real Time Kinematics (RTK) is a differential GNSS technique which can provide cm-level accurate positioning information in the vicinity, 10-20 km, of a base station. However, coverage and reception reliability is still a serious issue. Secondly, providing an accurate knowledge of lane boundaries for all roads is a non-trivial task. There are few companies, for example HERE and TomTom, working on high definition (HD) maps which aim at making this information available for self-driving cars through a cloud service. Although such HD maps will make the self-driving experience safer and smoother, a self-driving car cannot solely rely on availability of an external service.

A more reliable approach is to use an in-car camera to detect and track lane boundaries and to laterally localize the car in real-time. After reliably estimating the lane boundaries along with relative position and orientation of the ego vehicle, the problem boil downs to determining the required steering angle for maintaining the vehicle on a target path. The target path is continuously updated, from the estimated lane boundaries, such that the vehicle stays centered in the lane. A feedback control loop, which checks the current position and orientation of the vehicle against the target path, is therefore required to ensure that vehicle stays on the target path. However, the reliable perception of lanes is a major challenge. Although cameras provide a high fidelity view of the world, the interpretation technology needs various improvements.

3.2 Problem Statement

In the previous section, a brief description of possible schemes for realizing lane centering assist system is presented. This section focuses on camera-based LCAS, describing various challenges and obstacles that need to be overcome to execute this scheme. For visual perception of lanes, a highly reliable lane detection and tracking algorithm is required. The lane tracker must be able to track the ego lane in all highway situations, which is the intended use-case here. Furthermore, it must be robust against any kind of occlusions, lighting changes, road shadows as well as lane marking deteriorations. To this purpose, a reference algorithm is under research at TU/e, which can track lane boundaries in a given video sequence. Although results from the algorithm seem promising, it needs various functional and non-functional improvements before it can be deployed in a car.

Once the lane boundaries are estimated, camera image will be used to laterally localize the car in the lane as depicted in Figure 3. This requires accurate camera calibration and modeling. Finally, a lateral controller needs to be designed, which steers the vehicle onto a target path. Here it is worth mentioning that the control problem at hand requires a non-linear controller, as the required steering angle depends on the current speed of the vehicle. Another consideration, while designing the lateral controller, is that the lateral control objective must be defined in agreement with the vision algorithm.

In addition to these design challenges, there are implementation challenges regarding the realization of the LCAS system on NXP BlueBox. Although the vision SoC of the BlueBox is verified and thoroughly tested, the complete platform is still in an early development stage and the workflow for designing a vision-based driver assistance system is not yet well established. Thus, there is a need to define an effective workflow for rapid prototyping of advanced driver assistance systems on the BlueBox. Finally, as the BlueBox is a heterogeneous platform, a balanced load distribution and optimal utilization of different compute resources are needed to reach its full potential.

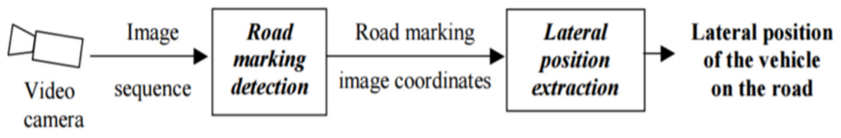

Video camera
Image sequence
Road marking detection
Road marking image coordinates
Lateral position extraction
Lateral position of the vehicle on the road

Figure 3: Lateral Localization using Video Camera

For further elaboration, the LCAS design and development problem are divided into three subsections. The first subsection describes various issues and improvements points for the lane tracker algorithm. The second subsection presents details on the lateral localization and control challenges in the context of LCAS, and finally, the third subsection lists various implementation related challenges and considerations that are specific to the BlueBox.

3.2.1. Visual Tracking of Ego Lane

A robust lane detection and tracking forms the core of a camera based LCAS system. Although camera-based lane detection and tracking is a highly active research area, current algorithms and techniques do not provide the required reliability. Furthermore, as the algorithm has to operate in conditions that are hard to anticipate, it must have a high immunity against disturbances. At TU/e, with these goals in mind, an advanced lane detection and tracking algorithm is being designed and researched. This algorithm is based on a probabilistic approach, which exploits hierarchical classification for detecting and tracking a lane. The hierarchy goes from pixel level to object level and at each level, prior probability maps are engineered to make the algorithm aware of various physical constraints of a lane. Although the preliminary performance of the algorithm is promising, optimization of probability maps is required at every hierarchical level to achieve the required functional reliability. However, the biggest limitation is its huge computation time.

As the perception technology is evolving, algorithms are growing more and more complex. Often the computation load also grows with complexity which reduces the response time of a system. In case of the driving task, however, there are strict requirements on the response time of a system. If a car needs to drive itself it must respond to a stimulus, generated by the real world, in real-time. This brings us to the question what is real-time in this situation and how much latency, time elapsed between a stimulus and the response, is permissible. To answer this question inspiration is drawn from a human perception system.

The latency of human perception system is in the range of 100-200ms [8]. This means that there is always delay of minimum 100ms in everything we see and detect with our eyes. However, humans have an exceptional ability to predict in advance which can significantly reduce the effective response time while tracking moving objects [9]. This explains why a cricketer could precisely hit a ball with his bat, speeding towards him with at 160 km/h, even though what he sees is actually 100ms late. Therefore, if a system is to drive a car, it must be able to detect any relevant stimulus in the real world strictly within 100ms. Besides, it must also be able to predict the future and at a much higher rate, for example with a maximum latency of 20ms [9].

As the predictive quality of algorithms is not comparable yet to that of a human mind, the detection requirement was made stricter, by reducing latency budget to only 60ms. The current MATLAB implementation of the reference algorithm, however, has a latency of more than 250ms on an Intel Corei7 computing platform. Moreover, as the current implementation was meant for research and quick testing only, it did not take into account various other non-functional aspects such as functional safety, modularity, and extensibility of the algorithm.

For example, in the adopted bottom-up approach, LCAS will be extended to handle the lane changes as well. In this case, a modular algorithm architecture will allow integration of new blocks to the current algorithm without the need to rewrite it from scratch. This demands a scalable algorithm architecture that divides the overall challenge to modular sub-problems. These individual puzzles thus can be solved and put together in order to achieve the desired solution. This will ensure easy adaptability and optimization of the individual modules which is indeed is very important if we consider LCAS as a subsystem in context of self-driving cars.

3.2.2. Lateral Localization and Control

After determining the position of lane boundaries and a target path in the acquired image, the next step is to extract the exact position and heading of the car in the lane. To this purpose, coordinates of the lane boundaries and the target path in the world coordinate system need to be estimated. This requires an accurate camera calibration; the optical characteristics of the camera as well as its relative position and orientation in the world causes pixels to have a different meaning, depending on the location of the pixel in the image plane. An accurate camera calibration accounts for these perspective effects and maps the pixels from the image coordinate system to a world coordinate system. Also, as a wider-angle lens is normally employed in the front facing camera, to cover a wide area, the radial distortion effects are expected to be significant and must also be accounted for.

Certain assumptions about the road geometry and the shape could simplify the extraction of the lateral position of the car from the image. For example, assuming a flat road makes the task simpler as plane projective mapping [10] can be used to transform the image coordinate to the world coordinate system. This, however, may produce inaccurate results because of unaccounted radial distortion of the lens. Another simpler calibration method is presented in [11] which accounts for the lens distortion, in addition to the perspective effect in the image, using scaling factors for every column in a row. The principle here is that the lateral position or error is typically measured along one predefined row in the image, therefore an accurate calibration of this row suffices. The calibrated row represents the look-ahead distance of the LCAS system in the world coordinate system and needs to be determined beforehand. Additionally, an unexplained variance in the lateral localization may be observed, due to the varying pitch angle of the camera, in case of a bumpy road. Therefore, the effects of camera pitch on the calibration need to be accounted for by incorporating vehicle pitch information in the process.

The camera is typically installed on the vehicle so that its optical axis is aligned with the symmetry plane of the vehicle as shown in Figure 4. This way, the heading angle of the car can be easily compared with the heading of the target path. If the car is traveling exactly on the target path, with the camera

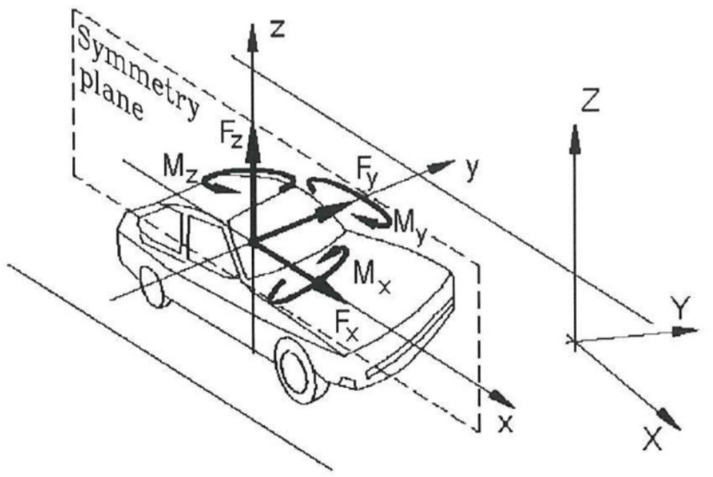


Figure 4: Description of Symmetry Plane [25]

mounted on the symmetry point of the vehicle and the optical axis aligned with the symmetry plane, the target point at a look ahead distance should appear in the middle of the calibrated image line. This way, any displacement of the target point on the calibrated line in the image can be directly translated into a lateral error at the look-ahead distance. This lateral error has to be minimized according to an appropriate control strategy. The path-following problem is a well-established control problem and has been extensively researched in robotics as well as automotive. The lateral control problem is a nonlinear control problem and there exist various approaches, nicely summarized in [12], to deal with the issue, however, suitability of these approaches in case of the LCAS needs to be evaluated to select an optimal controller.

3.2.3. Realization on NXP BlueBox

One major goal of the project is to evaluate the NXP BlueBox in term of its fitness for advanced driver assistance systems and self-driving cars. BlueBox will function as the central computing engine of the system and therefore without a detailed analysis of the platform this chapter would be incomplete. The platform is an integrated package for automated driving and is comprised of two independent systems on chip (SoCs). The S32V234-Safety processor is a platform which provides the required performance and reliability needed for perception applications, while LS2085A is an embedded computing processor that provides a high-performance data path and network interfaces to connect with the outer world.

The S32V234 belongs to the family of processors designed to support data-intensive applications like image processing. It is a highly heterogeneous architecture with various camera interfaces, an Image Signal Processor (ISP), a Graphical Processing Unit (GPU), two dedicated APEX cores designed to accelerate computer vision functions along with various safety and security features. A detailed block diagram of the system is presented in Figure 5.

The major challenge in porting an algorithm on the vision system is to optimally map it to various units in the system in order to achieve an overall boost in the performance. This requires an intimate knowledge of the individual processors, their capabilities, and limitations. For example, APEX processors are highly parallel computing units, with Single Instruction Multiple Data (SIMD) architecture, and can handle data level parallelism quite good. This makes them an ideal candidate for filtering tasks, for instance, where a kernel needs to be convolved over the whole image. In contrast, when the algorithm needs to randomly access pixels in an image, there will be no gain to use the APEX cores as the overhead of transferring the image to APEX cores and getting them back will reduce the overall performance. Similarly, ISP is a Multiple Instruction Multi Data (MIMD) architecture, which can achieve instruction level as well as data level parallelism, but due to the limited memory and processing power only essential preprocessing functions, for example, color conversions or rescaling the image, are targeted for this pipeline.

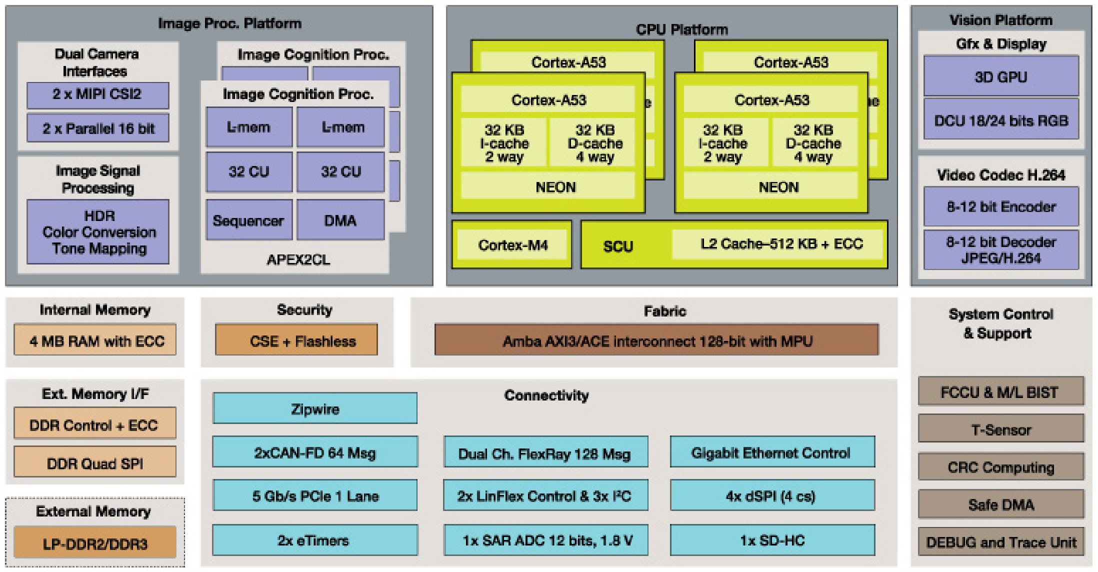


Figure 5 : Block Diagram of the S32V234 Safety Processor of BlueBox

The vision application, thus, needs to be analyzed for fitness of each function or a group of functions on a certain compute unit. Furthermore, as the APEX and ISP processors do not support floating point arithmetic, the corresponding calculations must be represented by an equivalent scaled integer form of fixed precision. This, in turn, demands a static and dynamic range analysis together with precision loss analysis for every variable, which is a non-trivial task. Moreover, various exponential and trigonometric functions must be approximated by simpler functions or numerical approximations if needed to execute on the accelerators.

3.3 Project Goals and Objectives

Based on the problem analysis the following six high-level objectives are identified that lead to a next-generation LCAS. These objectives provide a way to organize the project, for example into milestones, and are used to define the priorities of the subsequent activities. Furthermore, guidelines and main concerns corresponding to every objective are also included.

1. Functional Safety Analysis of LCAS

Functional safety is concerned with the overall safety of a system and focusses on ensuring correct behavior and operation of the components and subsystems. With the rise in the safety systems in the car, for example, lane keeping systems, functional safety analysis is becoming more and more important. The concept of functional safety advocates a safety-oriented design of the system that ensures minimal faults. It also includes safe handling of the faults, operator errors, environmental disturbances as well as hardware and software failures. As LCAS is a highly safety critical system it is important to analyze various safety risks before committing to the design of the system.

2. System Architecture of LCAS

After a thorough analysis of the problem and the relevant safety risks, a system architecture needs to be designed. The architecture involves logical placement of different components such that together they can achieve the desired functionality. Here it is important to keep in mind the context of self-driving cars as well, if the design is too specific to the functionality it is hard to adapt it later. Conversely, if the design is too generic and inclusive it may not meet the desired performance specifications. This trade-off should be solved by designing at an appropriate abstraction level.

3. TU/e Lane Tracker Optimization

The lane detection and tracking algorithm is a fundamental part of the system. The algorithm needs various optimizations as well as an overall good design to enable a real-time implementation of the algorithm. Without developing a concrete understanding of the algorithm developing a computation cost-effective design for the algorithm is not possible. Therefore, it is important to develop an extensive understanding of the algorithm to achieve this objective.

4. Software Application Design and Implementation

The software application is a container of the algorithm and its main responsibility is to execute different stages of the algorithm. However, the software is also responsible for many other transactions, for example, communication with the lateral controller. Perhaps, the most important quality aspect of a safety-critical software is its predictability. In an automotive grade, real-time application, where the lives are at stake, the software must have deterministic behavior. The requirements on other non-functional quality attributes, for example, modularity, maintainability, and reliability, are also strict. Therefore, appropriate measures should be taken while designing and implementing the software.

5. Lateral Control Design and Implementation

The software application is supposed to execute the algorithm and get the current and target heading of the vehicle. The difference between these headings must be calculated in a format that is appropriate for the control strategy. The control strategy is defined by a lateral controller and is responsible for ensuring that the vehicle stays on the target path. An appropriate control strategy, thus, need to be worked out here.

6. System Integration and Demonstration on Toyota Prius

Finally, the system needs to installed and deployed inside the Toyota Prius. An integration plan for the interfacing of various devices and subsystem must be carried out. In case of any inconsistencies or technical incompatibility, the system design might need to be updated or adjusted. Furthermore, a demonstration of the system on Toyota Prius is expected.

3.4 System Context

In previous sections, high-level challenges and objectives are presented along with various issues that need attention. The goal of this section is to provide a better understanding of the context in which the problem is to be solved. To this purpose, a system context diagram is presented in Figure 6, which is an outward facing view of the system the treats the system as a black box and tries to capture its interactions with the environment. A system context diagram helps in identifying the system boundaries and hence in defining the scope of the system.

As depicted in Fig.5, LCAS needs to interact with various other systems in order to reliably automate the lateral control of a vehicle. The driver will interact with the system through a Human Machine Interface (HMI), for example, to activate the system. Additionally, HMI is also responsible for keeping the driver informed of the current situation and alert him to an emergency situation. A front-facing camera detects and tracks the lane boundaries while RTK-GNSS and IMU sensors reinforce the process and are meant to ensure the validity of tracking results. Besides, it is important for developers that data from various sensors can be synchronized and logged for post analysis and laboratory validation. A physical view of various systems installed in the test vehicle along with their interconnections is depicted in Figure 7 for further clarity.

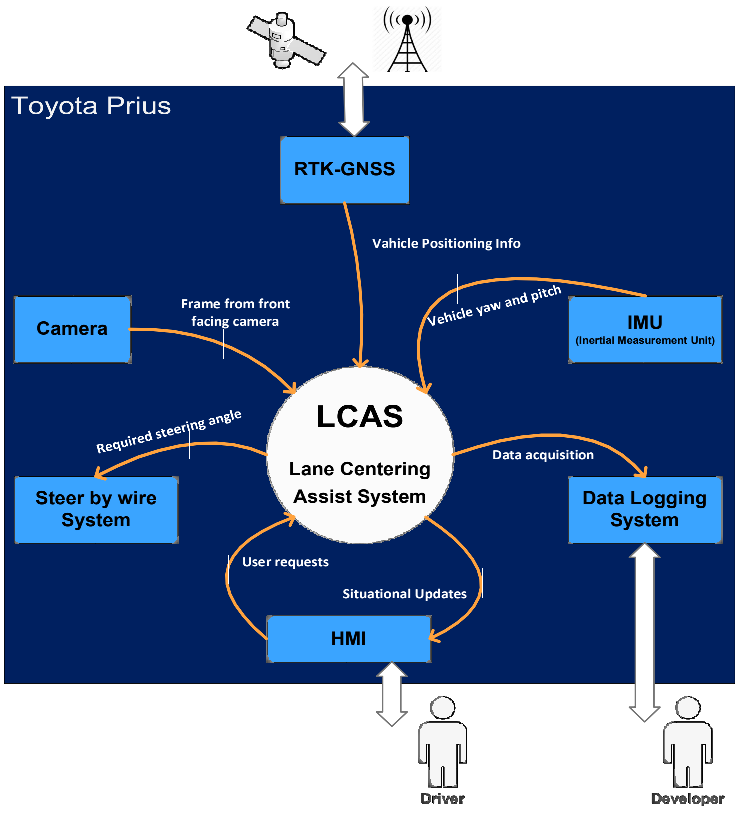


Figure 6: System Context Diagram

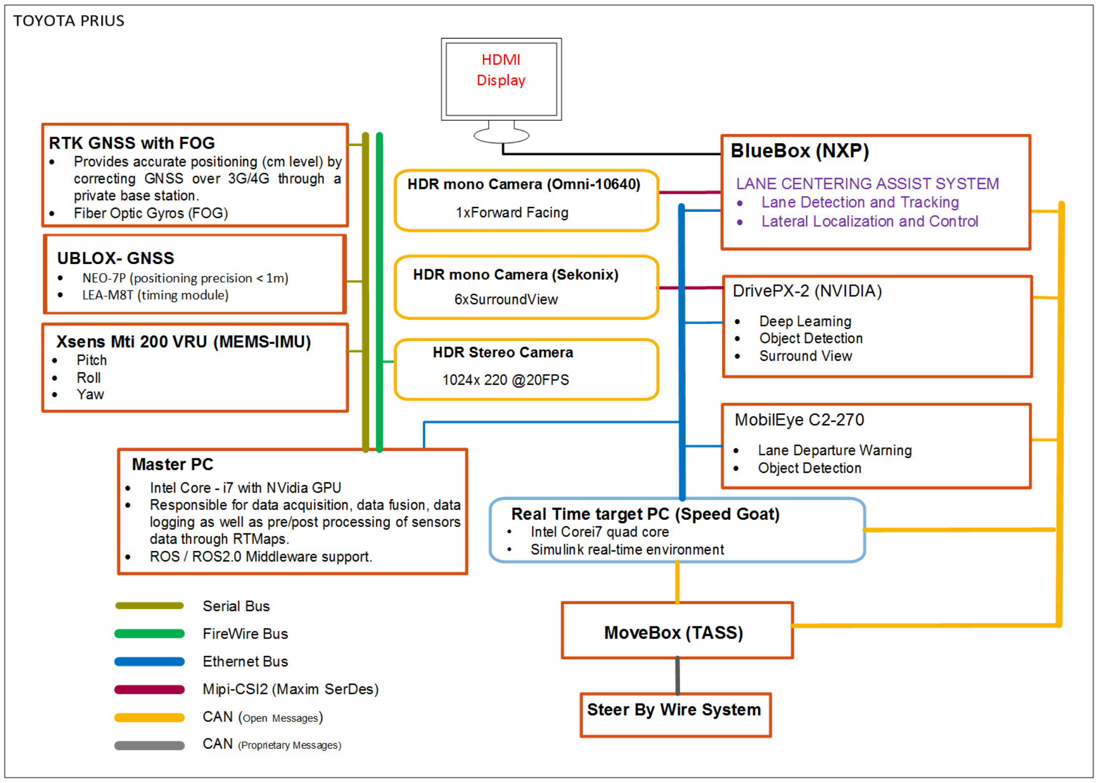

TOYOTA PRIUS
HDMI Display
RTK GNSS with FOG
Provides accurate positioning (cm level) by correcting GNSS over 3G/4G through a private base station.
Fiber Optic Gyros (FOG)
UBLOX- GNSS
NEO-7P (positioning precision < 1m)
LEA-M8T (timing module)
Xsens Mti 200 VRU (MEMS-IMU)
Pitch
Roll
Yaw
HDR mono Camera (Omni-10640)
1xForward Facing
HDR mono Camera (Sekonix)
6xSurroundView
HDR Stereo Camera
1024x 220 @20FPS
BlueBox (NXP)
LANE CENTERING ASSIST SYSTEM
Lane Detection and Tracking
Lateral Localization and Control
DrivePX-2 (NVIDIA)
Deep Leaming
Object Detection
Surround View
MobilEye C2-270
Lane Departure Warning
Object Detection
Master PC
Intel Core - i7 with NVidia GPU
Responsible for data acquisition, data fusion, data logging as well as pre/post processing of sensors data through RTMaps.
ROS / ROS2.0 Middleware support.
Real Time target PC (Speed Goat)
Intel Corei7 quad core
Simulink real-time environment
MoveBox (TASS)
Steer By Wire System
Serial Bus
FireWire Bus
Ethernet Bus
Mipi-CSI2 (Maxim SerDes)
CAN (Open Messages)
CAN (Proprietary Messages)

Figure 7 : System Context Diagram – A Physical View

4.Functional Safety Analysis

The system context presented in the previous chapter gives a synopsis of various interactions of the system under design with its environment. As these interactions could compromise the safety of the driver, it is critical to ensure their correctness in terms of functional safety. For such safety-critical systems, it is important to understand the concept of functional safety. Functional safety is concerned with the overall safety of a system and focusses on ensuring correct behavior and operation of the system as well as its components. It also includes safe handling of operator errors, environmental disturbances as well as hardware and software failures. It is crucial to consider the aspects of functional safety before making any design choices, as they can lead to serious safety repercussions.

4.1 Introduction

A major concern of the automotive industry is ensuring the safety of the passengers. With the rise in driver safety systems, for example, lane departure warning systems and collision avoidance systems, the concept of functional safety is gaining popularity. The automotive industry is expected to provide new and improved vehicle safety systems that can ensure the desired functionality with minimal faults. In November 2011, the automotive industry introduced a new standard on functional safety titled 'Road vehicles – Functional safety ISO26262'. This safety standard deals with electrical and electronic systems in production automobiles. As around 70 - 90% of all innovations in automotive nowadays is based on embedded systems [13], demands for ISO26262 compliant systems is growing fast.

An ISO26262 compliant system ensures that state-of-the-art processes and practices were followed while designing and developing a system. However, ISO26262 is more than just a collection of best practices and principles. It is a complete systems engineering approach and guides the process from concept to completion. Since there exist various system engineering approaches, the main motive behind the development of ISO26262 is to agree upon one basic understanding of the approach for developing safety-critical systems. As depicted in Figure 8, the standard is divided into various parts, however, in this assignment, the focus was mainly directed towards the concept phase and product development at the system level and software level.

The safety lifecycle of a system starts by defining the system, at the vehicle level, which is to be evaluated for functional safety. This is known as 'item definition' in ISO26262 vocabulary. The item definition also provides the context for the system. The next step is to carry out Hazard and Risk Analysis (HARA) for the item. The hazard analysis provides a list of safety goals, which are the top-level safety requirements represented in terms of functional objectives. The safety goals are derived by exploring every possible malfunction at the system level and the corresponding risks. Every risk is assessed in terms of severity, exposure, and controllability to assign it an Automotive Safety Integrity Level (ASIL). This is followed by defining a functional architecture of the system where the responsibility of the item is decomposed among various functional units. The functional architecture is then used to define a Functional Safety Concept (FSC) which is a list of all the Functional Safety Requirements (FSR) derived from the safety goals. The FSRs inherit ASIL specification from the corresponding safety goals, mapped onto specific units of the functional architecture. If necessary, the inherited ASIL can be decomposed by creating redundant requirements for the system. In case that redundant requirements are allocated to different components if one of the components fails to satisfy its requirement, the other component may still do so, thus improving system integrity [14].

At the end of the concept phase, the FSRs are translated into technical specifications and a system design is carried out which represents the left side of the "V" in product development at system level represented in Figure 8. Similarly, design and implementation of the individual hardware and software components are also carried out in accordance with 'V' development model. Finally, product development phase is concluded by integration of the components and system level validation represented by the right half of the 'V' in system level product development.

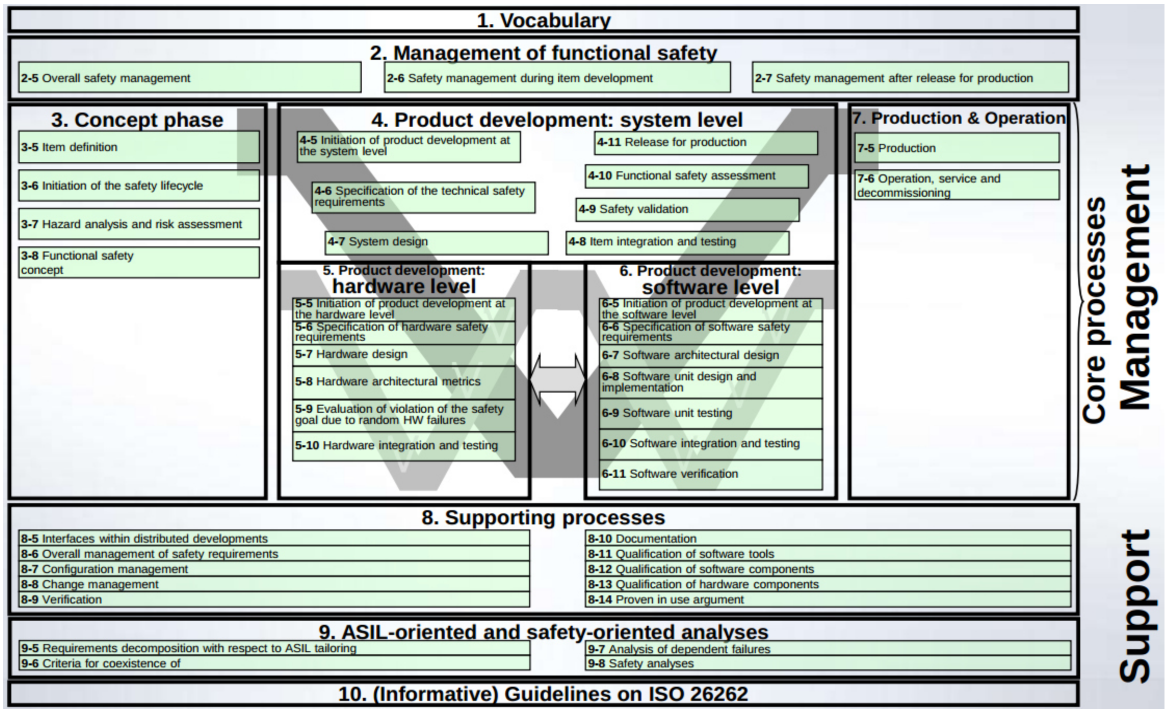

1. Vocabulary
2. Management of functional safety
2-5 Overall safety management
2-6 Safety management during item development
2-7 Safety management after release for production
3. Concept phase
3-5 Item definition
3-6 Initiation of the safety lifecycle
3-7 Hazard analysis and risk assessment
3-8 Functional safety concept
4. Product development: system level
4-5 Initiation of product development at the system level
4-6 Specification of the technical safety requirements
4-7 System design
4-8 Item integration and testing
4-9 Safety validation
4-10 Functional safety assessment
4-11 Release for production
5. Product development: hardware level
5-5 Initiation of product development at the hardware level
5-6 Specification of hardware safety requirements
5-7 Hardware design
5-8 Hardware architectural metrics
5-9 Evaluation of violation of the safety goal due to random HW failures
5-10 Hardware integration and testing
6. Product development: software level
6-5 Initiation of product development at the software level
6-6 Specification of software safety requirements
6-7 Software architectural design
6-8 Software unit design and implementation
6-9 Software unit testing
6-10 Software integration and testing
6-11 Software verification
7. Production & Operation
7-5 Production
7-6 Operation, service and decommissioning
Core processes
Management
8. Supporting processes
8-5 Interfaces within distributed developments
8-6 Overall management of safety requirements
8-7 Configuration management
8-8 Change management
8-9 Verification
8-10 Documentation
8-11 Qualification of software tools
8-12 Qualification of software components
8-13 Qualification of hardware components
8-14 Proven in use argument
9. ASIL-oriented and safety-oriented analyses
9-5 Requirements decomposition with respect to ASIL tailoring
9-6 Criteria for coexistence of
9-7 Analysis of dependent failures
9-8 Safety analyses
Support
10. (Informative) Guidelines on ISO 26262

Figure 8: Overview of ISO26262, Functional Safety Standard for Automotive [23]

4.2 Item Definition

The main safety function of a Lane Centering Assist System (LCAS) is to detect an unintentional drift from the lane on which it is traveling and to actively steer the vehicle back to the center of the lane. The system is also capable of taking the steering control, provided the operational conditions satisfies, on request from the driver using a Human Machine Interface (HMI). The LCAS is supposed to evaluate the operating conditions on its own and enable the HMI interface as well as the safety function. Once enabled the system keep an eye on the road and is capable of steering the vehicle when required or when requested. The system primarily uses the camera to detect lane markings, however, vehicle sensor data like yaw, pitch acceleration, GNSS position and wheel speed are also employed to improve the system reliability as well as to verify the current and target path generated by the visual perception of lane boundaries. The intention of the driver, to leave a lane is detected by the evaluating the indicator signal in which case, the automatic steering of the vehicle is not commissioned. Additionally, the driver can always override the system by applying and maintaining certain torque on the steering wheel.

The status information of LCAS is provided to the driver by means of audio-visual warnings and alerts. Although LCAS can autonomously control the lateral movement of the vehicle and assist in keeping it centered in the lane, the responsibility for the safe operation of the vehicle is primarily the responsibility of the driver. Hence, the driver needs to stay always alert to take control of the vehicle in emergency situations or situations that are novel for the system. In such situations, the system must alert the driver in time and fall back to a safe-operation mode. To limit the scope of LCAS, it is only enabled on high-ways with the forward driving speed of at least 50 km/h but less than 150 km/h. Furthermore, clear weather and lighting conditions are assumed.

4.3 Hazard Analysis and Risk Assessment (HARA)

To formulate a list of safety goals, Hazard Analysis and Risk Assessment (HARA) was carried out. A considerable effort was made to simplify the results of the analysis and have as few goals as possible. This is mainly because a thorough but incomprehensive list of safety goals is of little use for the system design process. The goal of safety analysis is to assist the design process, by providing a list of clear goals while covering every plausible scenario resulting from LCAS in the vehicle. Various driving condition and modes, considered in the process, are provided in Appendix A. The resulting ten safety goals at the vehicle level are presented in the following list along with their ASIL level.

SG1: LCAS must stay disabled unless all operational conditions are satisfied. (ASIL-D)

SG2: LCAS must alert the driver if the request from the driver to commission the active steering failed. (QM)

SG3: LCAS must ensure that active steering is commissioned automatically, without exception, when the vehicle is inadvertently straying off the lane. (ASIL-D)

SG4: Driver must be able to seamlessly override the LCAS steering inputs by maintaining a counter torque on the steering wheel at any stage of the system operation. (ASIL-D)

SG5: LCAS must regularly monitor its status as well as health, and inform the driver in case of any state change. (ASIL-D)

SG6: LCAS must ensure lateral stability of the vehicle when actively steering. (ASIL-D)

SG7: LCAS must ensure the integrity of the target path. The lateral error between the target path and the actual lane center must stay within a bound of +/-10cm while the heading error between the lane and the target path must be within +/-5 degrees. (ASIL-D)

SG8: LCAS must ensure that the vehicle follows the target path adequately. The lateral error must stay within a bound of +/-5cm while the heading error should stay within +/- 2 degrees. (ASIL-D)

SG9: Before disabling the driver interface, for example, when operating conditions are not satisfied anymore, LCAS must first decommission the active steering and wait for the driver to take control while driving in failsafe mode. (ASIL-D)

SG10: If the shutdown option is checked, LCAS must be nonexistent and must not influence or assist in any driving task. (ASIL-D)

4.4 Functional Safety Concept

As established in 4.3 , the LCAS is predominantly an ASIL-D system. In the functional safety concept, first, the safety goals are used to decompose the system into functional units followed by the derivation of FSRs which inherit the ASIL level from the safety goal. A safety requirement can, however, be made less strict by providing redundant components in the systems that also fulfill the requirements. This process is called ASIL decomposition and the possible combinations are presented in Appendix B. The FSRs are further refined by the technical specification and the principle of decomposition can also be applied here.

In the context of LCAS, the scope did not allow to follow the complete process until the evaluation of technical specifications, consider chapter 5 for details. However, a functional architecture of the system is derived based on the safety goals. This functional decomposing of the system is presented in Figure 9, on the left side, various sensors and inputs needed by the LCAS are represented while at the bottom are the outputs of the system. The container, in the center, represents various logical functions that need to be executed in order to achieve the desired functionality of the system. The interdependencies between various function blocks as well as the order in which they are evaluated are also evident from this functional decomposition.

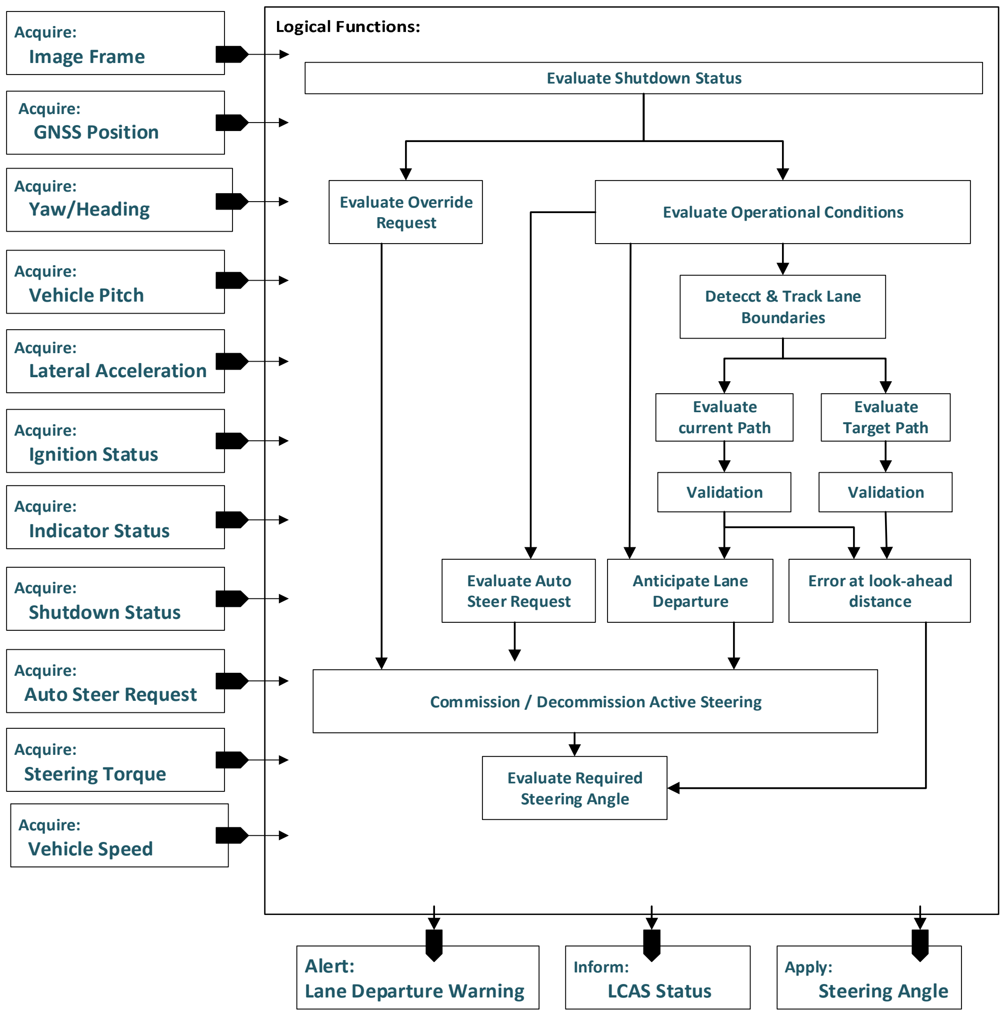

Acquire: Image Frame
Acquire: GNSS Position
Acquire: Yaw/Heading
Acquire: Vehicle Pitch
Acquire: Lateral Acceleration
Acquire: Ignition Status
Acquire: Indicator Status
Acquire: Shutdown Status
Acquire: Auto Steer Request
Acquire: Steering Torque
Acquire: Vehicle Speed
Logical Functions:
Evaluate Shutdown Status
Evaluate Override Request
Evaluate Operational Conditions
Detecct & Track Lane Boundaries
Evaluate current Path
Evaluate Target Path
Validation
Validation
Evaluate Auto Steer Request
Anticipate Lane Departure
Error at look-ahead distance
Commission / Decommission Active Steering
Evaluate Required Steering Angle
Alert: Lane Departure Warning
Inform: LCAS Status
Apply: Steering Angle

Figure 9: Functional Decomposition of the System

5.Scope Analysis and Delimitations

To this point, the problem analysis, as well as the functional safety analysis, were kept unconstrained of the time and design efforts requirements. The defined safety goals are evaluated on the vehicle level and so is the functional architecture for a full-fledged LCAS. However, a complete implementation of all functional blocks is not possible in the allotted time of the project. This chapter includes the time and resource constraints to set priorities and define the scope of the project. The aim of this chapter is to sufficiently inform the reader about the limits and coverage of this study along with the limiting factors. The analysis in previous chapters was deliberately kept broad as this project is expected to be continued in future and hence a detailed analysis will provide a head start for the subsequent projects, for example, it would be helpful in defining the goals of the future projects.

5.1 Introduction

It is important to narrow down the scope of the design problem and set priorities in order to improve the focus and hence the quality of the work. Often the scope and feasibility analysis aim at establishing a balance between the scope, the cost or resources and the allocated time for the project. In project management terms, this is normally known as the triple constraint triangle and is depicted Figure 10. The triple constraint triangle represents a model of the constraints of a project. The constraints are the areas where typically compromises or changes are introduced. In Figure 10, the attribute quality in the middle indicates that it is not negotiable while scope, time and cost on the vertices indicate that we need to balance these constraints in order to reach the desired quality. In this project, however, the allocated time, as well as resources, are fixed. Therefore major changes or compromises need to be made on the scope in order to ensure the product quality. It is worth mentioning here that this balancing of the scope was an iterative process. The project plan and progress were regularly evaluated against available time and resources to adjust the scope accordingly. The scope provided here is the result of final iterations.

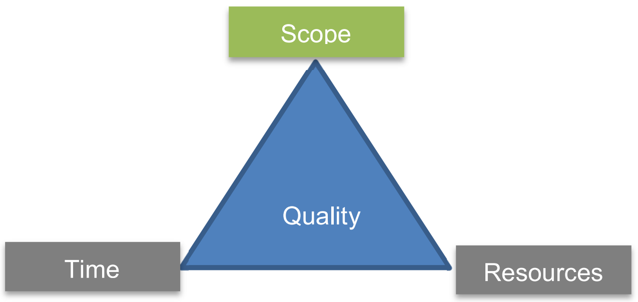


Figure 10: Triple Constraint Triangle

5.2 Limiting Factors

Before describing various delimitations, it is necessary to understand the process opted and the factors that lead to the scope limitation. As the design deals with highly innovative concepts, a static time allocation was not feasible. An iterative scope limitation strategy was employed to maximize the deliverables. Initially, the scope was kept broad and priorities were assigned to the activities needed for a full-fledged LCAS. To adjust the scope, activities with highest priorities were executed and the progress was evaluated with respect to the set goals and remaining time. Besides the delimitation originating from the project progress, some of the risks, that could delay the project, were already identified and taken into account for scope limitation.

The major risk was infeasibility to attain a real-time implementation of the state-of-the-art algorithm on the BlueBox. As the state-of-the-art algorithm is written for research purposes, it employs the computing resources extravagantly. However, the BlueBox, like any other embedded platform, has very limited resources and cannot afford such computations. Also, a pre-analysis of the algorithm revealed that it makes use of complex MATLAB constructs which could be very time consuming to implement in a high-performance programming language like C/C++. After consulting with the experts, it was established that this is the biggest risk in the project but at the same time it was the core enabler of the whole system and has the highest priority.

Furthermore, the BlueBox platform is still under development and in particular, the vision subsystem is never employed in any project of a similar scale. This is why the toolchain is incomplete and the workflow is not yet well established. The APEX and ISP accelerators have their own unique architecture, as they are designed specifically for image processing, and thus needed to be well understood. To make it more complicated, there were no experts amply available at the facility, who could be employed to assist the development on the special purpose hardware. This meant a steep learning curve which can consume a major part of the allocated project time.

5.3 Scope Definition and Delimitations

This study covers the complete design of the lane centering system with component-level details, however, the implementation of functional components is limited by this project scope. Realization of only the high priority functional units is covered in this project and rest is left for the future. However, a basic functional demonstration of the next generation LCAS is still in the scope of this project.

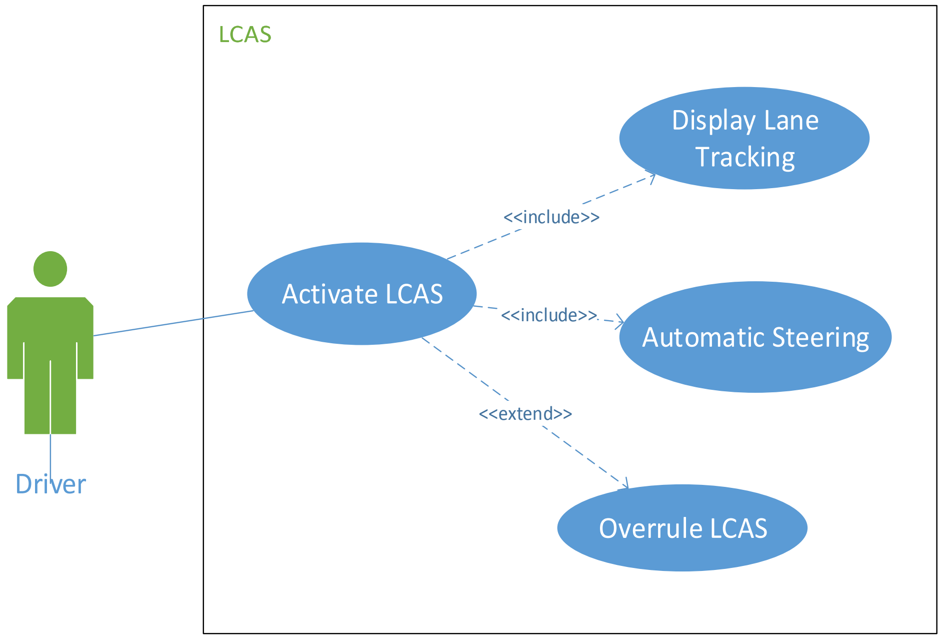
In the current scope of the project the sensor fusion, which integrates the visual information with the RTK-GNSS and IMU data from the vehicle is not covered. This means that in Figure 9, the functional units related to the verification of the current vehicle path and target vehicle path are not investigated further. Also, a supervisory control that automatically commissions/decommissions the active steering in the vehicle is not included in the current scope. Likewise, implementation of Human Machine Interface (HMI) is not covered, however, the interaction between the driver and the system will be provided as a simple activation and deactivation interface to the system. The system is enabled manually by the driver and the responsibility for ensuring the operational conditions, for example ensuring that camera is not occluded by mud or that visibility is not compromised, lies with the driver. Moreover, the realization and formal verification of the safety goals is also left as a future task. These goals, however, will be considered while deriving the requirements for the functional units that are covered in the scope of this assignment.

As various safety goals will be compromised in this scope, it is highly recommended to drive the vehicle only on a test track rather on an actual road. The current scope encompasses, a robust and real-time implementation of lane detection and tracking algorithm, a separate implementation for the BlueBox that exploits the hardware accelerators, estimation of the current path and a target path from the visual scene, generation of lateral error for the controller along with implementation and deployment of a lateral controller on the BlueBox. Additionally, the target path will be displayed in real-time on an HDMI screen inside the car. However, it is important to mention that a detailed testing and verification of the controller and consequently, that of the complete system is not included in the current scope. The controller design was kept simple and is only included to demonstrate the use case of LCAS. An advanced controller will be needed to ensure that the vehicle can steer optimally to maintain the target path. For this reason, the investigated approaches for lateral control are put in Appendix A, rather as the main contribution of this project.

The scope also encompasses verification of functional and nonfunctional aspects of the components designed and implemented in the project. For example, a software application for tracking lane boundaries will be designed and implemented; ensuring various quality attributes of this software application is covered in this scope. Furthermore, analyzing and implementing the ISO26262 requirements of the components such as lane tracking software application is also covered in this scope.

5.4 *Focused Use Case*

After taking into account various delimitations presented in the previous section, a focused use case is derived which will be considered further for deriving the technical requirements and specifications for the project. A UML Use Case diagram is presented in Figure 11, which highlights the interactions between the system and the driver and is self-explanatory.



- UseCase: LCAS1
- Name: Activate LCAS
- Scope: LCAS
- Level: User Goal
- Primary Actor: Driver
- Preconditions: LCAS is installed and running.
  Vehicle is driving on the highway with a speed of 50 Km/h in a lane.
  Driver is controlling the throttle and brake but not the steering.
  Lanes are visible on the HDMI Screen.
- Success Guarantees: Vehicle is driving in the lane without manual interventions or without any incident.
- Main Success Scenario:
  1. Driver: Activates the LCAS
  2. LCAS: Maintains the lane it is driving in.
  3. LCAS: Displays the Tracking Result on the HDMI display along with the target path.
- Extension: LCAS will be overruled if the driver provides manual steering.

Figure 11: Focused Use Case of LCAS

6.Requirements Elicitation

In chapter 1, various safety goals at vehicle level were evaluated followed by definition of the current scope in the previous chapter. Keeping in mind the stakeholder's concerns, safety goals, the current scope as well as the overall objectives, the detailed technical specification for the system are obtained in this chapter. These requirements and specifications are used as guidelines in defining the overall system design as well as in realization of the focused use-case.

6.1 Introduction

Requirement elicitation is the process of gathering requirements for a system mainly based on stakeholder concerns and customer objectives or goals. The word 'gathering' here could be misleading though, as the elicitation is not just a process of asking a customer what the system must do or must NOT do. It is a non-trivial task that involves careful analysis of various customer wishes, feasibility analysis, expert knowledge, viewpoint hopping, i.e. the ability to think from various perspectives, and above all, an ability to summarize the findings as a set of useful and useable SAMRT[1] requirements. Furthermore, it is an iterative process and the requirements listed here are gathered and refined incrementally. For better comprehension, the requirements are grouped into various categories and are listed in the next section.

6.2 Requirements

The requirements in this section are derived in order to achieve the current use case. However, acute care was taken that the requirements are consistent and are still usable in case of the proposed full-grown LCAS. Although the functional safety goals were considered, they are not directly linked to these requirements due to scope delimitations. Nevertheless, for a full-grown LCAS, it is highly recommended to link the requirements to the safety goals to ensure that each safety goal is satisfied. It is highly likely, that new safety-critical requirements might come to the surface through this process. The requirements presented in this section, hence, are a reduced set of functional and nonfunctional requirements representing only the focused used case. For clarity and better comprehension, the requirements are bundled in three categories, 'System Design', 'Algorithm Design' and 'Software Application Design'. The System Design requirements are the constraints and design specification on the system level. The 'Algorithm Requirements' represent the constraints and quality expectation from the algorithm design and likewise the 'Software Application Design'. As testing and verification at the system is not a part of the scope these are strictly design requirements. The requirements at the algorithm and software level, however, also include testing and implementation requirements.

6.2.1. System Design

REQ1. The top-level logical design of the system must be modular and upgradeable.

REQ2. Relation of various safety goals with the logical blocks must be made explicit by assigning every safety goal to at least one module.

REQ3. The system design must be reducible that is, focused use case must be derived from the full-fledged LCAS design.

REQ4. In the current use case, the system must be able to steer the vehicle, keeping it centered in the lane, on a highway with a minimum radius of 200m.

6.2.2. Algorithm Design, Implementation and Testing

REQ5. The algorithm must reliably track the lane boundaries with an uncertainty not more than +/- 5cm while driving on a highway with a radius of at most 200m.

REQ6. The algorithm must evaluate a target path and display it in real time.

REQ7. The system must generate a lateral error, the error between the current path and the target path at a frequency on not less than 15Hz.

REQ8. The total latency of the lane detection, the time interval between a stimulus and generation of the lateral error must not be more than 60ms. This time also includes the delay caused by the camera.

REQ9. The design of the algorithm must be modular and easier to extend.

REQ10. The algorithm must be split into testable modules following the 'divide and conquer' paradigm for algorithm design.

REQ11. The algorithm modules must be subjected to dependency analysis and independent modules must be executed in parallel for performance gains.

REQ12. The implementation of algorithm modules must be tested and verified independently.

REQ13. The complete implementation of the algorithm must be tested and verified using a sequence of image frames.

REQ14. The execution of the complete algorithm must be profiled over several hundred image frame and corresponding minimum, maximum and average time must be recorded.

6.2.3. Software Application Design, Implementation, and Testing

REQ15. The software application must provide decoupling between the algorithm and the control flow.

REQ16. The software design must support multiple algorithm versions which can be chosen at the compile time.

REQ17. The software design must be hierarchical and completely object-oriented.

REQ18. The individual class design must follow the principle of single responsibility and the principle of least knowledge from software engineering.[2]

REQ19. A clear graphical description of the control flow must be carried out for inspection and static analysis.

[2] Please see chapter 9 for detailed explanation of the principles.

REQ20. The design must provide at least two parallel paths for algorithm computation.

REQ21. The software design must provide a way to synchronize the parallel paths.

REQ22. The functions within a software component must be closely related.

REQ23. The size of a software component, for example, class, must not exceed 200 lines.

REQ24. The flow control of the software must be analyzed through inspection and tests.

REQ25. Software functions must have only entry and one exit point.

REQ26. The application must not have any virtual functions

REQ27. The use of pointers must be minimized.

REQ28. Software must not use raw pointers, only smart pointers are allowed.

REQ29. Software must not have any global variable.

REQ30. Every variable in the class must be initialized in the initializer list.

REQ31. There shall be no implicit type conversion.

REQ32. Software must show the camera feed on the screen for the driver at all times when the application is running.

REQ33. Software must provide a way to activate and deactivate lane tracking.

REQ34. Software must provide a way to activate and deactivate lateral error generation.

7.System Architecture

In chapter 1, functional safety goals were evaluated which lead to a functional architecture of the system Figure 10. This functional architecture is used to establish a system design for the LCAS. The proposed design has the ability to encompass all the defined functionalities of the LCAS. Besides this design for a fully qualified LCAS, a design with limited scope is also presented. The limited scope design is derived from the original design and is intended to serve only the use case presented in section 5.4 the two system designs are elaborated through a graphical illustration of their logical views.

7.1 Introduction

System design is the process of defining individual modules, together with the interfaces and data flow between the modules, to achieve a set of desired functionalities. The design needs to satisfy various requirements and constraints that are analyzed in the previous chapters. System design is a creative process and does not follow hard and fast rules, however, there are various guidelines and tools from systems engineering theory to assist the process. One such tool is CAFCR[3], which is an architectural reasoning framework that helps in making the design choices. In this assignment, the systems engineering approach employed to carry out the design task is a combination of the CAFCR architectural reasoning and the design guidelines from the automotive safety standard ISO26262.

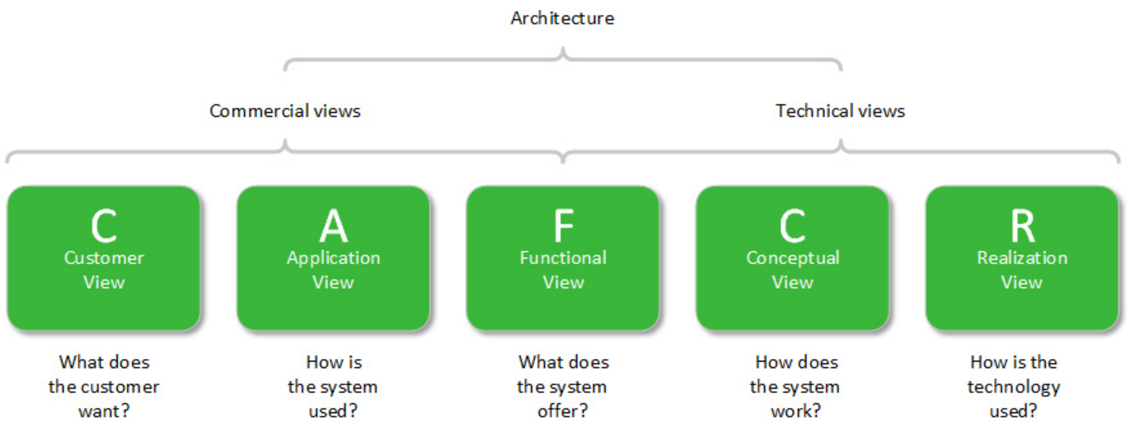


Figure 12: CAFCR, a Multi-View Method for System Architecting [15]

CAFCR provides various views, depicted in Figure 12, in order to evaluate various design choices and to critically analyze a design. The different views enable a designer to consider the system or product from various perspectives. The customer view provides answers to the *'Why'* question from the customer. In this view, the focus is to establish a value, need or usability of the system. Bearing on the LCAS, this need is established in 1stchapter. The customer view also includes defining the stakeholders and their concerns, which were covered in the 2nd chapter. The application view focusses on the interactions of the system with its environment. These interactions are often described by a context diagram of the system. In relation to the LCAS, this diagram is presented in section 3.4 . The functional view, which addresses *'What' question*, is covered by chapters 1, 5 and 6 of the report starting from the functional safety analysis and leading to the function and nonfunctional requirements of the system. Finally, this chapter marks the beginning of the '*How*' part of the report, which is answered by the concept and realization views of the system.

[3] Customer Objectives View, Application View, Functional View, Conceptual View, Realization View

7.2 System Architecture - LCAS

Based on the safety goals and the functional safety concept, a top-level design for the LCAS is proposed. The logical view of the system, with device level responsibilities, is shown in Figure 13. To provide a basic understanding of the system, consider various stages and states that the system assumes during operation. When the system is turned on the first time, it checks if enablement is possible by evaluating the operating conditions. The enabled state is an indication that active steering is available. If enabled, the LCAS provides active steering in only two conditions, either the driver requested the system to take the steering control or the system detected that the car is inadvertently straying of the lane. Once the active steering is commissioned, the system keeps the car centered in the driving lane. The active steering is decommissioned in case an indicator signal is detected or the driver provides manual steering.

Assessing the enablement of the system together with the commissioning of the active steering is handled by a '*supervisory control*' module, which is deployed on a real-time target machine named '*Speed Goat*'. The operational conditions, however, are evaluated in the software application, '*TUeLaneTracker*', running on the vision processor. The main reasons behind this design decision are that evaluation of the operating conditions includes assessing the weather and visibility conditions, which can be easily judged based on vision processing.

Once the system is enabled by the *supervisory control*, the image acquired from High Dynamic Range (HDR) camera is employed to detect and track the lane markings in the scene. The lane boundary information is passed to the '*lateral Kinematics*' module of the software application which estimates the current path along with computation of a target path. Subsequently, the lateral kinematics is verified using redundant sources such as RTK-GNSS, GPS, and IMU data. Although data from such devices cannot be used to reliably track a lane, it can be used to perform a sanity check on the target and the actual path estimated by the camera unit. This sanity check is performed in the verification block. In case the checks are positive the lateral error at a predefined look-ahead distance is calculated and passed to the lateral controller which is responsible for generating the required steering angle. Finally, the override functionality is implemented in the *Move-Box* which is the interface to various in-vehicle sensors and actuators.

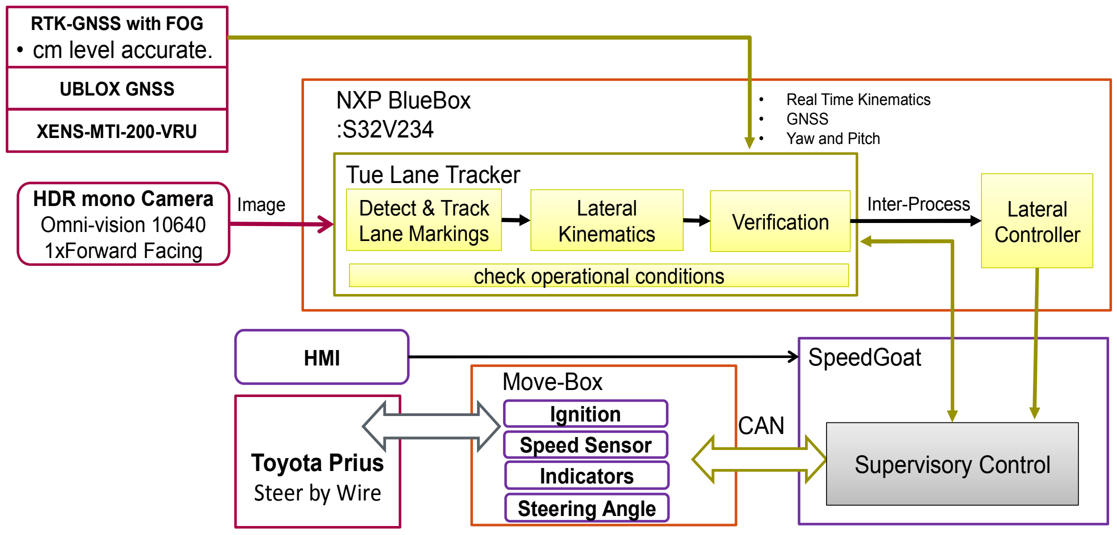


Figure 13: Tope Level System Design LCAS

7.1 System Architecture - Current Use Case

The system architecture presented in this presentation is only a simplification of the design presented in the previous section and is depicted in Figure 14. In the current scope of the system, as soon as the system is turned on, the HDR camera starts feeding images to the software application, '*TUeLaneTracker'*. If the images are properly shown on the HMI, the system is operational and the driver can turn on the lane tracking through the control interface of the software application. If the tracking results are visible on the screen and look promising to the driver, he may commission the active steering. Therefor commissioning of the active steering, in this use case, is a direct responsibility of the driver and so are the repercussions. The driver achieves this by signaling the application, through an application user interface, to send steering commands to the steer by wire system of the Toyota Prius. In case of an emergency, the driver can override the system by applying a torque on the steering wheel.

The physical links and the interfaces between the various components of the system and the vision processor, S32V234, of the BlueBox are depicted in Figure 15. An HDR mono camera, *'Omni-vision 10640*', is connected to the vision processor by means of a coaxial cable. The coaxial cable transmits the serialized stream of the image from the camera module. This means, that on the BlueBox end deserialization is needed, which is achieved through an external deserialization board 'Maxim-9286'attached to the vision processor. The processed frame is shown on the HDMI screen, using a '*Frame Buffer'* device in the vision processor. The Frame Buffer maps the processed pixels, from the software application memory, onto the physical screen. Also, the *lateral controller*, running as a separate software application, takes its input from the vision application, *'TUeLaneTracker'*, using a UDP socket. The link between the '*lateral controller'* and the Move-Box is established through a CAN network protocol. Additionally, an application-user interface is also depicted in a tabular form in Figure 15.

The user interface for the application is kept as simple as possible, for the reason that this interface will be removed altogether when the system will grow to its full maturity. In the context of full-fledged LCAS, the user is not allowed to command directly to the software application for safety reasons. In this case, the communication between the user and the software application is conducted by a supervisory control. In the current scope, however, the supervisor is not designed and implemented. Also, the redundant sources, '*RTK-GNSS'* and '*IMU'*, to verify the current path, as well as the generated target path, are not employed. Another delimitation is that the automatic evaluation of the operating conditions is not covered in this scope. Therefore, in the current setup, the driver himself is responsible for determining the safe conditions to enable and commission the active steering.

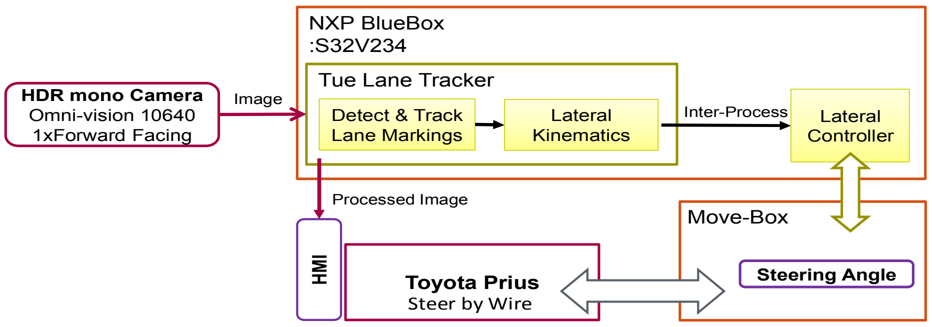
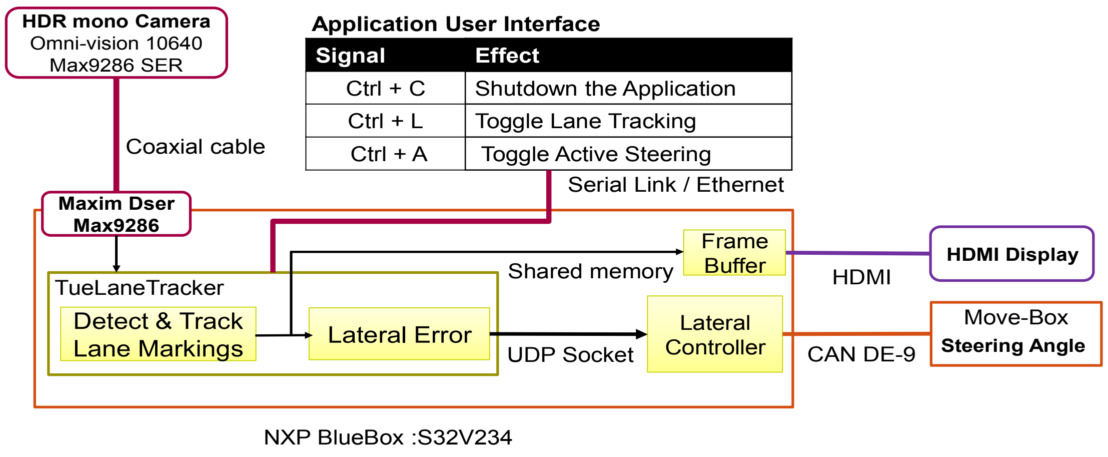


Figure 14: Top Level System Design for Focused the Use Case


Application User Interface

Signal	Effect
Ctrl + C	Shutdown the Application
Ctrl + L	Toggle Lane Tracking
Ctrl + A	Toggle Active Steering

Figure 15: Interfaces of the BlueBox with the Components

8.Lane Tracking Algorithm Design

In the previous chapter a system design is presented, in which the central component of the system is the software application, whose core function is to execute the lane detection and tracking algorithm. This chapter provides a comprehensive understanding of the algorithm by explaining the theory and working principle of the algorithm. A modular design of the algorithm is also presented and explained using a graphical illustration. Moreover, for comparison, a lane detection and tracking algorithm from NXP is also explained in this chapter. At the end of this chapter, some results from the algorithm are presented along with a discussion on the future work.

8.1 Introduction

Reliable lane perception is an essential enabler for advanced driving assistance systems. Visual cues from a forward-facing camera can be used to estimate the lateral position and orientation of the vehicle as well as that of the lane boundaries. Interpreting the visual cues is a perception problem, which enables various advanced applications and systems, including LCAS and self-driving cars. Many researchers have provided extensive contributions to solve this perception problem. However, a reliable localization along with detection and tracking of the lane boundaries remains a big challenge.

On the surface, the problem of lane detection and tracking does not appear particularly challenging. A brief literature study reveals that a simple Hough transform based algorithm can be employed to detect lane boundaries in 90% of the highway cases [16] . This initial impression, however, is quite misleading due to the high-reliability requirement of the automotive domain and variability in the operating conditions. As the algorithm is responsible for actively guiding the car in a lane, there is no room for an error. An incorrect estimation could lead to a fatal accident. The real challenge is not to detect the lane boundaries in a given sequence of images, it is to do so for every image frame acquired on the road within a defined uncertainty tolerance. Assuming a frame rate of 20 frames per second, in a thirty-minute journey, the algorithm has to process around 36,000 images without a single error. Additionally, as the algorithm operates in a highly dynamic environment, the high variability in operating conditions also add to the complexity of the challenge. Roads are, arguably, the most happening and vibrant places, which make reliable tracking of the lane a baffling research challenge. Although, application of the algorithm is restricted to only highways, there are various factors that need explicit attention. These factors include, but are not limited to, road infrastructure, condition and appearance of the lane markings, the lighting conditions, road shadows and various systematic likewise random occlusions caused by the traffic participants. One way to deal with the high variability is to have different algorithms or settings for different situations, however, inferring the situation on the road is a non-trivial task itself. Recently, Convolution Neural Network (CNN) based methods have gained popularity and are very effective in solving such visual perception challenges.

The main reason for the growing popularity of the Convolution Neural Networks (CNN) is their ability to cope with the high variability of the operating environment. CNN is a deep neural network based learning scheme, broadly classified as deep learning, which is inspired from the working of the biological mind. The approach focusses on end-to-end learning where the algorithm directly learns a desired reaction or classification from a set of examples. This is in contrast to end-to-end engineering approaches, where the system is engineered to react to a stimulus based on its knowledge of the world. Although end-to-end learning is quite remarkable in its ability to scale to different scenarios and adapt with dynamic operating conditions, it has its practical limitation in the current state of technology
.

One disadvantage of the CNN based approaches is that a huge amount of labeled data, training examples, is required to reliably train the algorithm, which is computationally an expensive task. Also, as the network directly learns classification or an appropriate response from training data, it is hard to prove that it has only learned the desired and appropriate responses. Furthermore, computation power required for online inference is of the order of magnitudes higher than of the end-to-end engineering or modeling based approaches. These are some of the limitations of the CNN based algorithms as compare to the model-based approaches, nevertheless, the two schemes are not mutually exclusive and various principles from the CNN based methods can be used in model-based approaches as well.

In this assignment, two model-based lane detection were analyzed. The first is a well-established lane tracking algorithm developed at NXP while the second is a state of the art algorithm recently investigated at the University of Technology Eindhoven. This algorithm borrows the concept of hierarchical classification from the deep learning approaches, for example, CNN. However, it is strictly an end to end engineering approach. This state of the art algorithm is selected and considered further for achieving the envisioned LCAS. A brief overview of the NXP algorithm is also presented to highlight the distinguishing characteristics of the selected algorithm.

8.2 NXP Lane Tracking Algorithm

The approach used in this algorithm is similar to one presented in [16], which is a classical approach normally employed for lane detection and tracking using a camera. A graphical representation of the approach is presented in Figure 16, where the blocks represent various steps executed in the algorithm. The algorithm first converts the acquired image to a gray-scale, which is then transformed into a bird-eye view through an inverse perspective transform. The inverse perspective transform removes the perspective effects of the camera and provides a top-down view of the road. A gradient filter is used to detect edges in the frame and the resulting frame is fed to a median filter to remove any noise in the gradient image. These steps are repeated until a predefined number of frames has been processed and buffered, which is then subjected to a temporal filter. The temporal filter is a max-pooling filter that selects the strongest responses from the buffered gradient frames. Subsequently, selection of the lane boundary pixels is carried out through a ray casting method. The lane boundary pixels are then further filtered for outliers and the resulting pixels are employed for line estimation. The line fitting is done through a least square fitting which aims at minimizing the square of the distance between the line and the estimated points. Finally, a Kalman filter is employed to smooth out the lane parameters and remove any measurements errors before the results are displayed on the screen. This classical scheme provides promising results on a straight lane with clear lane boundaries, however, does not handle curved roads. Also, this schemes cannot reliably determine the lane boundaries when the lane lane-marking s are not clearly visible or occluded by shadows on the road.

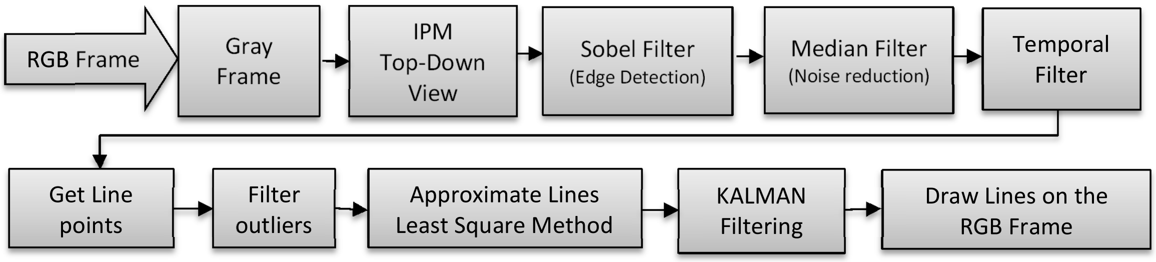

RGB Frame
Gray Frame
IPM Top-Down View
Sobel Filter (Edge Detection)
Median Filter (Noise reduction)
Temporal Filter
Get Line points
Filter outliers
Approximate Lines Least Square Method
KALMAN Filtering
Draw Lines on the RGB Frame

Figure 16: NXP Lane Detection and Tracking Algorithm

8.3 TU/e Lane Tracker

TU/e Lane Tracker algorithm is the lane detection and tracking algorithm that is under research at TU/e, with original ideas and work from Dr. Dubbelman. The functional and nonfunctional optimization of this algorithm forms the core of this design assignment. Therefore, we start with an overview of the working principles of the algorithm followed by an explanation of the design principles and the probabilistic theory employed in the approach. Finally, the actual algorithm design which introduced various improvements is detailed and explained using a graphical illustration. We conclude this section by presenting some of the distinguishing features of the algorithm, which differentiate it from other approaches, and a view on future possibilities.

8.3.1. Working Principle

To understand the working of the algorithm consider Figure 17, which provides a view of the road where the lanes need to be detected. The area of the image between the '*Base Line*' and the line passing through '*Vanishing Pint*' forms the region of interest for lane detection. TU/e algorithm starts with treating every pixel in the region of interest as a possible line that defines one of the lane boundaries. Every pixel in the region of interest is assigned a probability, which evaluate their fitness, based on their location in the image, color as well gradient information, for being a lane boundary. Assuming that each pixel represents a point on a straight line, intersections of these 'supposed' lines the 'Base Line' as well 'Purview Line' in the image are computed. The intersections are computed by exploiting the gradient orientation of the individual pixels.

Once the intersection points are estimated, they are weighted according to the probabilities of the source pixels and a corresponding multimodal histogram is computed at the 'Base Line' and 'Purview Line'. The multimodal histogram at the base, base histogram, represents the density of the pixels voting for a certain bin on the base line to be the lane boundary. Similarly, histogram at the purview, purview histogram, provides that information for the purview line. The base histogram is than matched with a number of precomputed histograms, belief histograms, which represent our prior belief about a lane. The belief histograms are computed for every feasible combination of a left and right boundary points on the base line. A high match between the base histogram and a precomputed belief histogram, high posterior probability, determines the two points on the base line representing left and right lane boundaries. The process is repeated for the purview histogram to calculate similarly the most probable, having highest posterior probability, vanishing point. Therefore, a lane is represented by two points on the 'Base Line' together with a vanishing point as depicted in Figure 17 . These three points in the image constitute the states of a lane model.

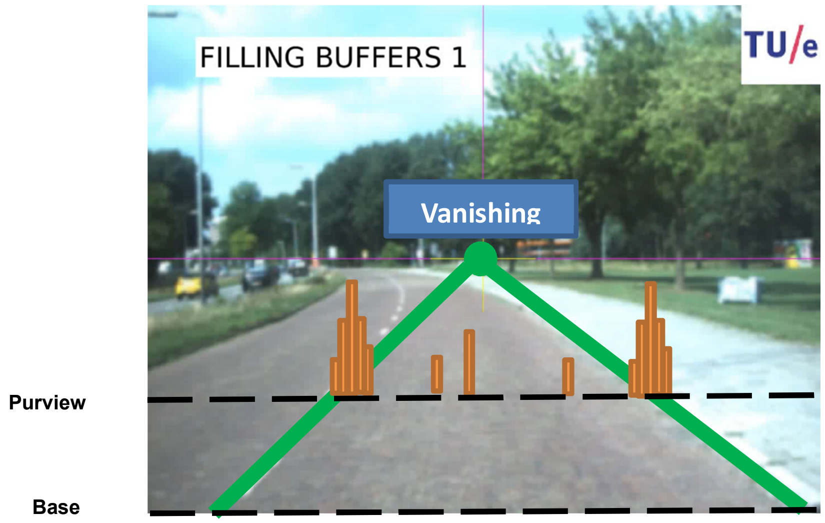


Figure 17: Working Principle of the TU/e Lane Detection and Tracking Algorithm

8.3.2. Design Principles

This algorithm exploits hierarchical classification, similar to deep learning methods, however unlike deep learning, classification at each hierarchical level is engineered instead of being trained through images. This makes the algorithm predictable and easier to verify. The basic principle employed in the design of this algorithm is to keep it completely probabilistic and avoid any hard decisions until the very end. This way the algorithm can avoid making premature decisions on the lane boundaries. For example even if a certain pixel has color that is unlikely for the lane boundary, the pixel information is kept until the final classification. This provides a good chance for the algorithm to detect even a lane which is missing some of the visual features of typical lane. Secondly, instead of computing a single complex rule for determining pixel probabilities various weak rules are combined to build the final probability. This provides a way to independently compute the probabilities from various channels, for example HSV color space, and then combine them in order to improve the functional performance of the algorithm. This also enables conditional inclusion of the rules which means that some probability rules will only be active in certain situations. Furthermore, the classification is carried out in a strict hierarchical order which goes from a pixel level to object level. The main idea behind maintaining the hierarchy is to allow the algorithm to start from a high recall, low precision and move progressively towards a high precision while maintaining the high recall.

In majority of the lane detection and tracking algorithms investigated, for example in section 8.2 as well as in [16] [17], a Bayes filter is employed as final step, to refine the output, in form of a Kalman or practical filter. However in this algorithm, the Bayes filter is the core classifier rather a post-processing step. The particular form of Bayes filter employed here is multimodal histogram filter. The algorithm first obtains a realization from the scene which is subjected to the classifier. The classifier selects a lane model, among various models, which best explains the realization, i.e. which maximizes the likelihood of the observed data. Moreover, as the lane has a well-defined structure an informed prior probability function of multimodal histogram form is employed in the classifier. The prior or belief takes into account the expected width of a lane as well as the standard deviation in lane widths.

Predict and update steps are cascaded to filter out any measurement or observation errors and to increase the robustness of the algorithm. The predict step estimates the prior probabilities of the current state using the belief, the initial prior, and posterior probability from the previous step. Ideally this prior can be further refined by incorporating information from various in-vehicle sensors including the data from the in-car navigation system. In the update step, we use the estimated prior probability to compute the posterior probability of the current state which is equal to the likelihood of the current state of the observed data. This posterior probability is then combined again with the belief in the next predict step to generate a new prior for the next update step. The chain continues in this fashion, cycling between the prior estimation and posterior update as depicted in Figure 18.

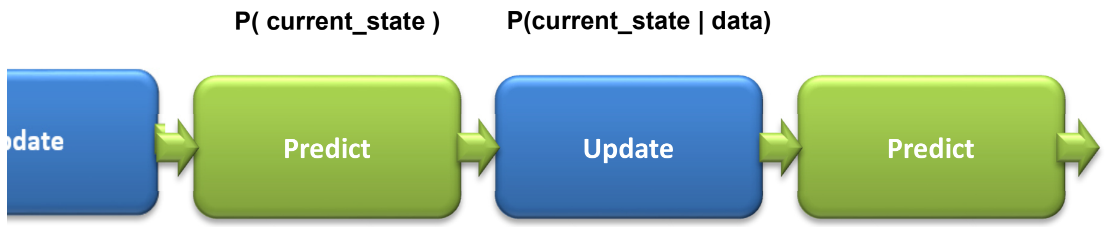


Figure 18: Update and Predict Chain in the Lane Detection and Tracking Algorithm

8.3.3. Proposed Algorithm Architecture

The algorithm itself, as a proof of concept, already existed and has been verified offline on a video sequence obtained during a test drive through the city of Eindhoven. The proof of concept existed as a monolithic code that employed mostly global scope variables. Though the algorithm was operational it was hard to find bottlenecks and optimize further. Therefore, the design of modular architecture was carried out with the goal of providing various functional and non-functional improvements in mind.

The functional improvements mainly came from optimizing the probability maps, tuning mixing weights of various probability channels as well mixing of the belief and posterior probability for calculating the prior probability maps for the next iteration. These optimizations were made possible through a modular restructuring of the algorithm. The modular design allowed easier manipulation of the individual modules and helped in studying the effect of a single module on the complete algorithm. Furthermore, as the algorithm needs to execute on an embedded device in real-time, various improvements from a design point of view were needed.

The APEX accelerators, computation engines of the BlueBox that accelerate vision processing, do not have a floating point unit and therefore computations in decimal points are not possible. It is thus essential that the blocks that are a good candidate for APEX acceleration must be identified and converted into equivalent integer arithmetic beforehand. This is, however, a non-trivial task as it requires range analysis for preventing any overflow and precision errors. To add to the complexity of the task, the analysis must account for all possible settings of the algorithm and not just the current one. For example, the fact that a 24-bit integer could hold certain calculation for a 640x480 image, without any precision issues, does not automatically imply that this also holds for higher resolution images. Similarly changing the resolution of the histogram bins can also drastically change the required bit length. Employing an unnecessarily long bit length could cause performance issues, while an insufficient bit length can cause precision errors. An optimal bit length determination for various variables was carried out using extensive static as well dynamic analysis of the algorithm.

The limitation of the fixed points also implied simplification of various exponential and tangent functions into equivalent simpler arithmetic. For example, the logistic probability density function was approximated by a simpler function, of the form $f(x) = \frac{1}{1+abs(x)}$, that does not require exponential terms. Also, instead of comparing the gradient orientation of the image pixels with a precomputed template of orientations the corresponding tangents were compared. This way we avoided computing the tangent inverse, which is expensive to calculate in a numerical form. The template represents the expected tangents of gradient orientation for all lines passing through the vanishing point. The template is shown in, where each color shade represents an isogon. If a pixel in the image represents a line it should have the same angle as the isogon at that location in the template.

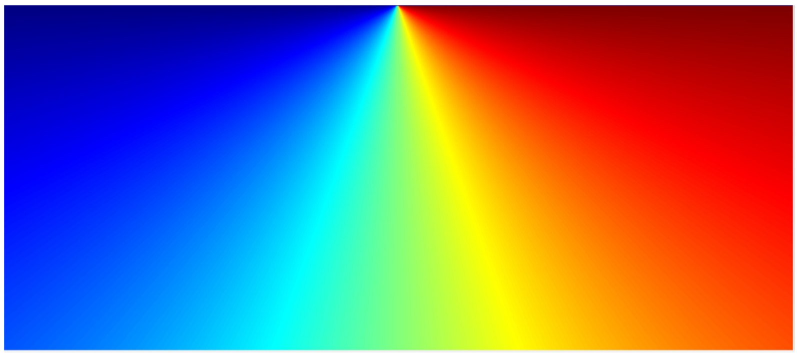


Figure 19: Gradient Orientations for Lines Passing Through the Vanishing Point.

Finally, a major design improvement in the algorithm was to rearrange the computations into a directed acyclic graph as presented in Figure 21 . This not only provided a clear picture of various blocks that needed to be computed along with their inter-dependencies but also the opportunities for non-functional improvements. For example, it provides indications as to which module need to be updated if new probability channels need to be introduced into the algorithm thus improving the extensibility of the algorithm. The directed acyclic graph also helped in identifying various hardware mapping opportunities, for example, the graph was employed to construct to a strategy for mapping the block on various hardware units, ISP, GPU and APEX, on the vision processor.

To describe the acyclic directed graph represented in Figure 21, first, consider the blocks depicted in blue as they represent the input blocks. The blocks with black borders represent the intermediate computations while the blocks in green represent the final output. The inputs blocks include a gray frame, a gradient orientation template, a focus template, a depth template as well as two filters namely lane and vanishing point filter. The filters are a collection of possible states for the lane boundary model. The lane filter is a 1-D filter representing a various possible combination of left and right offsets on the base line presented in Figure 17, while the VP filter is a 2-D filter which is a collection of the possible vanishing points. The depth template weighs the individual pixels based on their location in the image, this is done to remove the perspective effects of the camera. Similarly, focus template takes cares of cropping out an irrelevant scene from the image and is dependent on the movement of the vanishing point. The gray frame is the image obtained from the camera after converting it into the grayscale. Finally, the gradient orientation template is a collection of isogons representing expected tangent of the gradient orientation at a particular location in the image and is depicted in Figure 21.

The pipeline starts by extracting a region of interest from the gray frame and after preprocessing it with a Gaussian blur, the tangent of gradient orientation as well as gradient magnitude for very pixel in the region of interest is computed. Based on the gradient magnitude, the tangent of gradient orientation, as well as gray intensity values probability maps, are computed. In case of gray intensities and gradient magnitude, the probabilities are evaluated by defining an appropriate membership function. The probabilities over gradient orientation channel, however, are computed by comparing the gradient orientation of each pixel to that of the orientation template. The individual probability maps are then combined into a join probability map which is buffered along with the corresponding tangent of gradient orientations. When the buffer is filled with a predefined number of probability frames, temporal max pooling is performed. This selects the most probable pixels and the corresponding tangent of gradient orientations from a sequence of frames in a predefined time window. The probability maps are then further focused, taking into account the motion of vanishing point in the frame. The corresponding tangents of gradient orientations are employed to compute the intersections with the predefined lines in the image.

The computed intersections are weighted with the focus mask and the temporally filtered probability maps. The resultant weighted intersections are used to calculate histograms using pre-specified bins on the base line as well as on the purview line. The bins for the histogram are specified by the lane filter and vanishing point filter. The multimodal histogram represents the number of pixels that voted or supported a certain bin in its claim to be a lane boundary. From the computed histogram, conditional probability for a certain state of lane filter is calculated. The conditional probability is then combined with a transition model, which provides an updated prior, to compute the posterior of the states. The state with maximum posterior probability determines the boundary bins at the base line. The boundary bins are in turn used to compute the conditional probability of all possible vanishing points. This conditional probability is combined with the transition model, as in case of lane filter, to compute the corresponding posterior probability for vanishing points. The vanishing point with maximum posterior probability is selected as the valid vanishing point. The selected vanishing point in combination with the selected boundary bins at the base line forms the resultant lane model.

8.3.4. Distinguishing Characteristics

Although the algorithm provides only a linear approximation of lanes, every pixel takes part in determining the lines resulting in robust approximation. This is unique because, in most lane detection and tracking algorithms, only a subset of pixels which satisfies certain straight line criteria are employed. Thus, the line fitting is largely dependent on the filtering and sampling that precede. However, in case of TU/e algorithm, as no pixel is rejected, the effects of making an error in the preceding phase of probability assignment remains contained and the pixels are still considered, though with lower weights, while determining the lines. Furthermore, the vanishing point is handled explicitly, effects of which are visible while making turns and cornering on the highway, as opposed to NXP algorithm where the vanishing point is assumed to lie in a fixed rectangular area. The effect of having a fixed vanishing point is that as soon as the vanishing point moves out of the specified area none of the lane lines will be detected. Finally, another distinguishing feature of the proposed design includes a scalable and modular architecture that allows inclusions of various probabilistic channels and is easier to extend and scale to different situations.

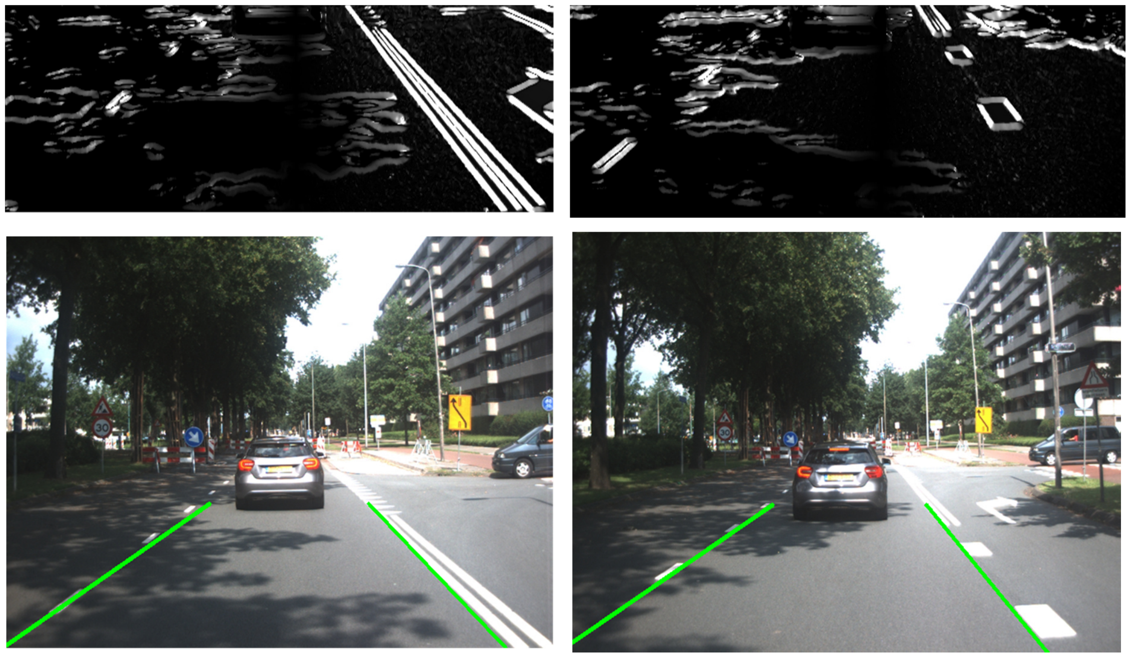


Figure 20: Final Probability Map and Corresponding Results from Lane Detection.

On the top, two final probability maps, collected at different time stamps, are presented while on the bottom corresponding detection and lane tracking results are shown. Note that although the probability maps indicate high probability for shadow pixels, the object level classifier selects the correct lane model.

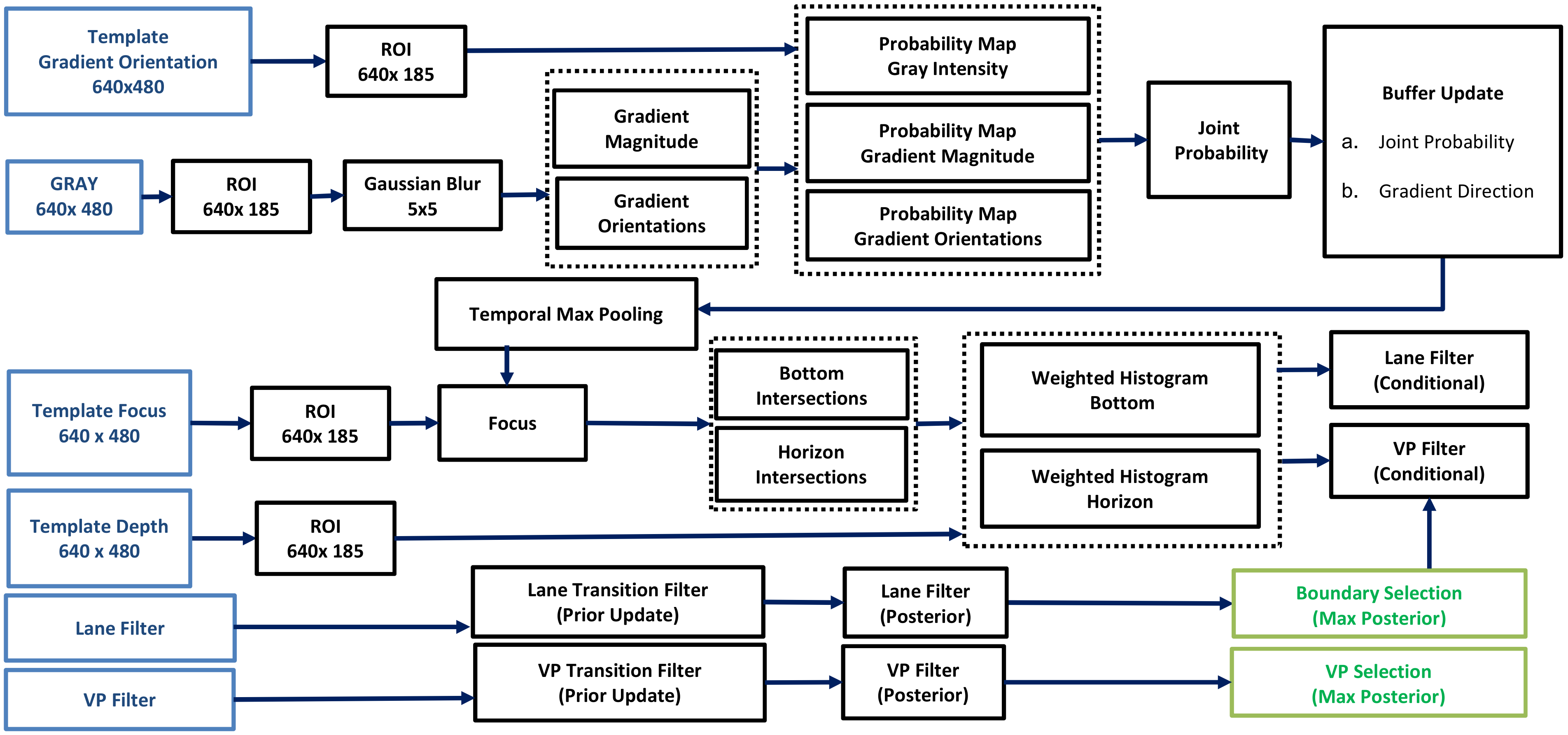


Figure 21: The Complete Design for TU/e Lane Detection and Tracking Algorithm

8.3.5. Discussion and Future Work

Experimental results show that the detection and tracking performance of the *TU/e Lane Tracker* is much better than that of conventional approaches. The preliminary results, in form of a video sequence, can be viewed at the open source repository[4] of this project. The video sequence depicts the results from of the designed algorithm using a sequence of images, collected while driving in the city of Eindhoven. A snapshot of one instance in this sequence is also shown in Figure 22, highlighting the functional performance of the algorithm while driving on sharp curves.

The probabilistic classification, in a strict designed hierarchy, results in a robust and reliable lane track, however, various aspects can still be improved[5]. As the probability maps are computed with our sub-optimal knowledge of how the lane features look like, it is far from optimal. In particular, when there are shadows that give an illusion of a line or other vehicles cutting in the lane and driving very close by, the performance of the algorithm deteriorates. Although most of the shadows are filtered out with the multimodal histogram matching at the object level, the performance is expected to improve by correctly classifying them on the pixel level as low probability. In Figure 20, the final probability maps, computed over various channels and temporally filtered, along with corresponding tracking results, are shown. It is evident that the probability maps are far from optimal and the shadows on the road are being perceived as a likely lane boundary.

As such instances and situations are hard to formalize as rules, one idea is to a use an additional channel that can be trained using the images, for example, by employing Convolution Neural Network (CNN). Here, the CNN can be trained for binary classification of the image regions with a lane or non-lane label. This classification can then be weighted and added to the overall probability. Finally, performing a normalization of the combined probability would decrease the probability of the pixels that were identified as non-lane pixels by the CNN. Additionally, another shortcoming at the moment is a linear approximation of the lanes which limits the look-ahead distance of the algorithm. To overcome this limitation, instead of two fixed lines in the scene, base line and purview line, a ‘far-purview’ line can be added to approximate the lane by a curve. The range of the look-ahead point thus can also be reliably extended, for example by making it speed dependent, enabling a more flexible control scheme. Finally, the single lane track algorithm will be extended to track multiple lanes in future, by including separate filters for the adjacent lanes.

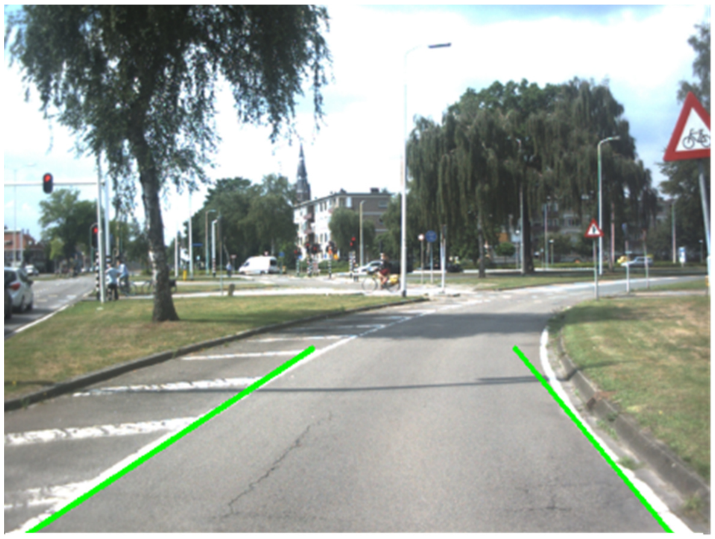


Figure 22: Lane Detection and Tracking While Turning

[4] https://github.com/RameezI/TUeLaneTracker

[5] Good is not good enough, once better is expected –Vin Scully

9.Software Application Design

In the previous chapter, comprehensive details of the lane tracking algorithm along with a design for the algorithm are presented, which predominantly answers the 'what' question, i.e. what is being computed? The questions, which still need to be answered are the '*when*' and '*where*' questions, i.e. when do the algorithm modules start executing and where do these computations happen. To answer these questions, this chapter was drafted. The algorithm design was kept strictly separate from the software design. The main reason for this is to allow algorithm developer to upgrade and refine the algorithm independent of the software design and vice versa. The software design, presented here, provides the possibility to include a different realization of the algorithm at the compile time. This implies, that the software application can be compiled to execute the algorithm on a special hardware, like BlueBox, but at the same time it can also be compiled for a standard platform.

9.1 Introduction

Software application design is the process of defining a structured solution for developing a software program. The design process often starts with agreeing on various quality attributes, for example, performance, reliability, and manageability. It involves making a series of decisions to ensure that the intended tasks will be executed with the desired performance and reliability. Creating a software design is a cumbersome process and requires focused thinking efforts without many tangible gains. Perhaps, this is the reason that the design task is often undervalued or totally ignored in a software development process. Gains of a careful design, however, are unparalleled and the damages that come from ignoring it cannot be undone in later stages.

The current adaptation of agile methods in software development has vastly underestimated these gains of a careful design. Although the underlying idea of agile methods [18] are quite valuable, the practiced approach, which emphasizes more on dogmatic processes and rigid rules, is quite misleading [19]. The current adaptation of agile methods works on the assumption that a coherent design will naturally emerge from an iterative implementation of the user stories or individual functionalities. This assumption that finally, the code will somehow organize itself, in the end, does not normally hold. Also, the law of entropy does not support this assumption. The main disadvantage of this approach is that the process of ensuring the desired quality attributes of the software becomes ambiguous. This ambiguity is much more worrisome in case of safety-critical applications. In safety-critical applications, the focus is not only on the product but also the process. It is not enough to establish that the product works, ensuring that the product will continue to work, as intended, is also essential. If the process is not well laid out and design is not explicitly considered, ensuring various quality attributes of the software becomes cumbersome. Therefore, for a safety-critical software, it is essential to follow an architecture-centric process, for example, the process defined by the V-Model. Perhaps, this is the reason that ISO26262 has made the 'V-Model' based development a requirement for developing safety-critical automotive applications. An illustration of the V-Model is presented in Figure 23

The adopted approach is a mix of the V-Model development and the some very valuable principles from the agile manifesto [20]. The design, realization, and validation processes were executed side by side in an iterative way. After establishing a preliminary version of the design, the focus was shifted to the implementation phase and subsequently to verification of the results, which revealed various design inconsistencies. At this point, the focus was moved back to the second iteration of the design to remove the inconsistencies followed again by a short realization and validation cycle and so forth. The process was kept agile to integrate changes or requirement updates at any stage of the design and development, however, the changes were also reflected in a centric design to remove any ambiguity. Thus the focus was not only on a working software that fulfills the requirements but also on the traceability at the design level.

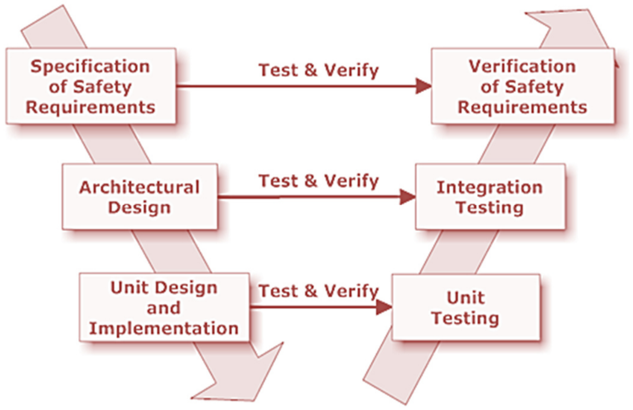

Specification of Safety Requirements
Test & Verify
Verification of Safety Requirements
Architectural Design
Test & Verify
Integration Testing
Unit Design and Implementation
Test & Verify
Unit Testing

Figure 23: The V-Model for Software Development

9.2 Design Principles

Before explaining the design itself, it is important to understand that the key principles of the design. This section provides details on the principles employed to achieve the targeted quality attributes of the software application described in section 6.2.3. The three principles are:

I. **Separation of Concerns** advocates dividing the application into distinct features with very low overlap between them. In the context of the current design, this implies that there must be a clear distinction between the algorithm and the control flow. Also, the interactions between the software and algorithm must be abstracted in ways so that there is no considerable overlap between the components.

II. **Single Responsibility Principle** is a well-established principle for object-oriented design which states that each class should be responsible for a single functionality of the software. A class must not be responsible for multiple functionalities as it makes the software harder to upgrade and manage.

III. **The principle of Least Knowledge** states that the each software component should have limited knowledge about the internal working of the other components. Furthermore, the components may only talk withclosely related components but not to the strangers. In case of object-oriented programming, this means that if class A is dependent on class B which in turn is dependent on class C, class A must not talk to class C directly. The main advantage of using this principle in software design is that the software will easier to maintain and adapt.

Besides these principles, various recommendation and guidelines from ISO26262 were considered while designing the software application. For example, it is highly recommended for ASIL '*A'* or higher level safety-critical applications to exhibit a hierarchical structure and an unambiguous graphical representation. Similarly, a restricted coupling between components and high cohesions within a component is recommended for ASIL level 'A' while highly recommended for ASIL level 'B' and above. Although within the time constraint and set priorities it was difficult to account for every single recommendation, a list of design recommendations observed in this assignment is presented in Table 1.

ISO-26262 Methods	ASIL-A	ASIL-B	ASIL-C	ASIL-D
Hierarchical structure of software components	HR	HR	HR	HR
Use of unambiguous graphical representation	HR	HR	HR	HR
High cohesion within each software component	R	HR	HR	HR
Restricted coupling between software components	R	HR	HR	HR
Range checks of input and output Data	HR	HR	HR	HR
Control flow analysis	R	R	HR	HR
Restricted size of interfaces	R	R	R	R
Restricted size of components	HR	HR	HR	HR
Appropriate scheduling properties	HR	HR	HR	HR

Table 1: ISO26262 Recommendations for Software Design

HR = Highly Recommended
R = Recommended

9.3 Logical View

A logical view describes the structural and the hierarchical aspect of the software and includes all major components along with their relationships. The logical view is also responsible for depicting the control flow of the software application. In Figure 24, a high-level view of the class domain objects is presented which depicts the aggregation of the state machine. A more detailed view is presented in Figure 25, which depicts the individual states along with their member variables and member functions. The diagrams are kept simple, for better comprehension, and hides various low-level details.

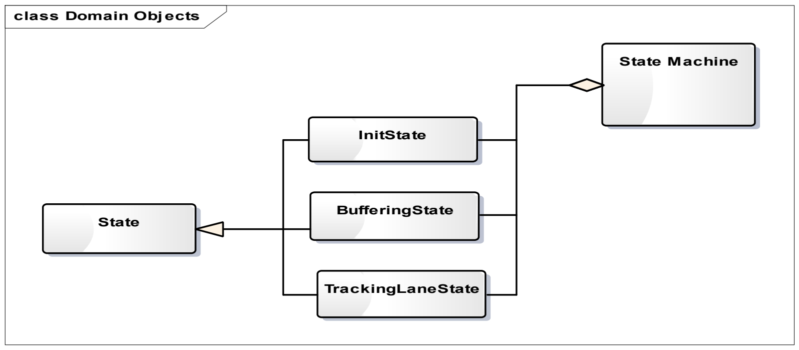
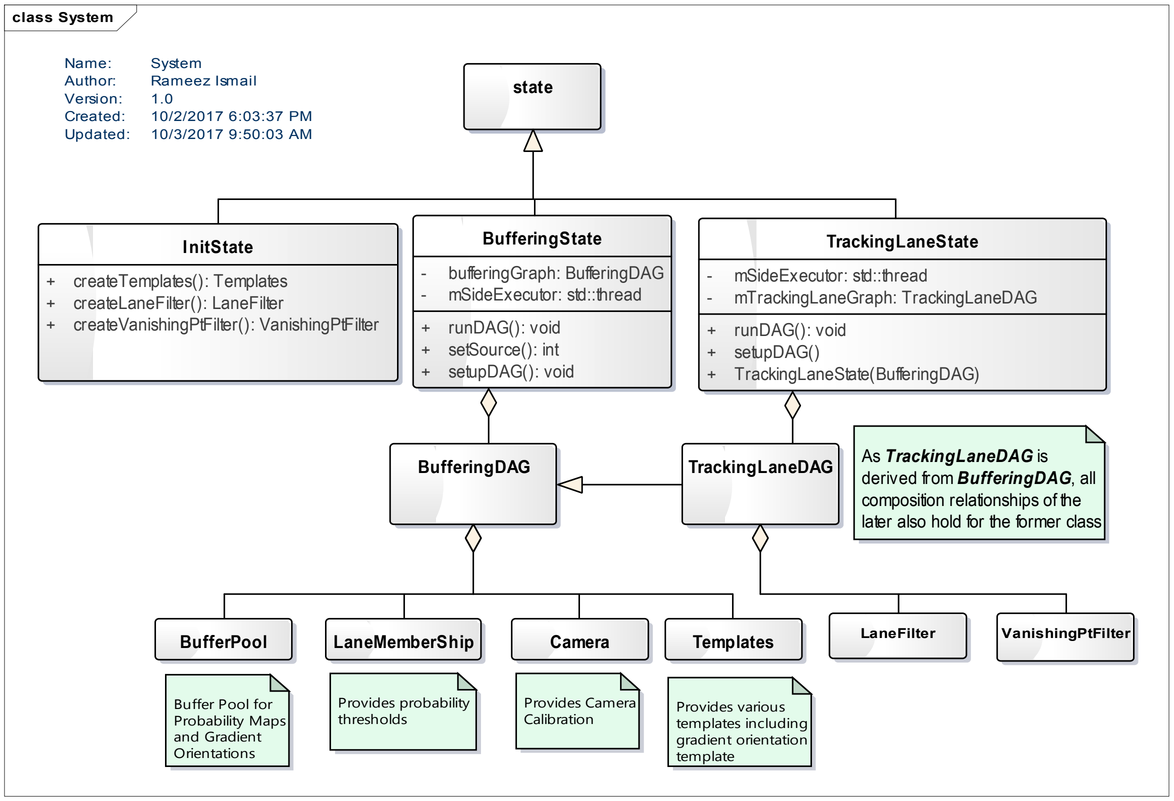


Figure 24: Domain Class Objects of the Software Application

In the software design, a clear boundary between the vision algorithm and the control flow of the software is drawn. This allows a software designer to extend the existing software design without any knowledge of the algorithm. Similarly, an algorithm developer does not need to know how the control flow of the software is managed. To achieve this, the algorithm is completely separated into two classes, the '*BufferingDAG'* and the '*TrackingLaneDAG'* depicted in Figure 25. This means that if optimization or upgrade of the lane tracking algorithm is needed, only these two classes are updated. Any changes in the algorithm do not affect the control flow of the software and vice versa. The software is thus agnostic of the underlying algorithm, this feature is employed to achieve two distinct implementations of the algorithm. The desired implementation is selected at the compile time and is executed by the software without any discrimination.



class System
Name: System
Author: Rameez Ismail
Version: 1.0
Created: 10/2/2017 6:03:37 PM
Updated: 10/3/2017 9:50:03 AM
state
InitState
+ createTemplates(): Templates
+ createLaneFilter(): LaneFilter
+ createVanishingPtFilter(): VanishingPtFilter
BufferingState
- bufferingGraph: BufferingDAG
- mSideExecutor: std::thread
+ runDAG(): void
+ setSource(): int
+ setupDAG(): void
TrackingLaneState
- mSideExecutor: std::thread
- mTrackingLaneGraph: TrackingLaneDAG
+ runDAG(): void
+ setupDAG()
+ TrackingLaneState(BufferingDAG)
BufferingDAG
TrackingLaneDAG
As TrackingLaneDAG is derived from BufferingDAG, all composition relationships of the later also hold for the former class
BufferPool
LaneMemberShip
Camera
Templates
LaneFilter
VanishingPtFilter
Buffer Pool for Probability Maps and Gradient Orientations
Provides probability thresholds
Provides Camera Calibration
Provides various templates including gradient orientation template

Figure 25: A Class Model of the Software Application.

The 'BufferingDAG' and 'TrackingLaneDAG' are designed to contain a sequence of blocks which records their execution time. Each block has an identifier, a 'begin' method and an 'end' method. On execution, each block records the minimum, maximum and average time spent by the processors to process the instructions within the 'begin' and 'end' methods of the block. The principle is that the instructions from the algorithm that are highly coherent, are grouped and placed in a block. A sequence of such blocks hence encompasses the complete algorithm. The sequence is strictly progressive and never loops back, hence the suffix DAG (Directed Acyclic Graph) for the classes. In this setup, the timings of each block can be precisely determined to provide the opportunity to quickly determine the bottlenecks in the algorithm performance and optimize accordingly. For simplification the internal details of DAGs are not depicted in Figure 24, please refer to section 9.4 for a detailed view.

The classes *'BufferingState'* and *'TrackingLaneState',* in Figure 25, provides a higher level of abstraction for communication between the control flow and the algorithm, The main responsibilities of these classes are to set up the corresponding DAG and to execute the DAG on two parallel threads. The synchronization between the two threads is explained in section 9.4 The two parallel threads are commissioned in the '*Sunday'* member function of the '*BufferingState'* and *'TrackingLaneState',* to provide DAG an alternate path along with the main execution path. The alternate path for the algorithm is provided by the 'mSideExecutor' member variable in the '*BufferingState'* and *'TrackingLaneState'* class. The synchronization between the two thread is explained in section 9.4

The 'BufferingState' also provides a method to select and configure the input source for the algorithm, for example, a specific camera or defining the input as an RTSP video stream. This selection and configuration are performed by the 'setSource' member function of the 'BufferingState' class. Various dependencies, which are needed by the algorithm are also presented, in Figure 25, these objects are either created in the DAG classes itself or supplied by the 'setupDAG' member function of the responsible state class. Since a basic view of the hierarchy and structure of the classes is now established, the control flow of the application is explained. The control flow answers the '*When'* question of the software application, i.e. when does a certain state execute and what will be the sequence of the execution. The control flow is depicted in the form of a state machine diagram in Figure 26.

The state machine starts from the booting state and generates various dependencies for the following states. If all the dependencies are successfully generated a '*Done'* synchronization label is generated leading to the buffering state. In the *'Buffering'* state only one part of the algorithm, which grabs the frame and computes corresponding probability maps, is executed. The setups for the graph, for example setting up the source, is performed only once when the control first enters the Buffering state. In the subsequent evaluations of the state, only buffering of the probability maps is performed. Once a specified number of probability frames have been buffered, the control is transferred to the '*Tracking Lane'* state by generating a '*Done'* signal. As in case of *Buffering* State, the setups and initialization are done only once. The tracking state is responsible for extracting the lane from the image and setting the lateral error for the controller. If an error is detected or user sends a quit signal, at any state, the state machine goes into 'Shutting Down' state and quit the software application.

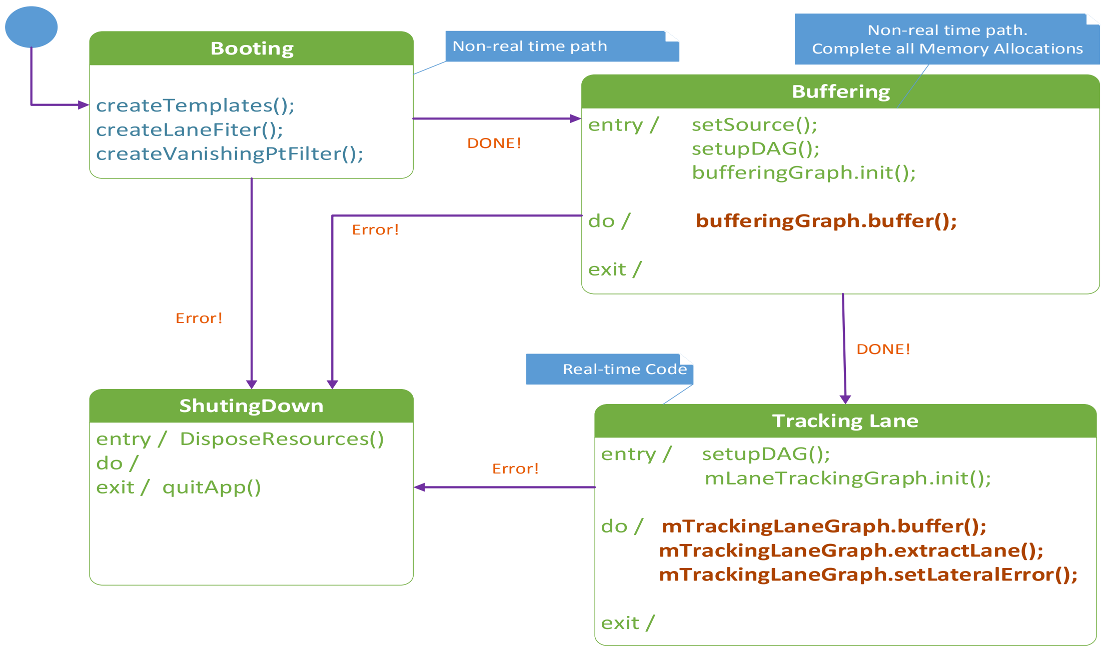


Figure 26: State Machine Model of the Software Application

9.4 *Process View*

The process view of the software application is a dynamic view which maps different processes to threads and explains the underlying concurrency and synchronization mechanisms. As stated, the main dynamic part of the application is the lane tracking algorithm which is distributed into two DAG classes. As the 'TrackingLaneDAG' class is the derived class of 'BufferingDAG' class, it is practical to explain only the concurrency and synchronization details of the extended class. To this purpose, a brief overview of various computations in the 'TrackingLaneDAG' are presented in
Figure 27. It must be taken into account that only the high-level blocks are shown here, for a complete block diagram of the algorithm please refer to section 8.3.3.

As shown, the computation blocks are arranged in three stripes of two different colors. The first two stripes, in yellow, represent the main thread while the third stripe, with black borders, represents the worker thread which executes auxiliary tasks of the algorithm. The two threads provide independent paths for executing the functions of lane tracking algorithm, however, synchronization is required to ensure the instructions are executed in the desired order. This synchronization is provided by mean of standard locks and condition variables.

The worker thread is divided into three different modes; A, B and C. In the presented illustration, after RGB to the gray conversion of the image the main thread signals the side thread that it is clear to execute the mode A which is synchronized just before executing the '*Temporal Pooling*' blocks. This means that the main thread has to wait if the side thread is not finished before the synchronization point. However, the labels are planned and placed such that there is no noticeable wait time involved. On the same lines, the Mode-B and Mode-C are started and synchronized. Mode-C is acute as it starts preparing the probability buffers for the next iteration while in the middle of the current iteration. A detailed description and illustration is provided in
Figure 27.

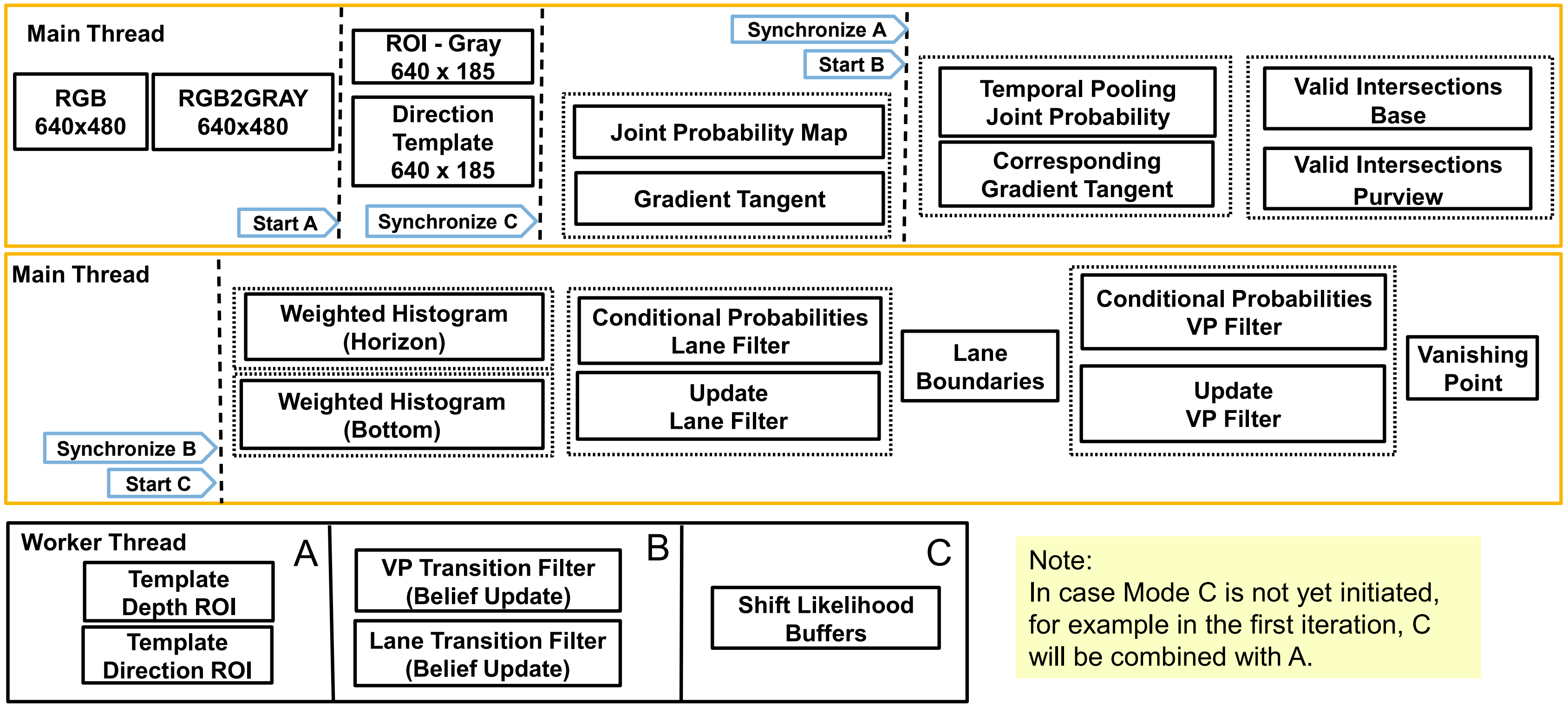

Main Thread
RGB 640x480
RGB2GRAY 640x480
Start A
ROI - Gray 640 x 185
Direction Template 640 x 185
Synchronize C
Synchronize A
Start B
Joint Probability Map
Gradient Tangent
Temporal Pooling Joint Probability
Corresponding Gradient Tangent
Valid Intersections Base
Valid Intersections Purview
Main Thread
Synchronize B
Start C
Weighted Histogram (Horizon)
Weighted Histogram (Bottom)
Conditional Probabilities Lane Filter
Update Lane Filter
Lane Boundaries
Conditional Probabilities VP Filter
Update VP Filter
Vanishing Point
Worker Thread
A
Template Depth ROI
Template Direction ROI
B
VP Transition Filter (Belief Update)
Lane Transition Filter (Belief Update)
C
Shift Likelihood Buffers
Note:
In case Mode C is not yet initiated, for example in the first iteration, C will be combined with A.

Figure 27: Process Diagram for the Software Application

After acquiring a gray scale image the main thread makes a non-blocking call to the worker thread, and the mode ‘A’ is set to execute. The main thread continues with the calculations of the ROI for the gray frame and that of direction template. At this point, the main thread waits to synchronize with the mode ‘C’. When it is confirmed that mode ‘C’ has already finished, the main thread continues evaluating the probability maps. Subsequently, the mode ‘B’ is set to start on the worker thread, while the main thread continues until base and purview intersections are calculated. Afterwards it waits for the mode ‘B’ to synchronize and the mode ‘C’ is initiated immediately. The mode ‘C’ is synchronized in the next iteration.

10. System Realization and Verification

In the previous chapters, a system design, as well as a design of the software application and algorithm, are presented. Although some implementation issues were implicitly discussed, the major focus was the design. Previous chapters have mainly answered the so-called 'WH' questions, i.e. *'Why', 'What', 'When' and 'Where'*. The goal in this chapter is to elaborate various implementation details of the LCAS and focus more on the *'How'* part. In particular, this chapter is concerned with the tools, technologies, software libraries and framework employed to realize the system. Another goal of this chapter is to help in establishing an effective workflow to realize an advanced vision-based system on the BlueBox. Furthermore, various enablers of the BlueBox, as well as platform specific details, are also highlighted in this chapter. This chapter is concluded by detailing the verification strategy employed in the project.

10.1 Introduction

The realization phase involves putting the project and the design plan into action while the verification is the process of confirming and consolidating the correctness of a certain realization. This may give an understanding that the realization and validation phase are executed only after the design is finished, however, in contrast, the design, realization, and verification often go together. At least, in high-tech and innovative projects, it is one of the most agreed ways of working. In the context of self-driving cars, as one needs to work with various novel technologies and uncertainties, a quick cycling between the designing, implementation and verification phases could lead to better productivity. This quick transition, however, is only possible if there exist an effective workflow and comprehensive understanding of the underlying technologies. One major achievement of this project is to define such a workflow and gain an acute understanding of the technologies involved, the details of which are summarized in following sections.

10.2 Workflow

In Figure 28, a high level illustration of the workflow is presented. At NXP, the Yocto project is being used to build a complete Board Support Package (BSP) for the S32V234 target platform. Yocto is an open source project that can be used to customize and package a Linux distribution in form of a deployable image that can run on the target platform. The Yocto setup can also generate a Software Development Kit (SDK) which provides a toolchain for cross development of libraries and applications for the target. This is beneficial when an application has certain dependencies, which are not available on the target platform. In this case, it is quite practical to cross-develop just the dependencies and deploy it in the target rather building and deploying the complete image. Also, the SDK provides an easy way to build and supply additional dependencies for building applications using the vision SDK.

The vision SDK provides various kernel modules and user-space libraries to communicate with the special purpose hardware units, for example, APEX and ISP units, in the platform. The acceleration in the vision applications is thus enabled by the vision SDK. An alternate approach is to use the graphical tool, S32V Design Studio that provides a graphical interface for developing APEX and ISP accelerated applications. In this assignment, however, the textual approach was employed as the vision SDK is slightly more mature in comparison with the design studio. Furthermore, the upcoming Robot Operating System (ROS2.0) is cross-compiled and deployed on the target to allow reliable communication between different systems and applications. However, due to some open issues, the middleware is not used in the current assignment. The planned role of the middleware is to quickly build, test and debug distributed systems which is much needed in the context of self-driving cars. Moreover, Simulink blocks for the CAN interface of S32V234 are available, which allows building and deploying a control application, using embedded coder, on the target. The control applications, therefore, will be deployed, tested and verified using Simulink environment.

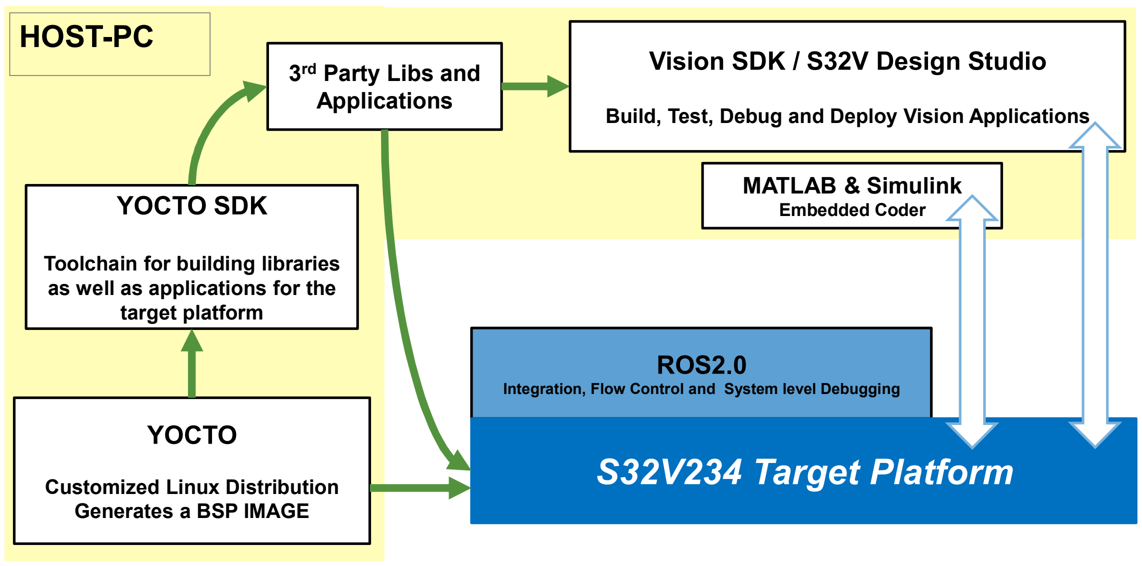

HOST-PC
3rd Party Libs and Applications
Vision SDK / S32V Design Studio
Build, Test, Debug and Deploy Vision Applications
MATLAB & Simulink
Embedded Coder
YOCTO SDK
Toolchain for building libraries as well as applications for the target platform
ROS2.0
Integration, Flow Control and System level Debugging
YOCTO
Customized Linux Distribution Generates a BSP IMAGE
S32V234 Target Platform

Figure 28: The Workflow Developed for Implementing Vision Based ADAS

10.3 Camera Interface

NXP BlueBox provides two options to interface with a remote camera. The first involves streaming the media over the Gigabit Ethernet, while the second approach is to link the camera to a MIPI-CSI-2 interface using dedicated serializers and a deserializer. The two schemes are depicted in Figure 30 and Figure 29 respectively. In case of MIPI-CSI-2 interface, the raw data from the camera sensor is serialized, through a Maxim serializer, and is transmitted via a coaxial cable with an effective data rate of up to 1.5Gbps. On the other end, a dedicated deserializer is used, which can deserialize up to four camera streams. The raw data is then processed by the ISP unit of the vision processor to obtain a final image in the desired format. This remote link with the camera is sometimes also referred to as Gigabit Multimedia Serial Link (GMSL). Alternatively, a Gigabit Ethernet link can be used to transmit a compressed stream of the camera feed. The stream is decoded by the ISP using dedicated hardware decoders, to provide the final Image. The effective data-rate, in this case, is around 100Mbps. The reason that GMSL camera is employed in the design of the LCAS has to do with the fact that it offers an automotive grade reliability with exceptionally low latency.

Although in the proposed design, a GMSL HDR, camera is employed to capture images, from the camera, the required ISP graph is not available. An ISP graph is a camera specific program, which processes the sensor data from the camera to transform it into pixels. Therefore without ISP graph, this link with the camera cannot be realized. The issue was identified in the initial phase of the project and Han Raaijmakers, who is one of the supervisors from NXP, is busy with the development of the required ISP graph for the camera. Some initial results have been achieved so far, but the stability and quality of image acquisition were still need improvements. Developing an ISP programs requires strenuous efforts as the program need to be executed on a highly specialized hardware, for further details on the ISP refer to Appendix E.

In the scope of the current project, we acquired the images, over Ethernet, using a Real Time Streaming Protocol (RTSP), from a stereo camera installed in the Toyota Prius. This is, however, only a temporary workaround, as RTSP stream does not provide the required automotive grade reliability and performance. Once the GMSL camera is stabilized and the corresponding ISP graph is developed, the original design is to be restored.

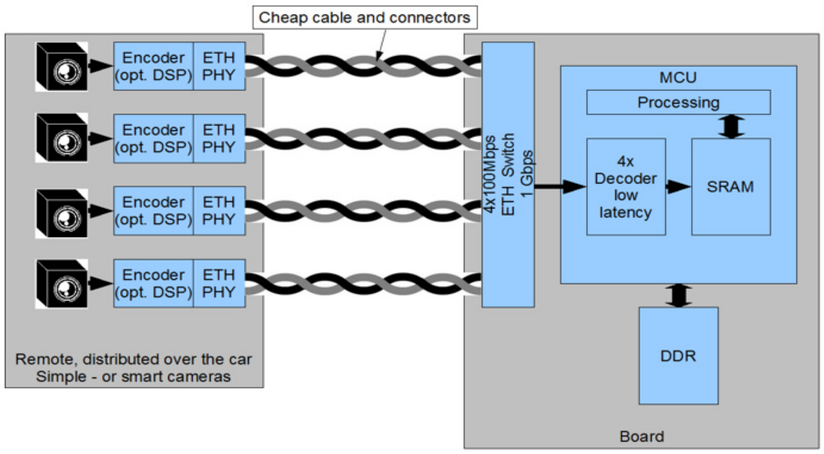


Figure 30 Camera Interface Using Gigabit Ethernet [26]

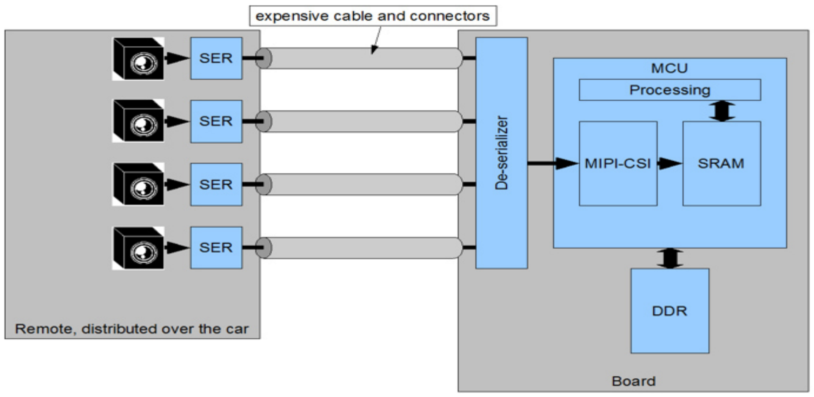

expensive cable and connectors
SER
SER
SER
SER
Remote, distributed over the car
De-serializer
MCU
Processing
MIPI-CSI
SRAM
DDR
Board

Figure 29 Remote Camera Interface Using MIPI-CSI-2 Standard [26]

10.4 APEX Acceleration

APEX cores are the main accelerators of the vision pipeline in the BlueBox and are characterized by their highly energy efficient parallel processing, which is a big contrast to power hungry GPU devices. Moreover, the cores also exhibit phenomenal compute performance. The reason for such an optimized performance is an explicit design, which is conceptualized especially for computer vision applications. In computer vision applications, the processor needs to process hundreds of thousands of pixels, however, often the independent computations are performed only on spatially localized pixels. This means that the complete image can be divided into blocks of pixels and all blocks can be processed in parallel using multiple computing units. The results from different computing units can be combined afterward to produce the desired output. This principle is exploited in the design of APEX cores and is illustrated in
Figure 31.

The APEX core is a massively parallel processor, with SIMD architecture, composed of 64 independent computing units that can process an image block independently. The vision pipeline in the BlueBox has two such APEX cores. Furthermore, a high bandwidth FDMA is employed to transfer large image blocks from the arm cores to the APEX cores. To better understand the working of an APEX core two important concepts, *'Tiling'* and *'Vectorization'* are described in the context of APEX processing. Also, APEX Core Framework (ACF), which provides a programming model for the APEX cores, is also detailed in this section.

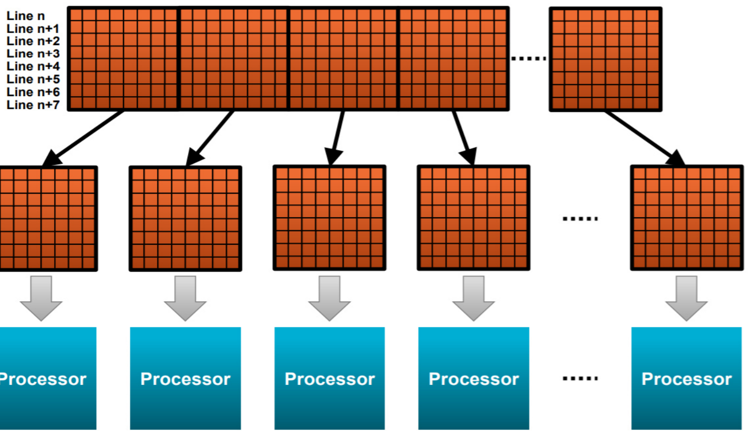


Figure 31: Principle of the APEX Acceleration

10.4.1. Tiling

Tiling is the process of dividing the data into tiles, this division is needed to optimize the data bandwidth required for moving the data in and out of the local memory of the APEX cores. Every APEX compute unit has a limited local memory, which cannot hold the data if a high-resolution image is divided to into only 64 parts. Therefore, the image is first tiled as shown in Figure 32, and each tile is then divided into 64 blocks to compute them in parallel. To maximize the performance the concept of pipelining are deployed, which means that APEX core starts loading the next tile into the local memory while it is still computing the current tile, similarly, other stages are also executed in a pipeline. The tile height normally ranges from one to few hundred rows in the image.

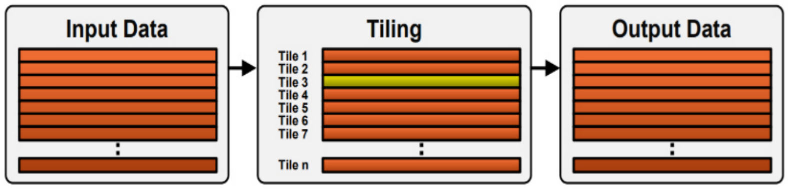


Figure 32: Tiling of the Image Data by the APEX Core [21]

10.4.2. Vectorization

The process of vectorization take a single tile as an input and divides it into 2D chunks as depicted in Figure 33. Each chunk is assigned to an independent compute unit and is executed in parallel. As Apex is a SIMD architecture, every compute unit executes the same instruction and exactly at the same time. This means by the time APEX finishes executing instructions on one chunk, all computing units would have their results computed. The results are fused and moved out of the local memory, to continue with the next tile.

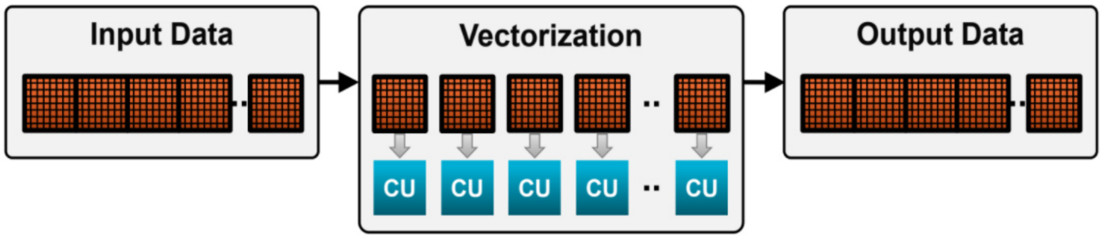


Figure 33 Vectorization of a Tile in the APEX Core [21]

10.4.3. APEX Core Framework

APEX Core Framework (ACF), is an abstraction layer for the APEX cores which provides a programming interface for the developers. It enables the developers to implement and execute the user code on the hardware without having to deal with various hardware-level details, for example, data allocation and movements are handled automatically by the framework. ACF creates a processing pipeline for the user application, on the APEX cores, and manages the following three activities:

1. Transfer of the image data from external host memory to APU memory.
2. Processing the input data by tiling, vectoring and finally integrating the individual results from the computing units.
3. Transfer the output data from APU memory back to host memory.

To achieve this a well-defined topology is employed by the framework, which consists of 'Kernels', *'Graphs' and 'Process'*. A *'Kernel'* is a well-defined unit of processing which is executed by the computing units in the APEX platform. The framework regards the kernel as a black box and is only concerned about the interfaces of the kernel. Therefore, by defining the metadata and the interface of the kernel, in accordance with framework guideline, any desired function can be computed on the APEX framework exploiting full capabilities of the parallel hardware. An example of APEX kernel is presented in Figure 34.

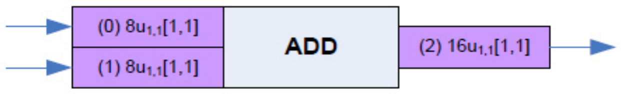


Figure 34: Example of an APEX Kernel

A Kernel, alone, is normally not enough to completely define a complex piece of an algorithm that needs to be executed on the APEX cores. An ACF '*Graph'* is therefore used to combine a various kernel in form of a directed acyclic graph. The information related to a graph is only a concerned with the interconnections of various kernels and the connections with input and output. An example graph is shown in shown in Figure 35.

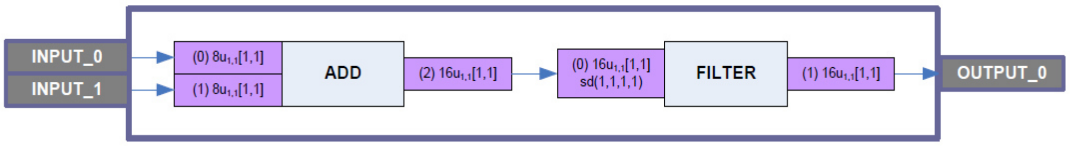


Figure 35: Example of an APEX Graph

Finally, APEX *'Process'* is ready to run a form of the graph that maps the graph to a certain APEX core. This also provides run-time handles for the ARM application, so that the designed graph can be loaded, configured and executed on the APEX cores.

10.5 Software Application Realization

Having explained various technologies, tools and frameworks, the implementation details of the software application are detailed in this section. As explained in chapter 9 , the software design can support various implementations of the algorithm. This design feature is employed to achieve two independent implementations of the lane detection and tracking application. One implementation is optimized for APEX based platforms, for example, BlueBox, while the other can run on any x86 or arm architectures. The implementation which does not depend on any special hardware unit will be referred to as a 'generic' implementation for the lack of better term.

The two implementation are an exact replica of each other in term of the functionalities, however, the performance and safety gains provided by S32V234 version are much needed in any automotive applications. The generic implementation is dependent only on OpenCV library, a collection of open source and well-tested computer vision functions, while the S32V234 specific implementation use, in addition to OpenCV, an accelerated function that is executed on the APEX cores. The main reasons for providing two distinct implementations of the system, range from establishing a comparison to open availability of the algorithm.

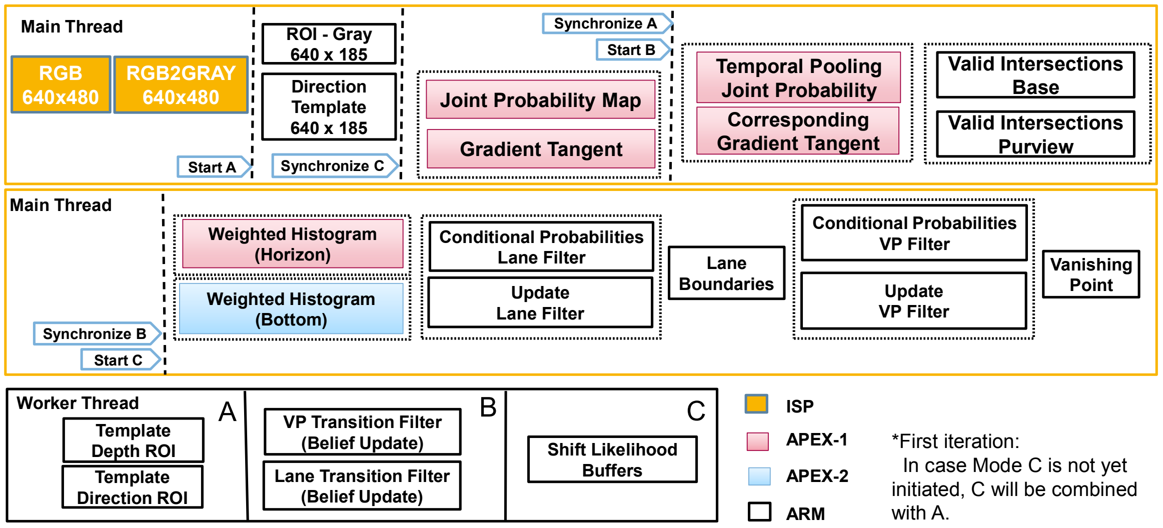
A graphical illustration of the S32V234 specific software application is provided in Figure 36, which provides the details of the mapping, i.e. on which hardware unit a certain block of the algorithm will be computed. In addition to the arm cores, it employs an Image Signal Processor (ISP) and two independent APEX cores. In the beginning, some of the algorithm blocks were also mapped to the GPU however, the performance of the onboard GPU was not very encouraging. Therefore, GPU was removed from the mapping concept and does not take part in general purpose computing. It must be noted here that the process flow is same as explained in 9.4 , the only difference is that instead of computing the colored block on the native arm cores, they are offloaded to the respective accelerators via a Fast Direct Memory Access (FDMA) module. Once the block has been computed the results are committed back to DRAM, mapped by the software application, running on the arm cores, through FDMA.

Besides providing two different implementations of the algorithm, the software implementation takes into account various other quality aspects of the software. For example, some of the recommendations from the ISO26262 that were followed during the implementation of the software application are listed in the table below.

ISO-26262 Methods	ASIL-A	ASIL-B	ASIL-C	ASL-D
One entry and one exit point in subprogram and functions	HR	HR	HR	HR
Avoid global variables	R	HR	HR	HR
Limited use of pointers	HR	HR	HR	HR
No Recursions	R	R	HR	HR
No implicit type conversion	R	R	HR	HR

Table 2: ISO262622 Recommendations for Software Implementation

HR = Highly Recommended
R = Recommended



Main Thread
RGB 640x480
RGB2GRAY 640x480
Start A
ROI - Gray 640 x 185
Direction Template 640 x 185
Synchronize C
Synchronize A
Start B
Joint Probability Map
Gradient Tangent
Temporal Pooling Joint Probability
Corresponding Gradient Tangent
Valid Intersections Base
Valid Intersections Purview
Main Thread
Synchronize B
Start C
Weighted Histogram (Horizon)
Weighted Histogram (Bottom)
Conditional Probabilities Lane Filter
Update Lane Filter
Lane Boundaries
Conditional Probabilities VP Filter
Update VP Filter
Vanishing Point
Worker Thread
A
Template Depth ROI
Template Direction ROI
B
VP Transition Filter (Belief Update)
Lane Transition Filter (Belief Update)
C
Shift Likelihood Buffers
ISP
APEX-1
APEX-2
ARM
*First iteration: In case Mode C is not yet initiated, C will be combined with A.

Figure 36: Mapping of Algorithm on ISP, APEX and ARM Cores

10.6 *Verification and Validation*

For verification of the lane tracking algorithm, unit testing verification was employed to independently test various units in the algorithm for correctness. In the unit tests, the results from a certain unit of the algorithm were compared with ground truth results at matrix level. The ground truth values were obtained through a MATLAB implementation of the algorithm, which was already validated over thousands of images. Once the units were verified, a frame level verification was performed. In the frame level verification, the designed algorithm, as well as the reference MATLAB algorithm, were given the same frame and the corresponding results were checked in an automated testing environment. After frame level testing, the newly designed algorithm was validated on a data set of around five thousand images.

A similar process of verification and validation was performed when accelerators were employed to generate yet another implementation of the algorithm. Moreover, validation of the latency of the algorithm was also performed at a unit level along with frame level. The execution time of the individual units of the algorithm was recorded as well as that of the complete frame. In the current latency verification set up, the units and frame level latency is recorded for every image processed. Thus a statistics of the experienced latency, for individual blocks in addition to the complete algorithm is obtained for verification of the results.

On software application level various tests were performed to ensure the correctness of the software as well as to ensure the quality attributes. For example, one quality attribute according to design specifications is modularity. This design aspect of the architecture has been verified by allowing software to choose one algorithm implementation out of the two at the compile time. Moreover, the dynamic characteristics, such as memory allocations, were verified by using an open source tool 'Valgrind' which checks for any event of memory leak. A formal verification of the lateral controller and hence the complete system is beyond the scope of this assignment. However, integration tests are performed to ensure that various modules can be combined as a system and work as one unit without any interfacing issues.

11. Conclusions

This design report started with a motivation for self-driving cars that why we need them, which led to a bottom-up approach to achieving a self-driving car. Afterwards, it presented LCAS as one of the several automated systems required for such a bottom-up approach to self-driving cars. A functional safety analysis of the LCAS, according to ISO26262, is presented to establish that not only the product but also the process followed in this assignment is the technical state of the art. The requirements, designs, implementation, and verification followed. Finally, this chapter closes the report first by providing an account of the achievements made in this assignment followed by the relation of this work with the vision of self-driving cars. Furthermore, some recommendations and final thoughts on future work are presented at the end of the chapter.

11.1 Results

The main technical goal in this assignment was to design and deploy a real-time lane centering assist system in the test vehicle Toyota Prius. The focus, however, was shifted more towards the reliable perception of lane in real-time. The key factor, for this focus shift, was the lack of a reliable lane detection and tracking algorithm that can compute in real-time. As the lane detection and tracking is the core enabler of the LCAS, it was given the highest priority. Nevertheless, in this assignment, we have not only achieved a reliable lane detection and tracking algorithm, which computes with a strikingly low latency, but also a complete system design for the next generation LCAS.

The major tangible result, from this assignment, is a system that can be deployed in the test vehicle to provide a reliable detection and tracking of the ego lane, in real-time. The functional performance of the lane tracker has been informally evaluated by visual means. According to which, the algorithm performance is far superior to the conventional approaches. This is more evident when the vehicle is taking sharp turns, changing lanes or when the lane markings are not clearly visible. The results from the algorithm can be viewed by visiting the open source repository for the project[6]. A video sequence, recorded while driving in the city of Eindhoven, is provided in the description of the repository. Also, in Figure 37 snapshots from this video sequence are depicted.

The main achievements include functional and nonfunctional improvement of the state-the-art algorithm, an open-source implementation that can be deployed on any platform along with a real-time implementation that runs on a futuristic platform for self-driving cars, NXP BlueBox. The algorithm in its original form had a latency of around *250ms*, on a Core-i7, quad-core processor using MATLAB. With the proposed design and performing various optimizations, the latency has dropped down to just *7ms*, which is 35 times lower. The latency on a quad-core S32V234 system on the BlueBox is around *50ms* without accelerators and preliminary results using APEX accelerators indicate a latency of maximum 25ms. It is worth mentioning here that the original requirement for real-time lane tracking was set to *60ms*. The results are summarized in Table 3.

Besides providing a real-time lane detection and tracking, the system is also capable of providing an active steering to keep the vehicle, automatically, centered in the current driving lane. However, this functionality still needs to be tested on the road. The most important assets of this assignment, though, are the intangible results and contributions. These contributions include, providing an effective workflow, to develop an advanced driver assistance systems using the NXP BlueBox and quantitative evaluation of various computing unit utilization schemes for the heterogeneous platform. This also includes identifying weak links in the toolchains of the platform that are limiting its productivity. Above all, the process of system design followed in this assignment, is expected to set a quality standard for developing safety-critical applications in the subsequent projects.

[6] https://github.com/RameezI/TUeLaneTracker

Platform	*Specifications*	*Design Specifications*	*La-tency*
X86-64	Core-i7 quad-core @ 2.4 GHz	Original algorithm - MATLAB	**250ms**
X86-64	Core-i7 quad core, @ 2.4 GHz	Proposed Design- Native	**7ms**
S32V234	4xArm Cores @ 1000MHz	Proposed Design - Native	**50ms**
S32V234	4x Arm Cores and 2x APEX Cores	Proposed Design - Native	**25ms**

Table 3: Performance of the Software Application on Different Platforms

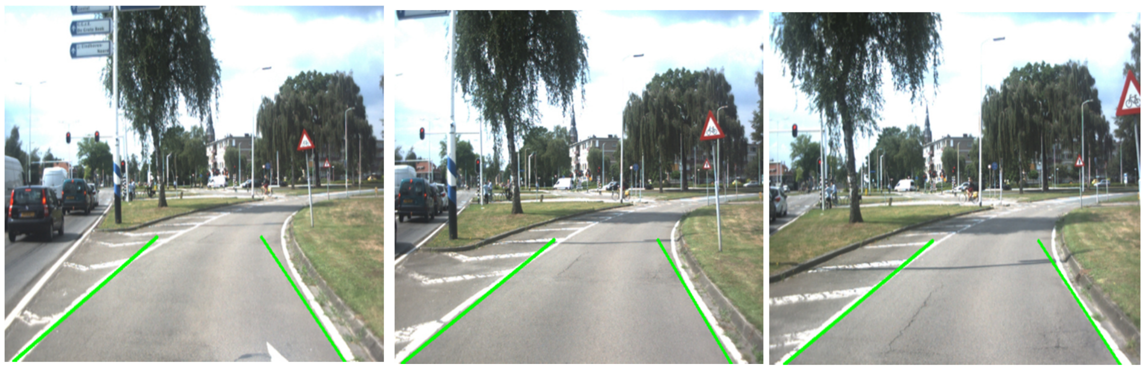


Figure 37: Lane Detection and Tracking with Proposed Design

As a concluding remark, the technology for self-driving cars has just started emerging and there is often a technology gap in the design assignments. For example, in case of the LCAS, this technology gap is the lack of a reliable lane tracker that computes in real-time. Through this assignment, on one hand, we have bridged this gap, by achieving a state-of-the-art lane tracker which compute with distinctly low latency, on the other hand, we have made a leap forward in smart-mobility, by creating concepts and understanding for the LCAS along with a design. Looking back, to evaluate how far we have come in this assignment, provides a sense of accomplishment. However, if we look forward, there is still a long way to go.

11.2 Future Work and Recommendations

The design and implementation of the LCAS provide a natural guide to future research in systems design as well as algorithm development. In the field of perception, advanced algorithms are heavily researched, however, there is less work on optimizing schemes or computation patterns that can lower the compute expense of such perception algorithms on modern compute devices. Consequently, often these advanced perception algorithms cannot make their way out of research departments. In this assignment, it is established that if such an algorithm is implemented with the architecture-centric approach, a considerable reduction in the compute expense can be achieved.

An architecture-centric design provides possibilities for explicit parallel programming. Also, it reveals various optimization options at the algorithm as well as compute device level. For example, providing an alternate path to the algorithm flow, optimizing the data flow between algorithm modules, precomputing various segments and pre-activating certain parts of the vision pipeline to name a few.

The designed lane detection and tracking algorithm need validation in various situations. In particular, it needs to be tested in challenging weather or lighting conditions. An advanced nonlinear controller is also required for accurate tracking of the target path and most importantly a realization of the supervisory controller, proposed in the LCAS system design, is essential from a functional safety point of view. Additionally, various recommendation regarding the algorithm has already been summarized in section 8.3.5. In respect of NXP BlueBox, the platform is an impressive balance of general purpose and domain-specific processing units which makes it energy efficient and yet a powerful compute device. The support and compatibility with open source libraries, for example, OpenCV, OpenCL, and OpenVX makes it even more attractive from the point of view of OEM and tier-1 suppliers. However, it still needs a stronger software support and a bigger ecosystem of tools around it. The available software and tools are not of the same automotive grade reliability as is the hardware and needs various improvements. Also, the platform needs to be promoted better by demonstrating advanced systems level applications, enabled by state-of-the-art algorithms. In my opinion, by overcoming these limitations, NXP BlueBox could gain an impregnable lead in defining the technology for the self-driving cars.

Bibliography

[1] D. Hendricks, J. Fell and M. Freedman, The relative frequency of unsafe driving acts in serious traffic crashes, The Office, 2001.

[2] T. JR, T. NS, M. ST, S. D and H. RD, “Tri-Level study of the causes of traffic accidents,” National Technical Information Service, Bloomington, 1977.

[3] D. Kahneman, Thinking, fast and slow, New York: Farrar, Straus and Giroux, 2011.

[4] S. Moss, "End of the car age: How cities are outgrowing the automobile," *The Guardian,* vol. 1, 2015.

[5] T. Keeney, "Mobility-as-a-service: Why self-driving cars could change everything," ARK Invest, New York, 2017.

[6] "Smart mobility," Technichal University of Eindhoven, [Online]. Available: https://www.tue.nl/en/research/strategic-area-smart-mobility/about-smart-mobility/. [Accessed 12 October 2017].

[7] M. Levy, D. Freeman and R. Malewski, "Comparing computing architectures for ADAS and autonomous vehicles," *Mechanix Illustrated,* March 2017.

[8] K. Amano, N. Goda, S. Nishida, Y. Ejima, T. Takeda and Y. Ohtani, "Estimation of the timing of human visual perception from magnetoencephalography," *Journal of Neuroscience,* vol. 26, pp. 3981--3991, 2006.

[9] D. Wardle, "The time delay in human vision," *The Physics Teacher,* vol. 36, pp. 442--444, 1998.

[10] R. Hartley and A. Zisserman, Multiple view geometry in computer vision, Cambridge University Press, 2003.

[11] M. Chen, T. Jochem and D. Pomerleau, "AURORA: A vision-based roadway departure warning system," in *Intelligent robots and systems*, IEEE, 1995, pp. 243-248.

[12] A. Lombard, X. Hao, A. Abbas-Turki and A. El Moudni, "Lateral control of an unmanned car using GNSS positionning in the context of connected vehicles," *Procedia Computer Science,* pp. 148--155, 2016.

[13] R. Hegde, G. Mishra and K. Gurumurthy, "An insight into the hardware and software complexity of ECUs in vehicles," in *Advances in computing and information technology*, Springer, 2011, pp. 99--106.

[14] J. D'Ambrosio and R. Debouk, "ASIL decomposition: the good, the bad, and the ugly," SAE International, 2013.

[15] B. Peters, "What drives a system architecht?," 22 August 2016. [Online]. Available: https://www.embedwise.com/2016/08/22/what-is-a-system-architect/. [Accessed 10 October 2017].

[16] A. Borkar, M. Hayes and M. T. Smith, "Lane detection and tracking using a layered approach," in *Advanced concepts for intelligent vision systems*, Berlin, Heidelberg, Springer, 2009, pp. 474--484.

[17] A. Borkar, M. Hayes and M. T. Smith, "Robust lane detection and tracking with RANSAC and kalman filter," *Image Processing (ICIP),* pp. 3261--3264, 2009.

[18] A. Cockburn, Agile software development, Boston: Addison-Wesley, 2002.

[19] H. Makabee, "Effective software design," 03 2014. [Online]. Available: https://effectivesoftwaredesign.com/2014/03/17/the-end-of-agile-death-by-over-simplification/. [Accessed 8 9 2017].

[20] M. Fowler and J. Highsmith, "The agile manifesto," *Software development,* pp. 28--35, 2001.

[21] S. Francois, *APEX programming workshop,* Munich: NXP Semiconductors, 2015.

[22] P. Zegelaar, J. Ploeg, W. Jansen and T. van der Sande, *4AT050 vehicle control,* Eindhoven: Technichal University of Eindhoven, 2016.

[23] "ISO26262 automotive safety standard," ISO - International organization for standardization, [Online]. Available: https://www.iso.org/obp/graphics/std//iso_std_iso_26262-7_ed-1_v1_en/fig_1.png. [Accessed 12 October 2017].

[24] "Lane keeping assist system," Maple Acura, [Online]. Available: http://www.mapleacura.com/lane-keeping-assist-system--lkas-.htm. [Accessed October 2017].

[25] D. I. I. Besselink, *Mechanics of road vehicles- 4AU10,* Eindhoven: Technichal University of Eindhoven, 2013.

[26] D. S. Herrmann, *ISP programming workshop,* Munich: NXP Semiconductors, 2016.

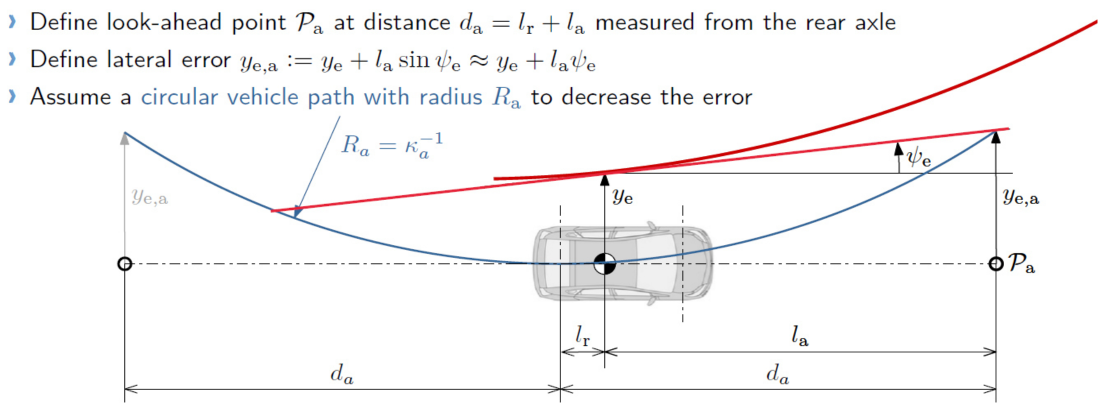
[27] A. Piovesan and J. Favaro, "Experience with ISO-262622 ASIL decomposition," Automotive Spin, Milano, 2011.

[28] "Automotive products by NXP," NXP Semiconductors, [Online]. Available: http://www.nxp.com/products/automotive-products:MC_50802. [Accessed October 2017].

Appendix A Lateral Control Strategy

The control strategy [22] is described in Figure 38, which assumes a circular path, shown in blue, for correcting the lateral error. The curvature of the target circular path depends on the lateral error $y_{e,a}$ at the look ahead distance 'l_a'. Once the curvature of the desired path is known, which can be found using simple geometry, the required steering angle can be obtained using a bicycle model, assuming constant velocity model. The steering angle is computed as $\delta = (l + \eta v x^2)\, ka$, where, l is wheel base, ka is the curvature of the path, η is the understeer gradient and v_x is the forward velocity. The lateral error at the look ahead distance is calculated using the forward facing camera. The final relation is evaluated as:

$$\delta = 2\left(\frac{l + \eta v x^2}{d_a^2}\right) y_{e,a}$$



- Define look-ahead point $\mathcal{P}_a$ at distance $d_a = l_r + l_a$ measured from the rear axle
- Define lateral error $y_{e,a} := y_e + l_a \sin\psi_e \approx y_e + l_a\psi_e$
- Assume a circular vehicle path with radius R_a to decrease the error

Figure 38: Control Strategy Using Lateral Error at Look Ahead Distance [22]

Appendix B ASIL Decomposition

Under certain circumstances, the ASIL for functional requirements can be lowered through the technique of ASIL decomposition, the possible combinations are shown in Figure 39

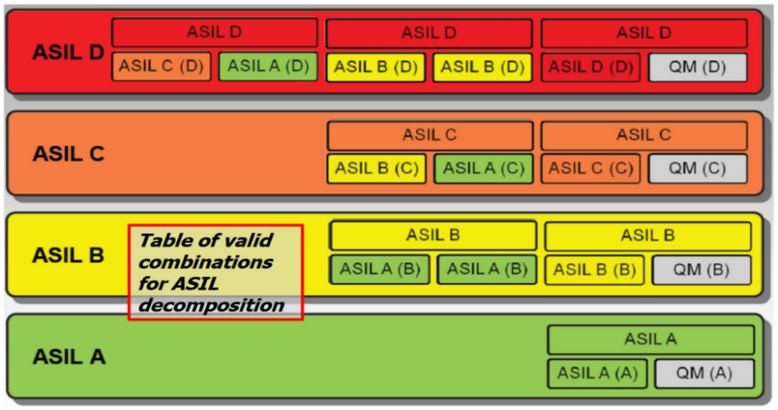


Figure 39: Valid Combinations for ASIL Decompositions [27]

Appendix C Scenarios and Driving Modes (HARA)

The various driving scenarios, considered in the process of Hazard Analysis are summarized below. The driving modes table provides information on the current state of the vehicle and the system while the driving scenarios one describes the road along with infrastructure and surroundings.

Driving Modes			
Modes	**Driving Speed (kph)**	**Operational Mode**	**LCAS Status**
High Speed (E0)	>50	Driving Straight	Disabled
High Speed (E1)	>50	Driving Straight	Enabled and Inactive
High Speed (E2)	>50	Driving Straight	Enabled and Inactive
High Speed (E3)	>50	Straying Sideways	Enabled and Inactive
High Speed (E4)	> 50	Changing Lane	Disabled
High Speed (E5)	> 50	Changing Lane	Enabled and Inactive
High Speed (E6)	> 50	Changing Lane	Enabled and Inactive

Operational Modes

Driving Scenarios		
ID	Road Type	Linearity
Highway (E1)	Highway Road	Straight
Highway (E2)	Highway Road	Curved (R>250m)
Country	Country Road	Straight/Curved

Operational Conditions

Appendix D Hazard Analysis and Risk Assessment (HARA)

Function	Hazard ID	Driving Scenario	Driving Mode	Malfunction	Hazard Description	Sever-ity	Exposure	Controllability	ID	Safety Goal	ASIL
LCAS enables itself automatically.	**Invalid Ena-ble**	Country road	High-speed (E0)	LCAS got enabled in invalid operating con-ditions.	The system is not able to perform the function in cur-rent operating conditions. The system may steer the vehicle unexpectedly which could lead to a fatal accident.	**S3**	**E4**	**C3**	**SG1**	**LCAS must stay disabled unless operational condi-tions are satisfied.**	**D**
LCAS takes control of the steering func-tion on driver's re-quest.	**Manual Commission Failure**	Highway (E1, E2)	High-speed (E1)	LCAS is activated by the user however acti-vation did not take ef-fect.	It can cause confusion for the driver and may lead to delayed response while the vehicle starts inadvertently straying from the lane.	**S0**	**E4**	**C1**	**SG2**	**LCAS must alert the driver if the request to commission the active steering failed.**	QM
LCAS automatically takes control of the active steering.	**Automatic Commission Failure**	Highway (E1, E2)	High-speed (E3)	The car is straying sideways, LCAS is expected to automati-cally steer the vehicle, but the system takes no steering actions.	Driver expects LCAS to be activated automatically and is not attentive. As the auto-matic activation failed this can lead to a lateral colli-sion.	**S3**	**E4**	**C3**	**SG3**	**LCAS must ensure that the active steering is com-missioned without excep-tion when the vehicle is in-advertently straying off the lane.**	**D**

	Unintended Commission	Highway (E1, E2)	High-speed (E5)	The driver is intentionally changing the lanes. System thinks of it as a lane drift and automatically start steering the vehicle back in the lane.	LCAS is trying to counter steer the car to keep it centered in the current while the driver intends to steer it away. This conflict may cause a collision.	**S3**	**E4**	**C3**	**SG4**	**Driver must be able to seamlessly override the steering inputs from LCAS by providing torque on the steering wheel.**	**D**
LCAS informs the driver of its current state.	**Uninformed Omission**	Highway (E1, E2)	High-speed (E2, E6)	Active steering is deactivated but the driver is not informed.	Driver expects LCAS to steer the car and is not alert anymore. As the activation failed this can cause a delayed response in an emergency situation leading to a lateral collision.	**S3**	**E4**	**C3**	**SG5**	**LCAS must regularly monitor its status and alert the driver in case of any state change.**	**D**
	Uninformed Commission	Highway (E1, E2)	High-speed (E1, E5)	Driver expects LCAS to be OFF however active steering is commissioned and the driver is not informed.	.While changing lanes this could result in a delayed override response from the driver.	**S3**	**E4**	**C2**			**C**
LCAS provide active steering to keep the vehicle centered in the lane.	**Unstable Yaw rate**	Highway (E1, E2)	High-speed (E2)	System steers the vehicle abruptly causing a high yaw rate of the vehicle.	An unstable yaw rate can cause instability in the vehicle leading to a fatal crash.	**S3**	**E4**	**C3**	**SG6**	**LCAS must ensure lateral stability when actively steering.**	**D**
	Unexpected Lateral Movement	Highway (E1, E2)	High-speed (E2)	System steers the vehicle on a fallacious or arbitrary target path.	Steering the car on an arbitrary path could lead the vehicle out of the lane thus resulting in a fatal accident.	**S3**	**E4**	**C3**	**SG7**	**The system must ensure the integrity of the target path. The lateral error between the target path and the actual lane center must stay within +/-10cm bound while the heading error between the lane and the target path must be within +/-5 degrees.**	**D**

	Target Path Tracking Failed	Highway (E1, E2)	High-speed (E2)	The vehicle is not able to follow the target path with the desired performance.	As system cannot maintain the target path, This could lead to a lateral accident.	**S3**	**E4**	**C3**	**SG8**	**System must ensure that the vehicle follows the target path adequately. The lateral error must stay within a bound of +/- 5cm while the heading error should stay within +/- 2 degrees.**	**D**
LCAS disable itself automatically.	**Ungraceful Disable**	Country	High-speed (E2, E6)	LCAS is active but the operational conditions do not satisfy anymore, The system disabled itself immediately without deactivating.	As the driver is relying on LCAS to steer the car a sudden disable could lead to an accident.	**S4**	**E4**	**C3**	**SG10**	**The system must deactivate the active steering control and wait for the driver to take control, while driving in safe mode, before disabling the interface.**	**D**
LCAS is shut down by the driver.	**Shutdown failure**	Highway	High-speed (E2)	LCAS system is shut down but it got enabled while driving.	As the driver has completely shut down the system, the system could have technical problems, turning it on without driver consent could lead to a fatal accident.	**S5**	**E4**	**C3**	**SG11**	**If the Shutdown option is checked, the system must be non-existent and must not influence or assist in any driving task.**	**D**

Appendix E Image Signal Processor (ISP)

The Image is acquired by the camera, through a MIPI CSI2 interface, and sent to the ISP line by line. Therefore, the camera does not need to wait until the whole frame is acquired. ISP is comprised of various Image Processing Units (IPUs) shown in Figure 40, which are mainly responsible extracting the pixel information form a serial stream coming from the camera sensor. As shown, various IPUs are employed, in form of a pipeline, to achieve this function. Because of the MIMD architecture, all compute engine are capable of executing different instructions at the same time. The intermediate results from the individual IPU are stored in a static ram before transferring to next IPU, the reason is the limited local memory of the individual compute engine. Finally, an FDMA module commits the results to a dynamic ram which is mapped by the user application.

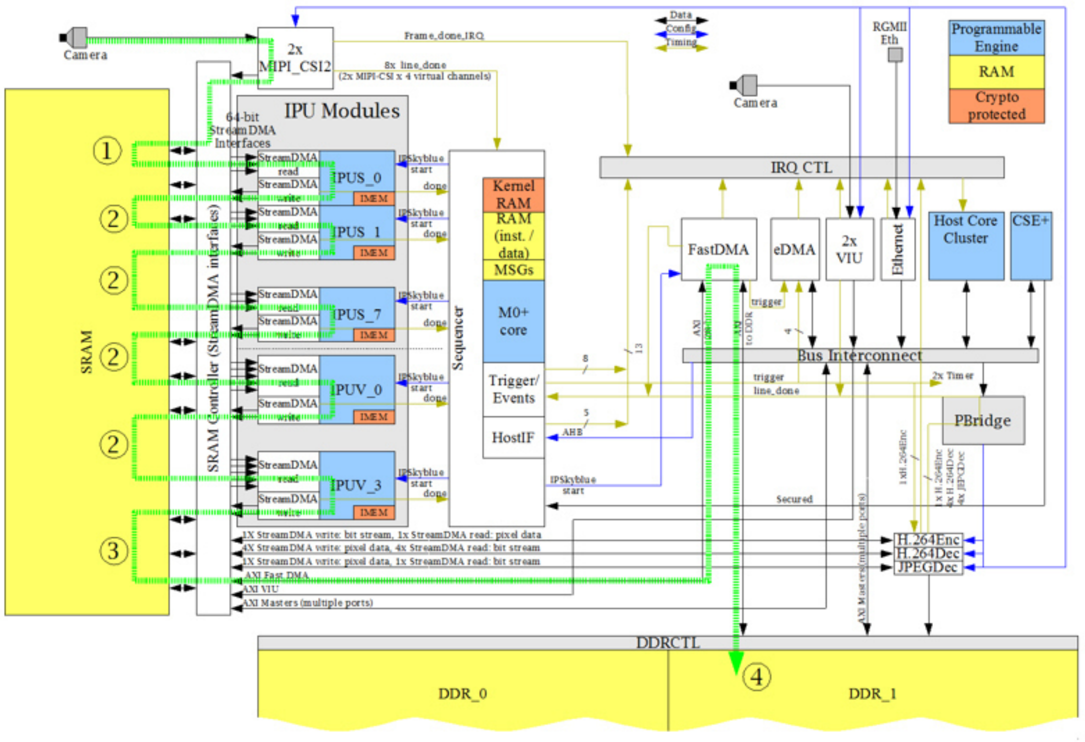


Figure 40: Image Processing Using ISP Module of the NXP BlueBox [26]

About the Author

Rameez Ismail is an automotive systems designer with a background in robotics and mechatronic systems. He enjoys working on multidisciplinary projects and is quite enthusiastic about the concepts of self-driving cars and smart mobility. Some of his expertise include machine learning, robotics, computer vision and artificial intelligence. Since November 2015, he is working as a PDEng trainee in the area of automotive systems design where he is being trained on the systems aspects of solving automotive design problems. During this training, he assisted various automotive companies in designing next-generation driver assistance systems and defining concepts for automated driving.

Where innovation starts