Chair of Computer Engineering, Dept. of Computer Science in cooperation with

Elektrobit

In Advanced Driver Assistance(ADAS)

Master Thesis

In

Development of a smart-phone based augmented reality view application for driver assistance systems.

Submitted in Fulfillment of the Requirements for The Academic Degree M.Sc.Automotive Software Engineering

Submitted by: Akshay Naresh Lotankar
Matr.-Nr. 388838

Supervisors: Prof. Dr. Wolfram Hardt
Dipl.-Inf. René Bergelt
Dipl.-Inf. Thomas Wagner(EB)

Date of submission: 27 March 2017
Abstract

The goal of this thesis is to develop a smartphone application for augmented reality view; it is an initial attempt to realize a driver assistance functionality using just a smartphone and an external lens. Initially it depicts a brief analysis about the most feasible development technologies for mobile application development, selecting a proper lens and positioning of the smartphone in the car. Later, it discusses the strategies for real-time object detection using OpenCV; the video frames are processed using the strategies to find patterns in the videos. Different techniques like Hough-line transform, watershed, contour detection, color segmentation, color thresholding and HAAR cascades are implemented and compared in terms of real time detection of the desired objects. Then a unified algorithm is implemented for the given scenario which overcomes the challenges faced during the conceptualization phase. Finally, the results are depicted with the snapshots of the real time detection done from the smartphone and then evaluated against the vision of the application and the achieved tasks. This thesis is concluded by stating the prospects of this mobile application in the future.

Keywords: augmented reality, OpenCV, driver assistance, color segmentation.
Declaration

I hereby declare that this master thesis in topic “Development of a smartphone based augmented reality view application for driver assistance systems”, is entirely the result of my own work and it has been written by me in its totality. Also, I certify that I elaborated this research independently. The work is based on the foundation of the information sources and literature used in the thesis that I have faithfully and properly cited.

Cheminitz, March 22, 2016
Place, Date

Akshay Naresh Lotankar
Acknowledgement

This research work had been a great opportunity for my personal and professional development. I am grateful for getting to meet so many wonderful people and professionals who motivated and guided me throughout this period.

I would like to express my deepest of regards to my professor Prof. Dr. Wolfram Hardt and my advisor Dipl.-Inf. René Bergelt. Their constant enlightenment on the subject and persistent motivation helped through each stage of my master thesis.

I wish to express my deepest gratitude to the department of Driver Assistance, Elektrobit GmbH for giving me the opportunity for pursuing my master thesis in their teams. My manager Dr. Reinhard Ernst, Line Manager, Team Manager, Driver Assistance, who in spite of being busy with his duties, spared time to hear, guide and keep me on the correct path. Ardent thank you to my mentor, Dipl.-Inf. Thomas Wagner, Project Manager, for taking part in useful decisions and giving necessary advices and guidance to make life easier. I choose this moment to acknowledge his contribution gratefully.

Lastly, I would like to thank my parents Mr. Naresh Lotankar and Mrs. Arti Lotankar and all my friends for always supporting and motivating me to do everything that felt unfathomable. I can’t thank them enough for everything they have done for me. I see this opportunity as a big stepping stone in my career development. A genuine thank to everyone for helping me successfully complete my master thesis.

Place: Chemnitz
Date: March, 22, 2017

Akshay Naresh Lotankar
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>vi</td>
</tr>
<tr>
<td>List of Tables</td>
<td>ix</td>
</tr>
<tr>
<td>List of Abbreviations</td>
<td>x</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.1.1 Connected Car</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Purpose</td>
<td>3</td>
</tr>
<tr>
<td>1.2.1 Safety</td>
<td>3</td>
</tr>
<tr>
<td>1.2.2 Automotive Innovations</td>
<td>5</td>
</tr>
<tr>
<td>1.2.3 Costs</td>
<td>5</td>
</tr>
<tr>
<td>1.3 Challenges</td>
<td>6</td>
</tr>
<tr>
<td>1.3.1 Real time performance</td>
<td>6</td>
</tr>
<tr>
<td>1.3.2 Security</td>
<td>7</td>
</tr>
<tr>
<td>1.4 Problem Statement</td>
<td>7</td>
</tr>
<tr>
<td>1.5 Thesis Structure</td>
<td>7</td>
</tr>
<tr>
<td>1.5.1 Chapter 2</td>
<td>7</td>
</tr>
<tr>
<td>1.5.2 Chapter 3</td>
<td>8</td>
</tr>
<tr>
<td>1.5.3 Chapter 4</td>
<td>8</td>
</tr>
<tr>
<td>1.5.4 Chapter 5</td>
<td>8</td>
</tr>
<tr>
<td>1.5.5 Chapter 6</td>
<td>8</td>
</tr>
<tr>
<td>1.5.6 Chapter 7</td>
<td>8</td>
</tr>
<tr>
<td>1.5.7 Chapter 8</td>
<td>9</td>
</tr>
<tr>
<td>2 State Of The Art</td>
<td>10</td>
</tr>
<tr>
<td>2.1 Evolution of Head-Up Display</td>
<td>10</td>
</tr>
<tr>
<td>2.2 Augmented Reality</td>
<td>11</td>
</tr>
<tr>
<td>2.3 Continental AR-HUDs</td>
<td>12</td>
</tr>
</tbody>
</table>
3 Approach

3.1 Technologies used for developing a camera based mobile application

3.1.1 Image processing

3.1.2 OpenCV

3.2 Development Platform

3.2.1 Slow Execution

3.2.2 Security Concerns

3.2.3 Garbage Collection

3.2.4 Performance

3.3 Virtualization Analysis

3.3.1 Issues

3.4 Determining the position for the smart phone on the dashboard

3.5 Selection of the Lens

3.5.1 Fisheye lens

3.5.2 Dot/Kogeto 360°

3.5.3 Issue
3.5.4 Bubblescope ............................................. 29
3.6 Conceptualization ........................................ 30
    3.6.1 iOS Development ................................... 30
3.7 Camera based Object Detection .......................... 34
    3.7.1 Feature based Detection ............................. 34
    3.7.2 Template based Object Detection .................. 35
3.8 Application Functionalities ............................. 36
    3.8.1 Camera based Lane Detection ....................... 36
    3.8.2 Camera based Face Detection ....................... 37
    3.8.3 Perspective Vision ................................. 37
3.9 Task Classification ..................................... 37
    3.9.1 Development framework for MacOSX ................ 37
    3.9.2 Programming language .............................. 37
    3.9.3 User interface ..................................... 39
3.10 Summary ................................................. 39

4 Prerequisites ............................................. 40
    4.1 Tools and Technologies ................................ 40
        4.1.1 Hardware ........................................ 40
        4.1.2 Software ........................................ 40
    4.2 Image Processing ..................................... 41
    4.3 Image preprocessing .................................. 41
        4.3.1 Noise Reduction ................................. 41
        4.3.2 Compatibility ................................... 42
    4.4 Image Container ....................................... 42
    4.5 OpenCV ............................................... 42
    4.6 Morphological Operations ............................. 43
        4.6.1 Erosion .......................................... 43
        4.6.2 Dilation ......................................... 43
        4.6.3 Opening .......................................... 44
        4.6.4 Closing .......................................... 44
4.7 Color Formats .............................................. 44
  4.7.1 BGR .................................................. 44
  4.7.2 HSV .................................................. 45
  4.7.3 Gray Scale .......................................... 46
  4.7.4 Binary ............................................... 46
4.8 Geometric Transformations of Image ......................... 46
  4.8.1 Affine Transform .................................... 46
  4.8.2 Perspective Transform .............................. 47
  4.8.3 Thresholding ........................................ 47
4.9 Image Smoothing .......................................... 48
4.10 OpenCV Algorithms ....................................... 49
  4.10.1 Canny Edge Detection .............................. 49
  4.10.2 Contour Detection .................................. 51
  4.10.3 Hough Line Transform .............................. 53
  4.10.4 Watershed Algorithm .............................. 55
  4.10.5 ROI .................................................. 57
5 Implementation .............................................. 59
  5.1 ROI determination ...................................... 59
  5.2 ROI Sharpening ........................................ 61
    5.2.1 Algorithm Explanation ............................ 61
  5.3 Implemented Algorithms for Lane Detection ............... 64
    5.3.1 Vanishing point .................................. 64
    5.3.2 Detection via integrating contour and watershed algo-
         rithms ............................................... 65
    5.3.3 Detection via contour and convex hull ............. 68
    5.3.4 Face Detection using Haar Cascades ................ 70
  5.4 Unified Algorithms ..................................... 75
    5.4.1 Lane Detection .................................. 75
    5.4.2 Face Detection .................................. 79
# Results and Evaluation

## 6.1 Results

- 6.1.1 Construction and Mounting
- 6.1.2 Working

## 6.2 Evaluation

- 6.2.1 Vision
- 6.2.2 Tasks achieved
- 6.2.3 Performance

---

# Conclusion

# Future Scope

# Bibliography
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Car and Technology Merger[1]</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Effects Of HUD on Driver’s Attention</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Mobile Interface for BMW</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>First Head-Up-Display[1]</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>AR-HUD</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>Lane Departure Warning[2]</td>
<td>16</td>
</tr>
<tr>
<td>11</td>
<td>HUDWAY Glass[3]</td>
<td>17</td>
</tr>
<tr>
<td>12</td>
<td>Available Adjustments[3]</td>
<td>18</td>
</tr>
<tr>
<td>13</td>
<td>BMW HUD Implementation</td>
<td>20</td>
</tr>
<tr>
<td>14</td>
<td>Circular Fisheye</td>
<td>27</td>
</tr>
<tr>
<td>15</td>
<td>Kogeto 360°</td>
<td>28</td>
</tr>
<tr>
<td>16</td>
<td>Bubblescope</td>
<td>29</td>
</tr>
<tr>
<td>17</td>
<td>Interface Builder</td>
<td>31</td>
</tr>
<tr>
<td>18</td>
<td>MVC Architecture</td>
<td>32</td>
</tr>
<tr>
<td>19</td>
<td>Object Detection Techniques</td>
<td>34</td>
</tr>
<tr>
<td>20</td>
<td>Functionalities</td>
<td>36</td>
</tr>
<tr>
<td>21</td>
<td>Application Behavior</td>
<td>38</td>
</tr>
<tr>
<td>22</td>
<td>Digital Image Processing</td>
<td>41</td>
</tr>
<tr>
<td>23</td>
<td>BGR Color Space</td>
<td>44</td>
</tr>
<tr>
<td>24</td>
<td>HSV Color Space</td>
<td>45</td>
</tr>
<tr>
<td>25</td>
<td>Types of Simple Thresholding</td>
<td>48</td>
</tr>
<tr>
<td>26</td>
<td>Otsu’s Thresholding</td>
<td>49</td>
</tr>
<tr>
<td>27</td>
<td>Canny Edge Detection</td>
<td>50</td>
</tr>
<tr>
<td>28</td>
<td>Contour Detection</td>
<td>52</td>
</tr>
<tr>
<td>29</td>
<td>Hough Line Transform</td>
<td>54</td>
</tr>
<tr>
<td>Page</td>
<td>Section</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Watershed Algorithm</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>ROI</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>Input Image</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>Perspective Transform</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>Sharpened Image</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>Perspective Image</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>Sharpened Image</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>ROI</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>Vanishing point</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>Vanishing Point Detection</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>Contour and watershed Integration</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>Watershed Marker Image</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>Contour Detection with Convex Hull</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>Contour and Convex Hull Detection</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>Image Features</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>Haar Features</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>Detection without lens</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>Detection with Lens</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>Unified Algorithm for Lane Detection</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>Sharpened ROI</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>Contour Detection on ROI</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>Largest Contour extracted</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>Foreground and Background Object separation</td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>Mirror Image on the mobile screen</td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>Face Detection</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>Detected Contour</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>Experimental Setup-Front View</td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>Experimental Setup-Side View</td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>Experimental Setup-Top View</td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>Results(a)</td>
<td></td>
</tr>
</tbody>
</table>
60  Results(b) ................................................. 85
61  Results(c) ................................................. 86
62  Results(d) ................................................. 86
63  Results(e) ................................................. 87
List of Tables

<table>
<thead>
<tr>
<th></th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>HUD Comparison</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>Probable Options for Virtualization</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>Evaluation Summary</td>
<td>89</td>
</tr>
</tbody>
</table>
List of Abbreviations

OpenCV  Open Source Computer Vision
OEM  Original Equipment Manufacturer
BMW  Bavarian Motor Works
VW  Volks Wagen
DFKI  German Research Center for Artificial Intelligence
RosPA  Road Safety Practitioners
AR-HUD  Augmented Reality Head-Up Display
ACC  Adaptive Cruise Control
LDW  Lane Departure Warning
GPS  Global Positioning System
HMI  Human Machine Interface
PD  Projection Distance
PGU  Picture Generating Unit
AR  Augmented Reality
TFT  Thin Film Transistor
LED  Light Emitting Diode
DMD  Digital Micromirror Device
OS  Operating System
3D  3 Dimensional
BSD  Berkeley Software Distribution
JNI  JAVA Native Interface
JVM  JAVA Virtual Machine
CPU  Central Processing Unit
EFI  Electronic Fuel Injection
VM  Virtual Machine
SMP  Symmetric Multi Processing
FOV  Field Of View
SDK  Software Development Kit
UI  User Interface
MVC  Model View Controller
IDE  Integrated Development Environment
GPU  Graphics Processing Unit
FPS  Frames per Second
RAM  Random Access Memory
RGB  Red Green Black
BGR  Blue Green Red
HSV  Hue Saturation Value
ROI  Region Of Interest
1 Introduction

1.1 Motivation

Recent innovations in the automotive industry have been in the fields of computers and electronics. One of the most significant innovations is the autonomous driving. It is predicted that by 2025, autonomous navigation will be introduced into mainstream by major OEMs. Autonomous driving is categorized into five different levels spanning from 0 to 4\cite{5}, levels 0-2 have been already introduced in certain vehicles, whereas levels 3 and 4 are currently in the development phase. Due to the evolution of autonomous vehicles the automotive industry is undergoing many standardization changes, so that the upgraded systems are able to cater to the requirements for high speed communication, maintaining huge data loads etc. But implementing these things is still a challenge.

On the other hand the mobile technology is gradually evolving. Now the mobile industries are focused on harnessing this technology to bridge gaps between different real time domains. It has been estimated that by 2017 the subscription figure will go above 9 billion. The penetration of mobiles phones in North America and Europe is said to be 70 and 80 percent by 2018\cite{1}. Open architecture mobile platforms when combined with the infrastructure of the app developers will unlock a whole new technical potential. For example, there is a tie between driving and hotels, driving and service stations. Most of the connected car applications will deal in the areas of real time car data, navigation, diagnosis database to provide car information, rather than just voice enabling an existing application.

1.1.1 Connected Car

One of the major focuses in developing a car is the interaction between the car and the driver. Millions of vehicles on the road with the infotainment systems, a touch screen with voice recognition capability serve as the user
Figure 1: Car and Technology Merger[1]
interface. This functionality is a subset of “Connected Car”, which is a new automotive mainstream. It is now developing its functions in parts by integrating with smartphone and third party mobile applications. To support this change the OEMs are merging with the software giants. To share the risks and reduce the costs, the automotive industries have formed a complex alliance. Google and Fiat Chrysler have paired up to develop 100 autonomous minivans; Google, Toyota, Tata and Microsoft have invested in Uber. Together BMW, VW and Daimler have acquired HERE maps[6].

1.2 Purpose

1.2.1 Safety

According to the statistics of RoSPA safety guide, 95% of the total accidents happen due to some human error and 76 percent of the road accidents happen solely because of human error([7]). Autonomous driving can be used to reduce the accidents caused by human error, thus making the roads safe. Not only that, but it acquaints the driver with the time to do other chores like business or entertainment purpose. Autonomous vehicles could considerably reduce the rate of traffic and congestion on the roads because these cars will travel at a higher speed in a systematic manner without hitting other cars, thus decreasing the extra fuel consumption. The table below is an estimate of the benefits caused because of Autonomous Vehicles.

There are two major causes for the distraction of the driver; checking the instrument panel and looking at the navigation panel. For one mile the driver looks away from the screen for approximately 300ft. When an HUD is projected in front of the driver this distraction is narrowed nearly to zero. This is a significant leap to curb the accidents caused by visual distractions[3].
Table 1: Estimates of Annual Economic Benefits from Autonomous Vehicles in the United States[4]

<table>
<thead>
<tr>
<th>Assumed Adoption Rate</th>
<th>10%</th>
<th>50%</th>
<th>90%</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cash cost savings from AVs</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lives saved (per year)</td>
<td>1,100</td>
<td>9,600</td>
<td>21,700</td>
</tr>
<tr>
<td>Fewer crashes</td>
<td>211,000</td>
<td>1,880,000</td>
<td>4,220,000</td>
</tr>
<tr>
<td>Economic cost savings</td>
<td>$5.5B</td>
<td>$48.8B</td>
<td>$109.7B</td>
</tr>
<tr>
<td>Comprehensive cost savings</td>
<td>$17.7B</td>
<td>$158.1B</td>
<td>$355.4B</td>
</tr>
<tr>
<td>Economic cost savings per AV</td>
<td>$430</td>
<td>$770</td>
<td>$960</td>
</tr>
<tr>
<td>Comprehensive cost savings per AV</td>
<td>$1,390</td>
<td>$2,480</td>
<td>$3,100</td>
</tr>
<tr>
<td><strong>Congestion benefits</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Travel time savings (millions of hours)</td>
<td>756</td>
<td>1680</td>
<td>2772</td>
</tr>
<tr>
<td>Fuel savings (millions of gallons)</td>
<td>102</td>
<td>224</td>
<td>724</td>
</tr>
<tr>
<td>Total savings</td>
<td>$16.8B</td>
<td>$37.4B</td>
<td>$63.0B</td>
</tr>
<tr>
<td>Savings per AV</td>
<td>$1,320</td>
<td>$590</td>
<td>$550</td>
</tr>
<tr>
<td><strong>Other AV impacts</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parking savings</td>
<td>$3.20</td>
<td>$15.90</td>
<td>$28.7</td>
</tr>
<tr>
<td>Savings per AV</td>
<td>$250</td>
<td>$250</td>
<td>$250</td>
</tr>
<tr>
<td>Vehicle miles traveled increased</td>
<td>2.0%</td>
<td>7.5%</td>
<td>9.0%</td>
</tr>
<tr>
<td>Change in total # of vehicles</td>
<td>≈ 4.7%</td>
<td>≈ 23.7%</td>
<td>≈ 42.6%</td>
</tr>
<tr>
<td>Annual savings: economic costs only</td>
<td>$25.5B</td>
<td>$102.2B</td>
<td>$201.4B</td>
</tr>
<tr>
<td>Annual savings: comprehensive costs</td>
<td>$37.7B</td>
<td>$211.5B</td>
<td>$447.1B</td>
</tr>
</tbody>
</table>

Figure 2: Effects Of HUD on Driver’s Attention
1.2.2 Automotive Innovations

There are several functionalities being developed which comprise the different levels of autonomous driving. For example the intelligent drive of Mercedes gives an option to the driver to allow the car to steer, brake and accelerate by itself(4). BMW i3 provides a mobile phone app, through which the driver can check the capacity of his car battery. The Volkswagen group is developing a system which would permit a driver to park his vehicle using his smartphone. Thus, it infers that the technology gap between a car and a computer is reducing day by day.

1.2.3 Costs

Installing HUD units in cars costs approximately 1000-2000 depending on the functionalities the drivers chooses. It is mostly available for the luxury segment vehicles. The cheap variants are still under the development phase. Considering the price, the car segment between 25000-35000 will be targeted which comprises majority of the automotive market.

Considering the above mentioned factors, it could be clearly observed how closely smartphone applications are integrated with the cars nowadays. In some cases the mobile applications have shown better results than the car applications, for example the parking application from Audi. Along with the ease of use, substituting a smartphone for any car functionality is price efficient. Smart phone applications can be developed independently without any dependency to the vehicle. Maintaining or upgrading a smartphone application is no big deal. In fact replacing the car application with a mobile application will effectively reduce the computational overhead required to build a car application.
1.3 Challenges

1.3.1 Real time performance

A connected car application functions mostly in a real time scenario for example the navigation app, so they have to provide a productive integration infrastructure, which should facilitate a restful web service for integration for communicating with the server based data.

Figure 3: Mobile Interface for BMW
1.3.2 Security

Building mobile applications for connected cars or for compatibility with connected cars so that the applications can interact with the underlying car platform is almost the same as developing an enterprise mobile application. Hence they should be equally cautious in terms of the application security.

1.4 Problem Statement

The aim of the thesis is to develop a smartphone phone based AR-HUD which assists the driver for navigation. This is done only by using an external lens which is attached to the rear camera of the phone. The datasets is a video from the lens. So, the only information available is 2-dimensional straight up view ahead of vehicle and the view on the driver’s side. The algorithm is able to detect the road trajectory as well as give a notification in case the driver is not in a specified position and at the same time the phone through its screen; should project a HUD on the windscreen showing the lane information. This system acquaints the driver with just a smartphone based navigation system.

1.5 Thesis Structure

This study is structured in the following manner:

1.5.1 Chapter 2

State of the Art depicts a brief overview about the ongoing work in the field of a head up display implementation in the automotive industry along with its approaches, advantages and shortcomings. As it is one of the latest evolving technologies in the field of automotive, the scope for a detailed literature research is challenging. An overview on the research approaches dealing in the similar domains has been presented.
1.5.2 Chapter 3

Approach introduces the approach to accomplish the goal of the master thesis. It explains in brief as to why these concepts are preferred for the given scenario. This chapter is essential to understand the later chapters where the implementation and results are explained in detail.

1.5.3 Chapter 4

Prerequisites gives an overview about all the tools and technologies which were implemented for the application development. It explains generic OpenCV algorithms which were later used as building blocks for developing the unified algorithms.

1.5.4 Chapter 5

Implementation is the realization of the approach. This chapter discusses the different methods which are used to implement the predefined approach. At the beginning of the chapter some prerequisites are mentioned, which are necessary to understand certain image processing algorithms. Then all the feasible algorithms which are implemented are explained along with their outputs. At the end unified algorithms are explained which are developed for achieving the tasks.

1.5.5 Chapter 6

Results and Evaluation depicts the snapshots of the outputs of the algorithms which are developed, it also explains the construction and working of the prototype built for verifying the algorithm in real-time. Later it compares the proposed goals with the achieved tasks in the given time frame.

1.5.6 Chapter 7

The conclusion is summarized in this chapter.
1.5.7 Chapter 8

Future Scope and Conclusion shows the future scope of the work in the context of this study. Suggestions with respect to the current work are presented.
2 State Of The Art

2.1 Evolution of Head-Up Display

The idea of HUDs was proposed in the automotive stream by General Motors in 1988\cite{8}. These displays were primarily used for displaying information such as speed, tachometer and other basic readings from the dashboard. Due to the technical advancements the generic HUDs are now replaced with a newer technology commonly known as augmented reality. Figure 4 depicts the first HUD which was implemented in automotive by General Motors.

Figure 4: First Head-Up-Display\cite{1}
2.2 Augmented Reality

“Augmented” is an extended form of a conventional HUD. For example, as compared to a conventional HUD which displays speed, in an AR-HUD the display is oriented and rendered at a better ergonomic position with respect to the driver. It is positioned in a virtual space in front of the driver at the center and is also acquainted with information related to ACC, LDW, Navigation and control warnings.

These AR systems are further more advanced; they are integrated with GPS systems, cameras, internet and smartphones. They transform the car’s windshield into an on-board information display.

A huge group of people comprising the physicists, anthropologists, electronic experts and graphic designers have turned their focus to AR-HUD
development. An AR-HUD optical system enables a driver by rendering an augmented display of the status of driver assistance systems and the importance of this information in the driver’s field of vision. This system is strongly connected with the environmental sensors of the driver assistance systems, as well as the navigation data, and the vehicle dynamic data.

2.3 Continental AR-HUDs

The AR-HUD development currently is at a pre-production stage. Kia K9 sedan has been equipped with this technology; rigorous testing and monitoring are being carried out. Continental has proposed to reveal the AR-HUD for production by later this year. As of now, there are 10 global OEMs which are offering HUD technology on a total of 19 models. It is also predicted that by the following year, the number of HUDs will be inclined to more than 5-million units worldwide.[2]

2.3.1 Implementation

The principle behind Continental AR-HUD is a 1.2-GHz quad-core processor which is known as the “AR-Creator,” the inputs to this processor are given by a radar sensor which is used for the purpose of ACC, Continental’s mono camera which is used for lane keeping and object detection, and an “eHorizon,” which is nothing but the information generated from the navigation/GPS system. As this has to function in real time i.e. when the vehicle is in motion, a Kalman filter algorithm is implemented which assists in the determination of the future state of the affairs.

2.3.2 Construction and mounting

Figure 6 is a pictorial representation of the construction of a Continental HUD. When the projection planes are taken into consideration, the conventional HUD information is considered “near”. That means, once the driver
positions himself by adjusting the seat position and height for the purposes of driving the vehicle; consequently an adjustment is needed so that the HUD display is viewed in a centered way. This is called as establishing an “eye box”.

The near plane appears to the driver 2.4 m away, or in effect, at the end of the hood. The scope of the information is 210 x 42 mm. It is produced by a PGU that consists of a TFT display that’s backlit by LEDs. This is then bounced off of a curved mirror that both enlarges the information and “places” it at the end of the hood.

Then there is the second layer, the “augmentation” layer. It appears to be 7.5 m in front of the driver. Now this distance is very important, if the information is rendered beyond 7.5m then there is a possibility of the projected information to interfere with the preceding vehicle on the road and
might interfere with the driving tasks.

There is a digital micro mirror device that is mounted in such a manner that it dissects the AR-HUD PGU, or “picture generating unit”. The technology behind DMD is developed by Texas Instruments. A DMD when commercially used has up to 8.8-million microscopic mirrors.

When implemented for AR-HUD; DMD is an optical semiconductor which includes thousands of tiny electro-statically tilted mirrors. The mirrors are illuminated by three basic colors LEDs-red, green, and blue- with some mirrors reflecting the light and others allowing it to travel through. After being incident on a focusing screen, it gets reflected onto a bigger mirror, and finally it is rendered on the windshield. The dimension of the augmented viewing region is 130 x 63 cm.

2.3.3 Characteristics

**Standard settings:** When the AR-HUD used with standard settings then the display consists of the speed of the vehicle, speed limits of the road and it could be also configured for rendering the control signals on the display. A pictorial representation of this function is shown in figure 7.

![Figure 7: Standard Settings](image)
**ACC:** In the case of ACC setting, the car that precedes is marked with a green crescent below its rear end, not only this but it displays a blue colored trapezoidal trail descending in size with respect to distant from the car. When the driver decides to accelerate and override the ACC, then as the distance between the cars decreases the green notification changes to red. A pictorial representation of this function is shown in figure 8.

![Figure 8: Adaptive Cruise Control](image)

**LDW:** In this case, when the car approaches a lane, a line of red dots coinciding that lane is displayed; these red dots are called as “cats eyes”. These are based on the roadmounted reflectors, perhaps more common to the European roads as compared to U.S.A. The feedback could be given in different ways, for example visual warning on the display, acoustic warning in the form of a beep or in our current case of Kia K9 which gives a haptic feedback by vibrating a seat on the side where the lane departure is happening. A pictorial representation of this function is shown in figure 9.

**Navigation:** There is a trail of blue arrows to depict the route. So, in case of a notification for a road or direction change, the arrows change to
the respective side of that change. This is commonly called a “fishbone”. A pictorial representation of this function is shown in figure10.

2.3.4 Limitations

It is a proprietary product, which means its application, is narrowed to a certain set of vehicles.
The research comprising the components and the technology being used is ongoing. Hence a generic version is made available which facilitates the driver just with the current speed of the vehicle, speed limit etc. There are other more functional and similar products at a cheaper price available.

The market price of this product is approximately €1000 which is well above average considering the product’s current available options and the fact that cost is a driving factor in the automotive industry.

## 2.4 HUDWAY Glass

![Figure 11: HUDWAY Glass][3]

It is a universal accessory which transforms a smartphone into a HUD independent of the car type. The major focus behind this product is to build an affordable product to improve driver’s safety and comfort mainly in the field of navigation. A typical setup of a HUDWAY Glass is depicted through figure11.
2.4.1 Implementation

The most important component is the glass. It is made up of a lightweight and sturdy plastic. The glass has an aspheric shape, which renders a 20% larger image while at the same time minimize spherical and chromatic aberrations. In lay man’s terms despite of the curved glass one can still see the road as it is, without any distortions. There is also a thin multilayered glass coating, its objective is to enhance the reflective properties of the glass and preserve its transparency, to avoid any sort of obstruction of the road.

![Figure 12: Available Adjustments][3]

To cater to the different types of vehicles, in which factors like dashboards of different shapes and sizes, difference in the driver’s position is a challenge. Their team has come up with two different mounts – one compact, the other – adjustable, which allows a wide angle view. There is a magnet in both the mounts which holds the mobile firmly, as well as one can detach it from the dashboard easily[9], [3].
2.4.2 Characteristics

It reflects a HUD based application to the glass and renders the display it in driver’s line of sight.

The speedometer is generic, it displays the current speed and give warnings when driver approaches the speed limits.

A connection with its application facilitate the driver with navigation functionality.

2.4.3 Limitations

- Driver assistance functionalities like LDW, ACC etc are not notified on the display.

- The screen, on which the display is rendered, is susceptible to bright light. One is not able to see display clearly during day time.

- The cradle which holds the phone may slip off the dashboard in case of a road bump

- The applications currently available for HUDWAY Glass are not working as expected, a number of user reviews from Amazon state that, it becomes slow during the real time scenario, which is a failure in case of tracking the current speed.

2.5 BMW HUD

The research from BMW in the field of HUDs is challenging to find, as it is in the development phase. The topics discussed below is an overview available on the BMW website[10].
2.5.1 Implementation

The working of BMW HUD is quite simple. It is depicted with the help of the image mentioned below. A TFT display with a very strong backlight generates the image, and the mirror produces the virtual image which gets displayed in the vision of the driver. The projection appears at a distance of about 2 meters over the motor canopy edge.

![Figure 13: BMW HUD Implementation](image)

2.5.2 Characteristics

The HUD offered by BMW is a full-colour display in models of new generation, it simply helps the driver to notice the different signals easily. For example the speed limits and control signals. The HUD can be configured by the driver.

A new generation of the pedestrian detection have been introduced, which proved to be sharper and faster to recognize by the driver and the reaction
time is shorter.

The BMW HUD can be conveniently positioned horizontally and vertically with the iDrive Controller for optimum display.

With extended image rotation, the BMW HUD offers individual adjustment options. Thus enabling the driver to focus on essential things- the most important information and the roadway.

The warning messages of the driver’s assistance are also displayed in the BMW HUD. These include for example, the person recognition of BMW Night Vision or the collision warnings.

2.5.3 Limitations

The HUDs are highly specific to the car models; the lower class models have a HUD with minimal options and a small display as compared to the higher end luxury segments. The price is to be estimated between €2000-3000.

2.6 Summary

<table>
<thead>
<tr>
<th>Table 2: HUD Comparision</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Functionality</strong></td>
</tr>
<tr>
<td>Driver Assistance Function</td>
</tr>
<tr>
<td>Performance</td>
</tr>
<tr>
<td>Hardware and Software</td>
</tr>
<tr>
<td>Cost</td>
</tr>
</tbody>
</table>

From the above stated scenarios, it could be observed that available products in the market concerning the AR-HUD technology today have a huge scope for optimization pertaining to their cost efficiency, technology and use cases. Hence the application has a vision to provide the driver assistance
functionalities using a smartphone and render these functionalities in AR on the windscreen.
3 Approach

The ideal vision of the thesis is depicted in this chapter. It discusses a narrow down process for the selected technologies to achieve the goals. Initially a brief introduction to all the technologies is mentioned and later the technologies which are chosen, are compared to depict their advantages.

3.1 Technologies used for developing a camera based mobile application

3.1.1 Image processing

It is a method to transform images by utilizing mathematical methods and digital signal processing. A possible input could be an image or a video frame from a camera and the output is either an image or other parameters related to the image[11].

3.1.2 OpenCV

It is an open-source library including a number of computer vision algorithms and machine learning functions to facilitate a standard infrastructure for computer vision applications and to fasten the implementation of human machine interface for commercial use[12]. Detailed descriptions of these concepts and their application have been presented in the next chapter.

3.2 Development Platform

One of the major challenges faced were to select the most appropriate development environment with respect to the efficiency and code complexity i.e. whether to develop an Android or iOS application to suite better with the image processing technologies. As we know that android development is JAVA oriented and OpenCV is natively written in C++. Java uses JNI which enables any Java program running in a JVM to call and be called by
native applications and libraries written in other languages such as C++[13][14]. But there is a number of issues face by following this technique:

3.2.1 Slow Execution

JNI layers wrap every native function, and calling through JNI is slower than working entirely in native code. In case of real time application the precompiled headers of C++ are handy to make the program execute efficiently.

3.2.2 Security Concerns

Native code is more susceptible to security issues like integer overflows, memory leaks etc.

3.2.3 Garbage Collection

Garbage collection in Java tends to stop all the CPU’s threads making it unsuitable for a real time application; one could overcome this problem by modifying the application to not produce garbage i.e. assigning some default values to avoid errors, by using a real time Java which is costly.

3.2.4 Performance

Writing a native code is preferable in cases where one wishes to implement a high performance algorithm, but it is not preferred when the application has an access to the phone camera, GPS and user interface.

3.3 Virtualization Analysis

Despite of the default development environment of MacOS-X is objective C, it was decided to implement the development of the product on an iOS platform because the environment provides the option to the developer to choose between C++ and Objective C[14].
Table 3: Probable Options for Virtualization

<table>
<thead>
<tr>
<th></th>
<th>Virtual Box</th>
<th>VM Ware</th>
<th>Parallels</th>
<th>Hyper-V</th>
</tr>
</thead>
<tbody>
<tr>
<td>64b Bit guest OS support</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>3D acceleration</td>
<td>Yes (Open GL 1.5)</td>
<td>No</td>
<td>Limited</td>
<td>Limited</td>
</tr>
<tr>
<td>Open Source</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>License Cost</td>
<td>Free</td>
<td>$80</td>
<td>$80</td>
<td>upto $3999</td>
</tr>
<tr>
<td>Mac OS X</td>
<td>20-30€</td>
<td>20-30€</td>
<td>20-30€</td>
<td>-</td>
</tr>
</tbody>
</table>

The next task was to search for a compatible hardware which is feasible for iOS development. Due to the cost factors, it was necessary to verify whether the development was possible via virtualization. It is a process which permits the system hardware to host multiple instances of various operating systems at the same time, in our case to virtualize a Mac OS X using a Windows system. Below mentioned is a detailed analysis of the probable ways of virtualizing on a Windows host[15].

3.3.1 Issues

- The virtualization is legal only if the Virtual Machine is running on Apple hardware[16].
- Mac OS X guests can only run on certain host hardware.
- Virtual Box does not provide Guest Additions for Mac OS X at this time.
- The graphics resolution currently defaults to 1024x768 as Mac OS X falls back to the built-in EFI display support.
- Mac OS X guests only work with one CPU assigned to the VM. Support for SMP will be provided in a future release.
- Basic versions of the Mac OS X and system experience guest hangs.
Based on this research and a formal discussion with the supervisor, it was then decided to implement the development of the application on legitimate Apple hardware.

3.4 Determining the position for the smart phone on the dashboard

One of the goals of the thesis is a perspective vision between the driver and the rendered display. In order to achieve this it was necessary to deduce a position for the smartphone so that the camera could get the maximum field of view, of the road ahead and the driver on the other side. Also the screen of the smartphone was supposed to be facing upwards, so that the display is reflected on the windscreen. But, it is impossible to get a FOV of $180^\circ$ using just one camera of the smartphone, when the phone is positioned on the dashboard.

3.5 Selection of the Lens

A lens which provides a FOV that covers the road ahead and the driver’s side was needed. This was one of the challenging parts because the algorithm to be implemented was completely dependent on the type of the lens which is selected. The main reason is that the image processing algorithms are highly specific to the type of input data being fed; in this case the outputs given by different lenses differed drastically in terms of shapes and sizes. Below mentioned are the different types of lenses which have been tested and eventually come up with the result.

3.5.1 Fisheye lens

It is a lens which produces an angle of usually $180^\circ$, there are also some special fisheye lens which produce an angle up to $235^\circ$ it is commonly known
as a super fisheye lens. They application is in the field of landscape, sport and surveillance cameras[17].

Types: There are two types of fisheye lenses as mentioned below, both produce a different effect.

**Full-Frame Fisheye Lenses:** It captures a rectangular image with the FOV of 180° along its diagonal. The vertical and the horizontal scopes of this lens are 100° and 150° respectively. They do not suffer from the problem of black edges like the circular fisheye lens.

**Issue:** As mentioned earlier a minimum of 180° FOV along the axes is necessary for the product, so this option was naturally ruled out without even experimenting.

**Circular Fisheye:** It is the one which captures a full of 180° degree view in all directions. This results in a circular image. The edges of the frame are
black, which is a typical problem for all the circular fisheye lenses. In effect this reduces the FOV.

**Issue:** To be rest assured instead of using 180° degree view fisheye a super fisheye lens was used which results a circular image with a FOV of 235°. Due to the black edge effect, the FOV of the lens was effectively reduced to less than 170°, practically making it impossible to use for further processing. Figure14 shows the lens output.

### 3.5.2 Dot/Kogeto 360°

As its name states, it is a lens which results into a 360° panoramic image or video in real time. It is a product by Kickstarter which is built exclusively as an iPhone accessory with their application. Figure15 shows the lens output.

![Figure 15: Kogeto 360°](image)
3.5.3 Issue

This product differed drastically to its definitions; the images which were taken were severely distorted. Despite having a 360° FOV it became impossible to use this lens.

3.5.4 Bubblescope

![Bubblescope Image]

Figure 16: Bubblescope

This lens is similar to the Kogeto 360°; it produces a 360° view. When used without its application i.e. BubblePix, it produces a donut shaped image as shown in figure. As one could clearly observe that this image is less distorted as compared to the other lenses. Hence, based on the results shown in figure16 it was decided to continue the research work with this lens.
3.6 Conceptualization

Based on the following points the concept is constructed to proceed with the thesis work.

3.6.1 iOS Development

iOS is an operating system for modular devices which are developed by Apple for example, iPhone, iPod Touch, and Apple TV. It is inspired by the OS X[18]. The SDK versions vary from 1.0-10.2. iOS 10.2 being the latest. A few important components needed for the development are explained in the figure.

**Interface Builder**  It is a tool which acquaints the developer with various UI elements which eases the creation of UI interface. One just needs to simply drag and drop the elements in the UI view.[18]. It is shown in figure17.

**iOS Simulator**  The iOS simulation can be implemented with two devices, iPad and iPhone with all their versions. It is useful to modify and visualize the application in terms of its width and height. Versions include iPhone5, 5s, 6, 6s and 7. With an iPhone it was possible to emulate the application.

**Framework**  The user interface framework provided by Apple for the application development of products like iPhone, iPad and iTouch is called Cocoa Touch. It is written in Objective C and based on Mac OS X. On the basis of a software architecture known as Model-View-Controller architecture; Cocoa Touch was developed. Cocoa Touch makes the high level application programming interfaces available for animation, networking and giving an appearance and behavior to the developed applications using less code[18].

**Software Architecture**  The MVC design pattern shown in figure18 allocates objects which perform the role of a model, view, or controller. Along
Figure 17: Interface Builder

with their roles, the communication protocols are also set by the architecture. All these are objects are classified by abstract boundaries and intercommunication between the objects happens through the boundaries. For
a well-designed Cocoa application one has to follow the MVC pattern[19]. The major advantage of this architecture is the reusability of the objects and interfaces to be well defined. Applications built with this architecture have a good extensibility than other applications. Below explained, are the components of the MVC pattern in brief.

**Model Objects** These objects gather the data pertaining to the application and define the logic and communication that manages and process that data. For example, a model object could represent a name in the address book. A model object can have to single and to multiple relationships with its other counterparts. The data which consist the permanent state of the application should stay in the model objects after the data is loaded in the application. As these objects have knowledge about a specific domain, they can be reused in similar problem domains. A model object is not allowed to have any kind of connection with the view object which shows its data and permits the user to modify that piece of information.
**Communication**  The communication between a view and a model object happens indirectly through the controller. The user may interact with the view object via actions, they can create or modify data, and these actions are communicated to the model for updating itself via the controller. When a model objects gets modified, it notifies a controller object which then updates the relevant view objects.

**View Objects**  It is everything that a user can see. It can react to the user actions. For example when one presses the button on the screen it might change its color, or might get a temporary depressed appearance depending on the user interface that is built. The purpose of this object is to represent the data from the specified model objects and also permit the modification of that data. As there is no domain specific information in a view object, its class can be reused for the logic of model and controller objects from a different application.

**Communication**  A view object communicates with the model object via a controller. It mainly transmits the modifications initiated by the user for example, when some numbers added in the text field, it goes through the controller and objects to the application’s model objects.

**Controller Objects**  A controller object acts as a mediator between one or more of an application’s model and view objects. Controller is an entity through which the model learns about the modification which are done in view objects and vice versa via a delegation pattern. The setup and synchronization of an application can be performed by a controller.

**Communication**  A controller object acknowledges the user actions made in the view objects and transmits the information in the form of the modifications to the model objects. After this when the model object up-
dates itself, the controller communicates this updated data from the model to the view object in order to display it.

3.7 Camera based Object Detection

Figure 19: Object Detection Techniques

All the tasks are related to the techniques of object detection based on camera. Thus a detailed overview about the techniques and concepts of object detection are elaborated below[20], [21], [22].

3.7.1 Feature based Detection

A feature based algorithm involves selecting regions of interest, encapsulating them into higher level features and comparing them with the next frame. The images could also be modified to another space for handling the modifications in picture intensity, size and orientation. Object detection and recognition then can be modified into a graph matching problem. Feature based detection could be further classified as follows
Shape-based Approach: This approach is one of the hardest due to the complexity of classifying the objects of interest in the images. To detect the border of an object, an image might have to be preprocessed, and this algorithm depends on the application. Different samples such as cars, animals and trees require different algorithms. For more complicated scenarios, noise reduction and scale invariant transformations could be preferred. In the case where the object is detected and tracked, its peripherals could be found by edge or boundary detection algorithms. The approach becomes complex when the scene consists of objects with occlusions and shading.

Color based Approach: When compared to other generic features of an image in a video frame, color is relatively constant and easily acquirable. Even though it is not the best detection technique, it is preferred for its low computational cost. This approach is based on tracking areas of similar normalized color from frame to frame. These areas are defined well within the scope of the object which is supposed to be tracked.

3.7.2 Template based Object Detection

If a model explaining an object in particular is available, then the detection becomes a mere process of feature comparisons between the model and the image frame under analysis. These comparisons with an exact match are often costly and the quality of the matching output depends on the details and the degree of precision of the object model. Template based detection could be further classified as follows.

Fixed Template Matching: This approach is useful when the shapes of the objects do not alter with respect to the viewing angle of the camera. This method is further divided into image subtraction and correlation.
Deformable Template Matching: This approach is favorable in the scenarios where the objects alter due to the rigid and nonrigid deformations. The major causes for these variations are either the object deformation or just by a different viewing angle from the camera.

3.8 Application Functionalities

The application functionalities are explained with the help of an activity diagram which depicts the concept of the set up built for evaluating the outputs.

3.8.1 Camera based Lane Detection

For an HUD to display the notification in AR, lane detection becomes a reference for positioning the display elements in between the trajectory on your lane and the driver for example on a two way highway one cannot show an arrow on the opposite lane for directions, that will obviously confuse
the driver. One more thing worth considering was its dynamic behavior for example it is useless to show a turn notification after missing the turn already.

3.8.2 Camera based Face Detection
It helps to determine the position of the driver with respect to the HUD. The HUD is supposed to render the display in the front of the driver. Hence this behavior cannot be achieved without face detection of the driver.

3.8.3 Perspective Vision
The algorithm for perspective vision would work in such a manner, that the HUD repositions itself to get an optimal intensity display for the driver. It can also be used as a warning mechanism for example when the driver is not within the range, an acoustic or visual warning in the form of a beep or continuous blinking of flashlight can be given by the smartphone.

3.9 Task Classification
Task classification was majorly dependent on the application functionalities. Based on an extensive research in those domains the tasks were classified as following

3.9.1 Development framework for MacOSX
The IDE provided for MacOSX is Xcode, which is common to every version of the OS.

3.9.2 Programming language
iOS application development could be done via Objective C, C++ and Swift which is lately introduced. OpenCV is written in C++; so it was an easy choice for me to use C++ for development[14].
Figure 21: Application Behavior
3.9.3 User interface

The goal for the UI was well defined i.e. a single view application which enables the user to start the application with a button press. The priority was to enable the application with the required functionalities.

3.10 Summary

This chapter gave us a brief outlook about the strategy that was followed to implement the tasks. Initially in every section a generalized process based on available technologies is mentioned. Towards the end of every section the process becomes specific with a fewer technologies as compared to the earlier state, the reasons for this specification are also presented.
4 Prerequisites

4.1 Tools and Technologies

4.1.1 Hardware

iPhone 5s

- OS: iOS 7
- Processor: dual-core 1.3GHz Cyclone processor with a PowerVR G6430 GPU
- Camera: 8 mega pixel rear camera with the recording capacity of 1080p@30fps

MacMini

- OS: El Capitan
- Processor: Intel Core i5, 2.6 GHz Dual-Core
- RAM: 8GB
- Graphics: Intel iris

Bubblescope: We have read about this lens in brief in the previous chapter

4.1.2 Software

- XCode
- OpenCV
- Objective C and C++
- XML
4.2 Image Processing

A digital image is a representation of an image as a finite set of digital values which are known as pixels. These pixels characterize the gray levels, intensities, heights, colors etc. Digital image processing modifies digital images through a digital computer. Its focus is to develop a computer system which could process an image in a desired manner. An image is the input to the system which processes it by some algorithms, and gives an output in the form of an image[23]. The most common example is snapchat. This application is used widely for processing images.

4.3 Image preprocessing

Some operations are needed to be performed on an image before advancing to a major image processing algorithm. The major reasons which came across to carry out these operations are as follows.

4.3.1 Noise Reduction

Certain images which we receive as raw input have distortions in it, due to the lens defects, instability while capturing the image or video in a dynamic condition, poor light scenarios, blurriness etc. This is termed as image noise.
It becomes necessary to reduce this noise as much as possible, in order to use 
the image for further processing.

4.3.2 Compatibility

Images can be received from different sources for example a smartphone, web-
cam; digital camera, etc. have different sizes and formats. But some image 
processing algorithms require data in a particular size and format for example 
a color image in RGB color space is a 3 channeled image, but certain algo-
rithms need a single channel i.e. a grayscale image as their input. Hence one 
must take care of such issues before going further with the implementation.

4.4 Image Container

Image can be acquired digitally in many different ways i.e. digital cameras, 
scanners, web cameras, computed tomography and magnetic resonance are 
some of the significant methods. The digital devices interpret each image as 
a set of numerical value which is reduced to numerical matrices and other 
data describing the matrix itself. The focus of OpenCV is to process and 
manage this data[11].

4.5 OpenCV

OpenCV is built around a C++ interface, to store the image in the memory 
it uses a C++ structure call Mat. This class comprises of two data parts: 
a matrix pointer containing the pixel values and a header to the matrix 
containing the size and the storage method for that matrix. The size of the 
header is always constant; the size of the matrix may vary with the images 
of different quality and sizes[11].

OpenCV is a library for image processing; hence one could not overlook 
the issue of overloading the code and slowing down the execution with mul-
tiple image matrices. To tackle this issue OpenCV uses a reference counting
mechanism. In this case the header plays a major role, the matrix can be shared between two instances referencing to the same address. The copy operator just copies the headers and pointer to the matrix instead of the whole data. In order to delete the multiple objects when they are not needed, the same mechanism comes into picture. Copying a header results in a raised counter and vice versa. When this counter reaches zero the matrix is released.

### 4.6 Morphological Operations

These are generic operations which are based on the image geometry. It is usually done on binary images, with two inputs, one is the source image and the other is a structuring element which determines the type of operation. The two generic operators are Dilation and Erosion; other operations like Opening and Closing have been derived from them\[11\].

#### 4.6.1 Erosion

Its property is to dissolve the image. The structuring element slides through the image. A pixel from the source image may be a 1 or 0 is only considered 1 when all the pixels under the structuring element is 1; if not then we could say that the image is eroded. The result of this operation is a redefined boundary; the pixels near the boundary will be discarded depending upon the size of the structuring element. It removes minor white noise and detaches the connected objects, gives a well defined look to the image.

#### 4.6.2 Dilation

It behaves exactly in the opposite manner to that of erosion. So in order to approximate a boundary or increase the thickness of the foreground object at least ‘1’ pixel under the structuring element has to be ‘1’. It increases the surface area of the object, to merge the broken parts of an object.
4.6.3 Opening

To perform opening; erosion is done after dilation. It is useful is removing small noises from the image.

4.6.4 Closing

Closing is the complement of Opening. It is nothing but, dilation after erosion. It is useful for closing small holes inside the foreground objects or noise on it.

4.7 Color Formats

<table>
<thead>
<tr>
<th>Column 0</th>
<th>Column 1</th>
<th>Column ...</th>
<th>Column m</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row 0</td>
<td>0,0</td>
<td>0,0</td>
<td>0,1</td>
</tr>
<tr>
<td>Row 1</td>
<td>1,0</td>
<td>1,0</td>
<td>1,1</td>
</tr>
<tr>
<td>Row ...</td>
<td>...0</td>
<td>...0</td>
<td>...1</td>
</tr>
<tr>
<td>Row n</td>
<td>n,0</td>
<td>n,0</td>
<td>n,1</td>
</tr>
</tbody>
</table>

Figure 23: BGR Color Space

4.7.1 BGR

This is the default color format used by OpenCV, because earlier this format was famous among the camera companies and software developers. For example in Windows, when giving the value for color by COLORREF the BGR format was used 0x00bbgrr.

In BGR blue occupies the most significant area, then comes green and red is the least significant one. For example: The hex code for pure blue in BGR is #FF0000, where FF represents the blue area and the 0s represent the red and green areas[11].

44
4.7.2 HSV

It is important to mention this color space, because majority of the operations based on thresholding and color segmentation require an image in HSV color format. “HUE” is the characteristic of a color by which we could name it as “green” or “red”. “Value” is a synonym for “lightness,” this characteristic classifies the color to some shade of gray, white and black being the extremes. Saturation depicts the difference of every color with respect to gray of the same lightness. Zero saturation indicates that the hue is also zero. It is a normalized color space[11].

The figure depicts the representation of HSV space in the form of a hex cone. Every cross section comprises a hexagon. The vertices of the cone represent colors magenta, red, blue, green, cyan, and yellow. A color in this
space has to be defined by stating a hue angle, the chroma level and the value level.

### 4.7.3 Gray Scale

This color space is one of the most significant color spaces in image processing. Almost all the image processing algorithms need a gray scale image as an input. This color space is scaled over different shades of gray, with white and black colors as the lightest and the darkest extremes respectively. The different shades of gray are represented by equal brightness levels of the three primary colors i.e. red, green and blue. The shade of gray color depends on the number which represents the brightness levels. For example black is represented as \( R = G = B = 0 \), and white is represented as \( R = G = B = 255 \). It is single channel in nature.

### 4.7.4 Binary

The term is self-explanatory; it represents an image which consists of only two possible values for pixels in an image. Pixels in a binary image could either be black or white. Hence it is also called as a bi-level or two level or bi-channel image.

### 4.8 Geometric Transformations of Image

Input being a video, feature extraction without any image processing operations was a challenge. These operations proved to be handy to extract the desired image regions for further processing[11].

#### 4.8.1 Affine Transform

This transformation is a matrix multiplication followed by a vector addition. It can be used to perform Rotations, Scaling and Translations of an image.
4.8.2 Perspective Transform

This operation is used to get a bird’s eye view; it works on the same principle as that of the human vision and the camera. For example, when human eyes see things from a short distance, they seem to get bigger compared to the things from a long distance. It could be defined through four pairs of points. By dragging these coordinates, the perspective projection can be managed, and the image is placed at the desired location.

4.8.3 Thresholding

It is simple classification process. It forms segments within an image based on a threshold. The threshold in this case is based on variation of the pixel intensity between the different object pixels. Once the classification is done and the important pixels are extracted, we can define them with a desired value to identify them according to our needs.

Simple Thresholding   It the value of the pixel is greater than the threshold, then it is associated with one value, else with another. This operation has three arguments which need to be satisfied; the image should be a grayscale image, the threshold value based on which the pixel values are classified and maxVal represents the value which is given when the pixel value is more than the threshold value. The Different styles of thresholding are Binary, Binary inverse, Trunc, Tozero and Tozero inverse which are depicted in the image below.

Otsu’s Binarization   An image whose histogram has two peaks is called a bimodal image. In such an image, we could take a value between the two histograms as our threshold, we need to follow a trial and error method to optimize this value. Otsu does the same; it automatically approximates a threshold value for the bimodal image from the image histogram. As shown
in the figure below, it does not give accurate results if the image is not bimodal.

4.9  Image Smoothing

It is achieved by convolving an image with a kernel of a low pass filter. It removes noise and edges which are high frequency in nature. There are several ways of smoothing an image, but the one which were found relevant to the task has been mentioned.

Instead of the conventional method of using a box filter, in this case a Gaussian kernel is used. The programmer has to specify the height and width of the kernel, which should be a positive and odd number. It removes the Gaussian noise from the image. The result of this filter has been shown below.
4.10 OpenCV Algorithms

4.10.1 Canny Edge Detection

It is used to detect the edges in an image. It is a multistage algorithm. It is one of the most optimal algorithms to detect the edges. Shown below is a flowchart of Canny Edge Detection and it has been explained later[11].

Algorithm Explanation

- Gray-scale image: This image is converted to a single channel gray scale image.

- Threshold output: This step decides the lower and upper threshold within the range of 0-255 for the algorithm to search for the edges in
further steps. It can be controlled dynamically by implementing a track bar for these limits. It helps us find the potential edges.

- Gaussian Blur: To smooth the image in order to remove noise. Usually
the kernel is of size 3.

- **Canny Edge Detector**: Canny, is a function provided of OpenCV processes the image for finding the edges where it takes the lower and upper thresholds, kernel size, an input image, and a Mat container to store the output.

- **Display result**: The result can be displayed in two ways, first we create a matrix of the same type and size that of our source image then we fill this image with zeros i.e. the image is completely black later by using another OpenCV function copyTo only the areas of the image that are identified as edges can be mapped, second is to display the edges on the source image itself.

### 4.10.2 Contour Detection

They can be defined as curves of same color and intensity which are spread along the boundary in a continuous manner, thereby simplifying the process of shape analysis and object detection and recognition. For better detection results a binary image is preferred, because the intensities are well distinguished. The figure below shows the process of detecting a contour[11].

**Algorithm Explanation**

- **Canny Edge Detector**: It can be observed that initially the edges are extracted by using the previous algorithm of canny edge detection.

- **Find Contours**: This is an OpenCV function which retrieves the contours from the image. It takes a binary image as input; the detected contours are stored in a vector of points. Hierarchy is an argument in this function which is an optional vector to store the order in which the contours are detected; this is used when a particular contour has to be selected from a set of contours. The mode argument allows the
Figure 28: Contour Detection
contour to be stored in different forms for example a list, tree, external and flood fill. The method and offset arguments are used to approximate a contour to a relative shape and in cases of an ROI analysis respectively.

- **Draw Contours and select color**: This function takes the vector of contours formed in the previous step and draws a contour of the specified color and thickness.

- **Display output**: The contours can be displayed in a separately created image as explained in the previous example or it can simply be superimposed on the source image as shown in the figure below.

### 4.10.3 Hough Line Transform

This algorithm is an extension for Hough transform. It is used to detect lines within an image. The figure below shows the process of detecting a line[11].

**Algorithm Explanation**

- **Canny Edge Detector**: It can be observed that initially we extract the edges by using the previous algorithm of canny edge detection.

- **Hough Line Function**: This is OpenCV function to detect lines, it takes 8 bit, single channel binary image as input, and gives an output in the form of a two-point vector of rho and theta. Rho is distance from the coordinate origin and theta is the line rotation angle in radians. A threshold in the form of angle can be given according to the application to detect specific lines for example only horizontal or only vertical lines.

- **Draw line**: Once all the possible lines are detected and stored in a vector, a line function is called in a loop to draw the lines by using rho and theta.
Figure 29: Hough Line Transform
• Display Result: The lines are displayed in a specific color on the source image.

4.10.4 Watershed Algorithm

It is a form of segmentation on the basis of color. It finds the objects to which a pixel belongs. This also helps us separating background from the foreground objects. The process of this color based segmentation is explained below[11].

Algorithm Explanation

• Source image: It can be a multichannel and color image.

• Otsu’s Binarization: The segments in our image are touching each other. Hence an approximation is needed instead of thresholding. For this we use Otsu’s Binarization. It is briefly explained in the previous subsection.

• Opening: To remove the white noises from the image we use Opening.

• Dilation: To remove the small holes from the image we use Dilation.

• Distant Transform: At this point, we know precisely that the areas near the center are the foreground objects and the areas near the boundary are the boundary. Only thing which is unknown is the boundary position. Hence we use Distant Transform. It deduces the object boundaries.

• Find Markers: Markers is an array of same size as that of original image, but with int32 data type. Now we clearly know the foreground and the background objects. The markers label the unknown objects as zero and the identified objects are labeled with integers starting from 1. The background objects are marked with just one, so that they are distinguished from the unknown objects.
Figure 30: Watershed Algorithm
• Assign: A set of colors can be specified for the output.

• Display output: The marker image can be displayed or also be superimposed on the image.

4.10.5 ROI

Image processing algorithms are computationally heavy. ROI selects an area, and the algorithm works only on this area. This speeds up the execution of the algorithm. The process of drawing an ROI is explained below[11].

![ROI Diagram]

Figure 31: ROI
Algorithm Explanation

- Source image: It can be a multichannel and color image.

- Construct Coordinate: It depends on the programmer how to define the area of ROI. One can draw a rectangle or circle or ellipse as ROI. Hence the coordinates have to be constructed accordingly. There are functions provided by OpenCV to draw the shapes.

- Decide Parameters: These parameters describe the position where one wants to position the ROI in the Image. Other parameters are the color of ROI, thickness, line type and shift type.

- Display ROI: Lastly we can draw the ROI on our image. One can observe that the image processing algorithm is applicable only in this region.
5 Implementation

5.1 ROI determination

The input image from Bubblepix has some distortions, as it is not a standard lens for image processing unlike fisheye which is used extensively in security purposes. OpenCV does not have a calibration process for this lens. Hence for feature extractions it was important to initially filter particular regions from the image which can be used for further processing.

In order to achieve the tasks as per their classification mentioned in the
previous chapter, the input has been processed in the following manner. Two ROIs are extracted from the source image, one in which the lanes ahead are visible and the other in which the driver’s face is visible. It is shown in the figures 33, 34, 35 and 36 respectively. By using perspective transform the area containing the site of the road ahead is extracted.

Figure 33: Perspective Transform

After this area is extracted it is sharpened by the addweighted operation over a Gaussian filtered image. This operation sharpens the contour boundaries which makes it easier for further processing.

Similarly the areas of interest are extracted for the face detection i.e from the driver’s side.
5.2 ROI Sharpening

The sharpening of the images was achieved by the following procedure[24].

5.2.1 Algorithm Explanation

- Input image: The input image is a donut shaped color image which is shown above.

- ROI: The areas where the lanes and driver’s face are depicted in the image are extracted from the input image.
Figure 35: Perspective Image

Figure 36: Sharpened Image
• Perspective Transform: This operation rotates and scales the ROI in a manner which is preferred for further image processing.

• Gaussian Blur: It removes the high frequency noises from the image
and gives it a smooth finish.

- Addweighted: It is an OpenCV function, the input image is subtracted from the blurred image. As a result we get a relatively sharpened image. The result is shown above.

5.3 Implemented Algorithms for Lane Detection

This subsection explains the algorithms which have been implemented based on the current technologies to pursue similar tasks for different applications. These algorithms depict the preferable ways to detect lane using a smartphone camera. Also mentioned below are the issues due to which the use of these algorithms had to be discontinued[25], [26], [27].

5.3.1 Vanishing point

Algorithm Explanation

- Step1: From the driver’s view the lanes ahead appear to be intersecting at a particular point instead of appearing to be straight, this is depicted by the image shown below. The algorithm determines this intersection, called as the vanishing point.

- step2: The vanishing point includes the adjacent lines as well i.e. the neighboring road lanes. So, the arguments in the Hough line transform has to be set to detect the desired lines based on their angles.

- Step3: Now the information regarding the angles and the intersection point can be used for further processing.

Issues In the scenario when the lens was not attached to the phone this algorithm was optimum with respect to the results of the detected lanes as compared to all the other algorithms. The figure 39 shows the lane detection using this algorithm without the lens.
In the case when a lens was attached to the camera the straight lines appeared to be curved and Hough line transform cannot be used to detect curved lines. Hence this algorithm had to be ruled out.

Inverse perspective mapping is another algorithm which requires the functionality of Hough line transform also had to be ruled out because of the same issue.

5.3.2 Detection via integrating contour and watershed algorithms

Algorithm Explanation
- The sharpened image is given as the input to the watershed algorithm which processes the image to detect foreground objects from the image.
- The marker image displays different intensities prominently. Hence it
processed further to the contour detection algorithm, here the desired contour can be extracted from a group by using the contour hierarchy function. This extracted contour consists the area between the lanes in which the car moves.

- Thus lanes can be easily extracted from this contour.

**Issues:** It computes the contours via approximation of the color and the transition phase of the contour is misjudging for the observer for example from figure41 one can see that the road trajectory has been merged with
other surroundings of the road.

Figure 41: Watershed Marker Image

5.3.3 Detection via contour and convex hull

Algorithm Explanation

- The initial step is to apply the contour detection algorithm on the ROI.

- A function to find the largest area is then implemented by using the contour image moments and via mathematical operations to find the most biggest contour from the image.

- This extracted area is approximated using the morphological operations by opening operations to define the lanes more specifically.
• By using the convex hull function which is a contour approximation method the lanes can be highlighted.

**Issues:** The discontinuities between the lanes and the road boundaries results in integrating all the small contours as largest contour. The disintegration of these contours based on contour hierarchy becomes impossible because the road scene changes every second. Hence this algorithm had to be ruled out.
5.3.4 Face Detection using Haar Cascades

It is an object detection technique proposed by Paul Viola and Michael Jones[28]. It is an approach based on machine learning. A trained cascade function is generated by using positive and negative images samples. Then this is used to detect objects in the given scenario. Positive samples are the images with the object to be detected and negative samples comprises of all the images without that object[29], [11].

Algorithm Explanation
Initially features from the given samples are extracted, for which the Haar features depicted in the figure are used. It is a convolution kernel, which gives the output as a single value obtained by a subtraction between the sum of pixels in white and black pixels in white and black rectangles respectively.

Such a process would lead to a huge computation load for example even a 24x24 window would result in more than 16000 features. It is necessary to eliminate the irrelevant features. Features based on certain prominent properties to detect a face for example observing the
figure below carefully it could be observed that the areas of the eyes are darker than the other parts of the faces and it is selected on the by comparing it with the bridge of the nose. This selection of features is done by Adaboost, it is a boosting algorithm which becomes more complex with every step[29].

- This selection is achieved by applying every feature on all training images. It will search for a threshold which will classify the faces as positive or negative. This threshold changes after every feature and the one with most negative scores is eliminated from the list.

- This means it is a series of weak classifiers which detect features at every level resulting into a strong final classifier. This learning of the classifier at every stage is called boosting.
• In opencv it is implemented by appending the classifiers files in xml or yaml format and using the function detectMultiscale. Once detected they can be displayed in the image simply by drawing a rectangle or a circle around the features.

**Issues:** Even though this algorithm was accurate for detecting the driver’s face; it is susceptible to different light conditions. Beyond a distance 1.5m the algorithm fails to detect the object. Figure 46 shows the output without the lens on the phone and the figure 47 shows the output with the lens on, where the ellipse flickers continuously to detect the correct object.
Figure 46: Detection without lens
5.4 Unified Algorithms

The algorithms mentioned above fail to work in the proposed conditions. Unifying these algorithms for the lane detection and face detection process overcomes the shortcomings as well as gives better outputs. A detailed description of these algorithms has been presented in this section.

5.4.1 Lane Detection

Algorithm Explanation

- ROI: The method mentioned previously generates Sharpened ROI.
Figure 48: Unified Algorithm for Lane Detection
• Color Thresholding: A filter which allows the shades of gray color from the ROI is implemented.

• Contour Detection: The basic contour detection algorithm is applied here. This algorithm detects all the contours gray color because of the thresholding which was done in the previous step.

• Contour Extraction: Once all the contours are detected, the contour
which contains the lanes needs to be extracted from it. Hence the contour with the largest area is extracted. It contains the trajectory of the road on which the cars moves.

![Figure 51: Largest Contour extracted](image)

- **Image Sharpening:** The image needs to be sharpened again so that the reflection of the trajectory is well defined and easily visible to the driver. Hence the image is sharpened by Gaussian blur and addweighted operations.

- **Color Background black:** From the driver’s perspective the whole image is not important, only the road trajectory has to be clearly visible. Hence the background object is painted black.

- **Color Foreground object:** White is the most brightest color visible to the driver as compared to all the other colors. The reflection of the contour should be bright, so that the driver can see it easily.
• Mirror Image: The image seen by the driver is not incident, it has to reflected from the windscreen. Hence the incident source of image, which in our case is the mobile screen has to be a mirror image.

5.4.2 Face Detection
Algorithm Explanation

- ROI: It is a sharpened image displaying the driver’s face. It is a color image.
• Color Segmentation: In this process all the shades of skin color are extracted. The color image is initially converted to its equivalent in HSV color space and then the shades are extracted, unlike thresholding where the color image is converted into a grayscale image.

• Contour Detection: A generic contour detection algorithm is implemented; it only detects the skin color contours due to the previous segmentation filter. This is implemented in a continuous loop to detect the contours in real time.

![Figure 55: Detected Contour](image)

• Give Warning: This step functions only when the driver is not aligned to the HUD display in front of him. So, in order to warn him a visual warning is given by blinking the flashlight of the smartphone until the driver appears to be in his spot.
6 Results and Evaluation

The results could be explained with the following images.

6.1 Results

![Experimental Setup-Front View](image1)

Figure 56: Experimental Setup-Front View

![Experimental Setup-Side View](image2)

Figure 57: Experimental Setup-Side View
6.1.1 Construction and Mounting

- The figures 56, 57, 58 depict a prototype which was built to simulate a dynamic scenario and verify the efficiency of the application.

- The laptop screen displays a sample video to verify the application functionality.

- The glass screen behaves as a windscreen of the car on which the HUD is going to be displayed.

- The board below acts as the dashboard of the car on which the smartphone with an attached lens is placed.

6.1.2 Working

Below mentioned are some photos which were taken during the detection. A black film was attached to the glass because it was challenging to capture the reflection with the display of the laptop on. But, the black film is not necessary while looking with a naked eye.
• A video is played on the laptop, and the application is started with a button press.

• The road trajectory is detected and a mirror image appears on the mobile screen.

• This image is reflected on the glass placed at an optimized angle for the driver to perceive the video at its optimum intensity with ease.

• On the other hand one can switch to the face detection algorithm with a button press.

• Now the algorithm searches for the driver’s face to be in a desired position.

• If it is detected then it functions as an endless loop, but in the condition where the face is not detected then a notification light blinks in the form the mobile flashlight.
Figure 59: Results(a)

Figure 60: Results(b)
Figure 61: Results(c)

Figure 62: Results(d)
6.2 Evaluation

The development process was not application specific, the procedures to achieve the tasks differed. After every predefined milestone the application was tested for its performance by trial and error approach using different ways to reach the same milestone and select the best fit. This approach proved to be very handy, because not only it assured of a working system after the completion of every milestone as well as was handy to optimize the result wherever possible. The scenarios in which the application was tested are mentioned below.

6.2.1 Vision

- To integrate the lane detection and face detection modules for implementing the perspective vision functionality.

- An AR-HUD which adjusts its display according to the gaze of the
6.2.2 Tasks achieved

- An AR-HUD which displays the trajectory of the road on which the car is moving in real time.
- Face detection of the driver.
- A visual warning mechanism in real time to detect whether the driver is in the FOV of the HUD.
- Proposed methodology for achieving the vision of the complete application.

6.2.3 Performance

Road with a multiple lanes: The algorithm is able to detect the trajectory of the lane on which the car is moving. Challenge in this scenario is during the lane change of the car. As the algorithm searches for the biggest contour in front, it gives some intermediate results for eg. it detects the sky as the road because at the instance of lane change the biggest contour is sky. But once the lane change has happened then the algorithm performs as specified.

Road with an empty single lane road: This is the most optimum scenario. It successfully detects the road ahead. Due to the wideness of the single lane, for most of the times the road is distinctively depicted as the biggest contour. Thus simplifying the process to detects the road trajectory accurately.

Roads with trees: The shadows of the trees change the shade of the road i.e. the gray color of the road becomes darker due to the tree shadows. The lane detection algorithm has a range for the shades of gray to be considered as a road, if the shadow is not in this range then it does not detect it. This
is a scenario only on a road densely covered with trees, otherwise in other
cases it detects the road.

Face Detection: It detects the face as a contour of the human skin color.
An exception in this process is the car with the interior of the same color.
Then even if the driver’s face is not properly aligned with the phone, it de-
tects the interior as the face and does not throw a warning.

Below mentioned are some scenarios where the application was not able
to detect the road.

Road during a bright sunlight: The approach is based on color threshold-
ing, so when the rays are incident on the road, they alter the color of that
region. Due to such discontinuities the environment other then road for eg.
pavements aside the road are detected as the trajectory.

Road during the night: This is a special case for which the algorithm has
to be altered. The shade of the road color at night is different than that of
the road during the day. Hence for detecting the road during night needs
alterations in the threshold values of the algorithm.

Table 4: Evaluation Summary

<table>
<thead>
<tr>
<th></th>
<th>Normal</th>
<th>Bright</th>
<th>Night</th>
<th>Rainy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple Lanes</td>
<td>✓</td>
<td>Limited</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Single Lane</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Road with Trees</td>
<td>✓</td>
<td>Limited</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Road during the rainy conditions: Due to the rains the input image is
severely distorted, which makes it impossible for extracting the features.
7 Conclusion

This thesis is an initial attempt to implement AR-HUD using a module device for a driver assistance function. Based on some similar products by Continental, BMW and HUDWAY. The application is implemented for an iPhone; this work mentions the generics of iOS development.

To facilitate an AR-HUD with perspective vision it was necessary to attach an external lens to the iPhone. This work followed a camera based approach to achieve the desired tasks, thus it revolves majorly around real time object detection. The various methodologies for object detection are analyzed and the task is functionally categorized into three major subtasks i.e. Lane detection, Face detection and perspective vision. Based on an extensive research the algorithms for achieving every subtask were specified; and they were tested for their efficiencies by building their application prototypes. OpenCV was integrated to the development process as a technology to pursue camera based real time object detection.

The algorithms were implemented in the working environment of the iPhone with the external lens attached. A Comparison based on the efficiency of their results, algorithms were unified to give an optimized output. The algorithms to detect the road trajectory and a visual warning system for determining the position of the driver have been implemented. An algorithm to facilitate perspective vision is proposed.
8 Future Scope

Narrowing the technology gap between the automotive and computing world is resulting into a standardized development in both the domains. This gives the research work a wide scope for its application in the field of driver assistance. Some of the important use cases are mentioned below.

Navigation: Applications like Offline maps and HERE maps have started introducing their HUDs software applications on the app store. They still lack in the AR domain, where as this is one of the generic functions of the application. Other navigation functionalities like tracking of Hotels, gas stations speed limits etc. can also be implemented. Hence a complete package in terms of real time navigation can be offered to the users via this application.

Driver Assistance: Along with the perspective vision of the AR-HUD display, an acoustic or visual warning system can be implemented, which will track the gaze of the driver and alert him in case if he is looking away from the road; similar to that of attention assist.

Distribute load of complex automotive systems: With time the car is becoming more functional, but in order to increase the functionality of a vehicle one cannot just merely add more components because of the monitory factors. Hence currently the trend of parallelism has emerged in the industry. It will exploit the current systems to cater to the requirements and perform efficiently, which in turn makes the design and implementation complex. Such applications hosted on native devices will reduce the complexity.

Free application: It is a free application, without any proprietary claims unlike BMW or continental. Also being a mobile application it can be implemented in any type and model of vehicle.
Costs: As mentioned earlier a navi pack with an HUD costs between 2000-3000€, the research is still ongoing to provide better functionalities in lower costs. When the current scenario is taken into account the option of a smartphone HUD is more feasible than a normal HUD.
References


