DETECTING DRIVABLE SURFACE
LANE DETECTION FOR SELF DRIVING CAR

Submitted in partial fulfillment of the requirements

for the award of the degree of

Bachelor of Technology

In

Computer Science & Engineering

Under the supervision of

Ms. Kavita Sheoran

By

Manoj Pandey (03215002714)

Akul Mehra (07615002714)

Ashu Goel (06815002714)

Harsh Nandan (00415007215)

To

Maharaja Surajmal Institute of Technology (*12 pts.*)

Affiliated to Guru Gobind Singh Indraprastha University

C-4, Janakpuri, New Delhi-58

CERTIFICATE

This is to certify that the project work done on *"Detecting Drivable Surface: Lane Detection For Self-Driving Car*" submitted to Maharaja Surajmal Institute of Technology, Janakpuri Delhi by Manoj Pandey (03215002714), Akul Mehra (07615002714), Ashu Goel (06815002714) and Harsh Nandan (00415007215) In partial fulfillment of the requirement for the award of degree of Bachelor of Technology, is a bonafide work carried out by him/her under my supervision and guidance. This project work is the original one and has not submitted anywhere else for any other degree.

Ms. Kavita Sheoran
(Project Guide)

Dr. Koyel Datta Gupta
(HOD, CSE MSIT)

ACKNOWLEDGEMENT

Team effort together with precious words of encouragement and guidance makes daunting tasks achievable. It is a pleasure to acknowledge the direct and implied help we have received at various stages in the task of developing the project. It would not have been possible to develop such a project without the furtherance on part of numerous individuals. We find it impossible to express our thanks to each one of them in words, for it seems too trivial when compare to the profound encouragement that they extended to us.

We are grateful to **Dr. Koyel Datta Gupta, *HOD(CSE)*,** for having given us opportunity to do this project, which was of great interest to us.

Our sincere thanks to **Ms. Kavita Sheoran** *(Project Guide)* for believing in us and providing motivation all through. Without her guidance this project would not be such a success.

At last we thank the almighty, who had given the strength to complete this project on time .

Finally we would like to thank our parents, all friends, and well wishers for their valuable help and encouragement throughout the project.

Manoj Pandey (03215002714)

Akul Mehra (07615002714)

Ashu Goel (06815002714)

Harsh Nandan (00415007215)

Abstract

Object recognition – technology in the field of computer vision for finding and identifying objects in an image or video sequence. Humans recognize a multitude of objects in images with little effort, despite the fact that the image of the objects may vary somewhat in different view points, in many different sizes and scales or even when they are translated or rotated. Objects can even be recognized when they are partially obstructed from view. This task is still a challenge for computer vision systems. Many approaches to the task have been implemented over multiple decades.

An autonomous car (also known as a driverless car, self-driving car, robotic car) and unmanned ground vehicle is a vehicle that is capable of sensing its environment and navigating without human input. Most of the accidents on the road occur due to the inattentiveness of the driver causing loss of lives and severe damage to vehicles. Self-driving cars are useful in avoiding accidents, enhances safety and improves the traffic conditions as it is the main purpose of vision-based driver assistance system of intelligent vehicles. Such systems have the ability to detect lane lines on the roads and keeps the car on track hence avoiding them to depart from the lane.

In this paper, with the use of computer vision and image processing libraries such as OpenCV, we propose an algorithm for detecting marks of road lane and road boundary for smart navigation of intelligent vehicles. Initially, color selection is done on images to select only those marks that represent road lanes and converted to grayscale. A region of interest is filtered to remove irrelevant objects from the image. Finally, algorithms such as Canny Edge Detection and Hough Transform are used to highlight the road lanes and boundaries. The experimental results obtained are tested across images and videos captured through cameras and LIDAR unit sensor on the top of a car. The results are highlighted road lanes on original images and videos.

TABLE OF CONTENTS

Chapter - 1

1. Introduction

1.1 Overview

Computer vision is an interdisciplinary field that deals with how computers can be made for gaining high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do.

Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information, e.g., in the forms of decisions. Understanding in this context means the transformation of visual images (the input of the retina) into descriptions of the world that can interface with other thought processes and elicit appropriate action. This image understanding can be seen as the disentangling of symbolic information from image data using models constructed with the aid of geometry, physics, statistics, and learning theory.

The classical problem in computer vision, image processing, and machine vision is that of determining whether or not the image data contains some specific object, feature, or activity. Different varieties of the recognition problem are described in the literature:

Object recognition (also called object classification) – one or several pre-specified or learned objects or object classes can be recognized, usually together with their 2D positions in the image or 3D poses in the scene. Blippar, Google Goggles and LikeThat provide stand-alone programs that illustrate this functionality.

Identification – an individual instance of an object is recognized. Examples include identification of a specific person's face or fingerprint, identification of handwritten digits, or identification of a specific vehicle.

Detection – the image data are scanned for a specific condition. Examples include detection of possible abnormal cells or tissues in medical images or detection of a vehicle in an automatic road toll system. Detection based on relatively simple and fast computations is sometimes used for finding smaller regions of interesting image data which can be further analyzed by more computationally demanding techniques to produce a correct interpretation.

1.2 About the Project

In this project, we have a set of images collected from dataset of roads images captured through the cameras and LIDAR sensors mounted on the top of self-driving car. Using these images of road lanes and boundaries, our motive for this project is detect road lanes and road boundaries, such that a self-driving car can track its movement and avoid crashes.

We propose an algorithm for detecting marks of road lane and road boundary for smart navigation of intelligent vehicles. Initially, color selection is done on images to select only those marks that represent road lanes and converted to grayscale. A region of interest is filtered to remove irrelevant objects from the image. Finally, algorithms such as Canny Edge Detection and Hough Transform are used to highlight the road lanes and boundaries. The experimental results obtained are tested across images and videos captured through cameras and LIDAR unit sensor on the top of a car. The results are highlighted road lanes on original images and videos.

1.3 Purpose

Our main purpose is the automation of the most common mode of medium to travel for people, driving. This will not only ensure the safety of people but also help us manage traffic and pollution in our environment, hence making our life simpler. Every year around 1.25 million people die as a result of road accidents. Around 20 to 50 million people suffer from non-fatal injuries, with many incurring a disability due to the result of their injury. Approximately 46% of the victims are due to driving cars. Most of these accidents can be avoided by using vision-based driving assistance system of intelligent vehicles or known as self-driving cars. According to Waymo (formerly called Google's Self Driving Car program), they stated: "It is pretty darn safe for people to ride in autonomous vehicles". Waymo has logged over 2 million miles on U.S. streets with fault in only one accident, making its cars by far the lowest at-fault rate of any driver class on the roads of U.S. This rate is about 10 times lower than our safest demographic of human drivers in the age group of 60-69 year olds and 40 times lower than learning drivers.

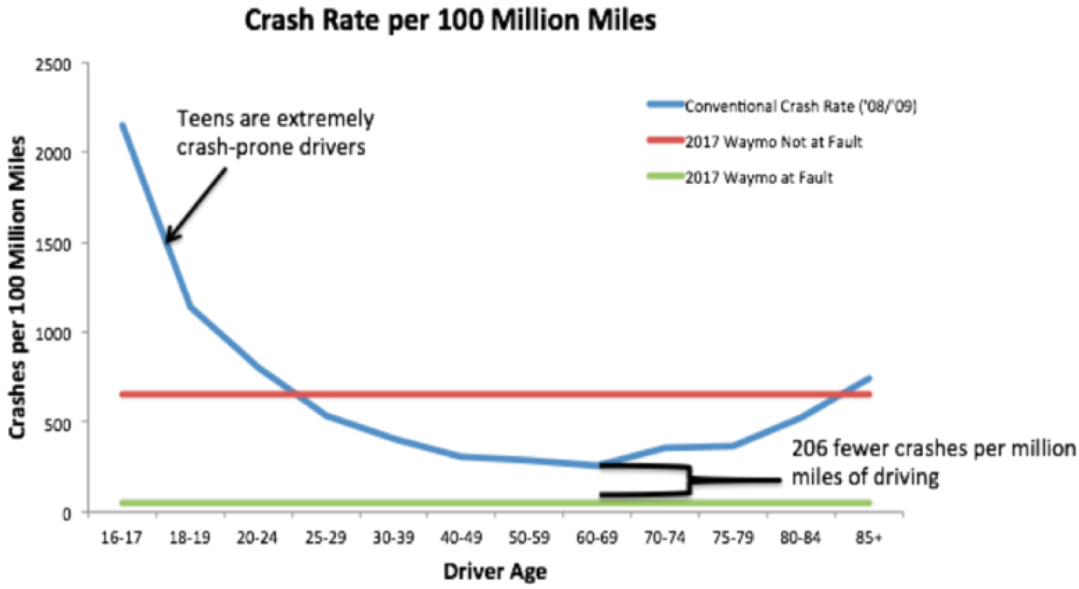

Fig 1.1: Crash Rate per 100 Million Miles

1.4 Scope

In this project, we have created an algorithm which can detect road lane lines and road boundaries through computer vision and image recognition. This algorithm can be used in many ways such as detecting where all road lanes are needed to be created and keeping a record of such locations, or finding locations of pot holes and detecting which roads are required for construction and maintainance. This algorithm can be used for making decisions about driving and how traffic and accidents can be avoided, making the world a better place.

1.5 Additional Functionality

The algorithm can be enhanced with multiple features for better decisions, such as:

- Detecting Pot Holes
- Detecting other vehicles
- Detecting Traffic Signs

Chapter - 2

2. Background

2.1 Definition

Computer vision is an interdisciplinary field that deals with how computers can be made for gaining high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do.

Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information, e.g., in the forms of decisions. Understanding in this context means the transformation of visual images (the input of the retina) into descriptions of the world that can interface with other thought processes and elicit appropriate action. This image understanding can be seen as the disentangling of symbolic information from image data using models constructed with the aid of geometry, physics, statistics, and learning theory.

As a scientific discipline, computer vision is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, or multi-dimensional data from a medical scanner. As a technological discipline, computer vision seeks to apply its theories and models for the construction of computer vision systems.

Sub-domains of computer vision include scene reconstruction, event detection, video tracking, object recognition, 3D pose estimation, learning, indexing, motion estimation, and image restoration.

2.2 History

In the late 1960s, computer vision began at universities that were pioneering artificial intelligence. It was meant to mimic the human visual system, as a stepping stone to endowing robots with intelligent behavior. In 1966, it was believed that this could be achieved through a summer project, by attaching a camera to a computer and having it "describe what it saw".

What distinguished computer vision from the prevalent field of digital image processing at that time was a desire to extract three-dimensional structure from images with the goal of achieving full scene understanding. Studies in the 1970s formed the early foundations for many of the computer vision algorithms that exist today, including extraction of edges from images, labeling of lines, non-polyhedral and polyhedral modeling, representation of objects as interconnections of smaller structures, optical flow, and motion estimation.

The next decade saw studies based on more rigorous mathematical analysis and quantitative aspects of computer vision. These include the concept of scale-space, the inference of shape from various cues such as shading, texture and focus, and contour models known as snakes. Researchers also realized that many of these mathematical concepts could be treated within the same optimization framework as regularization and Markov random fields. By the 1990s, some of the previous research topics became more active than the others. Research in projective 3-D reconstructions led to better understanding of camera calibration. With the advent of optimization methods for camera calibration, it was realized that a lot of the ideas were already explored in bundle adjustment theory from the field of photogrammetry. This led to methods for sparse 3-D reconstructions of scenes from multiple images. Progress was made on the dense stereo correspondence problem and further multi-view stereo techniques. At the same time, variations of graph cut were used to solve image segmentation. This decade also marked the first time statistical learning techniques were used in practice to recognize faces in images (see Eigenface). Toward the end of the 1990s, a significant change came about with the increased interaction between the fields of computer graphics and

computer vision. This included image-based rendering, image morphing, view interpolation, panoramic image stitching and early light-field rendering.

Recent work has seen the resurgence of feature-based methods, used in conjunction with machine learning techniques and complex optimization frameworks.

2.3 Evolution of Computer Vision

Areas of artificial intelligence deal with autonomous planning or deliberation for robotical systems to navigate through an environment. A detailed understanding of these environments is required to navigate through them. Information about the environment could be provided by a computer vision system, acting as a vision sensor and providing high-level information about the environment and the robot.

Artificial intelligence and computer vision share other topics such as pattern recognition and learning techniques. Consequently, computer vision is sometimes seen as a part of the artificial intelligence field or the computer science field in general.

Solid-state physics is another field that is closely related to computer vision. Most computer vision systems rely on image sensors, which detect electromagnetic radiation, which is typically in the form of either visible or infra-red light. The sensors are designed using quantum physics. The process by which light interacts with surfaces is explained using physics. Physics explains the behavior of optics which are a core part of most imaging systems. Sophisticated image sensors even require quantum mechanics to provide a complete understanding of the image formation process. Also, various measurement problems in physics can be addressed using computer vision, for example motion in fluids.

A third field which plays an important role is neurobiology, specifically the study of the biological vision system. Over the last century, there has been an extensive study of eyes, neurons, and the brain structures devoted to processing of visual stimuli in both humans and various animals. This has led to a coarse, yet complicated,

description of how "real" vision systems operate in order to solve certain vision related tasks. These results have led to a subfield within computer vision where artificial systems are designed to mimic the processing and behavior of biological systems, at different levels of complexity. Also, some of the learning-based methods developed within computer vision (e.g. neural net and deep learning based image and feature analysis and classification) have their background in biology.

Some strands of computer vision research are closely related to the study of biological vision – indeed, just as many strands of AI research are closely tied with research into human consciousness, and the use of stored knowledge to interpret, integrate and utilize visual information. The field of biological vision studies and models the physiological processes behind visual perception in humans and other animals. Computer vision, on the other hand, studies and describes the processes implemented in software and hardware behind artificial vision systems. Interdisciplinary exchange between biological and computer vision has proven fruitful for both fields.

Yet another field related to computer vision is signal processing. Many methods for processing of one-variable signals, typically temporal signals, can be extended in a natural way to processing of two-variable signals or multi-variable signals in computer vision. However, because of the specific nature of images there are many methods developed within computer vision which have no counterpart in processing of one-variable signals. Together with the multi-dimensionality of the signal, this defines a subfield in signal processing as a part of computer vision.

Beside the above-mentioned views on computer vision, many of the related research topics can also be studied from a purely mathematical point of view. For example, many methods in computer vision are based on statistics, optimization or geometry. Finally, a significant part of the field is devoted to the implementation aspect of computer vision; how existing methods can be realized in various combinations of software and hardware, or how these methods can be modified in order to gain processing speed without losing too much performance.

The fields most closely related to computer vision are image processing, image analysis and machine vision. There is a significant overlap in the range of techniques and applications that these cover. This implies that the basic techniques that are used and developed in these fields are similar, something which can be interpreted as there is only one field with different names. On the other hand, it appears to be necessary for research groups, scientific journals, conferences and companies to present or market themselves as belonging specifically to one of these fields and, hence, various characterizations which distinguish each of the fields from the others have been presented.

Computer graphics produces image data from 3D models, computer vision often produces 3D models from image data. There is also a trend towards a combination of the two disciplines, e.g., as explored in augmented reality.

The following characterizations appear relevant but should not be taken as universally accepted:

Image processing and image analysis tend to focus on 2D images, how to transform one image to another, e.g., by pixel-wise operations such as contrast enhancement, local operations such as edge extraction or noise removal, or geometrical transformations such as rotating the image. This characterization implies that image processing/analysis neither require assumptions nor produce interpretations about the image content.

Computer vision includes 3D analysis from 2D images. This analyzes the 3D scene projected onto one or several images, e.g., how to reconstruct structure or other information about the 3D scene from one or several images. Computer vision often relies on more or less complex assumptions about the scene depicted in an image.

Machine vision is the process of applying a range of technologies & methods to provide imaging-based automatic inspection, process control and robot guidance in industrial applications. Machine vision tends to focus on applications, mainly in manufacturing, e.g., vision based robots and systems for vision based inspection or measurement. This implies that image sensor technologies and control theory often are

integrated with the processing of image data to control a robot and that real-time processing is emphasised by means of efficient implementations in hardware and software. It also implies that the external conditions such as lighting can be and are often more controlled in machine vision than they are in general computer vision, which can enable the use of different algorithms.

There is also a field called imaging which primarily focus on the process of producing images, but sometimes also deals with processing and analysis of images. For example, medical imaging includes substantial work on the analysis of image data in medical applications.

Finally, pattern recognition is a field which uses various methods to extract information from signals in general, mainly based on statistical approaches and artificial neural networks. A significant part of this field is devoted to applying these methods to image data.

Photogrammetry also overlaps with computer vision, e.g., stereophotogrammetry vs. computer stereo vision.

Python: Python is a commonly used high-level programming language created for general-purpose programming. It was created by Guido van Rossum and initially made the debut in 1991. An interpreted language, Python has a design philosophy that emphasizes code readability (mostly using whitespace indentation to delimit code blocks rather than curly keywords or brackets), with a syntax that permits programmers to express ideas in lesser lines of code than might be used in programming languages such as C++ or Java.

OpenCV: OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a shared infrastructure for computer vision applications and to accelerate the use of machine perception in the industrial and commercial products. Being a BSD-licensed product, OpenCV makes it easy for startups and businesses to utilize and modify and extend the code.

The library has more than 2600 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art machine learning and computer vision algorithms. These algorithms can be utilized to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, combine and stitch images together to produce a high resolution image of an entire scene, find similar looking images with same properites from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. The library is used extensively in companies, research groups and by governmental bodies.

2.4 Implementation

In this research paper, we are detecting and highlighting lanes and boundaries of the road, so that decisions can be made for autonomous intelligent vehicles. A pipeline is created to process images and videos in real time and highlight roads. The algorithm is written using Python programming language, and the computer vision and image processing is done via OpenCV toolkit.

The usual method of lane detection is to first shot an image of road with the help of a camera fixed on the vehicle. Then the image is added a filter for color selection, selecting from a range of RGB value, from a little yellow color to all white and mapped that image to a grayscale image (binary image, i.e. only black and white) in order to minimize the processing time. Secondly, as presence of unwanted objects in the image will hinder the correct edge detection. Therefore, filters should be applied to remove these unwanted objects. These objects are removed by filtering a region of interest, where the road lane will be visible through the camera. Then the edge detector is used to yield an edge image by using canny edge filter with automatic thresholding to obtain those edges. Then that edged image is sent to the line detector after detecting the edges

which will produces a left and right lane boundary segment. This lane boundary scan uses Hough transform algorithm to perform the scan on the edge detected image. The scan returns a series of points on the right and left side. Finally, pair of polygons are fitted to these data points to represent the lane boundaries. For visualization purposes, the polygons are displayed on the original color image.

Chapter - 3

3. System Analysis

3.1 Applications

Applications range from tasks such as industrial machine vision systems which, say, inspect bottles speeding by on a production line, to research into artificial intelligence and computers or robots that can comprehend the world around them. The computer vision and machine vision fields have significant overlap. Computer vision covers the core technology of automated image analysis which is used in many fields. Machine vision usually refers to a process of combining automated image analysis with other methods and technologies to provide automated inspection and robot guidance in industrial applications. In many computer vision applications, the computers are pre-programmed to solve a particular task, but methods based on learning are now becoming increasingly common. Examples of applications of computer vision include systems for:

- Automatic inspection, e.g., in manufacturing applications
- Assisting humans in identification tasks, e.g., a species identification system
- Controlling processes, e.g., an industrial robot
- Detecting events, e.g., for visual surveillance or people counting
- Interaction, e.g., as the input to a device for computer-human interaction
- Modeling objects or environments, e.g., medical image analysis or topographical modeling
- Navigation, e.g., by an autonomous vehicle or mobile robot; and
- Organizing information, e.g., for indexing databases of images and image sequences.
- File:DARPA Visual Media Reasoning Concept Video.ogv
- DARPA's Visual Media Reasoning concept video

One of the most prominent application fields is medical computer vision or medical image processing. This area is characterized by the extraction of information from image data for the purpose of making a medical diagnosis of a patient. Generally, image data is in the form of microscopy images, X-ray images, angiography images, ultrasonic images, and tomography images. An example of information which can be extracted from such image data is detection of tumours, arteriosclerosis or other malign changes. It can also be measurements of organ dimensions, blood flow, etc. This application area also supports medical research by providing new information, e.g., about the structure of the brain, or about the quality of medical treatments. Applications of computer vision in the medical area also includes enhancement of images that are interpreted by humans, for example ultrasonic images or X-ray images, to reduce the influence of noise.

A second application area in computer vision is in industry, sometimes called machine vision, where information is extracted for the purpose of supporting a manufacturing process. One example is quality control where details or final products are being automatically inspected in order to find defects. Another example is measurement of position and orientation of details to be picked up by a robot arm. Machine vision is also heavily used in agricultural process to remove undesirable food stuff from bulk material, a process called optical sorting.

Military applications are probably one of the largest areas for computer vision. The obvious examples are detection of enemy soldiers or vehicles and missile guidance. More advanced systems for missile guidance send the missile to an area rather than a specific target, and target selection is made when the missile reaches the area based on locally acquired image data. Modern military concepts, such as "battlefield awareness", imply that various sensors, including image sensors, provide a rich set of information about a combat scene which can be used to support strategic decisions. In this case, automatic processing of the data is used to reduce complexity and to fuse information from multiple sensors to increase reliability.

Artist's Concept of Rover on Mars, an example of an unmanned land-based vehicle. Notice the stereo cameras mounted on top of the Rover.

One of the newer application areas is autonomous vehicles, which include submersibles, land-based vehicles (small robots with wheels, cars or trucks), aerial vehicles, and unmanned aerial vehicles (UAV). The level of autonomy ranges from fully autonomous (unmanned) vehicles to vehicles where computer vision based systems support a driver or a pilot in various situations. Fully autonomous vehicles typically use computer vision for navigation, i.e. for knowing where it is, or for producing a map of its environment (SLAM) and for detecting obstacles. It can also be used for detecting certain task specific events, e.g., a UAV looking for forest fires. Examples of supporting systems are obstacle warning systems in cars, and systems for autonomous landing of aircraft. Several car manufacturers have demonstrated systems for autonomous driving of cars, but this technology has still not reached a level where it can be put on the market. There are ample examples of military autonomous vehicles ranging from advanced missiles, to UAVs for recon missions or missile guidance. Space exploration is already being made with autonomous vehicles using computer vision, e.g., NASA's Mars Exploration Rover and ESA's ExoMars Rover.

Other application areas include:

Support of visual effects creation for cinema and broadcast, e.g., camera tracking (matchmoving).

Surveillance.

3.2 System Methods

The organization of a computer vision system is highly application dependent. Some systems are stand-alone applications which solve a specific measurement or detection problem, while others constitute a sub-system of a larger design which, for example, also contains sub-systems for control of mechanical actuators, planning,

information databases, man-machine interfaces, etc. The specific implementation of a computer vision system also depends on if its functionality is pre-specified or if some part of it can be learned or modified during operation. Many functions are unique to the application. There are, however, typical functions which are found in many computer vision systems.

Image acquisition – A digital image is produced by one or several image sensors, which, besides various types of light-sensitive cameras, include range sensors, tomography devices, radar, ultra-sonic cameras, etc. Depending on the type of sensor, the resulting image data is an ordinary 2D image, a 3D volume, or an image sequence. The pixel values typically correspond to light intensity in one or several spectral bands (gray images or color images), but can also be related to various physical measures, such as depth, absorption or reflectance of sonic or electromagnetic waves, or nuclear magnetic resonance.

Pre-processing – Before a computer vision method can be applied to image data in order to extract some specific piece of information, it is usually necessary to process the data in order to assure that it satisfies certain assumptions implied by the method.

Examples are

Re-sampling in order to assure that the image coordinate system is correct.

Noise reduction in order to assure that sensor noise does not introduce false information.

Contrast enhancement to assure that relevant information can be detected.

Scale space representation to enhance image structures at locally appropriate scales.

Feature extraction – Image features at various levels of complexity are extracted from the image data. Typical examples of such features are

Lines, edges and ridges.

Localized interest points such as corners, blobs or points.

More complex features may be related to texture, shape or motion.

Detection/segmentation – At some point in the processing a decision is made about which image points or regions of the image are relevant for further processing.

Examples are

Selection of a specific set of interest points

Segmentation of one or multiple image regions which contain a specific object of interest.

Segmentation of image into nested scene architecture comprised foreground, object groups, single objects or salient object parts (also referred to as spatial-taxon scene hierarchy)

High-level processing – At this step the input is typically a small set of data, for example a set of points or an image region which is assumed to contain a specific object.

The remaining processing deals with, for example:

Verification that the data satisfy model-based and application specific assumptions.

Estimation of application specific parameters, such as object pose or object size.

Image recognition – classifying a detected object into different categories.

Image registration – comparing and combining two different views of the same object.

Decision making Making the final decision required for the application, for example:

Pass/fail on automatic inspection applications

Match / no-match in recognition applications

Flag for further human review in medical, military, security and recognition applications

3.3 Proposed System

The proposed system recognizes marks of lanes and boundaries on the road. This will be a real-time processing of the video input from camera on the car. The project will take the video and divide it into various frames, out of which each frame will be converted from color image to grayscale, after which image processing takes place, such as color selection, to select a variation of color from yellow-orange to white, which are used for road marks and lanes. With the help of Edge Detection, we find out the edges of these lanes, and using Hough Transform, we generate the lane lines for these road lanes, which intercept at an imaginary point. This defines the road lane, a self-driving car needs to follow.

"Why an automated intelligent system"
With using intelligent systems for driving, this will decrease a lot of hassle and traffic jams. Also, this will result in increase of safety of people. With automated systems, wastage of fuel and pollution will also decrease, as fuel combustion would be under control of the machine and would be most efficient way.

3.4 Benefits of the System

Each of the application areas described above employ a range of computer vision tasks; more or less well-defined measurement problems or processing problems, which can be solved using a variety of methods. Some examples of typical computer vision tasks are presented below.

Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information, e.g., in the forms of decisions. Understanding in this context means the transformation of visual images (the input of the retina) into descriptions of the world that can interface with other thought

processes and elicit appropriate action. This image understanding can be seen as the disentangling of symbolic information from image data using models constructed with the aid of geometry, physics, statistics, and learning theory.

Recognition

The classical problem in computer vision, image processing, and machine vision is that of determining whether or not the image data contains some specific object, feature, or activity. Different varieties of the recognition problem are described in the literature:

Object recognition (also called object classification) – one or several pre-specified or learned objects or object classes can be recognized, usually together with their 2D positions in the image or 3D poses in the scene. Blippar, Google Goggles and LikeThat provide stand-alone programs that illustrate this functionality.

Identification – an individual instance of an object is recognized. Examples include identification of a specific person's face or fingerprint, identification of handwritten digits, or identification of a specific vehicle.

Detection – the image data are scanned for a specific condition. Examples include detection of possible abnormal cells or tissues in medical images or detection of a vehicle in an automatic road toll system. Detection based on relatively simple and fast computations is sometimes used for finding smaller regions of interesting image data which can be further analyzed by more computationally demanding techniques to produce a correct interpretation.

Currently, the best algorithms for such tasks are based on convolutional neural networks. An illustration of their capabilities is given by the ImageNet Large Scale Visual Recognition Challenge; this is a benchmark in object classification and detection, with millions of images and hundreds of object classes. Performance of convolutional neural networks, on the ImageNet tests, is now close to that of humans. The best algorithms still struggle with objects that are small or thin, such as a small ant on a stem of a flower or a person holding a quill in their hand. They also have trouble

with images that have been distorted with filters (an increasingly common phenomenon with modern digital cameras). By contrast, those kinds of images rarely trouble humans. Humans, however, tend to have trouble with other issues. For example, they are not good at classifying objects into fine-grained classes, such as the particular breed of dog or species of bird, whereas convolutional neural networks handle this with ease.

Several specialized tasks based on recognition exist, such as:

Content-based image retrieval – finding all images in a larger set of images which have a specific content. The content can be specified in different ways, for example in terms of similarity relative a target image (give me all images similar to image X), or in terms of high-level search criteria given as text input (give me all images which contains many houses, are taken during winter, and have no cars in them).

Computer vision for people counter purposes in public places, malls, shopping centres

Pose estimation – estimating the position or orientation of a specific object relative to the camera. An example application for this technique would be assisting a robot arm in retrieving objects from a conveyor belt in an assembly line situation or picking parts from a bin.

Optical character recognition (OCR) – identifying characters in images of printed or handwritten text, usually with a view to encoding the text in a format more amenable to editing or indexing (e.g. ASCII).

2D Code reading Reading of 2D codes such as data matrix and QR codes.

Facial recognition

Shape Recognition Technology (SRT) in people counter systems differentiating human beings (head and shoulder patterns) from objects

Motion analysis

Several tasks relate to motion estimation where an image sequence is processed to produce an estimate of the velocity either at each points in the image or in the 3D scene, or even of the camera that produces the images . Examples of such tasks are:

Egomotion – determining the 3D rigid motion (rotation and translation) of the camera from an image sequence produced by the camera.

Tracking – following the movements of a (usually) smaller set of interest points or objects (e.g., vehicles or humans) in the image sequence.

Optical flow – to determine, for each point in the image, how that point is moving relative to the image plane, i.e., its apparent motion. This motion is a result both of how the corresponding 3D point is moving in the scene and how the camera is moving relative to the scene.

Scene reconstruction

Given one or (typically) more images of a scene, or a video, scene reconstruction aims at computing a 3D model of the scene. In the simplest case the model can be a set of 3D points. More sophisticated methods produce a complete 3D surface model. The advent of 3D imaging not requiring motion or scanning, and related processing algorithms is enabling rapid advances in this field. Grid-based 3D sensing can be used to acquire 3D images from multiple angles. Algorithms are now available to stitch multiple 3D images together into point clouds and 3D models.

Image restoration

The aim of image restoration is the removal of noise (sensor noise, motion blur, etc.) from images. The simplest possible approach for noise removal is various types of filters such as low-pass filters or median filters. More sophisticated methods assume a model of how the local image structures look like, a model which distinguishes them from the noise. By first analysing the image data in terms of the local image structures, such as lines or edges, and then controlling the filtering based on local information from the analysis step, a better level of noise removal is usually obtained compared to the simpler approaches.

An example in this field is inpainting.

3.5 Domain where it can be used extensively

- Pot hole detection
- Detecting traffic light and Maintaining traffic conflicts
- Avoiding accidents
- Detecting and following traffic signs

Chapter - 4

4.Software Requirement & Specifications

4.1 Software Requirement

The project is implemented in Python as it provides the easy implementation of any high level mathematical equations. Python's focus on readability makes it an excellent choice, especially when compared to other languages. Python also have lots of libraries to make solve statistical or plot any equation on graph and show it an instant. The IDE for this project is simply Sublime Text and the project is executed via Terminal. Numpy, one of the great libraries for easy implementation of matrix and vectors has also been used in this project. One of the great things about PYTHON is that it can run on any operating system, given that the specific libraries are installed for project to run in that operating system. So, this project can be executed in Linux, Windows or Mac without any problem.

The project uses OpenCV toolkit for image recognition and image processing. OpenCV: OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a shared infrastructure for computer vision applications and to accelerate the use of machine perception in the industrial and commercial products. Being a BSD-licensed product, OpenCV makes it easy for startups and businesses to utilize and modify and extend the code.

4.2 Hardware Requirement

As the project involve big data set of images and videos and Image Recognitioin is a state-of-the-art system, its hardware requirements are high. Any system with at least i5 3rd generation or above processor, more than 6GB RAM, 1GB Hard Disk 5200rpm

is sufficient and a graphics card for rendering videos in real-time. It can run on lower configuration than above recommended but the computer vision toolkit will process the work very slow and it may take more than just few hours to be process images and videos for running. For live video processing, the above configuration is recommended.

4.3 Software Analysis Report

About Python

Python was created by Guido van Rossum during 1985- 1990. It is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactical constructions than other languages.

● Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

● Python is Interactive: You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

● Python is Object-Oriented: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

● Python is a Beginner's Language: Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Features (Python2.7):

4.3.1 Easy-to-Learn: Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

4.3.2 Easy-to-read: Python code is more clearly defined and visible to the eyes.

4.3.3 Easy-to-maintain: Python's source code is fairly easy-to-maintain.

4.3.4 A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

4.3.5 Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

4.3.6 Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

4.3.7 Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

4.3.8 Databases: Python provides interfaces to all major commercial databases.

4.3.9 GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

4.3.10 Scalable: Python provides a better structure and support for large programs than shell scripting.

About OpenCV

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage and is now maintained by Itseez. The library is cross-platform and free for use under the open-source BSD license.

OpenCV supports the Deep Learning frameworks TensorFlow, Torch/PyTorch and Caffe.

Officially launched in 1999, the OpenCV project was initially an Intel Research initiative to advance CPU-intensive applications, part of a series of projects including

real-time ray tracing and 3D display walls. The main contributors to the project included a number of optimization experts in Intel Russia, as well as Intel's Performance Library Team. In the early days of OpenCV, the goals of the project were described as:

Advance vision research by providing not only open but also optimized code for basic vision infrastructure. No more reinventing the wheel.
Disseminate vision knowledge by providing a common infrastructure that developers could build on, so that code would be more readily readable and transferable.
Advance vision-based commercial applications by making portable, performance-optimized code available for free—with a license that did not require code to be open or free itself.

Applications

OpenCV's application areas include:

2D and 3D feature toolkits
Egomotion estimation
Facial recognition system
Gesture recognition
Human–computer interaction (HCI)
Mobile robotics
Motion understanding
Object identification
Segmentation and recognition
Stereopsis stereo vision: depth perception from 2 cameras
Structure from motion (SFM)
Motion tracking
Augmented reality
To support some of the above areas, OpenCV includes a statistical machine learning library that contains:

Boosting
Decision tree learning
Gradient boosting trees
Expectation-maximization algorithm
k-nearest neighbor algorithm
Naive Bayes classifier
Artificial neural networks
Random forest
Support vector machine (SVM)
Deep neural networks (DNN)

4.4 Technologies and Requirements

IDE: Jupyter Notebook

Programming Language: Python

Libraries:
Matplotlib: for loading and displaying images,
numpy: For easy implementation of matrix and vectors,
opencv: for performing image operations,
moviepy: for video operations and converting them to frames and creating videos.

Chapter - 5

5. System Modelling and Design

Purpose

The purpose of this design document is to explore the logical view of architecture design, sequence diagram, data flow diagram of the structure of lane detection model for

performing the operations such as detecting road lanes and displaying the overlapped lanes on original image for visualization.

Scope

The Scope of this design document is to achieve the features of the system such as pre-processing the images, feature extraction, and displaying the overlapped lanes on original image for visualization.

5.1 Block Diagram and Algorithm

The proposed methodology uses some technique to remove the unwanted objects from the image and feature extraction to detect and classify the road lanes and boundaries. The proposed method comprises of 4 phases:

- Color Selection and preprocessing
- Filter Region of Interest
- Canny Edge Detection
- Hough Transform

The block schematic diagram of the proposed model is given in Fig 5.1

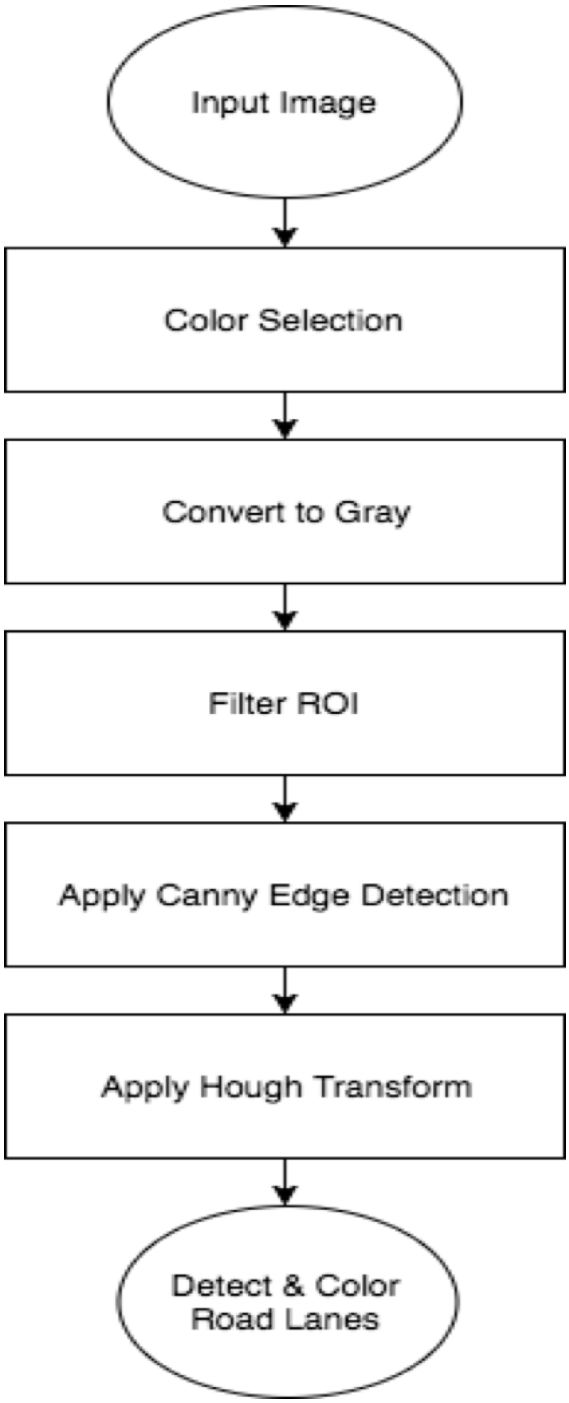

Fig 5.1: Block Diagram of Flow Chart

5.1.1 Color Selection and preprocessing

The pre-processing is a series of operations performed on input image. These operations are done to remove additional unwanted objects and information from the image, so that further processing can be done easily. Color selection is done to select only road lanes and road boundaries which lie in the color range of yellow to white. After color selection, the image is converted into grayscale to remove unwanted information.

5.1.2 Filter ROI

A region of interest (often abbreviated ROI), are samples within an data set identified for a particular purpose. The concept of a ROI is commonly used in many application areas. For example, in medical imaging, the boundaries of a tumor may be defined on an image or in a volume, for the purpose of measuring its size. The endocardial border may be defined on an image, perhaps during different phases of the cardiac cycle, for example end-systole and end-diastole, for the purpose of assessing cardiac function. In geographical information systems (GIS), a ROI can be taken literally as a polygonal selection from a 2D map. In computer vision and optical character recognition, the ROI defines the borders of an object under consideration. In many applications, symbolic (textual) labels are added to a ROI, to describe its content in a compact manner. Within a ROI may lie individual points of interest (POIs).

5.1.3 Canny Edge Detection

The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986. Canny also produced a computational theory of edge detection explaining why the technique works.

Among the edge detection methods developed so far, Canny edge detection algorithm is one of the most strictly defined methods that provides good and reliable detection. Owing to its optimality to meet with the three criteria for edge detection and the simplicity of process for implementation, it became one of the most popular algorithms for edge detection.

5.1.4 Hough Transform

The Hough transform is a feature extraction technique used in image analysis, computer vision, and digital image processing. The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting

procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform.

The classical Hough transform was concerned with the identification of lines in the image, but later the Hough transform has been extended to identifying positions of arbitrary shapes, most commonly circles or ellipses. The Hough transform as it is universally used today was invented by Richard Duda and Peter Hart in 1972, who called it a "generalized Hough transform" after the related 1962 patent of Paul Hough. The transform was popularized in the computer vision community by Dana H. Ballard through a 1981 journal article titled "Generalizing the Hough transform to detect arbitrary shapes".

Chapter - 6

Structure of Dataset

6.1 Dataset

The dataset is provided by Udacity from their Self-Driving Car Nanodegree partnered with Google. The dataset is a collection of recording of videos from a camera mounted at the top of car. Using this video clips, image frames are captured to be used to process and detect lane lines.

Fig 6.1: Dataset of Road Images

Humans are able to solve this segmentation problem with ease, but it's a little bit challenging for a computer program to correctly break up the image frames. Once the image has been segmented, the program then needs to detect each individual road mark and lanes. After breaking videos into images, we are required to highlight those lane lines and create a video from it.

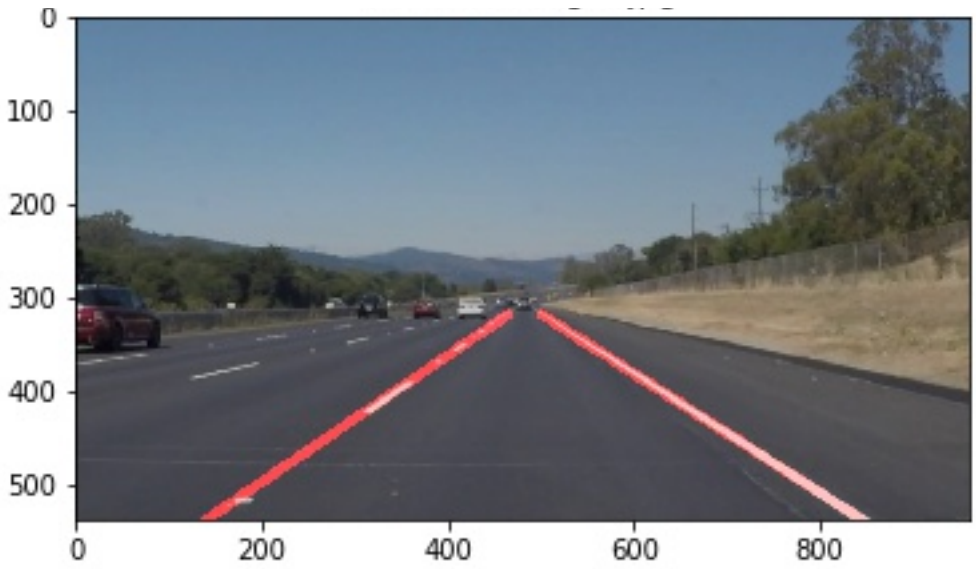

Fig 1(b): Resultant Image

Each frame is processed through the pipeline and overlapped on the original image. After processing each frame, they are combined and stitched back to form a video.

6.2 Implementation

A. Color Selection

Initially, the image is loaded using matplotlib library in RGB color, as shown in Fig 1. OpenCV provides inRange function to filter out colors lying between the two threshold values provided. Using this function, we filter out the color range from yellow-orange to white in RGB format ([210, 110, 70] to [255, 255, 255] respectively). After filtering out, this image is converted to grayscale image (binary image i.e. either black or white). Fig 2(a) and Fig 2(b) shows the output of this process.

Fig 2(a): Input Image

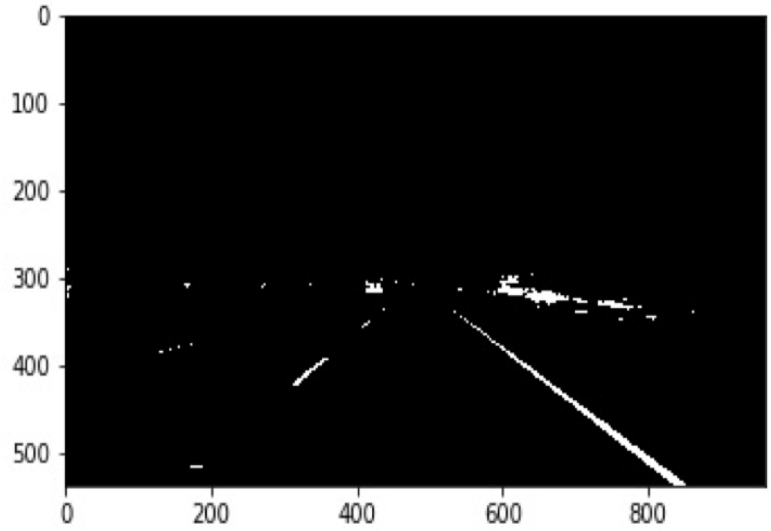

Fig 2(b): After Color Selection

B. Filter Region of Interest

Now we are needed to remove the unwanted objects from the image, which do not correspond to road and boundaries. As the camera is mounted on the top of the car, it captures all the images in the same fashion. That means, in all the images, the near part to the car of the image (i.e. at the bottom) is where most of the roads are covered. Hence, the bottom 40% of the image captures the road lanes and road boundaries. The result of this process is shown in Fig 3.

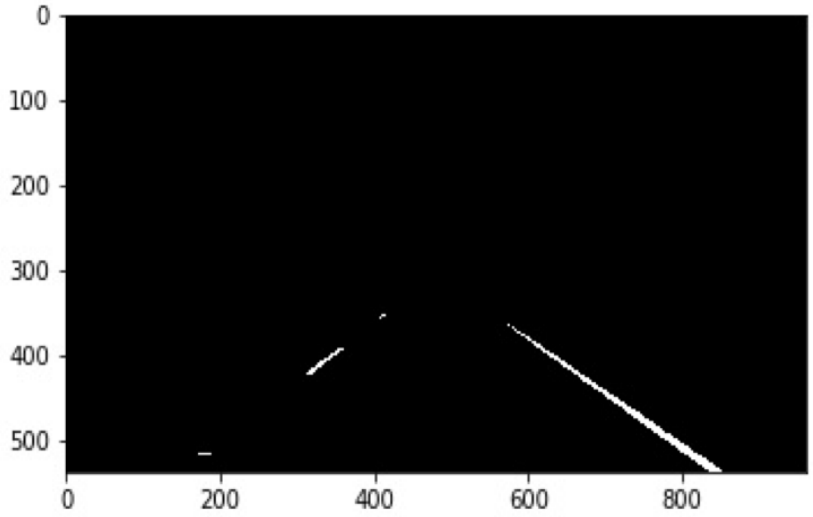

Fig 3: After Filter ROI

C. Canny Edge Detection

After filtering ROI, Canny Edge detection algorithm is used to find the edges in the input image and marks them in the output map. The image shows Canny Edge Detection applied on the output after filtering ROI. These resultant edges are shown below in Fig 4.

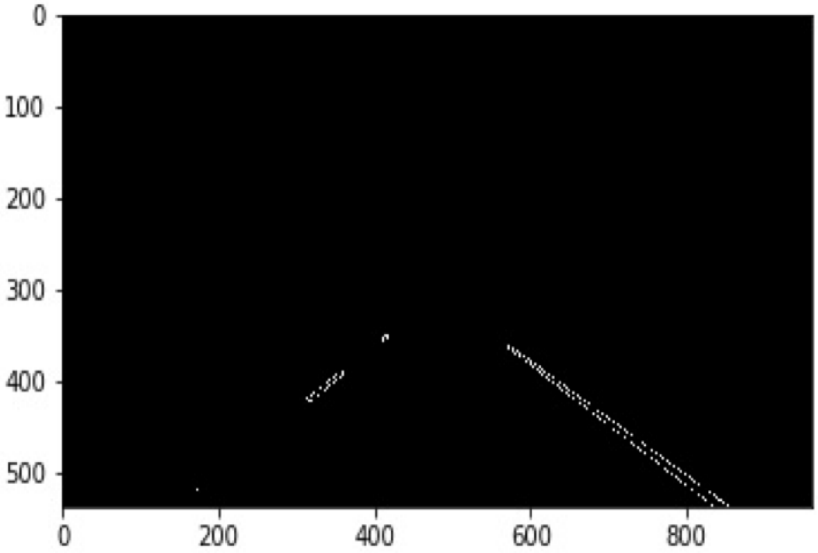

Fig 4: After Canny Edge Detection Filter

D. Hough Transform

The detected edges of road lanes image is now used to apply Hough Transform algorithm. This algorithm is used to find the direction of lane and join those points to form a complete line. This is done by joining those points that have the highest density of curves drawn through them in the histogram. After Getting the line direction, we extrapolate and extend that line to fit the road lane and cover the boundaries. The lines formed from are shown in Fig 5 and Fig 6.

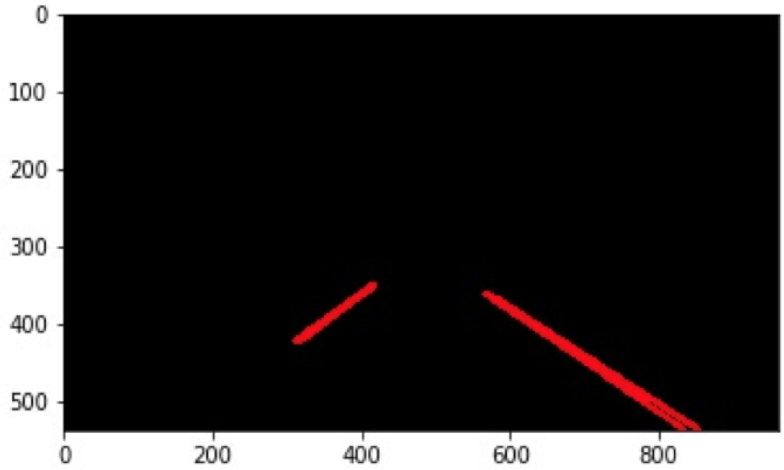

Fig 5: After Hough Transform

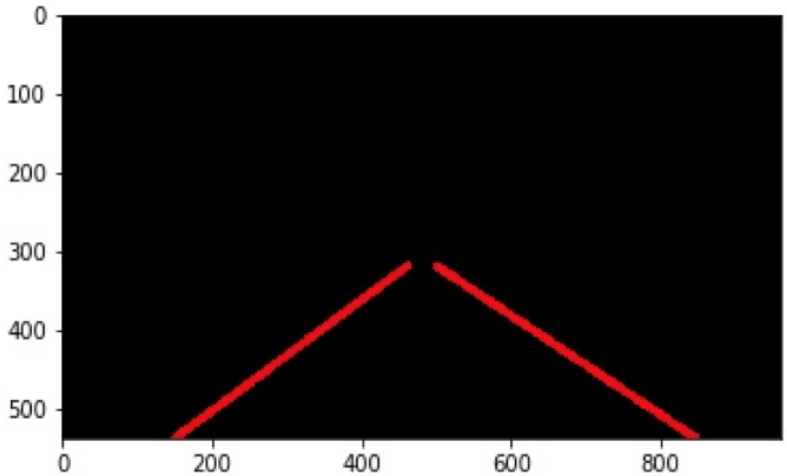

Fig 6: After Extrapolating Hough Transform

E. Color Road Lanes

After the lines are detected successfully, we draw them on the original color input image for visualization purpose. The area bounded by these lines is the lane in which an intelligent autonomous system should be driving. After overlapping, the visualized output is as shown below in Fig 7.

Fig 7: Lane Detected output

6.3 Output

This pipeline can now be used to detect lane lines from various images and videos and can be stitched together to form back the video. This will automate the driving for intelligent autonomous systems and help us to make the world a more better and safer place, with less of hassle and accidents, saving much more lives.

Chapter - 7

7.1 Conclusion and Results

In this paper, an algorithm for road lane and road boundary detection was proposed, based on image frames of dynamic video that were recorded from the camera mounted on the moving car. The procedure applied on the images in this algorithm was a combination of vanishing point, lane detection and filtering region of interest. During the lane detection, when the vehicle was moving, the camera covered the lane marks on the road. Hough transform was still able to generate the track assuming the lane was still there with breaks in it and extrapolating the line for the result. Additional features can be included to help the autonomous intelligent agents to make better decisions including detecting surrounding vehicles and environment and track their motion to avoid any conflicts and crashes, leading to severe damage and even loss of life. The lane detection technique plays an important role intelligent autonomous systems and various steps can be taken to increase the accuracy of results under different environmental conditions such as foggy day, sunny day, rainy day. More advancement are supposed to happen in this field in the near future and companies like Google and Tesla are making this easier with their programs partnered with Udacity and providing courses and certifications to developers. As per the studies, Self-driving cars have reduced accidents and increased safety for pedestrians and civilians than people driving, 10 times lower for age group of 60-69 years old and 40 times lower than learning drivers, making it much secure for everyone. This also reduces the pollution and emission of gases as efficient fuel combustion is done by bots.

7.2 Future Scope

The system can be designed for further enhancement. This could be developed according to the growing needs of the users. The work can be extended to increase the

results by using or adding some more relevant features. A lot of efforts have been made to get higher accuracy and there is tremendous scope of improving recognition accuracy by developing new feature extraction techniques or modifying the existing feature extraction techniques. Few future applications include Pot hole detection and traffic sign detection.

Chapter - 8

References

[1] A. Kaehler and G. Bradski, Learning OpenCV 3. New York: O'Reilly Media, 2015, pp. 120-180.

[2] F. Mariut, D. Petrisor and C. Fosalau, "Lane Mark Detection Using Hough Transform", In IEEE International Conference and Exposition on Electrical and Power Engineering, pp. 871 - 875, 2012.

[3] Anik Saha, Dipanjan Das Roy, Kaushik Deb and Tauhidul Alam, "Automated Road Lane Detection for Intelligent Vehicles" in Global Journal of Computer Science and Technology, Volume 12 Issue 6 Version 1.0, March 2012.

[4] Ajit Danti, P. S. Hiremath and Jyoti Y. Kulkarni, "An Image Processing Approach to Detect Lanes, Pot Holes and Recognize Road Signs in Indian Roads" in International Journal of Modeling and Optimization, Vol. 2, No. 6, December 2012.

[5] Abdulhakam.AM.Assidiq, Md. Rafiqul Islam, Othman O. Khalifa, Sheroz Khan."Real Time Lane Detection for Autonomous Vehicles". Computer and Communication Engineering, IEEE, 2008.

[6] M. Meuter, S. Muller, S. Hold, A. Mika, C. Nunn and A. Kummert, "A Novel Approach to Lane Detection and Tracking", In IEEE 12th International Conference on Intelligent Transportation Systems, pp. 1-6, 2009.

[7] Gurjashan Singh Pannu, Patiala Mohammad Dawud Ansari, Pritha Guptai, "Design and Implementation of Autonomous Car using Raspberry Pi", International Journal of Computer Applications (0975 – 8887) Volume 113 – No. 9, March 2015

Glossary

Image Basics:

- Digital images are represented as rectangular arrays of square pixels.
- Digital images use a left-hand coordinate system, with the origin in the upper left corner, the x-axis running to the right, and the y-axis running down.
- Most frequently, digital images use an additive RGB model, with eight bits for the red, green, and blue channels.
- Lossless compression retains all the details in an image, but lossy compression results in loss of some of the original image detail.
- BMP images uncompressed, meaning they have high quality but also that their file sizes are large.
- JPEG images use lossy compression, meaning that their file sizes are smaller, but image quality may suffer.
- TIFF images can be uncompressed or compressed with lossy or lossless compression.
- Depending on the camera or sensor, various useful pieces of information may be stored in an image file, in the image metadata.

OpenCV Images:

- OpenCV images are stored as three-dimensional NumPy arrays.
- In OpenCV images, the blue channel is specified first, then the green, then the red, i.e., BGR instead of RGB.
- Images are read from disk with the cv2.imread() method.
- We create a sizable window that automatically scales the displayed image with the cv2.namedWindow() method.
- We cause an image to be displayed in a window with the cv2.imshow() method.

- We cause our program to pause until we press a key with the cv2.waitKey(0) method call.
- We can resize images with the cv2.resize() method.

- NumPy array commands, like img[img < 128] = 0, and be used to manipulate the pixels of an OpenCV image.
- Command-line arguments are accessed via the sys.argv list; sys.argv[1] is the first parameter passed to the program, sys.argv[2] is the second, and so on.
- Array slicing can be used to extract subimages or modify areas of OpenCV images, e.g., clip = img[60:150, 135:480, :].
- Metadata is not retained when images are loaded as OpenCV images.

Drawing and Bitwise Operations:

- We can use the NumPy zeros() method to create a blank, black image.
- We can draw on OpenCV images with methods such as cv2.rectangle(), cv2.circle(), cv2.line(), and more.
- We can use the cv2.bitwise_and() method to apply a mask to an image.
- Creating Histograms
- We can load images in grayscale by passing the cv2.IMREAD_GRAYSCALE parameter to the cv2.imread() method.
- We can create histograms of OpenCV images with the cv2.calcHist() method.
- We can separate the RGB channels of an image with the cv2.split() method.
- We can display histograms using the matplotlib pyplot figure(), title(), xlabel(), ylabel(), xlim(), plot(), and show() methods.

Blurring images:

- Applying a low-pass blurring filter smooths edges and removes noise from an image.
- Blurring is often used as a first step before we perform Thresholding, Edge Detection, or before we find the Contours of an image.
- The averaging blur can be applied to an image with the cv2.blur() method.
- The Gaussian blur can be applied to an image with the cv2.GaussianBlur() method.
- The blur kernel for the averaging and Gaussian blur methods must be odd.

- Larger blur kernels may remove more noise, but they will also remove detail from and image.
- The int() function can be used to parse a string into an integer.

Thresholding:

- Thresholding produces a binary image, where all pixels with intensities above (or below) a threshold value are turned on, while all other pixels are turned off.
- The binary images produced by thresholding are held in two-dimensional NumPy arrays, since they have only one color value channel.
- The cv2.merge() method can be used to combine three single-channel image layers into a single, color image.
- Thresholding can be used to create masks that select only the interesting parts of an image, or as the first step before Edge Detection or finding Contours.
- Depending on its parameters, the cv2.threshold() method can perform simple fixed-level thresholding or adaptive thresholding.

Edge Detection:

- Sobel edge detection is implemented in the cv2.Sobel() method. We usually call the method twice, to find edges in the x and y dimensions.
- The two edge images returned by two cv2.Sobel() calls can be merged using the cv2.bitwise_or() method.
- Edge detection methods return data that is signed instead of unsigned, so data types such as cv2.CV_16S or cv2.CV-64F should be used instead of unsigned, 8-bit integers (uint8).
- The cv2.createTrackbar() method is used to create trackbars on windows that have been created by our programs.
- We use Python functions as callbacks when we create trackbars using cv2.createTrackbar().
- Use the Python global keyword to indicate variables referenced inside functions that are global variables, i.e., variables that are first declared in other parts of the program.

Contours:

- Contours are closed curves of points or line segments, representing the boundaries of objects in an image.