# 江 蘇 大 學

# JIANGSU UNIVERSITY

本 科 毕 业 论 文

# LANE LINE DETECTION SYSTEM WITH LINE RECOGNITION

学院名称: COMPUTER SCIENCE & TECHNOLOGY
专业班级: COMPUTER SCIENCE
学生姓名: CHANSA BWALYA; 5111710501

# PREFACE

"DEDICATION, HARD WORK, PLUS PATIENCE"

-NIPSEY HUSSLE

# SUMMARY

This thesis discusses the issue of identification of lane lines automatically by the car system, aiding the driver in staying within the correct lanes according to traffic ad safety regulations. The application technique is discussed as well as broken down into its basic components and stages.

Traffic accidents have become one of the most serious problems in today's world. Increase in the number of vehicles, human errors towards traffic rules and the difficulty to oversee situational dangers by drivers are contributing to the majority of accidents on the road.

Lane lines are road markings that indicate the correct and safe passage of road vehicles. These not only ensure an orderly manner of movement but also a safe way for all users to a certain extent. In humans, the markings on the road are observed and the motorists are expected to drive accordingly while observing their surroundings. That being said, it means there are two phases involved in this process. First is recognition, identifying that there are lane lines present. This can be hindered by certain factors such as debris or dirt on the roads preventing the drivers from easy sight. A practical example is heavy duty sand tipper trucks tend to leave spillage of sand on the roads which might cover up important road markings. Another factor is poor weather conditions such as heavy rain. A practical example is when a typhoon occurs, it is nearly impossible to see the lane lines and drivers are encouraged not to drive during such times. Flooded roads affect visibility of lane lines too. However, human error is one of the main causes of road traffic accidents as sometimes drivers may not concentrate enough and in some cases, the drivers may be sleepy and eventually make a mistake. To solve this experts have designed computer algorithms that are able to recognize these lanes lines through various forms of image processing. Not all problems are solved but a great portion of them are, particularly when human error is involved.

In image processing, **line detection** is an algorithm that takes a collection of n edge points and finds all the lines on which these edge points lie. The most popular line detectors are the Hough transform and convolution-based techniques. In this paper, I have designed and implemented an automatic lane marking detection algorithm. Various experiments show that the proposed approach can be very effective in lane detection. The algorithm is verified for both white lanes and yellow lanes by improving color detection and averaging temporal patterns. We will develop a simple pipeline using OpenCV and Python for finding lane lines in an image, then apply this pipeline to a full video feed.

Keywords: Line detection, Image processing, OpenCV, Hough Transform

## JIANGSU UNIVERSITY THESIS COPYRIGHT AUTHORIZATION

I fully understand the rules of the school concerning dissertations intellectual property rights during the study for a degree in school.
Working papers on intellectual property belong to Jiangsu University,
The contents of this document can be copied, scan or reproduced for the purpose of a scientific research and study only

Topic Title: LANE LINE DETECTION SYSTEM WITH LANE RECOGNITION

# Contents

# CHAPTER I: SYSTEM REVIEW

## BACKGROUND SYSTEM DEVELOPMENT

A lane lines detection system is a technology capable of detecting lanes lines on the road from images and video feed from an input source (camera). This system is able to recognize both yellow and white road markings. There are two main algorithms that are used in lane lines detection but ultimately, it all comes down to image processing and detection of sharp changes in contrast. Digital image processing is the use of a digital computer to process digital images through an algorithm. As a subcategory or field of digital signal processing, digital image processing has many advantages over analog image processing. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and distortion during processing.

In a lane detection system, a camera is placed behind the wind shield of the vehicle and images/video of road are captured. The lines on the road are interpreted and lanes are identified. Whenever the vehicle moves out of a lane unintentionally, a warning is given to the driver. Lane line detection is the initial step of improving road safety. There are two methods for lane detection; feature based approach and model based approach. The feature based approach use low level features such as edges whereas model based approach use geometric parameters for detecting lanes. Lane detection system poses many challenges and issues which includes lane appearance diversity, variation in clarity of image, changes in visibility conditions. We have to investigate the type of lane behaviors and the challenges in tracking to solve such lane detection and tracking problems. Usually yellow and white colors are used for lane markers but reflectors lane are marked with special colors. The number and width of lanes varies with countries.

There can be issues related to clarity in images due to the presence of shadows. The lane markers may be occluded by the nearby vehicles. When the vehicle comes out of a tunnel, there is an abrupt change in illumination. So the clarity of image is affected by extreme illumination too. The visibility of the lane markers decrease due to various weather conditions such as rain, fog, snow. The visibility can be less in night condition

## SYSTEM RESEARCH

Lane detection involves recognizing sharp changes in contrast in a picture or video of the road. From these changes, it's essentially possible to require note of what seems to be 'alerting features' or features that stand move into a district of interest of the input coming in from the cameras. However, even as the human driver may face certain challenges in identifying lane lines, the pc algorithm has similar difficulties.

A number of these difficulties are bad road conditions, presence of obstacles and poor atmospheric condition. Furthermore, bad lighting may eventually affect the standard of the image. Lane lines are marked differently with various varieties of lane marks. Within the appearance, form of these lane marks is elaborated. Their shapes alter from continuous lines through dashed lines and even reflectors maybe encountered on some roads. Lane marks color is additionally subject to vary, e.g. white, yellow, orange, and cyan colors are dead use at different locations.

Besides having various appearances, another challenge stems from the attitude effect which causes difficulties when trying to detect such narrow objects at distance. Lane marks are detected either supported their shape or their color. The least restrictive assumption about the lane markings is that they need a different appearance compared to the appearance of the road. Such assumption results in a full family of gradient-based features and their variants. Simple gradients may be computed from the initial image or from a smoothed version of it.

## LITERATURE SURVEY

The raw information provided by a lane detection system is the relative positions of lane lines with respect to the position of the automobile. Using this information, we may deduce several useful inputs such as lane crossing point, lane keeping performance, etc., to many kinds of automotive safety systems.

Most solutions propose at least one road model, either simple or complex, to represent lane marking. Road modeling can help immensely in boosting the performance of lane detection system by eliminating the outliers and also ignoring the evidence that is created by noises.
In this section, previously discovered lane detection algorithms are discussed. This section also investigates the best lane detection and tracking algorithms that can be selected according to specific road conditions.

Yim and Oh developed a 3 feature based lane detection algorithm. The features that were used are the starting position, orientation and intensity value. Within the initial step, a Sobel operator is applied to induce the sting information. The lane boundary is represented as a vector made up of the three features. The present lane vector is calculated supported the input image and therefore the previous lane model vector. Two windows, one for every, is employed for left and right boundaries. Assuming that N pixels in each horizontal line, N lane vector candidates are then generated. The simplest candidate is chosen supported the minimum distance from previous lane vector employing a weighted distance metric. For equalization, each feature is assigned a distinct weight. Then a lane inference system is employed to predict the new lane vector. If the road width had to change abruptly, the presently calculated vector is discarded and therefore the previous one is taken as current vector.

A lane detection mechanism for the urban environment is proposed by Sehestedt et al. Since the lane markers may not be adequately visible due to factors such as wear and tear, occlusions and sometimes complicated road geometry, a weak model is used for detecting lane markers. Within the inverse perspective mapped image, particle filter is applied from bottom row to top. The filter is tuned in such how to trace multiple lanes.

Aly presents a true time and robust approach to detect lane markers in urban roads. It first acquires a top view of the road image using inverse perspective mapping for avoiding the angle effect. Then the highest view is filtered using selective oriented two dimensional Gaussian kernel. The filter is adjusted specifically for bright lines with dark background with also a specific width. So it's high response to the road markers and retain only the very best values by selecting rickettsiosis quantile value from filtered image and removing all values below the edge. Then the straight lane lines are detected using a simplified Hough transform. Thereafter, a RANSAC line fitting is used. This provides initial guess to the RANSAC spline fitting step. Thereafter, a post processing step is finished to localize the spline and also extend it within the image. The algorithm does not perform tracking. It can detect any number of lane boundaries within the image not just the current lane.

Kim developed a lane detection algorithm which may handle challenging scenarios such as faded lane markers, lane curvatures and also splitting lanes. During the initial step, a gradient detector and an intensity change detector are used to remove the non-lane markers. Artificial Neural Networks (ANN) are then applied on remaining samples for lane detection. The detected lane marker pixels are then grouped using cubic splines. Hypotheses are generated from the random set of line segments. RANSAC algorithm facilitates in validating the hypotheses. Particle filtering is also employed for lane tracking.

Cheng et al introduced a hierarchical algorithm for lane detection. High dimensional feature points are extracted supported feature color extraction. it's accustomed distinguish structured roads from unstructured roads. Then connected components are applied onto the feature points. A feature vector is made for feature points exceeding a threshold. Eigen value Decomposition Regularized Discriminant Analysis is finished to cut back the variance of the model. Then maximum likelihood Gaussian parameters are estimated. The extracted feature points are then used as the detected lanes in the structured roads. For unstructured roads, the whole scene is split supported mean-shift segmentation. Each region is taken into account to be homogeneous and lane markers are detected using Bayes rule. Initially, color image is converted to gray scale and so shadow removal is performed. Modified Canny edge detector is employed for edge detection and so Hough transform is applied on the resultant binary image. Horizon line lane boundary scan is implemented using the sting detection results of Hough transform. The scan starts from the portion where the Hough lines meet the underside image border. The scan returns data points admire the perimeters. Then a hyperbola pair fitting is finished on the collected data points.

Borkar et al proposed lane detection supported Hough transform and iterated matched filters. RANSAC algorithm is employed to avoid outliers because of noise and other artifacts within the road. Kalman filter is employed to trace the lanes. the primary step within the algorithm is to convert the color image to gray scale and temporal blurring then on it image inverse perspective mapping (IPM) is applied. An adaptive threshold is applied on the IPM image to come up with a binary image. Each binary image is split into two halves and every one contains one lane marker. an occasional resolution Hough transform is applied on the binary images. A 1- dimensional matched filter is applied at each sample along the road to seek out the approximate center of every line. After estimating the middle, RANSAC algorithm is applied to the information points for lane detection.

Lin et al performed lane detection supported lateral inhibition property of human vision system. It enables the algorithm to withstand various types of weather. The algorithm isn't supported thresholding. It uses both 2D and 3D geometric information of the lane markers. During lane detection, the positive and negative second difference maps of the image are used. These provide a strong contrast between the road and lane marker. A dual Gaussian model is used to see whether the two peaks admire global maximum on right side of left lane marker and global minimum on left side of left marker. After detection, verification is performed to measure the correctness of detected lane marker. The verification is done to test the slopes and intercept relationships, the real road width and also the position of vanishing point.

A lane detection method suitable at midnight is proposed by Borkar et al. In this algorithm, the Region of Interest (ROI) is first initialized, eliminating the sky and other irrelevant objects. Within the next step, a gray scale image is obtained by averaging the three color channels. Temporal blurring is employed to elongate the dashed lanes. Adaptive thresholding is finished to extract the brilliant objects. The resultant binary image is split into left and right halves. A kind of Low Resolution Hough Transform is applied on each half for detecting straight lines. Then a Gaussian kernel is employed with iterated matched filter to extract lane markers.

The algorithm introduced by Liu et al has two main stages; initial detection of lanes and their subsequent tracking. The attitude effect is off from the image using inverse perspective mapping. Thereafter, a Statistical Hough transform (SHT) algorithm is applied on the IPM image for lane detection to be done. SHT operates on intensity pictures and uses multiple kernel density to show the Hough variables (ρ, θ). As SHT works on every pixel within the image it's computationally expensive and isn't suitable to run it on every frame. After initial detection, Particle Filter is deployed to find all the detected lanes and then update the parameters of the lane line model. The parameters of lane model are obtained and updated frame after frame. The SHT algorithm is computationally demanding and is combined with tracking using Particle Filter. For demonstrating the algorithm straight lane model is employed.

Lin et al proposed a way supported region of interest (ROI). The ROI first initialized and Sobel operator together with non-local maximum suppression is employed to seek out the sting pixels. After obtaining the sting image, an extended edge linking supported directional edge gap closing is completed to link those dashed lines in far vision field which are broken thanks to down-sampling. A raster scan is performed from the underside of image and also the place to begin of latest edge is revealed. Then a grip tracing is allotted and next pixel is added to the sting which is eight connected to the present pixel. For a given place to begin the tracing is completed in one orientation. The sting links are extended by adding a user specified number of pixels along the orientation to fill gaps. New pixels are then selected from the surrounding of start and end points. After this those edges with length but 15 pixels are eliminated. Then two edge link pairs are considered and if the gap between them satisfies the perquisite width of lane mark, then it's considered lane mark region. Next step is lane hypothesis verification, for that the color of lane-marks are checked. After the candidates are finalized, Hough transform is employed to work out ρ and θ values.

The lane detection algorithm is presented by Zhou et al has got three modules: lane model generation, parameter estimation and matching. The lane is modeled using the center line and three parameters lane width, original orientation and also the road curvature. The parameters within the model is reduced using middle line model which provides the starting

location, orientation and curvature. Using the lane width the ultimate lane models is obtained. The algorithm is targeted on the near field of view. For estimating the lane model parameters vanishing points are to be detected. The vanishing point detected supports Gabor texture analysis, which helps with estimating the orientation of each and every pixel in ROI. Finally of these pixel orientations vote for locating the vanishing point. Each pixel votes for quite one vanishing point thanks to the presence of artifacts. A Gaussian model is employed to optimize the likelihood and procure one vanishing point. After vanishing point detection is done, the width and orientation of lane is then estimated. For that, the road boundaries are detected using both the canny edge detector and Hough transform. Each lane model candidate have different curvature. To estimate the curvature a model matching algorithm is employed. A Gabor filter is employed along each candidate and also the one with maximum votes is that the best fit.

Daigavane et al proposed a lane detection method supported ant colony optimization (ACO). Originally, the input image is resized to 255 X 255 pixels for reducing the time taken in computation. The three channel RGB image is then converted to a single channel grayscale image. Then, median filter is employed to strain noise and to preserve the perimeters. Next the perimeters are detected using canny edge detector. The output binary image is used as input to the ant colony optimization. It generates the extra edge information that's lost in canny edge detection. ACO uses variety of ants which move supported the intensity variation within the image. Employing a probabilistic approach, each ant moves until it reaches another line segment. After those edges are properly connected Hough transform is employed to seek out straight lines. The lines cherish highest peaks in Hough space is extracted as lanes.

Guo et al develops a way where the input image has to first be converted to inverse perspective image. Next multiscale lane detection is completed in parallel on either side of image. To seek out the similarity of corresponding pixels in either sided of the road Normalized cross correlation (NCC) is completed. To test whether the detected lane markings are painted or not machine learning algorithm is employed. ANN classifier with two layers and 7 hidden nodes is applied on a tiny low image patch of 9X3 windows around each and each preserved road marking pixels. Integrating the intensity and geometry cues a weighted graph is built, with weight cherish the arrogance of a pixel to be a lane point. The lane boundary is estimated using particle filter, for that to happen, the Catmull Rom splines model is to be deployed. It’s suitable for curved lanes, lane changes, splitting and merging lanes. Y.C. Leng, et al proposes a lane detection system for urban roads. The sides are detected using Sobel operator. Then Hough transform is used to detect the straight lines. Within the road image lanes appear to intersect. At different height of the image, width of lanes will also differ. The minimum width of the

lanes is defined as minw and maximum lane as maxw. The width of a lane at each region wi (i=0,1,...4) must always between minw and maxw. For every left and right lane candidate an identical is finished supported the width of every candidate pair. If the width of pair in several height doesn't satisfy the factors is eliminated. After the extraction of left and right boundaries lane departure are often determined by position of lane boundary.

Liu et al. presents an improvement of traditional particle filter. It's suitable for linear parabolic lane model. Initially, the Inverse Perspective mapping (IPM) of images are obtained. Certain descriptors are used like color, position, gradient are accustomed detect lane markers in IPM image. The probabilistic distribution of line parameters is modeled as multi-kernel density and three candidates are found using Statistical Hough Transform. SHT are often extended to detect parabolic structures. Then the vertical edges are discovered using Sobel edge extraction. The quantity of particles depends on the dimension of the particle state. The algorithm is finished in hierarchical manner. The linear part is estimated first then the parabolic part is estimated. The linear part is assigned more particles than the parabolic part. to create the algorithm robust, a relentless percentage of initialization samples is introduced in every iteration. The linear part estimation is finished in two steps. First the state is predicted employing a stochastic process probability model. The second step is re-sampling employing a specific observation. Only the particles with high weight values are kept. In similar way particles are estimated for parabolic part also. If the lanes in vision field have high curvature, then the estimation error increases.

Borkar et al. developed a technique supported the parallel nature of lane markers. First Inverse Perspective Mapping is performed. Then the IPM image is converted to gray scale image. Then the image is filtered using Normalized Cross Correlation. Now see a set of straight lines using Polar Randomized Hough Transform (RHT). Each best fitting lines have high scoring coordinates or peaks in $(\rho,\theta)$ space. To work out if two lines are parallel lines peaks with identical $\theta$ value are often paired. Usually the lane markers are parallel but thanks to some imperfections in lens, captured image variation in lane marker placements etc. it should not be parallel always. That the constraint of identical $\theta$ value must be loosened. This will be achieved by applying a tolerance window. The video is tested in real time videos and obtained good results. Common difficulties face in lane detection like presence of shadows neighboring vehicles and surface irregularities are greatly reduced during this approach. There's difficulty in detecting tired lane markers. Tran et al uses a horizontal line detection to seek out the sub-region in image which is under the horizontal line where the lanes are present. A vertical mean distribution method is employed to work out the primary minimum that occur within the upper curve. This minimum position is taken into account because the horizontal line position. In day time the

minimum and in already dark maximum search along the vertical mean curve is taken into account. Then on the underside sub-image canny edge detector is applied. Supported the brink fixed there are often lanes and noise candidate within the edge image. Then k-means clustering and RANSAC algorithm is employed to detect the lanes. After the left and right lane boundaries are obtained, the intersection between them is employed as a horizontal line position for next frame of sequence. The algorithm concentrate on current lane only. When there's a crossing then also detection isn't correct.

Gopalan et al. models the contextual information shared between lane markers with the environment is modeled employing a pixel hierarchy feature descriptor. Here lane detection is finished employing a learning based approach. Things are often modeled as a two class detection problem comparable to lane marker and non -lane marker. Rather than using the local features of object in isolation here the knowledge shared by the item with its surrounding scene is employed. The pixel - hierarchy based descriptor is employed to hierarchically analyze several visual features like intensity patterns, textures and edges supported the region surrounding each pixel comparable to the item. Then from the contextual features the relevant features are selected employing a robust boosting algorithm. For tracking the lane markers in subsequent frames particle filter is employed. For tracking a static motion model is employed to represent the state of particles.

Liu et al. uses a top down approach for detecting lanes. First IPM is performed to get rid of the angle effect. The probability distribution of lane parameters is estimated by multiple kernel density, and therefore the uncertainties of visual cues are modeled using Gaussian kernels Statistical Hough Transform which was initially used for estimating straight lines is extended during this paper to detect parabolic and linear parabolic cases. After deriving likelihood function a partitioned particle filter (PPF) is employed for lane detection and tracking. The PPF uses prior knowledge to come up with variety of hypothesis or particles. Then the hypothesis is verified by derived likelihood function called measurement function using nearby data. Since PPF maintains multiple hypotheses of lane parameters it can handle several challenging situations like occlusions, shadows and lane changes. To detect lane markers, image descriptors supported appearance features is defined. Multiple kernel density is employed to estimate the probability distribution of lane parameters. The multiple kernel intensity is employed to estimate straight lines, parabolic and linear parabolic shapes. The PPF kernel model is strong against occlusion and noise.

Jung et al. introduce a lane detection algorithm which uses Haar like features to get candidate lane points. The image is split into two rectangular regions. Diagonally directional steerable filters are used, as lane marker appears diagonal thanks to perspective effect. Approximated steerable filter is

assigned to the Haar like features and optimal response is obtained. The left and right lanes are computed. Because the lanes are parallel, they'll converge at vanishing point. This hypothesis is verified to test the correctness of detected lanes. After detection lane departure are often determined supported the space between vanishing point and also the horizontal central line. If the space increases there's a lane departure.

Tan et al. developed a way for lane detection and classification. The lanes markers are detected supported a linear parabolic model. The lane marker pixels are assumed to possess higher intensity than the pixels on the pavement. A tiny low rectangular patch is employed to extract the statistical properties between the lane marker pixels and pavement related pixels. In each frame the consistency of each and every pixel within the patch such as lane marker with the distribution of pavement pixels are inspected to distinguish between pavement and lane marker pixels. Thereafter, the detection cascade classifier is used to classify the lane marker. Four binary classifiers are introduced. These can group the detected lane marking into five different classes: dashed, dashed-solid, solid dashed, single-solid, and double-solid.

Tsai et al. proposes a unique boundary determination algorithm for lane detection. It's supported local gradient direction. Sobel edge detector is employed for locating out the gradient magnitude and direction. Then two circular masks namely tracing mask and probing mask, are accustomed collect limited samples of gradient orientation. The initial gradient is decided supported the biggest histogram bin for orientation. The probing circular mask is employed to avoid deviation. A 3rd order polynomial fitting is additionally used.

Yoo et al. generates lane gradient maximized image from a color image supported linear discriminant analysis. This produces strong edging marks to the lane boundary in various kinds of illumination. Then canny edge detector is applied to urge edge image. After edge detection Hough transform is applied to urge initial lane detection. The Hough transform cannot represent curved lanes, so curve fitting is employed for detecting curved lanes. Three parameters of the quadratic curve are accustomed estimate the lane model. For the primary frame the training data is given manually and for remaining frames the training data are updated for adapting illumination changes.

Zhao et al. proposes a way which uses annealed particle filter for lane detection and tracking. A bar filter is employed because the lane marker are often described as black white-black bar like object. Since the lane marker usually have white or yellow color, the color cue is additionally used. To higher distinguish the colors RGB images are converted to HSV space. These cues are all processed separately. Rather than single factored step in conventional particle filter annealed particle filter an annealing is finished at

each enclose the video sequence. A stationary motion model is employed. The particles moved to the region of high probability. Within the correction step, the angle information of edge map is employed to live the weights of particles. The annealed particle filtering has three main steps including re-sampling, correction and weight normalization. For the lane detection and tracking a strong lane model is additionally used. It are often applied to both linear and curved roads.

Ozgunalp et al. uses Symmetrical Local Threshold (SLT) algorithm. It's supported dark - light-dark transition property of lanes. For every input point the common intensity value of all pixels to the left (within a range) in same row is calculated and, then average intensity is calculated for the correct hand side. If the intensity of pixel is bigger than both left and right averages then it's considered as a lane feature point and labelled within the feature map. Then a connected component analysis is applied. One either side of the feature points are hunted for International Journal of engineering & Information Technology (IJCSIT) Vol 7, No 4, August 2015 76 lane borders. If orientations of both the borders are near one another, then the common of the angles is taken into account because the orientation of the detected lane border. Otherwise the purpose is eliminated from the feature map. To scale back the noise a distance transform is employed. The space of every feature point from the closest non feature point in feature map is calculated and a weight is assigned. Thus the lane markers have higher weight than the isolated pixels. Also the middle of lane marker have a high value. Next Hough transform is applied and rho values is calculated for theta values near the orientation of feature points thus the computation speed is increased. Both the left and right lane are detected in a very frame and subsequently these lanes are tracked. To detect the sides in ROI, Canny edge detector is employed. The straight lines are detected from the binary output of canny edge extractor using Hough transform. To eliminate effect of noise local maxima features are searched along the estimated lane boundary. Then RANSAC algorithm is applied to eliminate outliers. The ultimate local maxima features are fit into a line. Next Kalman filter is employed to trace the lanes in remaining frames.

Bottazzi et al. proposes a histogram based illumination invariant lane detection method. A dynamic region of interest (DROI) is defined employing a prior triangle model. First the histogram of the full image and road frame are calculated. The difference between the 2 is employed to search out the illumination changes. From the ROI lane markers are segmented. The algorithm uses Lucas Kanade tracking to trace the lanes.

Wang et al. features overall optimal threshold converting the input image into binary format. Inverse perspective mapping is carried out to prevent the perspective effect. Then K means clustering is performed to partition n samples to k clusters. When all the points in a cluster are considered as

control points, B-spline fitting is implemented to obtain lane marker. The region of interest is divided into two sections, near vision field and far vision field. The input image is converted into IPM image first. The local Hough transform is first applied only on the near field of vision to detect the straight lines. In the far vision field an improved river flow method is used to extend the points found in near vision or curve lines that may be detected in the previous frame.

*(Hillel, 2006, Recent Progress in Road and Lane Detection - A survey p55-61)*

## CURRENT SYSTEM ANALYSIS

Traffic accidents have become one of the most serious problems in today's world. Increase in the number of vehicles, human errors towards traffic rules and the difficulty to oversee situational dangers by drivers are contributing to the majority of accidents on the road.

Lane lines are road markings that indicate the correct and safe passage of road vehicles. These not only ensure an orderly manner of movement but also a safe way for all users to a certain extent. In humans, the markings on the road are observed and the motorists are expected to drive accordingly while observing their surroundings. That being said, it means there are two phases involved in this process. First is recognition, identifying that there are lane lines present. This can be hindered by certain factors such as debris or dirt on the roads preventing the drivers from easy sight. A practical example is heavy duty sand tipper trucks tend to leave spillage of sand on the roads which might cover up important road markings. Another factor is poor weather conditions such as heavy rain. A practical example is when a typhoon occurs, it is nearly impossible to see the lane lines and drivers are encouraged not to drive during such times. Flooded roads affect visibility of lane lines too. However, human error is one of the main causes of road traffic accidents as sometimes drivers may not concentrate enough and in some cases, the drivers may be sleepy and eventually make a mistake. To solve this experts have designed computer algorithms that are able to recognize these lanes lines through various forms of image processing. Not all problems are solved but a great portion of them are, particularly when human error is involved.

Recognition algorithms can be divided into three main approaches.

1. *Line Based Lane detection algorithm*:

To guarantee a high working speed for the system, we may choose the approach that uses simple straight lines to represent lane marking model. This straight-line choice is based on the assumption that, in common scenarios, the lane markings appear on ROI do not have high curvature and can be adequately approximated by straight lines. After that, Lane Detection stage is realized by a technique called line normal random sampling. Finally, a Particle filter is exploited to track the positions of lane markings in tracking stage, as is shown in figure 1.1.

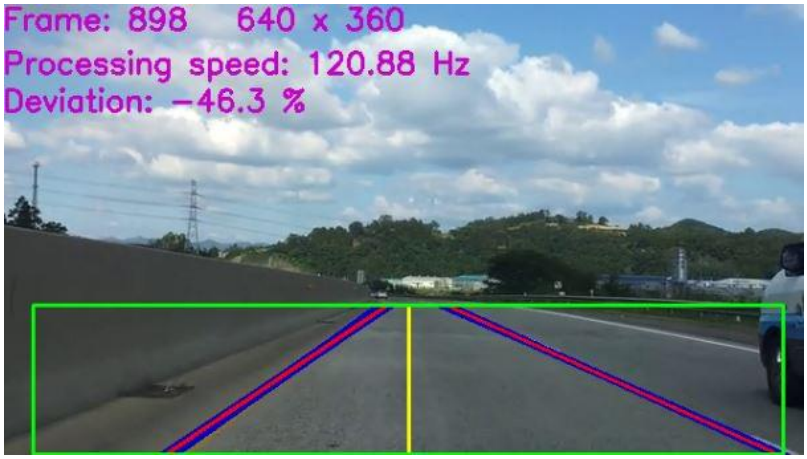



Figure 1.1: The final result of Line-based Lane Detection Algorithm

2. *Spline Based Lane detection algorithm:*

This approach uses more complex model that is cubic spline with four control points to represent lane markings. Particle Filter algorithm is still utilized for lane tracking with multi-dimensional particles corresponding to four control points of the cubic spline. Besides, the Inverse Perspective Mapping (IPM) technique is applied in Pre-processing stage to improve the result of the algorithm, as is shown in figure 1.2.


Figure 1.2: The final result of Spline-based Lane Detection Algorithm

### 3. *Hough Based Lane detection algorithm*

This algorithm has a totally different approach which models lane markings by straight lines and uses Hough Transform integrated with Kalman Filter in lane-marking detection and tracking. This approach can help lane detection system achieve higher speed with an adequate accuracy in common scenarios.
Hough-based Lane Detection Algorithm uses the same Pre-processing stage as Line-based Lane Detection Algorithm. In this paper, we focus on this algorithm and how it can be implemented, as is shown in figure 1.3.


Figure 1.3: The final result of Hough-based Lane Detection Algorithm

## FUNCTIONS OF THE LANE LINES REONITION SYSTEM

A lane lines detection system is a technology that is capable of detecting lanes lines on the road from images and video feed from an input source (camera).

I. This system is able to recognize both yellow and white road markings. There are two main algorithms that are used in lane lines detection but ultimately, it all comes down to image processing and detection of sharp changes in contrast. Digital image processing is the use of a digital computer to process digital images through an algorithm. As a subcategory or field of digital signal processing, digital image processing has many advantages over analog image processing. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and distortion during processing. In a lane detection system, a

camera is placed behind the wind shield of the vehicle and images/video of road are captured.

II. The lines on the road are interpreted and lanes are identified. Whenever the vehicle moves out of a lane unintentionally, a warning may be given to the driver. Lane line detection is the initial step of improving road safety. There are two methods for lane detection; feature based approach and model based approach.

- The feature based approach use low level features such as edges.
- Model based approach use geometric parameters for detecting lanes.

Lane detection system poses many challenges and issues which includes lane appearance diversity, variation in clarity of image, changes in visibility conditions. We have to investigate the type of lane behaviors and the challenges in tracking to solve such lane detection and tracking problems. Usually yellow and white colors are used for lane markers but reflectors lane are marked with special colors. The number and width of lanes varies with countries.

There can be issues related to clarity in images due to the presence of shadows. The lane markers may be occluded by the nearby vehicles. When the vehicle comes out of a tunnel, there is an abrupt change in illumination. So the clarity of image is affected by extreme illumination too. The visibility of the lane markers decrease due to various weather conditions such as rain, fog, snow. The visibility can be less in night condition

# CHAPTER II: SYSTEM DEVELOPMENT ENVIRONMENT AND RELATED TECHNOLOGY

In order to develop any system, before even starting designing, developers need to know the system platform and other configurations needed. That way, they know what language will be used and how to link all the technologies together in order to complete the project and ensure it works efficiently.

## DEVELOPMENT ENVIRONMENT

### Computer Vision

Computer vision is an interdisciplinary field, in which we are attempting to "describe the world that we see in one or extra photographs and to reconstruct its properties, such as shape, illumination, and coloration distributions". In different words, pc vision provides us the appreciation of the scene. The various domains of computer vision are shown in figure 2.1 below:

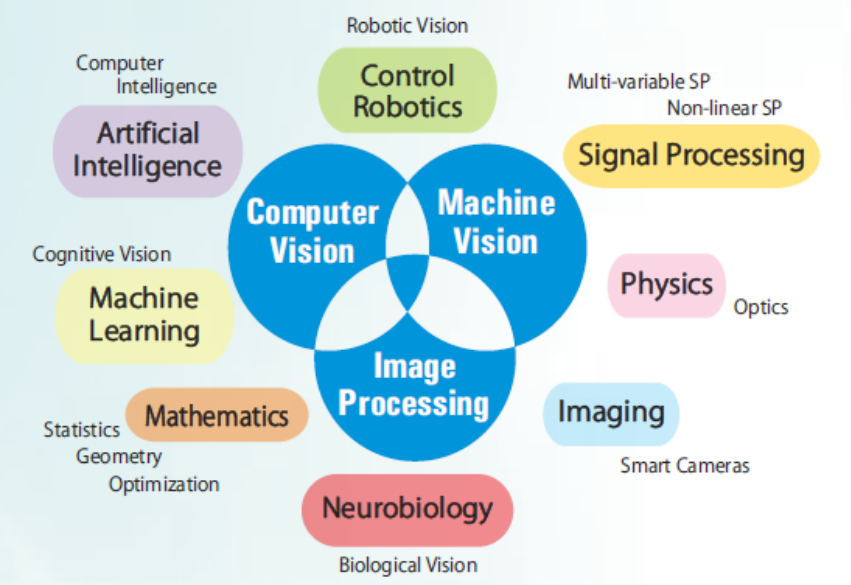
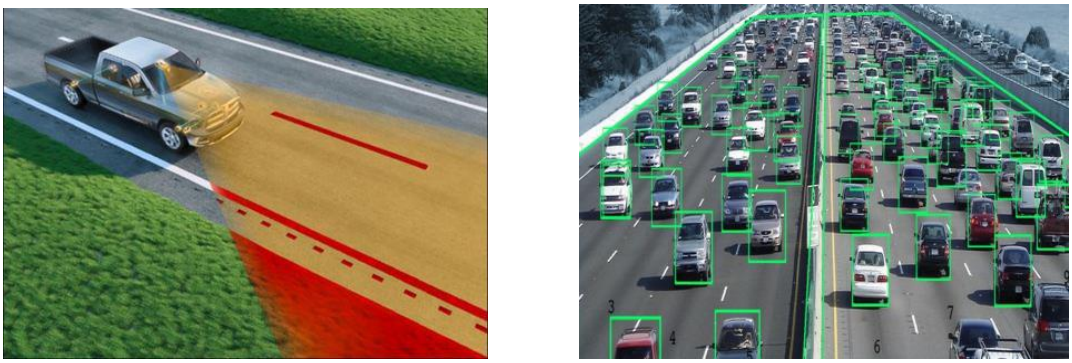



Figure 2.1: Domains of Computer Vision
http://miracleeyegroup.com/imaging-solutions/

In general, machine vision duties consist of techniques for:
• Acquiring: How do the sensors, for instance cameras, gather photos of the surrounding world? The way those pictures encode traits of the scene, such as vertices, edges, materials, illumination, etc.
• Processing: Based on how the data about the surrounding world is encoded, the photos are processed by using applying quite a few strategies of sign processing.
• Analyzing: What are the techniques or algorithms used to extract meaningful information from the processed pictures and provides the descriptions of the scene.
• Understanding: How are the descriptions of the objects and surrounding world stored and represented?

Nowadays, pc vision is being used extensively in a number types of applications in real life, for instance, optical personality attention (OCR), scientific imaging, car safety, surveillance, etc. Some of the applications are shown in figure 2.2 below:



a) b)

Figure 2.2: Some applications of computer vision in real life.
a) Automotive safety
b) Surveillance and traffic monitoring

## IMAGE PROCESSING

Image processing is a type of signal processing, which has an image, sequence of pics or video frames as the input, while the output is processed images, possibly in a special format. Image processing is often exploited in computer imaginative and prescient to manipulate digital images.
Vision can be considered the most necessary phase of human appreciation and it performs the identical function in computing device sensing. However, not like human, machines "see" the world as digital pictures of 2D or 3D scenes produced through sensors. The 2D digital image, which, in many cases, represents a projection of a 3D scene, can also be described as a two-dimensional array of intensity samples called pixels. Pixel values usually represent gray levels, colors, opacities, etc. 2D digital photo can be processed and manipulated with the aid of pc programs to produce useful records about the world.

The array of pixels is shown in figure 2.3 below:

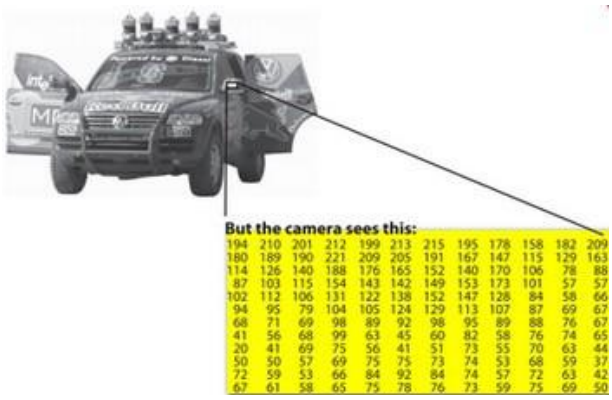



Figure 2.3: Digital image

Several methods in digital photo processing are noise removal, picture blur, image sharpening, object recognition, etc.

Image processing basically consists of the following three steps:

- Importing the image by photo acquisition tools;
- Analyzing and manipulating the photo;
- Output in which end result can be altered image or report that is based on image analysis.

There are two types of techniques used for image processing namely, analogue and digital image processing. Analogue photograph processing can be used for the tough copies like printouts and photographs. Image analysts use a number fundamentals of interpretation whilst the usage of these visible techniques. Digital picture processing methods help in manipulation of the digital images by using the use of computers. The three widely wide-spread phases that all kinds of statistics have to undergo while the usage of digital method are pre-processing, enhancement, and display, facts extraction.

## Image

A picture refers a 2D mild depth characteristic f(x, y), where(x, y) denotes spatial coordinates and the value of f at any point (x, y) is proportional to the brightness or gray degrees of the photograph at that point. A digital picture is a photograph f(x, y) that has been discretized both in spatial coordinates and brightness. The elements of such a digital array are called image factors or pixels.

### A simple image model

To be suitable for pc processing, a photograph f(x, y) will have to be digitalized each spatially and in amplitude. Digitization of the spatial

coordinates (x, y) is known as image sampling. Amplitude digitization is referred to as grey-level quantization.

The storage and processing necessities enlarge hastily with the spatial decision and the variety of gray levels.

Example: A 256 grey-level picture of measurement 256x256 occupies 64k bytes of memory.

## Types of image processing

1) Low level processing
2) Medium level processing
3) High level processing

Low level processing means performing basic operations on images such as reading an image resize, resize, image rotate, RGB to grey level conversion, histogram equalization etc., The output image obtained after low level processing is raw image. Medium level processing means extracting regions of interest from output of low-level processed image. Medium level processing deals with identification of boundaries i.e. edges .This process is called segmentation. High level processing deals with adding of artificial intelligence to medium level processed signal.

Low stage processing means performing fundamental operations on snap shots such as reading a photo resize, resizing, photo rotation, RGB to grayscale conversion, histogram equalization etc., The output picture after low stage processing is a raw image. Medium stage processing is generally extracting regions of activity from output of low-level processed image. Medium stage processing deals with identification of boundaries i.e. edges .This action is called segmentation. High stage processing handles adding artificial intelligence to medium degree processed signals.

There are various image formats available in different sizes and quality, as shown in the table 2.1 below:

*Table 2.1 shows various image formats.*

| Format name | Full name | Description | Recognized extensions |
|---|---|---|---|
| TIFF | Tagged Image File Format | A flexible file format supporting a variety image compression standard including JPEG | .tif, .tiff |

| | | | |
|---|---|---|---|
| JPEG | Joint Photographic Experts Group | A standard for compression of images of photographic quality | .jpg, .jpeg |
| GIF | Graphics Interchange Format | Frequently used to make small animations on the internet | .gif |
| BMP | Windows Bitmap | Format used mainly for simple uncompressed images | .bmp |
| PNG | Portable Network Graphics | Compresses full color images with transparency(up to 48bits/p | .png |

## OpenCV in Python

Originally an Intel lookup initiative, OpenCV is a cross-platform open source pc imaginative and prescient library, ordinarily employed for its real time image processing performance. It ambitions to supply well tested, optimized, open supply implementations of, state of the art photograph processing and computer vision algorithms. The library is written in C, making sure quickly and transportable code, and has been compiled for various embedded platforms.
Multiple language bindings are available for OpenCV, such as OpenCVDotNet and EmguCV.
Multiple bindings to OpenCV (OpenCV Python, and PyCV) have been created for Python, as well as the bindings automatically built with SWIG. More tools such as GPUCV have been created for OpenCV that ensure the use of graphics hardware to enhance CV performance on the GPU.
Many of the image processing techniques in this thesis, for example image reading, grayscaling, thresholding, etc., are achieved by using Open Source Computer Vision Library (OpenCV).


*OpenCV logo http://opencv.org/*

We do not have to worry about extraordinary codecs (mp4, avi, and more) as OpenCV does all the heavy lifting for you. It affords different ways to study a video. You can read a live video the using of your dashcam or use an external USB camera or read a saved video on your computer with the usage of an object of the VideoCapture module. The VideoCapture () constructor either an integer or the title of a file. The integer argument is the ID of the digital camera connected to the laptop.

## *OpenCV Functions*

### Changing the color space

OpenCV uses the cvtColor () function to facilitate the conversion of an image from one color space to another. It takes the image as the input and also the color space conversion code. The following are a few conversion codes:

- COLOR_BGR2GRAY
- COLOR_BGR2HSV
- COLOR_HSV2BGR
- COLOR_BGR2YUV
- COLOR_GRAY2BGR

The following is an example code for the conversion of a BGR image to a grayscale image. We can change the second parameter of the cvtColor () function by any of the earlier mentioned parameter values:

```
>>> import cv2
>>> img = cv2.imread ("image.jpg")
>>> gray = cv2.cvtColor (img, cv2.COLOR_BGR2GRAY)
>>> cv2.imwrite ("gray_image.jpg", gray)
>>> cv2.imshow ("image", gray)
```

The following figure 2.4 is the output of the preceding code:

*Figure 2.4: The original image is on the left and the output is on the right*

## Thresholding

OpenCV has an inbuilt threshold () function, which takes a grayscale image, threshold value, and new value to be assigned if the value is greater than the threshold and type of thresholding as input. The types of thresholding are:

- cv2.THRESH_BINARY
- cv2.THRESH_BINARY_INV
- cv2.THRESH_TRUNC
- cv2.THRESH_TOZERO
- cv2.THRESH_TOZERO_INV

The following code is an example for thresholding:

```
>>> import cv2
>>> img = cv2.imread("image.jpg")
>>> gray = cv2.cvtColor (img, cv2.COLOR_BGR2GRAY)
>>> new_img = cv2.threshold (gray,120,255,cv2.THRESH_BINARY)
>>> cv2.imwrite ("thresholding.jpg", new_img[1])
>>> cv2.imshow ("thresholding", new_img[1])
```

In the given example, all the pixels in the grayscale image, which are above 120, are set to white and the others are set to black, as shown in Figure 2.5, which is the output of the preceding code:

*Figure 2.5: The original image is on the left and the output is on the right*

## Gaussian blur

There is an inbuilt function, GaussianBlur (), which takes the image, dimension of the kernel, and standard deviation as input. If the input standard deviation is zero, then the standard deviation is calculated from the size of the kernel:

```
>>> import cv2
>>> img = cv2.imread("image.jpg")
>>> new_img = cv2.GaussianBlur (img,(5,5),0)
>>> cv2.imwrite ("gaussian_blur.jpg", new_img)
>>> cv2.imshow ("gaussian_blur.jpg", new_img)
```

The result of applying the Gaussian blur filter is as shown in figure 2.6:

*Figure 2.6: The original image is on the left and the output is on the right*

## Sobel edge detection

The Sobel () function in OpenCV can be used to find the edges of an image. The Sobel () function takes the image, output image depth, order of derivative in x and y direction, and the size of the kernel as input. In the following code, we will look at both horizontal and vertical edges:

```
>>> import cv2
>>> img = cv2.imread("image.jpg")
>>> gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
>>> x_edges = cv2.Sobel (gray,-1,1,0,ksize=5)
>>> cv2.imwrite ("sobel_edges_x.jpg", x_edges)
>>> y_edges = cv2.Sobel (gray,-1,0,1,ksize=5)
>>> cv2.imwrite ("sobel_edges_y.jpg", y_edges)
>>> cv2.imshow ("xedges", x_edges)
>>> cv2.imshow ("yedges", y_edges)
```

In the code, we have taken the order of derivative in x direction equal to 1 and the order of derivative in y direction equal to 0 for Sobel in the x direction and the opposite for the y direction. Also, we have used -1 for the image depth, which means the image depth of the output will be the same as the input. The output of Sobel in the x direction is as shown in figure 2.7:

*Figure 2.7: The image on the left is the input image and the image on the right is the output of Sobel in the x direction*

The following figure is the output in the y direction, as shown in figure 2.8:

*Figure 2.8: The image on the left is the input image and the image on the right is the output of Sobel in the y direction*

## Canny edge detector

The Canny () function in OpenCV was selected to be applied in this project. The Canny () function takes the image, min and max threshold values, and the size of

the kernel as input. The following code shows how to use canny edge detection in Python:

```
>>> import cv2
>>> img = cv2.imread ("image.jpg")
>>> gray = cv2.cvtColor (img, cv2.COLOR_BGR2GRAY)
>>> edges = cv2.Canny (gray, 100, 200, 3)
>>> cv2.imwrite ("canny_edges.jpg", edges)
>>> cv2.imshow ("canny_edges", edges)
```

The output of the preceding code is as shown in figure 2.9 below:

*Figure 2.9: The image on the left is the input image and the image on the right is the output of the canny edge detector*

(Nguyen, 2017)

## SYSTEM ARCHITECTURE ANALYSIS

The pipeline of this system can be simplified in the flowchart diagram 2.1.1 below:

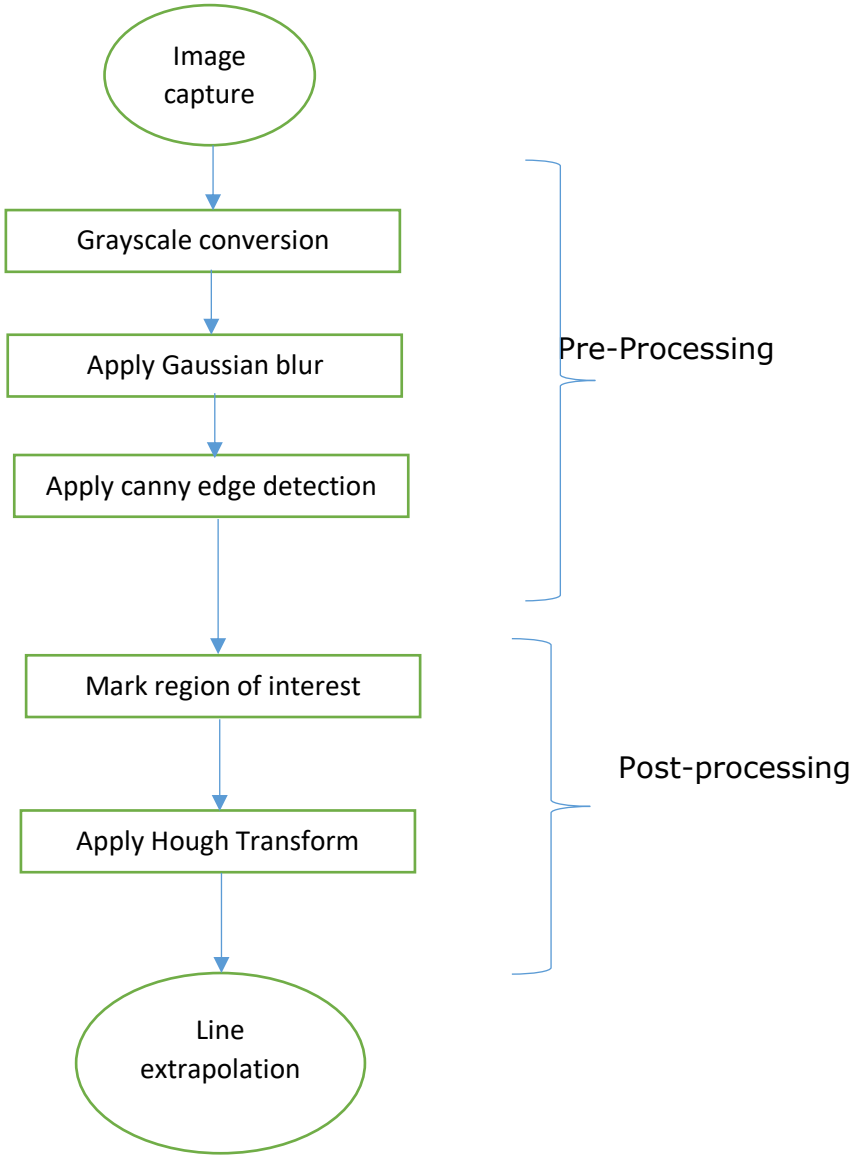



*Figure 2.1.1 flowchart of the detection process*.

## PRE-PROCESSING:

The first step to working with our images are to convert them to grayscale. This can be a critical step to using the Canny Edge Detector inside OpenCV.

During conversion to grayscale, we are collapsing 3 channels of pixel value (Red, Green, and Blue) into one channel with a pixel value range of [0,255]. It's easier to control one channel image than a multichannel image. Apply Gaussian blur: so as to accurately capture the sides within the image, image noise must be filtered out and this could be done by smoothening the image employing a Gaussian filter. Apply canny edge detection on smoothened image: This function is employed to spot high gradient pixels within the image. Globally, lane lines are usually white or yellow. The canny function computes gradients all told directions of the blurred image and traces the strongest gradients as white or yellow pixels. Pixels are compared to preset low and high thresholds. The gradients larger than the high threshold is accepted as a grip pixel. If it's below, it's rejected. A recommended low to high threshold ratio of 1:2 or 1:3.

## POST PROCESSING

Mark the region of interest: We don't want our car to be taking note to anything on the horizon, or maybe within the other lane. Our lane detection pipeline should specialize in what's before of the car. To do that, we are visiting create a mask called our region of interest (ROI). Everything outside of the ROI are set to black/zero, so we are only working with the relevant edges. During this image, a polygonal region is marked, representing the precise region of interest ROI. This region corresponds to the view of the camera, capturing the road and appropriate markings. Hough transform is applied: The Hough transformation converts an "x vs. y" line to a degree in "gradient vs. intercept" space. Points within the image will correspond to lines in Hough space. An intersection of lines in Hough space will henceforth correspond to a line in the Cartesian space. Using this method, we will find lines from the pixel outputs of the canny edge detection output. The Hough transform gives us small lines supported the intersections in Hough space. We will take this information and construct a worldwide left lane line and a right lane line. We do that by separating the little lines into two groups, one with a positive gradient and also the other with a negative gradient. thanks to the depth of the camera's view, the lane lines slant towards one another as we go further comprehensive, in order that should have the alternative gradients. At this point, we take the common gradient and common intercept values for every group and use it to construct our global lane lines from there. The lane lines will then be extrapolated to the sting detected pixel with the minimum y-axis coordinate and also the pixel with the most y-axis coordinate. When images are to be utilized in different areas of image analysis like beholding, it's important to scale back the quantity of information within the image while preserving the important, characteristic, structural information. Edge detection makes it possible to scale back the quantity of information in a picture considerably. However the output from a grip detector continues to be an image described by its pixels. If lines,

ellipses and then forth may well be defined by their characteristic equations, the quantity of information would be reduced even more. The Hough transform was originally developed to acknowledge lines, and has later been generalized to hide arbitrary shapes. This worksheet explains how the Hough transform is in a position to detect (imperfect) straight lines. The Hough space representation of Lines within the Hough Space Lines will be represented uniquely by two parameters.

Often, the shape in formula 2.1 is employed with parameters a b as shown below:

*Formula 2.1* $y = a \cdot x + b$

This form, however cannot be used to represent vertical lines. Therefore, the Hough transform uses the shape in formula 2.2, which might be rewritten to formula 2.3 to be kind of like formula 2.1. The parameters θ and r is that the angle of the road and also the distance from the road to the origin respectively. Formulae 1.2 and 1.3 are shown below:

*Formula 2.2* $r = x \cdot \cos\theta + y \cdot \sin\theta \Leftrightarrow$

*Formula 2.3* $y = (-\cos\theta/\sin\theta)\, x + r/\sin\theta$

All lines will be represented during this form when $\theta \in [0, 180]$ and $r \in R$ (or $\theta \in [0, 360]$ and $r \geq 0$). The Hough space for lines has therefore these two dimensions; θ and r, and a line is represented by one point, resembling a singular set of parameters (θ0, r0). The line-to-point mapping is shown below in figure 1.1.2:

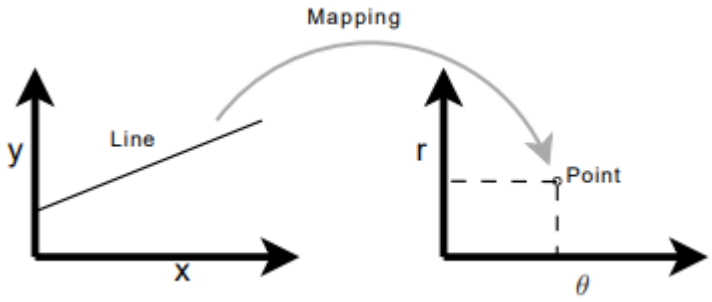



Figure 2.1.2: Mapping of one unique line to the Hough space.

## SYSTEM ANALYSIS

Detecting straight lines in this system can be divided into the following four main steps:

1. Edge detection, i.e. using canny edge detector.
2. Mapping the edge points onto the Hough space and storing in an accumulator.
3. Interpreting the accumulator to produce lines of infinite length. The interpretation is done by thresholding and possibly some other constraints.
4. Finally, conversion of infinite lines to finite lines.

   The finite lines can then be superimposed back onto the original image. The Hough transform itself is performed in step 2.

## Hough Space

The Hough transform is a parameter estimation method that uses poling to obtain a desired detection object and is suitable for lane detection. The essence is to map the coordinate space in the image into the Hough parameter space, and thereafter analyze the Hough space data by point-line duality to detect the geometry. The Hough transform can be used in the detection of curves, circles, lines, etc. The most widely used of Hough transform is to detect straight lines in a picture or image frames.

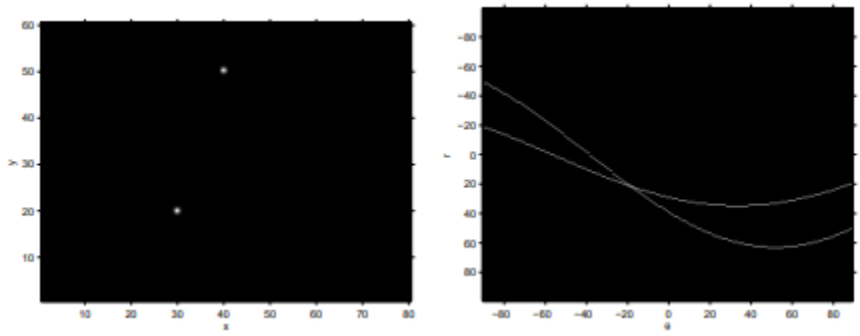
In the commonly known XY plan, XY space lines are lines and points are points. But in Hough space, *lines* correspond to *points in XY space* and **points** correspond to **lines in XY space**. This is what our pipeline for this process looks like:

1. Pixels are considered points in XY space
2. hough_lines() will transform these points into lines inside of Hough space
3. When these lines intersect, there is a **point of intersection** in Hough space
4. The point of intersection corresponds to a **line in XY space**

## Transformation to Hough Space

The Hough transform takes a binary edge map as the input and then it attempts to locate the edges placed as straight lines. The idea of the Hough transform is basically that every edge point in the edge map is transformed to all possible lines

that could pass through that point. Figure 2.1.3 illustrates this for two points. Figure 2.1.3 is shown below:



(a) Points p0 and p1 (b) All possible lines through p0 and/or p1 in the Hough space.

Figure 2.1.3: Transformation of two points to two lines in the Hough space transformation. The intersection of the lines indicates the line that pass through both p0 and p1.

A typical edge map includes many points, but the principle for line detection is the same as illustrated in 2.1.3 for two points. Each edge point is transformed into a line in the Hough space, and the areas where most Hough space lines intersect are interpreted as true lines in the edge map.

## The Hough Space Accumulator

To identify the areas where most of the Hough space lines intersect, an accumulator covering the Hough space is employed. When a detected edge point is transformed, bins within the accumulator is incremented for all lines that might submit to that time. The resolution of the accumulator is the basis of the precision with which lines will be detected. During this worksheet a resolution of 1 pixel for r and 1 degree for θ has been used. In general, the amount of dimensions of the accumulator corresponds to the amount of unknown parameters within the Hough transform problem. Thus, for ellipses a 5-dimensional space is required (the coordinates of its center, the length of its major and axis, and its angle). For lines 2 dimensions suffice (r and θ). This is often why it's possible to visualize the content of the accumulator.

## Detection of Infinite Lines

Infinite lines are detected by interpreting the accumulator when all edge points have been transformed. The most basic way the detect lines is to set a threshold for the accumulator, and interpret all values above the set threshold as a line. The

threshold could for instance be 50% of the largest value in the accumulator. This approach may occasionally be enough, but for many cases additional constraints have to be applied. As it is shown in the figure, many entrances in the accumulator around one true line in the edge map will have large values. Therefore a simple threshold has a tendency to detect several (almost identical) lines for every true line. To prevent this, a suppression neighborhood could be defined so that the two lines would be significantly different before both are detected. Figure 2.1.4 is shown below:

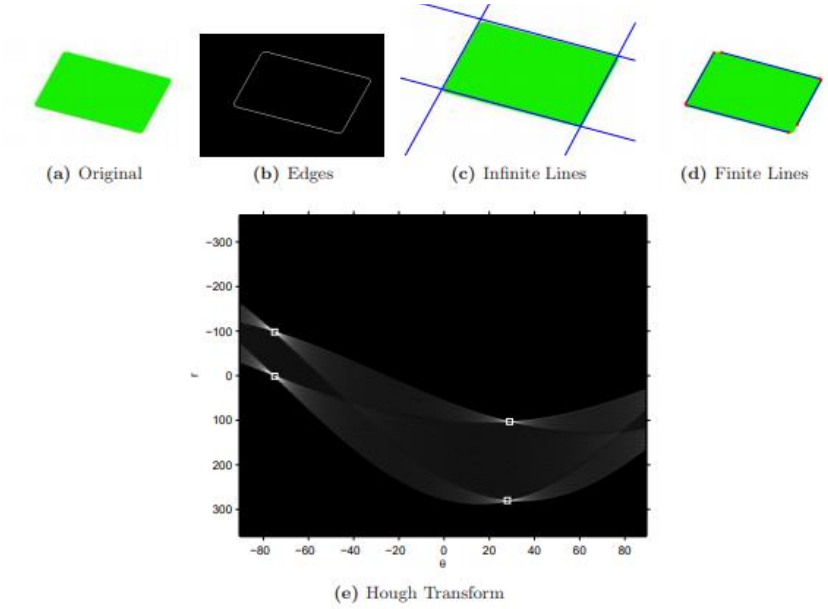



Figure 2.1.4: Line detection using the Hough transformation. The lines detected in the source image (Figure 2.1.4a) are marked with white boxes in the Hough transform

## Finite Lines

The Hough transform algorithm detects lines given only by two parameters r and θ and no additional data with regards to length. Thus, all detected lines are infinite long. If finite lines are desired, some additional analysis must be performed to see which areas of the image that contributes to every line. Several algorithms for doing this exist. One method is to store the coordinate information for all the points within the accumulator and use it to limit the lines. However, this might cause the accumulator to use way more memory. An alternative is to go looking along the infinite lines within the edge image to search out finite lines.

## Evaluation on Real Images

The figure below shows the Hough transform applied to a partly assembled pump from Grundfos. In Figure 2.1.5 (d), six true straight lines have been detected, while two detected lines does not correspond to true straight lines. The algorithm has been tricked by the many ellipses in the edge map. It is not immediately possible to avoid incorrect detections while preserving most of the true detections through tuning of the algorithm. Figure 2.1.5 is shown below:

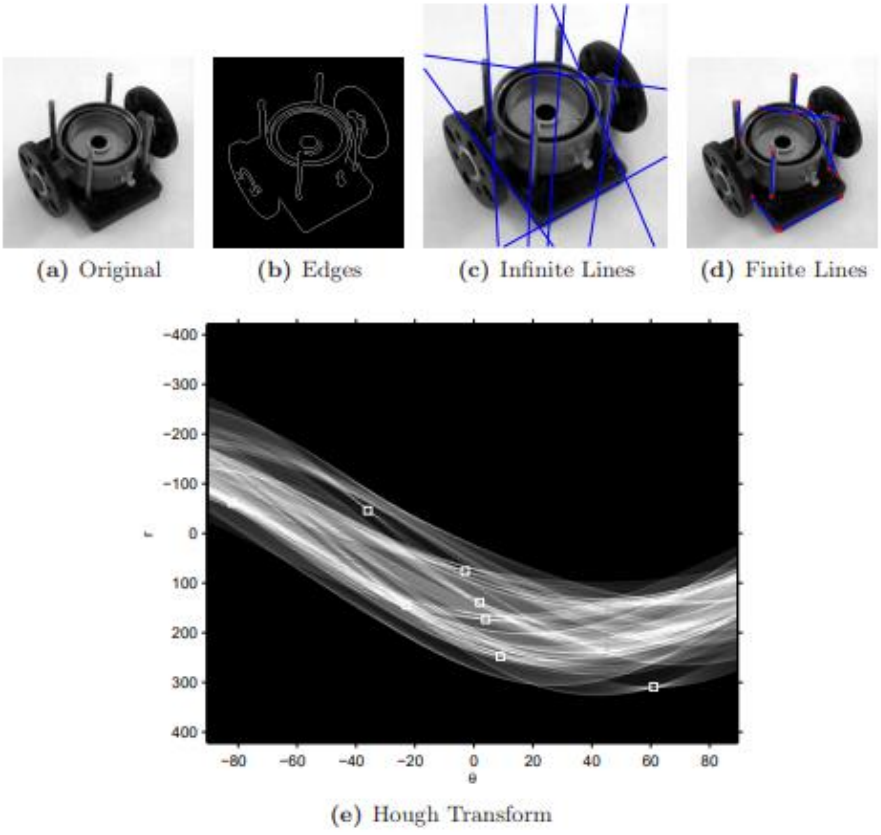



Figure 2.1.5: Line detection on a real image using the Hough transformation.

# CHAPTER III: SYSTEM ANALYSIS REQUIREMENT

## FEASIBILITY ANALYSIS

A **feasibility study** is an in depth analysis and evaluation of the entire scope of the proposed project. It is based on deep research on the project in order to support the process of decision making. Regarding this project, the feasibility study should at least provide the potential technical analysis, according to the background analysis of the system. Some aspects of feasibility study will include:

- Economic feasibility: Making sure the system is economically feasible by evaluating the system
- Technical feasibility: Checking the work conditions if both hardware and software meet the needs of developers. As for this system, Python 3 with OpenCV are capable and adequate in the development of the system.

## PERFORMANCE REQUIREMENTS ANALYSIS

The implementation of a software system requires passing through analysis to meet users need. As for this system, the main concerns are effectiveness and reliability, correctness and speed in other words.

For evaluating the overall performance of lane detection and tracking algorithms the result is compared with floor fact records set and a test is completed to decide whether there is true positive (TP) or false positive (FP) or false negative (FN) or true negative (TN). True positive happens when there exist a floor reality and it is detected with the aid of the algorithm. False positive happens when the algorithm detects a lane marker when there exist no floor truth. False negative takes place when ground truth exist in the photograph and the algorithm misses it. True negative takes place when there is no ground truth in the image and the algorithm is not detecting any. The most common metrics used for evaluating overall performance of lane detection algorithms are Precision, Recall, F-score, Accuracy, Receiver Operating Characteristic (ROC) curves and Dice Similarity Coefficient (DSC).

Precision is the fraction of detected lanes markers that are proper lane markers. Recall is the fraction of genuine lane markers that are detected. F-measure is the measure that combines precision and recall and is the harmonic suggest of precision and recall. Accuracy is the measure of how properly the true lane markers are effectively recognized and true negatives are excluded.

The metrics are represented as shown in the formulae 2.1 – 2.5. The ROC curve is acquired by way of plotting True Positive Rate (TPR) versus the False Positive Rate (FPR) for exceptional values of the extraction threshold. If vicinity under the curve is larger than the detection is good.

*Formula 3.1* Precision=TP / ( (T P +F P)

*Formula 3.2* Recall=TP/(TP+FN)

*Formula 3.3* F−measure=2*Precision*Recall/(Precision + Recall)

*Formula 3.4* Accuracy= (TP+TN) / (TP+TN+FP+FN)

*Formula 3.5* DSC=2*TP/((TP+FP+P)

Where TP is the number of true positives, FP is the number of false positives, TN is the number of true negatives, FN is the number of false negatives and P is the number lanes detected.

# CHAPTER IV: GENERAL SYSTEM DESIGN

## ENVIRONMENT SETTINGS

When trying to comprehend the whole concept of lane line detection and recognition system, the main questions that come forward are how to run the system and on which platform. Microsoft Windows 10 or android 6-10 as the optimal operating systems, Python 3 as the software. The system is working off of image processing, so image capture tools (camera) and graphic cards are essential to this system.

## SYSTEM FUNCTION STRUCTURE

The system function structure is a simple pipeline with each step consequentially occurring. The figure 4.1 below illustrates this;

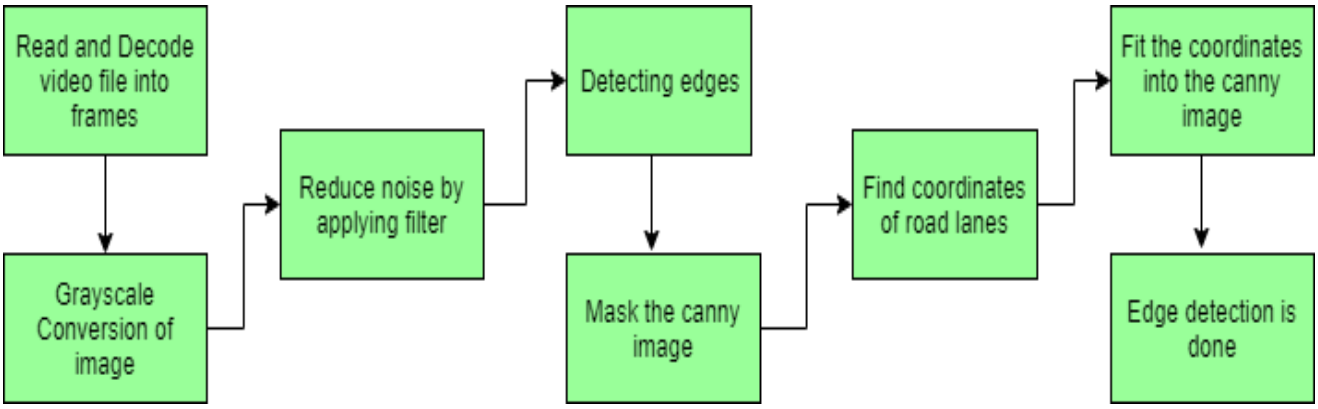



*Figure 4.1 system pipeline*

## IMPLEMENTATION ILLUSTRATION

In the first step, a video is read and decoded into frames. Singular frames are the unit used during the processing. A single frame is illustrated below in figure 4.2:

*Figure 4.2 showing a typical road and its lane markings.*

Next, a grayscale conversion is carried out as is shown in figure 4.3 below:

*Figure 4.3 grayscale image.*

In the case that the image frame is unclear, a noise reduction filter is applied using the discussed OpenCV functions. In this project, a Gaussian blur is applied.

Next, we will apply a mask to the image to return the pixels we're interested in, as is shown in figure 4.4 below:

*Figure 4.4 showing a mask of the image*

Next in the pipeline is application of canny edge detection to the image. The result of this operation is as show in figure 4.5 below:

*Figure 4.5 showing the result of canny edge detection.*

The coordinates of the road lanes are to be preset, this is known as the region of interest. This is shown in the figure 4.6 below:

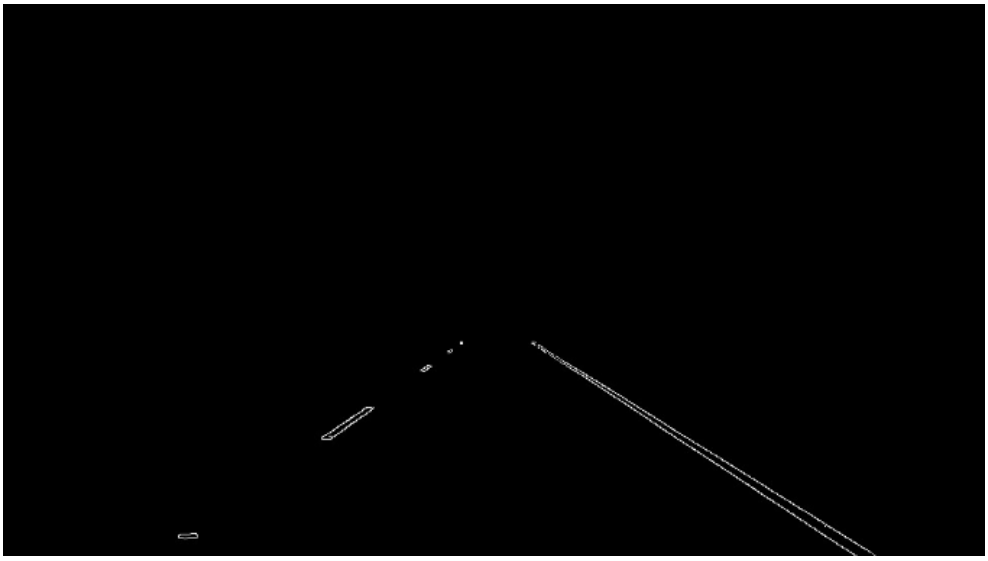


*Figure 4.6 showing the region of interest.*

A Hough transform is carried out on the image and the result image is as shown in figure 4.7 below:

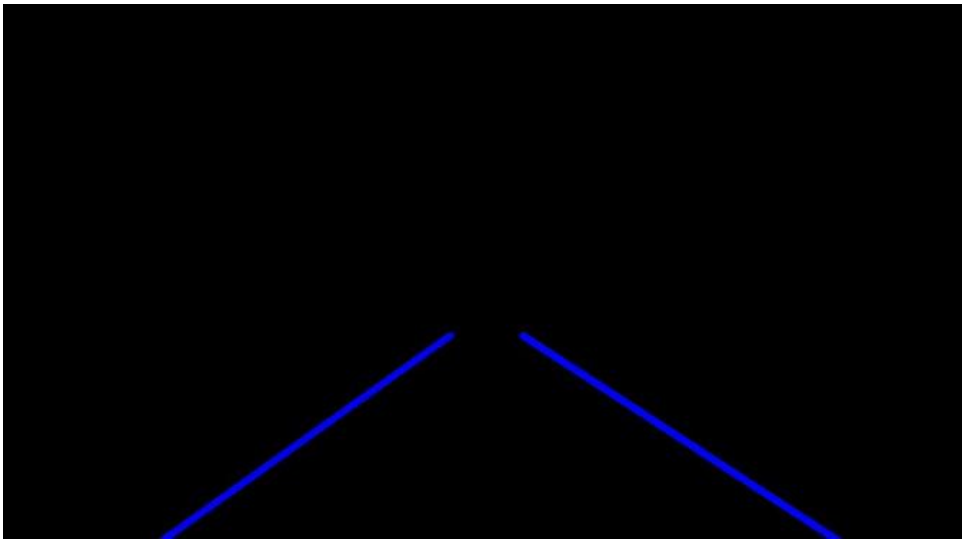


*Figure 4.7 showing the result of Hough transformation.*

Once we have our two main lines, we can average our line image with the original, unaltered image of the road to have a nice, smooth overlay that will display the guidelines we want and need.

The resulting image is as shown in figure 4.8 below:

*Figure 4.8 showing averaged and smoothened lane lines.*

# CHAPTER V: DETAILED DESIGN AND IMPLEMENTATION

## PROGRAMMING METHOD

With preexisting functions, python and OpenCV made it easy to implement various image processing tools in order to achieve this system.

Python modules need to be imported using the code lines:

*import cv2*

*import numpy as np*

We need to determine which lines are in the left group and which are in the right group.

The obvious difference between the two groups of line segments is the direction of their slope, thus, their gradients. The slope of a line measures the angle of that line, horizontal lines will always a slope of 0 and a vertical lines will have a slope of ∞. Almost all of the lines we will detect will have a slope somewhere in between these values, because their angles span from the bottom of the image towards the center at the horizon.

Another important fact to take note of is that the direction of a line's slope describes whether the line is moving up or down as you travel along the line from left to right. A line with negative slope is said to be traveling downwards, while a line with positive slope is said to be traveling upwards. This is how slope direction works in normal coordinate systems where the origin is at the bottom left corner. In our coordinate system, however, the origin is in the top left corner, and so our slope directions will be reversed, with negative slopes traveling upwards and positive slopes traveling downwards.

In our images, the left lane markings all have a negative slope, meaning the lines travel upwards towards the horizon as we move from left to right along the lines. On the other hand, all of our right lane markings have a positive slope, traveling downwards towards the bottom of the image as we move along them from left to right. This will be the distinction we use to group the left and right lines. Furthermore, the lane markings appear extreme in slope, so we will not consider any line with a slope absolute value less than 0.5. This means that we'll reject any line which doesn't move quickly towards the horizon or the bottom of the image, leaving only lines which are nearer to vertical than horizontal.

This is achieved by the code segment below:

*def make_coordinates(image, line_parameters):*

```
    slope, intercept = line_parameters
    y1 = image.shape[0]
    y2 = int(y1*(3/5))
    x1 = int((y1 - intercept)/slope)
    x2 = int((y2 - intercept)/slope)
    return np.array([x1, y1, x2, y2])

def average_slope_intercept(image, lines):
    left_fit = []
    right_fit = []

    for line in lines:
        x1, y1, x2, y2 = line.reshape(4)
        parameters = np.polyfit((x1, x2), (y1, y2), 1)
        slope = parameters[0]
        intercept = parameters[1]
        if slope < 0:
            left_fit.append((slope, intercept))
        else:
            right_fit.append((slope, intercept))
    left_fit_average = np.average(left_fit, axis=0)
    right_fit_average =  np.average(right_fit, axis=0)
    left_line = make_coordinates(image, left_fit_average)
    right_line = make_coordinates(image, right_fit_average)
    return np.array([left_line, right_line])
```

Generally, canny () parses the pixel values on account of their directional derivative (i.e. gradient). What's left over are the edges — or where there is a steep derivative in at least one direction. As canny () computes the gradient, we will have to provide

thresholds for this. The most commonly recommended are a low to high threshold ratio of 1:2 or 1:3. In this task, we use 1:3. This is illustrated in the code segment below:

```
def canny(image):
    gray = cv2.cvtColor(lane_image, cv2.COLOR_RGB2GRAY)
    blur = cv2.GaussianBlur(gray, (5, 5), 0)
    canny = cv2.Canny(blur, 50, 150)
    return canny
…

def display_lines(image, lines):
    line_image = np.zeros_like(image)
    if lines is not None:
        for x1, y1, x2, y2 in lines:

            cv2.line(line_image, (x1, y1), (x2, y2), (255, 0, 0), 10)

    return line_image
```

### *Region of interest ROI*

We do not need the car system to be paying attention to anything on the horizon, or even in the other lane. Our lane detection pipeline should focus on what is in front of the car only. To do that, we are to create another mask called our region of interest (ROI). Everything on the outside of the ROI will be set to black/zero. Hence, we are only working with the required edges.

```
def region_of_interest(image):
    height = image.shape[0]
```

```
    polygons = np.array([
    [(200, height), (1100, height), (550, 250)]
    ])
    mask = np.zeros_like(image)
    cv2.fillPoly(mask, polygons, 255)
    masked_image = cv2.bitwise_and(image, mask) #
    return masked_image
…
```

### *Image frame test*

After definition, all the functions of the whole process can be orderly arranged and summarized as is shown below:

```
image = cv2.imread('test_image.jpg')
lane_image = np.copy(image)
#canny_image = canny(lane_image)
#cropped_image = region_of_interest(canny_image)
#lines = cv2.HoughLinesP(cropped_image, 2, np.pi/180, 100, np.array([]), minLineLength=40, maxLineGap=5)
#averaged_lines = average_slope_intercept(lane_image, lines)
#line_image = display_lines(lane_image, averaged_lines)
#combo_image = cv2.addWeighted(lane_image, 0.8, line_image, 1, 1)
#cv2.imshow("result", combo_image)
#cv2.waitKey(0)
```

### *Live video test*

It is a few short lines to edit one frame at a time, to creating a rolling average and processing 40FPS. The code used to process a video is as shown in figure 5.1.1 below:

```
cap = cv2.VideoCapture("test2.mp4")
while(cap.isOpened()):
    _, frame = cap.read()
    canny_image = canny(frame)
    cropped_image = region_of_interest(canny_image)
    lines = cv2.HoughLinesP(cropped_image, 2, np.pi/180, 100, np.array([]), minLineLength=40, maxLineGap=5)
    averaged_lines = average_slope_intercept(frame, lines)
    line_image = display_lines(frame, averaged_lines)
    combo_image = cv2.addWeighted(frame, 0.8, line_image, 1, 1)
    cv2.imshow("result", combo_image)
    if cv2.waitKey(1) == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

## CHAPTER VI: CONCLUSION AND FUTURE DEVELOPMENTS

This paper presents a lane line detection algorithm under the regular conditions. The algorithm uses Gaussian blur, grayscaling, canny edge detection and Hough transform to detect and hence determine the distribution of lane lines. The experimental results show that the method can work effectively and accurately, using simple hardware and obtain good results, both in the case of complicated road information and simple road information.

However, this implementation technique evaluated has some shortcomings. This paper does not consider the impact of strong light and unclear road markings on the road environment. Another major area of concern is handling the region of interest. A change in the angle of view due to variations in road steepness could potentially affect the preset static region of interest.

In the future, we can combine various conditions to continue to improve lane detection as hardware components such as the cameras used and the processors improve. The application of Particle Filter brings the flexibility and greatly increases the accuracy of the system. This method is a promising direction for future development. Especially, by being integrated with the information of vehicle motion model, lane width, path planning and GPS, it can handle many challenging situations such as missing lane markings, changes in direction of the vehicle, sudden changes in lane width, crossroads etc. and be able to be applied to Automated Vehicle Control System in reality without significantly increasing in calculation.

## REFERENCES

[1] Mohamed Aly. Real time detection of lane markers in urban streets. IEEE Intelligent Vehicles Symposium, pages 7{12, June 2008.
[2] N. Apostolo_ and A. Zelinsky. Robust vision based lane tracking using multiple cues and particle _ltering. IEEE IV2003 Intelligent Vehicles Symposium, June 2003.
[3] D.H. Ballard. Generalizing the hough transform to detect arbitrary shapes. Pattern Recognition, 13(2):111 { 122, 1981.
[4] R. Berriel, E. de Aguiar, V. V. de Souza Filho, and T. Oliveira-Santos. A Particle Filter-based lane marker tracking approach using a cubic spline model. Proceedings of the 28th SIBGRAPI Conference on Graphics, Patterns and Images, pages 149{156, 2015.
[5] M. Bertozzi and A. Broggi. GOLD: a parallel real-time stereo vision system for generic obstacle and lane detection. IEEE Transactions on Image Processing, 7(1):62{81, January 1998.
[6] A. Borkar, M. Hayes, M. T. Smith, and S. Pankanti. A layered approach to robust lane detection at night. IEEE Workshop on Computational Intelligence in Vehicles and Vehicular Systems, pages 51{57, March 2009.
[7] A. Borkar, M. Hayes, and Mark T. Smith. Robust lane detection and tracking with RANSAC and Kalman Filter. Proceedings of the 16th IEEE International Conference on Image Processing, pages 3225{3228, 2009.