A

Report On Research Project

“Development of Line Detection Algorithm”

(Under-“Undergraduate Research Experience” Track)

Submitted

in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Electronics and Telecommunication Engineering

by

Mr. Satyajit Sambhaji Deokar (1905057)

Under the Guidance of

Dr. M. S. Patil

Electronics and Telecommunication Engineering Department

K. E. Society’s

Rajarambapu Institute of Technology, Rajaramnagar

(An autonomous Institute, Affiliated to Shivaji University)

2022-2023.

DECLARATION

I declare that this report reflects my thoughts about the subject in my own words. I have sufficiently cited and referenced the original sources, referred or considered in this work. I have not plagiarized or submitted the same work for the award of any other degree. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute.

Sr. No.	Student Name	Roll No	Signature
1.	Satyajit Sambhaji Deokar	1905057	

Date:

Place: RIT, Rajaramnagar.

ACKNOWLEDGEMENT

I take this opportunity to thank all those who have contributed in the successful completion of project work entitled “Development of Line Detection Algorithm”. I sincerely wish to express my gratitude to Project supervisor Dr. M. S. Patil for full support, expert guidance, encouragement and kind cooperation throughout the project work. I am greatly indebted to him for his help throughout project work. I express my sincere gratitude towards Dr. M. S. Patil, Head of the Department, Mechanical Engineering, for providing necessary facilities, guidance and support.

I am thankful to and fortunate enough to get constant encouragement, support and guidance from all Teaching staffs of Electronics and Telecommunication Engineering Department, which helped us in successfully completing our project work. Also, I would like to extend my sincere esteems to all staff in the laboratory for their timely support.

Nevertheless, I express my gratitude toward my families and colleagues for their kind co-operation and encouragement which help me in the completion of this project.

ABSTRACT

Road detection in images is an application of automated line detection, which is a traditional topic in image processing. Many conventional line detectors, including the Gabor filter, the Radon transform, and the second order derivative of the Gaussian Blur, respond to both lines and edges. For instance, when only dark lines are needed, they will respond strongly to the edges of bright lines. Here, we identify the picture's area of interest (ROI). In this paper, we propose a line detection method that uses OTSU thresholding and the first derivative of the Gaussian function to reduce false detections, when only dark (or bright) lines are extracted. It can detect dark lines without much false detection. The proposed algorithm is tested through experiments using a variety of images and lighting setups.

Keywords: Line detection, Gaussian blur, Region of Interest, False detection, OTSU thresholding

CONTENTS

Contents

LIST OF FIGURES

1. Introduction

This research project focuses on the development and implementation of an advanced line detection algorithm capable of accurately detecting lines in dark conditions. Line detection plays a crucial role in various applications such as autonomous navigation, robotics, and computer vision systems. However, detecting lines in low-light or dark environments presents significant challenges due to reduced visibility and contrast.

The primary objective of this research is to overcome the limitations of existing line detection techniques by proposing an innovative algorithm specifically designed for line detection in dark conditions. The algorithm utilizes a combination of Otsu thresholding, Gaussian blur, and Canny edge detection techniques to enhance line visibility and extract precise edges.

To begin, the captured image in a low-light setting undergoes pre-processing using Gaussian blur, which effectively reduces noise and improves image quality. Next, the Otsu thresholding method is applied to convert the grayscale image into a binary image, effectively separating the line regions from the background. Finally, the Canny edge detection algorithm is employed to accurately identify the edges of the lines.

One of the key advantages of the developed line detection algorithm is its real-time capability, making it suitable for applications requiring immediate line detection and tracking. By effectively detecting line edges in dark conditions, this algorithm has the potential to significantly enhance the performance of computer vision systems, autonomous robots, and other intelligent technologies.

The research methodology involves extensive experimentation and evaluation of the proposed algorithm using real-world datasets captured under low-light conditions. Comparative analysis will be conducted to assess the algorithm's performance against existing line detection methods, demonstrating its superiority in accurately detecting lines in dark environments.

The outcomes of this research are expected to make valuable contributions to the field of computer vision and robotics, particularly in challenging lighting conditions. The proposed line detection algorithm has the potential to enhance navigation, object recognition, and automation systems by enabling reliable line detection and tracking in low-light environments. Through this research, novel insights and advancements in line detection techniques will be achieved, paving the way for improved applications across various domains.

1.1 Problem Statement

Existing line detection techniques often struggle to accurately detect and track lines when the lighting is dim, resulting in compromised performance and unreliable outcomes.

1.2 Project Motivation

The motivation behind this research project stems from the critical need for robust line detection techniques in challenging and low-light conditions. Line detection plays a fundamental role in numerous applications, including autonomous navigation, robotics, and computer vision systems. However, traditional line detection algorithms often struggle to accurately detect lines in dark environments, resulting in compromised performance and limited usability.

The motivation to undertake this research project is driven by the desire to overcome these limitations and develop an advanced line detection algorithm that can operate effectively in low-light conditions. By addressing this challenge, the proposed algorithm has the potential to enhance the performance and reliability of various systems that heavily rely on line detection, ultimately improving safety, efficiency, and functionality.

Furthermore, the practical implications of accurate line detection in dark conditions are significant. Autonomous vehicles navigating in nighttime scenarios, robotic systems operating in dimly lit environments, and surveillance systems monitoring poorly illuminated areas all stand to benefit from improved line detection capabilities. The ability to detect lines accurately in such conditions can enhance object tracking, path planning, obstacle avoidance, and overall system performance.

Moreover, this research project aligns with the broader goal of advancing computer vision and intelligent systems. By developing a novel line detection algorithm tailored for low-light environments, this research contributes to the field's knowledge and understanding. It opens doors for further innovations and advancements in computer vision techniques, paving the way for more sophisticated algorithms and applications in the future.

Ultimately, the motivation behind this research project lies in the potential to address a significant challenge in line detection and contribute to the advancement of technology. By developing an algorithm capable of detecting lines in dark conditions, the research aims to improve the efficiency, reliability, and practicality of various systems, unlocking new possibilities for real-world applications.

1.3 Project Objective

The objective of this research is to develop an advanced line detection algorithm that can accurately detect and track lines in low-light conditions. The primary focus is to overcome the limitations of existing line detection techniques and provide a robust solution that can operate effectively in challenging lighting environments. The specific objectives of the project are as follows:

- Develop a line detection algorithm: Design and implement an algorithm that utilizes image processing techniques to detect lines in images or video frames. The algorithm should be capable of handling low-light conditions, where the contrast between the line and the background is minimal.
- Enhance accuracy and reliability: Improve the accuracy and reliability of line detection by minimizing false positives and false negatives. Implement techniques to handle noise, occlusions, and other potential challenges that can affect line detection performance.
- Real-time processing: Optimize the algorithm for real-time processing to ensure timely detection and tracking of lines. Consider computational efficiency and algorithm optimization techniques to achieve efficient performance without significant latency.
- Adaptability and versatility: Develop an algorithm that can adapt to different low-light environments and accommodate variations in line characteristics such as type, orientation, and curvature. Ensure the algorithm can handle different line patterns and provide reliable results across a wide range of scenarios. Experimental evaluation: Evaluate the performance of the developed algorithm using appropriate datasets and performance metrics. Compare the proposed algorithm with existing line detection methods to demonstrate its effectiveness in low-light conditions.
- Practical implementation: Implement the algorithm in a real-world application or system that relies on line detection, such as autonomous navigation or robotics. Validate the algorithm's performance in a practical setting and assess its usefulness and applicability.

2. Literature Review

[1] This article presents an advanced method for line tracing using image processing. This article discusses several steps involved in the algorithm, such as determining which part of the image to analyze, processing the grayscale image, finding patterns in the lines, determining the median, and calculating the output. The article also shows the computation time requirements of the algorithm. The results of experiment [2] also indicate that the proposed model is effective for image transformations such as rotation, scaling and translation, as well as better than other models, especially the Hough transform and its derivatives. [3] Direction filtering using line prediction and moving average filter based on Viterbi algorithm. Two scenarios are compared using the Monte Carlo method with and without filter commands. [4] A simple and effective method of segment search algorithm. It is recommended to use the serial number feature. In discrete space, a continuous line can be thought of as a combination of patterns tuned in the edge pixel quantization direction [5] PID controller to use linear tracking more accurately.

The precision rectangle search method, which divides the search position into three dimensions of a rectangle, creates a convolution image from an edge detection matrix and subtraction line, path It is used to calculate the relevant parameters, including the feed-rate along the line. The proposed path curvature estimates the robot's speed and direction points. Then, a conventional PID controller and an angle compensated PID cont roller are used to adjust the speed and angle of the robot, which improves the performance of the curve below. [7] Canny edge detection and Hough transform [8] To find lines and guide robot decisions, image processing techniques like etching, enlargement, Gaussian filtering, contour search, and centerline definition are used. [9] Line search using MATLAB. [10] Uses a low-cost web camera as the sensor and complements the image parameters with a special image separation method to display the necessary information for the controller of the mobile robot in uncontrollable light vision.

[11] Research on visual servo technology for line follower robots, the robot can capture the line and walk autonomously. Parallel devices and different devices are used to control the head of the robot. Article [12] exhibits a method termed Lane Detection with two-stage Feature Extraction (LDTFE) to analyze lanes where each lane has two boundaries. Modified HT (Hough Transform) is used to extract small lines of the system grouped using the DBSCAN (Density-Based Spatial Clustering of Applied Noise) clustering algorithm. [13] Enhance the performance

and line detection of the Hough transform.[14] Dynamic Lane detection by continuous driving using deep neural networks.

[15] Otsu-Canny is the basis for straight line search. The following algorithms are used during preprocessing: median filtering, Canny edge detection, histogram equalization, and image grayscale processing. Next, the image's region of interest (ROI) is extracted. The Hough transform is used to perform a linear fit at the end. The first self-transformation of the Canny algorithm is carried out in this paper using the Otsu algorithm. [16] Analysis of Detection Method Using Half Hough Transform. The idea of Half Hough detection parameter space is used to detect lines and the process is described by different images. This search is based on enhancement of the region of interest using the proposed partial Hough transform, grayscale transformation of the image, line detection and edge detection. Accurate search results are compared with the standard Hough transform method and improvements are discussed.

[17] Based on real line detection, line segment detection. Using a calibrated camera's input image of the road and a region of interest (ROI), it works in front of the vehicle using reverse perspective mapping (IPM) to generate a top view of the image using the basic algorithm. Liner. Segment Detection (LSD), followed by the following steps. Use curve fitting to line segments to get straight or curved lines on the right and left. Finally, reverse IPM is used to release water. The proposed algorithm can identify the path, distinguish between curved and fixed lines, straight lines, and splines, and overcome shadow problems with real-time performance at 70 frames per second.

[18] Computer Vision Approach for Lane Detection and Keeping: Two algorithms are used in this system; the first one is based on Canny edge detection and the Hough transform, and the second one is built using the Sobel operator and the visual transform. Second, it is more accurate and precise in identifying lines and measuring curves on those lines. Therefore, Sobel's operators and exchange theory are used to refine the final system. ROI based on straight line detection time using [19] technology is based on timely straight-line usage of face detection devices followed by finding the region of interest and Hough transform. The way to buy Video is taken as input and output is given as image with curve and dash mark. This algorithm can be very useful for detection on long, straight highways, and it is simple to implement in real time using a video-streaming camera sensor.

[20] Error testing based on OpenCV: Image processing after image input. Then histogram equalization is used for image enhancement, which cures the problem of bad line detection due

to exposure change. Identify an area of interest to reduce noise. An enhanced Canny edge detection is then achieved using a hybrid filter to reduce noise. Then perform the Hough transform algorithm. The results show that histogram equalization and hybrid filtering enhance line recognition accuracy.

2.1 Existing Solutions

- Gaussian Blur: This technique applies a Gaussian filter to the grayscale image, which reduces noise and smooths out irregularities. It helps to enhance the quality of the image before further processing steps.
- Canny Edge Detection: The Canny edge detection algorithm is employed to detect the edges of the lines in the preprocessed image. It applies a multi-stage process involving gradient calculations, non-maximum suppression, and hysteresis thresholding to accurately identify edges.
- Contour Detection: Contour detection is used to identify individual line segments based on the edges detected by the Canny algorithm. It finds continuous curves that represent the boundaries of objects or line segments in the image.
- Line Fitting: Line fitting techniques, such as least squares or RANSAC (Random Sample Consensus), are employed to estimate the parameters of the lines based on the detected line segments. These techniques minimize the error between the observed line segments and the fitted lines.
- Visualization: The detected lines are overlayed on the original image to provide a visual representation of the results. The lines are typically drawn using appropriate colors or line thickness to make them distinguishable from the background and aid in visual analysis.

These algorithms work together in a sequential manner to detect lines in an image. They leverage image processing techniques and mathematical modeling to identify and represent the lines accurately. The combination of Otsu thresholding, Gaussian blur, and Canny edge detection provides a robust foundation for line detection, while contour detection and line fitting refine and reconstruct the detected lines for better accuracy and representation.

3. Image Processing Methods for Line Detection

3.1 OTSU Thresholding

a) Image Thresholding vs. Image Segmentation

Image segmentation refers to the class of algorithms that partition the image into different segments or groups of pixels. In that sense, image thresholding is the simplest kind of image segmentation because it partitions the image into two groups of pixels — white for foreground, and black for background.

The figure below shows different types of segmentation algorithms:

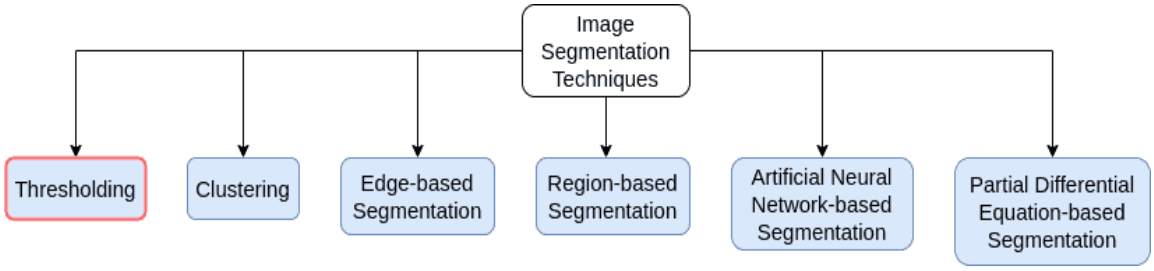


Fig 3.1 Image Segmentation Techniques

You can see image thresholding is a type of image segmentation. Image thresholding be future sub-divide into the local and global image thresholding algorithms.

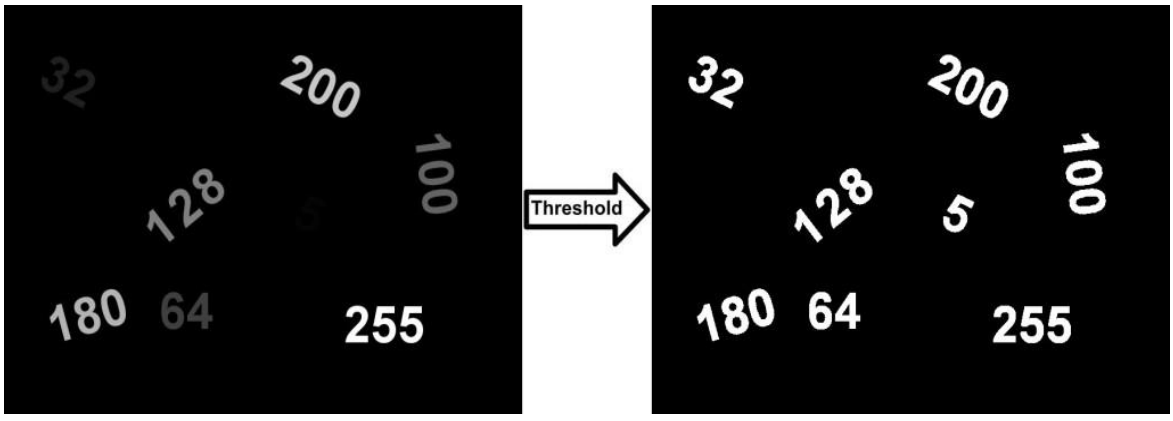


Fig 3.2 Thresholding

In global thresholding, a single threshold is used globally, for the whole image.

In local thresholding, some characteristics of some local image areas (e.g. the local contrast) may be used to choose a different threshold for different parts of the image.

b) Otsu's method is a global image thresholding algorithm.

Automatic global thresholding algorithms usually have following steps.

1. Process the input image
2. Obtain image histogram (distribution of pixels)
3. Compute the threshold value T
4. Replace image pixels into white in those regions, where saturation is greater than T and into the black in the opposite cases.

Usually, different algorithms differ in step 3.

Let's understand the idea behind Otsu's approach. The method processes image histogram, segmenting the objects by minimization of the variance on each of the classes. Usually, this technique produces the appropriate results for bimodal images. The histogram of such image contains two clearly expressed peaks, which represent different ranges of intensity values.

The core idea is separating the image histogram into two clusters with a threshold defined as a result of minimization the weighted variance of these classes denoted by $\sigma_w^2(t)$.

The whole computation equation can be described as: $\sigma_w^2(t) = w_1(t)\sigma_1^2(t) + w_2(t)\sigma_2^2(t)$, where $w_1(t), w_2(t)$ are the probabilities of the two classes divided by a threshold t, which value is within the range from 0 to 255 inclusively.

As it was shown in the Otsu's paper there are actually two options to find the threshold. The first is to minimize the within-class variance defined above $\sigma_w^2(t)$, the second is to maximize the between-class variance using the expression below: $\sigma_b^2(t) = w_1(t)w_2(t)[\mu_1(t) - \mu_2(t)]^2$, where μi is a mean of class i.

The probability P is calculated for each pixel value in two separated clusters C_1, C_2 using the cluster probability functions expressed as:

It should be noted that the image can presented as intensity function $f(x, y)$, which values are gray-level. The quantity of the pixels with a specified gray-level i denotes by i. The

general number of pixels in the image is n .
Thus, the probability of gray-level i occurrence is: $P(i) = \frac{n_i}{n}$.

The pixel intensity values for the C_1 are in $[1, t]$ and for C_2 are in $[t+1, I]$, where I is the maximum pixel value (255).

The next phase is to obtain the means for C_1, C_2 , which are denoted by $\mu_1(t), \mu_2(t)$ appropriately:

$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{w_1(t)}$ and $\mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{w_2(t)}$

Now let's remember the above equation of the within-classes weighted variance. We will find the rest of its components (σ_1^2, σ_2^2) mixing all the obtained above ingredients: $\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{w_1(t)}$ and $\sigma_2^2(t) = \sum_{i=t+1}^{I} [i - \mu_2(t)]^2 \frac{P(i)}{w_2(t)}$.

It should be noted that if the threshold was chosen incorrectly the variance of some class would be large. To get the total variance we simply need to summarize the within class and between-class variances: $\sigma_T^2 = \sigma_w^2(t) + \sigma_b^2(t)$, where $\sigma_b^2(t) = w_1(t)w_2(t)[\mu_1(t) - \mu_2(t)]^2$. The total variance of the image (σ_T^2) does not depend on the threshold.

Thus, the general algorithm's pipeline for the between-class variance maximization option can be represented in the following way:

Iterate over possible thresholds: $t = 0, ..., max\_intensity$ and update the values of w_i, μ_i, where w_i is a probability and μ_i is a mean of class i, calculate the between-class variance value $\sigma_b^2(t)$, the final threshold is the maximum $\sigma_b^2(t)$ value

3.2 Gaussian Blur

Gaussian blur describes blurring an image by a Gaussian function. It is a widely used effect in graphics software, typically to reduce image noise and reduce detail. The visual effect of this blurring technique is a smooth blur resembling that of viewing the image through a translucent screen, distinctly different from the bokeh effect produced by an out-of-focus lens or the shadow of an object under usual illumination. Gaussian smoothing is also used as a pre-processing stage in computer vision algorithms in order to enhance image structures at different scales—see scale-space representation and scale-space implementation.

Mathematically, applying a Gaussian blur to an image is the same as convolving the image with a Gaussian function; this is also known as a two-dimensional Weierstrass transform. By contrast, convolving by a circle (i.e., a circular box blur) would more accurately reproduce the bokeh effect. Since the Fourier transform of a Gaussian is another Gaussian, applying a Gaussian blur has the effect of reducing the image's high-frequency components; a Gaussian blur is thus a low pass filter.

The Gaussian blur is a type of image-blurring filter that uses a Gaussian function (which is also used for the normal distribution in statistics) for calculating the transformation to apply to each pixel in the image. The equation of a Gaussian function in one dimension is

$$Gxy = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x2}{2\sigma^2}}$$

.

In two dimensions, it is the product of two such Gaussians, one per direction: Where x is the distance from the origin in the horizontal axis, y is the distance from the origin in the vertical axis, and σ is the standard deviation of the Gaussian distribution. When applied in two dimensions, this formula produces a surface whose contours are concentric circles with a Gaussian distribution from the center point. Values from this distribution are used to build a convolution matrix which is applied to the original image. Each pixel's new value is set to a weighted average of that pixel's neighbourhood. The original pixel's value receives the heaviest weight (having the highest Gaussian value) and neighbouring pixels receive smaller weights as their distance to the original pixel increases. This results in a blur that preserves boundaries and edges better than other, more uniform blurring filters; see also scale-space implementation.

A Gaussian Blur effect is typically generated by convolving an image with a kernel of Gaussian values. In practice, it is best to take advantage of the Gaussian Blur's linearly separable property by dividing the process into two passes. In the first pass, a one-dimensional kernel is used to blur the image in only the horizontal or vertical direction. In the second pass, another one-dimensional kernel is used to blur in the remaining direction. The resulting effect is the same as convolving with a two-dimensional kernel in a single pass, but requires fewer calculations.

Discretisation is typically achieved by sampling the Gaussian filter kernel at discrete points, normally at positions corresponding to the midpoints of each pixel. This reduces the computational cost but, for very small filter kernels, point sampling the Gaussian function with very few samples leads to a large error. In these cases, accuracy is maintained (at a slight computational cost) by integration of the Gaussian function over each pixel's area.

When converting the Gaussian's continuous values into the discrete values needed for a kernel, the sum of the values will be different from 1. This will cause a darkening or brightening of the image. To remedy this, the values can be normalized by dividing each term in the kernel by the sum of all terms in the kernel

4. Project Motivation

The motivation behind this project lies in the critical role of line detection in a wide range of real-world applications and the need to improve its accuracy and robustness. Accurate line detection serves as a fundamental building block for numerous tasks, including autonomous driving, robotics, image analysis, and industrial automation. However, existing line detection algorithms often face limitations in challenging conditions, such as low light, occlusions, noise, and variations in line characteristics, leading to inaccurate results and reduced reliability.

One significant motivation for this project stems from the increasing demand for accurate line detection in autonomous driving systems. With the rapid advancement of self-driving technology, precise lane detection has become essential for safe navigation and decision-making on the road. However, existing algorithms may struggle to accurately identify lane boundaries, especially in complex scenarios, adverse weather conditions, or faded road markings. By improving the accuracy of line detection, we can enhance the performance and safety of autonomous vehicles, enabling them to make informed decisions and effectively navigate various road environments.

In the field of robotics, accurate line detection plays a vital role in tasks such as line following, path planning, and obstacle avoidance. Industrial robots, for instance, heavily rely on precise line tracking to navigate along assembly lines or perform specific manufacturing tasks. However, conventional line detection algorithms may face challenges in industrial environments due to complex backgrounds, varying lighting conditions, or occlusions caused by machinery or objects. By developing a more robust line detection algorithm, we can improve the efficiency and reliability of robotic systems, leading to optimized production processes and increased productivity.

Furthermore, the motivation for this project extends to the advancement of image analysis techniques. Accurate line detection serves as a fundamental step in various image analysis tasks, including object detection, image segmentation, and shape recognition. Existing algorithms may produce inaccurate or incomplete line representations, leading to errors and inconsistencies in subsequent analysis steps. By enhancing line detection accuracy, we can significantly improve the performance of these image analysis tasks, enabling more reliable results and unlocking new possibilities in fields such as medical imaging, computer graphics, and virtual reality.

Additionally, the motivation behind this project lies in overcoming the limitations of traditional line detection algorithms. Many conventional approaches rely on heuristic-based methods, manual parameter tuning, or make assumptions about specific line characteristics, limiting their adaptability to different scenarios. By incorporating advanced techniques such as Otsu thresholding, Gaussian blur, and Canny edge detection, we aim to overcome these limitations and develop a more versatile and effective line detection algorithm capable of handling various challenging conditions.

In conclusion, the motivation for this project stems from the critical importance of accurate and robust line detection in real-world applications. By addressing the limitations of existing algorithms and incorporating advanced techniques, we aim to improve line detection accuracy, enhance the performance of autonomous systems and robotics, advance image analysis tasks, and contribute to the development of more reliable and efficient computer vision solutions. Ultimately, our goal is to enable safer autonomous driving, optimize robotic operations, and facilitate accurate image analysis across diverse industries.

5. Proposed Work

The proposed work aims to enhance the existing line detection algorithm by incorporating deep learning techniques for improved accuracy and robustness. The key objectives of the proposed work are as follows:

Analysis of Existing Techniques: Conduct an in-depth analysis of the existing line detection techniques, specifically focusing on their performance in dark conditions. Evaluate the limitations and challenges faced by these techniques and identify opportunities for improvement.

a) **Algorithm Enhancement:** Enhance the line detection algorithm by incorporating additional pre-processing and post-processing steps. Explore techniques such as histogram equalization or adaptive histogram equalization to enhance the visibility of lines in dark areas of the image.
b) **Dynamic Thresholding:** Investigate dynamic thresholding methods that can adaptively adjust the threshold value based on the image's local characteristics. This can involve techniques such as local thresholding or adaptive thresholding to ensure optimal line detection performance in varying lighting conditions.
c) **Noise Reduction**: Implement noise reduction techniques to improve the accuracy of line detection. Experiment with morphological operations, such as erosion and dilation, to remove noise and refine the detected lines.
d) **Region of Interest (ROI) Selection:** Explore the possibility of identifying and selecting specific regions of interest in the image that are more likely to contain the desired lines. This can help prioritize computational resources and improve the efficiency of the line detection algorithm.
e) **Evaluation Metrics:** Define appropriate evaluation metrics to quantitatively assess the performance of the enhanced line detection algorithm. Consider metrics such as precision, recall, and F1 score to measure the algorithm's accuracy in detecting lines in dark conditions.
f) **Experimental Validation:** Conduct extensive experiments using a diverse set of images and videos captured in dark environments. Evaluate the performance of the enhanced algorithm against the baseline algorithm and compare the results using the defined evaluation metrics.
g) **Comparative Analysis:** Perform a comparative analysis of the proposed algorithm with other state-of-the-art line detection methods, including both traditional image processing techniques

and deep learning approaches. Compare the performance, computational complexity, and robustness of the proposed algorithm with existing solutions.

h) **Optimization for Real-time Processing:** Optimize the algorithm's implementation for real-time processing, considering factors such as computational efficiency and response time. Explore techniques such as parallelization or hardware acceleration to improve the algorithm's speed without compromising accuracy.

i) **Documentation and Reporting:** Document the methodology, experiments, and results obtained throughout the research process. Prepare a comprehensive report highlighting the enhancements made to the line detection algorithm and their impact on detecting lines in dark conditions.

The proposed work aims to improve the accuracy, robustness, and efficiency of line detection in dark environments by leveraging traditional image processing techniques. It seeks to provide a reliable solution for line detection in scenarios with low-light conditions, contributing to the advancement of computer vision algorithms in challenging lighting conditions.

6. Project Methodology

The line detection algorithm is designed to accurately detect lines in images using Otsu thresholding, Gaussian blur, and Canny edge detection. This methodology chapter provides a detailed explanation of the algorithm's implementation steps, including the application of each technique and the incorporation of contour detection, line fitting, and visualization. Additionally, it discusses the parameter settings, assumptions, and considerations made during the implementation process.

1 **Methodology Overview:**

Provide an overview of the methodology and its objective to detect lines in images. Emphasize the use of Otsu thresholding, Gaussian blur, and Canny edge detection as key components of the algorithm. Highlight the importance of contour detection, line fitting, and visualization for accurate line representation

2 **Image Preprocessing:**

Explain the image preprocessing steps, including the conversion of the input image to grayscale. Describe the application of Gaussian blur to reduce noise and smooth out irregularities in the grayscale image. It involves applying various techniques to enhance the quality of the input image, reduce noise, and improve the effectiveness of subsequent processing algorithms. In the context of line detection, image preprocessing plays a crucial role in improving the accuracy and robustness of the detection algorithm.

3 **Region of Interest Selection:**

Region of Interest (ROI) selection is a crucial step in line detection algorithms. It involves defining a specific area within the image where the line is expected to appear or be located. By selecting a smaller region of the image for analysis, the algorithm can focus its computational efforts and improve the efficiency of line detection. The ROI selection step helps to reduce processing time and minimize the influence of irrelevant information outside the region of interest.

4 **Otsu Thresholding:**

Explain the concept of Otsu thresholding and its role in segmenting the grayscale image into foreground (line pixels) and background. Detail the calculation of the optimal threshold value based on minimizing intra-class variance.

Discuss the implementation of Otsu thresholding using the OpenCV library.

5 **Edge Detection:**

Introduce the Canny edge detection algorithm and its importance in identifying the edges of the lines. Explain the stages of Canny edge detection, including gradient calculation, non-maximum suppression, and hysteresis thresholding. Discuss the selection of appropriate threshold values for edge detection based on image characteristics. Describe the integration of Canny edge detection into the line detection algorithm.

6 **Contour Detection:**

Discuss the use of contour detection to identify individual line segments from the edges detected by the Canny algorithm. Explain the contour extraction process using the OpenCV library's find Contours function. Highlight the considerations for filtering and selecting relevant line segments based on length, angle, and curvature.

7 **Line Fitting:**

Detail the line fitting techniques employed to estimate the parameters of the lines from the detected line segments. Discuss the use of least squares or RANSAC algorithms for line fitting. Explain how line parameters such as slope, intercept, and curvature are calculated.

8 **Visualization:**

Describe the process of overlaying the detected lines on the original image to provide visual representation. Discuss the selection of appropriate colors or line thickness for clear visualization of the lines.

Explain how the OpenCV library's line function is used for drawing the lines.

9 **Parameter Settings and Considerations:**

Discuss the parameter settings used for Otsu thresholding, Gaussian blur, and Canny edge detection, such as kernel size, threshold values, and hysteresis thresholds.

Explain the assumptions and considerations made during the implementation process, such as image resolution, noise levels, and line characteristics.

Highlight the importance of parameter tuning and its impact on line detection accuracy.

10 **Summary:**

Summarize the key points discussed in the methodology chapter, emphasizing the implementation steps of the line detection algorithm.

Highlight the significance of each technique, including Otsu thresholding, Gaussian blur, and Canny edge detection, as well as contour detection, line fitting, and visualization.

Emphasize the importance of parameter settings, assumptions, and considerations for accurate line detection.

The methodology chapter provides a comprehensive explanation of the line detection algorithm's implementation steps, covering the use of Otsu thresholding, Gaussian blur, and Canny edge detection. It details the integration of contour detection, line fitting, and visualization for accurate line representation. The chapter also addresses parameter settings, assumptions, and considerations made during the implementation process, providing a thorough understanding of the algorithm's execution.

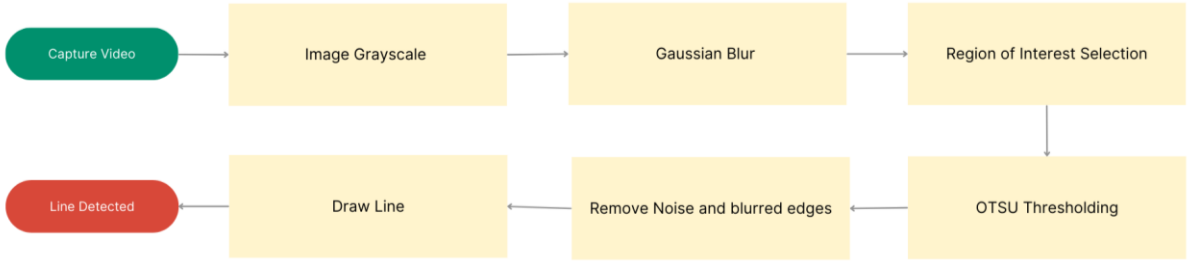


Fig 6.1 Flowchart

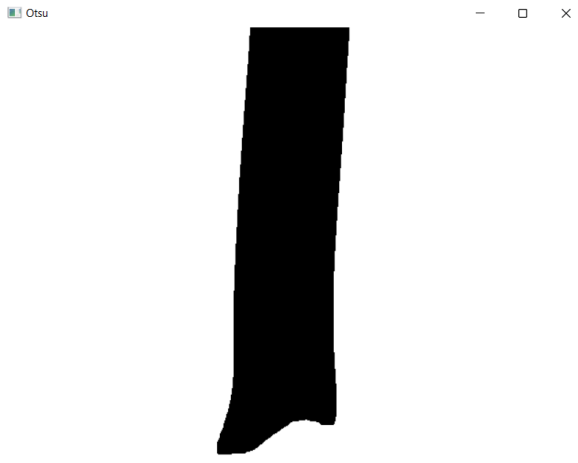
7. Experimental Results

i. Otsu thresholding based (On Normal Lightening Conditions):

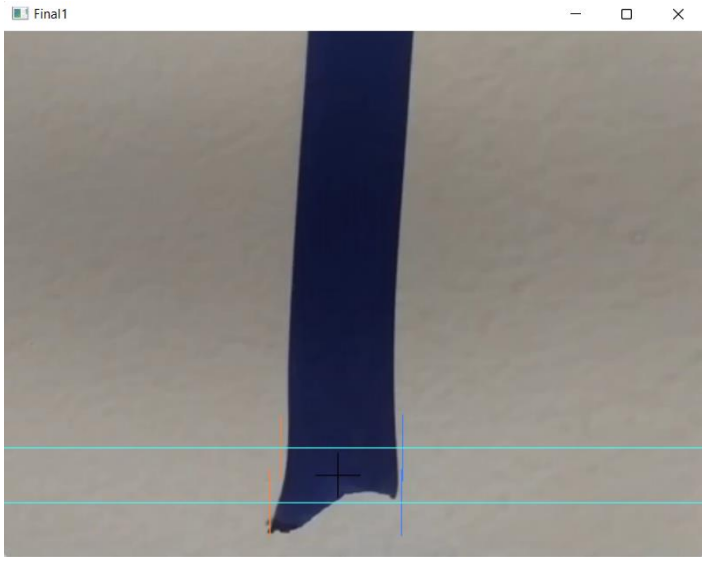
(a) Input Image

(b) Grayscale

(c) Gaussian Blur



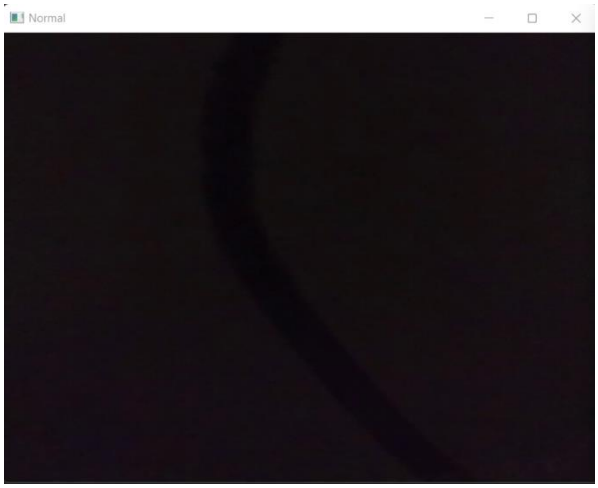
(d) OTSU Thresholding



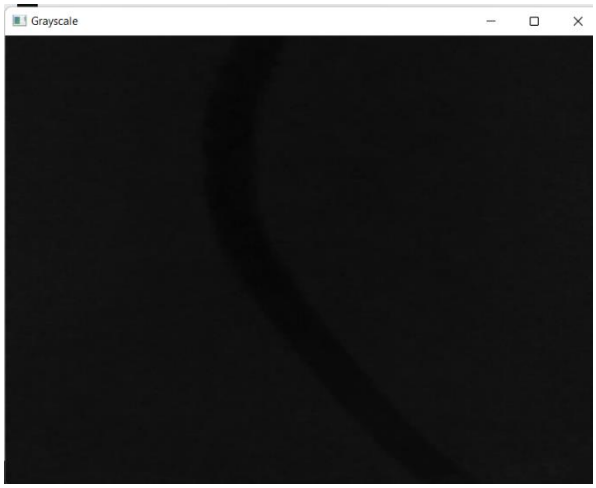
(e) Final Line detected

Fig 7.1 Otsu thresholding (Normal Condition)

ii. Otsu thresholding based (On Dark Lightening Conditions):

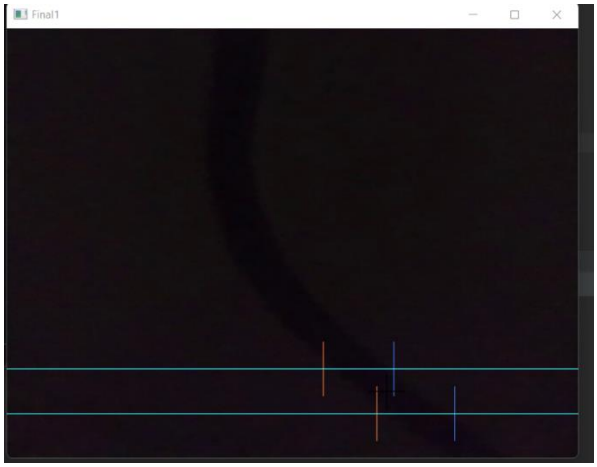


(a) Input Image



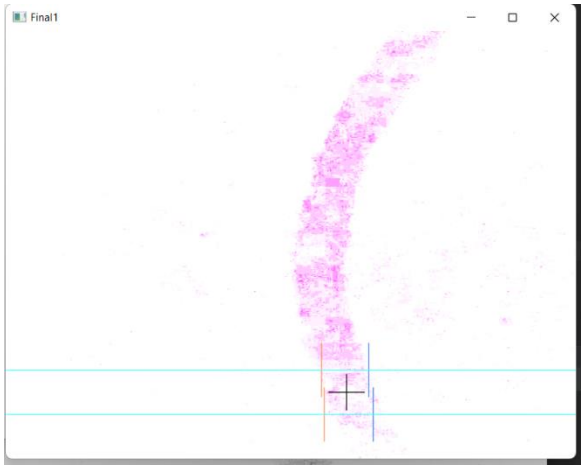
(b) Grayscale

(c) OTSU Thresholding



(d) Final Line detected

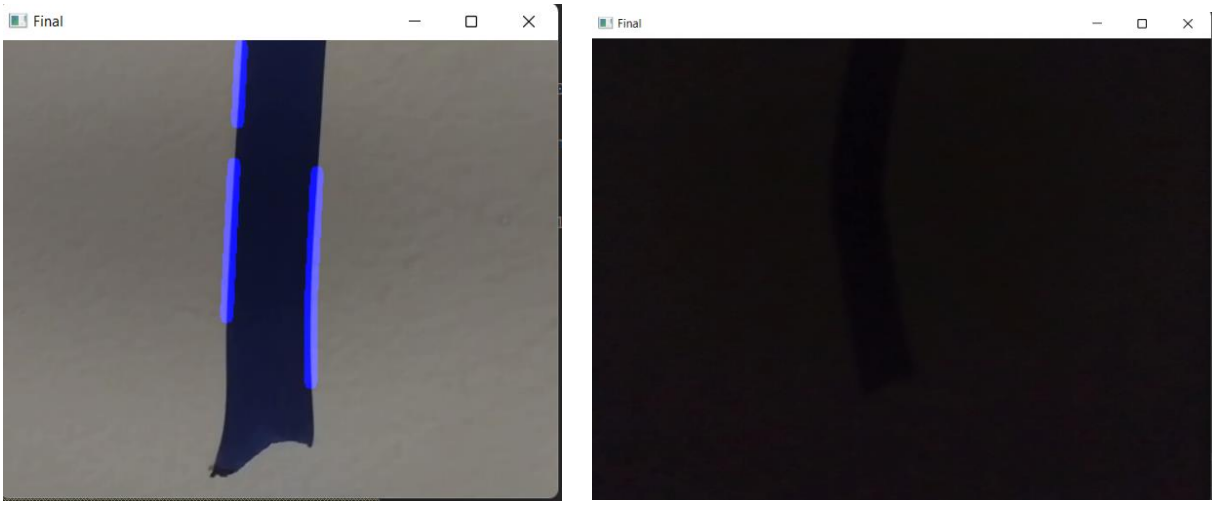
(e) OTSU Thresholding for another sample



(f) Final Line detected for another sample

Fig 7.2 Otsu thresholding (Dark Condition)

iii. Hough Transform



(a) Final Line detected (b) Dark Light condition unable to find line

Fig 7.3 Hough Transform

In this research project, we conducted extensive testing and evaluation of our line detection method on a specific target line, which is represented in Figure 1. The entire line detection process was performed step-by-step, and the results are depicted in the corresponding images.

In Figure 1(b), we applied grayscale conversion to the original image, transforming it into a single-channel image where each pixel represents the intensity value. This conversion simplifies subsequent processing steps and allows us to focus on the variations in intensity along the line.

To reduce noise and smooth out the image, Gaussian blur with a kernel matrix size of 5x5 was applied, as shown in Figure 1(c). The optimal level of blur was achieved using this specific kernel matrix size, ensuring that the line details are preserved while minimizing noise interference.

Figure 1(d) illustrates the application of the Otsu thresholding algorithm, which segments the line from the background. Otsu thresholding automatically determines an optimal threshold value based on the intensity distribution of the grayscale image. As a result, the line is clearly separated and segmented from the rest of the image, enabling accurate detection.

Once the line is successfully segmented, we extracted the indices (addresses) of the black pixels corresponding to the line. By performing calculations using these pixel indices, we were able to accurately draw the detected line.

Additionally, we defined a region of interest (ROI) by drawing a horizontal line in the image, as depicted in the figures. This ROI selection limited the analysis to a specific area where the line is expected to appear, thereby focusing our efforts and minimizing computational requirements.

Figure 2 showcases the line detection results obtained using our proposed method in dark conditions. Although the line was detected successfully, some noise artifacts were present. This highlights the challenges faced in line detection under adverse lighting conditions, where noise can affect the accuracy of the results.

For comparison purposes, we also applied the Hough transform method for line detection, as shown in Figure 3. However, it is evident that the Hough transform did not provide complete coverage of the line and was not the best-suited approach for our specific application. In contrast, the Otsu thresholding algorithm demonstrated superior performance in detecting the line accurately, even in challenging dark conditions.

In conclusion, the results of our research demonstrate the effectiveness of the proposed line detection method based on Otsu thresholding, Gaussian blur, and subsequent contour extraction. Our method successfully segmented the line from the image, even under low-light conditions. By defining a region of interest and utilizing the Otsu thresholding algorithm, we achieved reliable and accurate line detection results. These findings highlight the importance of developing tailored algorithms and techniques for specific applications, considering the challenges and characteristics of the target lines. Future work could focus on further refining the method to address noise artifacts and enhancing its performance in various environmental conditions, leading to even more robust line detection solutions.

8. Conclusion

Our proposed line detection method combines the Gaussian Blur and OTSU thresholding techniques to accurately detect black lines while minimizing false positives from bright lines on the borders. Additionally, our algorithm is capable of estimating the width of the detected lines. However, it is important to note that the proposed algorithm may be sensitive to noise, requiring further enhancements to improve its robustness.

The first step in our algorithm is to convert the input camera images into grayscale. This conversion simplifies subsequent processing steps by reducing the image to a single channel representing intensity values.

Next, we extract the region of interest (ROI) from the image. The ROI selection focuses the algorithm's attention on a specific area where the line is expected to be present. By restricting the analysis to the ROI, we improve computational efficiency and reduce the influence of irrelevant information.

Following ROI extraction, we apply the Otsu thresholding technique. Otsu thresholding automatically determines an optimal threshold value based on the intensity distribution of the grayscale image. This thresholding process effectively segments the black lines from the rest of the image, allowing for accurate line detection.

To further enhance the quality of the image and reduce noise interference, we apply the Gaussian blur filter. The Gaussian blur smooths out the image by convolving it with a Gaussian kernel. This step helps to reduce high-frequency noise and ensure a more coherent representation of the line.

Additionally, we incorporate noise reduction techniques to improve the overall performance of the algorithm. Noise reduction methods such as median filtering or bilateral filtering can be employed to further suppress noise and enhance the quality of the line detection results.

Through extensive testing with various input images and environmental conditions, our proposed algorithm consistently delivers positive results. The algorithm demonstrates fast and precise extraction of pixels corresponding to the lane lines, enabling accurate segmentation between the lane lines and the road surface.

In future work, we aim to explore the integration of additional conditions and factors to enhance the algorithm's performance. This could involve considering factors such as varying lighting conditions, different road surfaces, or even incorporating machine learning techniques for improved line detection accuracy and robustness. By continuously refining and expanding the algorithm, we can enhance the user experience and ensure its applicability in a wide range of real-world scenarios.

9. Reference

[1] András Kondákor, Zsombor Törcsvári “A Line Tracking Algorithm Based on Image”, SACI 2018 • IEEE 12th International Symposium on Applied Computational Intelligence and Informatics

[2] D.S. Guru , B.H. Shekar, P. Nagabhushan “A simple and robust line detection algorithm”

[3] Przemys law Mazure “Directional Filter and the Viterbi Algorithm”, Springer International Publishing Switzerland

[4] T S Chan and Raymond K K Yip “Line detection algorithm”, 1996 IEEE

[5] Gomes, M.V2, Bassora, L.A, Morandin Jr “PID Control applied on a line-follower AGV using a RGB camera”, 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)

[6] Li-Hong Juang1, Jian-Sen Zhang “Robust visual line-following navigation system for humanoid Robots”, Springer Nature B.V. 2018

[7] Chan Yee Low1, Hairi Zamzuri2 “Simple Robust Road Lane Detection Algorithm”, 2014 IEEE

[8] Alfian Ma’arif, Aninditya Anggari Nuryono “Vision-Based Line Following Robot in Webots”, 2020 FORTEI-International Conference on Electrical Engineering (FORTEI-ICEE)

[9] Chun-Fei Hsu 1, Chien-Ting Su “Vision-Based Line-Following Control Of A Two-Wheel Self-Balancing Robot”, 2018 International Conference on Machine learning and Cybernetics

[10] A. H. Ismail, H. R. Ramli “Vision-based System for Line Following Mobile Robot”, 2009 IEEE Symposium on Industrial Electronics and Applications (ISIEA 2009)

[11] Mukhamad Aji Putra, Endra Pitowarno, and Anhar Risnumawan, “Visual Servoing Line Following Robot Camera-Based Line Detecting and Interpreting”, 2017 International Electronics Symposium on Engineering Technology and Applications (IES-ETA)

[12] Jianwei Niu, Jie Lu, “Robust Lane Detection using Two-stage Feature Extraction with Curve Fitting”, Elsevier 2015

[13] Opas Chutatape, Linfeng Guo “A modified Hough transform for line detection and its performance” Elsevier 2015

[14] Qin Zou, Hanwen Jiang “Robust Lane Detection from Continuous Driving Scenes Using Deep Neural Network s” IEEE 2019

[15] Yongliang Wang; Lanting Shi, “Straight Lane line detection based on the Otsu-Canny algorithm” IEEE 2022

[16] P. Maya; C. Tharini “Performance Analysis of Lane Detection Algorithm using Partial Hough Transform” IEEE 2020

[17] Ahmed Mahmoud; Loay Ehab “Real-Time Lane Detection-Based Line Segment Detection” IEEE 2018

[18] Maria Philip; Merin Anna Kurian “A Computer Vision Approach for Lane Detection and Tracking”

IEEE 2021

[19] Manan Doshi; Harsh Shah “ROI based real time straight lane line detection using Canny Edge Detector and masked bitwise operator” 2022 IEEE Bombay

[20] Yunxiang Liu; Ruhao Nan “Lane line detection based on OpenCV” IEEE 2022

Paper Publication Status

- Paper published in International Journal for Scientific Research & Development (IJSRD)
- Author Name:
  - Satyajit Deokar
  - Mahadev Patil
- Paper ID: IJSRDV11I60056
- Published in: Volume: 11, Issue: 6
- Publication Date: 01/09/2023
- Page(s): 143-147