Lane Detection Using OpenCV

The project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

Bachelor of Technology/Master of Technology

In

Computer Science and Engineering

School of Engineering and Sciences

Submitted by

Raghu Sai K - AP20110010311

Tarun Tangirala - AP20110010329

K Sai Sri Latha - AP20110010713

Sai Tejaswi - AP20110010353

Under the Guidance of

Dr. Hemantha Kumar

SRM University–AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

November 2023

Table of Contents

Abstract

Lane detection is a crucial element of Autonomous Driving and Advanced Driver-assistance Systems (ADAS), which enable safe navigation by recognizing and tracking lanes on the road. This document delves into the basics of lane detection using the OpenCV computer vision library. It goes over several preprocessing procedures, the Hough Transform's use, problems, real-time implementation, and future improvements.

By the end of this document, you will learn how OpenCV can be utilized for lane detection and its importance in improving road safety. The feature-based method takes advantage of edges and locally relevant visual features, which are sensitive to illumination effects but largely insensitive to road shapes. These local visual features include gradient, color, brightness, texture, orientation, and variations. To suit low levels of features that are more resilient to illumination effects, model-based techniques utilize global road models; nevertheless, they are sensitive to road shapes.

Introduction

The Lane Detection project entails utilizing OpenCV to create a computer vision algorithm to detect and track lanes on the road. This algorithm is critical for autonomous vehicles since it helps them recognize the driving lane boundaries, allowing them to operate safely and autonomously. The project's goal is to recognize lane markers using image processing techniques such as edge detection, color thresholding, and perspective modification.

Using OpenCV's functions and libraries, the algorithm can detect lane lines accurately, even in challenging lighting and weather conditions. The research also incorporates lane tracking algorithms, which entail anticipating the path of the lanes and updating the position as the vehicle advances. The system may also visualize the detected lanes by superimposing them on the input video or image, providing a clear depiction of the lane borders for analysis and decision-making. Overall, the Lane Detection research demonstrates OpenCV's capabilities in improving the perception and navigation of autonomous systems.

By visualizing the detected lanes and applying them to an input video or image, the method provides valuable visual assistance for analysis and decision-making. This project demonstrates how OpenCV may be used to precisely perceive and interact with the road environment, hence increasing the safety and efficiency of autonomous vehicles.

Lane Detection Basics

2.1 What is Lane Detection?

The process of detecting and following lanes or lane markers on a road using visual data from cameras or other sensors is known as '' Lane Detection ''. It is an essential technological advancement for driver assistance programs and self-driving cars.

Within the advanced driver aid system, lane detection and tracking are the most advanced functions. Tracking down white lines on the road is known as lane detection. By employing previously identified lane markers to guide the motion model, lane tracking helps the car stay on the intended course.

2.2 Significance of Lane Detection

Advanced driving assistance systems (ADAS) and driverless cars both depend on lane detection. For navigation and safety, it gives information on the road's design and the car's location within its lane. Lane detection plays a crucial role in:

1. Drivers are assisted in maintaining their lane using lane-keeping assistance.
2. When a driver inadvertently crosses a lane, the lane departure warning system warns them of it.
3. Safe road navigation is made possible for self-driving cars by autonomous driving.
4. Understanding the layout of roads and surrounding environment geometries for proper route planning
5. Advanced Parking Assistance.

2.3 OpenCV in Lane Detection

A large selection of image processing techniques, including lane detection, are available through the OpenCV open-source computer vision library. Implementing lane identification techniques is made easier by its functions for edge and line detection, picture editing, and detection.

Preprocessing Steps

Preprocessing images improves their quality and provides contextual information, making it a vital step in the lane detection process. Image filtering, edge recognition, color segmentation, and noise reduction are a few crucial preprocessing methods. By reducing unnecessary noise, isolating the lane from the background, and enhancing the contrast between the lane and its surroundings, these techniques improve the identification of the lane by lane recognition algorithms.

Furthermore, preprocessing lessens the influence of numerous variables including illumination, weather, and other environmental circumstances, which enhances the lane-detecting algorithm's robustness and accuracy.

3.1 Image Acquisition

The initial stage in lane recognition is to obtain distinct and precise pictures from the camera set up on the vehicle. A high-quality source image is essential to the success of the other preparation stages.

Image acquisition plays a crucial role in lane detection as it provides the relevant visual data for the algorithm to identify and mark the lanes on the road. This is achieved by using specialized cameras that can capture images in vivid, clear detail, like organic light-emitting diodes (OLEDs).

3.2 Gray-Scale Conversion

By reducing the image to a single channel, grayscale conversion makes additional processing easier. It is simpler to deal with grayscale photos for edge detection. Compared to color photos, greyscale photographs are easier to process and have a stronger contrast.

```
import cv2
import numpy as np
img = cv2.IMREAD("test.jpg")
Gray_scale = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Fig. Test_image

Fig. Gray Scale Conversion

3.3 Gaussian Blurring

To minimize the noise in the image and improve edge identification, Gaussian blurring is used. To avoid false positives in the lane-detecting procedure, this smoothing technique is essential.

One popular image processing method for lane detection is Gaussian blur. It is significant because it aids in noise reduction and image smoothing, facilitating lane edge detection. Using a convolutional kernel, the Canny edge detection operator determined the gradient of the image intensity to locate edges in a picture.

```
blurred_image = cv2.GaussianBlur(img, cv2.COLOR_RBG2GRAY)
```

Fig. Gaussian Blur

3.4 Canny Edge Detection

Canny edge detection is a popular image processing edge detection approach. It has been widely utilized in lane recognition applications because of its ability to recover sharp borders in low-noise images. To remove noise from the image, the Canny edge detection algorithm first applies a Gaussian filter on it. A Sobel filter is then applied to identify the image's edges.

Using a hysteresis threshold to identify which pixels are regarded as edges is the last stage. An efficient technique for identifying edges in photos is the Canny edge detection algorithm.

Syntax: cv2.Canny(*image, t_lower, t_upper, apeture_size, L2Gradient*)

Where

Image = source image which Gaussian blur was applied

t_lower = lower threshold value in Hysteresis Thresholding

t_upper = Higher Threshold value in Hysteresis Thresholding

apeture_size = matrix size of Sobel Filter

L2Gradient = Boolean parameter to improve Edge Gradient.

An Example of Sobel Filter (3X3):

-1	0	1
-2	0	2
-1	0	1

Here are some of the advantages of using Canny edge detection for lane detection:

1. Identifying edges is highly efficient in identifying crisp edges in images with minimal noise.
2. It can be applied to many different situations and is comparatively easy to apply.

```
blurred_img = cv2.GaussianBlur(img, cv2.COLOR_RBG2GRAY)
Canny_img = cv2.Canny(blurred_img, 50, 150)
```

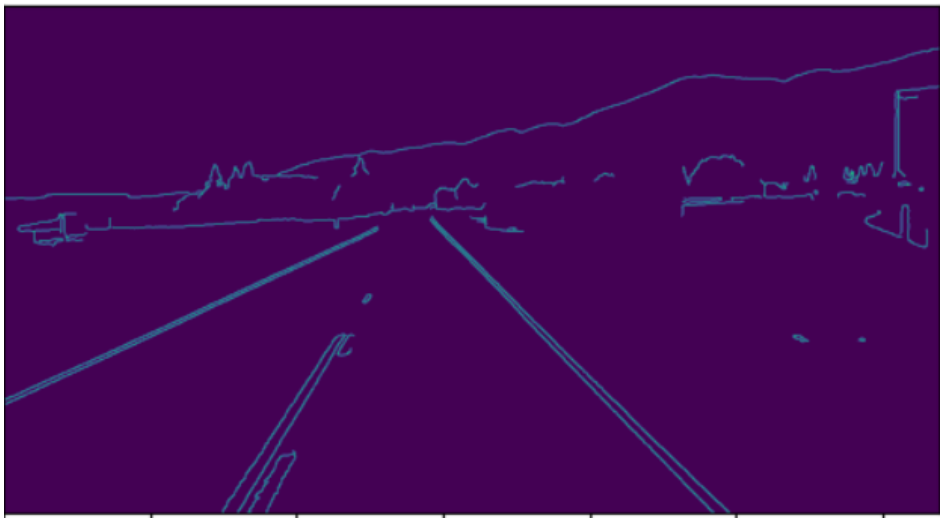

Fig. Canny Edge Detection

Region Of Interest (ROI) Selection

4.1 Defining ROI

A region of interest (ROI) in lane line detection is a particular area of an image that is filtered or otherwise altered to extract information. The area where a car should be driven is known as the ROI.

A designated region of interest (ROI) is established within the image where the lanes are anticipated to appear. Doing this makes the processing effort lessened, and inaccurate detections from other items in the scene are avoided.

4.2 Masking of ROI

Choosing a particular region of interest (ROI) within the image and adding a mask to it is known as ' Masking ROI ' in lane line detection.

Usually, the mask is a binary image with low contrast values in the background and high contrast values in the areas of interest. Masking ROI can aid in lane line recognition by helping to separate and make more visible the areas of the image that contain the lane lines. When lanes need to be precisely identified and tracked, such as in driver assistance systems, this method can be helpful.

```
def region_of_interest(image):
   mask = np.zeros_like(image)
   height = 600
   width = 1000
   roi_vertices = np.array([[(0, height), (width / 2, height / 2),
(width,    height)]], np.int32)
   cv2.fillPoly(mask, roi_vertices, 255)
   masked_region = cv2.bitwise_and(image, mask)

   return masked_region
```

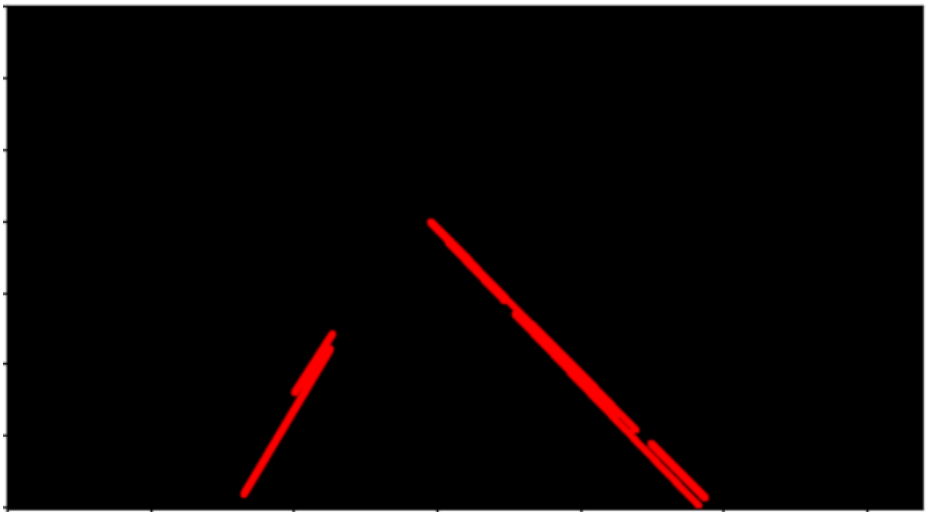

Fig. Region of Interest

Hough Transform for Lane Detection

5.1 Understanding Hough Transform

In computer image processing, the Hough transform is a mathematical method for identifying edges and lines in an image. The Hough transform can be used in lane line detection to take an image of a road scene and extract the lines that indicate the lanes.

By using this technique, the image is converted into a new coordinate system in which points are used to depict lines. The lanes can then be found by thresholding and analyzing the converted image. Self-driving automobiles and other vehicles that need to track and recognize lanes frequently employ this technique.

The algorithm works as follows in its essential steps:

1. It extracts two interest points at a time from an image and assumes those points form a straight line given the collection of points.
2. The image's edge features are acquired by preprocessing.
3. The transformation performed by Hough can locate every line in the picture.

5.2 Applying Hough Transform

To find and extract the lane lines, the ROI edges can be subjected to the Hough Transform. It locates intersections in a Hough space and recognizes lines by translating them into polar coordinates. By identifying the (ρ, θ) pairs with more intersections than a threshold, the Hough transform method finds lines.

```
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([]), 40, 5)
line_image = display_lines(lane_image, lines)

plt.imshow(line_image)
plt.show()
```

5.3 Drawing Lane Lines on the Image

Detected lines can be drawn on the original image to visualize the results

```
def display_lines(image, lines):
    line_image = np.zeros_like(image)
    if lines is not None:
        for line in lines:
            x1, y1, x2, y2 = line.reshape(4)
            cv2.line(line_image, (x1,y1), (x2,y2), (255,0,0), 10)
        result_image = cv2.addWeighted(image, 0.8, line_image, 1,
1)
    return result_image
```

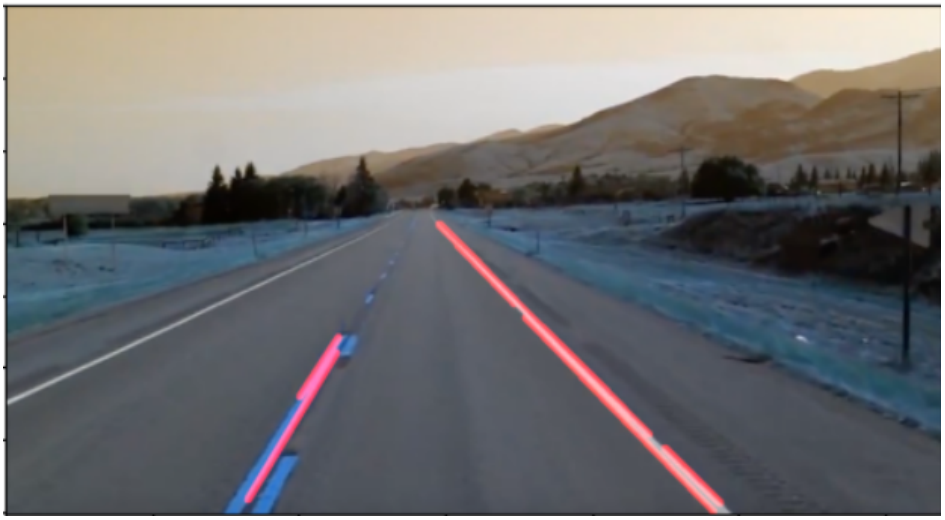

Fig. Drawing Lines on Images

Challenges and Solutions

6.1 Adverse Weather Conditions

In bad weather—such as rain, snow, or fog—lane detecting can be difficult. Lane detection algorithms have difficulty correctly identifying lane lines due to poor visibility and road conditions.

Solutions:

1. For redundancy, use additional sensors like radar and LiDAR.
2. Use cutting-edge image processing methods to improve visibility

6.2 Lane Marking Variability

Because of things like faded paint, damage from the road, or temporary markings, lane markers might differ in look and quality. This fluctuation can make lane spotting difficult.

Solutions:

1. Develop reliable algorithms that can adjust to various lane marking styles.
2. Employ machine learning methodologies to enhance the precision of detection in diverse scenarios.

Real-Time Lane Detections

Processing an uninterrupted video stream from a camera is required for real-time lane detection. For driver assistance systems and automated driving, this is essential. In order to extract lane markings from input video pictures, a real-time lane recognition system combines machine learning approaches with image processing algorithms.

Real-time lane detection can boost transit system efficiency, lessen traffic, and increase road safety.

```
cap = cv2.VideoCapture("test_video.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([]),
40, 5)
   averaged_lines = average_slope_intercept(frame, lines)
   line_image = display_lines(frame, averaged_lines)
   combo_image = cv2.addWeighted(frame, 1, line_image, 1, 1)

   plt.imshow(combo_image)
   cv2.imshow("result", combo_image)
   if cv2.waitKey(1) == ord("q"):
       break
cap.release()
cv2.destroyAllWindows()
```

Conclusion

We explored the foundations of lane detection with OpenCV in this document. The significance of lane detection, different preprocessing procedures, the use of the Hough Transform, difficulties, and real-time implementation were all explored.

A vital component of technology for improving traffic safety and enabling driverless cars is lane detection. It is essential for preventing collisions, helping drivers, and allowing self-driving cars to travel securely.

The lane-detecting field is always changing. Future developments could be,

1. Integration with cutting-edge sensors to improve dependability and precision.
2. Machine learning for better lane spotting in difficult situations.
3. Enhanced real-time processing power for systems that react and operate more quickly.

The field of lane detection with OpenCV is full of energy and promise for the future of transportation.

References

1. OpenCV | Real Time Road Lane Detection - GeeksforGeeks
2. Lane Detection | Papers With Code
3. [1807.01726] LaneNet: Real-Time Lane Detection Networks for Autonomous Driving
4. Real-Time Lane Detection for Autonomous Vehicles