Project Report on

Lane detection

Submission to THE ROBOTICS CLUB - SNIST as a part of TAB Major project-24

THE ROBOTICS CLUB

Integrating Knowledge...

THE ROBOTICS CLUB-SNIST

SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY

(AUTONOMOUS)

(Affiliated to JNTU University, Hyderabad)

Yamnampet, Ghatkesar, Hyderabad – 501301.

2024

CERTIFICATE

This is the project work titled **‘Lane detection’** by **Poojitha,Surya,Likhitha,Sangeetha,Sai varun,Mani kanth**. This is a record of the project work carried out by them during the year 2022-2023 as part of POST INDUCTION under the guidance and supervision of

Ms. Aishwarya
TAB Executive
(Image processing)

Mr. RAKESH
Chairman

Ms. SAATHVIKHA
vice-chairman

MR. BHARAGAV
Supervisor
&
MR. DEEKSHITH
Supervisor

Mr. Narayana
The President of
THE ROBOTICS CLUB

Dr. A. PURUSHOTHAM
Faculty Advisor
Mechanical Department

DECLARATION

The project work reported in the present thesis titled **"Lane detection"** is a record of work done by Team (team number) in **THE ROBOTICS CLUB** as a part of **TAB Major project.**

No part of the thesis is copied from books/ journals/ Internet and wherever the portion is taken, the same has been duly referred in the text. The report is based on the project work done entirely by TEAM and not copied from any other source.

ACKNOWLEDGMENT

This project report is the outcome of the efforts of many people who have driven our passion to explore into implementation of **Lane detection.** We have received great guidance, encouragement and support from them and have learned a lot because of their willingness to share their knowledge and experience. Primarily, we would like to express our gratitude to our executive **'AISHWARYA'**. His guidance has been of immense help in surmounting various hurdles along the path of our goal.

We thank our TAB heads **Mr. RAKESH and Ms. SAATHVIKHA** for being with us till the end of the project completion.

We also thank our faculty advisor **Dr. A. Purushotham**, Professor Mechanical Department, who encouraged us during this project by rendering his help when needed.

Contents

LANE DETECTION USING IMAGE PROCESSING

Poojitha,Surya,Likhitha,Sangeetha,Sai varun,Mani kanth

Problem Statement

The objective is to develop a computer vision system capable of detecting and visualizing lane markings from a video feed of a road scene. This involves capturing video frames, applying perspective transformations to get a bird's eye view, thresholding to isolate lane colors, and using a sliding window technique to identify lane positions accurately. The system should allow for dynamic adjustments of the HSV color thresholds through a user interface, enhancing its robustness across varying lighting conditions.

Abstract

This project presents a method for real-time lane detection using video footage of roadways. Utilizing OpenCV, we read frames from a video file and apply a series of image processing techniques to identify lane markings. The process begins with perspective transformation to achieve a bird's eye view of the road, followed by color thresholding in the HSV space to isolate lane colors. Dynamic trackbars enable real-time adjustment of HSV thresholds, enhancing adaptability. A histogram-based approach determines the base points of the lanes, and a sliding window technique tracks the lanes vertically along the frame. The result is a robust lane detection system capable of handling different road scenes and conditions.

Introduction

The advent of autonomous driving and advanced driver-assistance systems has revolutionized the automotive industry, promising increased safety, efficiency, and convenience. Central to the functionality of these systems is the ability to accurately detect and follow road lanes. Lane detection enables vehicles to stay within their designated lanes, make informed decisions about lane changes, and avoid collisions. However, the task of lane detection is fraught with challenges, including varying lighting conditions, occlusions from other vehicles, shadows, road surface variations, and complex road geometries.

Traditional lane detection methods, which often rely on simplistic edge detection or predefined templates, have proven inadequate in addressing these challenges. Consequently, there is a growing need for more sophisticated and reliable image processing techniques that can robustly detect lane markings in real-time and under diverse conditions.

This project aims to address these challenges by developing a lane detection system that employs a combination of advanced image processing techniques. The system utilizes edge detection algorithms to identify potential lane boundaries, applies color thresholding to enhance lane visibility, and uses the Hough Transform to accurately detect and characterize lane lines. By integrating these methods, the proposed system aims to achieve high accuracy and robustness in lane detection, thereby enhancing the overall safety and effectiveness of autonomous driving and ADAS technologies.

The subsequent sections of this report will detail the methodology, implementation, and evaluation of the proposed lane detection system. The effectiveness of the system will be demonstrated through extensive testing on various datasets, highlighting its capability to perform reliably in real-world driving scenarios.

Architecture

Software and Tools

Programming language: Python

Modules

cv2 (OpenCV)

Used for capturing video, performing image processing operations, creating trackbars, and displaying images.

numpy

Used for numerical operations, creating arrays, and handling image data.

Implementation and Working

Data Acquisition

Video Capture:

The video file "LaneVideo.mp4" is opened using cv2.VideoCapture.Frames are read in a loop until the video ends or an exit condition is met.

Preprocessing

Frame Resize:

Each frame is resized to 640x480 for consistent processing.

Perspective Transformation:

Four points (tl, bl, tr, br) are manually selected on the frame to define the region for perspective transformation.

These points are transformed to a bird's-eye view using cv2.getPerspectiveTransform and cv2.warpPerspective.

Lane Detection

HSV Thresholding:

The transformed frame is converted to HSV color space using cv2.cvtColor.

Lower and upper HSV thresholds are dynamically adjusted using trackbars.A binary mask is created using cv2.inRange to highlight lane lines based on HSV values.

Histogram Analysis:

A histogram of the lower half of the binary mask is computed using numpy.sum to identify the base positions of the left and right lanes.

The base positions are determined using numpy.argmax.

Sliding Window Technique:

Starting from the base positions, sliding windows are used to detect lane pixels.For each window, contours are found using cv2.findContours.

The centroids of the contours are calculated using cv2.moments.The base positions of the lanes are updated based on the detected centroids.

Visualization

Overlaying Results:

The detected lane lines are visualized by drawing rectangles on the binary mask.The original frame, bird's-eye view, binary mask, and sliding window results are displayed using cv2.imshow.

User Interaction

Dynamic Adjustment:

Trackbars are used to dynamically adjust the HSV threshold values for fine-tuning the lane detection.

Termination

Exit Condition:

The loop continues until the video ends or the 'Esc' key is pressed.All OpenCV windows are closed, and the video capture object is released.

Code:

```
import cv2
import numpy as np

vidcap = cv2.VideoCapture("LaneVideo.mp4")
success, image = vidcap.read()

def nothing(x):
    pass

cv2.namedWindow("Trackbars")

cv2.createTrackbar("L - H", "Trackbars", 0, 255, nothing)
cv2.createTrackbar("L - S", "Trackbars", 0, 255, nothing)
cv2.createTrackbar("L - V", "Trackbars", 200, 255, nothing)
cv2.createTrackbar("U - H", "Trackbars", 255, 255, nothing)
cv2.createTrackbar("U - S", "Trackbars", 50, 255, nothing)
cv2.createTrackbar("U - V", "Trackbars", 255, 255, nothing)

while success:
    success, image = vidcap.read()
    if not success:
        break

    frame = cv2.resize(image, (640, 480))

    # Choosing points for perspective transformation
    tl = (222, 387)
    bl = (70, 472)
    tr = (400, 380)
    br = (538, 472)

    cv2.circle(frame, tl, 5, (0, 0, 255), -1)
    cv2.circle(frame, bl, 5, (0, 0, 255), -1)
    cv2.circle(frame, tr, 5, (0, 0, 255), -1)
    cv2.circle(frame, br, 5, (0, 0, 255), -1)

    # Applying perspective transformation
    pts1 = np.float32([tl, bl, tr, br])
    pts2 = np.float32([[0, 0], [0, 480], [640, 0], [640, 480]])

    # Matrix to warp the image for birdseye window
    matrix = cv2.getPerspectiveTransform(pts1, pts2)
    transformed_frame = cv2.warpPerspective(frame, matrix, (640, 480))

    # Object Detection
    # Image Thresholding
    hsv_transformed_frame = cv2.cvtColor(transformed_frame, cv2.COLOR_BGR2HSV)
```

```
l_h = cv2.getTrackbarPos("L - H", "Trackbars")
l_s = cv2.getTrackbarPos("L - S", "Trackbars")
l_v = cv2.getTrackbarPos("L - V", "Trackbars")
u_h = cv2.getTrackbarPos("U - H", "Trackbars")
u_s = cv2.getTrackbarPos("U - S", "Trackbars")
u_v = cv2.getTrackbarPos("U - V", "Trackbars")

lower = np.array([l_h, l_s, l_v])
upper = np.array([u_h, u_s, u_v])
mask = cv2.inRange(hsv_transformed_frame, lower, upper)

# Histogram
histogram = np.sum(mask[mask.shape[0] // 2:, :], axis=0)
midpoint = histogram.shape[0] // 2
left_base = np.argmax(histogram[:midpoint])
right_base = np.argmax(histogram[midpoint:]) + midpoint

# Sliding Window
y = 472
lx = []
rx = []

msk = mask.copy()

while y > 0:
    # Left threshold
    img = mask[y-40:y, left_base-50:left_base+50]
    contours, _ = cv2.findContours(img, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    for contour in contours:
        M = cv2.moments(contour)
        if M["m00"] != 0:
            cx = int(M["m10"] / M["m00"])
            cy = int(M["m01"] / M["m00"])
            lx.append(left_base - 50 + cx)
            left_base = left_base - 50 + cx

    # Right threshold
    img = mask[y-40:y, right_base-50:right_base+50]
    contours, _ = cv2.findContours(img, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    for contour in contours:
        M = cv2.moments(contour)
        if M["m00"] != 0:
            cx = int(M["m10"] / M["m00"])
            cy = int(M["m01"] / M["m00"])
            rx.append(right_base - 50 + cx)
            right_base = right_base - 50 + cx

    cv2.rectangle(msk, (left_base - 50, y), (left_base + 50, y - 40), (255, 255, 255), 2)
    cv2.rectangle(msk, (right_base - 50, y), (right_base + 50, y - 40), (255, 255, 255), 2)
```

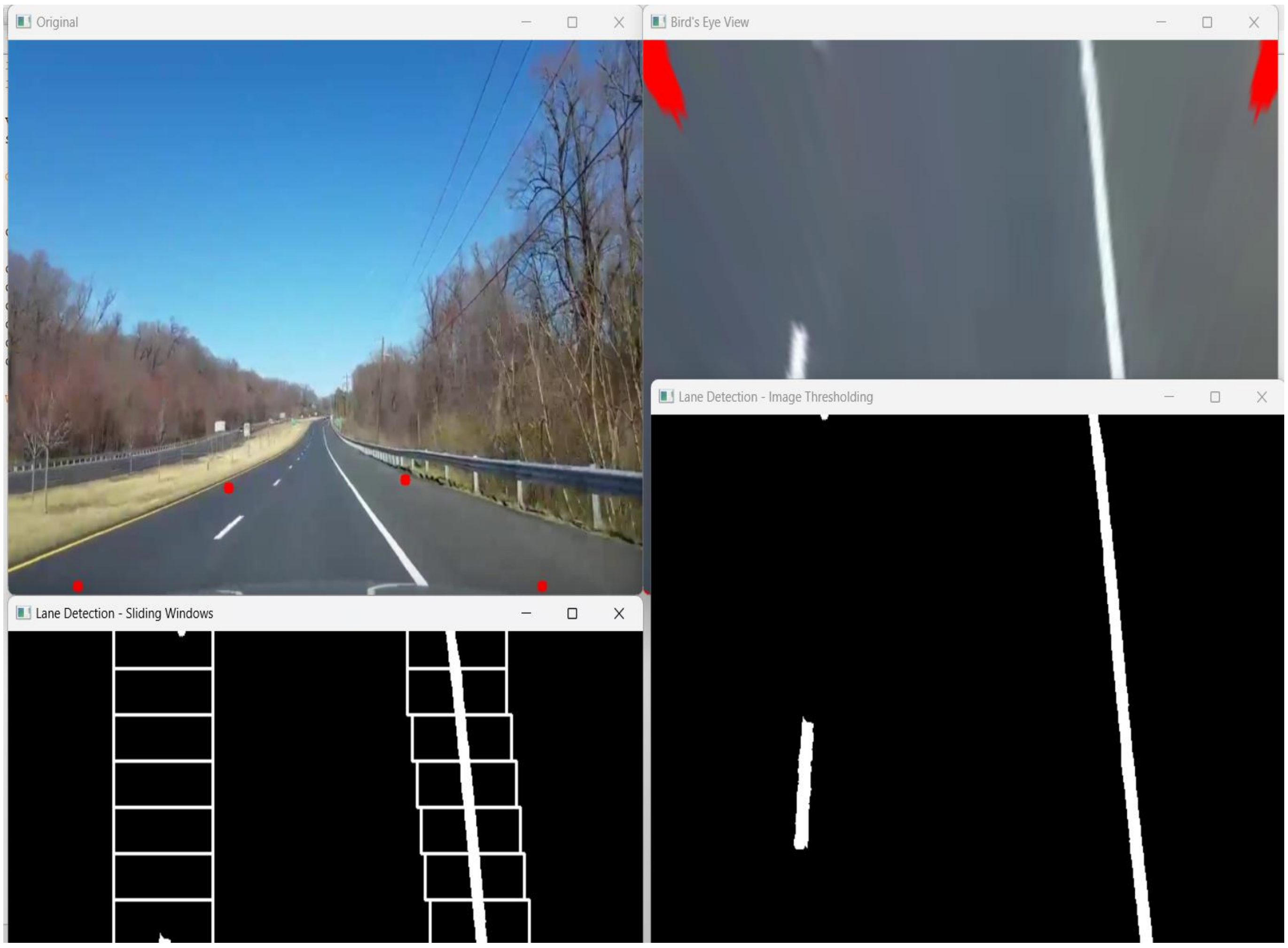
```
        y -= 40

    cv2.imshow("Original", frame)
    cv2.imshow("Bird's Eye View", transformed_frame)
    cv2.imshow("Lane Detection - Image Thresholding", mask)
    cv2.imshow("Lane Detection - Sliding Windows", msk)

    if cv2.waitKey(10) == 27:
        break

cv2.destroyAllWindows()
vidcap.release()
```

OUTPUT:


Original
Bird's Eye View
Lane Detection - Image Thresholding
Lane Detection - Sliding Windows

Conclusion

The lane detection project using image processing provides an effective solution for identifying and tracking lane lines in video streams. By employing a combination of perspective transformation, HSV thresholding, histogram analysis, and the sliding window technique, the system is able to accurately detect lane lines and overlay them on the original frame. The use of dynamic trackbars for real-time adjustment of HSV thresholding parameters enhances the system's adaptability to varying lighting conditions, making it more robust and reliable.

This project demonstrates a comprehensive approach to lane detection, highlighting key aspects of image preprocessing, lane line identification, and visualization. The modular design allows for easy adjustments and improvements, particularly in the thresholding stage, which is critical for handling different road and lighting scen