Image processing project

Omri Drori, Nili Alfia

February 2024

1 Introduction

Lane detection is an essential task that include various application like autonomous driving, lane departure warning systems, and road scene analysis for safety. Utilizing techniques such as edge detection, color segmentation, and Hough transform, computer vision algorithms can identify and track lane markings on roads captured by cameras mounted on vehicles.
In this project we focused on few features of lane detection.

2 Lane detection and Lane departure warning

2.1 Introduction to Lane change alert

Change Lane alert is a crucial task for autonomous driving and advanced driver-assistance systems (ADAS). It involves identifying lane boundaries on the road to provide feedback to the vehicle's control system or the driver. The driver should know if he changed lane (even if he meant to or he accidentally deviated from the lane). The provided algorithm detects lanes within a defined region of interest using edge detection and line detection techniques. It identifies lane changes by comparing the positions of lane markings between consecutive frames. When a lane change is detected, a corresponding alert message is displayed on the video frame to notify the driver or the autonomous vehicle system.

2.2 Algorithm Overview - Line Detect

The provided algorithm has a few steps as describes blow:
Preprocessing: First convert the input video frame to grayscale.Then Apply Gaussian blur to reduce noise. Last step is to use Canny edge detection to detect edges in the image.
Region of Interest (ROI) Selection: Define a squared polygonal region of interest (ROI) within the frame where lane detection will be performed. The absolute numbers are hyper-parameters based on experiment and based on the video x and y axis (the width and height).

Masking: Apply a mask to the Canny edge-detected image to isolate the region of interest.
Hough Transform: Use Hough transform to detect lines.

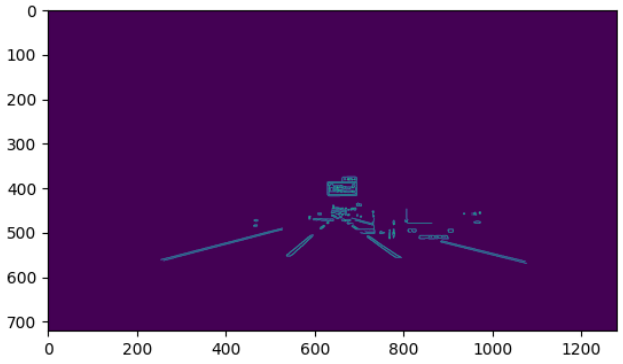

Figure 1: After last step of preprocess - the part of the frame that detected (included road lines)

2.3 Algorithm Overview - Lane change detection based on Midpoint algorithm:

Hyper-Parameters:
Based on the video file we made experiments for few hyper-parameters:
1. The change in x-axis pixels - configured to be 40. This determine the detected change in lanes.
2. lane-change-display-frames-counter: The number of frames that we want to show the detected change. Configured to be 70. We wanted continuous message display on the image.
3. A slope to consider horizontal line: we want an enough value to consider a slope as horizontal. We chose 0.1 (above this is not horizon).

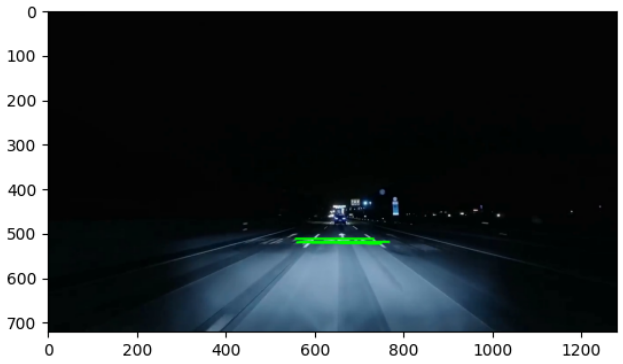

Figure 2: The horizontal lines we want to avoid (before the fix)

Main Steps:
First classify detected lines based on their slope. The slop helps us to find horizon lines.
Calculate midpoints for each detected line.
Compare the midpoints of current and previous frames to detect lane changes. Then determine the direction of the lane change (left or right).
Alert Display:
Display a lane change alert message on the video frame when a lane change is detected. Display the message for a specified number of frames.

Figure 3: Start move to the left - Midpoint distance change

3 Lane detection in curve road in varing lighting conditions

3.1 Preprocessing for Curve Detection

Preprocessing is a crucial step to enhance the image quality and improve the accuracy of lane detection in curved road scenarios. This subsection outlines the methods used to preprocess the images before detecting lanes.

Gamma Correction: To adjust the brightness of the images, gamma correction is applied. This process modifies the luminance of the pixels, making the lane markings more distinguishable from the road surface, especially in varying lighting conditions.

Color Space Conversion: The images are converted from BGR to different color spaces such as HSV (Hue, Saturation, Value) and HLS (Hue, Lightness, Saturation). These conversions are beneficial for isolating features like lane markings based on their color characteristics.

Normalization and Histogram Equalization: To ensure consistency in image brightness and contrast, normalization is performed on the color channels. Additionally, Adaptive Histogram Equalization (CLAHE) is used to enhance the contrast, making it easier to identify lane markings.

Noise Reduction: A Gaussian blur is applied to the images to smooth out noise. This step is essential for reducing the impact of irrelevant features and artifacts that could interfere with lane detection.

Edge Detection and Masking: The processed images are then subjected to

edge detection to highlight the lanes. A mask is applied to focus on regions of interest, such as the road ahead, minimizing the impact of surrounding objects and scenarios not relevant to lane detection.
These preprocessing steps are integral for preparing the images for the subsequent lane detection process, especially under the complex conditions presented by curved roads. All the results after each step of the preprocessing piplines can be viewed in image 1-11.

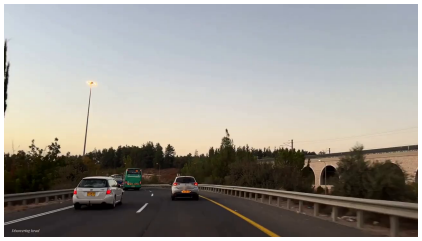

Figure 4: Original Frame

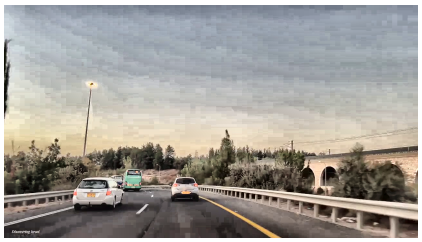

Figure 5: After Histogram Equalization

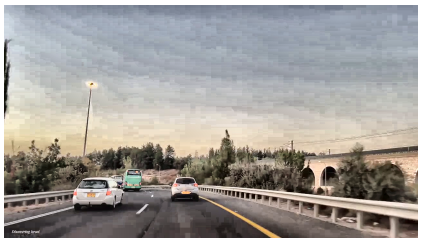

Figure 6: After Normalization

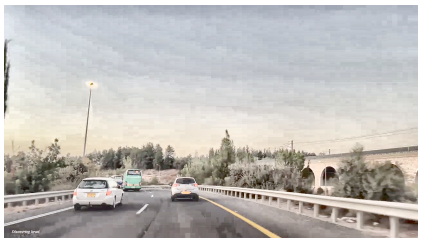

Figure 7: After Gamma Correction

Figure 8: After Gray Scale Conversion

Figure 9: Saturation Channel

Figure 10: Lightness Channel

Figure 11: After Gaussian Blur

Figure 12: Masked Frame

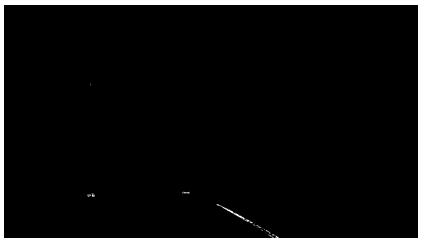

Figure 13: Filtering Based on HSV Yellow Parts

Figure 14: Region of Interest

3.1.1 Handling Extreme Lighting Conditions

Towards the end of the video sequence (to compare one frame example is showed in image 12 and 13 to compare with image 1, there is a segment where the lighting conditions differ drastically from previous scenes. The change is so significant that traditional preprocessing methods like gamma correction and histogram equalization are unable to adequately compensate for the disparities in illumination. For these specific frames, a distinct preprocessing pipeline with varied threshold values is employed to accurately detect the lane lines.
This tailored approach involves setting unique lower and upper bounds for color detection thresholds, which are better suited for the extreme lighting conditions encountered.

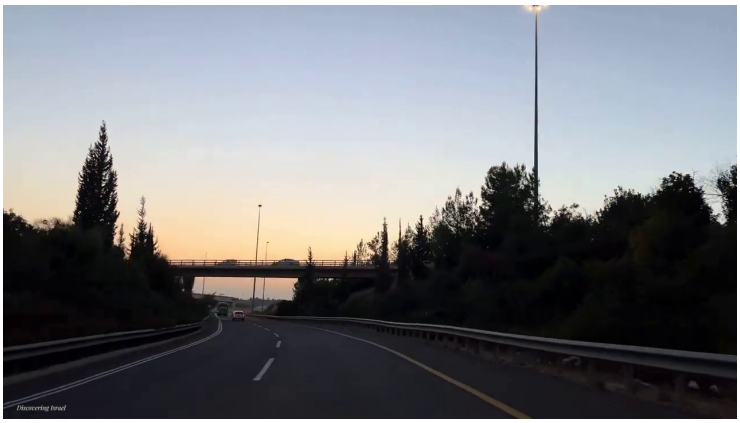

Figure 15: Extremly different lighting condition

Figure 16: Another extremly different lighting condition

3.2 Lane Detection Algorithm: Sliding Window Technique

The Sliding Window technique is a pivotal component in our curved lane detection algorithm, particularly adept at handling curved lanes as opposed to straight lines. This method was chosen over techniques such as the Hough Transform because curved lanes do not conform to the linear model that the Hough Transform is based on. Curved lanes require a more flexible approach that can adapt to the curvature.

Why Sliding Window? The Sliding Window approach divides the image vertically into a stack of horizontal slices and then searches for lane pixels starting from the bottom slice upwards. For each slice, a window is placed around the peak of the histogram of pixel intensities, which likely represents the lane's base. This window is slid left or right within each slice to follow the curvature of the lane. By doing so, the algorithm can "track" the lane lines even as they curve.

Implementation Details: The process begins by computing a histogram of

the bottom half of the binary image, which provides a starting point for the left and right lanes. Windows are then placed around these starting points and slid vertically to cover the entire height of the image. Within each window, if the number of pixels exceeds a certain threshold ($minpix$), the window's position is re-centered to the mean position of those pixels. This re-centering step allows the window to follow the curve of the lane line. The indices of the pixels within these windows are collected and used to fit a polynomial that models the lane line.

3.3 Polynomial Fitting:

Once the lane pixels are identified, a second-order polynomial is fitted to these points for both the left and right lanes. This obained using least square.
Temporal Smoothing: To account for the variability in detection from frame to frame, the algorithm employs a smoothing technique where the current polynomial coefficients are averaged with those of the previous frame, weighted more heavily towards the historical data. Specifically, if the lane detection for the current frame is successful, the polynomial coefficients left_fit and right_fit are updated as a weighted sum of the new and previous coefficients. For example the formula for updating the left lane's polynomial:

$$\text{left\_fit} = 0.1 \times \text{current\_left\_fit} + 0.9 \times \text{prev\_left\_fit}$$

4 Crosswalk Detection

4.1 Introduction to Crosswalk Detection

Crosswalk detection is a significant task for autonomous driving and advanced driver-assistance systems (ADAS) as it involves identifying pedestrian crossing zones for safety and navigation purposes. The algorithm developed aims to detect crosswalk lines using a combination of computer vision techniques to process video data in real-time.

4.2 Algorithm Overview

The crosswalk detection algorithm utilizes a series of image processing steps to identify the characteristic patterns of crosswalks, such as the elongated rectangular shapes of the white stripes on the road surface.
Region of Interest (ROI) Extraction: The first step involves defining a polygonal region of interest (ROI) on the frame to focus on the road area where crosswalks are typically found. This step reduces the computational load by ignoring regions of the frame that are not relevant to the task.
Color Thresholding: The algorithm then applies color thresholding within the ROI to isolate the white/gray colors of the crosswalk stripes from the rest of the scene. The chosen thresholds are designed to capture the color range of crosswalk stripes under various lighting conditions.

Contour Analysis: After thresholding, the algorithm performs contour analysis to find continuous regions corresponding to the crosswalk stripes. It filters out irrelevant contours based on area and aspect ratio criteria, retaining only those that are elongated and have an area within a specified range.
Morphological Operations: To enhance the detection, morphological operations such as erosion and closing are applied. Erosion helps remove small shapes and noise, while closing bridges the gaps between the stripes, ensuring that each stripe is detected as a single, coherent contour.
Final Contour Processing: The final contours, which represent potential crosswalk stripes, are processed to calculate their minimum area bounding rectangles. These rectangles are then drawn onto the frame to visually confirm the detection of the crosswalk. The results of the detection for the two crosswalks in our video can be viewed in images 13 and 14.

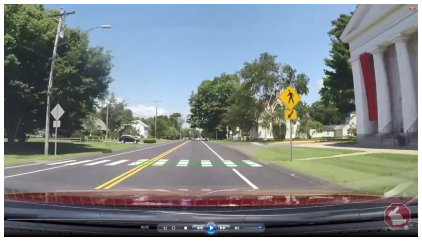

Figure 17: First cross walk detection

Figure 18: Second cross walk detection