Table Of Contents

[Lane Detection 2](#_Toc187795919)

[Overview 2](#_Toc187795920)

[Key Functionalities 3](#_Toc187795921)

[Camera Calibration 3](#_Toc187795922)

[Image preprocessing 3](#_Toc187795923)

[Edge Detection 3](#_Toc187795924)

[Perspective Warp 4](#_Toc187795925)

[Lane Separator 4](#_Toc187795926)

[Curve Fit 4](#_Toc187795927)

[Temporal Analysis 4](#_Toc187795928)

[Lane Change Detection 4](#_Toc187795929)

[Crosswalk Detection 5](#_Toc187795930)

[Overview 5](#_Toc187795931)

[Key Functionalities 6](#_Toc187795932)

[Edge Detection & Line Extraction 6](#_Toc187795933)

[Horizontal Line Filtering 6](#_Toc187795934)

[Parallel Pair Detection 6](#_Toc187795935)

[Crosswalk Bounding Box Identification 6](#_Toc187795936)

[Intensity Based Stripe Verification 6](#_Toc187795937)

[Temporal Detection State and Smoothing 7](#_Toc187795938)

[Overall Pipeline integration, Robustness and loose coupling 7](#_Toc187795939)

[Results 8](#_Toc187795940)

Table Of Figures

[Figure 1: Camera Calibration - checkboard 5](#_Toc187617486)

[Figure 2: Left - distorted lanes, Right - undistorted lanes 6](#_Toc187617487)

[Figure 3: Left - Edges after Image Processing and Canny, Right – Edges after Perspective Warp and ROI Cropping 6](#_Toc187617488)

[Figure 4: Temporal visualization to determine the correct threshold for lane change detection 7](#_Toc187617489)

[Figure 5: Temporal Median Grayscale Intensity to understand the range of shadowed regions for shadow removal 8](#_Toc187617490)

[Figure 6: Region of Interest Visualization for debug 9](#_Toc187617491)

# Lane Detection

## Overview

In this task, we implement a comprehensive pipeline for lane detection using computer vision techniques. The system is designed to operate under both day and night conditions, handling various challenges such as shadows, lane curvature, and lane changes. The pipeline utilizes Python and OpenCV, integrating several preprocessing and image processing steps to achieve robust lane detection.

We implement the following process:

1. **Camera Calibration**: Undistort the original image, correcting for lens distortion and enhancing accuracy in lane detection.
2. **Preprocessing and Edge Detection**: Adjust image contrast using histogram equalization techniques like CLAHE to enhance visibility under varying conditions. Convert images to grayscale and use adaptive thresholding, Gaussian blur, and the Canny edge detector to detect lane edges.
3. **Perspective Transformation**: Apply a perspective warp to create a bird's-eye view of the road, simplifying the detection of lane lines and curves.
4. **Lane Detection Pipeline**: Detect left and right lane boundaries using the sliding window technique. Fit polynomial curves to the detected lane points to model lane boundaries.
5. **Temporal Analysis**: Compute a temporal median from multiple frames to reduce noise and adapt to transient changes in brightness. Detect lane changes by analyzing the deviation in fitted lane curves across frames. Determine threshold for lane change using temporal analysis.
6. **Video Processing**: Process video frames to overlay detected lanes and lane change indicators and save the results to a video file.

## Key Functionalities

### Camera Calibration

Camera lenses distort incoming light to focus it onto the camera sensor, often introducing minor inaccuracies in light distortion. This can lead to measurement errors in computer vision tasks. Camera calibration mitigates this by generating a distortion model that corrects for lens distortions using a known reference object, such as an asymmetric checkerboard *Figure 1: Camera Calibration - checkboard*.

### Image preprocessing

Conversion to the LAB color space and enhancement of the L-channel using CLAHE to improve visibility of lanes. In day time shadow removal is applied before CLAHE to increase edges contrast. Masking of white lane pixels using HSV color space. Adaptive thresholding to account for different lighting conditions. Gaussian blurring to reduce noise. *Figure 2: Left - distorted lanes, Right - undistorted lanes*

### Edge Detection

Canny edge detection method to identify edges in the blurred image. The intensity thresholds for edge detection are dynamically calculated based on the median pixel intensity of the image. This makes the algorithm adaptive to varying lighting conditions, which is crucial for robustness in day and night scenarios.

### Perspective Warp

Detecting curved lanes directly in the camera’s perspective can be challenging. By applying a perspective transformation, we obtain a bird’s-eye view of the lanes, simplifying the task. Assuming the lane exists on a flat 2D surface, a polynomial curve can then be fitted to accurately represent the lane boundaries. This module handles both perspective warping and ROI extraction. *Figure 3: Left - Edges after Image Processing and Canny, Right – Edges after Perspective Warp and ROI Cropping*

*Figure 6: Region of Interest Visualization for debug*

### Lane Separator

After acquiring the birds eye view of the ROI of the edges we use the sliding window algorithm to differentiate between the left and right lane boundaries so that we can fit two different curves representing the lane boundaries.

The algorithm itself is as follows: Starting from the initial position, the first window measures how many pixels are located inside the window. If the amount of pixels reaches a certain threshold, it shifts the next window to the average lateral position of the detected pixels. If insufficient pixels are detected, the next window starts in the same lateral position. This continues until the windows reach the other edge of the image.

### Curve Fit

After acquiring the lanes, this part is straight forward, we apply polynomial regression to the left and right lane pixels individually using of degree 2 following the logic that when the equation becomes a good estimation of a straight line and when we get a curve.

### Temporal Analysis

The temporal median is calculated over a fixed history of grayscale frames to estimate overall scene brightness and determine day or night conditions.

Temporal median filtering smooths out noise across frames, making the system more robust to transient artifacts.

### Lane Change Detection

Lane change detection is implemented by monitoring significant deviations in the fitted curves over time, with thresholds set for flagging changes.

*Figure 4: Temporal visualization to determine the correct threshold for lane change detection*

*Figure 5: Temporal Median Grayscale Intensity to understand the range of shadowed regions for shadow removal*

# Crosswalk Detection

## Overview

This task implements a comprehensive pipeline for detecting crosswalks in video frames, integrating seamlessly with the existing lane detection module. The system is designed to operate under varying lighting conditions (such as the lane detection) by leveraging computer vision techniques for robust feature extraction, spatial transformations, and temporal smoothing. The pipeline is implemented in Python using OpenCV and includes several steps ranging from region-of-interest selection to final bounding box overlay on top of lane-detected frames.

We implement the following process:

1. **Frame Preprocessing**: Adjust image contrast using histogram equalization techniques like CLAHE to enhance visibility under varying conditions. Convert images to grayscale and use adaptive thresholding and Gaussian blur.
2. **Perspective Transformation**: Apply a perspective warp to create a bird's-eye view of the road, simplifying the detection of approaching crosswalks.
3. **Crosswalk Detection Pipeline**:
   1. **Edge Detection and Line extraction:** apply Canny edge detection on the warped image (masked with the defined ROI) to highlight potential crosswalk boundaries – horizontal line segments extraction with probabilistic Hough transform.
   2. **Parallel Line pairing:** Compute slopes of the detected lines and group those whose slopes are nearly identical, identify potential parallel line pairs that are spaced consistently within a predefined threshold (e.g., 35–100 pixels).
   3. **Candidate crosswalk detection:** Combine consecutive parallel line pairs, measuring vertical spacing to ensure it is within the valid range for typical crosswalk patterns. Generate bounding boxes around these candidate crosswalk regions.
   4. **Stripe verification:** Using intensity analysis, crop each candidate bounding box and analyze its intensity profile (row-wise or column-wise) to identify periodic striping, a hallmark of crosswalks. Discard regions that do not exhibit the expected stripe spacing or have an invalid aspect ratio (e.g., less than a specified width-to-height ratio).
4. **Temporal Analysis**: Maintain a short history of crosswalk detections using a sliding window. This helps filter out spurious results and ensures consistent detections across frames. Use a smoothing factor (α) to interpolate the crosswalk bounding box position over time, reducing jitter in bounding box placement.
5. **Video Processing**: Process video frames to overlay detected crosswalks and save the results to a video file.

## Key Functionalities

### Edge Detection & Line Extraction

After warping, the image undergoes Canny edge detection to highlight potential crosswalk boundaries. The Probabilistic Hough Transform then locates almost vertical line segments with an adjustable tollerance threshold for dealing with image distortion and various camera angles, filtering by parameters such as minLineLength and maxLineGap to reduce false positives.

### Horizontal Line Filtering

Crosswalk stripes usually appear horizontally in the warped view. Detected lines are examined for near-constant y-coordinates, with an adjustable parrallel tolerance variable, leaving only horizontal (or nearly horizontal) lines. This step greatly narrows the search space for potential crosswalk patterns.

### Parallel Pair Detection

The system calculates the slope of each horizontal line and groups those whose slopes match within a small tolerance. Such parallel pairs—appearing at consistent vertical distances—are strong indicators of crosswalk-like structures.

### Crosswalk Bounding Box Identification

Analyze consecutive parallel pairs, determine whether the spacing between the lines are falling within a pre-determine specified range. When suitable spacing is confirmed, bounding boxes are created around these regions, generating potential crosswalk candidates.

### Intensity Based Stripe Verification

Each candidate bounding box is cropped and subjected to an intensity profile analysis. By summing intensities row-wise (each value contains the sum of the entire column pixel intesity).

Then by reducing the median value, we look for periodic zero-crossings.

the algorithm detects the characteristic striping pattern of crosswalks. Candidates without a valid frequency (e.g., stripe spacing outside intensity\_stripe\_tolerance) or an inappropriate shape ratio (valid\_crosswalk\_shape\_ratio) are discarded.

### Temporal Detection State and Smoothing

A separate and dedicated module tracks all detection results over a short frame history window (that can be adjusted as needed). This way we aim to avoid flickering or transient false positives. Each frame inspection updates a moving window, and only after several consistent hits over multiple frames, and in accordance with a 60% hit rate across the recent history (3 out of 5), the system displays a crosswalk detection alert.

A smoothing factor applied over the crosswalk rectangle position points to further stabilize the bounding box position from frame to frame.

### Overall Pipeline integration, Robustness and loose coupling

Once a crosswalk is confirmed, its bounding box is overlaid back onto the original perspective using the inverse homography (Minv). This output is then combined with an existing lane detection pipeline, allowing the final video to display both detected crosswalks and lane boundaries in real time. Debug images and annotated frames can be saved for diagnostic and visualization purposes.

In addition, the crosswalk detection pipeline has been implemented with a Debugger Class and various adjustable fields to allow a compatible and robust approach for using the crosswalk detector in various videos.

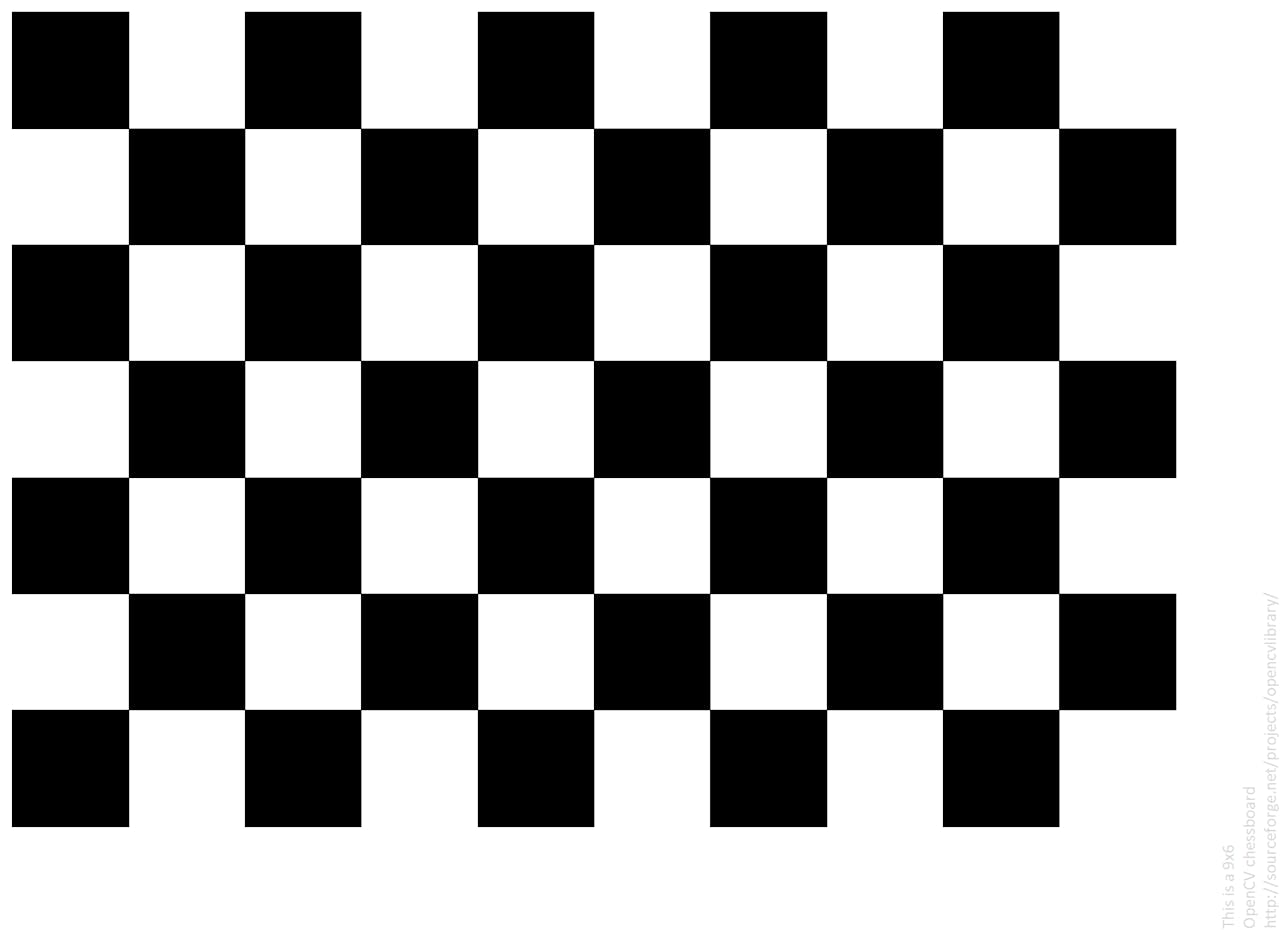
Results

Figure 1: Camera Calibration - checkboard

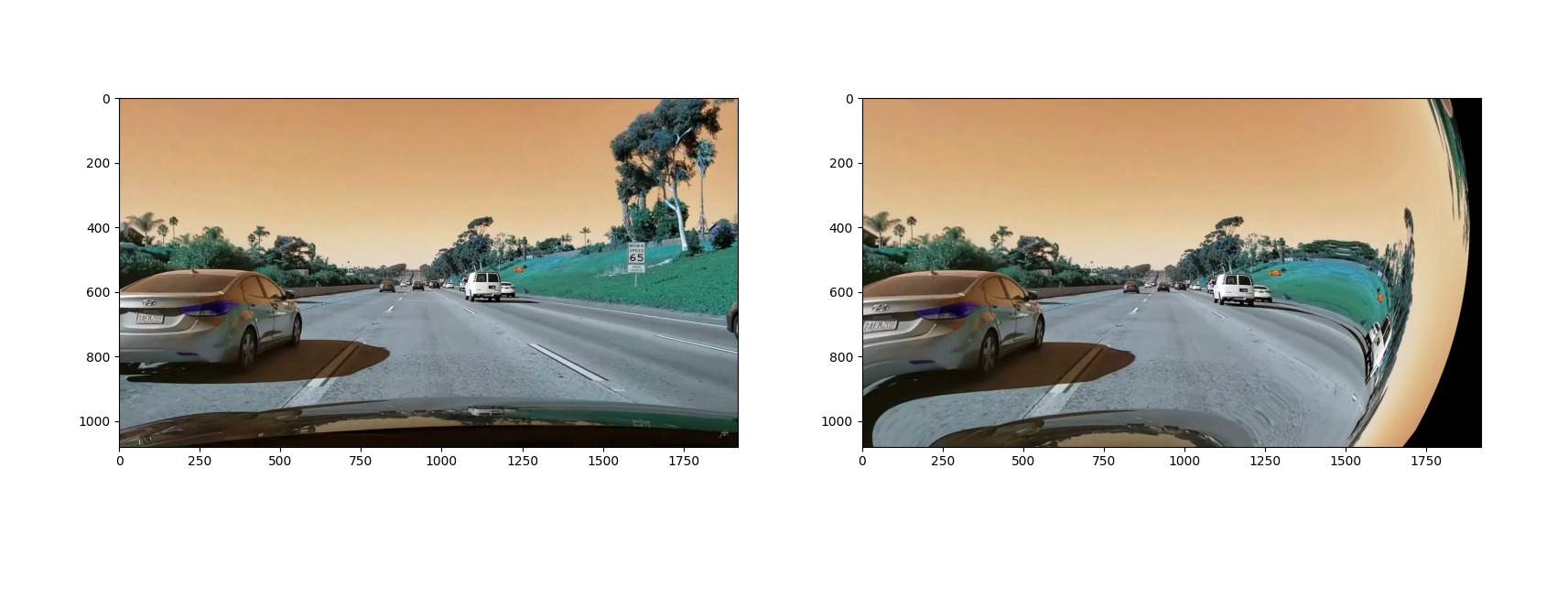


Figure 2: Left - distorted lanes, Right - undistorted lanes

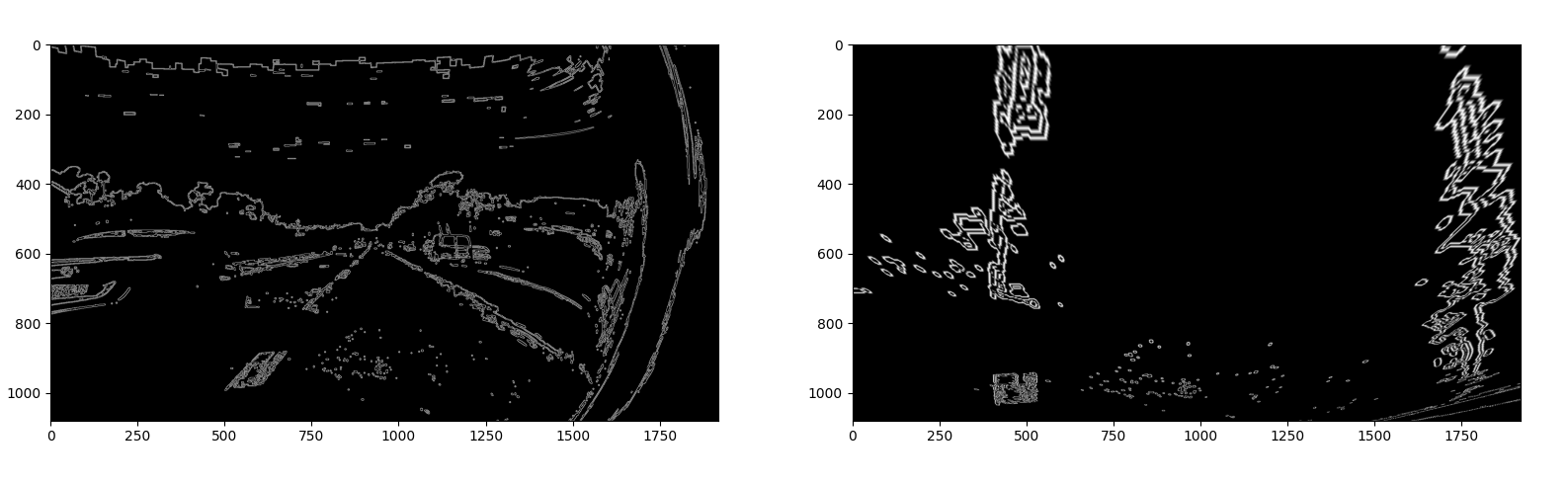


Figure 3: Left - Edges after Image Processing and Canny, Right – Edges after Perspective Warp and ROI Cropping

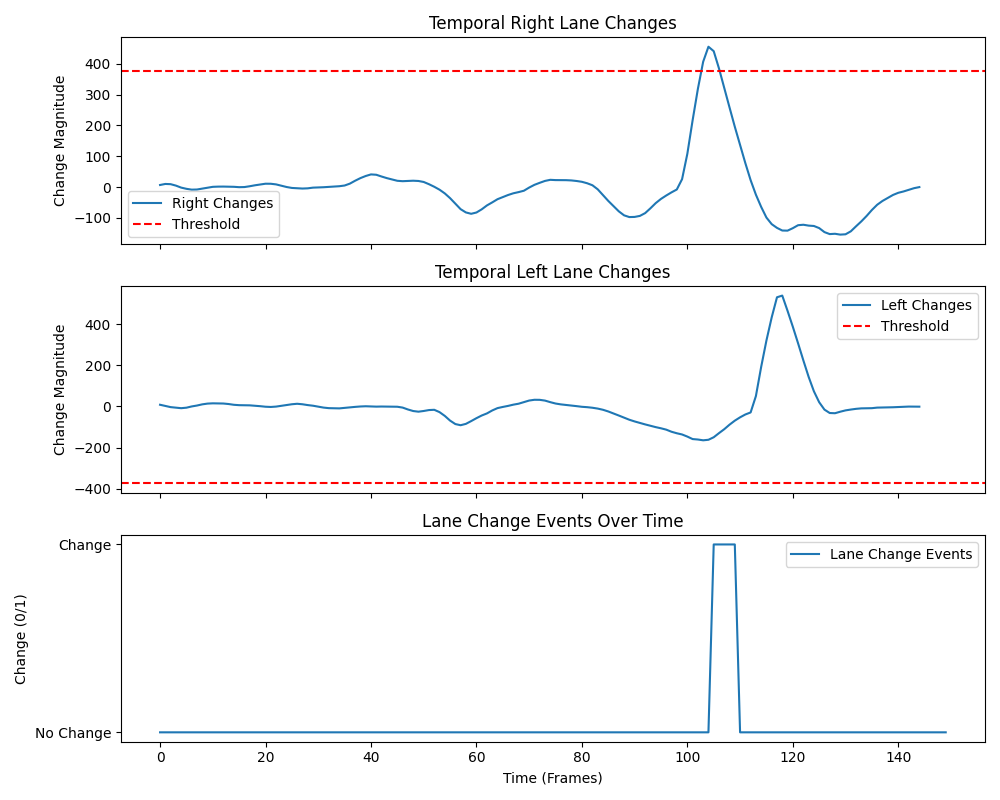


Figure 4: Temporal visualization to determine the correct threshold for lane change detection

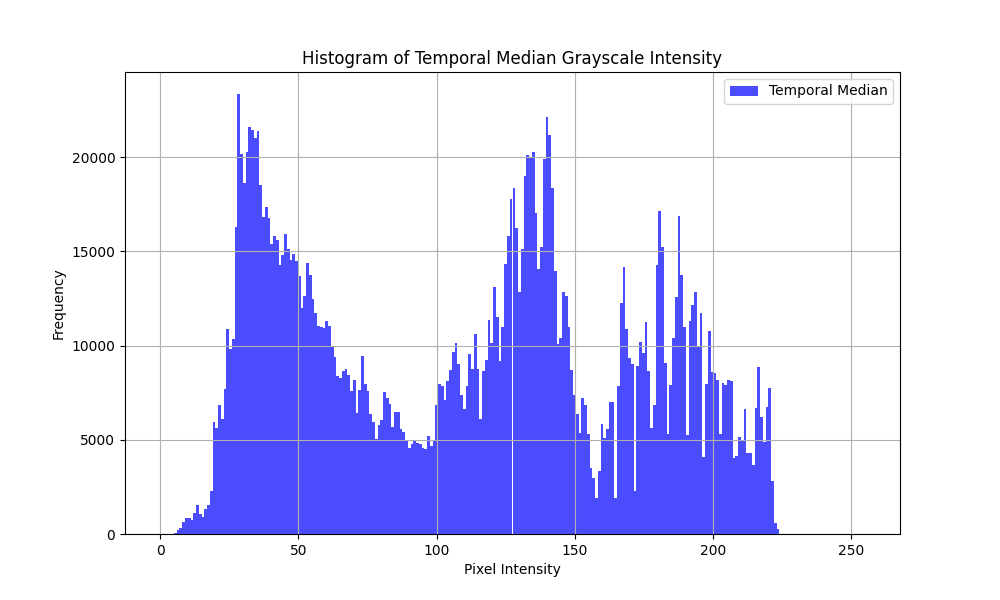


Figure 5: Temporal Median Grayscale Intensity to understand the range of shadowed regions for shadow removal

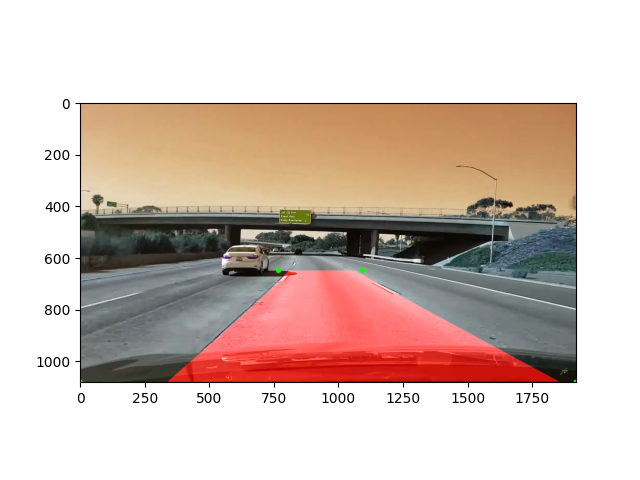


Figure 6: Region of Interest Visualization for debug









