Introduction to Vehicles and Sensors

Lane Detection and Tracking for Advanced Driver Assistance Systems

Muhammad Faran Akram (dolw0e)

Abstract:

This report presents a classical computer vision pipeline for real-time lane detection and lateral offset estimation as part of a Lane Keeping Assistance (LKA) system. The approach leverages image preprocessing, region-of-interest (ROI) masking, perspective transformation, histogram-based lane pixel extraction, and polynomial fitting to identify lane boundaries and estimate vehicle positioning. Qualitative evaluation on test videos demonstrates the effectiveness of the pipeline under controlled conditions and highlights challenges under varying illumination and road geometry. The report concludes with a discussion on operational design domain (ODD), performance limitations, and a comparison with deep learning-based lane detection approaches.

Introduction:

Lane detection is a fundamental component of Advanced Driver Assistance Systems (ADAS), enabling functionalities such as Lane Departure Warning (LDW) and Lane Keeping Assistance (LKA). The primary objective of this work is to design a classical image-processing-based pipeline capable of detecting lane markings in real-world driving videos and estimating the lateral offset of the vehicle from the lane center.

The project builds a complete perception pipeline, from raw camera frames to a lane overlay and offset computation, using Python and OpenCV. The implementation emphasizes interpretability, algorithmic clarity, and modular design to facilitate potential integration into an LKA control framework.The goals of this study are:

1. To develop a robust, modular lane detection pipeline using OpenCV.

2. To tune preprocessing thresholds and geometric transformations for different lighting and road conditions.
3. To analyze the system's performance on real driving footage.
4. To compare classical vision-based methods with modern learning-based alternatives in terms of robustness and explainability.

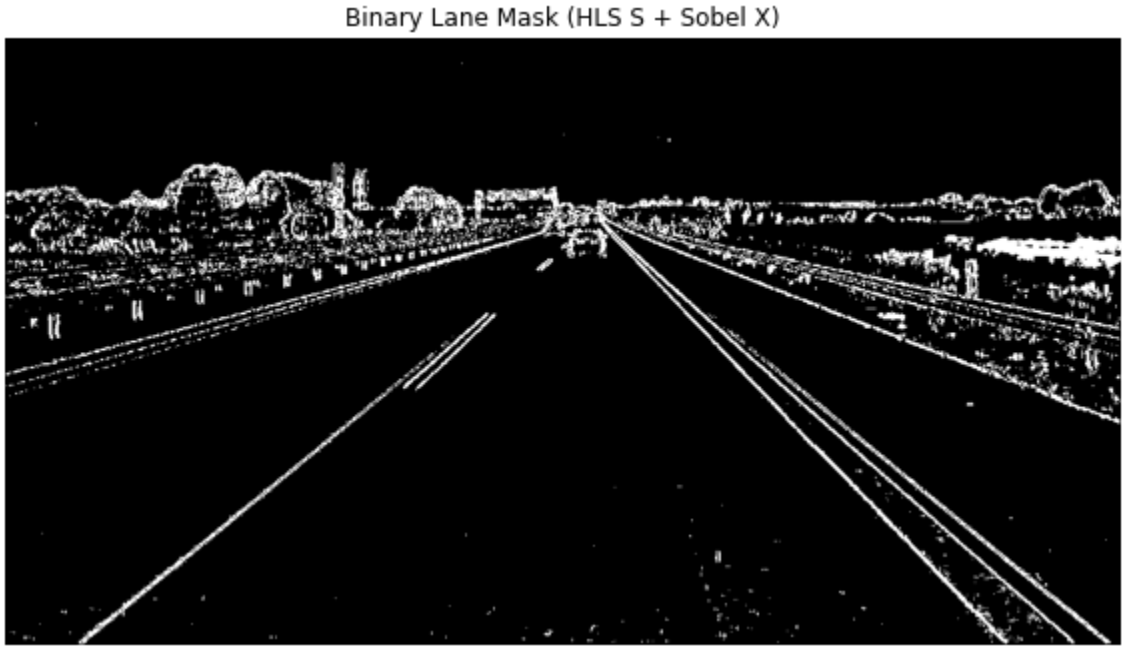
Methodology:

1. **Image Preprocessing and Thresholding:**

   Raw RGB frames are first processed through a color and gradient thresholding pipeline to highlight lane markings. The method leverages:

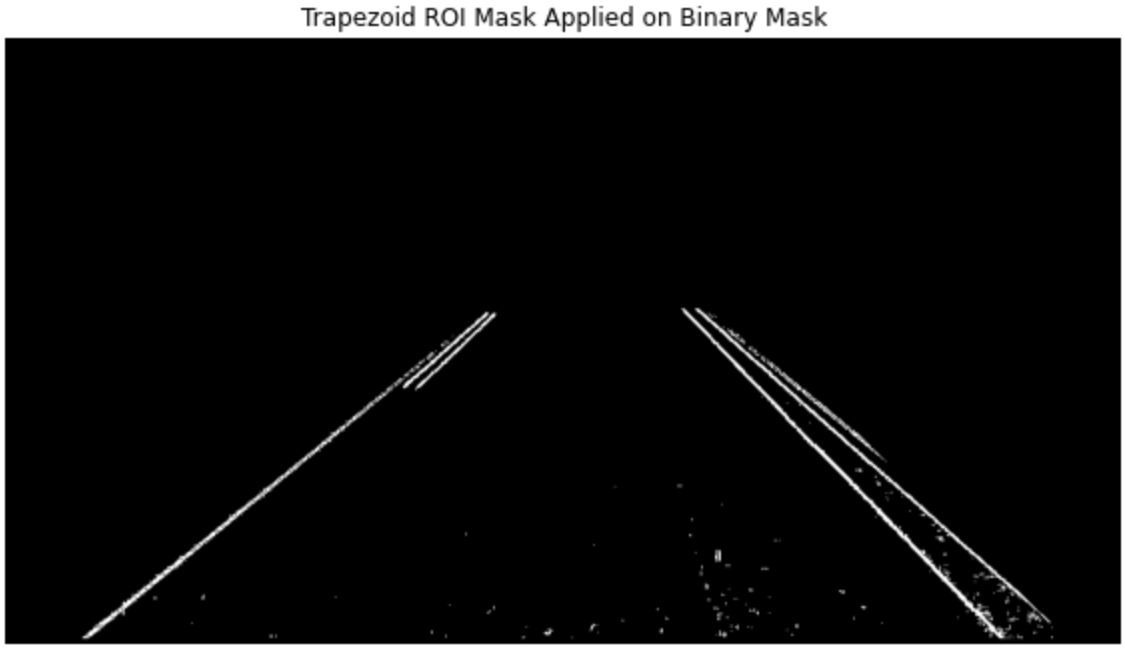
   - S channel thresholding in the HLS color space to capture bright, saturated lane markings under various lighting conditions.
   - Sobel X gradient thresholding on the L channel to extract vertical intensity changes typical of lane edges.

   Both binary masks are combined to generate a composite binary image emphasizing lane features:



2. **Region of Interest (ROI) Masking:**

To suppress irrelevant background features, a trapezoidal ROI is defined corresponding to the expected position of lane markings. The function ***apply_trapezoid_roi()*** applies a binary mask to retain only pixels within the defined region. This step ensures computational efficiency and improves the reliability of subsequent lane extraction.

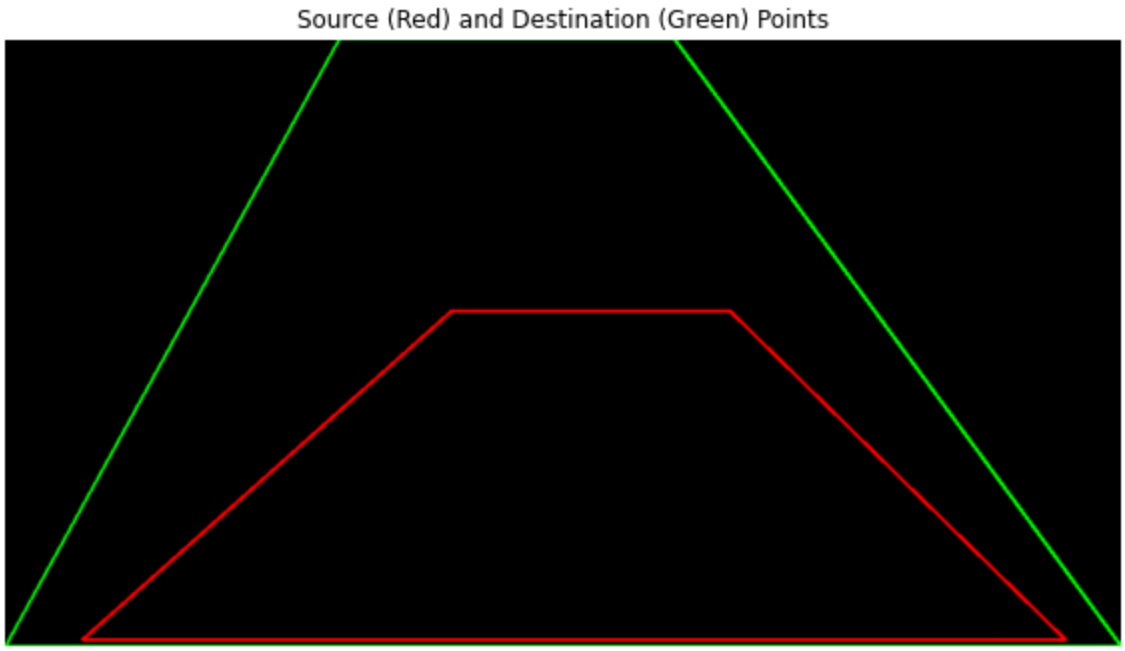


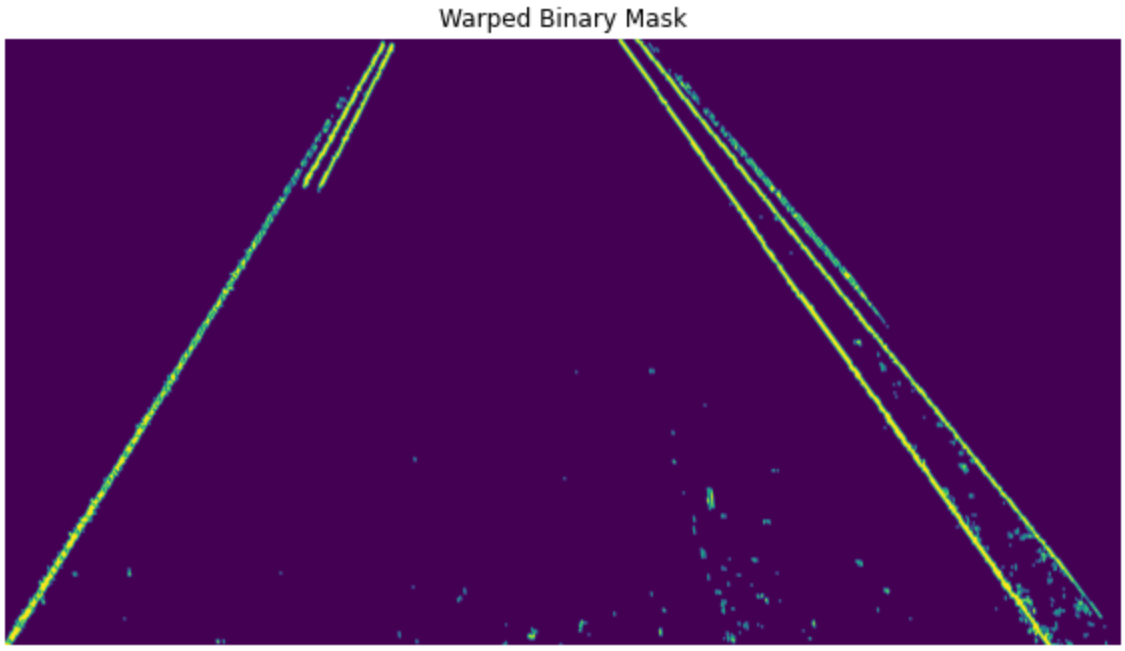
3. **Perspective Transformation (IPM):**

The masked image is then transformed into a bird's-eye view using an Inverse Perspective Mapping (IPM), implemented via OpenCV's cv2.getPerspectiveTransform() and cv2.warpPerspective() functions.

The transformation is defined by four source points (src) in the input image and destination points (dst) forming a rectangle in the top-down view.

This process rectifies perspective distortion, making lane lines appear parallel and simplifying polynomial fitting.





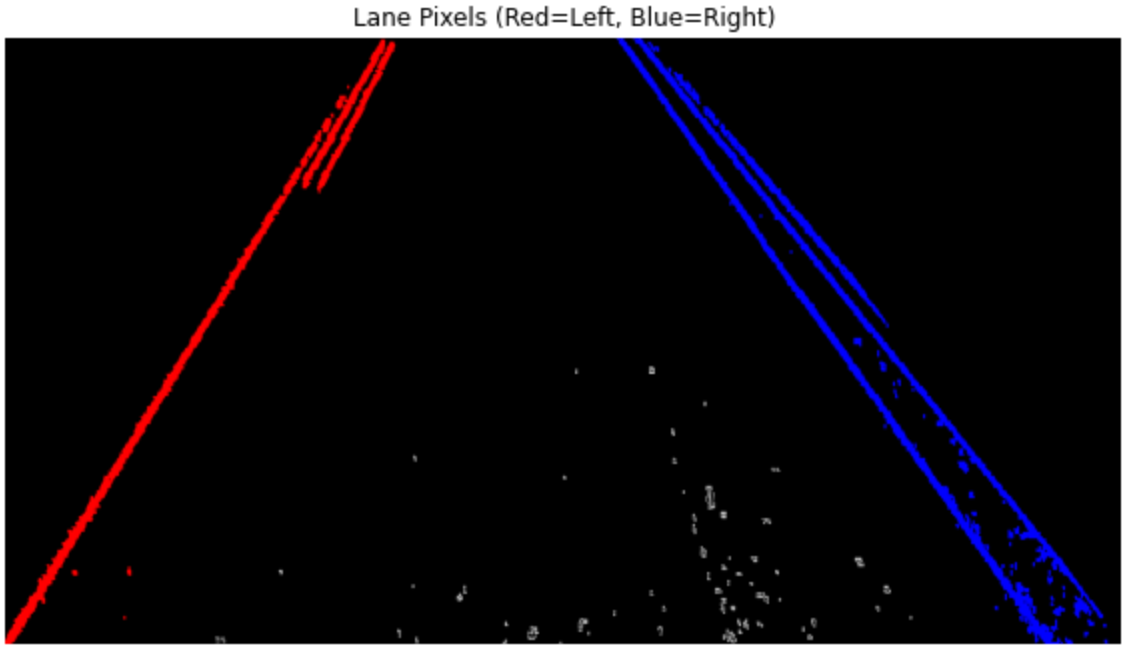
4. **Lane Pixel Extraction via Sliding Window Search**

   Lane pixels are extracted using a **histogram-based sliding window approach**:

1. The histogram of the lower half of the warped binary image identifies peak x-locations for left and right lane bases.

2. Sliding windows (typically 9) are then moved upward along the image height.
3. Non-zero pixels within a fixed margin (≈100 px) around each window are classified as lane points.
4. Window centers are dynamically recentered based on detected pixel means.

This approach, implemented in the function `find_lane_pixels()`, effectively isolates lane points even under mild occlusion or noise.



5. **Polynomial Fitting and Confidence Estimation**

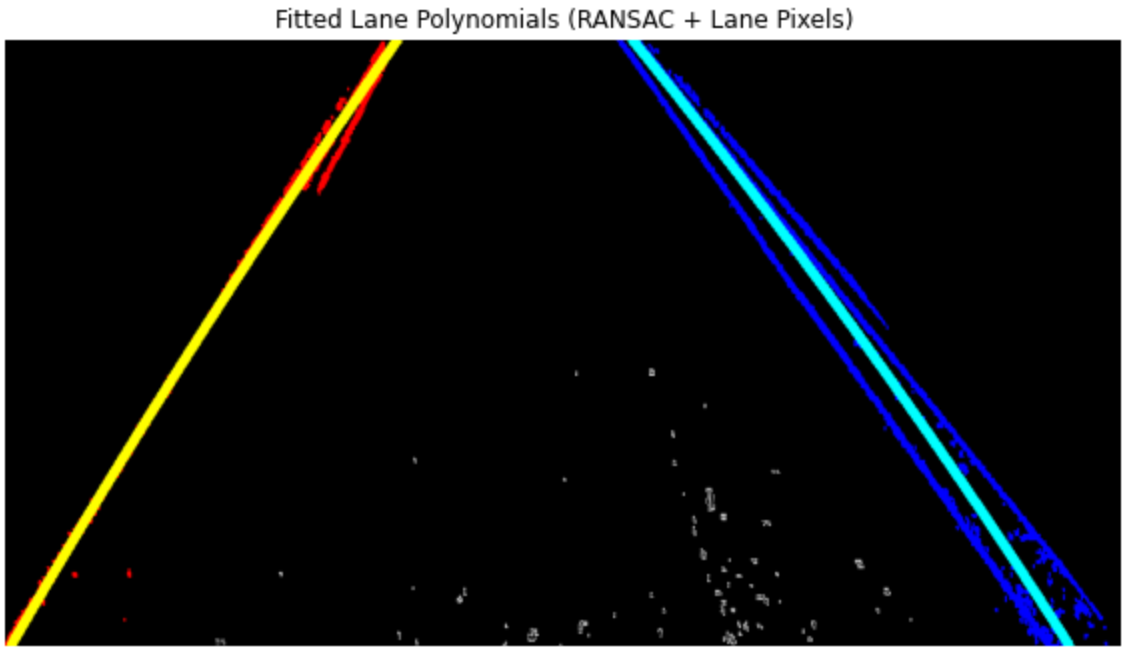
   Detected pixels are fitted with second-order polynomials:

   x=ay2+by+cx = a y^2 + b y + cx=ay2+by+c

   using least-squares regression.
   A confidence score is assigned based on the ratio of detected lane pixels to a minimum threshold:

   conf=min(1.0,NpixelsNmin)\text{conf} = \min\left(1.0, \frac{N_{\text{pixels}}}{N_{\text{min}}}\right)conf=min(1.0,NminNpixels)

where Nmin≈500N_{\text{min}} \approx 500Nmin≈500.
High-confidence fits produce stable curvature estimation, while low-confidence detections trigger dashed visualization for transparency.



6. **Lane Visualization and Overlay**

The detected lane boundaries are visualized using the function `draw_lane()`.

- **Polynomial fits** are drawn for both lanes.
- The **lane area** between them is filled with green to illustrate the drivable region.
- The visualization is inverse-warped back to the original camera perspective using the inverse matrix `Minv`.
  Finally, the overlay is merged with the original frame using weighted addition to

produce a clear, interpretable lane visualization.

7. Full Video Processing Pipeline

The complete process—thresholding, ROI masking, perspective warp, sliding window fitting, lane visualization, and offset computation—is encapsulated in a unified function `process_video()`.

This function processes each frame of a video sequentially, producing an annotated video stream showing lane lines, filled regions, and detection confidence in real time.

Calibration and Parameters:

Empirical calibration was performed based on visual alignment with the camera's viewpoint rather than explicit intrinsic/extrinsic calibration. Key parameters used:

These values were optimized through visual inspection to balance sensitivity and robustness.

Results and Evaluation:

The pipeline successfully detects and tracks lanes across consecutive frames, providing visually consistent overlays and accurate curvature estimation. The filled green lane area aligns closely with true road markings in most cases.

Successful cases:

- Well-marked lanes under daylight.
- Mild curvature and low traffic occlusion.

Failure cases:

- Overexposure and shadows.
- Missing lane markings or non-asphalt surfaces.
- High curvature near image edges.