Final Report: A Lane Departure Warning System using Canny Edge Detection and Hough Transform

Course: CSE730: Computer Vision and Image Understanding
Submitted By:
Name: Ayush Bajpai
Roll Number: cse22030
Registration Number: 884

---

1. Introduction

A Lane Departure Warning System (LDWS) is a critical component of modern Advanced Driver-Assistance Systems (ADAS). Its primary function is to enhance driver safety by monitoring the vehicle's position relative to road lane markings and issuing a warning if the vehicle begins to drift out of its lane unintentionally. Such systems are fundamental for preventing road accidents caused by driver inattention, distraction, or fatigue.

This project implements a complete, adaptive LDWS pipeline using a combination of classical and advanced computer vision techniques. The system's core is built upon the Canny edge detector for identifying potential lane boundaries and the Hough Transform for parameterizing straight lines from these edges.

To achieve superior robustness, this implementation incorporates several advanced enhancements. A sophisticated **HSV Shadow Mitigation** technique is included to handle challenging lighting. A **Genetic Algorithm** is used to dynamically optimize the Canny edge detector's thresholds, allowing the system to adapt in real-time. Finally, a **Kalman Filter** is employed for temporal tracking of the detected lanes, providing smooth, stable estimations that are resilient to noise and temporary occlusions.

The implemented pipeline systematically processes video frames through several stages: preprocessing with shadow mitigation, adaptive Canny edge detection, adaptive ROI selection, Hough line detection, line validation, Kalman filter tracking, and a final lane departure analysis.

2. Methodology

The lane detection process is a multi-stage pipeline that transforms a raw input image into a clear and actionable lane departure warning. Each step is designed to refine the data and extract the necessary features for the final analysis.

2.1. Image Preprocessing & Enhancement

The initial step focuses on preparing the image to make lane features more prominent. The input image is converted to grayscale, and a Gaussian blur with a 5x5 kernel is applied to smooth the image and reduce noise. To improve robustness, a color-based enhancement technique is also employed. The image is converted to the HSV (Hue, Saturation, Value) color space to create masks that specifically isolate white and yellow colors. These masks are combined and used to selectively brighten the corresponding areas in the grayscale image, making the lanes stand out more clearly.

2.2. Shadow Mitigation using HSV Color Space

Hard-edged shadows from objects like buildings or overpasses pose a significant challenge, as their boundaries can be mistaken for lane lines. This system includes a dedicated shadow mitigation stage to address this.

The methodology is based on the principle that shadows primarily decrease a pixel's brightness (Value) while having a lesser effect on its color (Hue and Saturation). The process involves two steps:

1. Shadow Detection with NSVDI: First, shadows are detected using the Normalized Saturation-Value Difference Index (NSVDI). For each pixel (i,j) in the HSV image, this index is calculated as:
   $$\mathrm{NSVDI}(i,j) = \frac{S(i,j) - V(i,j)}{S(i,j) + V(i,j)}$$
   In shadow regions, Saturation (S) tends to be higher while Value (V) is lower, resulting in a high positive NSVDI value. A binary shadow mask is created by thresholding this index.
2. **Shadow Compensation with LCC:** The detected shadow regions are then "relit" using **Linear Correlation Correction (LCC)**. This method adjusts the intensity distribution of the shadow pixels to match that of adjacent, non-shadowed regions. It does this by scaling the V-channel values in the shadow mask based on the mean (μ) and standard deviation (σ) of the shadow (s) and non-shadow (ns) areas, effectively equalizing the illumination.

While this function is fully implemented in the codebase, it is an optional step in the final pipeline and can be activated for scenarios with particularly challenging shadow conditions.

2.3. Canny Edge Detection

The Canny edge detector identifies sharp changes in intensity, corresponding to lane

boundaries. The algorithm involves four main steps: Gaussian smoothing, gradient calculation, non-maximum suppression, and hysteresis thresholding. The gradient magnitude G is calculated as G = $\sqrt{G_x^2 + G_y^2}$. The final step uses a low and high threshold to classify edge pixels.

2.3.1. Adaptive Threshold Optimization via Genetic Algorithm

A key challenge with the Canny detector is that fixed thresholds fail under varying lighting conditions. This system addresses this by using a Genetic Algorithm (GA) to automatically find optimal thresholds. The GA is an optimization technique inspired by natural selection that runs periodically (e.g., every 120 frames) to adapt to changing scenery.

The GA works as follows:

- **Individuals & Population:** An "individual" is a pair of (low, high) Canny thresholds. A "population" is a collection of these individuals.
- **Fitness Function:** The quality of each individual is evaluated by a custom fitness function. It applies the threshold pair to the current frame and measures the "lane-ness" of the resulting edges based on three criteria:
    1. **Edge Density:** Rewards a density of edge pixels that is neither too sparse nor too cluttered.
    2. **Connectivity:** Prefers fewer, longer connected edge components, as these are more characteristic of lane lines.
    3. **Line Quality:** Uses a quick Hough Transform to evaluate how well the edges form straight lines.
- **Evolution:** The GA then proceeds through an evolutionary cycle of selection (choosing the fittest individuals), crossover (combining pairs to create offspring), and mutation (introducing small random changes). This process is repeated for several generations, converging on a set of thresholds that maximizes the fitness score for the current conditions.

The core of the GA is its fitness function, which evaluates the quality of a given threshold pair. The total fitness is a weighted sum of three distinct metrics designed to quantify how "lane-like" the resulting edges are:

$$\text{Fitness} = 0.4 \cdot F_{\text{density}} + 0.4 \cdot F_{\text{connectivity}} + 0.2 \cdot F_{\text{quality}}$$

The components are calculated as follows:

- $F_{density}$ (Edge Density Score): This metric rewards an optimal edge density, penalizing images that are too sparse or too cluttered. Given an optimal target density of 5% (d_{opt} = 0.05), the score is calculated based on a penalty for deviating from this ideal:

$$\text{density} = \frac{\text{edge}_\text{pixels}}{\text{total}_\text{pixels}}$$

$$\text{penalty} = \frac{|\text{density} - d_{\text{opt}}|}{d_{\text{opt}}}$$

$$F_{\text{density}} = \max(0, 100 \cdot (1 - \text{penalty}))$$

- $F_{connectivity}$ (Connectivity Score): This metric favors fewer, larger connected components, which is characteristic of continuous lane lines. It is calculated from the average size of the detected components:

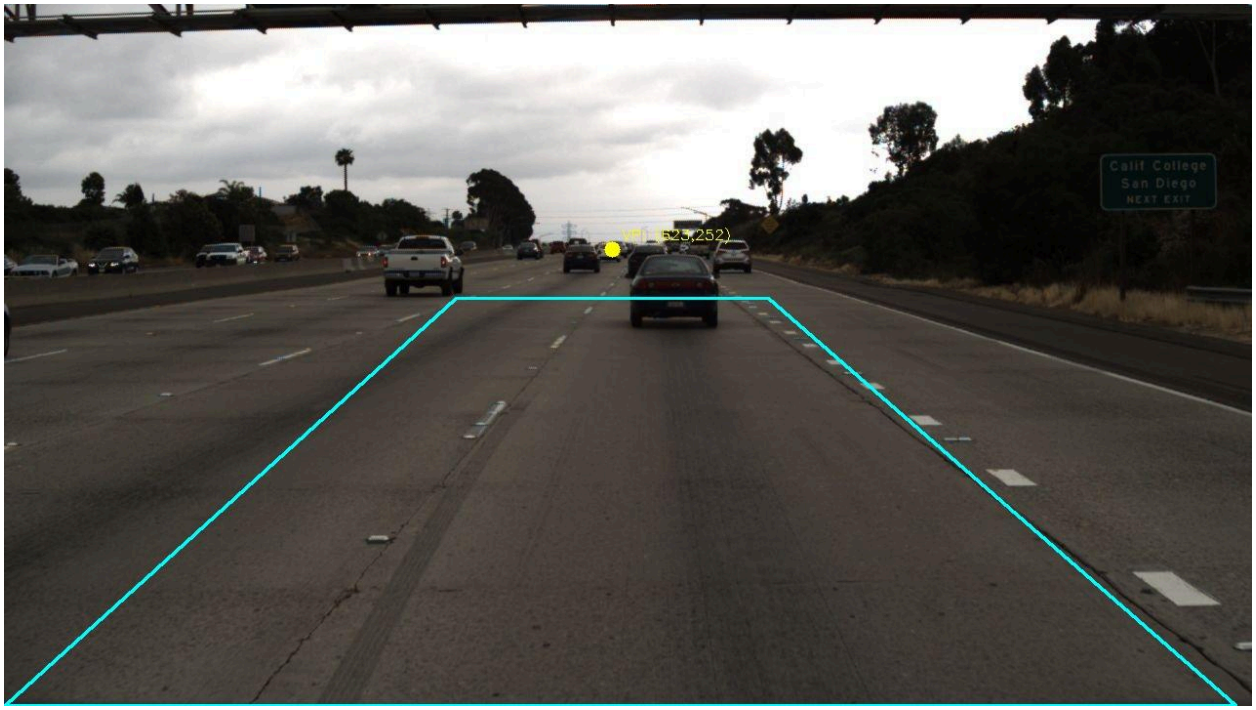
  $$F_{\text{connectivity}} = \min\left(\frac{\text{avg}_\text{component}_\text{size}}{50.0}, 100.0\right)$$

- $F_{quality}$ (Line Quality Score): This metric uses a preliminary Hough Transform to assess how well the edges form straight lines. It rewards a moderate number of lines, penalizing both a lack of lines and an excess of lines (indicative of noise). The score is a piecewise function of the number of detected lines, n:

  $$F_{\text{quality}} = \begin{cases} 0 & \text{if } n = 0 \\ \min(5n, 100) & \text{if } 1 \leq n \leq 20 \\ \max(0, 100 - 2(n - 20)) & \text{if } n > 20 \end{cases}$$

2.4. Region of Interest (ROI) Selection

To focus only on the road, this system uses an *adaptive ROI* calculated from the scene's geometry. This is achieved by estimating the **vanishing point**—the point on the horizon where parallel lane lines appear to converge. The vanishing point is estimated by performing an initial Hough detection on the frame, calculating the intersection points of the resulting lines, and using DBSCAN clustering to find the densest group of intersections. A trapezoidal ROI is then defined with its top edge relative to this vanishing point, ensuring the ROI adapts to changes in the road's perspective.



2.4.1. Vanishing Point Estimation

The theoretical basis is **perspective projection**, where parallel lines in 3D world space (like lane markings) appear to converge at a single **vanishing point (VP)** in the 2D image.

The estimation is performed by finding the consensus intersection of line candidates detected by an initial Hough Transform. The calculation uses **homogeneous coordinates**:

1. A line in polar form (ρ,θ) is converted to a 3D homogeneous vector l:
   $l = [\cos(\theta), \sin(\theta), -\rho]^T$
2. The intersection point v of two distinct lines, l_1 and l_2, is found using their vector cross product:
   $v = l_1 \times l_2$
3. The resulting homogeneous point $v = [x', y', w']^T$ is converted back to 2D image coordinates (x,y):
   $(x, y) = (x'/w', y'/w')$

The **DBSCAN clustering algorithm** or **Median method** is applied to the set of intersection points, and the center of the largest cluster is taken as the final estimated vanishing point, $VP = (v_x, v_y)$

2.4.2. Adaptive Trapezoid Formulation

Once the vanishing point (v_x, v_y) is estimated, a dynamic trapezoidal ROI is constructed. The four vertices of the trapezoid are defined relative to the VP's position, ensuring the ROI adapts

to the vehicle's pitch (up/down movement).

2.4. Hough Line Transform

After isolating the edges within the adaptive ROI, the Hough Transform is used to identify lines by transforming points into a (ρ,θ) parameter space, where a line is represented by:
$\rho = x\cos(\theta) + y\sin(\theta)$
Peaks in the accumulator array of this space correspond to prominent lines. The Probabilistic Hough Transform (HoughLinesP) is used as it efficiently returns the endpoints of detected line segments.

2.5. Line Validation and Classification

The Hough Transform produces multiple line candidates that must be filtered. A confidence score is calculated for each line to rank its quality:

$$S(L) = w_1 \cdot A + w_2 \cdot \frac{\text{Length}(L)}{100} + w_3 \cdot G_{VP}(L) + w_4 \cdot \frac{C_{\text{color}}(L)}{255}$$

Where:

- A is the accumulator score (constant 1.0 in this implementation).
- Length(L) is the geometric length of the line segment.
- GVP(L) is a geometric score penalizing distance to the vanishing point:
  $G_{VP}(L) = \frac{1.0}{1.0 + (\text{distance}_{\text{to\_vp}}/50.0)}$
- Ccolor(L) is the average grayscale intensity of pixels along the line.

The weights are set to w1=0.3,w2=0.3,w3=0.2,w4=0.2.The highest-scoring lines that also satisfy slope and position constraints (negative slope for left lanes, positive for right) are selected.

2.6. Temporal Tracking with Kalman Filter

To ensure smooth and stable lane detection over time, a Kalman Filter is used. Per-frame detection can be jittery and fail on temporary occlusions. The Kalman filter addresses this by tracking the state of each lane line across frames. It operates in a recursive predict-update cycle.

The **State-Space Model** for tracking a lane line (defined by slope m and intercept c) is:

- **State Vector (x_k):** A 4D vector tracking the line's parameters and their velocities.
  $x_k = \begin{bmatrix} m \\ c \\ \dot{m} \\ \dot{c} \end{bmatrix}$

- **State Transition Matrix (F)**: Defines a constant velocity motion model, predicting the next state based on the current state and velocity.
  $$F = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
- **Measurement Vector (z_k):** The observed slope and intercept (mobs,cobs) from the current frame's validated Hough detection.
- **Measurement Matrix (H)**: Maps the 4D state space to the 2D measurement space, as only position parameters are directly observed.
  $$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

In the **Predict-Update Cycle**:

1. **Predict:** The filter predicts the lane's current position based on the previous state: $\hat{x}_{k|k-1} = F\hat{x}_{k-1|k-1}$.
2. **Update**: It then corrects this prediction using the current measurement. The updated state is a weighted average of the prediction and the measurement, where the weighting is determined by the Kalman Gain (K_k):
   $\hat{x}k|k = \hat{x}k|k-1 + K_k(z_k - H\hat{x}_{k|k-1})$

   This produces a final output that is smoother and more robust to noise or missed detections than single-frame analysis alone.

2.7. Lane Departure Analysis

With the stabilized lane lines, the system calculates the lane center at a fixed reference point on the image. It compares this to the vehicle's position (assumed to be the image center) to determine an offset. A departure ratio is calculated, and a corresponding warning is issued: **SAFE**, **CAUTION**, or **DANGER**.

1. **Lane Position Sampling:** The x-coordinates of the left and right lane lines are calculated at a fixed horizontal sample line in the image ($sample_y$).
2. Center Calculation: The lane center and vehicle center are determined:
   $$L_{\text{center}} = \frac{x_{\text{left}} + x_{\text{right}}}{2}$$
   $$V_{\text{center}} = \frac{\text{image}_{\text{w}}\text{idth}}{2}$$
3. Departure Ratio Calculation: The vehicle's offset from the lane center is calculated and normalized by half the lane width to produce a departure ratio:
   $$\text{offset} = V_{\text{center}} - L_{\text{center}}$$

$$\text{lane}_\text{w}\text{idth} = x_\text{right} - x_\text{left}$$

$$\text{Departure Ratio} = \frac{|\text{offset}|}{\text{lane}_\text{w}\text{idth}/2}$$

4. **Warning Generation:** A warning is issued based on this ratio:
    - **DANGER:** If Departure Ratio > 0.7 (70%)
    - **CAUTION:** If Departure Ratio > 0.4 (40%)
    - **SAFE:** Otherwise.

3. Implementation Details

The system was implemented in Python using **OpenCV** for computer vision tasks and **NumPy** for numerical operations. The logic is encapsulated within the EnhancedLaneDepartureWarning class, which includes the GeneticThresholdOptimizer for Canny tuning and Kalman filter objects for tracking. The system processes video files, overlaying detected lanes and status information onto the output frames.

4. Results and Discussion

The implemented system successfully detects and tracks lane markings. The use of a Genetic Algorithm for Canny thresholding makes the edge detection noticeably more reliable across different scenes in the test videos. The Kalman filter provides significant stability, preventing the rendered lanes from flickering and allowing the system to "remember" the lane position during brief moments when markings are obscured.

The final output provides clear visual feedback, with colored lines indicating the tracked lanes and a text-based warning level that communicates the current driving status.

Limitations:
Despite the enhancements, the system's effectiveness may be reduced in scenarios with:

- **Poor Weather:** Heavy rain, snow, or fog.
- **Faded Markings:** Worn-out or non-existent lane lines.
- **Complex Road Geometry:** Sharp turns or complex intersections.
- **Harsh Lighting:** Strong shadows or intense glare.

5. Conclusion

This project successfully demonstrates the development of an advanced Lane Departure Warning System. By augmenting a classical computer vision pipeline with a Genetic Algorithm for parameter optimization and a Kalman Filter for robust temporal tracking, the system achieves a high degree of adaptability and stability. This work provides a practical application of fundamental and advanced image processing concepts, showcasing a system that is significantly more robust than a basic implementation.

6. References

1. Duda, R. O., & Hart, P. E. (1972). Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Communications of the ACM, 15*(1), 11–15.
2. Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8*(6), 679–698.
3. Kálmán, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering, 82*(Series D), 35-45.
4. Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems*. MIT Press. (Reference for Genetic Algorithms).
5. Barnard, S. T. (1983). Interpreting perspective images. *Artificial intelligence, 21*(4), 435-462. (Reference for Vanishing Point Estimation).
6. Ma, L., et al. (2015). A robust method for shadow detection and compensation. *Journal of Visual Communication and Image Representation*, 28, 97-107. (Reference for NSVDI).