11/3/2025

Assignment 1

Computer Vision

This assignment focuses on applying computer vision techniques for two tasks:

1. **Cartoonifying an image** by smoothing colors and enhancing edges using filters
2. **Detecting road lanes** using edge detection and the Hough Transform.

Both tasks involve image processing techniques like noise reduction, edge detection, and feature extraction to transform or analyze images effectively.

IDs	**Team Members**
7818	Mohamed Hussein
7925	Fares Amin
7721	Alaa Elkhouly

Part 1:

1. How the Code Works (The theory behind it) :

a) Median Filter

i. <u>What It Does:</u>

The median filter is a non-linear filter used primarily for noise reduction. It works exceptionally well at removing "salt and pepper" noise (random bright and dark spots) without blurring the edges as much as linear filters (like Gaussian filters) might.

i. <u>How It Works:</u>

The filter works by sliding a window (kernel) over the image.
For each pixel, it considers the intensities of the pixels in the window.
It then replaces the center pixel's value with the median value of those intensities.
Because the median is less sensitive to outliers than the mean, the filter can remove noise while preserving sharp boundaries.

b) Laplacian Filter

i. <u>What It Does:</u>

The Laplacian filter is a second-order derivative operator used for edge detection. It highlights areas of rapid intensity change in an image, which typically correspond to edges.

ii. <u>How It Works:</u>

The Laplacian operator computes the second derivative of the image intensity.
This operation emphasizes regions where the intensity changes rapidly (i.e., edges).
However, because second derivatives are highly sensitive to noise, applying the Laplacian filter directly to a noisy image could result in many false edges—hence the prior use of the median filter.
Once the raw edges are computed, a binary threshold is applied to create a sketch-like (black-and-white) effect by converting the grayscale edges into a crisp mask where pixels are either "edge" (white) or "non-edge" (black).

c) Bilateral Filter

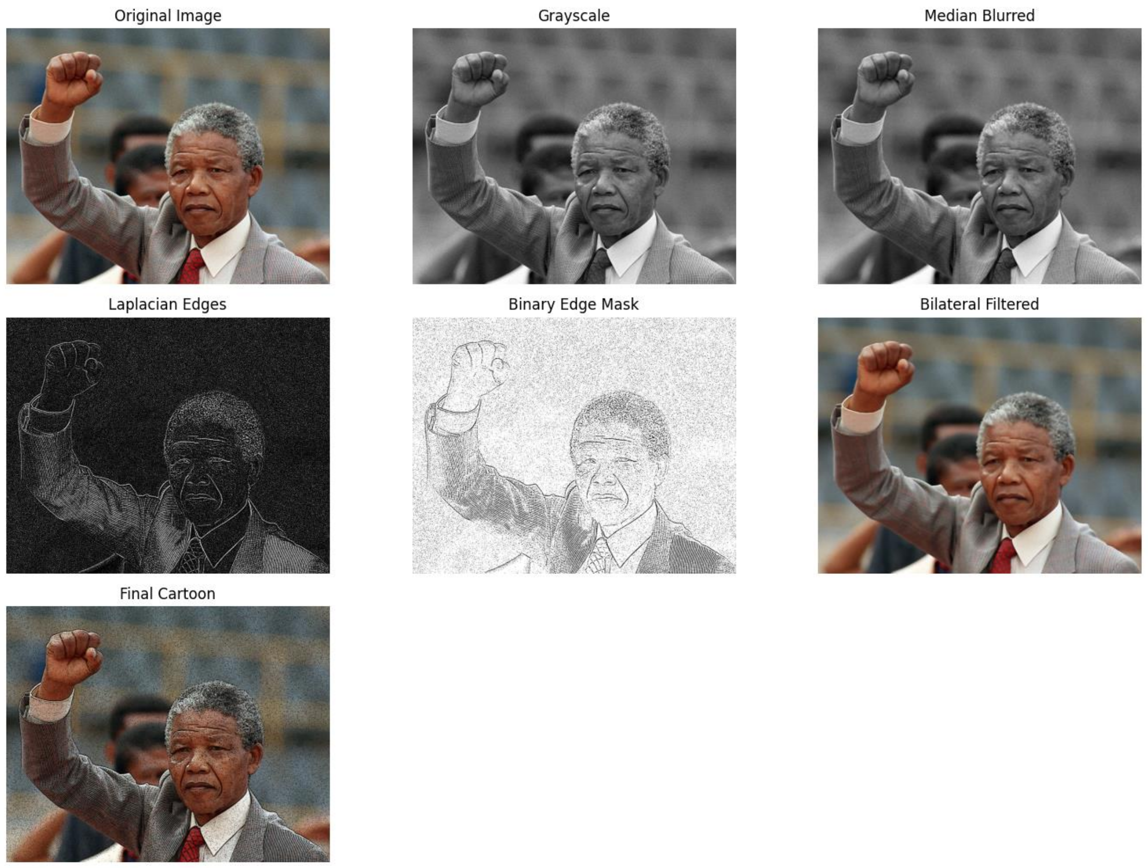
i. <u>What It Does:</u>

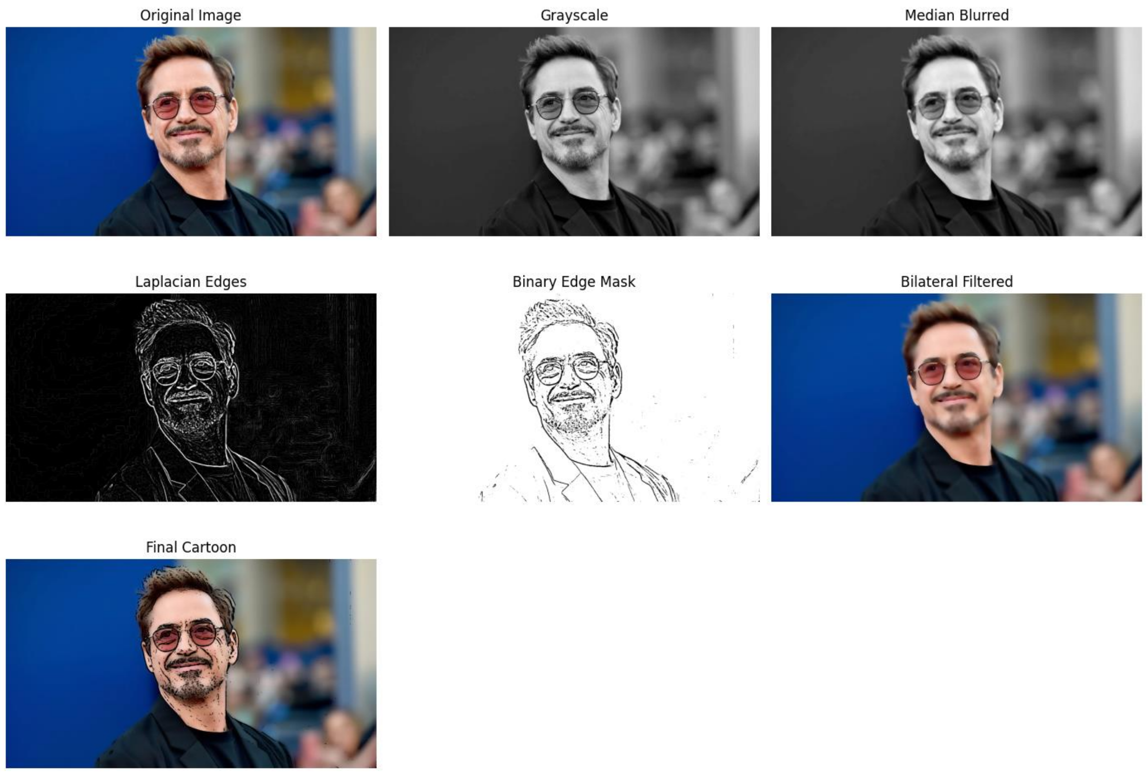
The bilateral filter is an edge-preserving smoothing filter. It reduces image noise and detail in flat regions while keeping edges sharp.

ii. <u>How It Works:</u>

Unlike traditional filters that consider only the spatial proximity of pixels, the bilateral filter combines both spatial closeness and intensity similarity.
It applies a weighted average where weights are determined by:
Domain (Spatial) Weight: A Gaussian function of the distance between the center pixel and the neighboring pixels.
Range (Intensity) Weight: A Gaussian function of the difference in intensity between the center pixel and its neighbors.

2. Results:


Original Image
Grayscale
Median Blurred
Laplacian Edges
Binary Edge Mask
Bilateral Filtered
Final Cartoon


Original Image
Grayscale
Median Blurred
Laplacian Edges
Binary Edge Mask
Bilateral Filtered
Final Cartoon

Part 2:

1. How the Code Works (The theory behind it) :

a) Edge Detection – Canny Edge Detector

The Canny Edge Detector is used for detecting edges in an image. By adjusting canny_threshold1 and canny_threshold2, you can control the sensitivity of edge detection. It identifies potential lane edges by highlighting strong intensity changes. The process involves several steps:

- **Grayscale Conversion**: The image is converted to grayscale since color information is unnecessary for edge detection.
- **Gaussian Blurring**: Applied to smooth the image and reduce noise. Edge detection algorithms are sensitive to noise, so this step ensures better results.
- **Gradient Calculation**: The intensity gradient of the image is computed using Sobel filters to identify areas where the intensity changes sharply (potential edges).
- **Non-Maximum Suppression**: This step removes pixels that are not the local maxima of the gradient, keeping only the strongest edges.
- **Double Thresholding**: Two thresholds are used (low and high). Strong edges (above the high threshold) are kept, while weak edges (between the low and high thresholds) are considered only if they are connected to strong edges.
- **Edge Tracking by Hysteresis**: Weak edges that are connected to strong edges are preserved, while isolated weak edges are discarded.

b) Region of Interest (ROI) Masking

After edge detection, the image contains edges from various objects (not just the road lanes). To focus only on relevant edges (the lanes), we define a **region of interest (ROI)**:

- The **ROI is a polygon** that selects only the part of the image where lanes are expected (e.g., the bottom half of the image forming a trapezoidal or triangular shape).
- A **binary mask** is created where pixels inside the polygon are set to white (255), while the rest are set to black (0).
- Applying bitwise_and between the edge-detected image and the mask ensures only edges within the ROI are retained.
- The **Region of Interest Masking** removes unnecessary edges and focuses on the road area.

c) Line Detection – Hough Transform

The **Hough Transform** is used to detect straight lines from the edge-detected image (finds and draws the most likely lane lines based on detected edges). This algorithm works by transforming points in Cartesian coordinates into **Hough space**, where a line is represented by parameters:

$$\rho = x\cos\theta + y\sin\theta$$

Where:

- ρ is the perpendicular distance from the origin to the line.
- θ is the angle of the line with respect to the x-axis.

Steps of the Hough Transform:

1. **Vote Accumulation**: Each edge pixel in the image votes for possible lines that pass through it. These votes are stored in an accumulator array.
2. **Finding the Most Voted Lines**: The highest peaks in the accumulator array correspond to the most likely lane lines.
3. **Drawing the Detected Lines**: Once the line segments are identified, they are drawn on the original image.

Parameters used in cv2.HoughLinesP:

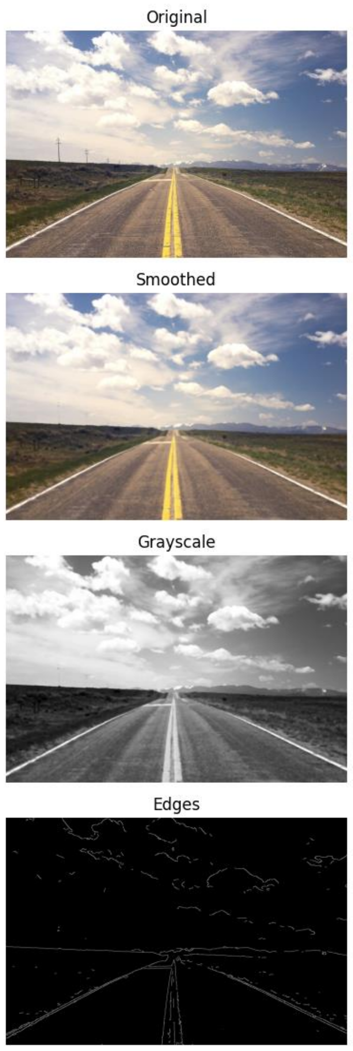
- rho: The distance resolution of the accumulator (in pixels).
- theta: The angle resolution (in radians).
- hough_threshold: The minimum number of votes required to consider a line.
- minLineLength: The shortest line to be detected.
- maxLineGap: The maximum gap between line segments that can be connected.

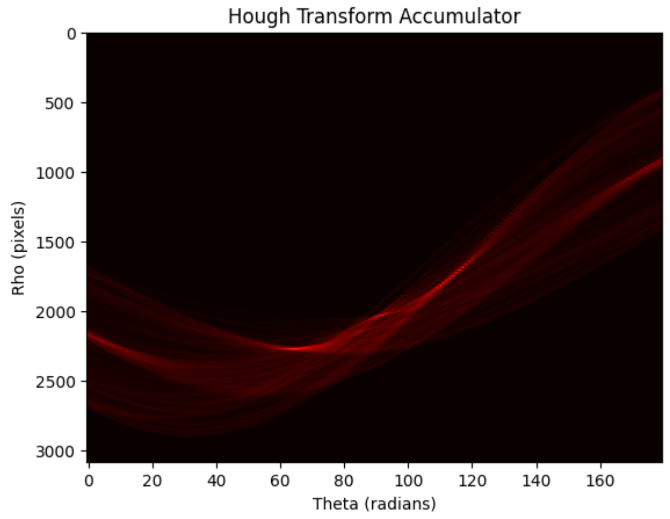
d) Overlaying the Lane Lines

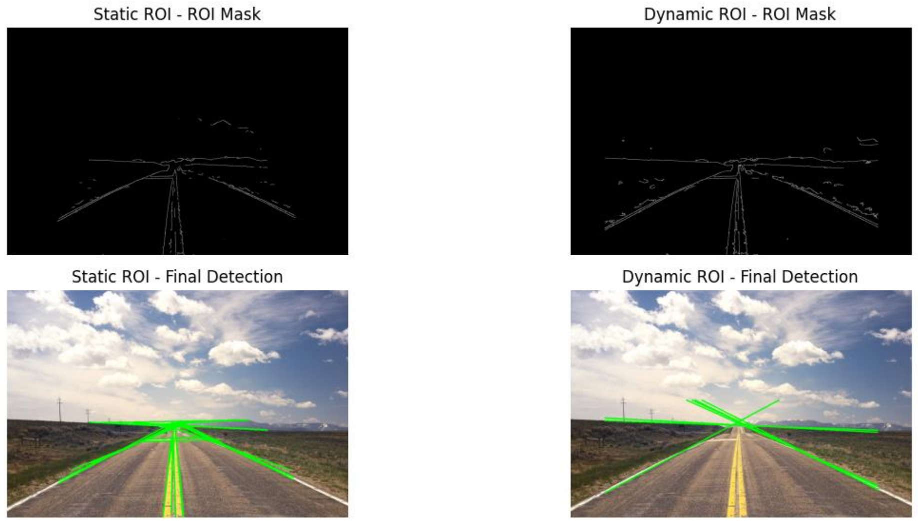
Once the lane lines are detected, they are **drawn on the original image** using cv2.line(). The detected lines are colored in green and overlaid on the original road image. The result is an image with detected lanes overlaid on the original road.

2. Results:

road10.jpg



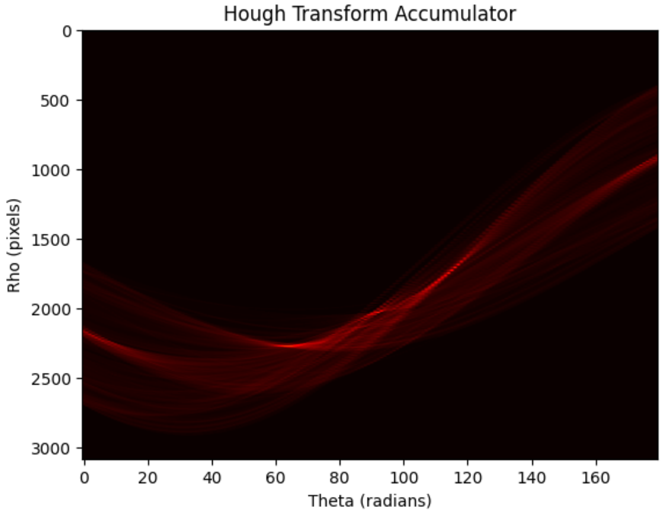




road3.jpg

Original

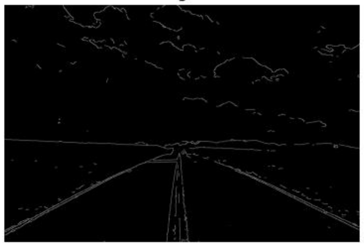
Hough Transform Accumulator



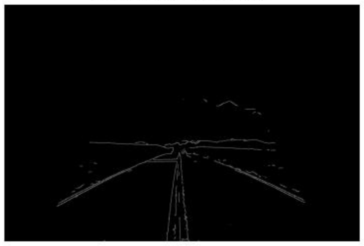
Smoothed

Grayscale

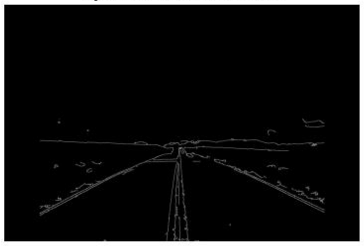
Edges



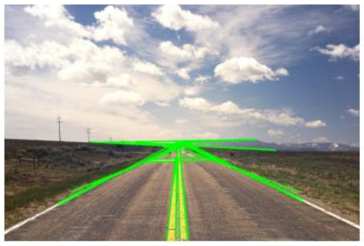
Static ROI - ROI Mask



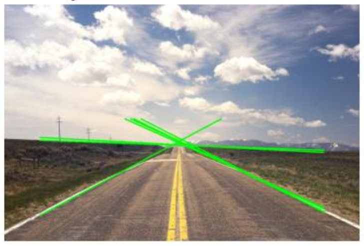
Dynamic ROI - ROI Mask

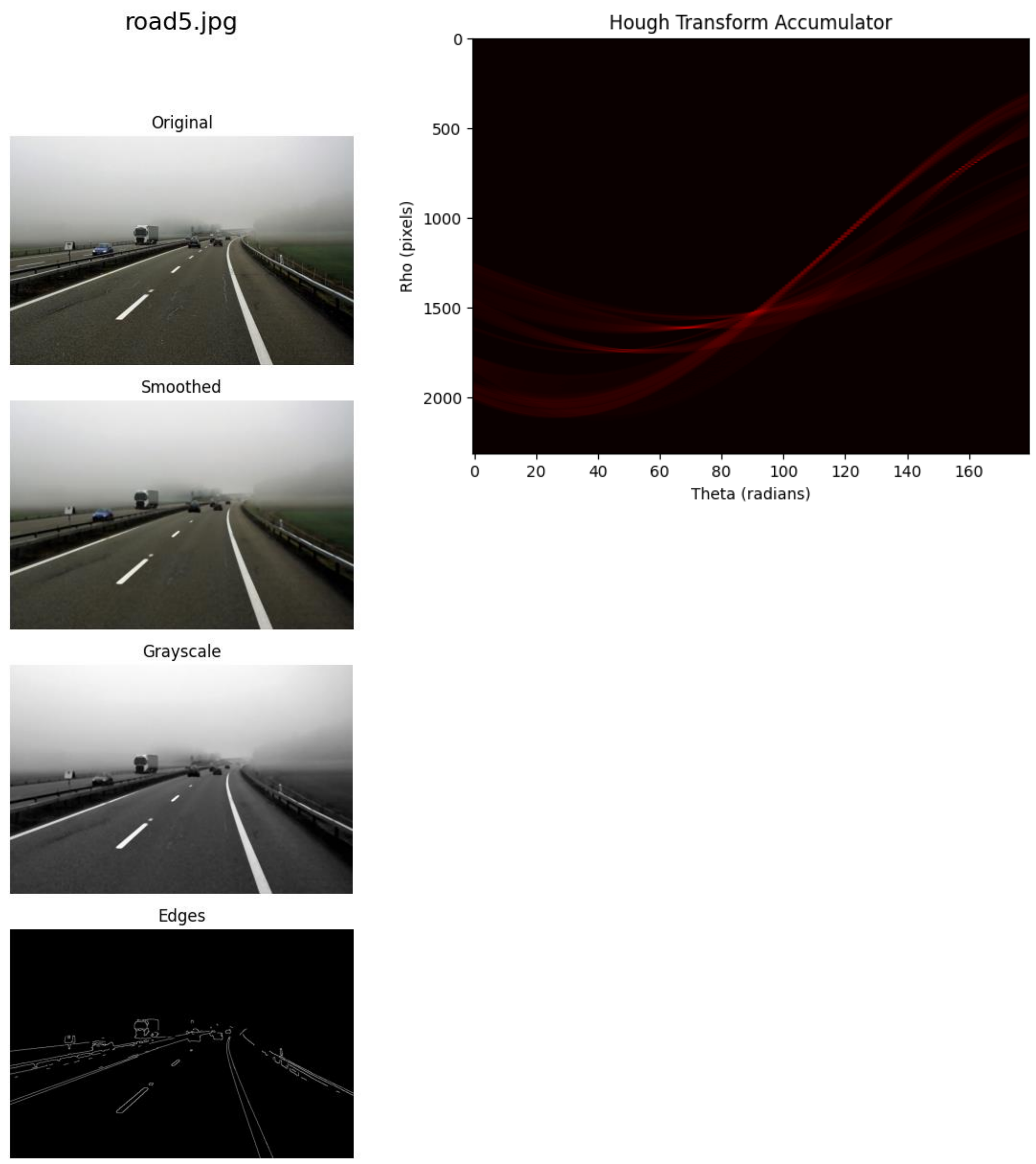


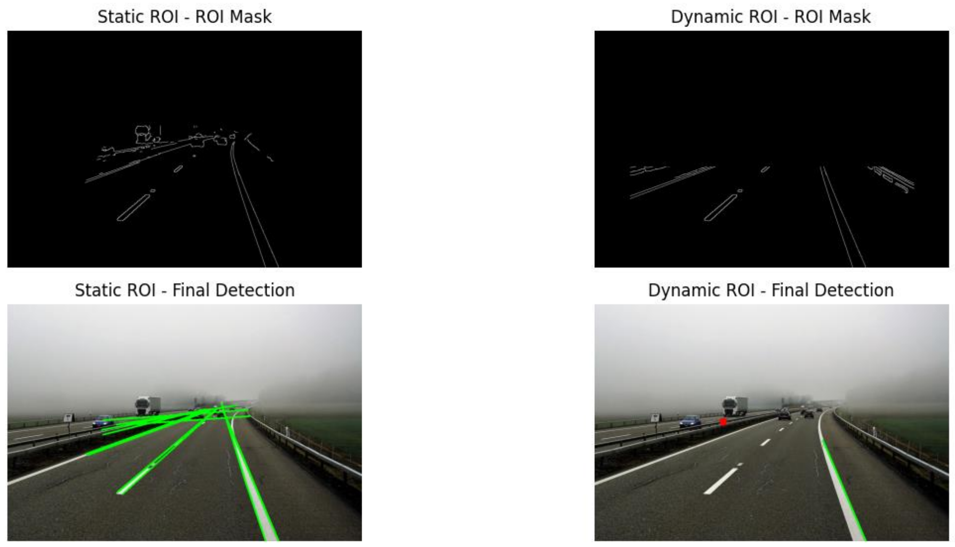
Static ROI - Final Detection



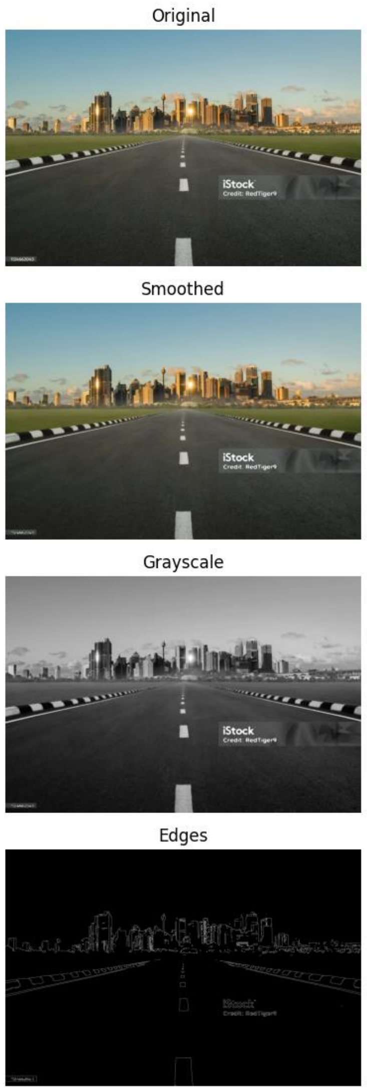
Dynamic ROI - Final Detection

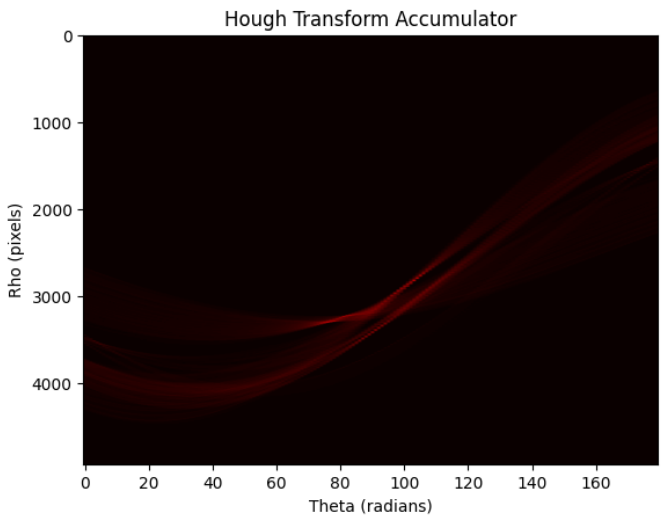


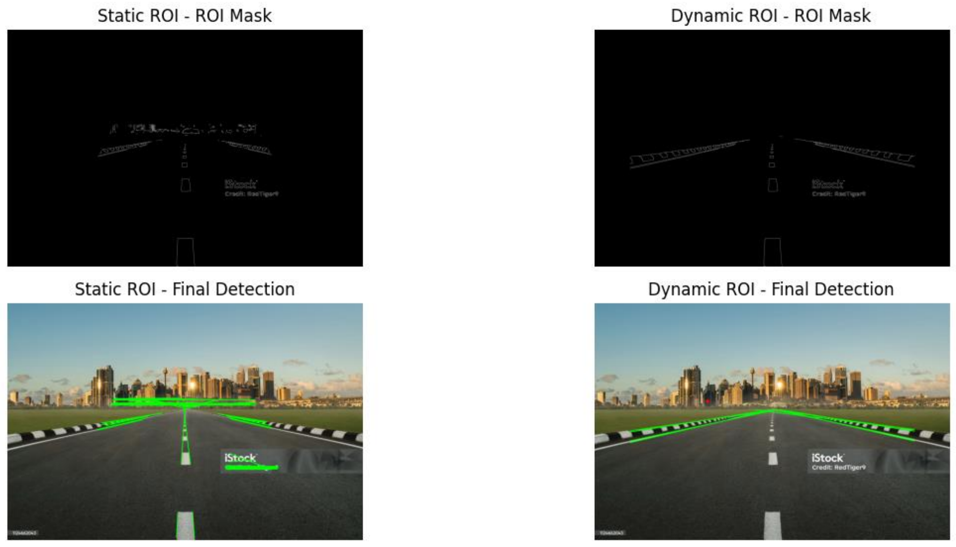

road5.jpg
Original
Smoothed
Grayscale
Edges
Hough Transform Accumulator
Rho (pixels)
0
500
1000
1500
2000
Theta (radians)
0
20
40
60
80
100
120
140
160


Static ROI - ROI Mask
Dynamic ROI - ROI Mask
Static ROI - Final Detection
Dynamic ROI - Final Detection

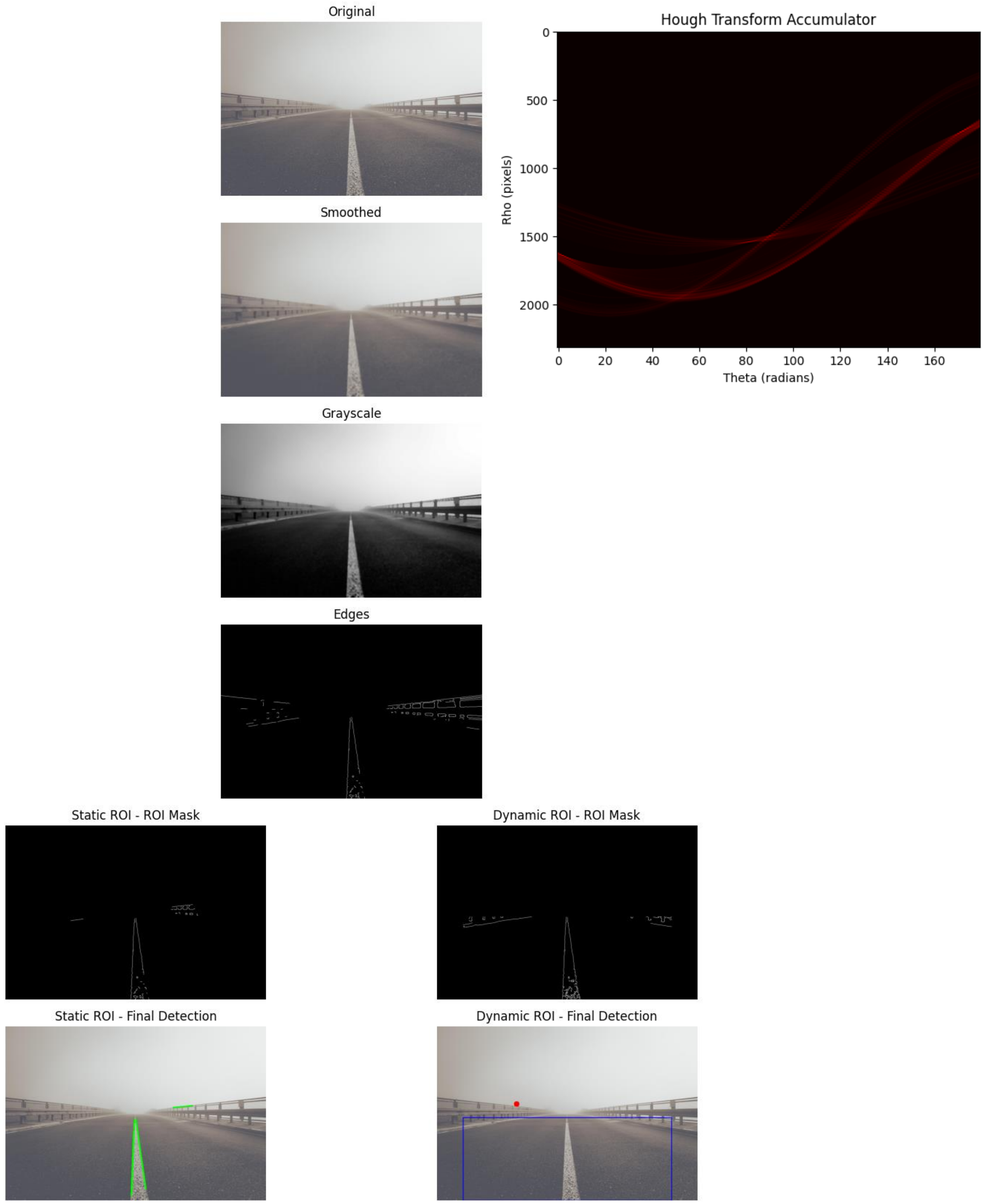
road6.jpg

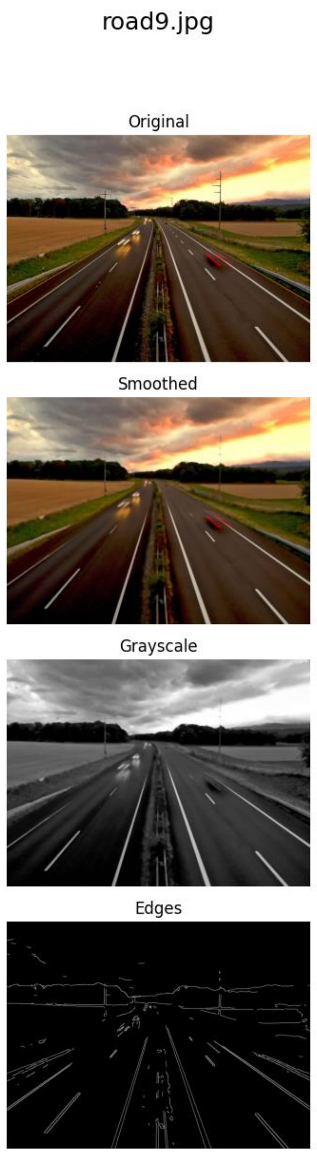

Original
Smoothed
Grayscale
Edges
iStock
Credit: RedTiger9

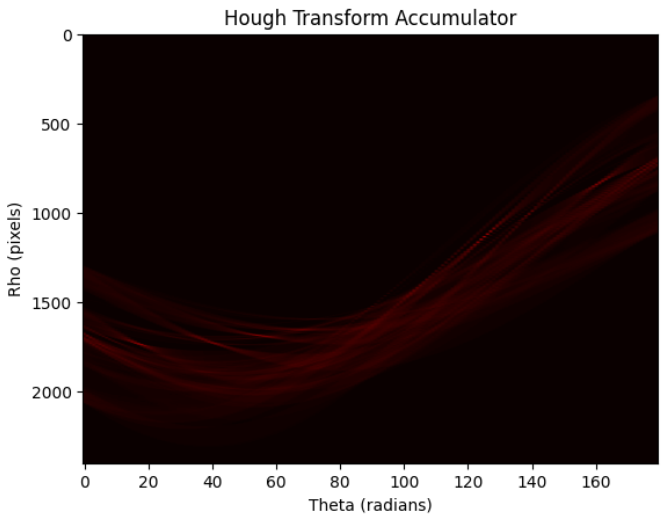

Hough Transform Accumulator
Rho (pixels)
Theta (radians)
0
1000
2000
3000
4000
0
20
40
60
80
100
120
140
160

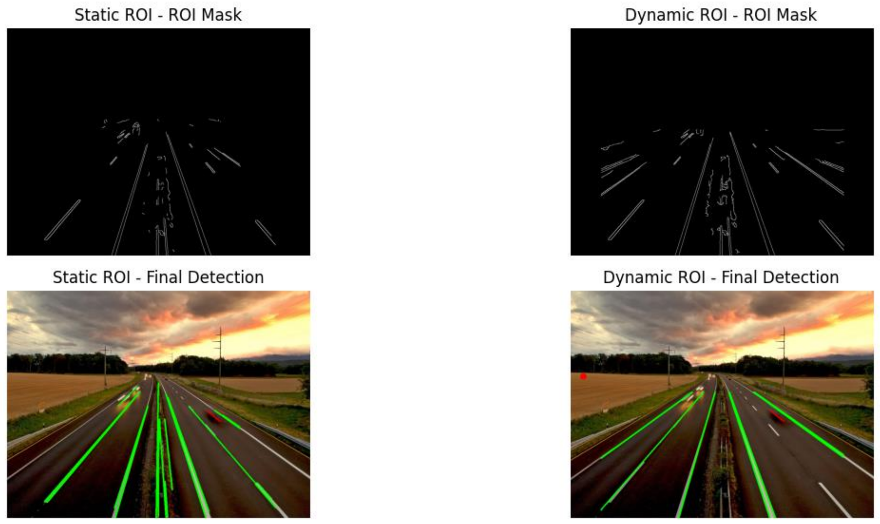

Static ROI - ROI Mask
Dynamic ROI - ROI Mask
Static ROI - Final Detection
Dynamic ROI - Final Detection
iStock
Credit: RedTiger9

road8.jpg


Original
Hough Transform Accumulator
Rho (pixels)
Theta (radians)
0
500
1000
1500
2000
0
20
40
60
80
100
120
140
160
Smoothed
Grayscale
Edges
Static ROI - ROI Mask
Dynamic ROI - ROI Mask
Static ROI - Final Detection
Dynamic ROI - Final Detection


road9.jpg
Original
Smoothed
Grayscale
Edges


Hough Transform Accumulator
Rho (pixels)
0
500
1000
1500
2000
Theta (radians)
0
20
40
60
80
100
120
140
160


Static ROI - ROI Mask
Dynamic ROI - ROI Mask
Static ROI - Final Detection
Dynamic ROI - Final Detection