

GROUP 17

LANE DETECTION FOR AUTONOMOUS VEHICLES

Presented by

D PRANEETH - cb.sc.u4aie24109

MEERA S - CB.SC.U4AIE24133

MITHUL AKSHAY - CB.SC.U4AIE24135

RAAMAN NAMPUTHIRI - CB.SC.U4AIE24148

INTRODUCTION

The rapid growth of the automotive industry has led to an increasing demand for intelligent systems that enhance road safety and driving comfort.

Lane detection and object recognition systems are critical components of **Advanced Driver Assistance Systems (ADAS)**.

They provide real-time feedback to drivers, helping them maintain proper lane discipline and avoid collisions by identifying potential obstacles like cars, pedestrians, or other road hazards.

This project focuses on integrating lane detection and object recognition using computer vision techniques to improve road safety and driving efficiency.

By analyzing video frames from a camera feed, **the system detects lane markings and identifies objects** such as vehicles, providing real-time instructions or warnings to the driver.

OBJECTIVES

Lane Detection:

- Develop an efficient algorithm to detect lane markings on road .

Object Recognition:

- Implement object detection using YOLOv3 to identify and classify road obstacles such as cars, bicycles, and pedestrians with high accuracy in different light conditions.

Traffic Analysis:

- Identify traffic situations, such as the presence of cars in close proximity, and generate appropriate warnings like "Go Slow, Traffic Ahead."

METHODOLOGY

- 1. Load the required files (YOLO weights, configuration, and class (coco names)) and ensure the resources are available for the program.**
- 2. Load the class names of objects of YOLOv3 from coco names file (e.g., "person", "car").**
- 3. Load and verify the input video for processing.**
 1. Opens the input video file for frame-by-frame processing.
 2. Reads each and every frame from the video.
If reading fails (end of video or error), the loop exits.

4. Noise Reduction

Since edge detection is easily influenced by noise in the image we need to preprocess the image with noise reduction using gaussian filter.

Process:

1. Gaussian Blur applies a Gaussian filter to the image, which uses a kernel of weights to smooth the image.
2. Each pixel's value is replaced by the weighted average of its neighbours, with closer pixels contributing more to the average.
3. This operation helps remove random noise that could interfere with edge or lane detection.

5. Image Preprocessing

we will be converting RGB image to HSV since HSV (Hue, Saturation, Value) is a colour model that separates image intensity (Value) from colour information (Hue and Saturation).

It is particularly effective for detecting colours like yellow lanes, regardless of lighting.

Process:

1. **Conversion:** Convert the image from BGR (Blue, Green, Red) to HSV.

- **Hue:** Represents the type of colour (e.g., yellow, blue).
- **Saturation:** Intensity or purity of the colour.
- **Value:** Brightness of the colour.

2. **Thresholding:**

- A specific range of HSV values (lower and upper bounds) is defined for the desired colour (e.g., yellow).
- Pixels within this range are set to white in a binary mask (where yellow lanes are highlighted), and others are set to black

6. Canny edge detection

Canny Edge Detection is used to detect edges (sudden changes in intensity) in an image. This step is crucial for identifying lane boundaries after color filtering.

Process:

1. Gradient calculation:

Here we find gradients in G_x and G_y directions . By finding gradients we can know the major change in intensities. we also here find the edge direction by using formula $\theta = \tan^{-1}(G_x/G_y)$.

2. Non maximum suppression:

The output should have thin edges and suppresses non-edge pixels to ensure that only the most prominent edges are preserved. The algorithm goes through all the points and finds out the pixels with the maximum value of gradient in edge direction (found in previous method).

3.Double thresholding:

For this step we need two threshold values (min values and max values). Any edges with intensity gradient more than max value are sure to be edges and those below min value are non edges.

4.Hysteresis Thresholding:

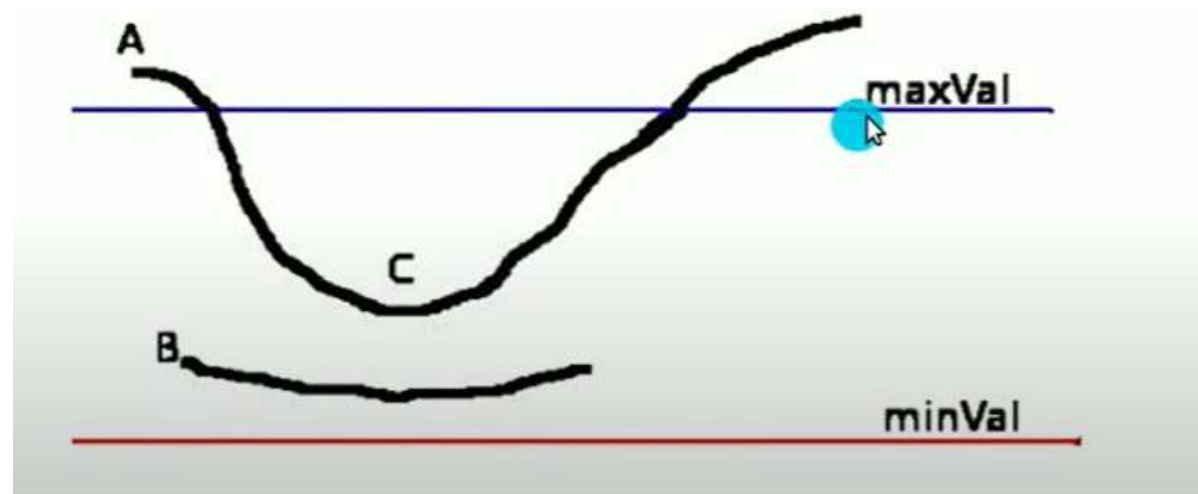
If they are connected to sure edge pixels they are considered to be part of edges otherwise they are discarded .



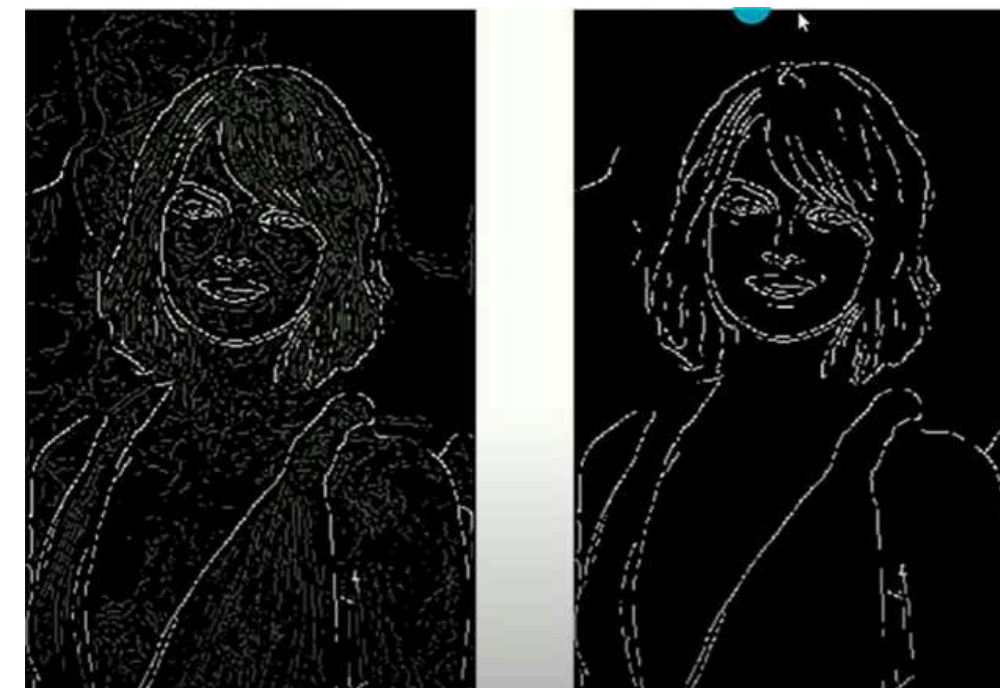
Non Maximum Suppression



Double Thresholding



Hysteresis



After Hysteresis



Figure: Input Image to Canny's Process



Figure: Output Image after Canny's Process

6. Hough Line Transform

The Hough Transform is used to detect the lane markings in the video.

Before applying the Hough Transform, the code first detects edges in the image using the Canny edge detector.

By detecting the lines that correspond to the lane markings, the code can determine if the vehicle is staying within the lanes

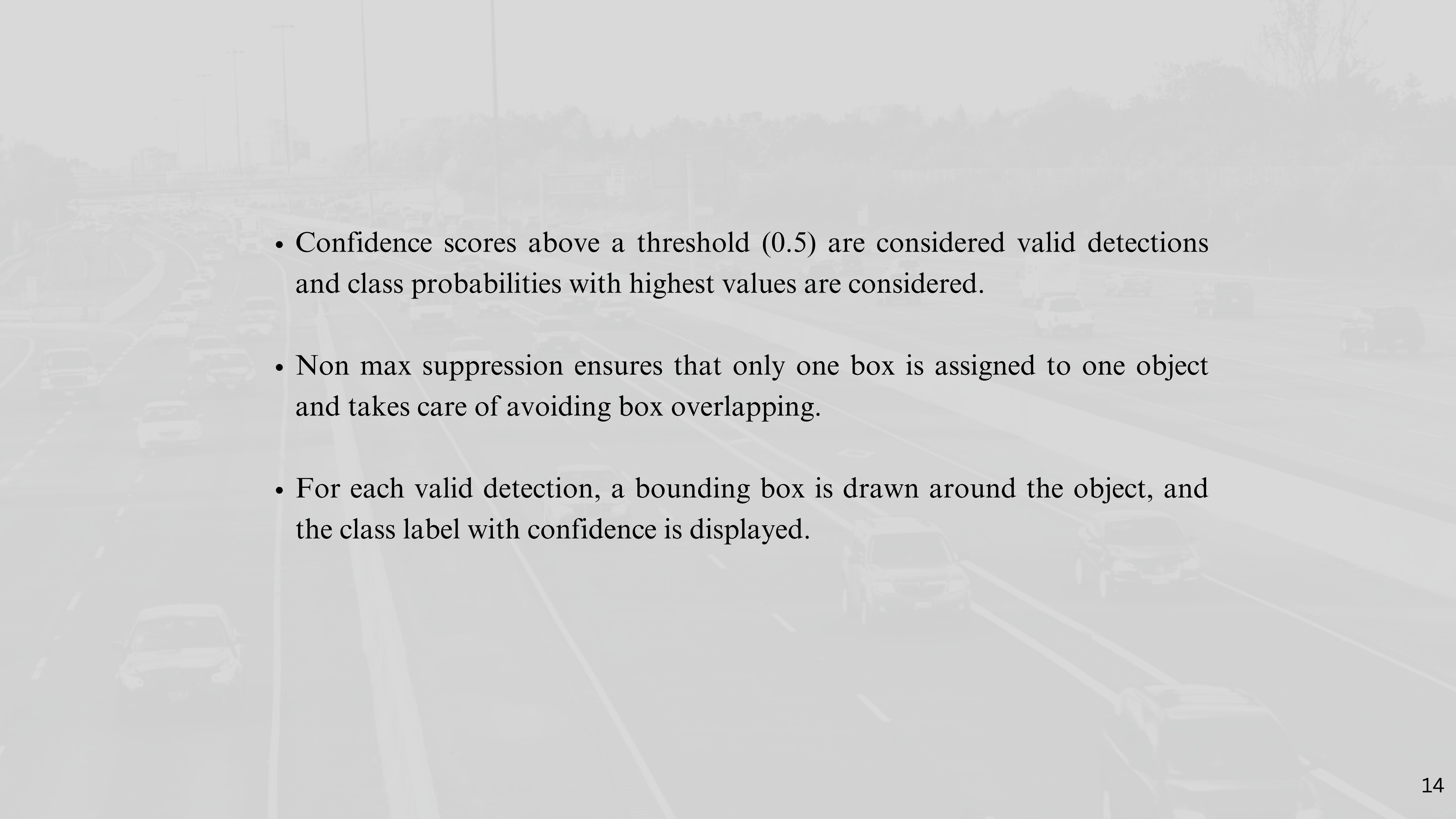
It can detect lines even if they are partially visible, noisy, or have gaps.



Hough Line Transform

7. Object detection:

- The frame is resized to a fixed size (416x416) and normalized to fit the YOLOv3 input requirements
- A blob(the image after resizing) is created to pass through yolo v3 network to make predictions
- Output includes bounding box coordinates(location and size of the detected object), confidence scores(certainty that a bounding box contains an object it is between 0 and 1.), and class probabilities(probability of detected object belonging to each of the predefined class it is also between 0 and 1).

- 
- Confidence scores above a threshold (0.5) are considered valid detections and class probabilities with highest values are considered.
 - Non max suppression ensures that only one box is assigned to one object and takes care of avoiding box overlapping.
 - For each valid detection, a bounding box is drawn around the object, and the class label with confidence is displayed.

OVERVIEW OF METHODOLOGY

1. INPUT VIDEO PROCESSING

Load the video using OpenCV and verify successful loading. Sequentially process each frame.

2. PREPROCESSING WITH GAUSSIAN BLUR

Apply gaussian Blur to reduce noise and smooth the image.

3. COLOR FILTERING USING HSV

Convert the frame into HSV color space, isolate the lanes using defined HSV ranges.

4. EDGE DETECTION WITH CANNY

Highlight significant edges using canny Edge Selection with appropriate thresholds.

5. LINE DETECTION USING HOUGH TRANSFORM

Detect straight lines in the edge-detected image and overlay them on the original frame.

6. VISUALIZATION

Display the original frame with detected lines and the edge-detected frame.



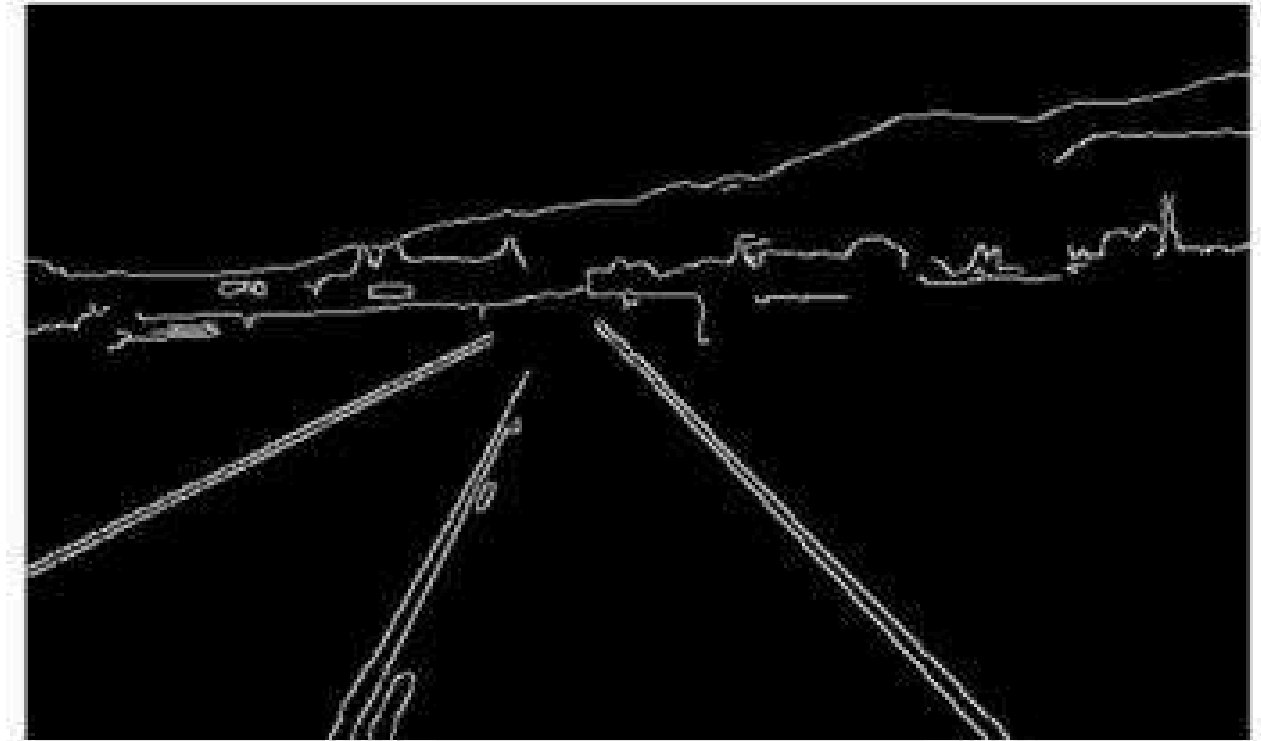
Input Image



Grayscale Image



Gaussian Blur



Canny Edge Detection Image

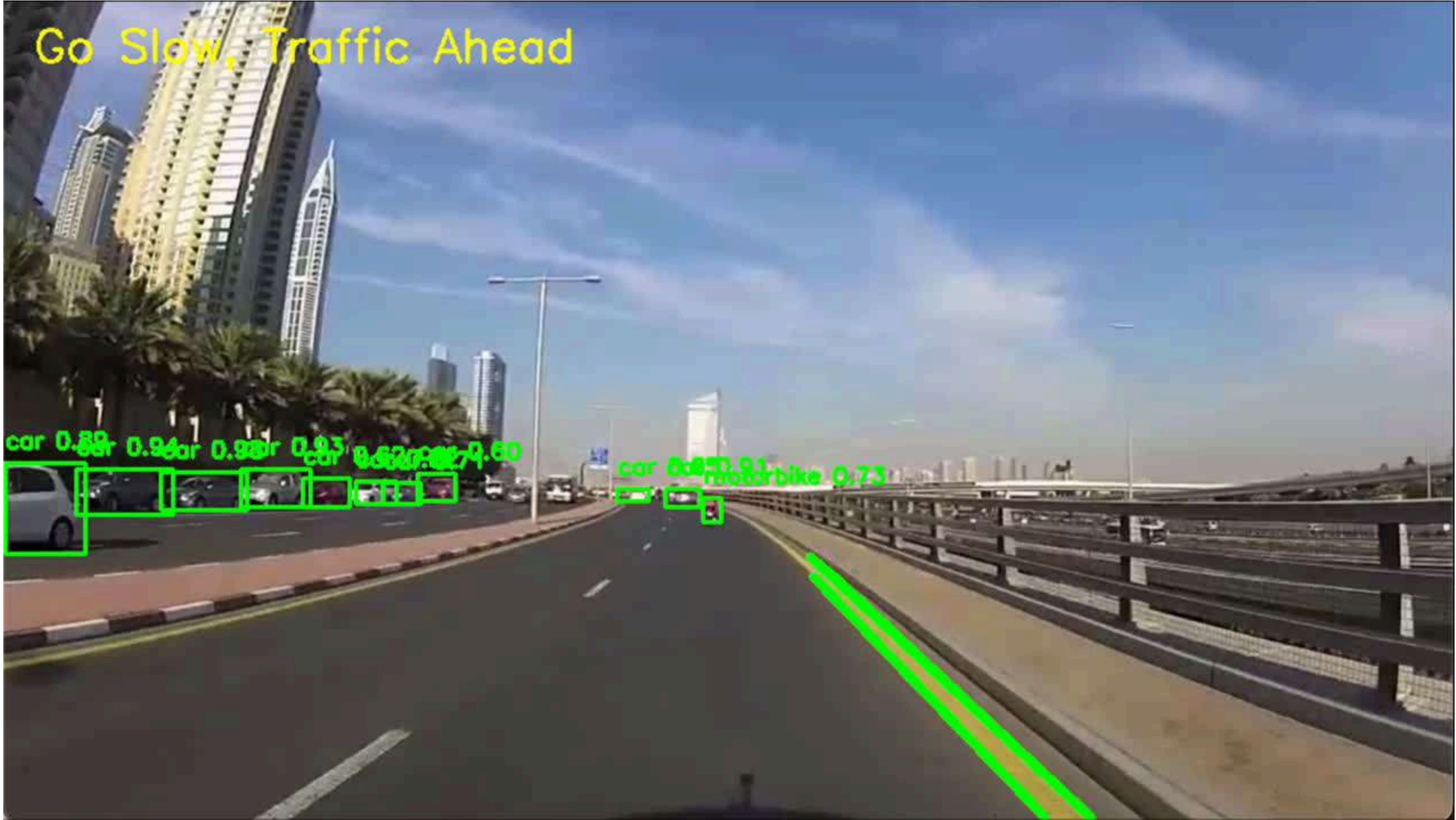
RESULT

1. Lane Detection:

- Green lines highlight the detected lanes.
- Instructions like "Stay on the Lane" or "Go Slow, Traffic Ahead" are displayed.

2. Object Detection:

- Bounding boxes with labels (e.g., "car", "person") are drawn around detected objects.



METRICS

- According to our research the accuracy of the modals using HSV were around 93.00% and using Deep Learning techniques were 95.00%.
- Our model shows promising results with an accuracy of 96.00% with an ADAS system and also using the YOLO v3 model for object detection.

LIMITATIONS AND FUTURE SCOPE

Nighttime Performance Issues –

- The model struggles with lane detection in low-light conditions. This could be due to insufficient contrast between the lane lines and the road.

Glare and Shadows –

- Headlights from other vehicles or strong shadows might cause false detections or missing lane lines.

FUTURE SCOPE

- Train YOLOv3 on a Nighttime Dataset: Collect and annotate nighttime images for better generalization.
- Adjust brightness to improve visibility.
- Modify HSV values to detect faint lane markings.



**THANK YOU
SO MUCH!**

Presented by Group 17