# Project Title

## Autonomous Vehicle Navigation System Using Computer Vision

- Clearly specify the main objective of your project in the title, which should focus on how computer vision techniques are used to assist in autonomous vehicle navigation.

### 2. Project Overview

- **Description:**
  - Provide a brief overview explaining the purpose of the system.
  - **Example:** "This project aims to develop an autonomous vehicle navigation system capable of detecting vehicles, traffic lights, and lane markings in real-time using computer vision techniques. The system interprets the driving environment from a video feed and makes decisions based on object detection, traffic signal recognition, and lane tracking to assist in safe vehicle navigation."
- **Objectives:**
  - Outline the goals of the project (e.g., accurate object detection, smooth lane following, reliable traffic light recognition, and real-time decision-making).

### 3. Installation and Setup

**Prerequisites:**

- **Hardware:**
  - A computer with a GPU (for faster video processing and deep learning models).
  - A camera (or a video file) for testing the navigation system.
- **Software:**
  - Python 3.x
  - OpenCV
  - YOLOv3 model files (config, weights, and class names).
  - Libraries like numpy, matplotlib, tensorflow, etc.
- **Example Commands to Install Dependencies:**
- bash
- Copy code
- pip install opencv-python numpy matplotlib tensorflow

**Directory Structure:**

- **Project Folder Structure Example:**

- bash
- Copy code
- /home/moni/Desktop/automous vehicle/yolo
- ├── all_in_one.py
- ├── yolov3.cfg
- ├── yolov3.weights
- ├── coco.names
- Explain the purpose of each file (e.g., all_in_one.py is the main script, yolov3.cfg contains the network configuration, yolov3.weights contains the pre-trained weights, and coco.names is the class label file).

**Running the Project:**

- **Example:** To run the main script, use:
- bash
- Copy code
- python3 all_in_one.py --input road_video.mp4 --output output_video.avi
- Describe the command-line options (e.g., --input for the input video file, --output for saving the processed video).

## 4. Features

**Existing Features:**

1. **Object Detection (YOLOv3):**
   - The system uses YOLOv3 to detect objects like cars, pedestrians, and traffic lights in real-time.
   - It implements Non-Maximum Suppression (NMS) to filter redundant boxes and maintain detection accuracy.
2. **Lane Detection:**
   - Lane detection is achieved using computer vision techniques:
     - **Canny Edge Detection**: Used to identify edges within the video frame.
     - **Hough Line Transform**: Detects straight lines in the frame that correspond to lane markers.
     - Provides guidance for the vehicle to stay within lane boundaries.

**New Features Added:**

1. **Traffic Light Color Recognition:**
   - Detects traffic lights and identifies the color (red, yellow, green) using the HSV color space.

- Controls the vehicle's response based on light colors (e.g., stop for red, slow for yellow, go for green).

2. **Steering Angle Calculation:**
   - Based on lane detection, the system calculates the vehicle’s steering angle.
   - Uses lane line slopes and positions to determine if the vehicle should turn left, right, or continue straight.
3. **Frame Skipping for Speed Optimization:**
   - To maintain real-time performance, the system skips processing some frames when high processing load is detected, improving overall efficiency.
4. **Movement Control:**
   - The system integrates multiple detections (object, lane, traffic light) to control vehicle movements like turning left/right, accelerating, or stopping.

## 5. System Design

**Architecture Overview:**

- Explain the general flow of the system:
  1. **Input Module**: Reads the video feed from a camera or a pre-recorded file.
  2. **Preprocessing Module**: Applies transformations such as resizing, color space conversion, and noise reduction.
  3. **Object Detection Module**: YOLOv3 is used to detect objects like cars, pedestrians, and traffic lights.
  4. **Lane Detection Module**: Uses edge detection and line detection algorithms to track lane lines.
  5. **Traffic Light Detection Module**: Identifies the presence and color of traffic lights.
  6. **Movement Control Module**: Makes decisions based on detected objects, lanes, and traffic lights.
  7. **Output Module**: Displays the processed video with annotations for detected objects, lanes, and actions like 'Turn Left', 'Stop', etc.

**Modules:**

- **Object Detection Module:** YOLOv3-based model for detecting multiple objects in real-time.
- **Lane Detection Module:** Canny edge detection + Hough Line Transform.
- **Traffic Light Detection Module:** Color detection using the HSV color space for traffic light color identification.
- **Steering Angle Calculation Module:** Uses lane positions to calculate the angle.
- **Movement Control Module:** Makes final decisions regarding acceleration, deceleration, or turning based on input from the object and lane detection modules.

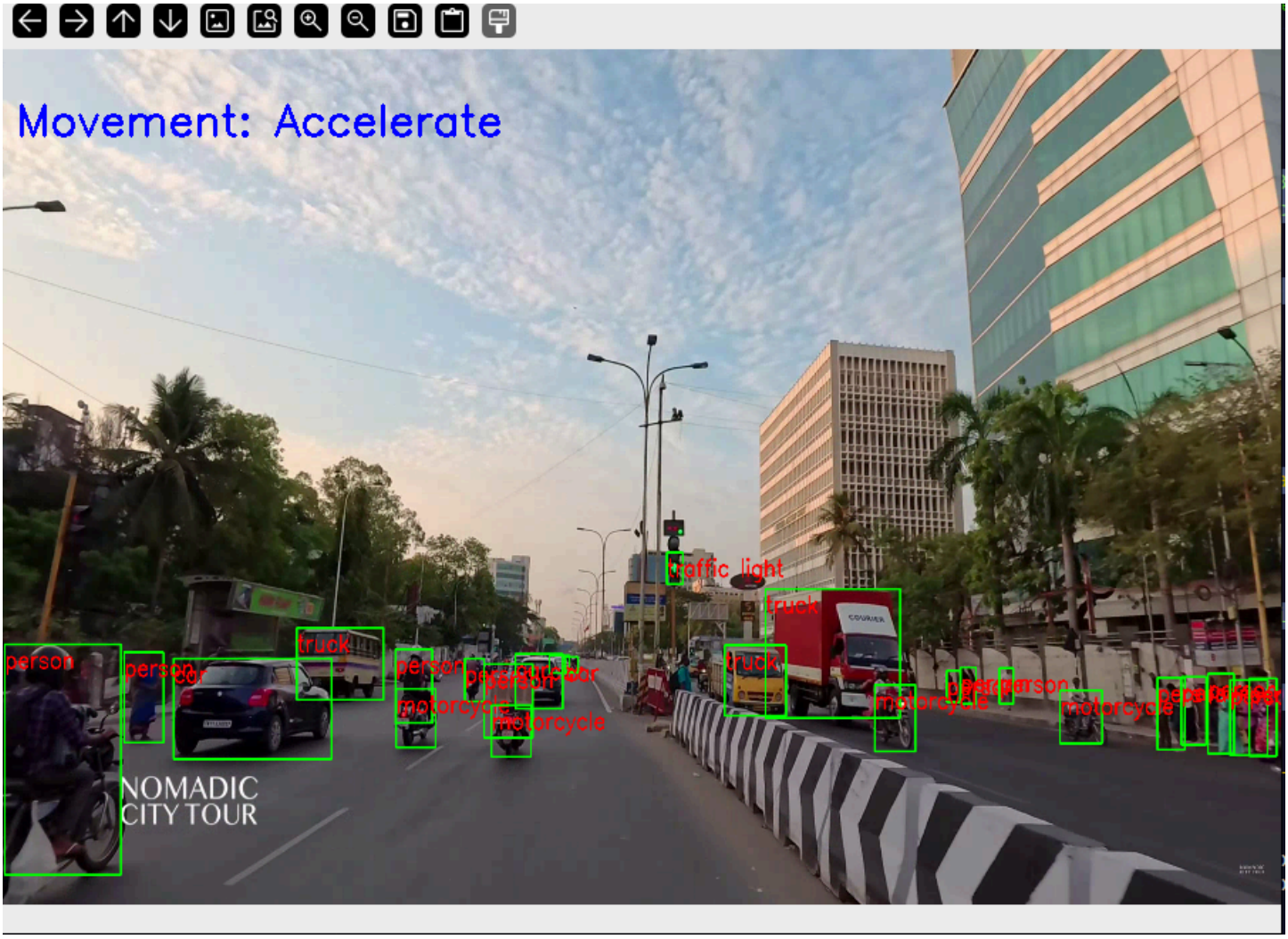
## 6. Input/Output

**Input:**

- Road videos taken from the vehicle's point of view.
- These videos should contain common road elements like lanes, traffic lights, vehicles, pedestrians, etc.

**Output:**

- A processed video showing:
  - Detected objects (e.g., cars, traffic lights) with bounding boxes and labels.
  - Lane lines clearly marked.
  - Text annotations for steering actions like “Turn Left”, “Turn Right”, “Stop”, or “Accelerate.”
  -
- **Screenshot Example:** Include an image from the processed output to show what the user can expect.



## 7. Performance Considerations

**Processing Speed:**

- Discuss the trade-offs between using high-resolution video and maintaining real-time performance.
- The frame skipping method to optimize video processing by skipping unnecessary frames without losing critical information.

**Real-time Constraints:**

- Factors affecting real-time processing such as video frame rate, object detection model complexity (YOLOv3), and the use of GPU acceleration.

## 8. Challenges Faced and Solutions

**Challenges:**

1. **Object Overlap:**
   - When multiple objects are detected within close proximity, bounding boxes might overlap or interfere with each other.
2. **Low Light/Weather Conditions:**

- Difficulties in detecting lanes and objects under poor lighting or adverse weather conditions like rain or fog.
3. **Balancing Detection Accuracy and Speed:**
   - YOLOv3 offers high accuracy but can slow down when working with high-resolution video.

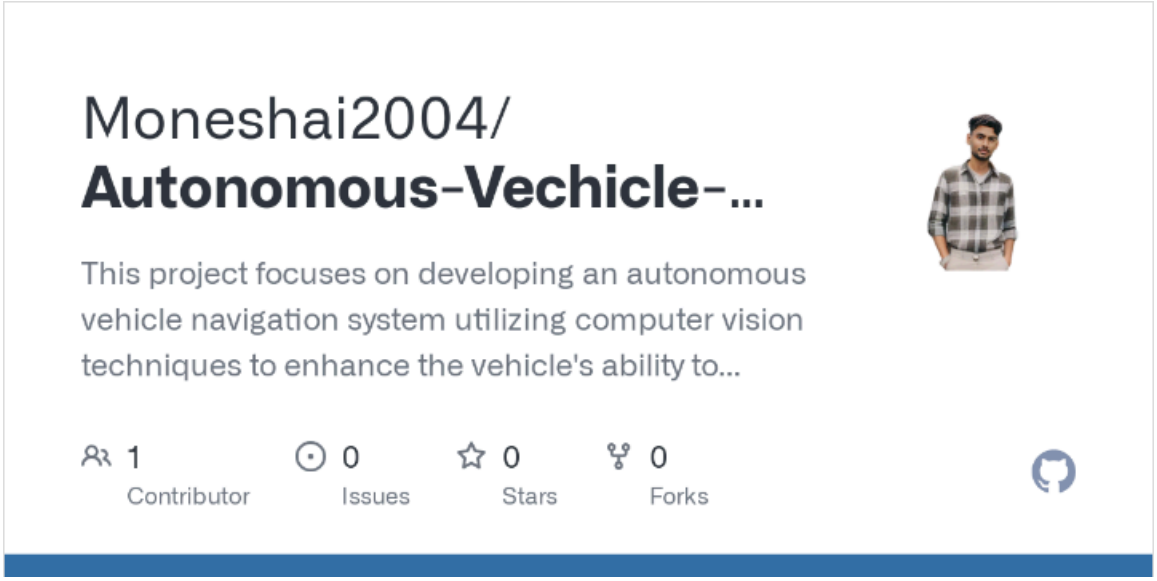
**Solutions:**

1. **Non-Maximum Suppression (NMS):**
   - Implemented to handle overlapping bounding boxes, keeping the highest-confidence detection.
2. **Dynamic Lane Thresholds:**
   - Adaptive thresholds for edge detection to handle different lighting and weather conditions.
3. **Frame Skipping and GPU Acceleration:**
   - Skipping non-critical frames and utilizing GPU for processing to maintain real-time speeds.

## 9. Future Enhancements

- **Custom Object Detection Models:**
  - Training a custom YOLO model to detect additional road objects like cyclists, road signs, or animals.
- **Weather Robustness:**
  - Enhancing the system to function accurately under different weather conditions (e.g., rain, snow).
- **Sensor Fusion:**
  - Integrating additional sensors such as LiDAR or RADAR for better accuracy in detecting objects and obstacles, especially in low-visibility scenarios.

**Github Link:**




## 10. Conclusion

- **Summary:**
  - The project successfully developed an autonomous navigation system using YOLOv3 for object detection, lane detection, and traffic light recognition. The system can make real-time decisions based on environmental input and can be further enhanced by training custom models and adding more sensor data.