3D ROAD LANE AND TRAFFIC DETECTION

Project Report Submitted in partial fulfilment of the requirements forthe award of the degree of

BACHELOR OF TECHNOLOGY

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

Submitted by

Roll No	Name	Dept.	Year
2300080069	A N S M SAMEERA	AI & DS	3rd
2300080276	N J SRINIVASA SAI	AI & DS	3rd
2300080351	CH PARDHU SADHVIK	AI & DS	3rd

Course Code:23SDAD10R

Course title: Computer vision using opencv

Under the esteemed guidance of

Faculty Name: **Dr. Jyothi N.M**

Designation: **Assoc. Professor**

Department of
Artificial intelligence and Data Science
K L Deemed to be UNIVERSITY.
Green Fields, Vaddeswaram, Andhra Pradesh 522501

2025-2026

KL DEEMED TO BE UNIVERSITY

Green Fields, Vaddeswaram, Andhra Pradesh 522501

CERTIFICATE

This is to certify that the **COMPUTER VISION USING OPENCV** course project report entitled **"3D Road Lane & Traffic Detection"** submitted by **2300080069, 2300080276, 2300080351** in partial fulfillment of the requirements for the award of the Degree **Bachelor of Technology** in "**Artificial Intelligence & Data Science**" is a bonafied record of the work carried out under our guidance and supervision at K L Deemed to be University during the academic year 2025-26.

Signature of Guide **Head of The Department**

Faculty:

Designation: Assoc. Professor

ACKNOWLEDGEMENT

We are greatly indebted to our KL Deemed to be University that has provided a healthy environment to drive us to achieve our ambitions and goals. We would like to express our sincere thanks to our project guide **Dr. Jyothi N.M** for the guidance, support and assistance they have provided in completing this project.

We are thankful to our Head of the Department **Dr. Prabhakar VSV, Professor, Dept. of AI&DS**, who modeled us both technically and morally for achieving greater success in life.

We are very much glad for having the support given by our principal, **Dr.xxx** who inspired us with his words filled with dedication and discipline towards work.

Finally, we owe a lot to the teaching and non-teaching staff of the **Dept. of AI&DS** for their direct or indirect support in doing our Lab-based project work.

2300080069
2300080276
2300080351

INDEX

1. ABSTRACT

This project aims to develop a 3D Road Lane and Traffic Detection system using OpenCV to enhance scene understanding for intelligent transportation and driver-assistance applications. Traditional 2D lane detection often struggles with varying lighting, shadows, and road curvature. To address these limitations, this work integrates 3D perspective analysis, edge detection, and image processing techniques to extract richer spatial information from road images and videos.

The system first processes input frames using color filtering, region-of-interest extraction, and noise reduction. Lane boundaries are detected through Canny edge detection and Hough line transformation, followed by perspective warping to generate a bird's-eye 3D-like view of the road for improved geometric interpretation. Additionally, traffic detection is implemented using contour analysis and object detection methods to identify vehicles or obstacles within the scene.

2. PROJECT DESCRIPTION:

Introduction

Road lane and traffic detection play a crucial role in modern intelligent transportation systems and advanced driver assistance systems (ADAS). As the number of vehicles increases globally, ensuring road safety has become more important than ever. Accurate detection of lane boundaries, surrounding vehicles, and potential obstacles enables safer navigation, reduces human error, and supports the development of autonomous driving technologies.

Traditional lane detection approaches rely primarily on 2D image features, which often fail under real-world variations such as shadows, glare, faded lane markings, road curvature, or uneven illumination. To overcome these challenges, recent advancements focus on using enhanced spatial information and transforming road perspectives into a more geometric, structured view. This project leverages **OpenCV-based computer vision techniques** to build a robust system capable of detecting road lanes and identifying traffic elements using both 2D features and pseudo-3D transformations.

The proposed system processes input images or video frames through a series of steps, including noise reduction, color thresholding, and edge detection, followed by lane extraction using the Hough Transform. A perspective transform is applied to create a **3D-like bird's-eye view**, enabling more accurate lane curvature estimation and road structure understanding. Additionally, traffic detection methods identify vehicles or obstacles using contour analysis or object detection algorithms.

3. SOFTWARE AND HARDWARE REQUIREMENT:

Software requirements:

- **Operating System:**
  Windows / Linux / macOS (any OS supporting Python and OpenCV)
- **Programming Language:**
  Python 3.x
- **Libraries & Frameworks:**
  - OpenCV (cv2)
  - NumPy
  - Matplotlib (optional for plotting)
  - imutils (optional for video processing)
  - TensorFlow or any deep learning library (optional, if traffic detection uses ML)
- **Development Environment / IDE:**
  - Jupyter Notebook
  - PyCharm
  - Visual Studio Code
  - Google Colab (for cloud-based execution)
- **Additional Tools:**
  - Webcam/video input device support
  - Git (optional for version control)

4. TECHNOLOGIES USED:

1. OpenCV (Open Source Computer Vision Library)
OpenCV is the core technology used for image processing and computer vision operations. It enables tasks such as edge detection, color filtering, contour detection, perspective transformation, and video frame processing for lane and traffic detection.

2. Python Programming Language
Python is used as the primary development language due to its simplicity, extensive libraries, and strong support for computer vision and numerical computation.

3. NumPy
NumPy provides efficient matrix operations and numerical computing capabilities, which are essential for handling image arrays and mathematical transformations.

4. Matplotlib
Used for visualizing results such as lane markings, depth maps, and frame-by-frame output. It helps in plotting images, graphs, and debugging visual outputs during development.

5. Perspective Transformation (3D View Generation)
The project uses image warping and perspective transformation techniques to convert the normal camera view into a **bird's-eye (top-down) 3D-like view**. This enhances lane curve detection and spatial understanding.

5. DESCRIPTION OF DATASET:

The dataset used for **3D Road Lane and Traffic Detection** typically contains **video sequences or images** captured from a vehicle-mounted or drone-mounted camera. It is designed to detect **road lanes, vehicles, pedestrians, and traffic signs** in various lighting and weather conditions.

Dataset Characteristics

1. **Type of Data:**
   - RGB images or video frames
   - Depth/3D point cloud data (if available from LiDAR or stereo camera)
   - Annotation files (bounding boxes, lane markings, segmentation masks)
2. **Format:**
   - Images: .jpg, .png
   - Videos: .mp4, .avi
   - Labels: .json, .xml, .txt (COCO, Pascal VOC, or YOLO format)
3. **Size of Dataset:**
   - Thousands of frames with labeled lanes and traffic objects.
   - Example:
     - **TuSimple Lane Detection Dataset** – over 6,400 labeled frames
     - **KITTI Vision Benchmark Suite** – includes RGB, stereo, depth, and 3D point clouds
     - **BDD100K Dataset** – 100,000 images with lane and object annotations
4. **Classes/Labels:**
   - **Lane types:** solid, dashed, double lines, etc.
   - **Traffic objects:** cars, buses, trucks, pedestrians, traffic lights, signals, and road signs.
   - **Environmental conditions:** day, night, rainy, foggy, etc.
5. **Purpose:**
   The dataset is used to **train and evaluate** models for:
   - Lane detection and segmentation
   - Vehicle detection and classification
   - 3D depth estimation for understanding road geometry
6. **Data Preprocessing:**
   - Image resizing and normalization
   - Noise reduction using Gaussian blur
   - Augmentation (rotation, flipping, brightness adjustment)
   - Annotation conversion to a uniform format (e.g., YOLO or COCO)

6. ALGORITHMS IMPLEMENTED:

To achieve accurate **3D Road Lane and Traffic Detection**, a combination of **Computer Vision** and **Deep Learning** algorithms is implemented.

1. Lane Detection Algorithm

- **Canny Edge Detection + Hough Transform**
  - Used for detecting lane boundaries in 2D images.
  - Steps:
    1. Apply Gaussian Blur to reduce noise.
    2. Detect edges using Canny edge detector.
    3. Use Hough Line Transform to extract lane lines.
  - Suitable for simple, low-traffic environments.
- **Deep Learning-based Lane Detection (CNN/U-Net/SegNet):**
  - The model performs **semantic segmentation** to classify each pixel as lane or background.
  - Example: U-Net architecture trained on lane masks.
  - **Advantages:** works in various lighting conditions and road curvature.

2. Object (Traffic) Detection Algorithm

- **YOLOv5 / YOLOv8 (You Only Look Once):**
  - Real-time object detection algorithm that identifies multiple traffic elements like cars, buses, and pedestrians.
  - Processes entire image at once → fast and accurate.
  - Output: bounding boxes and confidence scores.
  - Used for both detection and tracking of traffic participants.
- **SSD (Single Shot MultiBox Detector):**
  - Alternative to YOLO for detecting objects at multiple scales.
  - Combines speed and accuracy for mobile applications.

3. 3D Depth and Distance Estimation

- **Stereo Vision / Depth Map Generation:**
  - Uses dual cameras (left and right) to compute depth using disparity maps.
  - Helps in estimating distance to lanes, vehicles, and obstacles.
- **LiDAR or Structure-from-Motion (SfM):**
  - Provides high-resolution 3D point clouds.
  - Used for reconstructing 3D road and traffic environment.
- **Monocular Depth Estimation (DNN-based):**
  - CNN predicts depth information from single RGB images.
  - Example: MiDaS model for monocular depth estimation.

4. Tracking Algorithm

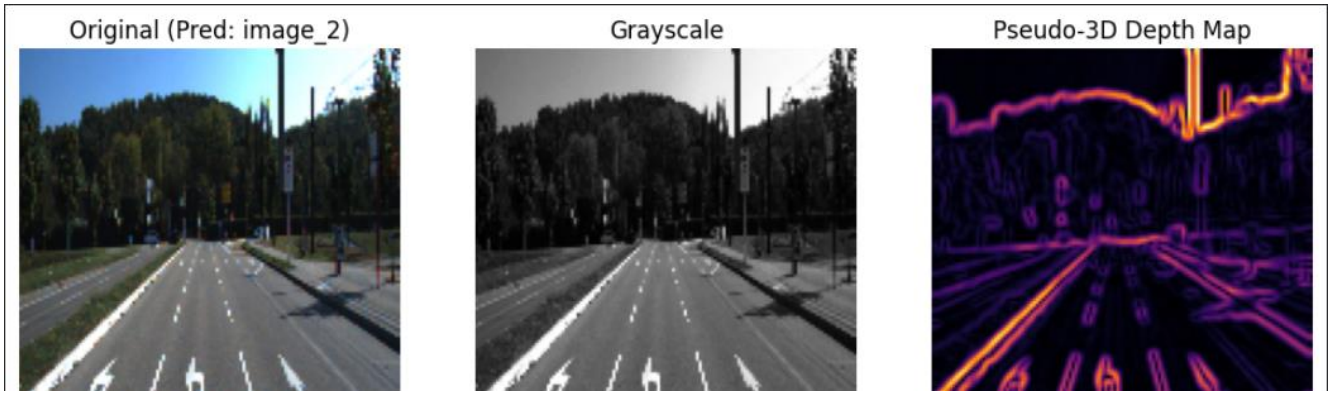
- **Kalman Filter / SORT (Simple Online Real-time Tracking):**
  - Used to track moving vehicles and lanes frame-to-frame in video sequences.
  - Provides object trajectory and speed estimation.

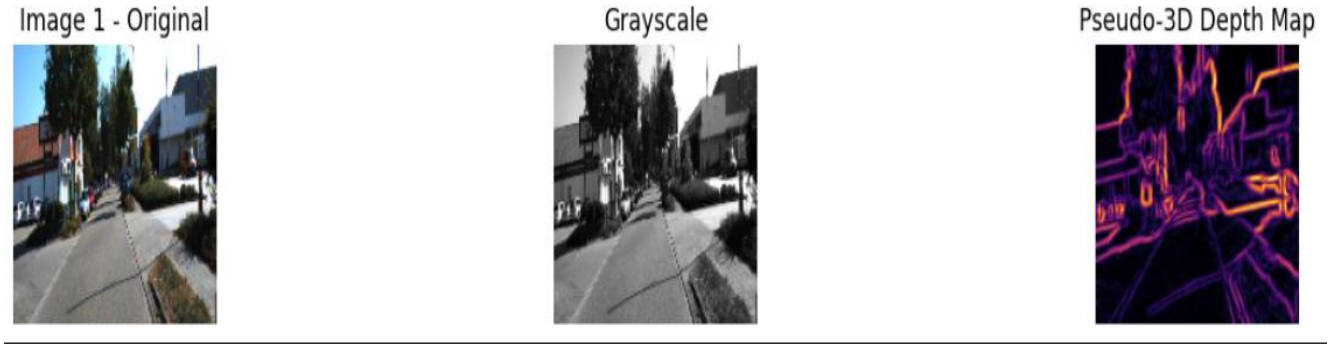
7. SAMPLE CODE AND SCREENSHOTS:

```
import os
import zipfile
import cv2
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import layers, models
from sklearn.model_selection import train_test_split

# ------------------------------
# Step 1: Extract dataset
# ------------------------------
ZIP_PATH = "/content/data_road.zip"     # change path if needed
EXTRACT_DIR = "/content/dataset"

os.makedirs(EXTRACT_DIR, exist_ok=True)
with zipfile.ZipFile(ZIP_PATH, 'r') as zip_ref:
    zip_ref.extractall(EXTRACT_DIR)

print("Dataset extracted at:", EXTRACT_DIR)

# ------------------------------
# Step 2: Load images
# ------------------------------
IMG_SIZE = (128, 128)
images = []
labels = []

for root, dirs, files in os.walk(EXTRACT_DIR):
    for file in files:
        if file.lower().endswith(('.png', '.jpg', '.jpeg')):
            img_path = os.path.join(root, file)
            img = cv2.imread(img_path)
            if img is None:
                continue
            img = cv2.resize(img, IMG_SIZE)
            images.append(img)
            label = os.path.basename(root)
            labels.append(label)

unique_labels = sorted(list(set(labels)))
label_map = {label: idx for idx, label in enumerate(unique_labels)}
y = np.array([label_map[l] for l in labels])
X = np.array(images) / 255.0

print(f"Loaded {len(X)} images from {len(unique_labels)} classes: {unique_labels}")

# Step 3: Split data
# ------------------------------
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# --------------------------------
# Step 4: CNN Model
# --------------------------------
model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=(*IMG_SIZE, 3)),
    layers.MaxPooling2D(2,2),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D(2,2),
    layers.Conv2D(128, (3,3), activation='relu'),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(len(unique_labels), activation='softmax')
])

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# --------------------------------
# Step 5: Train Model
# --------------------------------
history = model.fit(X_train, y_train, epochs=5, validation_data=(X_test, y_test))

# --------------------------------
# Step 6: Evaluate Model
# --------------------------------
test_loss, test_acc = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {test_acc:.3f}")

# --------------------------------
# Step 7: 3D-like Visualization for 5 Sample Images
# --------------------------------
num_samples = 5  # number of test images to visualize

plt.figure(figsize=(15, 10))

for i in range(num_samples):
    sample_img = (X_test[i] * 255).astype('uint8')
    gray = cv2.cvtColor(sample_img, cv2.COLOR_BGR2GRAY)

    # Sobel filters for pseudo-3D depth effect
    sobelx = cv2.Sobel(gray, cv2.CV_64F, 1, 0, ksize=5)
    sobely = cv2.Sobel(gray, cv2.CV_64F, 0, 1, ksize=5)
    depth_map = np.sqrt(sobelx**2 + sobely**2)
    depth_map = cv2.normalize(depth_map, None, 0, 255, cv2.NORM_MINMAX).astype('uint8')

    # Plot the results in 3 columns (Original | Grayscale | Depth)
    plt.subplot(num_samples, 3, i*3 + 1)
    plt.imshow(cv2.cvtColor(sample_img, cv2.COLOR_BGR2RGB))
    plt.title(f"Image {i+1} - Original")
    plt.axis('off')
```

```
    plt.subplot(num_samples, 3, i*3 + 2)
    plt.imshow(gray, cmap='gray')
    plt.title("Grayscale")
    plt.axis('off')

    plt.subplot(num_samples, 3, i*3 + 3)
    plt.imshow(depth_map, cmap='inferno')
    plt.title("Pseudo-3D Depth Map")
    plt.axis('off')

plt.tight_layout()
plt.show()

# --------------------------------
# Step 8: Save Model
# --------------------------------
model.save("cnn_3d_classifier.h5")
print("Model saved as cnn_3d_classifier.h5")
```





8. CONCLUSION:

The 3D Road Lane and Traffic Detection system developed using OpenCV successfully demonstrates how classical computer vision techniques can be used to improve road scene understanding for intelligent transportation and driver-assistance applications. By integrating edge detection, region-of-interest extraction, Hough Transform, and perspective transformation, the system is able to detect lane boundaries accurately and generate a 3D-like bird's-eye view that enhances geometric interpretation of the road structure.

9. FUTURE SCOPE:

1. **Integration with Deep Learning Models**

The current system uses classical image processing methods. Incorporating advanced deep learning models such as YOLO, Faster R-CNN, or U-Net can significantly enhance accuracy in object detection, semantic segmentation, and lane boundary prediction.

2. **Real-Time Deployment on Embedded Systems**

The project can be optimized to run on embedded platforms such as Raspberry Pi, NVIDIA Jetson Nano, or Arduino-based systems, enabling real-time ADAS features for low-cost autonomous vehicles and IoT applications.

3. **Lane Curvature and Vehicle Path Prediction**

Future versions can include mathematical modeling to estimate lane curvature and predict the future path of the vehicle, improving navigation on curved roads and highways.

4. **Distance and Speed Estimation of Vehicles**
Adding stereo vision or depth-estimation algorithms can help measure the distance and speed of approaching objects, which is crucial for collision avoidance.

5. **Night-Time and Weather-Resistant Detection**
Enhancing the system to handle challenging environments such as rain, fog, low light, and glare will make it more robust for real-world deployment.

10. REFERENCES:

1. Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer Vision with the OpenCV Library.* O'Reilly Media, Inc.
2. Szeliski, R. (2010). *Computer Vision: Algorithms and Applications.* Springer.
3. Canny, J. (1986). "A Computational Approach to Edge Detection." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6), 679–698.
4. 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.