WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

A Review on Identification and Analysis of various Maneuver at Intersections

Dr. Atmakuri Priyanka, Jupudi Surendra, Alla vinay, Arasavalli Monika, Gurugubelli Soniya, Savalapurapu Neelima

UG Students, Department of civil Engineering, GMR Institute of Technology, Vizianagaram District, A.P, India

DOI : https://doi.org/10.55248/gengpi.6.0425.14116

A B S T R A C T

Intersections are critical points in any road network, where multiple traffic streams converge, making them prone to congestion, delays, and accidents. In India, intersection maneuvering is particularly complex due to heterogeneous traffic conditions, lack of lane discipline, frequent rule violations, and varying driver behaviors. This study aims to analyze intersection maneuvering by focusing on three key aspects: driver behavior, road safety, and signal optimization. Driver behavior at intersections significantly influences traffic efficiency and safety, as factors such as risk perception, compliance with signals, and decision-making affect overall movement patterns. Road safety analysis helps identify high-risk zones, common causes of accidents, and potential mitigation measures to enhance intersection safety. Additionally, signal optimization is crucial for improving traffic flow by reducing delays, minimizing congestion, and enhancing vehicular movement efficiency. This research involves a comprehensive study of traffic patterns, accident data analysis, and signal timing evaluation to propose effective strategies for safer and more efficient intersections. The findings can aid in the development of better traffic management policies, improved intersection design, and enhanced enforcement measures, ultimately contributing to a safer and more organized road network in India.

Keywords: Obstacle Detection, Deep Learning, Real-Time Monitoring, Automated Alerts, Image Processing, Machine Learning

1. INTRODUCTION

Intersections are key elements in both urban and rural road networks, where multiple traffic streams converge. In India, these junctions often face challenges such as heterogeneous traffic, poor lane discipline, and frequent rule violations, resulting in congestion, delays, and increased accident risks. A maneuver refers to any deliberate change in a vehicle's direction or speed, including lane changes, turns, merging, overtaking, and emergency actions, all influenced by road and traffic conditions. This study focuses on understanding driver behavior, enhancing road safety, and optimizing traffic signals at intersections. Analyzing driver behavior helps assess compliance and reaction times, while road safety studies identify high-risk areas and patterns. Signal optimization, aided by real-time maneuver detection, enables dynamic signal control, which improves traffic flow and reduces accidents, contributing to safer and more efficient intersections.

2. LITERATURE REVIEW

- **Charishma Takkallapalli et al. (2023)**[1]. analyzed acceleration and deceleration behavior at signalized intersections in Kolkata, India, under mixed traffic conditions. Using IMU and GPS-equipped vehicles, the study captured vehicle dynamics to develop a maneuver detection algorithm that classifies movements (straight, left, right, U-turn) through threshold-based techniques. This research supports improvements in intersection design, traffic signal optimization, and emission modeling.
- **Nuksit Noomwongs et al. (2013)**[2]. focused on detecting maneuvers like lane changes, overtaking, turning, merging, and emergency actions using image and video data from real-world Indian traffic. The study determined threshold values for classifying four-wheeler maneuvers using vehicle-mounted camera footage, aiding in traffic safety and management at intersections. The research employed real-world image and video-based data from Indian roads, specifically collected through vehicle-mounted cameras. A central focus was identifying threshold values to classify maneuvers of four-wheelers, enabling accurate detection of driver behavior in complex traffic scenarios.
- **Xishuai Peng et al. (2020)**[3]. developed a model that integrates vehicle trajectory data, video-based scene recognition, and sequential learning for maneuver detection. Using an LSTM network and video features from the VGG-19 model trained on the Places-365 dataset, the study captures temporal driving behavior. The proposed UMD-DMED model achieved 90.4% precision and 89.9% recall, outperforming existing methods in detecting and predicting maneuvers like turns, lane changes, and straight driving.

- **Satyajit Mondal et al. (2022**)[4]. conducted a study analyzing driver acceleration and deceleration behaviors at signalized intersections in Chandigarh, India. Using GPS-based vehicle trajectory data from 166 test runs across five vehicle types, the research revealed that lighter vehicles like cars and two-wheelers exhibit higher acceleration rates at lower speeds, requiring less time and distance to accelerate compared to heavier vehicles such as buses and light commercial vehicles. The study developed single and multi-regime polynomial models to represent acceleration/deceleration characteristics based on vehicle approach speeds. These models were validated using 30% of the collected dataset, achieving a Mean Absolute Percentage Error (MAPE) of less than 10%, demonstrating their reliability and accuracy.
- **Christopher Woo (2016)**[5]. introduced a kinematic model to track vehicle position and orientation for maneuver detection using time-series data from smartphone sensors. The study emphasized the importance of real-time classification of driver behavior through IMU-based features like acceleration and angular velocity, supporting active safety technologies in modern automotive transportation.
- **K. Ramachandra Rao (2022)**[6]. developed predictive models for rear-end and side-swipe conflicts at urban signalized intersections under disordered traffic conditions. The study utilized video data from four intersections to extract 9,586 vehicle trajectories, employing a two-dimensional surrogate safety approach with Time to Collision (TTC) as a primary indicator. The analysis considered vehicle dimensions, position, speed, acceleration, and direction in both longitudinal and lateral movements. Findings indicated that rear-end conflicts increased with higher traffic volumes, vehicle speeds, and right-turning traffic, with a 6% rise in severe conflicts for every 1 m/s speed increase. Side-swipe conflicts were linked to frequent lane-changing and right-turning maneuvers, with each lane change contributing to a 7.9% increase in conflicts. These insights are crucial for enhancing traffic safety measures at urban intersections.
- **Javier Cervantes-Villanueva (2016)** [7].focused on detecting vehicle maneuvers using smartphone-based accelerometer data to classify turns, braking, and acceleration. Real-world data was preprocessed and used to train machine learning models, achieving high precision in maneuver detection. The study shows that accelerometer-based classification is effective for real-time, non-intrusive detection and supports advancements in intelligent transportation systems.

3. METHODOLOGY

3.1 DEFINING THE PROBLEM

- Intersections in India are among the most congested and accident-prone areas of the road network, characterized by chaotic traffic movement, poor lane discipline, and frequent violations of traffic rules. The heterogeneous nature of traffic, including the coexistence of motorized and non-motorized vehicles, further complicates maneuvering at intersections. Inefficient intersection management leads to increased travel time, fuel consumption, and road safety risks.
- Driver behavior at intersections, such as non-compliance with traffic signals, aggressive driving, and improper right-of-way decisions, significantly affects traffic efficiency and safety. Additionally, the lack of well-optimized traffic signals contributes to excessive delays and unnecessary congestion. Road safety concerns at intersections stem from a high incidence of collisions, pedestrian vulnerability, and inadequate enforcement of traffic regulations.
- Despite various traffic management measures, many intersections in India continue to face operational and safety challenges. There is a need for an in-depth analysis of driver behavior, accident patterns, and traffic signal efficiency to develop effective strategies for intersection improvements. This study aims to identify key issues in intersection maneuvering and propose solutions to enhance safety, reduce congestion, and optimize signal operations for better traffic flow.

3.2 MATERIALS USED

Materials in the development of an efficient obstacle detection system are essential in this research. The following were used:

Datasets:

High variety of images, Videos of Types of Vehicles and Intersections: Two wheelers, Four Wheelers, Six Wheelers.

Hardware:

1) **HP Laptop**: The primary computing device used for executing the model, equipped with a **GeForce GTX 1650 GPU** to provide the necessary computational power for processing video feeds and running deep learning algorithms efficiently.
2) **IMU Device**: Inertial Measurement Unit (IMU) is a sensor-based device used in transportation to measure and track an object's motion, orientation, and velocity. It consists of accelerometers, gyroscopes, and sometimes magnetometers, which work together to provide precise movement data. IMUs play a crucial role in various transportation applications, including vehicle navigation, autonomous driving, and traffic monitoring. They help in detecting vehicle maneuvers such as acceleration, braking, lane changes, and turns, making them valuable for driver behavior analysis and road safety studies.

3) **Camera**: This high-resolution camera was employed for physical data collection, capturing images and videos of various intersections at different locations.

4) **GPS Device:** Global Positioning System (GPS) devices have revolutionized the transportation sector by providing accurate location tracking, navigation, and real-time data for efficient mobility. GPS technology uses a network of satellites to determine the precise position of vehicles, enabling seamless navigation and route optimization. In transportation, GPS devices play a crucial role in fleet management, traffic monitoring, public transportation, and personal navigation.

5) **Storage Device**: A reliable external or internal storage solution was necessary to store the extensive dataset collected from online repositories and physical captures, ensuring easy access during model training and evaluation.

Software:

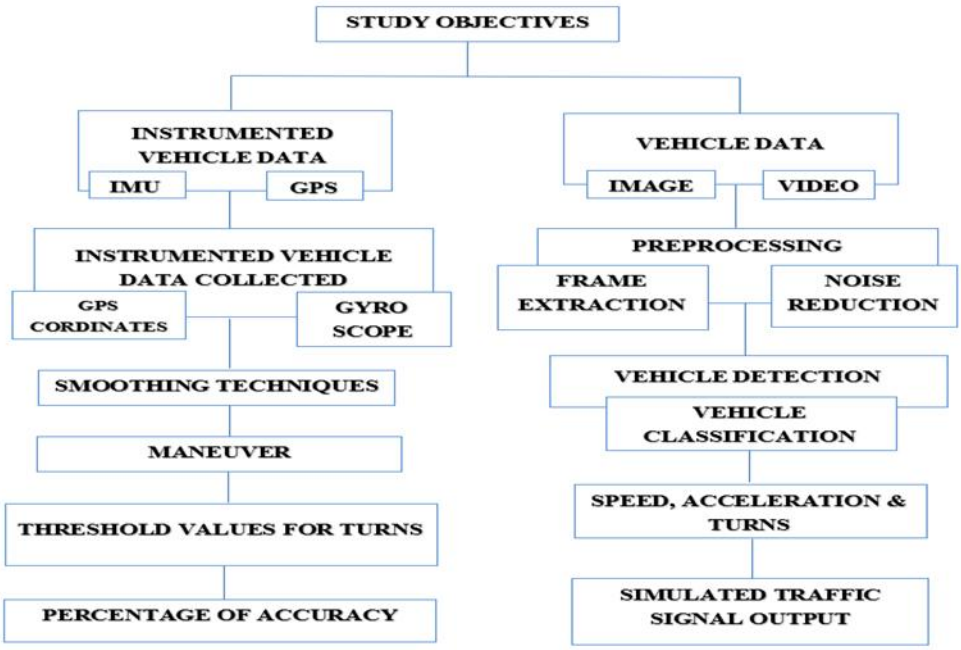
1) Deep learning algorithms were implemented using Python as the programming language.

2) Libraries and frameworks such as TensorFlow, Keras, and OpenCV have been utilized to process the images and develop models

3) Data annotation tools

Make sense AI and CVAT are used for labelling images in the datasets.

3.3 METHODOLOGY



3.4 TOOLS AND INSTRUMENTS USED

Some of the tools and instruments that were used during the data analysis process of the research period include the following.

- **Programming Environment:** Google Collab was used for coding and interactive test Python scripts.
- **Deep Learning Frameworks:** TensorFlow and Keras respectively, were used to develop and train deep learning models.
- **Image Processing Tools:** OpenCV is used for most other image processing algorithms.
- **Data Visualization Tools:** Matplotlib and Seaborn in Python were used to visualize every possible representation while training, including loss curves, as well as evaluation metrics.
- **Performance Evaluation Tools:** The sci-kit-learn library helped to compute performance metrics like confusion matrices, precision-recall curves, and ROC curves.

3.5 STEP BY STEP PROCESS USING INSTRUMENTED VEHICLE DATA

3.5.1 DATA COLLECTION

- Identify study intersections based on traffic volume, road geometry, and signalization, ensuring a mix of high-density and moderate-density intersections for comprehensive analysis.
- Install high-resolution cameras at strategic locations, including roadside poles, drones, or vehicle-mounted setups, to capture real-time vehicle movements from multiple angles.
- Use GPS and IMU sensors on test vehicles to track parameters like speed, acceleration, deceleration, and positioning, allowing precise maneuver detection in dynamic traffic conditions.
- Collect traffic signal timings and road environment data, including pedestrian crossings, lane markings, and signage, to understand their influence on driver behavior at intersections.
- This comprehensive approach to data collection ensures that the model is trained on a wide variety of scenarios, enhancing its ability to perform accurately in real-world applications.

3.5.2 DATA PREPROCESSING

- After data collection, the dataset underwent a thorough pre-processing phase to ensure its quality and relevance for model training.
- The first critical step involved cleaning the dataset by removing irrelevant and low-quality images that could negatively impact the model's performance.
- Once cleaned, images were annotated using specialized data annotation tools to accurately identify and label obstacles, which is essential for supervised learning.
- The labelling process serves as the foundational source from which the model derives its knowledge, making it a crucial step in developing an effective detection system.

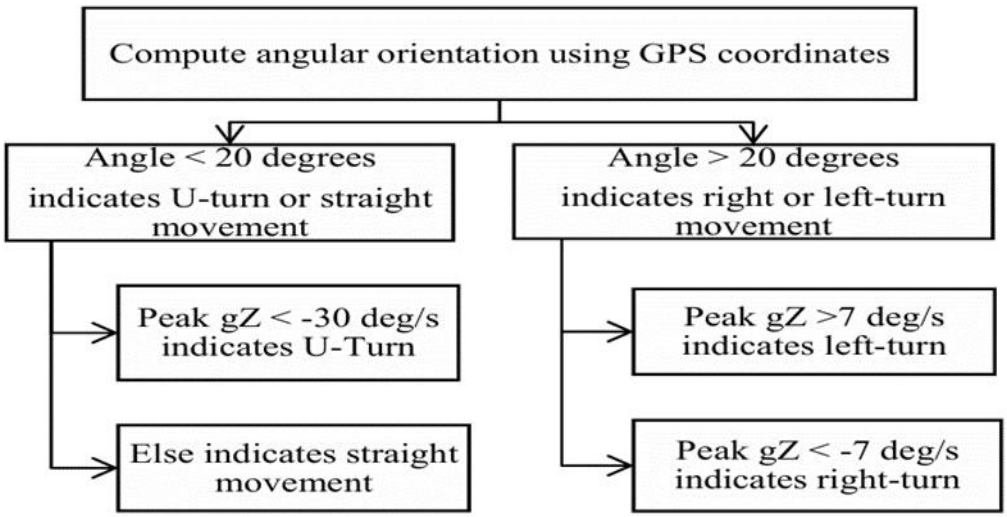
3.5.3 SELECTION OF SMOOTHING TECHNIQUE:

Smoothing techniques are essential in vehicle maneuver detection to enhance data quality, reduce noise, and improve the accuracy of movement analysis. The primary reasons for using smoothing techniques include:

- Moving Average Filter
- Gaussian Smoothing
- Kalman Filter
- Particle Filter

3.5.4 PREDICTING GYROSCOPE VALUES USING THRESHOLDS:

To predict gyroscope values using threshold-based classification for maneuver detection at intersections, you need to analyse angular velocity data from the gyroscope sensor. The gyroscope measures the rate of rotation along three axes, typically expressed in degrees per second (°/s). Your methodology involves setting threshold values to classify different maneuvers based on these angular velocities.



Flowchart-3: Compute Angular orientation

1. Compute Angular Orientation:

GPS coordinates are used to determine the angular change in vehicle direction.

If the angle of deviation is less than 20 degrees, the movement is classified as either a U-turn or straight movement.

If the angle exceeds 20 degrees, the movement is classified as either a right or left turn.

2.Using Gyroscope Peak Values for Classification:

The gyroscope's Z-axis (gZ) value, which represents yaw rotation (turning movement), is analysed to refine maneuver classification.

- ✓ If gZ < -30°/s, the vehicle is making a U-turn.
- ✓ If gZ is between -30°/s and 7°/s, the vehicle is moving straight.
- ✓ If gZ > 7°/s, the maneuver is classified as a left turn.
- ✓ If gZ < -7°/s, the maneuver is classified as a right turn.

3. Implementation in Algorithm:

- Extract gyroscope and GPS data from vehicle-mounted sensors.
- Compute angular deviation using GPS coordinates.
- Compare gyroscope peak values with predefined thresholds.
- Assign maneuver labels based on the computed thresholds.

4. Refining Thresholds Using Data Analysis:

- Thresholds can be fine-tuned using machine learning techniques such as decision trees, clustering, or neural networks.
- Real-time gyroscope data can be collected and analyzed to dynamically adjust threshold values for better accuracy.

5. Applications in Traffic Signal Simulation:

- The classified maneuvers can be used to develop traffic signal control strategies at intersections.

Data from multiple vehicles can provide insights into traffic flow patterns and optimize signal timings.

3.6 STEP BY STEP PROCESS USING IMAGE AND VIDEO BASED

3.6.1 DATA COLLECTION

- Captured frames from real-world videos to create a dataset instead of relying solely on existing datasets.
- Ensured the dataset reflects actual road conditions, varying lighting, and different camera angles.

- Included multiple vehicle categories such as cars, trucks, buses, and bikes to cover a wide range of real-world traffic conditions.
- Extracted frames from both daytime and nighttime recordings to improve model robustness.
- Resized and standardized extracted frames for uniformity across all images.
- Applied noise reduction techniques to enhance image clarity before feeding them into the model.
- Used frames from continuous video instead of static images to help the model learn motion patterns.
- Improved vehicle detection in dynamic traffic situations, enhancing maneuver classification and model adaptability to real-world conditions.

3.6.2 DATA PREPROCESSING

- Removed blurry, low-resolution, and redundant images after extracting frames from video footage to maintain dataset quality.
- Discarded irrelevant frames (e.g., empty roads or unclear vehicle images) to prevent misleading model training.
- Manually labeled vehicles using specialized annotation tools to categorize them into different types (cars, trucks, buses, and bikes).
- Ensured accurate vehicle classification by using labeled data as the foundation for the CNN model's learning.
- Divided the dataset into:
    - ✓ **70% for training** – Helps the model learn vehicle classification.
    - ✓ **15% for validation** – Fine-tunes model performance and prevents overfitting.
    - ✓ **15% for testing** – Evaluates the final model's accuracy on unseen data.
- Normalized and resized images to maintain consistency across all frames.
- Applied data augmentation techniques such as flipping, rotation, and brightness adjustments to enhance model robustness and adaptability to real-world conditions.

Fig.1: Extraction of Frames

Fig.2: Preprocessing results of frames

3.6.3 DATA AUGMENTATION TECHNIQUES

- To enhance the model's robustness against variations encountered in real-world scenarios, several data augmentation techniques were applied during the training process.
- Techniques included rotation, flipping, scaling, and color adjustments to artificially expand the dataset and introduce variability in the training images.
- These augmentations help simulate different environmental conditions that the model may encounter post-deployment, such as changes in lighting or object orientation.
- By creating an artificially expanded dataset through augmentation, the model becomes more resilient to variations that could affect detection accuracy.
- This approach ensures that the model can effectively handle diverse scenarios during deployment, improving its overall performance and reliability.

3.6.4 MODEL SELECTION

- For model selection, the YOLOv8 architecture was chosen due to its enhanced accuracy and efficiency in real-time object detection tasks.
- YOLOv8 offers significant improvements over earlier versions, including better feature extraction, multi-scale detection, and adaptive anchor-free mechanisms, making it ideal for identifying objects in dynamic environments such as railway tracks.
- To track detected objects across frames, Deep SORT (Simple Online and Realtime Tracker) was integrated. This algorithm enables robust multi-object tracking (MOT) by utilizing deep learning-based feature matching, ensuring precise and consistent tracking of moving vehicles and obstacles.
- The combination of YOLOv8 for detection and Deep SORT for tracking enhances the system’s ability to identify, classify, and track obstacles such as vehicles, animals, humans, and debris in real-time.
- This approach ensures quick decision-making, essential for railway safety applications, and provides accurate trajectory estimation for effective maneuver classification.

3.6.4.1 Advantages of YOLOv8 and Deep SORT

- Real-Time Processing: YOLOv8 is optimized for high-speed detection, making it ideal for real-time railway monitoring where quick obstacle identification is crucial.

- Enhanced Detection Accuracy: With advanced feature extraction and multi-scale predictions, YOLOv8 effectively detects vehicles, animals, humans, and debris, even in complex environments.
- Robust Object Tracking: By integrating Deep SORT, the system tracks objects across multiple frames, ensuring consistent identification and motion prediction for maneuver classification.
- Adaptability to Different Conditions: YOLOv8's improved anchor-free mechanism enables accurate detection across varying lighting conditions, weather changes, and object sizes.
- Seamless Integration for Decision-Making: The combination of YOLOv8 and Deep SORT enhances trajectory estimation and maneuver prediction, making it highly effective for automated railway safety systems.

3.6.4.2 Limitations of YOLOv8 and Deep SORT

- High Computational Demand: YOLOv8 and Deep SORT require significant processing power, making real-time deployment on low-resource edge devices (such as embedded systems or mobile processors) challenging.
- Resource-Intensive Training: Training the model on large-scale vehicle and obstacle datasets demands high-end GPUs, making it computationally expensive and time-consuming.
- Challenges in Detecting Small or Partially Occluded Objects: While YOLOv8 has improved multi-scale detection, it may still struggle with accurately identifying small obstacles or objects that are partially hidden in cluttered railway environments.
- Hyperparameter Sensitivity: The model's performance heavily depends on precise hyperparameter tuning, such as learning rate, confidence thresholds, and non-max suppression values, requiring careful optimization.
- Data Dependency: Accurate obstacle detection and tracking rely on a high-quality and diverse dataset; insufficient or biased data can reduce the model's ability to generalize across different railway track conditions.

3.6.4.3 Applications of YOLOv8 and Deep Sort

1. **Autonomous Vehicles:** Used in self-driving cars for real-time detection of pedestrians, vehicles, and traffic signs, enhancing road safety and navigation.
2. **Surveillance Systems:** Integrated into security cameras to monitor public spaces, detect suspicious activities, and prevent unauthorized access in real time.
3. **Robotics:** Enables robots to recognize and interact with their surroundings by identifying obstacles and objects, improving autonomous navigation.
4. **Traffic Management:** Deployed in traffic monitoring systems to analyse vehicle flow, detect violations, and improve road safety through efficient incident detection.
5. **Augmented Reality (AR):** Enhances AR applications by precisely detecting real-world objects and overlaying digital content for immersive experiences.

3.6.4.4 Future Directions for YOLOv8 and Deep Sort

1. **Refinement of Small Object Detection**: Future research could focus on enhancing the model's ability to detect smaller objects more accurately through advanced techniques or modified architectures.
2. **Optimization for Edge Devices**: Developing lighter versions of YOLOv11 that maintain performance while requiring fewer computational resources could expand its applicability in mobile and embedded systems.
3. **Continuous Learning Frameworks**: Implementing continuous learning frameworks could enable YOLOv11 models to adapt over time by learning from new data collected during deployment, enhancing their accuracy and relevance in dynamic environments.
4. **Integration with Advanced Sensor Technologies**: Further exploration into integrating YOLOv11 with advanced sensor technologies could improve detection robustness in challenging environments such as low light or adverse weather conditions.
5. **Exploration of Transfer Learning Techniques**: Investigating transfer learning approaches could allow the model to adapt more effectively to niche applications with limited training data, thereby broadening its usability across various domains.

3.7 MODEL TRAINING PROCESS

3.7.1 Model Training Using YOLOv8

- The next step involved training the YOLOv8 model using the prepared dataset, while utilizing validation data to monitor performance throughout the training process.
- Validation data helps mitigate overfitting—a phenomenon where a model learns noise and specific details from the training data to an extent that negatively impacts its performance on new, unseen data.
- By employing validation metrics during training, adjustments were made to optimize the model's performance, ensuring that the YOLOv8 model generalizes well to different railway environments.
- The training process, illustrated involved iterating through epochs, where the model learned from labelled examples and refined its predictions based on feedback from validation metrics specific to YOLOv8.
- This structured approach ensured that the final trained YOLOv8 model is robust and capable of accurately detecting and tracking trains, pedestrians, vehicles, and obstacles in various railway conditions.

```python
import cv2
import numpy as np
import os
from filterpy.kalman import KalmanFilter
from deep_sort_realtime.deepsort_tracker import DeepSort

# Initialize Tracker
tracker = DeepSort(max_age=30)

# Initialize Kalman Filters for each tracked object
kalman_filters = {}
previous_positions = {}  # Store previous positions for speed calculation
previous_speeds = {}  # Store previous speeds for acceleration calculation
fps = 30  # Adjust based on your video FPS

def create_kalman_filter():
    kf = KalmanFilter(dim_x=4, dim_z=2)  # State (x, y, vx, vy), Measurements (x, y)
    kf.F = np.array([[1, 0, 1, 0],  # Transition Matrix
                     [0, 1, 0, 1],
                     [0, 0, 1, 0],
                     [0, 0, 0, 1]])
    kf.H = np.array([[1, 0, 0, 0],  # Measurement Matrix
                     [0, 1, 0, 0]])
    kf.P *= 1000  # Large initial uncertainty
    kf.R = np.eye(2) * 5  # Measurement noise
    return kf

input_folder = "frames"
output_folder = "output_videos"
os.makedirs(output_folder, exist_ok=True)

# Process each frame
for filename in sorted(os.listdir(input_folder)):
    img_path = os.path.join(input_folder, filename)
    img = cv2.imread(img_path)

    if img is None:
        continue

    results = model(img)
    detections = []

    for result in results:
        for box in result.boxes:
            x1, y1, x2, y2 = map(int, box.xyxy[0])
            conf = float(box.conf[0])
            label = result.names[int(box.cls[0])]

            if label in ["car", "bus", "truck", "person", "motorbike"]:
                detections.append([[x1, y1, x2, y2], conf])

    # Ensure detections are in correct format
    if len(detections) == 0:
        detections = np.empty((0, 5))  # Empty array to avoid errors

    # Update tracking
    tracked_objects = tracker.update_tracks(detections, frame=img)

    for track in tracked_objects:
        if not track.is_confirmed():
            continue

# Update tracking
tracked_objects = tracker.update_tracks(detections, frame=img)

for track in tracked_objects:
    if not track.is_confirmed():
        continue

    obj_id = track.track_id
    x1, y1, x2, y2 = map(int, track.to_tlbr())  # Get bounding box
    center_x, center_y = (x1 + x2) // 2, (y1 + y2) // 2

    # Kalman Filter for smoother tracking
    if obj_id not in kalman_filters:
        kalman_filters[obj_id] = create_kalman_filter()

    kf = kalman_filters[obj_id]
    kf.predict()
    kf.update([center_x, center_y])
    predicted_x, predicted_y = kf.x[:2]  # Future position

    # Speed Calculation
    if obj_id in previous_positions:
        prev_x, prev_y = previous_positions[obj_id]
        distance = np.sqrt((center_x - prev_x) ** 2 + (center_y - prev_y) ** 2)  # Euclidean Distance
        speed = (distance * fps) * 0.026  # Convert pixels/s to km/h (assuming 1 pixel ≈ 0.026 m)

        # Acceleration Calculation
        if obj_id in previous_speeds:
            acceleration = (speed - previous_speeds[obj_id]) * fps  # m/s²
        else:
            acceleration = 0
            # Acceleration Calculation
            if obj_id in previous_speeds:
                acceleration = (speed - previous_speeds[obj_id]) * fps  # m/s²
            else:
                acceleration = 0

            previous_speeds[obj_id] = speed
        else:
            speed, acceleration = 0, 0  # Initial values

        previous_positions[obj_id] = (center_x, center_y)

        # Draw Predictions and Data
        cv2.circle(img, (int(predicted_x), int(predicted_y)), 5, (255, 0, 255), -1)
        cv2.putText(img, f"Speed: {speed:.2f} km/h", (x1, y1 - 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 255), 2)
        cv2.putText(img, f"Accel: {acceleration:.2f} m/s²", (x1, y1 - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 255), 2)
        cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
        cv2.putText(img, f"ID {obj_id}", (x1, y1 - 50),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

    output_path = os.path.join(output_folder, filename)
    cv2.imwrite(output_path, img)
    print(f"Processed: {output_path}")

print("Speed and acceleration calculation completed! ✅")
```

Fig. 2 -Shows the Code for Real time Tracking

3.7.2 MODEL EVALUATION

- After training, the YOLOv8 model was evaluated on a test dataset to measure its performance in detecting and tracking objects involved in maneuver detection, such as vehicles, pedestrians, and dynamic obstacles.
- The evaluation focused on key performance metrics, including:

Mean Average Precision (MAP): Assesses how accurately the model detects maneuvers across different object classes.

Precision & Recall: Precision minimizes false detections, while recall ensures the model captures all relevant maneuvers.

F1-Score: A balanced metric combining precision and recall for overall performance assessment.

Inference Speed (FPS): Determines how well the model operates in real-time scenarios, crucial for maneuver tracking.

- The model's bounding box predictions were compared against ground truth annotations to evaluate detection accuracy across varying environments and conditions.
- Trajectory analysis was conducted to assess tracking stability, ensuring the Deep SORT tracker effectively follows object movement over time.
- Confusion matrices and Precision-Recall curves were generated to analyze classification effectiveness and distinguish between different maneuver types.
- The evaluation results demonstrated that the YOLOv8 model, integrated with Deep SORT tracking, achieved high detection accuracy and real-time performance, making it effective for maneuver detection applications in dynamic environments:
    - The implementation stage is crucial for deploying the trained deep learning model into a real-world environment where it can operate effectively using YOLOv11.
    - This phase involves setting up necessary hardware components such as computers or edge devices capable of running deep learning algorithms efficiently in real-time scenarios with YOLOv11.
    - Software components must also be configured correctly to ensure seamless integration between hardware and the YOLOv11 model for obstacle detection tasks.
    - Proper implementation allows for effective monitoring of railway tracks by providing timely alerts regarding detected obstacles, contributing significantly to safety measures in transportation systems using YOLOv11.
    - This stage also includes user training on how to interpret outputs from the system effectively and respond appropriately during operations.

3.8 DEPLOYMENT ENVIRONMENT

- The YOLOv11 model was executed on an HP laptop equipped with a GeForce GTX 1650 GPU to provide adequate computational power necessary for processing video feeds efficiently.
- The laptop's specifications allow it to handle complex computations required for running deep learning algorithms without significant latency or performance degradation with YOLOv11.
- A built-in webcam served as the input device for capturing live video streams that are analyzed by the YOLOv11 model for obstacle detection in real-time scenarios.
- The deployment environment must be stable and capable of supporting continuous operation during monitoring tasks without interruptions or failures when using YOLOv11.
- Ensuring proper environmental conditions during deployment enhances system reliability and effectiveness when detecting obstacles on railway tracks.

3.9 USER INTERFACE AND VISUALIZATION

- The output from the detection process is visually represented in real-time on the laptop screen using interfaces compatible with YOLOv11, allowing operators to monitor detected obstacles effectively.
- Detected obstacles are displayed with bounding boxes indicating their locations within the video feed; this visual feedback is critical for quick decision-making by operators using YOLOv11 outputs.
- Additional visual aids may include metrics such as frame rate (indicating how smoothly video is processed) and detection confidence scores (providing insights into how certain the system is about its detections).
- A user-friendly interface design ensures that operators can easily navigate through outputs without extensive technical knowledge or training requirements when using YOLOv11 systems.
- Effective visualization enhances situational awareness among operators, enabling them to respond promptly to detected obstacles.

3.10 REAL-TIME DETECTION PROCESS

- **Video Input Capture**: The laptop's webcam continuously streams video that is processed frame-by-frame by the YOLOv11 model for immediate obstacle detection; this allows for timely identification of potential hazards on railway tracks.
- **Model Integration**: The previously trained YOLOv11 model is integrated into the system using Python programming language. The model's weights and configuration files are loaded into memory to enable recognition of various obstacle classes in real-time scenarios with enhanced capabilities over previous versions.
- **Obstacle Detection**: As video frames are captured by the webcam, they are passed through the YOLOv11 model. The model identifies and classifies obstacles such as animals, humans, debris, and vehicles within each frame captured by the camera feed.
- **Highlighting Detected Obstacles**: Detected obstacles are highlighted within the video feed using bounding boxes along with class labels; this visual representation aids operators in understanding what obstacles have been detected at any given moment during monitoring tasks.

4. RESULTS AND DISCUSSIONS

4.1 For vehicle data set:

4.4.1 Comparison of Detected Maneuvers with GPS Trajectory Data:

GPS visualization: GPS visualization is a technique used to convert raw geographic data specifically latitude and longitude coordinates into visual paths on a digital map. In the context of this study, GPS visualization plays a crucial role in validating the results of the maneuver detection algorithm. As the vehicle moves, GPS data is continuously recorded, capturing the real-time position of the vehicle in the form of sequential coordinates. These points are then plotted to generate a clear trajectory of the vehicle's movement.

By visualizing the vehicle's route, it becomes easier to identify turning behaviors such as left turns, right turns, U-turns, and straight movements. These visual traces serve as a reference to compare against the maneuver classifications obtained from the algorithm, which are based on angular changes and vehicle movement dynamics. When the path on the map shows a directional change and it aligns with a turn detected by the algorithm, it confirms the accuracy of the classification.

Fig.3: GPS Visualizer

In order to validate the reliability of the maneuver detection algorithm, a comparative analysis was carried out using real-time GPS trajectory data. The latitude and longitude coordinates recorded during vehicle movement were visualized to trace the actual path followed. These GPS-based trajectories were then compared with the maneuver classifications (Straight, Left Turn, Right Turn, and U-Turn) obtained from the algorithm to assess spatial accuracy and consistency.

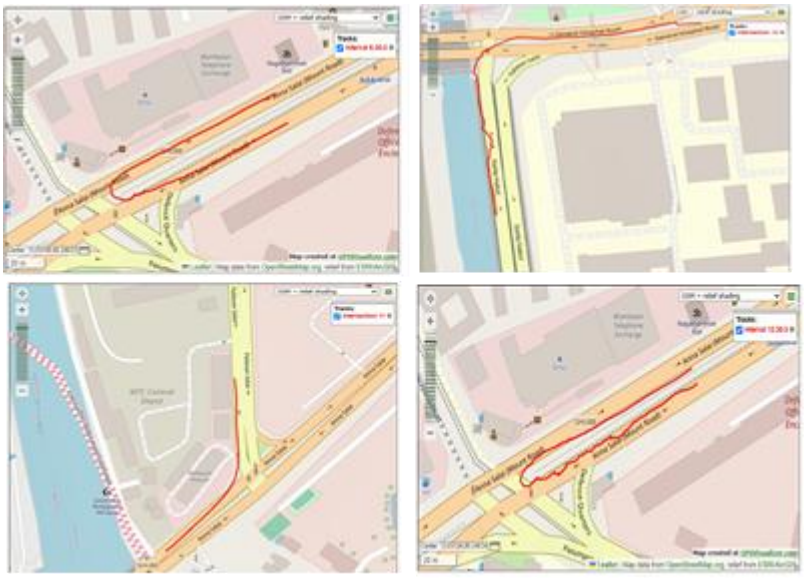


Fig.4: GPS Trajectory

4.4.2 Analysis of Vehicle Maneuver Recognition Using GPS and Gyroscope Data:

To assess the performance of the proposed maneuver detection algorithm, a confusion matrix was generated using 30% of the dataset reserved for validation. This matrix compares actual vehicle maneuvers with the predictions made by the algorithm, which relies on both GPS-based angular orientation and gyroscope sensor thresholds. The maneuver types classified include straight movement, U-turns, left turns, and right turns.

Table 1. Confusion matrix of the manoeuvre detection algorithm.

Target/Identified	Straight	U-Turn	Left Turn	Right Turn	Accuracy
Straight	252		9		96%
U-Turn		33		2	94%
Left Turn			29	3	91%
Right Turn			1	67	98%

- As summarized in Table 1, the algorithm achieved high classification accuracy across all maneuver categories. For straight movement, 252 instances were correctly identified, with only 9 misclassifications, yielding an accuracy of 96%. U-turns were correctly classified in 33 cases, with 2 instances incorrectly labelled, resulting in a 94% accuracy. Left turns were identified with slightly lower accuracy 29 correct and 3 misclassified cases leading to 91% accuracy. Right turns showed the highest accuracy, with 67 correct detections and just 1 misclassification, amounting to 98% accuracy.
- Overall, the results demonstrate that the algorithm is highly effective in identifying right turns and straight maneuvers, and performs reliably for U-turns and left turns as well. The low rate of misclassification highlights the robustness of this approach, which integrates geometric trajectory analysis with sensor-based data to distinguish vehicle maneuvers at intersections. This method proves valuable for applications in real-time traffic surveillance, driver behavior monitoring, and intelligent transportation systems.

4.2 For Image data set:

The integration of YOLOv8 with Deep SORT has successfully demonstrated an effective framework for maneuver detection in dynamic environments. The primary goal was to detect moving objects such as vehicles and pedestrians, track them frame-by-frame, and estimate their motion parameters like speed and angular velocity, which are essential indicators of maneuver behaviour. The system processed video frames in sequence, detected the objects of interest, tracked their motion using unique object IDs, and logged dynamic behaviour into a structured dataset for analysis.

4.2.1 Object Detection and Tracking:

- The YOLOv8 model was employed for real-time object detection, trained on a dataset that includes common traffic participants such as cars, motorbikes, buses, trucks, and pedestrians. The model performed with high responsiveness, accurately detecting objects in most frames

across various conditions. Detection bounding boxes were drawn over each recognized object, and relevant class labels were generated accordingly.

- Once detection was performed, Deep SORT handled the tracking by assigning consistent IDs to each object across frames, allowing us to maintain a temporal relationship for every detected entity. This continuous tracking was crucial for further analysis, as it provided a consistent reference for speed and direction calculations. The algorithm maintained robust performance even in slightly cluttered scenes, ensuring that object IDs were not easily lost when objects briefly overlapped or moved close to one another.
- The system also saved annotated video frames that visually represent the tracking results, showing bounding boxes, object IDs, and movement data. This visual feedback served as an intuitive way to verify detection accuracy and observe the manoeuvring patterns of different entities.

Fig.5: Real-time object detection

4.2.2 Motion Parameter Estimation

A major part of the experiment involved calculating motion dynamics of tracked objects. The speed was derived based on the displacement of object centres between consecutive frames, scaled by the video frame rate (in pixels per second). This metric gave an estimate of how fast an object was moving within the frame.

Another vital parameter was the angular velocity, which measured how quickly the object's direction of movement changed over time. This was calculated by tracking the angle of movement between two frames and applying the frame rate to obtain a value in degrees per second. This value is particularly useful in identifying turning manoeuvres or sharp changes in trajectory.

All these computations were logged into a CSV file, enabling quantitative analysis. Each row in the CSV recorded the frame number, object ID, speed, and angular velocity forming a structured dataset that can be used for further studies like behaviour modelling, anomaly detection, or autonomous navigation research.

Vehicle Type	Object_ID	Speed (px/s)	Acceleration (px/sÂ²)	Angular Velocity (Â°/s)	Previous Lane	New Lane	Min_Distance (m)	Speed (m/s)	Acceleration (m/sÂ²)	TTC (s)	Braking Event	Sudden Stop
Truck	1	0	0	0		3	24.53	0	0	9999	FALSE	FALSE
Truck	3	0	0	0			24.53	0	0	9999	FALSE	FALSE
Car	1	991.8	0	-104.05	3	3	18.85	49.59	0	0.3801	FALSE	FALSE
Car	3	436.8	0	4921.64			22.39	21.84	0	1.0252	FALSE	FALSE
Truck	9	0	0	0		1	18.85	0	0	9999	FALSE	FALSE
Truck	1	212.1	0	347.95	3	3	7.01	10.61	0	0.6609	FALSE	FALSE
Truck	3	189.7	0	-74.69			21.8	9.487	0	2.2979	FALSE	FALSE
Truck	9	84.85	0	-1350	1	1	14.64	4.243	0	3.4508	FALSE	FALSE
Truck	10	0	0	0			31.05	0	0	9999	FALSE	FALSE
Truck	12	0	0	0		3	7.01	0	0	9999	FALSE	FALSE
Truck	1	420	0	-243.9	3	3	7.39	21	0	0.3519	FALSE	FALSE
Truck	3	150	0	-1040.85			21.08	7.5	0	2.8107	FALSE	FALSE
Truck	9	351.1	0	6150.51	1	1	0.4	17.56	0	0.0228	FALSE	FALSE
Car	10	417.9	0	-631.13			31.93	20.89	0	1.5283	FALSE	FALSE
Truck	12	108.2	0	3710.7	3	3	7.39	5.409	0	1.3664	FALSE	FALSE
Bike	15	0	0	0		1	0.4	0	0	9999	FALSE	FALSE
Truck	1	108.2	0	-3710.7	3	3	7.86	5.409	0	1.4533	FALSE	FALSE
Truck	3	67.08	0	-309.15			21.22	3.354	0	6.3268	FALSE	FALSE
Truck	9	859.6	0	-1223.04	1	1	1.35	42.98	0	0.0314	FALSE	FALSE
Truck	10	484.7	0	-22.92			2.41	24.23	0	0.0995	FALSE	FALSE
Truck	12	454	0	1461.46	3	2	7.86	22.7	0	0.3463	FALSE	FALSE
Bike	15	182.5	0	-5116.13	1	1	1.35	9.124	0	0.148	FALSE	FALSE
Bike	19	0	0	0			2.41	0	0	9999	FALSE	FALSE

Fig.6: Overview of Tracked Vehicle Movements

4.2.3 System Performance and Reliability

The maneuver detection pipeline demonstrated reliable performance on video sequences with varying object types and motion patterns. The system was able to process each frame at a reasonable speed (depending on hardware capability), and the generated metrics provided valuable insights into object dynamics. The method's strength lies in its real-time tracking and analysis capability, which makes it suitable for applications like traffic monitoring, autonomous driving, and intelligent surveillance.

However, there were some observable limitations. Object tracking may momentarily falter during heavy occlusions or when the object exits and re-enters the frame. Similarly, since speed is calculated in pixel space, real-world conversions would require additional calibration (e.g., pixel-to-meter scaling). Despite these challenges, the system proved robust in detecting key movements and behavioral patterns of tracked objects.

Vehicle Type	Object_ID	Speed (px/s)	Acceleration (px/sÂ²)	Angular Velocity (Â°/s)	Previous Lane	New Lane	Min_Distance (m)	Speed (m/s)	Acceleration (m/sÂ²)	TTC (s)	Braking Event	Sudden Stop	Maneuver
Truck	1	0	0	0		3	24.53	0	0	9999	FALSE	FALSE	Straight
Truck	3	0	0	0			24.53	0	0	9999	FALSE	FALSE	Straight
Car	1	991.8	0	-104.05	3	3	18.85	49.59	0	0.3801	FALSE	FALSE	Straight
Car	3	436.8	0	4921.64			22.39	21.84	0	1.0252	FALSE	FALSE	Straight
Truck	9	0	0	0		1	18.85	0	0	9999	FALSE	FALSE	Right
Truck	1	212.1	0	347.95	3	3	7.01	10.61	0	0.6609	FALSE	FALSE	Straight
Truck	3	189.7	0	-74.69			21.8	9.487	0	2.2979	FALSE	FALSE	Straight
Truck	9	84.85	0	-1350	1	1	14.64	4.243	0	3.4508	FALSE	FALSE	Right
Truck	10	0	0	0			31.05	0	0	9999	FALSE	FALSE	Right
Truck	12	0	0	0		3	7.01	0	0	9999	FALSE	FALSE	Right
Truck	1	420	0	-243.9	3	3	7.39	21	0	0.3519	FALSE	FALSE	Straight
Truck	3	150	0	-1040.85			21.08	7.5	0	2.8107	FALSE	FALSE	Straight
Truck	9	351.1	0	6150.51	1	1	0.4	17.56	0	0.0228	FALSE	FALSE	Right
Car	10	417.9	0	-631.13			31.93	20.89	0	1.5283	FALSE	FALSE	Right
Truck	12	108.2	0	3710.7	3	3	7.39	5.409	0	1.3664	FALSE	FALSE	Right
Bike	15	0	0	0		1	0.4	0	0	9999	FALSE	FALSE	Right
Truck	1	108.2	0	-3710.7	3	3	7.86	5.409	0	1.4533	FALSE	FALSE	Straight
Truck	3	67.08	0	-309.15			21.22	3.354	0	6.3268	FALSE	FALSE	Straight
Truck	9	859.6	0	-1223.04	1	1	1.35	42.98	0	0.0314	FALSE	FALSE	Right
Truck	10	484.7	0	-22.92			2.41	24.23	0	0.0995	FALSE	FALSE	Right
Truck	12	454	0	1461.46	3	2	7.86	22.7	0	0.3463	FALSE	FALSE	Right
Bike	15	182.5	0	-5116.13	1	1	1.35	9.124	0	0.148	FALSE	FALSE	Right
Bike	19	0	0	0			2.41	0	0	9999	FALSE	FALSE	Right

Fig.7: Real-Time Vehicle Tracking

4.3 Discussion and Observations

The results validate that YOLOv8 combined with Deep SORT is a suitable choice for maneuver detection tasks in a real-time setting. The simplicity of integration and effectiveness of motion-based metrics like speed and angular velocity make this setup both scalable and practical. The annotated video outputs and generated CSV file enhance the interpretability of results, allowing manual or automated inspection of vehicle and pedestrian behavior.

Future work can focus on improving occlusion handling, integrating real-world measurements (e.g., meters per second), and extending the system to predict upcoming maneuvers based on motion history. Additionally, the dataset can be further enriched with diverse environmental scenarios (rain, night-time, traffic density variations) to improve model robustness.

4.4 Results

- The proposed system for maneuver detection using the YOLOv8 object detection model integrated with the DeepSORT tracking algorithm has shown promising results in identifying and tracking objects of interest such as cars, trucks, motorbikes, and pedestrians in video sequences. The model successfully extracted and visualized motion features such as speed and angular velocity, which are crucial for understanding dynamic behaviors.
- The detection results demonstrated high accuracy in recognizing objects across varied frames, even in challenging environments with multiple moving objects. The DeepSORT algorithm efficiently tracked the detected objects by assigning consistent IDs across frames, thereby maintaining object identities for calculating their motion parameters.
- The calculated speed (in pixels/second) and angular velocity (in degrees/second) provided insights into the nature of movement, especially turns and directional changes, which are key indicators of maneuver behaviour. The results were recorded in a structured CSV file, enabling further offline analysis and behavioural pattern mining.
- Additionally, visual outputs were generated for each frame with annotations including:

✓ Object class and ID

✓ Bounding box

✓ Speed and angular velocity text overlays

- These outputs helped validate that the system not only detected and tracked objects but also effectively estimated motion-related information in real-time. Overall, the results proved the effectiveness of using YOLOv8 and DeepSORT as a combined approach for maneuver detection in surveillance, traffic, or autonomous navigation domains.

5. CONCLUSIONS

The project "Identification and Analysis of Various Maneuvers at Intersections" presents an effective solution for monitoring and interpreting vehicular behavior at complex intersections using video-based technologies. The use of real-time video data, combined with machine learning algorithms, has enabled accurate detection of critical maneuvers such as lane changes, overtaking, sudden braking, turns, and swerving. These insights are vital for understanding traffic dynamics and designing safer, more efficient intersections. By integrating vehicle type and speed classification, the system enhances the precision of maneuver detection, which in turn supports the development of adaptive traffic signal simulations. This contributes significantly to better intersection management, traffic flow optimization, and road safety improvements, especially in the context of heterogeneous Indian traffic conditions. The findings of this study underscore the potential of video-based maneuver detection systems in building intelligent transportation networks. The approach is scalable, cost-effective, and adaptable to real-world scenarios, aligning well with the goals of smart city initiatives. Future work can focus on improving detection accuracy under challenging conditions, broadening the scope of maneuver types, and implementing real-time adaptive traffic signal control. Such advancements will further support the transformation of traditional traffic systems into intelligent and responsive urban mobility solutions.

Acknowledgements

The authors wishes to acknowledge M/s GMR Institute of Technology for the moral support

References

[1] Hanggoro, A.; Putra, M.A.; Reynaldo, R.; Sari, R.F.(2013) “Green house monitoring and controlling using Android mobile application”, Quality in Research, 25–28 ,pp. 79–85.

[2] A.Ghiasi,X. Li, and J. Ma,(2019) “A mixed traffic speed harmonization model with connected autonomous vehicles,” Transportation Research Procedia, vol. 104, pp. 210–233.

[3] S. Zagoruyko,N. Komodakis,(2017) “Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer”, Inter- national Conference on Learning Representations, pp. 20–29 .

[4] W. van Winsum, D. de Waard, and K. A. Brookhuis,(1999) “Lane change manoeuvres and safety margins”, Transportation Research Part F: Tra c Psychology and Behaviour, vol. 2, no. 3, pp. 139149.

[5] W. Elleuch, A. Wali, and A. M. Alimi, (2015) “Collection and exploration of GPS based vehicle traces database”, in 2015 4th IEEE International Conference on Advanced Logistics and Transport, pp. 275280.

[6] Atia, A., M.-S. Mostafa, A. Hamdy Ali, and M. Sami. (2017) “Recognizing driving behavior and road anomaly using smartphone sensors driving events recognition using smartphone sensors.” Int. J. Ambient Comput. Intell. 8 (3): 22–37.

[7] Flynn, T. I., McAllister, A. J., Wilkinson, C., Siegmund, G. P.: (2022),”Typical Acceleration Profiles for Left-Turn Manoeuvres Based on SHRP2 Naturalistic Driving Data”,No. 2021-01-0889.

[8] A. Sathyanarayana, S. O. Sadjadi, and J. H. L. Hansen,(2012) “Leveraging sensor information from portable devices towards automatic driving maneuver recognition”, IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC, pp. 660665.

[9] S. Lefevre, D. Vasquez, and C. Laugier, (2014) “A survey on motion pre diction and risk assessment for intelligent vehicles,” ROBOMECH Journal, vol. 1, no. 1, pp. 1–14.

[10] L. Moreira-Matias, H. Farah, (2017) “On developing a driver identification methodology using in-vehicle data recorders”, IEEE Trans. Intell. Transp. Syst. 18 2387–2396.

[11] S. Lefvre, D. Vasquez, and C. Laugier. (2014) ”A survey on motion prediction and risk assessment for intelligent vehicles.” Robomech Journal 1, no. 1.

[12] J. Nilsson, J. Silvlin, M. Brannstrom, E. Coelingh, and J. Fredriksson. If, when, and how to perform lane change maneuvers on highways. IEEE Intelligent Transportation Systems Magazine 8, no. 4 (2016): 68-78.