A purple rectangular sign with white text

Description automatically generated

*A Synopsis on*

***Vehicle Cut-in Detection Using IDD***

***Submitted for the Intel Unnati Industrial Training Program 2024***

***Team***

**Sai Deepak M (1NT22EC144)**

**Tushar DM (1NT22EC178)**

**Shivasai Lahari Mulakala (1NT22EC155)**

**Syed Ibraz Hussain (1NT22EC171)**

Under the Guidance of

**Dr. Rajesh N**

Professor

Dept. of Electronics and Communication Engineering

Blue text on a white background

Description automatically generated

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**YELAHANKA, BENGALURU- 560064**

Contents

[CHAPTER 1 :Introduction 3](#_Toc171185927)

[CHAPTER 2 : Literature Survey 5](#_Toc171185928)

[CHAPTER 3 : Motivation and Objectives 7](#_Toc171185929)

[CHAPTER 4 Architecture of the Proposed System 8](#_Toc171185930)

[CHAPTER 5 Hardware/Software Description 8](#_Toc171185931)

[CHAPTER 6 Design Specification/Dataset Description 9](#_Toc171185932)

[CHAPTER 7 Expected Outcomes 9](#_Toc171185933)

# : Introduction

Imagine you're driving down the highway, cruising along in your lane, when suddenly another vehicle cuts you off, leaving you mere seconds to react. This heart-stopping scenario plays out on roads every day, highlighting the crucial need for advanced safety features in our vehicles. That's where vehicle cut-in detection comes in - a vital component of advanced driver-assistance systems (ADAS) that helps cars anticipate and respond to unexpected intrusions.

To make our roads safer, the current detection systems have limitations, often relying on faulty sensor data that can lead to accidents. That's why we've taken a different approach, using the Indian Driver Dataset (IDD) and machine learning to create a more comprehensive and reliable detection system. By harnessing the power of Python and OpenCV, we've developed a model that can efficiently process and analyse visual data, making it a game-changer for road safety. Since the Indian driver’s dataset has the drawback of consisting only images where it is difficult for one to know the speed of both the reference and the incoming vehicle, hence being unable to accurately gauge the relative speed, due to which calculating TTC becomes a problem, and the accurate distance can vary depending on the reference vehicle size too.

In this report, we'll dive into the nitty-gritty of our research, exploring the methodology, results, and conclusions we've drawn. We'll examine how our trained model performs, discussing its accuracy and reliability in detecting vehicle cut-ins. And, we'll show how our findings can revolutionize the development of advanced driver-assistance systems, leading to fewer accidents and safer roads for everyone.

Detection of sudden cut-in of vehicles (two/three/four wheelers, pushcarts etc) into the path line of the reference car is the main motive of this problem statement hence, we have classified the IDD dataset into 3 primary classes namely for cut-in, collision, and safe with respect to the reference vehicle. In the current trend, the automation for avoidance of collisions relies on a combination of vehicle detection, distance estimation, and time-to-collision (TTC) calculation (using vehicle speed data from GPS or Engine). But this current technology has drawbacks like, the vehicle needs to be in front of the driver to be detected and other one is vehicles that abruptly cut into the driver’s path typically are not considered for collision avoidance. So, we have trained a working machine learning model which clears these drawbacks.

The Indian Driving Dataset (IDD) is an extensive dataset designed for autonomous driving research, particularly suited to the unique and complex driving conditions found in Indian roads. Utilizing the IDD dataset, our project aims to enhance the capabilities of machine learning models in detecting and tagging objects that appear either partially or fully in front of a driver. By incorporating additional data from various sources to enlarge the training process, we strive to improve the accuracy and robustness of these models. The primary objectives include estimating the distance and Time-To-Collision (TTC) for detected objects and providing timely warnings when the TTC falls below a predefined threshold of 0.5 to 0.7 seconds. This proactive approach aims to significantly enhance driver safety by enabling early detection and warning of potential collisions.

We have harnessed the powerful ability of python jupyter that’s simple yet functional That's what Python Jupyter offers - an incredible tool that's revolutionizing the way we work with data. With Jupyter Notebook, we got the perfect blend of code execution, visualization, and documentation, making it it possible to train our model efficiently. In our project, Python Jupyter is the ultimate wingman, providing a flexible and intuitive interface for data analysis, visualization, and machine learning tasks.

We also made use of OpenCV, the rockstar of computer vision libraries. With its vast array of functions and tools, OpenCV to help us tap into the power of image and video data . From filtering and feature detection to object recognition and tracking, In our vehicle cut-in detection system, OpenCV plays a starring role, helping us process visual data, and identify vehicles with ease that boxes each vehicle individually and checks its status with respect to the classes we have defined. And when we combine OpenCV's superpowers with Python Jupyter's flexibility, we get a toolkit that's unstoppable.

# : Literature Survey

This literature survey reviews research on detecting when a vehicle suddenly cuts into another vehicle's lane, focusing especially on studies that use the IDD (Indian Driving Dataset) dataset. Detecting these cut-ins is crucial for making self-driving cars and advanced driver assistance systems safer. The IDD dataset is valuable because it shows the unique and complex driving conditions found on Indian roads, which helps in developing and testing detection methods. This survey gathers current research findings, looks at different detection methods.

Vehicle cut in detection consists of two parts, vehicle detection and vehicle tracking. The detection can be done using density histogram of a differential area. In the vehicle detecting part, the system identifies a vehicle in the adjacent lane by analysing the input image and storing it as a template pattern when detected. In the vehicle tracking part, the system tracks the detected vehicle by comparing the template pattern with new images to determine the vehicle's position, providing this information to the ACC system. The system registers the rear image of the vehicle as the template image for tracking. By capturing the image just before the density changes from high to low as the vehicle passes the detection area, it ensures accurate tracking by providing sufficient features for the template pattern. This method helps in outputting the nearest position of the vehicle to the ACC system.

The performance of the proposed cut-in warning system was evaluated using various indicators and experimental data. The evaluation included the true and false warning rates of the cut-in events, as well as the latency between the cut-in event and the correct warning. The evaluation process involved manually inputting a cut-in ground truth that recorded the start and end frame numbers of cut-in events for each video. The GT consisted of three sections: the non-event, the cut-in, and the 'don't care' zone, which was difficult to determine whether it was a cut-in event or not. The system's performance was assessed based on the number of true positives and false positives in the cut-in events, as well as the latency of the warnings. The evaluation also considered the robustness of the proposed method, with a high true warning rate of about 98.6% and a low false warning rate. Additionally, the average number of frames elapsed until a warning was triggered after a cut-in occurred was measured, demonstrating a fast response performance.

Traditional vehicle detection methods like Histogram of Oriented Gradient (HOG) and Haar-like features were commonly used before the era of Convolutional Neural Networks (CNNs). Cascaded detectors, Deformable part-based models (DPM), and Support Vector Machines (SVM) were prevalent approaches. However, CNNs revolutionized object detection in 2012, inspired by the brain's visual cortex. These multilayered networks process data from input to output, with deep CNNs having multiple hidden layers for better generalization.

The system aims to enable drivers to perceive dangerous situations earlier to avoid fatal accidents, utilizing vision-based perception like human visual perception. The system runs on the TMS320DM642 DSP of Texas Instruments Corporation, offering processing capabilities of up to 4800 million instructions per second (MIPS) at a clock rate of 600 MHz’s. The system is designed to detect the precise location of the host vehicle on the highway, recognize preceding vehicles, and measure their distance and velocity to prevent collisions.

Test results show an average processing time of 42 ms and a recognition rate of 98%, demonstrating the system's potential for improved performance and future application in autonomous driving systems.

Overall, the evaluation process involved analysing the system's ability to accurately detect cut-in events, minimize false warnings, and provide timely warnings, showcasing the effectiveness and reliability of the proposed cut-in warning system.

# : Motivation and Objectives

**3.1 Motivation**

* Accident Prevention: Sudden vehicle cut ins can cause accidents. Early detection allows for proactive measures like alerting drivers or activating autonomous braking, reducing collision risks.
* Improved Driver Assistance: ADAS can use accurate cut-in detection to provide timely warnings, enhancing driver awareness and safety.
* Improved Navigation: Precise cut-in detection helps autonomous vehicles navigate dynamic traffic, improving overall safety and decision-making.

**3.2 Objectives of the Proposed Project**

* A working ML model for detecting cut-in.
* 99%+ true positives for cut-in detection.
* Detect and tag objects as soon as they appear partially or fully in front of the driver.
* Estimate distance and TTC.
* Provide a warning when the TTC is below a predefined threshold.
* Recommended 0.5-0.7 secs.

# Architecture of the Proposed System

Data Collection and Preprocessing

Model Selection

Model fitting.

Model evaluation.

Testing Model

1. Data Collection and Preprocessing:

Data exploration in IDD datasets involves acquiring images of vehicles. Annotating the dataset is crucial to label instances where vehicles perform cut-ins, ensuring clarity and consistency in data interpretation. Cleaning the data includes removing irrelevant or corrupted entries, and standardizing formats to facilitate further analysis. Exploration allows understanding the dataset's structure and distribution, aiding in informed preprocessing decisions. Preprocessing steps may include normalizing timestamps, converting coordinates into a standard format, and handling missing or erroneous data points. This phase ensures that subsequent analytical processes are built on a reliable foundation, minimizing errors and optimizing computational efficiency.

Additionally, exploring metadata like vehicle types, speeds, and environmental conditions provides contextual insights that can influence cut-in detection algorithms. Overall, thorough data exploration and preprocessing are foundational to extracting meaningful patterns and building accurate models for vehicle cut-in detection in IDD datasets.

2. Model Selection:

Machine learning models in vehicle cut-in detection utilize annotated data to train classifiers capable of distinguishing cut-in events from normal traffic behaviour. Supervised learning approaches, including support vector machines (SVMs) and neural networks, process extracted features such as vehicle trajectories, speeds, and relative positions. Feature engineering techniques enhance model performance by focusing on relevant attributes that differentiate cut-in events. Iterative model refinement through cross-validation ensures generalizability and robustness across diverse driving scenarios.

Additionally, incorporating deep learning architectures such as convolutional neural networks (CNNs) can capture complex patterns in video and motion data, further improving detection accuracy. Machine learning models play a critical role in automating cut-in detection processes, enabling proactive safety measures and enhancing the efficiency of intelligent transportation systems.

3. Model Fitting:

Model fitting for vehicle cut-in detection involves selecting appropriate machine learning algorithms such as Support Vector Machines (SVMs), Random Forests, or neural networks based on their suitability for handling complex vehicle trajectory and image data. The process begins with feature engineering, extracting relevant features like vehicle positions, velocities, accelerations, and distances between vehicles from IDD datasets. After preprocessing the data to handle missing values and normalize features, the dataset is split into training and testing sets. Hyperparameter tuning optimizes algorithm settings using techniques like grid search, while model training involves feeding the training data into the chosen algorithm to learn patterns that differentiate cut-in events from normal driving behaviour.

4. Model Testing:

Testing of vehicle cut-in detection models involve assessing their performance against established metrics such as precision, recall, F1-score, and accuracy. This process ensures that models effectively distinguish between cut-in events and false positives or negatives in real-world scenarios. Cross-validation techniques validate model generalizability by testing performance across different subsets of annotated data. Error analysis identifies common misclassifications or biases, guiding iterative improvements in model design and training strategies. Statistical significance testing validates the reliability of observed performance metrics, supporting confident deployment in safety-critical applications.

5. Model Evaluation:

Model evaluation for vehicle cut-in detection using IDD datasets is critical to ensure the accuracy and reliability of algorithms in real-world scenarios. Several key metrics are used for evaluation involves Precision measures the proportion of correctly identified cut-ins among all detected cut-ins, while Recall assesses the proportion of actual cut-ins that are correctly identified by the model. The F1-score, a harmonic mean of precision and recall, provides a balanced measure of overall model performance.

By systematically applying these evaluation methodologies, developers can enhance the accuracy, reliability, and safety of vehicle cut-in detection systems, contributing to improved performance in intelligent transportation systems.

# Hardware/Software Description

For the vehicle cut-in detection using the IDD dataset problem statement for the Intel industrial training, the hardware and software used and required are as follows:

Hardware:

1. Computer System: A computer system with sufficient processing power and memory to handle the dataset and run the necessary algorithms.
2. Graphics Processing Unit (GPU): A powerful GPU to accelerate the processing of large amounts of data and complex algorithms efficiently.
3. Storage: Sufficient storage space to store the IDD dataset and the trained models.

Software:

1. Python: A programming language commonly used for machine learning and computer vision tasks.
2. OpenCV: An open-source computer vision and machine learning software library used for image and video processing.
3. TensorFlow or PyTorch: Deep learning frameworks for building and training neural networks for object detection tasks.
4. IDE (Integrated Development Environment): Software like Jupyter Notebook, PyCharm, or Visual Studio Code for coding, testing, and debugging the algorithms.
5. Data Processing Libraries: Libraries like NumPy, Pandas for data manipulation, and scikit-learn for machine learning tasks.
6. Annotation Tools: Software tools for annotating the IDD dataset for training the model.

# Design Specification/Dataset Description

Dataset Description: IDD is a novel dataset for road scene understanding in unstructured environments. It consists of 10,000 images, finely annotated with 34 classes collected from 182 drive sequences on Indian roads. The dataset consists of images obtained from a front facing camera attached to a car. The car was driven around Hyderabad, Bangalore cities and their outskirts. The images are mostly of 1080p resolution, but there is also some images with 720p and other resolutions.

# Expected Outcomes

* Train a new ML model for detecting cut-in.
* Use any extra data from other sources to augment training ML models.
* Calculate the accuracy of performance in detection.

**References**

1. Hiroto Morizane, Hiroshi Takenaga, Yoshiki Kobayashi and Kouzou Nakamural, “Cut-In Vehicle Recognition System”, 1999 IEEE, pp.976-980
2. Kunfeng Wang, Zhenjiang Li, Yuan Sun, Xin Qiao, and Fei-Yue Wang, Fellow, “An Embedded System for Vision-based Driving Environment Perception”, IEEE
3. Kyoungtaek Choi, Ho Gi Jung, “Cut-in vehicle warning system exploiting multiple rotational images of SVM cameras”, 2019 Elsevier Ltd., pp.81-89
4. Sahana Punagin, Nalini Iyer “Vehicle detection on unstructured roads based on Transfer learning”, Creative Commons Attribution 4.0 International License, September 8th, 2022, pp.1-13