REAL TIME VEHICLE DETECTION AND COUNTING USING ML

A Project Report submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY

ANANTAPUR, ANANTAPURAMU

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

M.HARISH BABU	**(20HR1A05C7)**
S.K.REHMAN	**(20HR1A0598)**
M.MANJULA	**(20HR1A0572)**
P.PRAVEEN	**(20HR1A0582)**
P.TEJASWINI	**(20HR1A0584)**

Under the esteemed guidance of

Mr. N.V.Srinivasan, M.E

Assistant Professor, CSE Department

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MOTHER THERESA INSTITUTE OF ENGINEERING AND TECHNOLOGY

Melumoi (Post), Palamaner-517408.

Approved by AICTE, New Delhi and Affiliated to JNTUA, Anantapuramu-515002

NAAC Accredited and An ISO 9001:2015 Certified Institution

2020-2024

MOTHER THERESA INSTITUTE OF ENGINEERING AND TECHNOLOGY

AN ISO 9001:2015 CERTIFIED INSTITUTION

(Approved by AICTE, New Delhi and Affiliated to J.N.T.U.A., Anantapuramu)

Melumoi (Post), Palamaner-517 408, Chitoor (Dist), A.P.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Certificate

This is to certify that the Project (20A05801) Report entitled

REAL TIME VEHICLE DETECTION AND COUNTING USING ML

is the bonafide work done and

Submitted by

M.HARISH BABU	**(20HR1A05C7)**
S.K.REHMAN	**(20HR1A0598)**
M.MANJULA	**(20HR1A0572)**
P.PRAVEEN	**(20HR1A0582)**
P.TEJASWINI	**(20HR1A0584)**

In the Department of Computer Science and Engineering, Mother Theresa Institute of Engineering And Technology, Palamaner affiliated to J.N.T.U.A., Anantapuramu in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science & Engineering during 2023-2024.

Submitted on: ________

Internal Guide	**HOD**
Mr.N.V.Srinivasan, M.E	Dr.P.C.Prabhu Kumar, M.E, Ph.D.,
Assistant Professor	Associate Professor
Internnal Examinar	**External Examinnar**

ACKNOWLEDGEMENTS

Any achievement, be it scholastic or otherwise does not depend solely on the individual effortbut on the guidance, encouragement and cooperation of intellectuals, elders and friends. We would like to take this opportunity to thank them all.

We feel ourselves honoured for placing our warm salutation to **THE MANAGEMENT** Mother Theresa Institute of Engineering and Technology, Palamaner, which gave us the opportunity to obtain a strong base in B. Tech and profound knowledge.

We express our sincere thanks to **Dr. M.LAKSHMIKANTHA REDDY, M.Tech, Ph.D,** our beloved Principal for his encouragement and suggestions during our course of study.

With deep sense of gratitude we acknowledge **Dr. P.C.PRABHU KUMAR, M.E, Ph.D** Head of the Dept., Computer Science& Engineering, for his valuable support and help in processing our Socially Relevant Project.

We also express sincere thanks to our Socially Relevant Project Coordinator **Mr. J.NAVEEN KUMAR, M.Tech,** Assistant Professor in Department of Computer Science and Engineering, for encouraging us in doing this Socially Relevant Project.

We whole-heartedly express gratitude and regards to our guide **Mr. N.V.SRINIVASAN, M.E,** Assistant Professor in Department of Computer Science and Engineering, for constant support and encouragement.

Finally, we would like to express our sincere thanks to **Faculty Members** of C.S.E Department, and Lab Technicians, Friends & Family members, who all have motivated and helped usto do this Project.

M.HARISH BABU	**(20HR1A05C7)**
S.K.REHMAN	**(20HR1A0598)**
M.MANJULA	**(20HR1A0572)**
P.PRAVEEN	**(20HR1A0582)**
P.TEJASWINI	**(20HR1A0584)**

ABSTRACT

Traffic Analysis has been a problem that city planners have dealt with for years. Smarter ways are being developed to analyze traffic and streamline the process. Analysis of traffic may account for the number of vehicles in an area per some arbitrary time period and the class of vehicles. People have designed such mechanism for decades now but most of them involve use of sensors to detect the vehicles i.e. a couple of proximity sensors to calculate the direction of the moving vehicle and to keep the vehicle count. Even though over the time these systems have matured and are highly effective, they are not very budget friendly. The problem is such systems require maintenance and periodic calibration. Therefore, this study has purposed a vision based vehicle counting and classification system. The system involves capturing of frames from the video to perform background subtraction in order detect and count the vehicles using Gaussian Mixture Model (GMM) background subtraction then it classifies the vehicles by comparing the contour areas to the assumed values. The substantial contribution of the work is the comparison of two classification methods. Classification has been implemented using Contour Comparison (CC) as well as Bag of Features (BoF).method.

KEYWORDS: Machine learning, Video surveillance, detection, video classification, Gaussian Mixture Model, Bag of Features, Support Vector Machine.

CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

S.No	ACRONYM	DESCRIPTION
1	ITS	Intelligent Transportation Systems
2	SVM	Support Vector Machine
3	CV	Computer Vision
4	KNN	K Nearest Neighbors
5	GMM	Gaussian Mixture Model
6	SIFT	Scale Invariant Feature Transform
7	IMU	Inertial Measurement Unit
8	URL	Uniform Resource Locator
9	IDE	Integrated Development Environment
10	SCM	Source Control Management
11	OCR	Optical Character Recognition
12	NLP	Natural Language Processing
13	PNG	Portable Network Graphics
14	SVG	Scalable Vector Graphics
15	XML	Extensible Markup Language
16	JSON	JavaScript Object Notation
17	JPEG	Joint Photographic Experts Group
18	GIF	Graphics Interchange Format
19	BPM	Beats Per Minute
20	CC	Contour Comparison
21	BoF	Bag of Features
22	LTV	Low Transport Vehicle
23	MTV	Medium Transport Vehicle
24	HTV	Heavy Transport Vehicle
25	MOG	Mixture of Gaussians
26	GMG	Gaussian Mixture Model Background Subtraction

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The need of efficient management and monitoring of road traffic has increased in last few decades because of the increase in the road networks, the number and most importantly the size of vehicles. Intelligent traffic surveillance systems are very important part of modern day traffic management but the regular traffic management techniques such as wireless sensor networks[1], Inductive loops[2] and EM microwave detectors[3] are expensive, bulky and are difficult to install without interrupting the traffic. A good alternative to these techniques can be video based surveillance systems.

Video surveillance systems[4-8] have become cheaper and better because of the increase in the storage capabilities, computational power and video encryption algorithms[9]. The videos stored by these surveillance systems are generally analysed by humans, which is a time consuming Job. To overcome this constraint, the need of more robust, automatic video based surveillance systems has increased interest in field of computer vision.

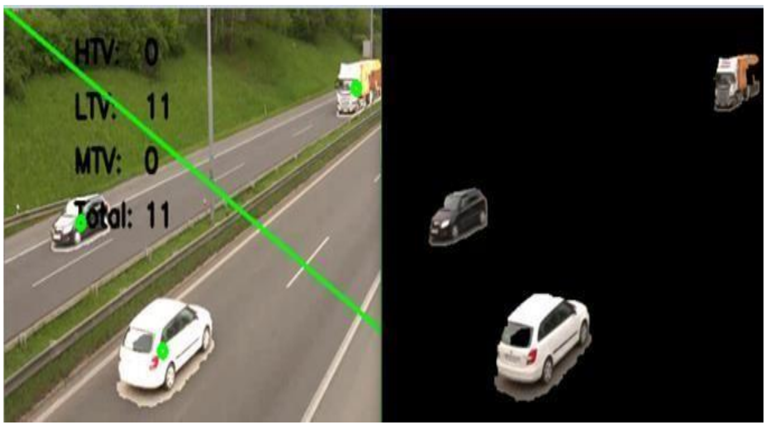


Fig 1: Frame of video before and after subtraction of background.

The objectives of a traffic surveillance system is to detect, track and classify the vehicles but they can be used to do complex tasks such as driver activity recognition, lane recognition etc. The traffic surveillance systems can have applications in a range of fields such as, public security, detection of anomalous behaviour, accident detection, vehicle theft detection, parking areas, and person identification. A Traffic surveillance system usually contains two parts, hardware and software. Hardware is a static camera installed on the roadside that captures the video feed and the software part of the system is concerned with processing and analyses. These systems could be portable with a microcontroller attached to the camera for the real-time processing and analyses or just the cameras that transmit the video feed to a centralized computer for further processing.

Various approaches were made to develop such systems that can detect, count and classify the vehicles and can be used for traffic surveillance in intelligent transportation systems. This section covers the discussion about such systems and the knowledge about the methods used to develop such systems.

Computer vision technology is using for traffic monitoring in many countries [10], [11]. The development of computer vision technology over video based traffic monitoring for detecting moving vehicles in video streams become an essential part in ITS [12], [13]. A good number of work has been done on vehicle tracking and detection using computer vision technology. In 2005, Hasegawa and Kanade [14] introduced a system for detecting and classifying the moving objects by its type and colour. In this process, a series of images of a specific location were supplied and vehicles from these images were identified. In 2013, Nilesh et al. designed and developed a system using python with OpenCV for detecting and counting moving vehicles. It can automatically identify and count moving objects as vehicle in real-time or from recorded videos, which basically used background subtraction, image filtering, image binary and segmentation method. In 2014, Da Li et al. developed real-time moving vehicle detection, tracking, and counting system also using python with OpenCV including adaptive subtracted background method in combination with virtual detector and blob tracking technology. Virtual detector constructs a set of rectangular regions in each input image frame and blob tracking method generates input image frames, the absolute difference between the background image and foreground blobs corresponding to the vehicles on the road. The above systems have some limitations like tackling shadows, occlusion of multiple vehicles that appear in a single region. Peek Traffic Corporation commercially developed several video traffic detection systems at the present time.

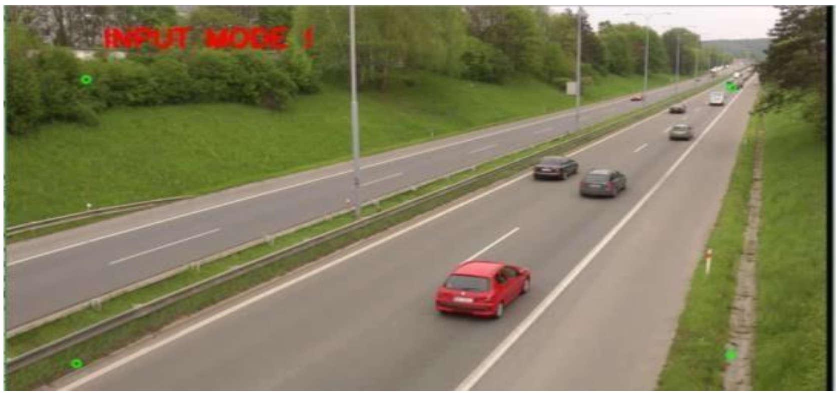


Fig 2: Region of interest selection input mode.

Efficient management and monitoring of road traffic have become increasingly critical in recent decades due to the expansion of road networks and the proliferation of vehicles, particularly larger ones. Intelligent traffic surveillance systems play a vital role in modern traffic management, yet traditional techniques like wireless sensor networks, inductive loops, and EM microwave detectors are expensive, bulky, and disruptive to install without interrupting traffic flow. Video-based surveillance systems offer a promising alternative, becoming cheaper and more effective with advancements in storage capabilities, computational power, and video encryption algorithms.

Videos stored by surveillance systems are typically analysed by humans, a time-consuming process prompting the need for more robust, automatic video-based surveillance systems leveraging computer vision technology. The objectives of traffic surveillance systems extend beyond mere detection, tracking, and classification of vehicles to encompass complex tasks like driver activity recognition and lane detection. These systems find applications in public security, anomaly detection, accident detection, vehicle theft prevention, parking management, and person identification.

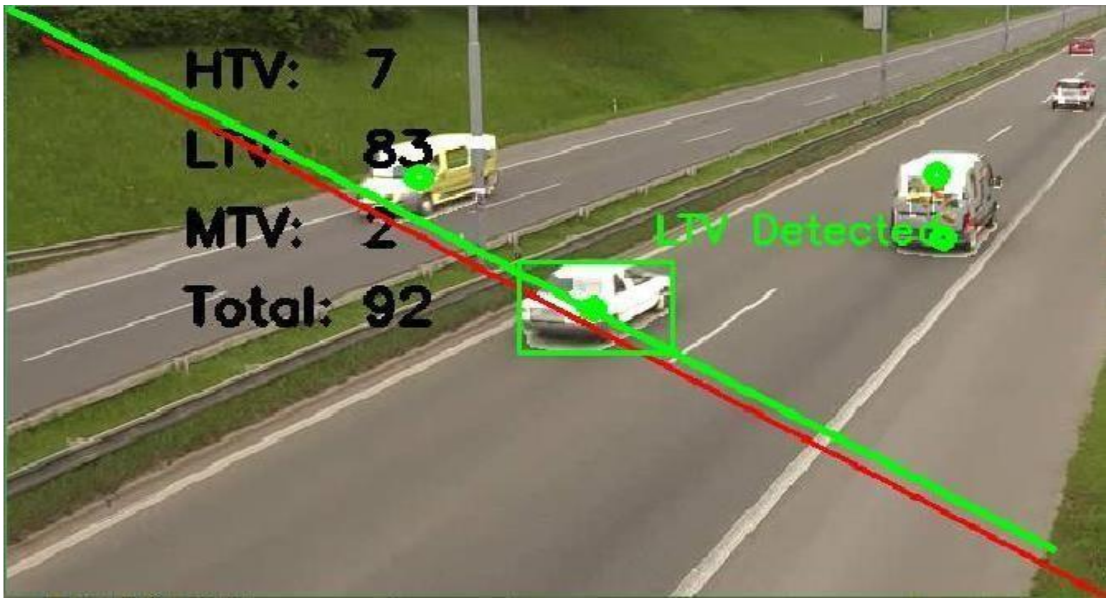


Fig 3: Increment in corresponding variables when vehicle centroid touches imaginary line in ROI (Mask Applied)

A typical traffic surveillance system comprises hardware (e.g., static cameras) and software for video feed processing and analysis. These systems can be portable with microcontrollers for real-time processing or transmit video feeds to centralized computers for analysis. Various approaches have been developed for vehicle detection, counting, and classification in traffic surveillance systems using computer vision technology.

Computer vision technology has been widely adopted for traffic monitoring, with significant advancements in video-based traffic monitoring for detecting moving vehicles. Researchers have developed systems for vehicle tracking and detection, including methods for detecting and classifying moving objects based on type and color. Techniques such as background subtraction, image filtering, and segmentation are commonly used for real-time vehicle detection and counting.

Despite advancements, challenges persist, including the handling of shadows and occlusions caused by multiple vehicles appearing in a single region. Commercial entities like Peek Traffic Corporation have developed video traffic detection systems to address these challenges. Ongoing research and development efforts aim to enhance the accuracy, reliability, and scalability of real-time vehicle detection and counting systems, further advancing the field of intelligent transportation systems.

PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library

Purpose

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

1.2 DETAILED DESCRIPTION OF THE PROJECT /SYSTEM REQUIREMENTS

The project entails creating a user-friendly application with a graphical user interface. It relies on Python, Flask, and other software tools for development.

Hardware requirements include a Pentium IV processor or higher, 256 MB RAM, and minimum 512MB of hard disk space. The system design emphasizes maintainability, robustness, reliability, and a reasonable size for efficient performance. The system is intended to convey information about past activities, current status, and future projections, as well as trigger and confirm actions.

Additionally, the project involves the use of a browser-compatible version to ensure accessibility across various platforms.

1.3 PROJECT OBJECTIVES AND SCOPE

The primary objective is to analyze the design of applications to make them more user-friendly, with well-ordered navigation and reduced user input. The project aims to achieve maintainability, robustness, reliability, and efficiency in terms of size and speed.

The scope encompasses the technical and user-oriented aspects, ensuring that the system is not only functionally effective but also socially feasible, focusing on the level of acceptance and user confidence in the system. Training and user familiarization are important aspects considered in the project.

CHAPTER 2

LITERATURE SURVEY

2.1 SUMMARY OF EXISTING ARTICLES

Tursun, M and Amrulla, G [4] proposed a video based real-time vehicle counting system using optimized virtual loop method. They used real time traffic surveillance cameras deployed over roads and compute how many vehicles pass the road. In this system counting is completed in three steps by tracking vehicle movements within a tracking zone called virtual loop. Another video based vehicle counting system was proposed by Lei, M., et al. [5]. In this system surveillance cameras were used and mounted at relatively high place to acquire the traffic video stream, the Adaptive background estimation and the Gaussian shadow elimination are the two main methods that were used in this system. The accuracy rate of the system depends on the visual angle and ability to remove shadows and ghost effects. The system's incompetency to classify vehicle type is the core limitation of the system.

Bas et al. proposed a video analysis method to count vehicles [10] based on an adaptive bounding box size to detect and track vehicles in accordance with estimated distance from the camera. The Region of Interest (ROI) is identified by defining a boundary for each outbound and inbound in the image. Although the algorithm is improved to deal with some weather conditions it is unable to track vehicles when they change their directions.

Mithun, N.C., et al proposed a vehicle detection and classification system using time spatial image and multiple virtual detection line[6]. A two-step K nearest neighborhood (KNN) algorithm is adopted to classify vehicles via shape invariant and texture based features. Experiments confirm the better accuracy and low error rate of proposed method over existing methods since it also considers the various illumination conditions.

Habibu Rabiu proposed a vehicle detection and classification for cluttered urban intersection [11]. In this system background subtraction and kalman filter algorithm are used to detect and track the vehicles and Linear Discriminant Analysis classifier is used for proper classification of vehicles.

Detection of vehicles in a video based traffic surveillance system is first and very important phase as it greatly impacts the other algorithms such as tracking and classification of the vehicles hence an accurate detection and segmentation of the foreground moving object is very important. Many of the techniques are used for foreground detection like frame differencing [12]. Frame differencing can be considered as the simplest foreground detection and segmentation method as it is based on the close relationship among the sequence of motion images.

An improved frame differencing method was presented by Collins [7] which uses difference between multiple frames to compute the foreground instead of just using the initial frame. Another method named as Optical Flow Field method was brought out by Gibson[13]. Wu, K, et al. proposed that the optical flow represents the velocity of mode within an image [14].Optical Flow method takes image within the detecting area as a vector field of velocity; each vector represents the transient variation of the position for a pixel in the scenery. Another method used to detect foreground is average model [8]. In average model the average grey value of a pixel in a sequence of frames is considered as the background value of same pixel. A GMM was proposed by Friedman, N. and S. Russell.[15] and was refined for real time tracking by Stauffer, C. and W.E.L. Grimson[16, 17].

Gaussian Mixture Model relies on assumptions that the background is visible more frequently than any foreground regions. Elgammal proposed Kernel density estimation based nonparametric background model [18]. Kernel density estimation method evaluates the samples of video data using kernel functions and it samples data which has maximal probability density as background is selected. A bag of features model is the one which represents images as order less collections of local features [19]. The name bag of features came from the bag of words representation used in text based information retrieval.

Scale Invariant Feature Transform algorithm (SIFT) was introduced by David Lowe in 1999 [20] and refined in 2004 [21]. SIFT is used to extract the features from dataset. In SIFT features are invariant to image scaling, rotation and partially invariant to change in illumination and 3D camera viewpoint [21]. SIFT with SVM requires less number of dataset. With any supervised learning model, first SVM is trained to cross validate the classifier. Then trained machine is used to make predictions and classify the data [22]. Bhushan et al. [23], proposed a vehicle detection method based on morphological operations including binarization, edge detection, and top-hat processing, masking operation. Proposed system fails to give good results n cloudy environment.

Raúl et al. [24], proposed a classification technique for moving objects in which information processing is employed via clustering and classification algorithms. Proposed methodology provides sufficient accuracy for it to be employed for real-time applications but still the system can be further improved for vehicle classification in complex scenarios such as weather and illumination conditions.

A. Suryatali and V.B. Dharmadhikari in [25], proposed a Computer Vision based vehicle detection that counts and also classifies the vehicles in to heavy and light weight vehicles; object detection is accomplished by making use of kalman filter for background subtraction and then openCV library is used finally to detect the object in processed frame. Nilakorn et al. [26], proposed vehicle detection and counting prototype which uses different steps for background subtraction and object detection then uses CV techniques such as thresholding, hole-filling and adaptive morphological dilation to remove noise and enhance the foreground objects in particular frame from video. Proposed system provides limited functionality for the objects appearing in detection zone if they are occluded or small.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

A vehicle detection and classification system using time spatial image and multiple virtual detection line[6]. A two-step K nearest neighborhood (KNN) algorithm is adopted to classify vehicles via shape invariant and texture based features. Experiments confirm the better accuracy and low error rate of proposed method over existing methods since it also considers the various illumination conditions. People have designed such mechanism for decades now but most of them involve use of sensors to detect the vehicles i.e. a couple of proximity sensors to calculate the direction of the moving vehicle and to keep the vehicle count. Even though over the time these systems have matured and are highly effective, they are not very budget friendly. The problem is such systems require maintenance and periodic calibration.

3.2 DISADVANTAGES OF EXISTING SYSTEM

- Watching videos manually to understand traffic is slow and can make mistakes.
- The old ways like sensor networks need a lot of fixing and checking, which costs a lot.
- These old methods might not work well when there are more cars on the road.
- Because they're old-fashioned, they might not always spot cars right or tell them apart correctly.
- Setting up these systems can be really hard and takes a lot of time and money.
- The traditional methods may lack flexibility to adapt to changing traffic patterns and advancements in technology

3.3 PROPOSED SYSTEM

The system could be used for detection, recognition and tracking of the vehicles in the video frames and then classify the detected vehicles according to their size in three different classes. The proposed system is based on three modules which are background learning, foreground extraction and vehicle classification as shown in fig. 1. Background subtraction is a classical approach to obtain the

foreground image or in other words to detect the moving objects. We have proposed an adaptive video based vehicle
detection, classification, counting for real-time traffic data collection. The proposed system was built using python programming language and OpenCV. The main objective for developing this system is to collect vehicle count and classification data. So that we can build intelligent transportation network based on historical traffic data. The proposed system can engender traffic data by detecting, classifying, counting It's a plug & play system and applied YOLO algorithm as a background subtraction technique. The proposed system was tested at different six locations in Hyderabad under different traffic and environmental conditions.

3.4 ADVANTAGES OF PROPOSED SYSTEM

- The new system uses advanced computer vision techniques to find and classify vehicles, while the old one relies on simpler methods like sensors.
- Instead of people having to watch videos all the time, the new system can do it automatically, saving time.
- The new system is cheaper because it doesn't need expensive sensors and doesn't require people to watch it constantly.
- It's easy to expand the new system to cover more areas or locations, while the old one may struggle with this.
- In the future, the new system could learn to do even more things, like watching live videos and choosing the best spots to monitor.

CHAPTER 4

SYSTEM REQUIREMENT SPECIFICATION

4.1 HARDWARE REQUIREMENTS

- Python 3.6.9 or later: The core programming language for developing the software.
- OpenCV-Python 4.3.0.38: Used for image and video processing tasks such as reading video files, frame manipulation, and background subtraction.
- Imutils 0.5.4: Provides convenience functions to simplify basic image processing operations in OpenCV.
- SciPy 1.4.1: Required for various scientific computing tasks, possibly utilized for certain image processing functionalities.
- Any Browser (Particularly Chrome): The system needs to be compatible with various browsers, particularly Chrome, to ensure accessibility across different platforms.
- Operating Systems Supported: The system is intended to be compatible with the following operating systems:
  Windows 7

  Windows XP

4.2 SOFTWARE REQUIREMENTS

- Processor: Pentium IV 2.4 GHz or higher
- RAM: 256 MB minimum
- Hard Disk Space: Minimum of 512 MB
- Monitor: 14" Color Monitor
- Mouse: Optical Mouse
- Additional: Floppy Drive with 1.44 Mb capacity

CHAPTER 5

FEASIBILITY STUDY

5.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are,

- **ECONOMICAL FEASIBILITY**
- **TECHNICAL FEASIBILITY**
- **SOCIAL FEASIBILITY**

5.2 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

5.3 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

5.4 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER 6

SYSTEM DESIGN

6.1 DATAFLOW / BLOCK DIAGRAM

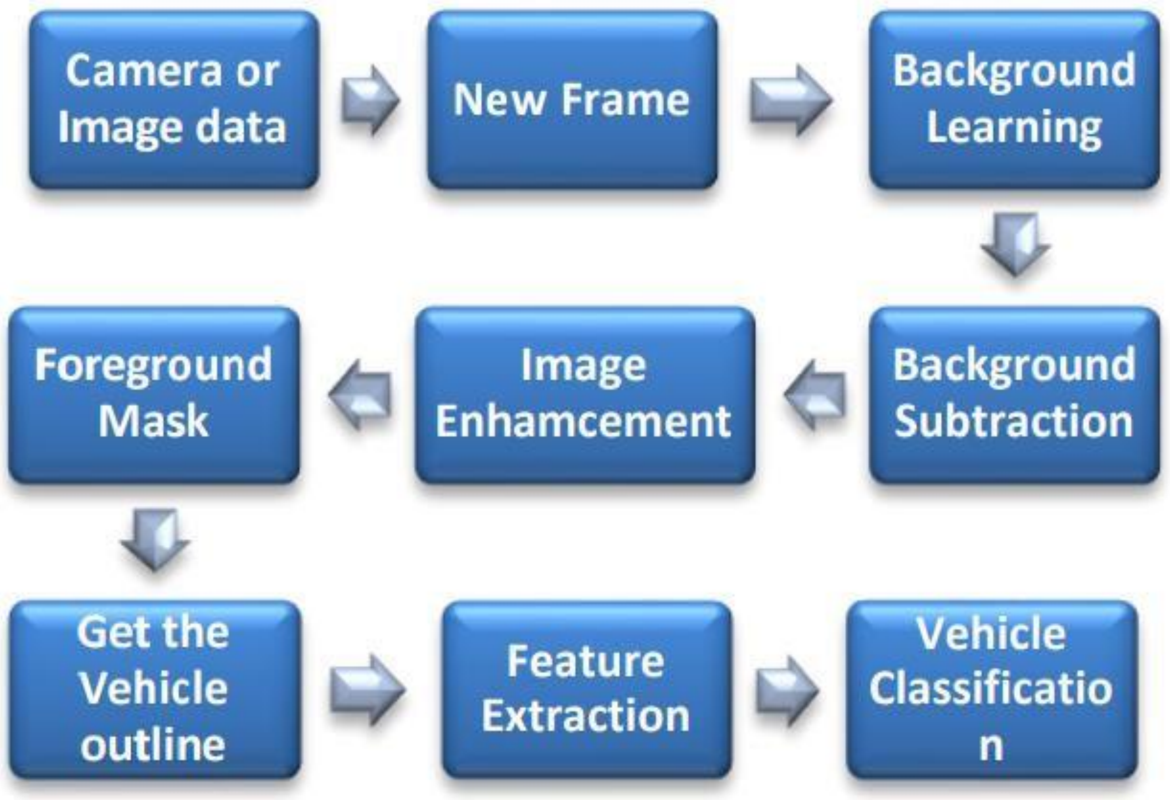


Fig 4: Block diagram of proposed vehicle detection and counting system.

6.2 DESCRIPTION OF THE FLOW

The workflow unfolds as follows. Initially, data acquisition involves gathering video streams or footage from traffic cameras situated strategically to cover the target area comprehensively. Following this, preprocessing steps are implemented, including frame extraction and potential resizing or cropping to optimize processing efficiency. Additionally, normalization techniques may be applied to enhance image quality and consistency, priming the data for subsequent analysis.

Once preprocessed, the heart of the system lies in object detection algorithms, leveraging pre-trained models like YOLO or Faster R-CNN to accurately identify vehicles within each frame. This detection phase serves as the cornerstone for vehicle tracking and counting, enabling the system to assign unique identifiers to vehicles and monitor their movements across frames seamlessly. This tracking capability ensures continuity in counting, even amidst challenging scenarios such as occlusions or overlapping vehicles.

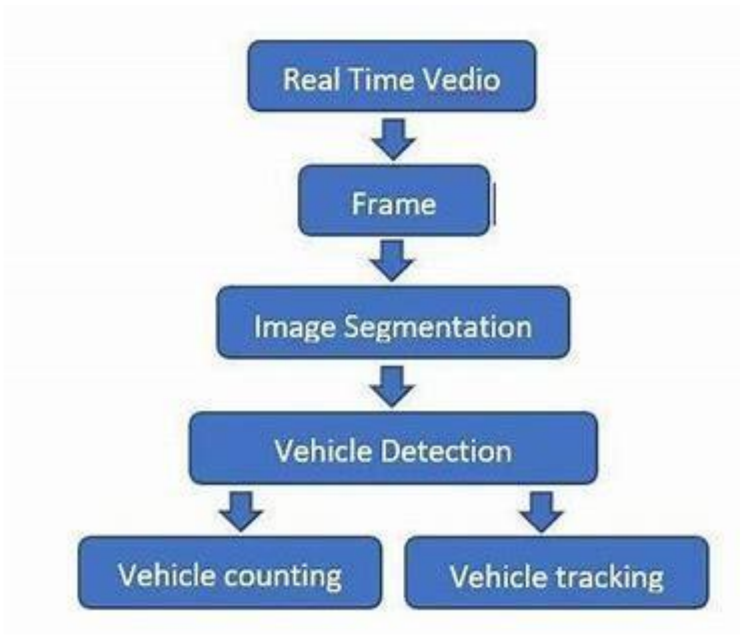


Fig 5: Flow chart of real time vehicle detection

With vehicles detected and tracked, the system proceeds to the counting phase, tallying the number of vehicles passing through predefined regions or lines of interest in real-time. By integrating with visual overlays or graphical representations, the system provides instant feedback on vehicle counts, facilitating efficient traffic monitoring and management. Moreover, the system can generate detailed reports or analytics, offering insights into traffic patterns and trends over specific timeframes or geographical areas.

To optimize system performance, parameters such as detection thresholds and tracking algorithms are fine-tuned through rigorous testing and validation. This iterative process ensures robustness and reliability, critical for real-world deployment. Once optimized, the system is deployed on suitable hardware infrastructure capable of handling real-time processing demands, either as standalone solutions or integrated within existing traffic management frameworks.

Ongoing maintenance and updates are essential to sustain system effectiveness. Regular monitoring of performance metrics, prompt resolution of issues, and periodic updates to adapt to evolving traffic conditions or advancements in machine learning techniques are paramount. By adhering to this workflow, your project on real-time vehicle detection and counting harnesses the power of machine learning to deliver actionable insights for effective traffic management and planning.

Input & Output Design

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

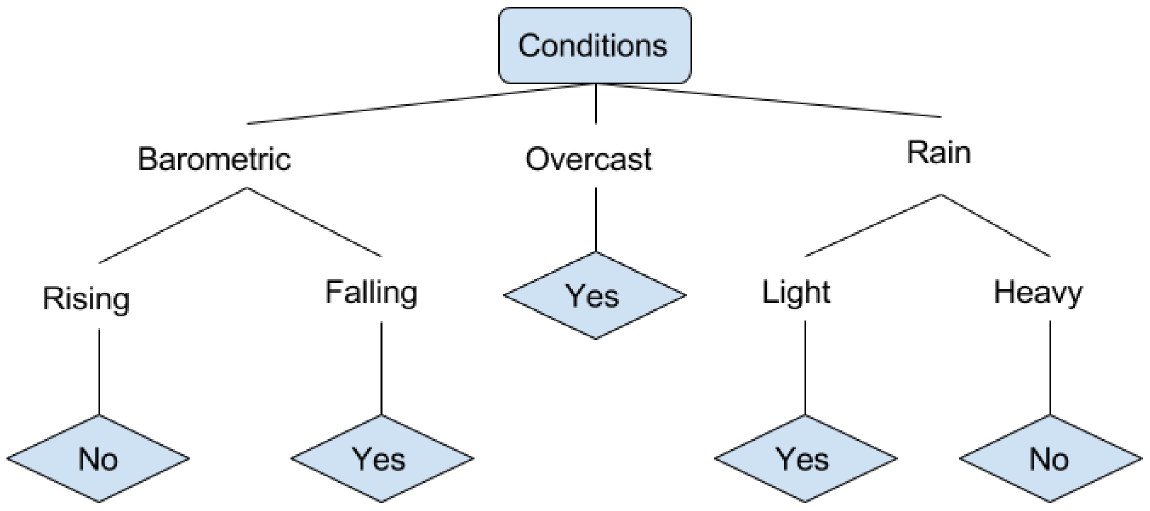


Fig 6: Flow of work in different weather conditions

OBJECTIVES

1.Input Design is the process of converting a user-oriented description of the input into a computerbased system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry

screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3.When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

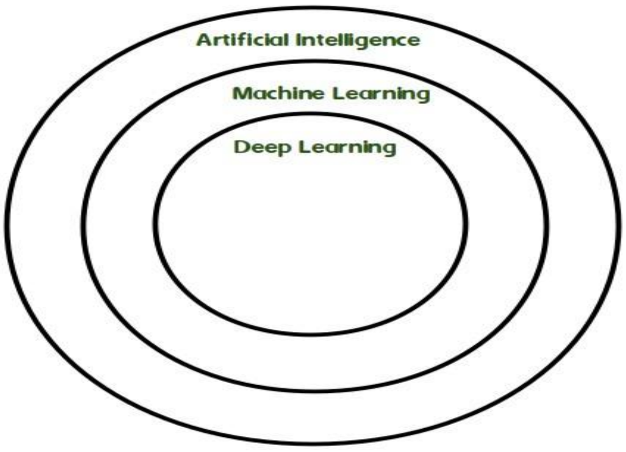


Fig 7: Categorization of Mathine learning Models

<u>OUTPUT DESIGN</u>

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

CHAPTER 7

PROJECT IMPLEMENTATION

7.1 SOFTWARE TOOL

➢ **Vision Based Vehicle Counting and Classification System**

The system proposed a vision-based approach involving background subtraction and Gaussian Mixture Model (GMM) to detect and count vehicles in video frames. Introduced two classification methods, Contour Comparison (CC) and Bag of Features (BoF), for vehicle classification.

➢ **Traffic Surveillance System**

Explained the significance of traffic surveillance systems in modern traffic management, emphasizing applications and components. Highlighted the role of computer vision technology in traffic monitoring and Intelligent Transportation Systems (ITS), referencing prior work in vehicle tracking and detection. Literature Survey on Vehicle Counting and Classification. Reviewed several methods and systems for real-time vehicle counting and classification, summarizing their strategies and limitations. Emphasized the importance of precise foreground detection and outlined various methodologies used for this purpose.

➢ **Background Subtraction and Object Detection Methods**

Discussed different techniques for foreground detection, including frame differencing, optical flow field, and Gaussian Mixture Model (GMM). Introduced the bag of features model and the Scale Invariant Feature Transform algorithm (SIFT) for image representation and feature extraction.

➢ **Vehicle Detection and Classification Techniques**

Explored various methodologies for vehicle detection and classification, such as kernel density estimation, SIFT features with Support Vector Machines (SVM), and morphological operations. Highlighted the potential and limitations of proposed techniques for real-time applications. This information reflects the software tools and methods utilized in the domain of vehicle counting, classification, and surveillance systems, focusing on computer vision technologies and their application in traffic monitoring and analysis.

7.2 PROGRAMMING

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the nonprofit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including objectoriented, imperative, functional and procedural, and has a large and comprehensive standard library.

What is Python

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

Web development (server-side), software development, mathematics, system scripting. **What can Python do**

Python can be used on a server to create web applications.

Python can be used alongside software to create workflows.

Python can connect to database systems. It can also read and modify files.

Python can be used to handle big data and perform complex mathematics.

Python can be used for rapid prototyping, or for production-ready software development. **Why Python**

Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).

Python has a simple syntax similar to the English language.

Python has syntax that allows developers to write programs with fewer lines than some other programming languages.

Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.

Python can be treated in a procedural way, an object-orientated way or a functional way.

Good to know

The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.

In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

Python was designed for readability, and has some similarities to the English language with influence from mathematics.

Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.

Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Python Install

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

```
C:\Users\Your Name>python --version
```

To check if you have python installed on a Linux or Mac, then on linux open the command line or on Mac open the Terminal and type: python --version

If you find that you do not have python installed on your computer, then you can download it for free from the following website: https://www.python.org/

Python Quickstart

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

```
C:\Users\Your Name>python helloworld.py
```

Where "helloworld.py" is the name of your python file.

Let's write our first Python file, called helloworld.py, which can be done in any text editor.
helloworld.py print("Hello, World!")
Simple as that. Save your file. Open your command line, navigate to the directory where you saved your file, and run:

```
C:\Users\Your Name>python helloworld.py
```

The output should read:

```
Hello, World!
```

Congratulations, you have written and executed your first Python program.

The Python Command Line

To test a short amount of code in python sometimes it is quickest and easiest not to write the code in a file. This is made possible because Python can be run as a command line itself.

Type the following on the Windows, Mac or Linux command line:

```
C:\Users\Your Name>python
```

Or, if the "python" command did not work, you can try "py":

```
C:\Users\Your Name>py
```

From there you can write any python, including our hello world example from earlier in the tutorial:

```
C:\Users\Your Name>python

Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>>print("Hello, World!")
```

Which will write "Hello, World!" in the command line:

```
C:\Users\Your Name>python

Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>>print("Hello, World!")

Hello, World!
```

Whenever you are done in the python command line, you can simply type the following to quit the python command line interface:

```
exit()
```

Virtual Environments and Packages

Introduction

Python applications will often use packages and modules that don’t come as part of the standard library. Applications will sometimes need a specific version of a library, because the application may require that a particular bug has been fixed or the application may be written using an obsolete version of the library’s interface.

This means it may not be possible for one Python installation to meet the requirements of every application. If application A needs version 1.0 of a particular module but application B needs version 2.0, then the requirements are in conflict and installing either version 1.0 or 2.0 will leave one application unable to run.

The solution for this problem is to create a virtual environment, a self-contained directory tree that contains a Python installation for a particular version of Python, plus a number of additional packages.

Different applications can then use different virtual environments. To resolve the earlier example of conflicting requirements, application A can have its own virtual environment with version 1.0 installed while application B has another virtual environment with version 2.0. If application B requires a library be upgraded to version 3.0, this will not affect application A's environment.

Creating Virtual Environments

The module used to create and manage virtual environments is called venv. venv will usually install the most recent version of Python that you have available. If you have multiple versions of Python on your system, you can select a specific Python version by running python3 or whichever version you want.

To create a virtual environment, decide upon a directory where you want to place it, and run the venv module as a script with the directory path: python3 -m venv tutorial-env
This will create the tutorial-env directory if it doesn't exist, and also create directories inside it containing a copy of the Python interpreter, the standard library, and various supporting files.

A common directory location for a virtual environment is .venv. This name keeps the directory typically hidden in your shell and thus out of the way while giving it a name that explains why the directory exists. It also prevents clashing with .env environment variable definition files that some tooling supports.

Once you've created a virtual environment, you may activate it.

On Windows, run: tutorial-
env\Scripts\activate.bat On
Unix or MacOS, run: source
tutorial-env/bin/activate

(This script is written for the bash shell. If you use the csh or fish shells, there are alternate activate.csh and activate.fish scripts you should use instead.)

Activating the virtual environment will change your shell's prompt to show what virtual environment you're using, and modify the environment so that running python will get you that particular version and installation of Python. For example:

```
$ source ~/envs/tutorial-env/bin/activate

(tutorial-env) $ python

Python 3.5.1 (default, May  6 2016, 10:59:36)

>>> import sys

>>>sys.path

['', '/usr/local/lib/python35.zip', ...,

'~/envs/tutorial-env/lib/python3.5/site-packages']

>>>
```

12.3. Managing Packages with pip

You can install, upgrade, and remove packages using a program called pip. By default pip will install packages from the Python Package Index, <https://pypi.org>. You can browse the Python Package Index by going to it in your web browser, or you can use pip's limited search feature:

```
(tutorial-env) $ pip search astronomy skyfield              - Elegant
astronomy for Python gary                - Galactic astronomy and
gravitational dynamics.
novas              - The United States Naval Observatory NOVAS astronomy library astroobs            -
Provides astronomy ephemeris to plan telescope observations PyAstronomy          - A collection of
```

astronomy related tools for Python. pip has a number of subcommands: "search", "install", "uninstall", "freeze", etc. (Consult the Installing Python Modules guide for complete documentation for pip.)

You can install the latest version of a package by specifying a package's name:

```
(tutorial-env) $ pip install novas

Collecting novas

Downloading novas-3.1.1.3.tar.gz (136kB)

Installing collected packages: novas

Running setup.py install for novas

Successfully installed novas-3.1.1.3
```

You can also install a specific version of a package by giving the package name followed by == and the version number:

```
(tutorial-env) $ pip install requests==2.6.0

Collecting requests==2.6.0

Using cached requests-2.6.0-py2.py3-none-any.whl

Installing collected packages: requests

Successfully installed requests-2.6.0
```

If you re-run this command, pip will notice that the requested version is already installed and do nothing. You can supply a different version number to get that version, or you can run pip install -upgrade to upgrade the package to the latest version:

```
(tutorial-env) $ pip install --upgrade requests

Collecting requests
Installing collected packages: requests Found
existing installation: requests 2.6.0
Uninstalling requests-2.6.0:
Successfully uninstalled requests-2.6.0 Successfully installed requests-2.7.0
```

pip uninstall followed by one or more package names will remove the packages from the virtual environment.

pip show will display information about a particular package:

```
(tutorial-env) $ pip show requests

Metadata-Version: 2.0

Name: requests

Version: 2.7.0

Summary: Python HTTP for Humans.

Home-page: http://python-requests.org

Author: Kenneth Reitz

Author-email: me@kennethreitz.com

License: Apache 2.0

Location: /Users/akuchling/envs/tutorial-env/lib/python3.4/site-packages Requires:
```

pip list will display all of the packages installed in the virtual environment:

```
(tutorial-env) $ pip list
novas (3.1.1.3) numpy
(1.9.2) pip (7.0.3)
```

requests (2.7.0) setuptools (16.0) pip freeze will produce a similar list of the installed packages, but the output uses the format that pip install expects. A common convention is to put this list in a requirements.txt file:

```
(tutorial-env) $ pip freeze > requirements.txt
(tutorial-env) $ cat requirements.txt
novas==3.1.1.3 numpy==1.9.2
requests==2.7.0
```

The requirements.txt can then be committed to version control and shipped as part of an application. Users can then install all the necessary packages with install -r:

```
(tutorial-env) $ pip install -r requirements.txt

Collecting novas==3.1.1.3 (from -r requirements.txt (line 1))
Collecting numpy==1.9.2 (from -r requirements.txt (line 2))

Collecting requests==2.7.0 (from -r requirements.txt (line 3))

Installing collected packages: novas, numpy, requests

Running setup.py install for novas

Successfully installed novas-3.1.1.3 numpy-1.9.2 requests-2.7.0
```

pip has many more options. Consult the Installing Python Modules guide for complete documentation for pip. When you've written a package and want to make it available on the Python Package Index, consult the Distributing Python Modules guide.

Cross Platform

Platform. Architecture (executable=sys.executable, bits='', linkage='')
Queries the given executable (defaults to the Python interpreter binary) for various architecture information.

Returns a tuple (bits, linkage) which contain information about the bit architecture and the linkage format used for the executable. Both values are returned as strings.

Values that cKNNot be determined are returned as given by the parameter presets. If bits is given as '', the sizeof(pointer) (or sizeof(long) on Python version < 1.5.2) is used as indicator for the supported pointer size.

The function relies on the system's file command to do the actual work. This is available on most if not all Unix platforms and some non-Unix platforms and then only if the executable points to the Python interpreter. Reasonable defaults are used when the above needs are not met.

Note On Mac OS X (and perhaps other platforms), executable files may be universal files containing multiple architectures.

To get at the "64-bitness" of the current interpreter, it is more reliable to query the sys.maxsize attribute:

```
is_64bits = sys.maxsize> 2**32 platform.machine
()
```

Returns the machine type, e.g. 'i386'. An empty string is returned if the value cKNNot be determined.

platform.node ()

Returns the computer’s network name (may not be fully qualified!). An empty string is returned if the value cKNNot be determined.

platform. Platform(aliased=0, terse=0)

Returns a single string identifying the underlying platform with as much useful information as possible.

The output is intended to be human readable rather than machine parseable. It may look different on different platforms and this is intended.

If aliased is true, the function will use aliases for various platforms that report system names which differ from their common names, for example SunOS will be reported as Solaris. The system_alias() function is used to implement this.

Setting terse to true causes the function to return only the absolute minimum information needed to identify the platform.

platform.processor()

Returns the (real) processor name, e.g. 'amdk6'.

An empty string is returned if the value cKNNot be determined. Note that many platforms do not provide this information or simply return the same value as for machine(). NetBSD does this.

platform.python_build()

Returns a tuple (buildno, builddate) stating the Python build number and date as strings.
platform.python_compiler()
Returns a string identifying the compiler used for compiling Python.

platform.python_branch()

Returns a string identifying the Python implementation SCM branch.

New in version 2.6.

platform.python_implementation()

Returns a string identifying the Python implementation. Possible return values are: ‘CPython’, ‘IronPython’, ‘Jython’, ‘PyPy’.

New in version 2.6.

platform.python_revision()

Returns a string identifying the Python implementation SCM revision.

New in version 2.6.

platform.python_version()

Returns the Python version as string 'major.minor.patchlevel'.

Note that unlike the Python sys.version, the returned value will always include the patchlevel (it defaults to 0).

platform.python_version_tuple()

Returns the Python version as tuple (major, minor, patchlevel) of strings.

Note that unlike the Python sys.version, the returned value will always include the patchlevel (it defaults to '0'). platform.release()
Returns the system’s release, e.g. '2.2.0' or 'NT' An empty string is returned if the value cKNNot be determined.

platform.system()

Returns the system/OS name, e.g. 'Linux', 'Windows', or 'Java'. An empty string is returned if the value cKNNot be determined.

platform.system_alias(system, release, version)

Returns (system, release, version) aliased to common marketing names used for some systems. It also does some reordering of the information in some cases where it would otherwise cause confusion.

platform.version()

Returns the system's release version, e.g. '#3 on degas'. An empty string is returned if the value cKNNot be determined.

platform.uname()

Fairly portable uname interface. Returns a tuple of strings (system, node, release, version, machine, processor) identifying the underlying platform.

Note that unlike the os.uname() function this also returns possible processor information as additional tuple entry.

Entries which cKNNot be determined are set to ''. **Java**
Platform
platform.java_ver(release='', vendor='', vminfo=('', '', ''), osinfo=('', '', ''))

Version interface for Jython.

Returns a tuple (release, vendor, vminfo, osinfo) with vminfo being a tuple (vm_name, vm_release, vm_vendor) and osinfo being a tuple (os_name, os_version, os_arch). Values which cKNNot be determined are set to the defaults given as parameters (which all default to ''). Windows Platform platform.win32_ver(release='', version='', csd='', ptype='')

Get additional version information from the Windows Registry and return a tuple (release, version, csd, ptype) referring to OS release, version number, CSD level (service pack) and OS type (multi/single processor).

As a hint: ptype is 'Uniprocessor Free' on single processor NT machines and 'Multiprocessor Free' on multi processor machines. The 'Free' refers to the OS version being free of debugging code. It could also state 'Checked' which means the OS version uses debugging code, i.e. code that checks arguments, ranges, etc.

Note This function works best with Mark Hammond's win32all package installed, but also on Python 2.3 and later (support for this was added in Python 2.6). It obviously only runs on Win32 compatible platforms. **Win95/98 specific** platform.popen(cmd, mode='r', bufsize=None)

Portable popen() interface. Find a working popen implementation preferring win32pipe.popen(). On Windows NT, win32pipe.popen() should work; on Windows 9x it hangs due to bugs in the MS C library.

Mac OS Platform

platform.mac_ver(release='', versioninfo=('', '', ''), machine='')

Get Mac OS version information and return it as tuple (release, versioninfo, machine) with versioninfo being a tuple (version, dev_stage, non_release_version).

Entries which cKNNot be determined are set to ''. All tuple entries are strings. **Unix Platforms** platform.dist(distname='', version='', id='', supported_dists=('SuSE', 'debian', 'redhat', 'mandrake', ...))

This is an old version of the functionality now provided by linux_distribution(). For new code, please use the linux_distribution().

The only difference between the two is that dist() always returns the short name of the distribution taken from the supported_dists parameter.

Deprecated since version 2.6. platform.linux_distribution(distname='', version='', id='', supported_dists=('SuSE', 'debian', 'redhat', 'mandrake', ...), full_distribution_name=1)

Tries to determine the name of the Linux OS distribution name. supported_dists may be given to define the set of Linux distributions to look for. It defaults to a list of currently supported Linux distributions identified by their release file name.

If full_distribution_name is true (default), the full distribution read from the OS is returned.
Otherwise the short name taken from supported_dists is used.
Returns a tuple (distname,version,id) which defaults to the args given as parameters. id is the item in parentheses after the version number. It is usually the version codename.

Note This function is deprecated since Python 3.5 and removed in Python 3.8. See alternative like the distro package. New in version 2.6.

platform.libc_ver(executable=sys.executable, lib='', version='', chunksize=2048)

Tries to determine the libc version against which the file executable (defaults to the Python interpreter) is linked. Returns a tuple of strings (lib, version) which default to the given parameters in case the lookup fails.

Note that this function has intimate knowledge of how different libc versions add symbols to the executable is probably only usable for executables compiled using gcc. The file is read and scKNNed in chunks of chunksize bytes.

2. Using the Python Interpreter

2.1. Invoking the Interpreter

The Python interpreter is usually installed as /usr/local/bin/python3.8 on those machines where it is available; putting /usr/local/bin in your Unix shell's search path makes it possible to start it by typing the command: python3.8

to the shell. 1 Since the choice of the directory where the interpreter lives is an installation option, other places are possible; check with your local Python guru or system administrator. (E.g., /usr/local/python is a popular alternative location.)

On Windows machines where you have installed Python from the Microsoft Store, the python3.8 command will be available. If you have the py.exe launcher installed, you can use the py command. See Excursus: Setting environment variables for other ways to launch Python.

Typing an end-of-file character (Control-D on Unix, Control-Z on Windows) at the primary prompt causes the interpreter to exit with a zero exit status. If that doesn't work, you can exit the interpreter by typing the following command: quit().

The interpreter's line-editing features include interactive editing, history substitution and code completion on systems that support the GNU Readline library. Perhaps the quickest check to see whether command line editing is supported is typing Control-P to the first Python prompt you get. If it beeps, you have command line editing; see Appendix Interactive Input Editing and History Substitution for an introduction to the keys. If nothing appears to happen, or if ^P is echoed, command line editing isn't available; you'll only be able to use backspace to remove characters from the current line.

The interpreter operates somewhat like the Unix shell: when called with standard input connected to a tty device, it reads and executes commands interactively; when called with a file name argument or with a file as standard input, it reads and executes a script from that file.

A second way of starting the interpreter is python -c command [arg] ..., which executes the statement(s) in command, analogous to the shell's -c option. Since Python statements often contain spaces or other characters that are special to the shell, it is usually advised to quote command in its entirety with single quotes.

Some Python modules are also useful as scripts. These can be invoked using python -m module [arg] ..., which executes the source file for module as if you had spelled out its full name on the command line.

When a script file is used, it is sometimes useful to be able to run the script and enter interactive mode afterwards. This can be done by passing -i before the script.

All command line options are described in Command line and environment.

Argument Passing

When known to the interpreter, the script name and additional arguments thereafter are turned into a list of strings and assigned to the argv variable in the sys module. You can access this list by executing import sys. The length of the list is at least one; when no script and no arguments are given, sys.argv[0] is an empty string. When the script name is given as '-' (meaning standard input), sys.argv[0] is set to '-'. When -c command is used, sys.argv[0] is set to '-c'. When -m module is used, sys.argv[0] is set to the full name of the located module. Options found after -c command or -m module are not consumed by the Python interpreter's option processing but left in sys.argv for the command or module to handle.

Interactive Mode

When commands are read from a tty, the interpreter is said to be in interactive mode. In this mode it prompts for the next command with the primary prompt, usually three greater-than signs (>>>); for continuation lines it prompts with the secondary prompt, by default three dots (...). The interpreter prints a welcome message stating its version number and a copyright notice before printing the first prompt:

```
$ python3.8
```

Python 3.8 (default, Sep 16 2015, 09:25:04)

[GCC 4.8.2] on linux

Type "help", "copyright", "credits" or "license" for more information.

Continuation lines are needed when entering a multi-line construct. As an example, take a look at this if statement:

```
>>>the_world_is_flat = True >>>ifthe_world_is_flat:
...     print("Be careful not to fall off!")
```

Be careful not to fall off!

For more on interactive mode, see Interactive Mode.

2.2. The Interpreter and Its Environment

2.2.1. Source Code Encoding

By default, Python source files are treated as encoded in UTF-8. In that encoding, characters of most languages in the world can be used simultaneously in string literals, identifiers and comments — although the standard library only uses ASCII characters for identifiers, a convention that any portable code should follow. To display all these characters properly, your editor must recognize that the file is UTF-8, and it must use a font that supports all the characters in the file.

To declare an encoding other than the default one, a special comment line should be added as the first line of the file. The syntax is as follows:

# -*- coding: encoding -*- where encoding is one of the valid
codecs supported by Python.
For example, to declare that Windows-1252 encoding is to be used, the first line of your source code file should be:

```
# -*- coding: cp1252 -*-
```

One exception to the first line rule is when the source code starts with a UNIX "shebang" line. In this case, the encoding declaration should be added as the second line of the file. For example:

```
#!/usr/bin/env python3
```

INTRODUCTION TO ARTIFICIAL INTELLIGENCE

"The science and engineering of making intelligent machines, especially intelligent computer programs". -John McCarthy

Artificial Intelligence is an approach to make a computer, a robot, or a product to think how smart human think. AI is a study of how human brain think, learn, decide and work, when it tries to solve problems. And finally this study outputs intelligent software systems.The aim of AI is to improve computer functions which are related to human knowledge, for example, reasoning, learning, and problem-solving.

The intelligence is intangible. It is composed of

- Reasoning
- Learning
- Problem Solving
- Perception
- Linguistic Intelligence

The objectives of AI research are reasoning, knowledge representation, plKNNing, learning, natural language processing, realization, and ability to move and manipulate objects. There are long-term goals in the general intelligence sector.

Approaches include statistical methods, computational intelligence, and traditional coding AI. During the AI research related to search and mathematical optimization, artificial neural networks and methods based on statistics, probability, and economics, we use many tools. Computer science attracts AI in the field of science, mathematics, psychology, linguistics, philosophy and so on.

Trending AI Articles:

- Cheat Sheets for AI, Neural Networks, Machine Learning, Deep Learning & Big Data
- Data Science Simplified Part 1: Principles and Process
- Getting Started with Building Realtime API Infrastructure
- AI & NLP Workshop

Applications of AI

- Gaming − AI plays important role for machine to think of large number of possible positions based on deep knowledge in strategic games. for example, chess,river crossing, N-queens problems and etc.

- Natural Language Processing − Interact with the computer that understands natural language spoken by humans.
- Expert Systems − Machine or software provide explanation and advice to the users.
- Vision Systems − Systems understand, explain, and describe visual input on the computer.
- Speech Recognition − There are some AI based speech recognition systems have ability to hear and express as sentences and understand their meanings while a person talks to it. For example Siri and Google assistant.
- Handwriting Recognition − The handwriting recognition software reads the text written on paper and recognize the shapes of the letters and convert it into editable text. · Intelligent Robots − Robots are able to perform the instructions given by a human. Major Goals
- Knowledge reasoning
- PlKNNing
- Machine Learning
- Natural Language Processing
- Computer Vision
- Robotics

IBM Watson

"Watson" is an IBM supercomputer that combines Artificial Intelligence (AI) and complex inquisitive programming for ideal execution as a "question answering" machine. The supercomputer is named for IBM's founder, Thomas J.

Watson. IBM Watson is at the forefront of the new era of computing. At the point when IBM Watson made, IBM communicated that "more than 100 particular techniques are used to inspect perceive sources, find and make theories, find and score affirm, and combination and rank speculations." recently, the Watson limits have been expanded and the way by which Watson works has been changed to abuse new sending models (Watson on IBM Cloud) and propelled machine learning capacities and upgraded hardware open to architects and authorities. It isn't any longer completely a request answering figuring system arranged from Q&A joins yet can now 'see', 'hear', 'read', 'talk', 'taste', 'translate', 'learn' and 'endorse'.

Machine Learning

Introduction

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people.

Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.

Any technology user today has benefitted from machine learning. Facial recognition technology allows social media platforms to help users tag and share photos of friends. Optical character recognition (OCR) technology converts images of text into movable type. Recommendation engines, powered by machine learning, suggest what movies or television shows to watch next based on user preferences. Self-driving cars that rely on machine learning to navigate may soon be available to consumers

Machine learning is a continuously developing field. Because of this, there are some considerations to keep in mind as you work with machine learning methodologies, or analyze the impact of machine learning processes.

In this tutorial, we'll look into the common machine learning methods of supervised and unsupervised learning, and common algorithmic approaches in machine learning, including the knearest neighbor algorithm, decision tree learning, and deep learning. We'll explore which programming languages are most used in machine learning, providing you with some of the positive and negative attributes of each. Additionally, we'll discuss biases that are perpetuated by machine learning algorithms, and consider what can be kept in mind to prevent these biases when building algorithms.

Machine Learning Methods

In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed.

Two of the most widely adopted machine learning methods are supervised learning which trains algorithms based on example input and output data that is labeled by humans, and unsupervised learning which provides the algorithm with no labeled data in order to allow it to find structure within its input data. Let’s explore these methods in more detail.

Supervised Learning

In supervised learning, the computer is provided with example inputs that are labeled with their desired outputs. The purpose of this method is for the algorithm to be able to “learn” by comparing its actual output with the “taught” outputs to find errors, and modify the model accordingly. Supervised learning therefore uses patterns to predict label values on additional unlabeled data.

For example, with supervised learning, an algorithm may be fed data with images of sharks labeled as fish and images of oceans labeled as water. By being trained on this data, the supervised learning algorithm should be able to later identify unlabeled shark images as fish and unlabeled ocean images as water.

A common use case of supervised learning is to use historical data to predict statistically likely future events. It may use historical stock market information to anticipate upcoming fluctuations, or be employed to filter out spam emails. In supervised learning, tagged photos of dogs can be used as input data to classify untagged photos of dogs.

Unsupervised Learning

In unsupervised learning, data is unlabeled, so the learning algorithm is left to find commonalities among its input data. As unlabeled data are more abundant than labeled data, machine learning methods that facilitate unsupervised learning are particularly valuable.

The goal of unsupervised learning may be as straightforward as discovering hidden patterns within a dataset, but it may also have a goal of feature learning, which allows the computational machine to automatically discover the representations that are needed to classify raw data.

Unsupervised learning is commonly used for transactional data. You may have a large dataset of customers and their purchases, but as a human you will likely not be able to make sense of what similar attributes can be drawn from customer profiles and their types of purchases. With this data fed into an unsupervised learning algorithm, it may be determined that women of a certain age range who buy unscented soaps are likely to be pregnant, and therefore a marketing campaign related to pregnancy and baby products can be targeted to this audience in order to increase their number of purchases.

Without being told a “correct” answer, unsupervised learning methods can look at complex data that is more expansive and seemingly unrelated in order to organize it in potentially meaningful ways. Unsupervised learning is often used for anomaly detection including for fraudulent credit card purchases, and recommender systems that recommend what products to buy next. In unsupervised learning, untagged photos of dogs can be used as input data for the algorithm to find likenesses and classify dog photos together.

Approaches

As a field, machine learning is closely related to computational statistics, so having a background knowledge in statistics is useful for understanding and leveraging machine learning algorithms.

For those who may not have studied statistics, it can be helpful to first define correlation and regression, as they are commonly used techniques for investigating the relationship among quantitative variables. Correlation is a measure of association between two variables that are not designated as either dependent or independent. Regression at a basic level is used to examine the relationship between one dependent and one independent variable. Because regression statistics can be used to anticipate the dependent variable when the independent variable is known, regression enables prediction capabilities. Approaches to machine learning are continuously being developed. For our purposes, we’ll go through a few of the popular approaches that are being used in machine learning at the time of writing. k-nearest neighbor

The k-nearest neighbor algorithm is a pattern recognition model that can be used for classification as well as regression. Often abbreviated as k-NN, the k in k-nearest neighbor is a positive integer, which is typically small. In either classification or regression, the input will consist of the k closest training examples within a space.

We will focus on k-NN classification. In this method, the output is class membership. This will assign a new object to the class most common among its k nearest neighbors. In the case of k = 1, the object is assigned to the class of the single nearest neighbor.

Let’s look at an example of k-nearest neighbor. In the diagram below, there are blue diamond objects and orange star objects. These belong to two separate classes: the diamond class and the star class.

When a new object is added to the space — in this case a green heart — we will want the machine learning algorithm to classify the heart to a certain class.

When we choose k = 3, the algorithm will find the three nearest neighbors of the green heart in order to classify it to either the diamond class or the star class.

In our diagram, the three nearest neighbors of the green heart are one diamond and two stars. Therefore, the algorithm will classify the heart with the star class.

Among the most basic of machine learning algorithms, k-nearest neighbor is considered to be a type of “lazy learning” as generalization beyond the training data does not occur until a query is made to the system.

Decision Tree Learning

For general use, decision trees are employed to visually represent decisions and show or inform decision making. When working with machine learning and data mining, decision trees are used as a predictive model. These models map observations about data to conclusions about the data’s target value.

The goal of decision tree learning is to create a model that will predict the value of a target based on input variables.

In the predictive model, the data’s attributes that are determined through observation are represented by the branches, while the conclusions about the data’s target value are represented in the leaves.

When “learning” a tree, the source data is divided into subsets based on an attribute value test, which is repeated on each of the derived subsets recursively. Once the subset at a node has the equivalent value as its target value has, the recursion process will be complete.

Let’s look at an example of various conditions that can determine whether or not someone should go fishing. This includes weather conditions as well as barometric pressure conditions.

In the simplified decision tree above, an example is classified by sorting it through the tree to the appropriate leaf node. This then returns the classification associated with the particular leaf, which in this case is either a Yes or a No. The tree classifies a day’s conditions based on whether or not it is suitable for going fishing.

A true classification tree data set would have a lot more features than what is outlined above, but relationships should be straightforward to determine. When working with decision tree learning, several determinations need to be made, including what features to choose, what conditions to use for splitting, and understanding when the decision tree has reached a clear ending.

Introduction to Deep Learning

What is deep learning Deep learning is a branch of machine learning which is completely based on artificial neural networks, as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. In deep learning, we don’t need to explicitly program everything. The concept of deep learning is not new. It has been around for a couple of years now.

It’s on hype nowadays because earlier we did not have that much processing power and a lot of data. As in the last 20 years, the processing power increases exponentially, deep learning and machine learning came in the picture.

A formal definition of deep learning is- neurons

Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones.

In human brain approximately 100 billion neurons all together this is a picture of an individual neuron and each neuron is connected through thousand of their neighbours. The question here is how do we recreate these neurons in a computer. So, we create an artificial structure called an artificial neural net where we have nodes or neurons. We have some neurons for input value and some for output value and in between, there may be lots of neurons interconnected in the hidden layer.

7.3 CODING

```
From Main File import
numpy as np import imutils
import time from scipy
import spatial import cv2
from input_retrieval import
*
list_of_vehicles = ["bicycle","car","motorbike","bus","truck", "train"]

FRAMES_BEFORE_CURRENT = 10

inputWidth, inputHeight = 416, 416

LABELS, weightsPath, configPath, inputVideoPath, outputVideoPath,\
        preDefinedConfidence, preDefinedThreshold, USE_GPU=
parseCommandLineArguments()
np.random.seed(42)

COLORS    =    np.random.randint(0,    255,    size=(len(LABELS),    3),
        dtype="uint8")
def        displayVehicleCount(frame,        vehicle_count):
        cv2.putText(
                                        frame, #Image

                                                            'Detected Vehicles: ' + str(vehicle_count), #Label

                              (20, 20), #Position
                    cv2.FONT_HERSHEY_SIMPLEX, #Font
                                        0.8, #Size

                              (0,  0xFF,  0),  #Color
                              2, #Thickness
                                                            cv2.FONT_HERSHEY_COMPLEX_SMALL,

                                        )
def    boxAndLineOverlap(x_mid_point,    y_mid_point,    line_coordinates):    x1_line,
          y1_line, x2_line, y2_line = line_coordinates #Unpacking
          if (x_mid_point >= x1_line and x_mid_point <= x2_line+5) and\

                              (y_mid_point >= y1_line and y_mid_point <= y2_line+5):

                              return True

                              return False



          def displayFPS(start_time, num_frames):
                    current_time = int(time.time())
                    if(current_time > start_time):
                              os.system('clear') # Equivalent of CTRL+L on the terminal
                              print("FPS:", num_frames)              num_frames = 0
                              start_time = current_time       return start_time,
          num_frames def drawDetectionBoxes(idxs, boxes, classIDs,
          confidences, frame):
                    # ensure at least one detection exists
                    if len(idxs) > 0:
                              # loop over the indices we are keeping
                              for i in idxs.flatten():
                                                            # extract the bounding box coordinates

                                                            (x, y) = (boxes[i][0], boxes[i][1])

                                                            (w, h) = (boxes[i][2], boxes[i][3])
                    # draw a bounding box rectangle and label on the frame
                    color = [int(c) for c in COLORS[classIDs[i]]]
                cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)
        text = "{}: {:.4f}".format(LABELS[classIDs[i]],
        confidences[i])
                    cv2.putText(frame, text, (x, y - 5),
        cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)
                                #Draw a green dot in the middle of the box

                                        cv2.circle(frame, (x + (w//2), y+
(h//2)), 2, (0, 0xFF, 0), thickness=2)

def initializeVideoWriter(video_width, video_height, videoStream):

# Getting the fps of the source video
sourceVideofps = videoStream.get(cv2.CAP_PROP_FPS)

# initialize our video writer fourcc = cv2.VideoWriter_fourcc(*"MJPG")
return cv2.VideoWriter(outputVideoPath, fourcc, sourceVideofps,
                            (video_width, video_height), True)


def boxInPreviousFrames(previous_frame_detections, current_box, current_detections):

        centerX, centerY, width, height = current_box
        dist = np.inf #Initializing the minimum distance
        for i in range(FRAMES_BEFORE_CURRENT):
                coordinate_list = list(previous_frame_detections[i].keys())         if
len(coordinate_list) == 0: # When there are no detections in the previous frame
                        continue

                temp_dist, index = spatial.KDTree(coordinate_list).query([(centerX, centerY)])
                if (temp_dist < dist):
                    dist = temp_dist
                frame_num = i
                                coord = coordinate_list[index[0]]

        if (dist > (max(width, height)/2)):
                    return False
```

```
 current_detections[(centerX, centerY)] = previous_frame_detections[frame_num][coord]   return
True


def count_vehicles(idxs, boxes, classIDs, vehicle_count, previous_frame_detections, frame):

                        current_detections = {}

        # ensure at least one detection exists
        if len(idxs) > 0:
                # loop over the indices we are keeping
                for i in idxs.flatten():
                                                # extract the bounding box coordinates

                                                (x, y) = (boxes[i][0], boxes[i][1])

                                                (w, h) = (boxes[i][2], boxes[i][3])


                centerX = x + (w//2)            centerY = y+ (h//2)
        if (LABELS[classIDs[i]] in list_of_vehicles):
                        current_detections[(centerX, centerY)] = vehicle_count

   if (not boxInPreviousFrames(previous_frame_detections, (centerX, centerY, w, h),
current_detections)):
                                        vehicle_count += 1

                        ID = current_detections.get((centerX, centerY))
                        if (list(current_detections.values()).count(ID) > 1):
                                                        current_detections[(centerX,
centerY)] = vehicle_count

                                        vehicle_count += 1

                        cv2.putText(frame, str(ID), (centerX, centerY),\
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, [0,0,255], 2)


                        return vehicle_count, current_detections

print("[INFO] loading YOLO from disk...") net =
cv2.dnn.readNetFromDarknet(configPath, weightsPath)
```

```
#Using GPU if flag is passed if
USE_GPU:
        net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)
        net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA)


ln = net.getLayerNames() ln = [ln[i[0] - 1] for i in
net.getUnconnectedOutLayers()]
videoStream = cv2.VideoCapture(inputVideoPath)
video_width = int(videoStream.get(cv2.CAP_PROP_FRAME_WIDTH)) video_height
= int(videoStream.get(cv2.CAP_PROP_FRAME_HEIGHT))


# Specifying coordinates for a default line
x1_line = 0 y1_line = video_height//2
x2_line = video_width y2_line
= video_height//2


#Initialization
previous_frame_detections = [{(0,0):0} for i in range(FRAMES_BEFORE_CURRENT)]
num_frames, vehicle_count = 0, 0 writer = initializeVideoWriter(video_width,
video_height, videoStream) start_time = int(time.time())
# loop over frames from the video file stream while
True:
 print("================NEW          FRAME================")
num_frames+= 1  print("FRAME:\t", num_frames)
        # Initialization for each iteration
        boxes, confidences, classIDs = [], [], []
        vehicle_crossed_line_flag = False


        #Calculating fps each second          start_time, num_frames =
displayFPS(start_time, num_frames)
                                # read the next frame from the file
                                (grabbed, frame) = videoStream.read()
        # if the frame was not grabbed, then we have reached the end of the stream
        if not grabbed:
                break


                                            blob = cv2.dnn.blobFromImage(frame, 1 /
255.0, (inputWidth, inputHeight),
            swapRB=True,
crop=False)  net.setInput(blob)    start
= time.time()  layerOutputs =
net.forward(ln)     end = time.time()


        # loop over each of the layer outputs
        for output in layerOutputs:
            # loop over each of the detections
            for i, detection in enumerate(output):
                                            # extract the class ID and confidence (i.e.,
probability)
                # of the current object detection
                scores = detection[5:]
            classID = np.argmax(scores)
        confidence = scores[classID]
  if confidence > preDefinedConfidence:


  box = detection[0:4] * np.array([video_width, video_height, video_width, video_height])
                                                (centerX, centerY, width, height) =
box.astype("int")
                    # use the center (x, y)-coordinates to derive the
top                 # and and left corner of the bounding box
                x = int(centerX - (width / 2))
        y = int(centerY - (height / 2))
```

```
                                        boxes.append([x, y, int(width), int(height)])
            confidences.append(float(confidence))
            classIDs.append(classID)
                                                        idxs = cv2.dnn.NMSBoxes(boxes, confidences,
preDefinedConfidence,

                                        preDefinedThreshold)

                                # Draw detection box

                                                        drawDetectionBoxes(idxs, boxes, classIDs,
confidences, frame)


            vehicle_count, current_detections = count_vehicles(idxs, boxes, classIDs, vehicle_count,
previous_frame_detections, frame)


                                                # Display Vehicle Count if a vehicle has passed the line
                                        displayVehicleCount(frame, vehicle_count)


   # write the output frame to disk

                                writer.write(frame)

            cv2.imshow('Frame', frame)   if
cv2.waitKey(1) & 0xFF == ord('q'):
                                break

            # Updating with the current frame detections
            previous_frame_detections.pop(0) #Removing the first frame from the list
            # previous_frame_detections.append(spatial.KDTree(current_detections))
            previous_frame_detections.append(current_detections)


# release the file pointers
print("[INFO] cleaning up...")
writer.release() videoStream.release()
```

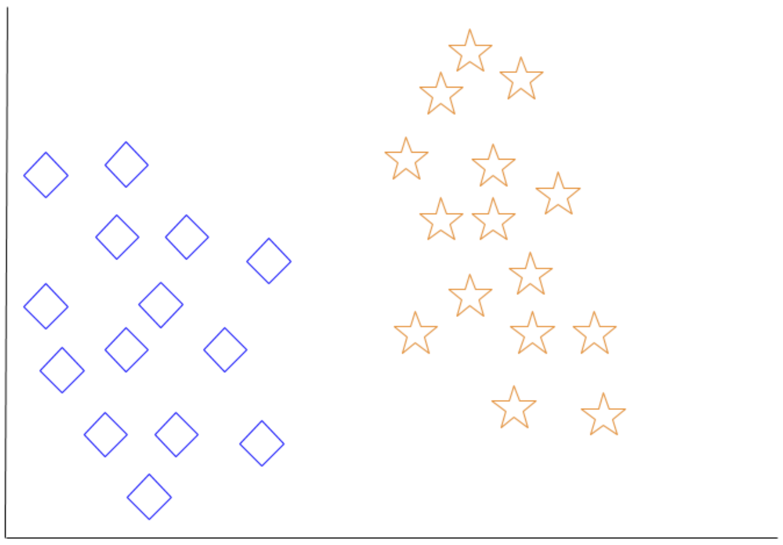
CHAPTER 8

IMPLEMENTATION

8.1 Module description with algorithm / Pseudocode

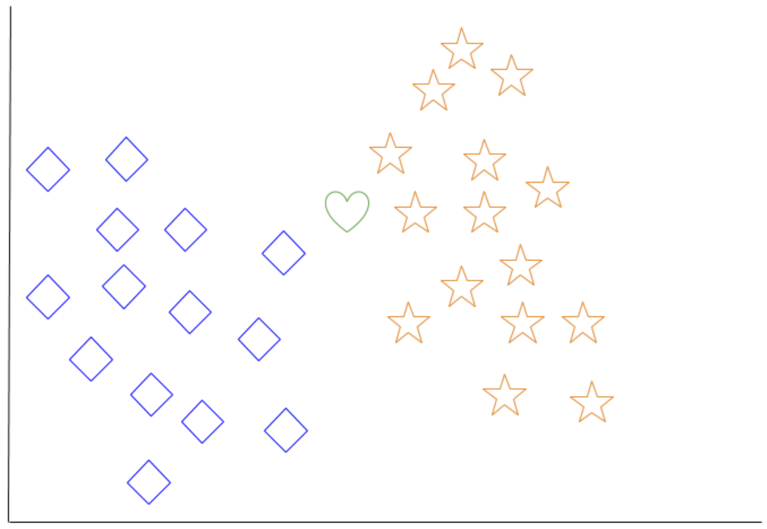
Modules Description Background Learning Module

This is the first module in the system whose main purpose is to learn about the background in a sense that how it is different from the foreground. Furthermore as proposed system works on a video feed, this module extracts the frames from it and learns about the background. In a traffic scene captured with a static camera installed on the road side, the moving objects can be considered as the foreground and static objects as the background. Image processing algorithms are used to learn about the background using the above mentioned technique.



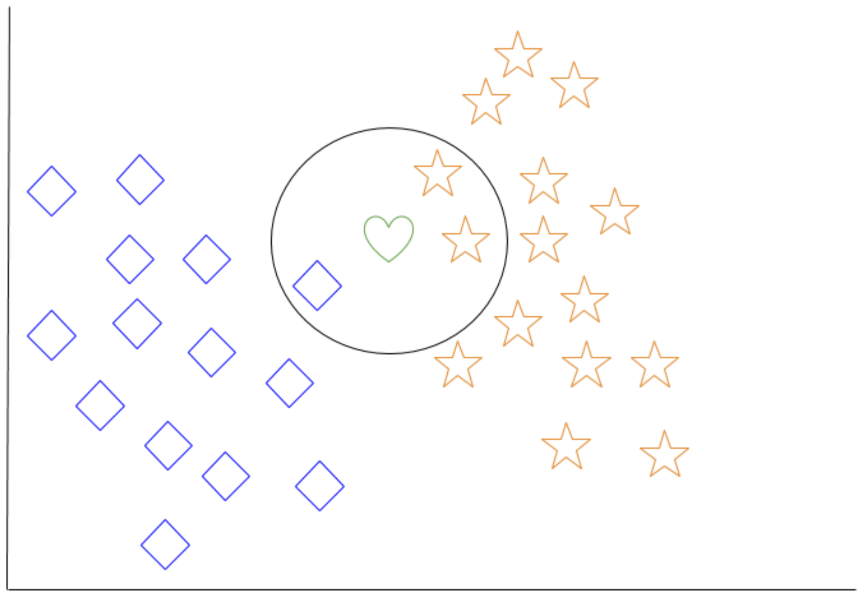
Foreground Extraction Module

This module consists of three steps, background subtraction, image enhancement and foreground extraction. Background is subtracted so that foreground objects are visible. This is done usually by static pixels of static objects to binary 0. After background subtraction image enhancement techniques such as noise filtering, dilation and erosion are used to get proper contours of the foreground objects. The final result obtained from this module is the foreground.



Vehicle Classification Module

The third and the last module in the proposed system is classification. After applying foreground extraction module, proper contours are acquired. Features of these contours such as centroid, aspect ratio, area, size and solidity are extracted and are used for the classification of the vehicles.



CHAPTER 9

TESTING

9.1 TESTING METHODOLOGIES

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable mKNNer. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at   exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cKNNot be reached from a black box level

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cKNNot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.
The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

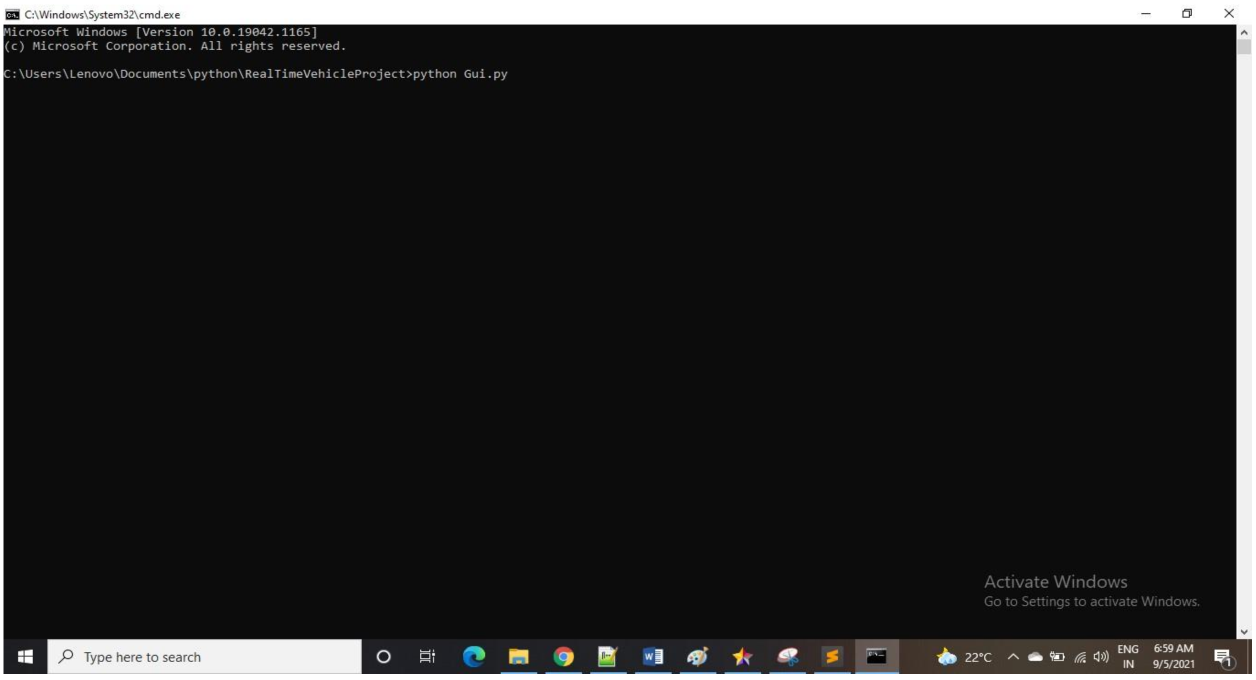
Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER 10

RESULTS


C:\Windows\System32\cmd.exe
Activate Windows
Go to Settings to activate Windows.
Type here to search
22°C
ENG IN
6:59 AM
9/5/2021

```
Microsoft Windows [Version 10.0.19042.1165]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lenovo\Documents\python\RealTimeVehicleProject>python Gui.py
```

Fig 8: Running the GUI file

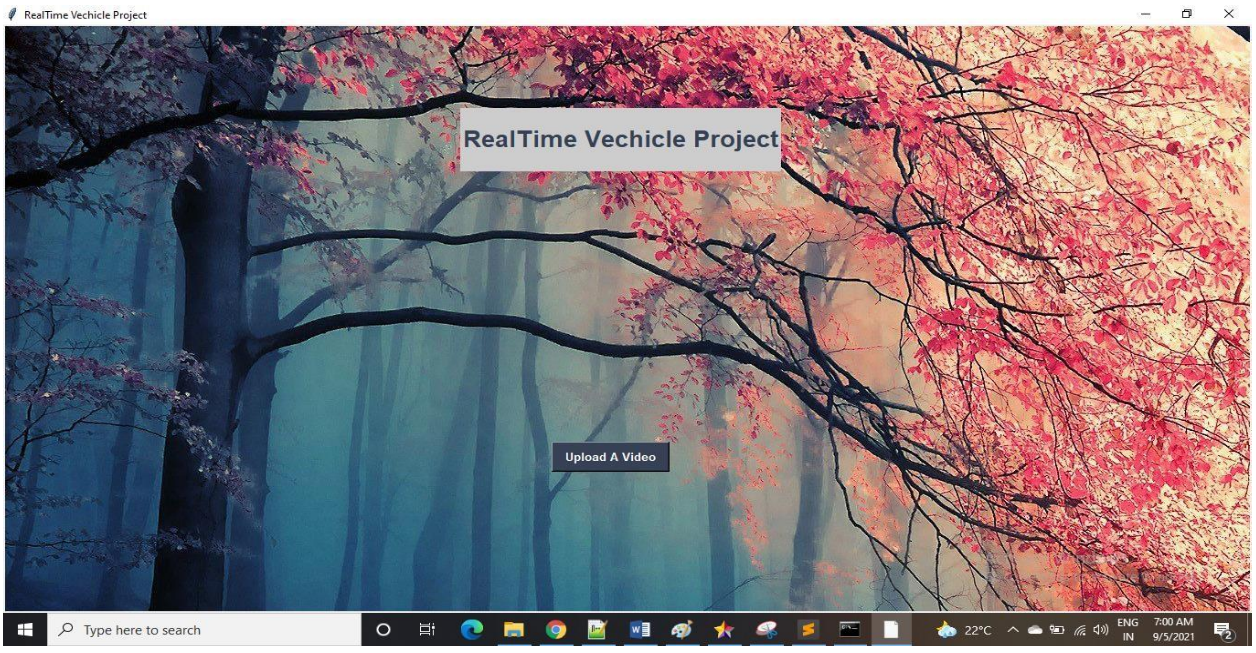


Fig 9: Interface of the project

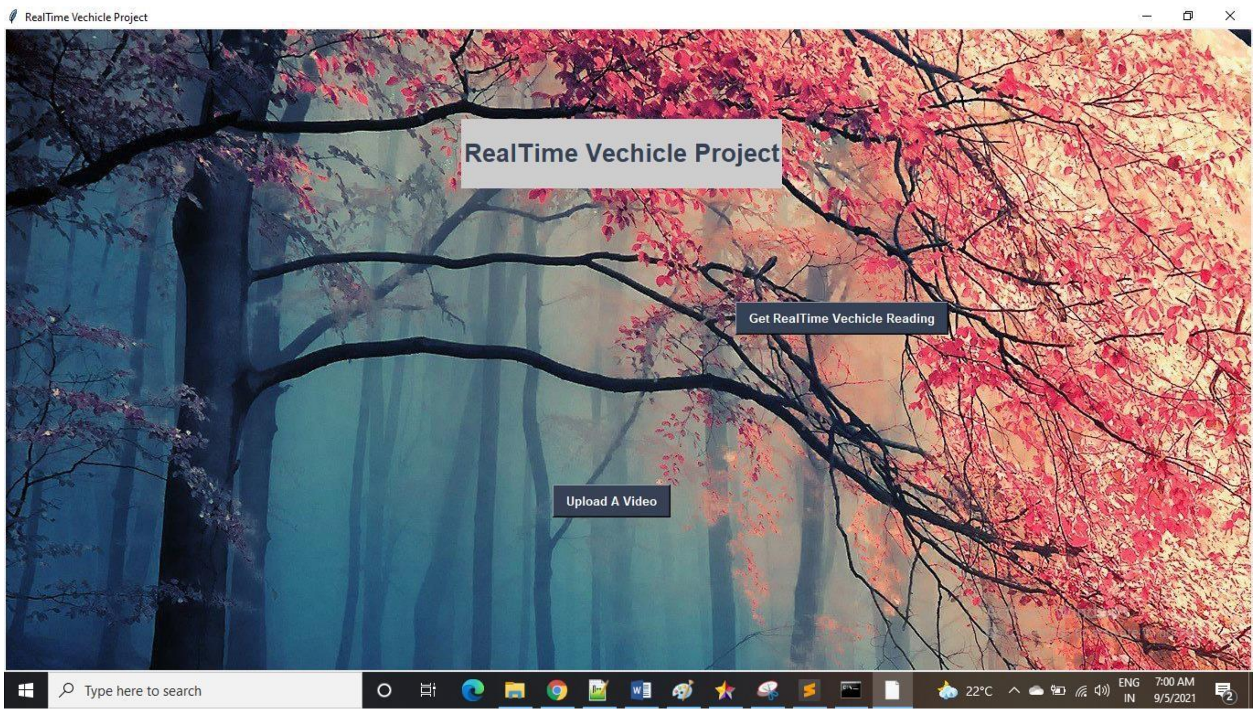
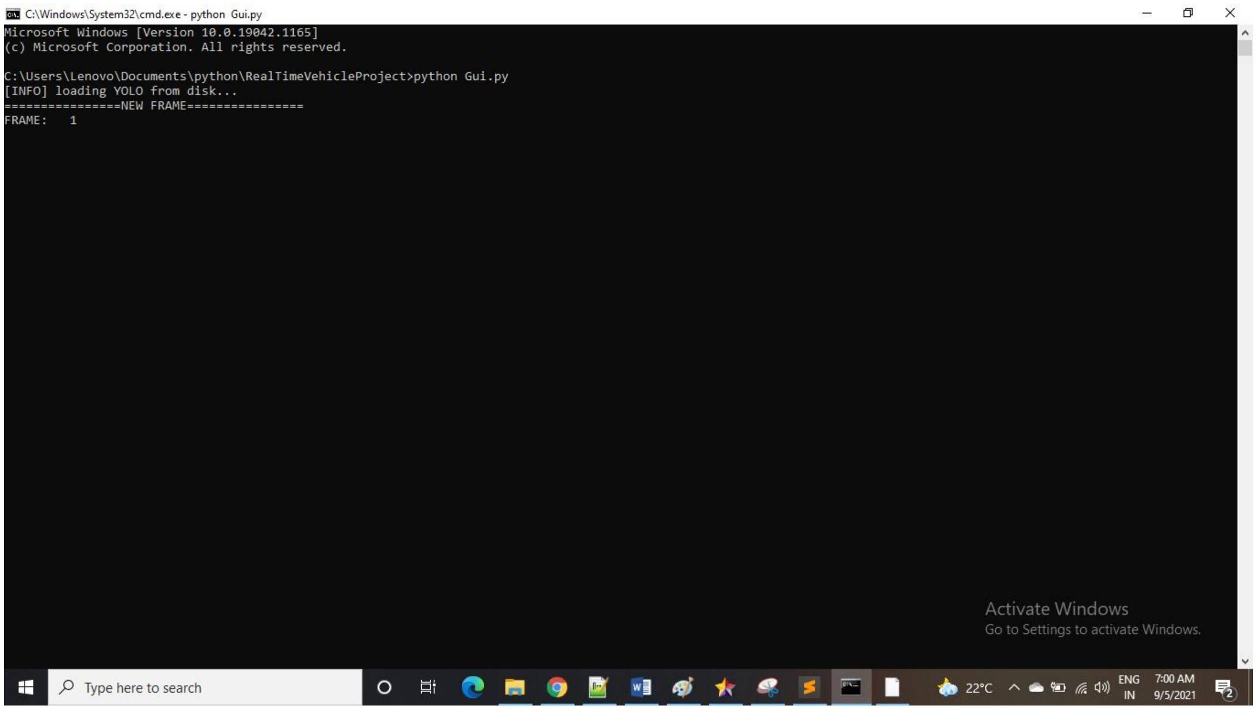

RealTime Vechicle Project
RealTime Vechicle Project
Get RealTime Vechicle Reading
Upload A Video
Type here to search
22°C
ENG
IN
7:00 AM
9/5/2021

Fig 10: Interface after uploading the file


C:\Windows\System32\cmd.exe - python Gui.py

```
Microsoft Windows [Version 10.0.19042.1165]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lenovo\Documents\python\RealTimeVehicleProject>python Gui.py
[INFO] loading YOLO from disk...
================NEW FRAME================
FRAME:   1
```

Activate Windows
Go to Settings to activate Windows.
Type here to search
22°C
7:00 AM
9/5/2021

Fig 11: Loading the YOLO

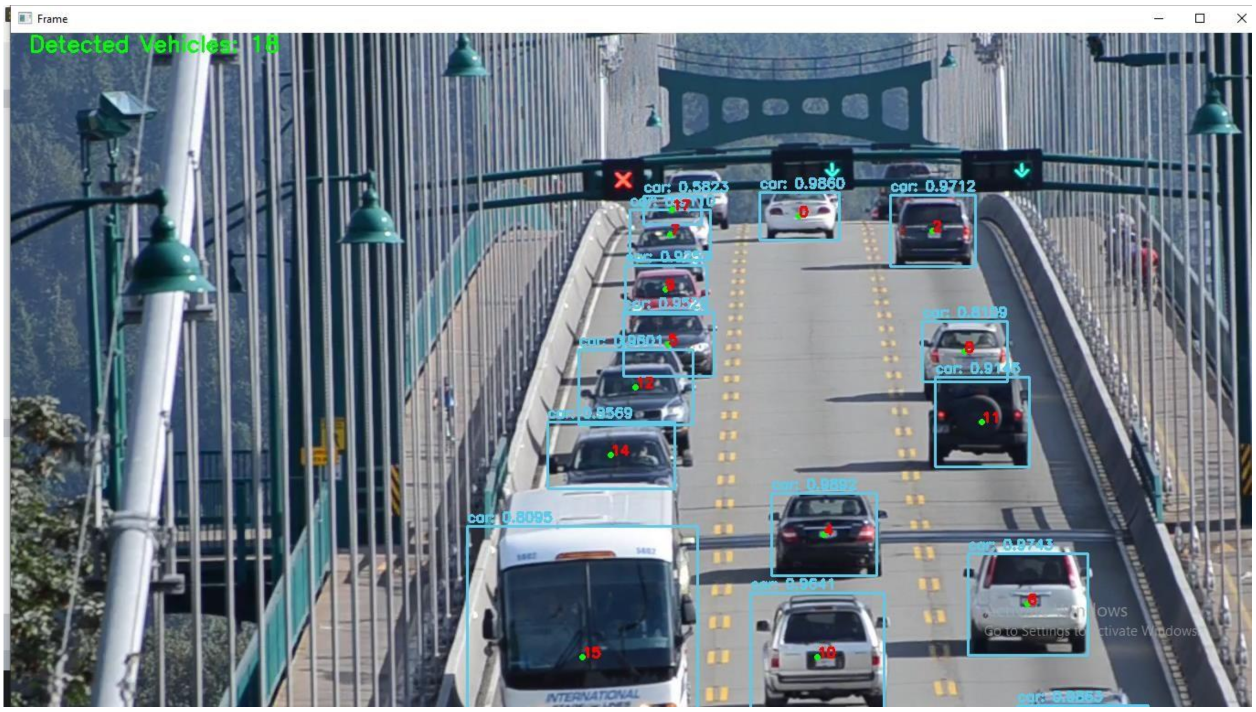


Fig 12: Output video after detection

CHAPTER 11

CONCLUSION

The proposed solution is implemented on python, using the OpenCV bindings. The traffic camera footages from variety of sources are in implementation. A simple interface is developed for the user to select the region of interest to be analyzed and then image processing techniques are applied to calculate vehicle count and classified the vehicles. We have developed video based vehicle detection, classification, counting for real-time traffic data collection. We have used Background Subtraction Yolo algorithm, OpenCV, and python for developing the system. In the proposed system, we have considered all day and night shadowing, and different lighting situations. Also, we have considered the moving shadow of moving vehicles.

CHAPTER 12

FUTURE ENHANCEMENT

In future, we will collect more traffic data from different locations of Hyderabad city using our proposed system and will apply several machine learning algorithms for mining traffic patterns of Hyderabad city.

REFERENCES

[1] I. Alam, M. F. Ahmed, M. Alam, J. Ulisses, D. M. Farid, S. Shatabda, and R. J. F. Rossetti, "Pattern mining from historical traffic big data," in IEEE Technologies for Smart Cities (TENSYMP 2017). IEEE, July 2017, pp. 1–5.

[2] I. Alam, D. M. Farid, and R. J. F. Rossetti, "The prediction of traffic flow with regression analysis," in Emerging Technologies in Data Mining and Information Security, ser. Advances in Intelligent Systems and Computing, A. A., D. P., M. J., B. A., and D. S., Eds. Springer, Singapore, 2019, vol. 813, pp. 661–671.

[3] A. Talebpour, H. S. Mahmassani, and S. H. Hamdar, "Effect of information availability on stability of traffic flow: Percolation theory approach," Transportation Research Procedia, vol. 23, pp. 81–100, 2017.

[4] M. Dell'Orco and M. Marinelli, "Modeling the dynamic effect of information on drivers' choice behavior in the context of an advanced traveler information system," Transportation Research Part C: Emerging Technologies, vol. 85, pp. 168–183, 2017.

[5] A. Csikos, T. Charalambous, H. Farhadi, B. Kulcs ´ ar, and H. Wymeersch, ´ "Network traffic flow optimization under performance constraints," Transportation Research Part C: Emerging Technologies, vol. 83, pp. 120–133, 2017.

[6] M. Zhou, X. Qu, and X. Li, "A recurrent neural network based microscopic car following model to predict traffic oscillation," Transportation Research Part C: Emerging Technologies, vol. 84, pp. 245–264, 2017.

[7] F. B. Ghorghi and H. Zhou, “Traffic control devices for deterring wrongway driving: historical evolution and current practice,” Journal of Traffic and Transportation Engineering, vol. 4, pp. 280–289, 2017.

[8] A. Abadi, T. Rajabioun, and P. A. Ioannou, “Traffic flow prediction for road transportation networks with limited traffic data,” IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 2, pp. 653–662, 2015.

[9] S.-Y. Cheung, and P.P. Varaiya, “Traffic surveillance by wireless sensor networks: Final report”, PhD diss., University of California at Berkeley, 2006.

[10] S. Oh, S. Ritchie, and C. Oh, “Real-time traffic measurement from single loop inductive signatures”, Transportation Research Record: Journal of the Transportation Research Board, (1804), pp. 98-106, 200

REAL TIME VEHICLE DETECTION AND COUNTING USING ML

Mr. N V SRINIVASAN [1], S K REHMAN [2], M HARISH BABU [3], M MANJULA [4], P PRAVEEN [5], P TEJASWINI [6]

Assistant Professor [1], Student [2,3,4,5,6]

Department of Computer Science and Engineering, Mother Theresa Institute of Engineering and Technology,

Palamaner - 517408, Chittoor District, Andhra Pradesh[1,2,3,4,5,6]

Abstract—Traffic Analysis has been a problem that city planners have dealt with for years. Smarter ways are being developed to analyze traffic and streamline the process. Analysis of traffic may account for the number of vehicles in an area per some arbitrary time period and the class of vehicles. People have designed such mechanism for decades now but most of them involve use of sensors to detect the vehicles i.e. a couple of proximity sensors to calculate the direction of the moving vehicle and to keep the vehicle count. Even though over the time these systems have matured and are highly effective, they are not very budget friendly. The problem is such systems require maintenance and periodic calibration. Therefore, this study has purposed a vision based vehicle counting and classification system. The system involves capturing of frames from the video to perform background subtraction in order detect and count the vehicles using Gaussian Mixture Model (GMM) background subtraction then it classifies the vehicles by comparing the contour areas to the assumed values. The substantial contribution of the work is the comparison of two classification methods. Classification has been implemented using Contour Comparison (CC) as well as Bag of Features (BoF) and Support Vector Machine (SVM) method.

Key words—Video surveillance, detection, video classification, Gaussian Mixture Model, Bag of Features, Support Vector Machine.

I. INTRODUCTION

The need of efficient management and monitoring of road traffic has increased in last few decades because of the increase in the road networks, the number and most importantly the size of vehicles. Intelligent traffic surveillance systems are very important part of modern day traffic management but the regular traffic management techniques such as wireless sensor networks[1], Inductive loops[2] and EM microwave detectors[3] are expensive, bulky and are difficult to install without interrupting the traffic. A good alternative to these techniques can be video based surveillance systems[4].

Video surveillance systems[4-8] have become cheaper and better because of the increase in the storage capabilities, computational power and video encryption algorithms[9]. The videos stored by these surveillance systems are generally analyzed by humans, which is a time consuming Job. To overcome this constraint, the need of more robust, automatic video based surveillance systems has increased interest in field of computer vision.

The objectives of a traffic surveillance system is to detect, track and classify the vehicles but they can be used to do complex tasks such as driver activity recognition, lane recognition etc. The traffic surveillance systems can have applications in a range of fields such as, public security, detection of anomalous behavior, accident detection, vehicle theft detection, parking areas, and person identification. A Traffic surveillance system usually contains two parts, hardware and software. Hardware is a static camera installed on the roadside that captures the video feed and the software part of the system is concerned with processing and analyses. These systems could be portable with a microcontroller attached to the camera for the real-time processing and analyses or just the cameras that transmit the video feed to a centralized computer for further processing.

II. RELATED WORK

Various approaches were made to develop such systems that can detect, count and classify the vehicles and can be used for traffic surveillance in intelligent transportation systems. This section covers the discussion about such systems and the knowledge about the methods used to develop such systems.

Tursun, M and Amrulla, G [4] proposed a video based real-time vehicle counting system using optimized virtual loop method. They used real time traffic surveillance cameras deployed over roads and compute how many vehicles pass the road. In this system counting is completed in three steps by tracking vehicle movements within a tracking zone called virtual loop. Another video based vehicle counting system was proposed by Lei, M., et al. [5]. In this system surveillance cameras were used and mounted at relatively high place to acquire the traffic video stream, the Adaptive background estimation and the Gaussian shadow elimination are the two main methods that were used in this system. The accuracy rate of the system depends on the visual angle and ability to remove shadows and ghost effects. The system's incompetency to classify vehicle type is the core limitation of the system.

Bas et al. proposed a video analysis method to count vehicles [10] based on an adaptive bounding box size to detect and track vehicles in accordance with estimated distance from the camera. The Region of Interest (ROI) is identified by defining a boundary for each outbound and inbound in the image. Although the algorithm is improved to deal with some weather conditions it is unable to track vehicles when they change their directions.

Mithun, N.C., et al proposed a vehicle detection and classification system using time spatial image and multiple virtual detection line[6]. A two-step K nearest neighborhood (KNN) algorithm is adopted to classify vehicles via shape invariant and texture based features. Experiments confirm the better accuracy and low error rate of proposed method over existing methods since it also considers the various illumination conditions.

Habibu Rabiu proposed a vehicle detection and classification for cluttered urban intersection [11]. In this system background subtraction and kalman filter algorithm are used to detect and track the vehicles and Linear Discriminant Analysis classifier is used for proper classification of vehicles.

Detection of vehicles in a video based traffic surveillance system is first and very important phase as it greatly impacts the other algorithms such as tracking and classification of the vehicles hence an accurate detection and segmentation of the foreground moving object is very important. Many of the techniques are used for foreground detection like frame differencing [12]. Frame differencing can be considered as the simplest foreground detection and segmentation method as it is based on the close relationship among the sequence of motion images.

An improved frame differencing method was presented by Collins [7] which uses difference between multiple frames to compute the foreground instead of just using the initial frame. Another method named as Optical Flow Field method was brought out by Gibson[13]. Wu, K, et al. proposed that the optical flow represents the velocity of mode within an image [14].Optical Flow method takes image within the detecting area as a vector field of velocity; each vector represents the transient variation of the position for a pixel in the scenery. Another method used to detect foreground is average model [8]. In average model the average grey value of a pixel in a sequence of frames is considered as the background value of same pixel. A GMM was proposed by Friedman, N. and S.

Russell.[15] and was refined for real time tracking by Stauffer, C. and W.E.L. Grimson[16, 17]. Gaussian Mixture Model relies on assumptions that the background is visible more frequently than any foreground regions. Elgammal proposed Kernel density estimation based nonparametric background model [18].

Kernel density estimation method evaluates the samples of video data using kernel functions and it samples data which has maximal probability density as background is selected. A bag of features model is the one which represents images as order less collections of local features [19]. The name bag of features came from the bag of words representation used in text based information retrieval. Scale Invariant Feature Transform algorithm (SIFT) was introduced by David Lowe in 1999 [20] and refined in 2004 [21]. SIFT is used to extract the features from dataset. In SIFT features are invariant to image scaling, rotation and partially invariant to change in illumination and 3D camera viewpoint [21]. SIFT with SVM requires less number of dataset. With any supervised learning model, first SVM is trained to cross validate the classifier. Then trained machine is used to make predictions and classify the data [22]. Bhushan et al. [23], proposed a vehicle detection method based on morphological operations including binarization, edge detection, and top-hat processing, masking operation. Proposed system fails to give good results n cloudy environment. Raứl et al. [24], proposed a classification technique for moving objects in which information processing is employed via clustering and classification algorithms. Proposed methodology provides sufficient accuracy for it to be employed for realtime applications but still the system can be further improved for vehicle classification in complex scenarios such as weather and illumination conditions. A. Suryatali and V.B. Dharmadhikari in [25], proposed a Computer Vision based vehicle detection that counts and also classifies the vehicles in to heavy and light weight vehicles; object detection is accomplished by making use of kalman filter for background subtraction and then openCV library is used finally to detect the object in processed frame. Nilakorn et al. [26], proposed vehicle detection and counting prototype which uses different steps for background subtraction and object detection then uses CV techniques such as thresholding, hole-filling and adaptive morphological dilation to remove noise and enhance the foreground objects in particular frame from video. Proposed system provides limited functionality for the objects appearing in detection zone if they are occluded or small.

III. PROPOSED METHODOLOGY

The system could be used for detection, recognition and tracking of the vehicles in the video frames and then classify the detected vehicles according to their size in three different classes. The proposed system is based on three modules which are background learning, foreground extraction and vehicle classification as shown in fig. 1. Background subtraction is a classical approach to obtain the foreground image or in other words to detect the moving objects.

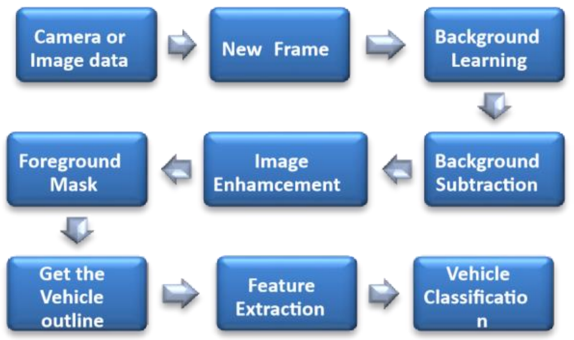


Fig.1. Block diagram of proposed vehicle detection, counting and classification system.

A. *Background Learning Module*

This is the first module in the system whose main purpose is to learn about the background in a sense that how it is different from the foreground. Furthermore as proposed system works on a video feed, this module extracts the frames from it and learns about the background. In a traffic scene captured with a static camera installed on the road side, the moving objects can be considered as the foreground and static objects as the background. Image processing algorithms are used to learn about the background using the above mentioned technique.

B. *Foreground Extraction Module*

This module consists of three steps, background subtraction, image enhancement and foreground extraction. Background is subtracted so that foreground objects are visible. This is done usually by static pixels of static objects to binary
0. After background subtraction image enhancement techniques such as noise filtering, dilation and erosion are used to get proper contours of the foreground objects. The final result obtained from this module is the foreground.

C. Vehicle Classification Module

The third and the last module in the proposed system is classification. After applying foreground extraction module, proper contours are acquired. Features of these contours such as centroid, aspect ratio, area, size and solidity are extracted and are used for the classification of the vehicles.

IV. DETAILED METHODOLOGY

The first step of the proposed system is to grab a data video on which we want to perform the classification. After video selection, ROI is defined. ROI needs a careful human supervision because region of interest and imaginary line plays important role in classification.

After ROI is defined, the system performs series of tasks i.e. applying background mask, subtracting mask, performing binary threshold, morphology using erosion and dilation, median blur, applying masked data to the frame, convert frame to gray scale. Contours are detected after these operations. Once contours are detected; system analyses the moments of the contours, marks the detected contours and centroid is calculated. If calculated centroid is in the range of the diagonal, system moves towards further operation for classification else system will be redirected towards the detection of contours again. The last step is the classification; the system classifies the vehicles with two different methods i.e. using SVM and with the CC.

The classification using SVM is used in which SIFT features are calculated for the contours and used as input to the SVM. Three types of vehicles are identified by SVM which are Low Transport Vehicle (LTV), Medium Transport Vehicle (MTV) and Heavy Transport Vehicle (HTV). SVM classifies the vehicle using the features extracted with the help of SIFT and then corresponding variables are incremented according to the output i.e. LTV, MTV and HTV. In addition, classification of the vehicles using CC is done. Once the centroid calculated is in the range of diagonal; the properties of contours are extracted. The features extracted are compared with the assumed values and output is calculated. In the end the corresponding variables are incremented according to the output.

A. *Region of interest*

ROI is a particular portion of an image on which an operation is to be performed. ROI gives the flexibility to just work with in a particular region instead of manipulating the whole image. In proposed system, selection of region of interest is very important to reduce the false positives in the detection and classification of vehicles. Selection of ROI is pretty simple, once the video is started, the user has to press the “I” key on the keyboard to activate the input mode. Afterwards the user uses his mouse to select the four points on the video which defines the region of interest. Once selected, pressing of any key on the keyboard selects the region of interest, crops it and shows the new video feed on only that region. Fig. 2 shows ROI selection input mode. Notice the four green dots on the screen, these are the points defining the ROI and were placed using mouse clicks.

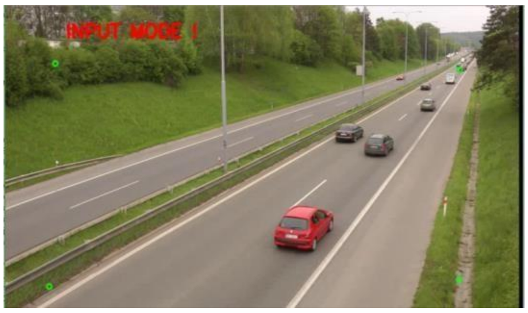


Fig.2. Region of interest selection input mode.

B. *Background subtraction*

Proper background subtraction is a vital pre-processing step in creation of any visual surveillance system as the accuracy of whole process of classification of the objects depends on it. The systems like visitor counter[27] in which a static camera captures the video of people entering a building and the system could count the number or a system where a camera captures the video of the vehicles on the road for the similar purpose.

The background subtraction could be an easy job if we already have an image of the background like the image of the building or the road. In cases defined above, background image could be removed and foreground objects could be obtained but most of the time the situation is varying. The backgrounds can be dynamic or initial information of the scene might not be available. Furthermore, the background subtraction becomes more difficult if the objects in the video also have shadows since they also move with the people or vehicles, then the normal background subtraction will detect the shadows as foreground objects too.

Several algorithms have been introduced for the situations like that; some of them are implemented in openCV such as BackgroundSubtractionMOG [28] which use Gaussian distributions to create the model of the background in the image. It uses about 3 to 5 Gaussian distributions for this purpose. Another background subtraction implemented in openCV is called BackgroundSubtractorGMG which is based on [27] and combines the background image estimation technique with Bayesian segmentation.

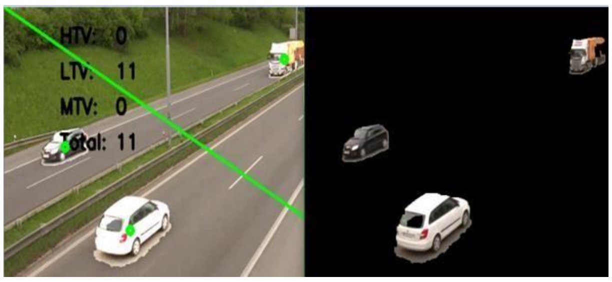


Fig.3. Frame of video before and after subtraction of background.

The algorithm used in the implementation of proposed system is called backgroundSubtractorMOG2. It is based on two studies[29] and [30] by Zikovic. One of the important features of this algorithm is that unlike [28] in which the number of distributions for the creation of background model BackgroundSubtractorMOG2 are uses defined, an automated approach and selects an appropriate number of Gaussian mixtures for the pixel. Furthermore algorithm is also better at handling the illumination changes in the scene. The algorithm also gives the ability to define if the shadow of the objects is to be detected or not. Note that the default settings of the implementation are set to detect the shadows.

C. *Contour extraction*

Contours are the boundaries of the shape which are used for the shape detection and recognition. The accuracy of the process of finding the contours can be defined as the canny edge detection performed on a binary image. OpenCV provide cv2.findContours() method to find the contours.

D. *Counting vehicles*

Vehicle counting can be done by checking if the centroid of a vehicle has touched or crossed the imaginary line in ROI. Imaginary line is the line that diagonally appears by connecting two ROI points. Once the centroid of a vehicle in ROI crosses the imaginary line, the system counts the vehicle. From fig.

4, it is noticeable that when centroid of vehicle touches the imaginary line in ROI, it increments in the variable "Total" and variable of respective category.

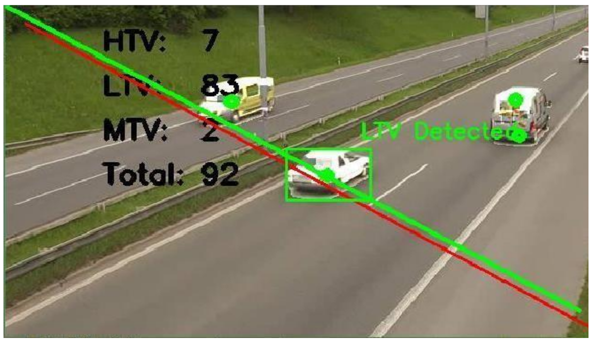


Fig.4. Increment in corresponding variables when vehicle centroid touches imaginary line in ROI (Mask Applied)

E. *Classification*

The classification is done using two different methods in implementation of the system which are: i) Classification using CC ii) BoF and SVM method.

1) Classification using Contour Comparisons

In this method, the contour properties such as solidity and area of the contour are extracted and compared with already assumed values to determine whether the vehicle is LTV, HTV or MTV. The contours are basically all connected pixels in an image. Once the background subtraction is performed and foreground objects are detected; then contours of these foreground object are detected.

We have used openCV's findContours() method to detect the contours of the foreground object. The method gives a list of all the contours in the image. It is necessary to choose the biggest contour in the image. Hence, a minimum limit on the width and height of the contours is defined so that bigger contours can be chosen. Once these contours have been selected, a number of properties of these contours are extracted which can later be used to classify the vehicles. Some of these properties are area, solidity, aspect ratio etc. Particularly, areas of contours are focused which are compared to the assumed values of the vehicles.

First of all a bounding box is drawn over the contour and its centroid is found which when intersects with the imaginary line, the vehicles are detected and classification algorithm is triggered to classify the vehicles. The following values are assumed:

If the area is between 500 and 8000, the vehicle is classified as LTV.

Classification of MTV and HTV is twostep process:

1. If area is between 500 and 125000 the vehicle could be either MTV or HTV depending upon the height.
2. If height of the vehicle is about its width (width-30 to width+30) it is classified as MTV else it is classified as HTV.

2) BoF and SVM method

This is another method of classification which uses BoF along with SVM algorithm to classify the vehicles. The BoF is based on the bag of words algorithm where each word corresponds to each feature of the image saved in the bag. The SVM classifier is created with four classes which are LTV, MTV, HTV and Others. The LTV class contains the features of the small vehicles, MTV contains the medium vehicles and HTV contains the big vehicles, whilst others contain the vehicles that don't fall into any class such as bikes etc. The images of all vehicles are placed in the particular folders as defined above and the SVM classifier is trained over these folders. Once the classifier is trained, it gives a file called BoF.pkl which is used in the code to test the classifier. The system captures the SIFT features of the foreground objects and compares them with the already trained SVM and classifies the vehicles into defined categories.

V. SIMULATION RESULTS

The proposed system is implemented on python, using the openCV bindings. The traffic camera footages from variety of sources have been used and have found the implementation to be effective.

Basically, it is divided into two parts: detection and classification. Five sample videos of different traffic scenes are used for comparison of the system results with true values. Fig. 5 presents the comparison of total vehicles counted by both CC and BoF and SVM methods with the true values. It is observable that CC method is showing smallest discrepancy with counts of ground truth related to video 2 and counts related to video 1 are showing highest discrepancy. BoF and SVM is depicting smallest variation with ground truth count related to video 3 and showing highest variation with counts related to video 1.

Fig. 6 to Fig. 10 compares the simulation results for the classification. The proposed system classifies the detected vehicles into three categories i.e. LTV, MTV and HTV using two different methods. First one deploys the contour area and assumed area of vehicles for the comparison and second method deploys the BoF and SVM for classification. From fig. 6 to fig. 10, it is evident that in all the videos; CC method classifies LTV, MTV and HTV very well as compared to BoF and SVM method except for video 3. The values of CC method are more close to ground truth values than BoF and SVM method.

The results concerning the classification errors of the both methods which are employed in the experiments for different video sequences of road traffic are shown in Table I. From the table it is apparent that classification errors of CC method are lower as compared to BoF and SVM method in all videos.

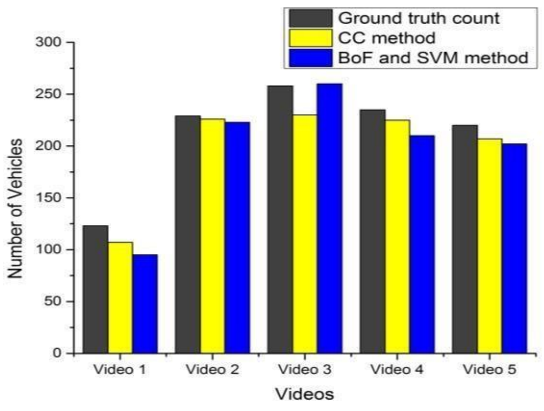


Fig.5. Comparison of original count and system count values of five videos

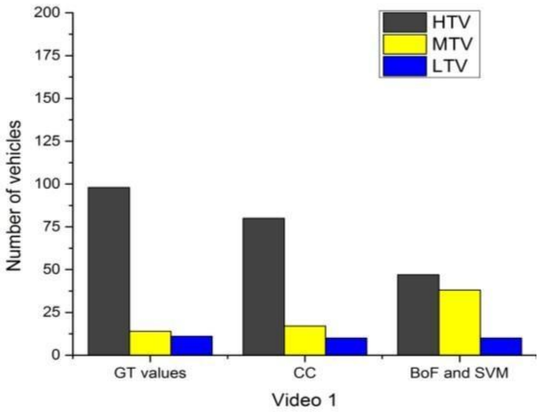


Fig.6. Vehicle classification comparisons between original count, CC, SVM and BoF methods in video 1

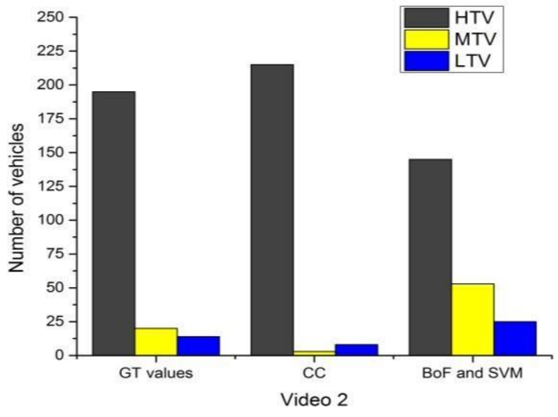


Fig.7. Vehicle classification comparisons between original count, CC, SVM and BoF methods in video 2

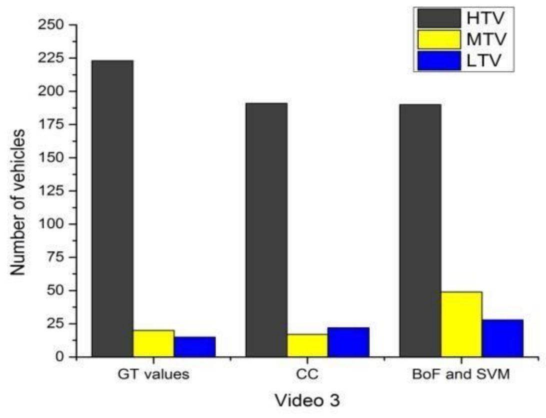


Fig.8. Vehicle classification comparisons between original count, CC, SVM and BoF methods in video 3

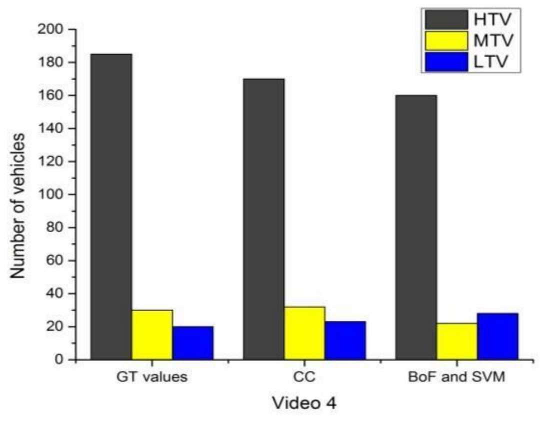


Fig.9. Vehicle classification comparisons between original count, CC, SVM and BoF methods in video 4

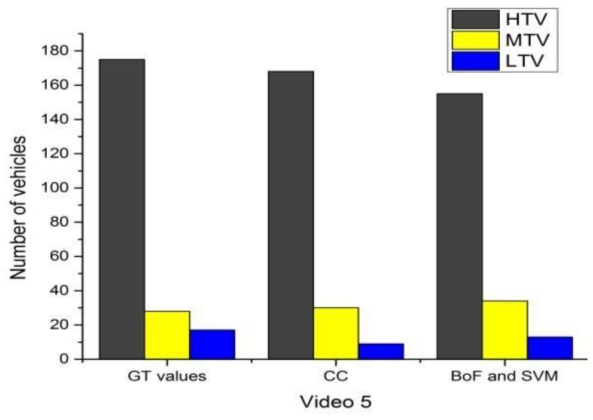


Fig.10. Vehicle classification comparisons between original count, CC, SVM and BoF methods in video 5

Table 1. Vehicle classification errors of CC, SVM and BoF methods

	Ground truth count	Classification Error % (CC)	Classification Error % (BoF and SVM)
Video 1	123	13	22.76
Video 2	229	1.3	2.6
Video 3	258	10.85	0.7
Video 4	235	4.25	10.6
Video 5	220	5.9	8.18

VI. CONCLUSION AND FUTURE WORK

The proposed solution is implemented on python, using the OpenCV bindings. The traffic camera footages from variety of sources are in implementation. A simple interface is developed for the user to select the region of interest to be analyzed and then image processing techniques are applied to calculate vehicle count and classified the vehicles using machine learning algorithms. From experiments it is apparent that CC method outperforms than BoF and SVM method in all results and gives more close classification results to the ground truth values.

Currently proposed system works with already captured videos but it can be modified to be used for processing live microcontrollers. video streams[4] by adding One of the limitations of the system is that it is not efficient at detection of occlusion of the vehicles which affects the accuracy of the counting as well as classification. This problem could be solved by introducing the second level feature classification such as the classification on the bases of color. Another limitation of the current system is that it needs human supervision for defining the region of interest. The user has to define an imaginary line where centroid of the contours intersects for the counting of vehicles hence the accuracy is dependent on the judgment of the human supervisor. Furthermore the camera angle also affects the system hence camera calibration techniques could be used for the detection of the lane for the better view of the road and increasing the efficiency. The system is not capable of detection of vehicles in the night as it needs the foreground objects to be visible for extraction of contour properties as well as features for the classification using SIFT features[31].The system could also be improved for better accuracy using the more sophisticated image segmentation and artificial intelligence operations.

ACKNOWLEDGMENT

We are incredibly thankful to Mr. Jakub Sochor, a Ph.D. student at department of computer graphics and multimedia, BRNO University of technology Czech Republic for providing us access to a phenomenal dataset for conducting our research. We highly appreciate every sort of assistance provided by Mehran UET, Jamshoro.

REFERENCES

[1] S.-Y. Cheung, and P.P. Varaiya, "Traffic surveillance by wireless sensor networks: Final report", PhD diss., University of California at Berkeley, 2006.

[2] S. Oh, S. Ritchie, and C. Oh, "Real-time traffic measurement from single loop inductive signatures", Transportation Research Record: Journal of the Transportation Research Board, (1804), pp. 98-106, 2002.

[3] B. Coifman, "Vehicle level evaluation of loop detectors and the remote traffic microwave sensor", Journal of transportation engineering, vol. 132, no.3, pp. 213-226, 2006.

[4] M. Tursun, and G. Amrulla, "A video based real-time vehicle counting system using optimized virtual loop method", IEEE 8th International workshop on Systems Signal Processing and their Applications (WoSSPA), 2013.

[5] M. Lei, D. Lefloch, P. Gouton, K. Madani, "A video based real-time vehicle counting system using adaptive background method", IEEE International conference on Signal Image Technology and Internet Based Systems (SITIS'08), pp. 523-528, 2008.

[6] N.C. Mithun, N.U. Rashid, and S.M. Rahman, "Detection and classification of vehicles from video using multiple timespatial images", IEEE Transactions on Intelligent Transportation Systems, vol 13, no 3, p. 1215-1225, 2012.

[7] R.T. Collins, et al., "A system for video surveillance and monitoring", VASM final Report, Robotics Institute, Carnegie Mellon University, 2000, pp.1-68.

[8] G. Yang, "Video Vehicle Detection Based on Self Adaptive Background Update", Journal of Nanjing Institute of Technology (Natural Science Edition), vol 2, p. 13, 2012.

[9] F. Liu, and H. Koenig, "A survey of video encryption algorithms", computers & security, vol. 29, no 1, pp. 3-15, 2010.

[10] E. Bas, A.M. Tekalp, and F.S. Salman, "Automatic vehicle counting from video for traffic flow analysis", IEEE Intelligent Vehicles Symposium, 2007.

[11] H. Rabiu, "Vehicle detection and classification for cluttered urban intersection", International Journal of Computer Science, Engineering and

Applications, vol 3, no 1, p. 37, 2013.

[12] M. Seki, H. Fujiwara, and K. Sumi, "A robust background subtraction method for changing background", Fifth IEEE Workshop on Applications of Computer Vision, 2000.

[13] J.J. Gibson, "The perception of the visual world", 1950.

[14] K. Wu, et al., "Overview of video-based vehicle detection technologies", 2011.

[15] N. Friedman, and S. Russell, "Image segmentation in video sequences: A probabilistic approach", Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence, 1997, Morgan Kaufmann Publishers Inc.

[16] C. Stauffer, and W.E.L. Grimson, "Learning patterns of activity using realtime tracking", IEEE Transactions on pattern analysis and machine intelligence, 2000. Vol 22, no 8, pp. 747-757, 2000.

[17] C. Stauffer, and W.E.L. Grimson, "Adaptive background mixture models for real-time tracking", IEEE Computer Society Conference Computer Vision and Pattern Recognition, 1999.

[18] A. Elgammal, D. Harwood, and L. Davis, "Non parametric model for background subtraction", European conference on computer vision, Springer, 2000.

[19] S. O'Hara, and B.A. Draper, "Introduction to the bag of features paradigm for image classification and retrieval", arXiv preprint arXiv:1101.3354, 2011.

[20] D.G. Lowe, "Object recognition from local scale-invariant features", Computer vision, Seventh IEEE international conference, 1999.

[21] D.G. Lowe, "Distinctive image features from scale invariant keypoints", International journal of computer vision, vol 60, no 2, pp. 91-110, 2004.

[22] V. Ramakrishnan, A. K. Prabhavathy, and J. Devishree, "A survey on vehicle detection techniques in aerial surveillance", International Journal of Computer Applications, vol 55, no 18, 2012.

[23] B. Pawar, V.T.Humbe, L. Kundani, "Morphology Based Moving Vehicle Detection". International Conference On Big Data Analytics and computational Intelligence (ICBDACI), pp. 217-223, 2017.

[24] R.H. Pena-Gonzalez, M.A Nuno-Magada, "Computer vision based real-time vehicle tracking and classification system", IEEE 57th International Midwest Symposium on Circuits and systems (MWSCAS), pp. 679-682, 2014.

[25] A. Suryatali, V.B. Dharmadhikari, "Computer Vision Based Vehicle Detection for Toll Collection System Using Embedded Linux", International Conference on Circuit, Power and Computing Technologies (ICCPCT), pp. 1-7, 2015.

[26] N. Seenouvong, U. Watchareeruetai, C. Nuthong, K. Khongsomboon, "A Computer Vision Based Vehicle Detection and Counting System", IEEE 8th International conference on Knowledge and Smart Technology (KST), pp.224227, 2016.

[27] A.B. Godbehere, A. Matsukawa, and K. Goldberg, "Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation", IEEE, American Control Conference (ACC), pp. 4305-4312, 2012.

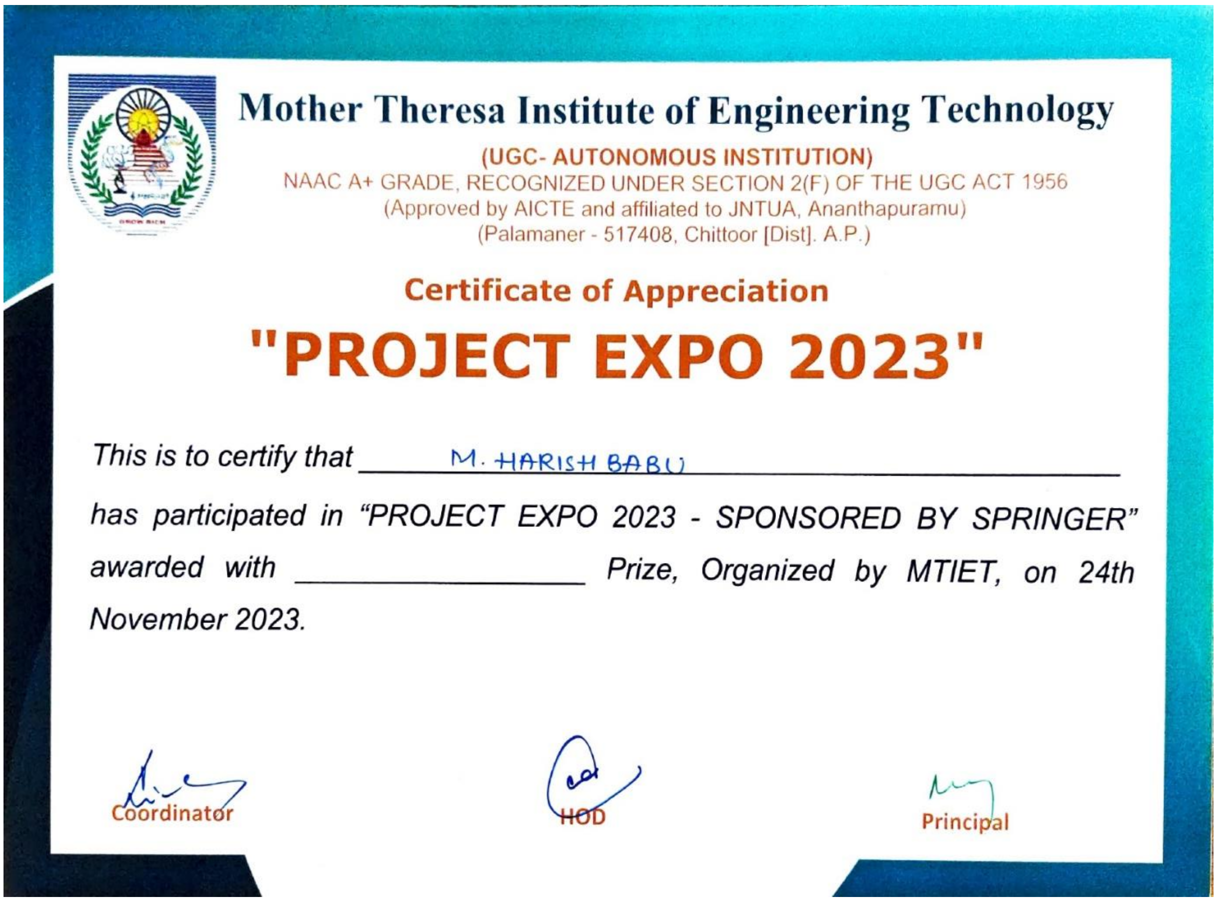
[28] P. KaewTraKulPong, and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection", Video-based surveillance systems, Springer. pp. 135-144, 2002.

[29] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction", 17th International Conference on Pattern Recognition(ICPR), 2004.

[30] Z. Zivkovic, and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction", Pattern recognition letters, vol 27, no 7, pp. 773-780, 2006.

[31] K. Robert, "Night-time traffic surveillance: A robust framework for multi-vehicle detection, classification and tracking", Sixth IEEE International Conference on advanced Video and Signal Based Surveillance (AVSS'09), pp. 1-6, 200

CERTIFICATIONS PROVIDED BY THE INSTITUTION



Mother Theresa Institute of Engineering Technology

(UGC- AUTONOMOUS INSTITUTION)

NAAC A+ GRADE, RECOGNIZED UNDER SECTION 2(F) OF THE UGC ACT 1956

(Approved by AICTE and affiliated to JNTUA, Ananthapuramu)

(Palamaner - 517408, Chittoor [Dist]. A.P.)

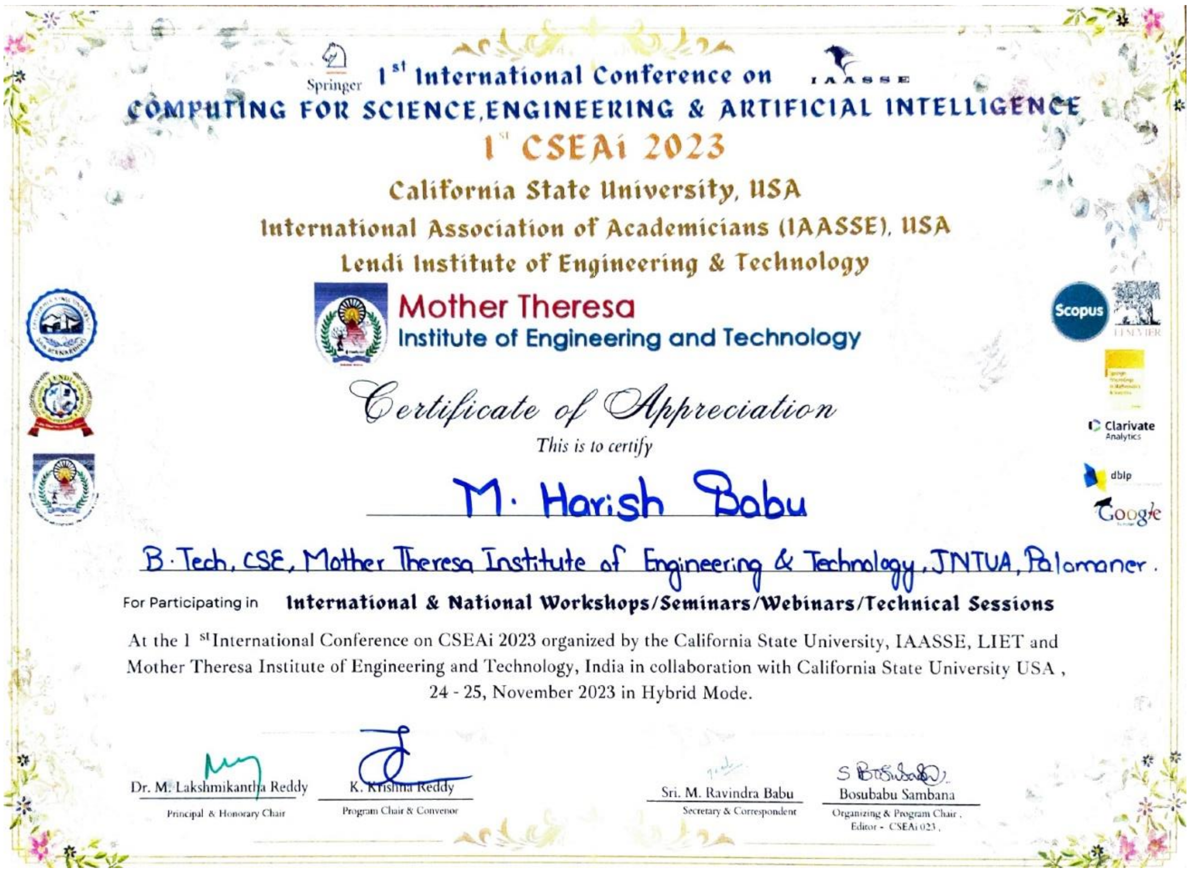
Certificate of Appreciation

"PROJECT EXPO 2023"

This is to certify that M. HARISH BABU

has participated in "PROJECT EXPO 2023 - SPONSORED BY SPRINGER" awarded with ________________ Prize, Organized by MTIET, on 24th November 2023.

Coordinator HOD Principal



Springer 1st International Conference on IAASSE

COMPUTING FOR SCIENCE,ENGINEERING & ARTIFICIAL INTELLIGENCE

1st CSEAi 2023

California State University, USA

International Association of Academicians (IAASSE), USA

Lendi Institute of Engineering & Technology

Mother Theresa Institute of Engineering and Technology

Certificate of Appreciation

This is to certify

M. Harish Babu

B. Tech, CSE, Mother Theresa Institute of Engineering & Technology, JNTUA, Palamaner.

For Participating in International & National Workshops/Seminars/Webinars/Technical Sessions

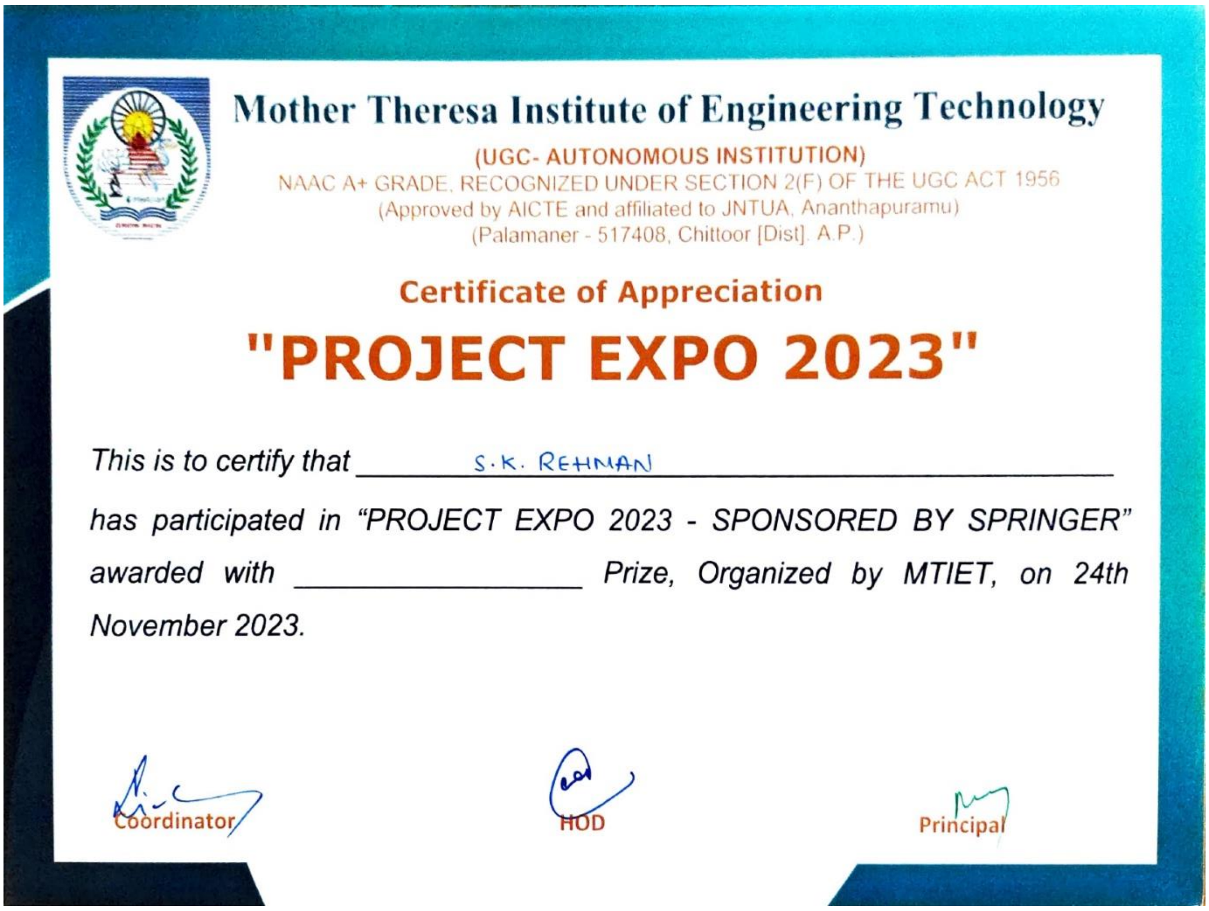
At the 1st International Conference on CSEAi 2023 organized by the California State University, IAASSE, LIET and Mother Theresa Institute of Engineering and Technology, India in collaboration with California State University USA, 24 - 25, November 2023 in Hybrid Mode.

Dr. M. Lakshmikantha Reddy, Principal & Honorary Chair

K. Krishna Reddy, Program Chair & Convenor

Sri. M. Ravindra Babu, Secretary & Correspondent

Bosubabu Sambana, Organizing & Program Chair, Editor - CSEAi 2023



Mother Theresa Institute of Engineering Technology

(UGC- AUTONOMOUS INSTITUTION)

NAAC A+ GRADE, RECOGNIZED UNDER SECTION 2(F) OF THE UGC ACT 1956

(Approved by AICTE and affiliated to JNTUA, Ananthapuramu)

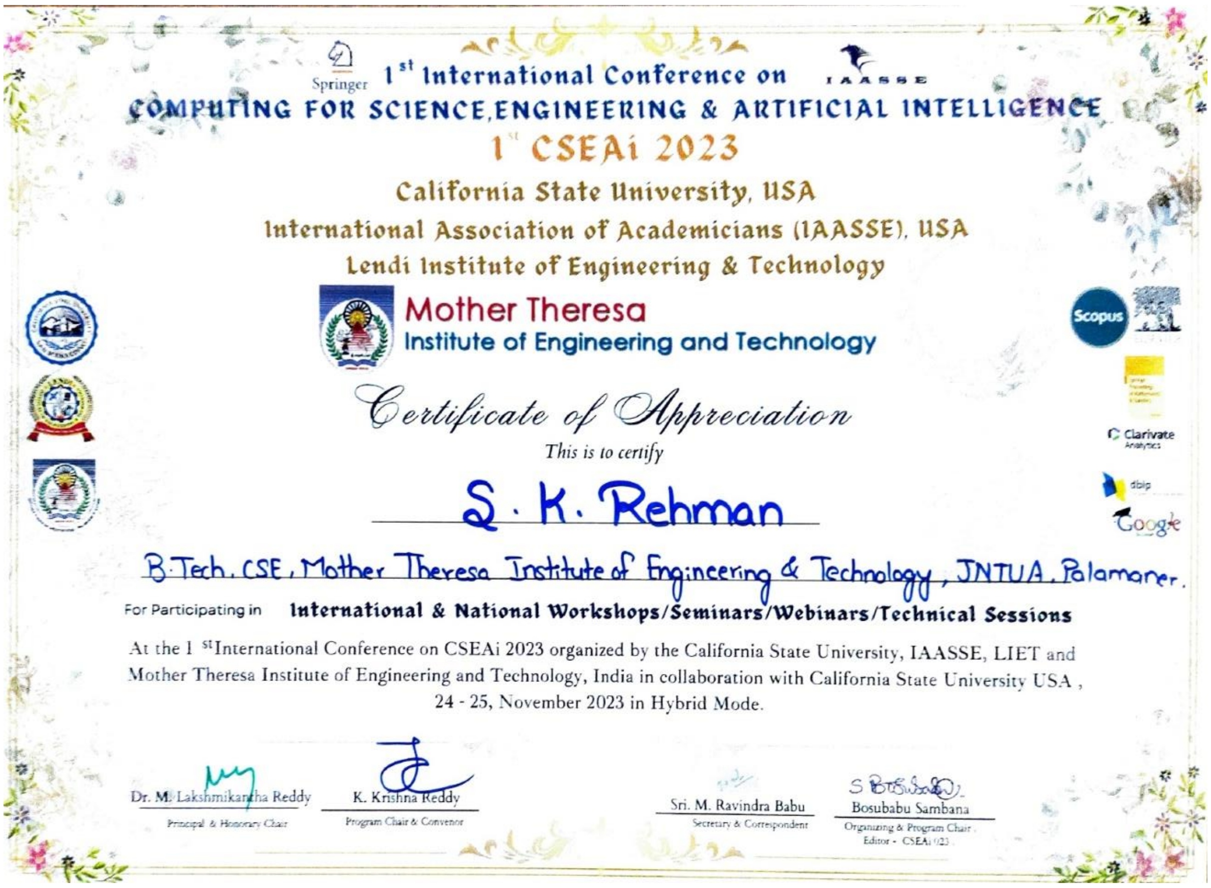
(Palamaner - 517408, Chittoor [Dist]. A.P.)

Certificate of Appreciation

"PROJECT EXPO 2023"

This is to certify that S.K. REHMAN *has participated in "PROJECT EXPO 2023 - SPONSORED BY SPRINGER" awarded with ________________ Prize, Organized by MTIET, on 24th November 2023.*

Coordinator HOD Principal



Springer 1st International Conference on IAASSE

COMPUTING FOR SCIENCE,ENGINEERING & ARTIFICIAL INTELLIGENCE

1st CSEAi 2023

California State University, USA

International Association of Academicians (IAASSE), USA

Lendi Institute of Engineering & Technology

Mother Theresa Institute of Engineering and Technology

Certificate of Appreciation

This is to certify

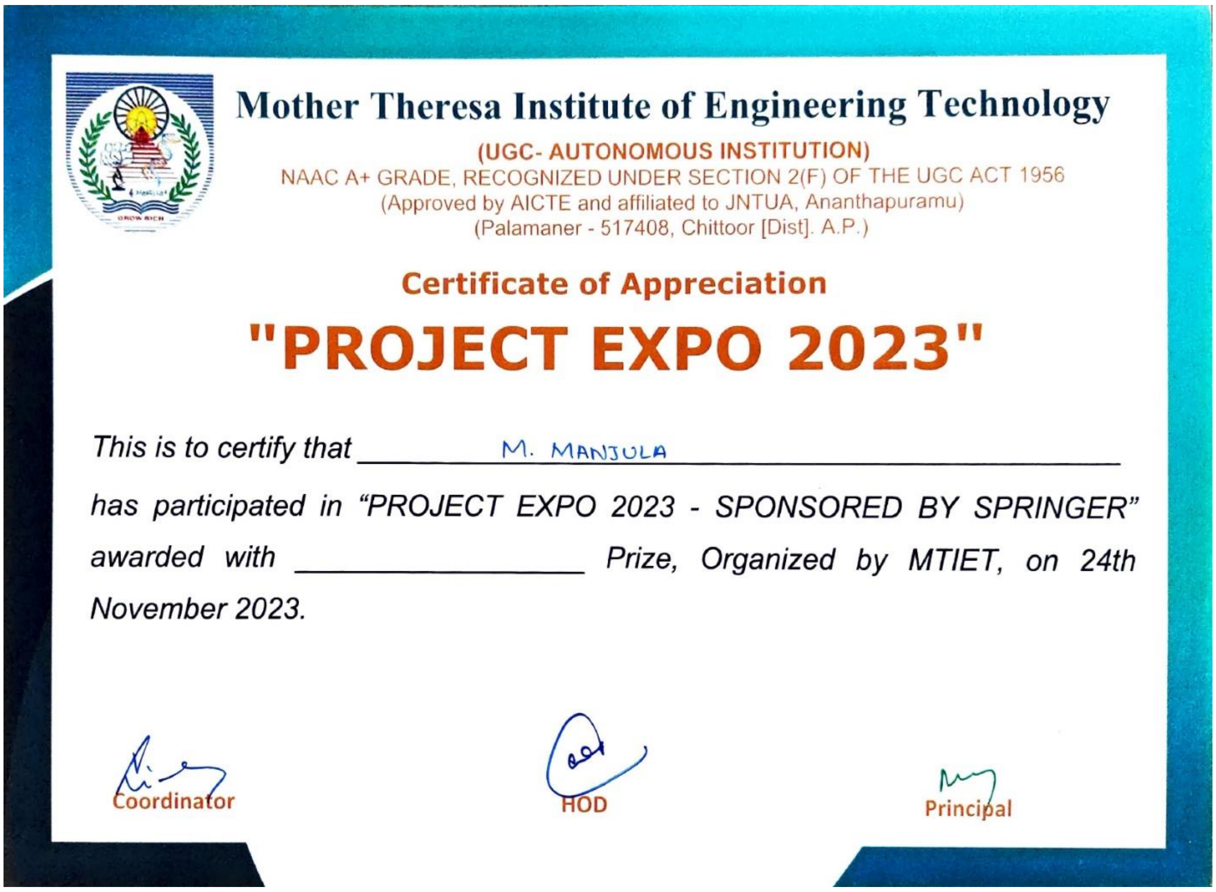
S. K. Rehman

B.Tech. CSE, Mother Theresa Institute of Engineering & Technology, JNTUA, Palamaner.

For Participating in **International & National Workshops/Seminars/Webinars/Technical Sessions**

At the 1 st International Conference on CSEAi 2023 organized by the California State University, IAASSE, LIET and Mother Theresa Institute of Engineering and Technology, India in collaboration with California State University USA , 24 - 25, November 2023 in Hybrid Mode.

Dr. M. Lakshmikantha Reddy	K. Krishna Reddy	Sri. M. Ravindra Babu	Bosubabu Sambana
Principal & Honorary Chair	Program Chair & Convenor	Secretary & Correspondent	Organizing & Program Chair, Editor - CSEAi 2023



Mother Theresa Institute of Engineering Technology

(UGC- AUTONOMOUS INSTITUTION)
NAAC A+ GRADE, RECOGNIZED UNDER SECTION 2(F) OF THE UGC ACT 1956
(Approved by AICTE and affiliated to JNTUA, Ananthapuramu)
(Palamaner - 517408, Chittoor [Dist]. A.P.)

Certificate of Appreciation

"PROJECT EXPO 2023"

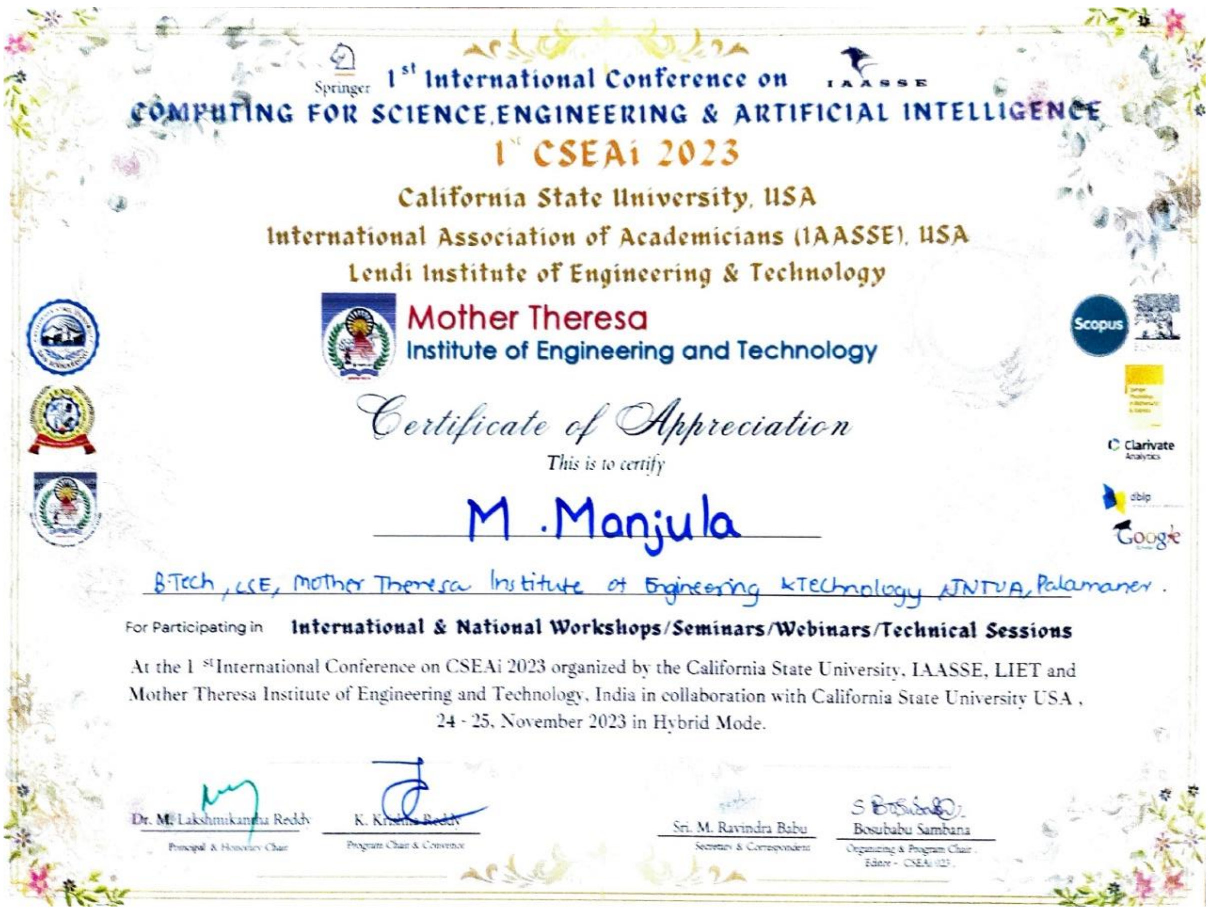
This is to certify that M. MANJULA

has participated in "PROJECT EXPO 2023 - SPONSORED BY SPRINGER" awarded with ________________ Prize, Organized by MTIET, on 24th November 2023.

[illegible] Coordinator

[illegible] HOD

[illegible] Principal



Springer 1st International Conference on IAASSE

COMPUTING FOR SCIENCE,ENGINEERING & ARTIFICIAL INTELLIGENCE

1st CSEAi 2023

California State University, USA

International Association of Academicians (IAASSE), USA

Lendi Institute of Engineering & Technology

Mother Theresa Institute of Engineering and Technology

Certificate of Appreciation

This is to certify

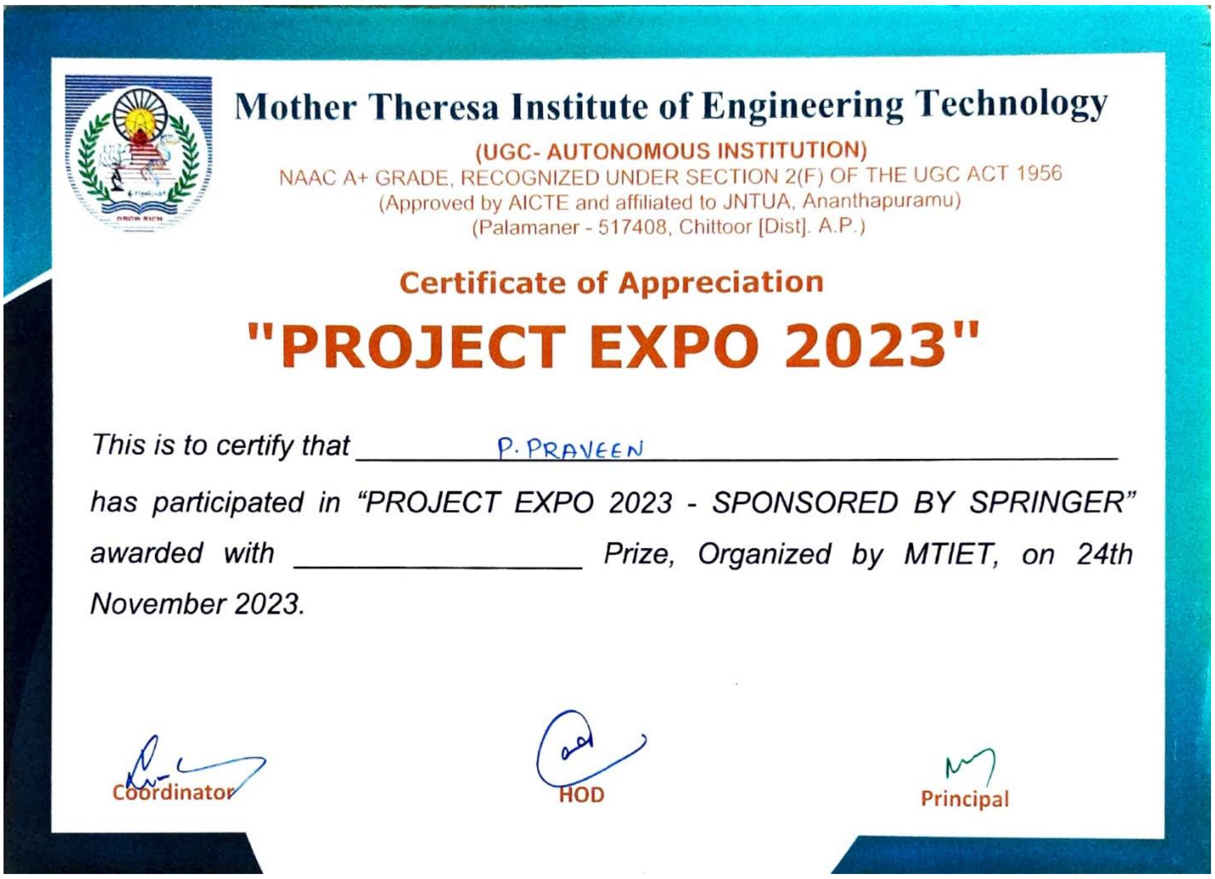
M .Manjula

B.Tech, ECE, Mother Theresa Institute of Engineering &Technology JNTUA, Palamaner.

For Participating in **International & National Workshops/Seminars/Webinars/Technical Sessions**

At the 1 st International Conference on CSEAi 2023 organized by the California State University, IAASSE, LIET and Mother Theresa Institute of Engineering and Technology, India in collaboration with California State University USA , 24 - 25, November 2023 in Hybrid Mode.

Dr. M. Lakshmikantha Reddy	K. Krishna Reddy	Sri. M. Ravindra Babu	Bosubabu Sambana
Principal & Honorary Chair	Program Chair & Convener	Secretary & Correspondent	Organizing & Program Chair, Editor - CSEAi 023



Mother Theresa Institute of Engineering Technology

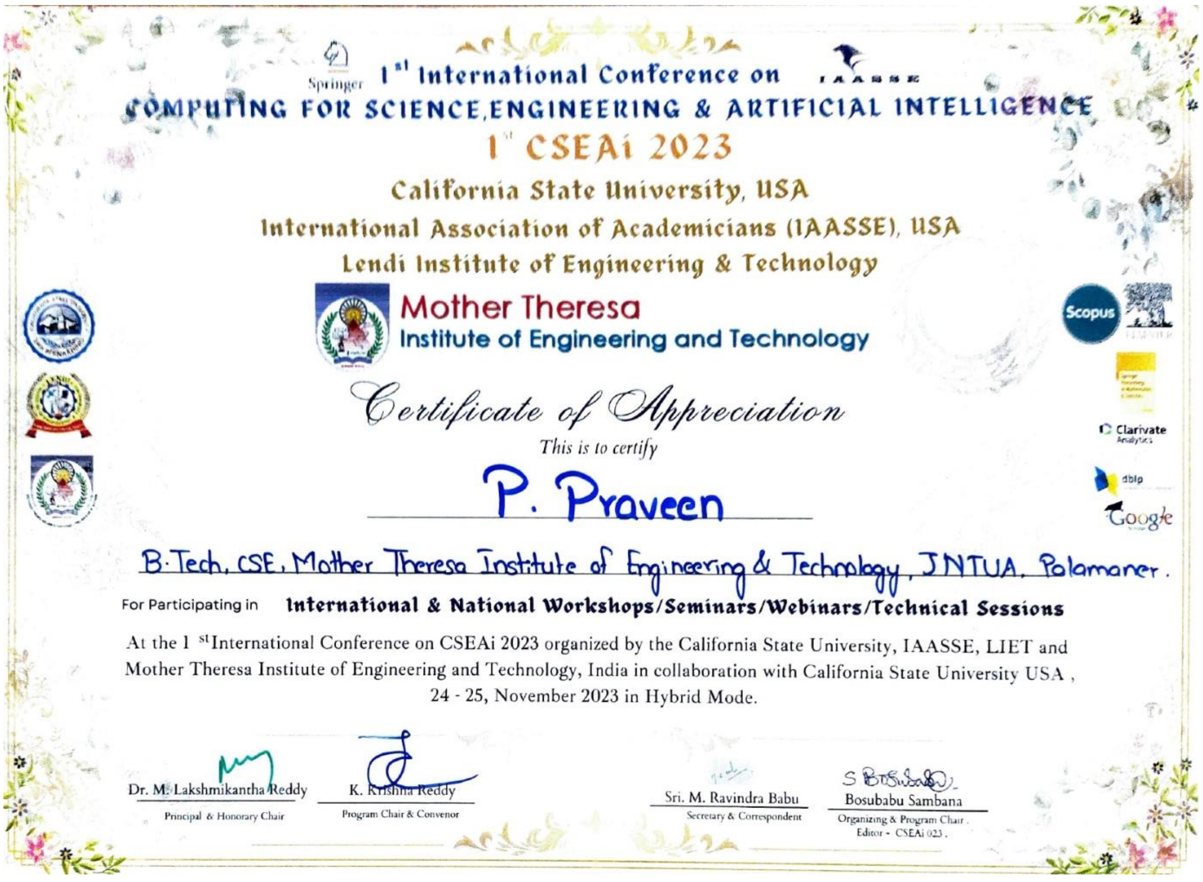
(UGC- AUTONOMOUS INSTITUTION)
NAAC A+ GRADE, RECOGNIZED UNDER SECTION 2(F) OF THE UGC ACT 1956
(Approved by AICTE and affiliated to JNTUA, Ananthapuramu)
(Palamaner - 517408, Chittoor [Dist]. A.P.)

Certificate of Appreciation

"PROJECT EXPO 2023"

This is to certify that P. PRAVEEN *has participated in "PROJECT EXPO 2023 - SPONSORED BY SPRINGER" awarded with* ____________ *Prize, Organized by MTIET, on 24th November 2023.*

Coordinator HOD Principal



1st International Conference on
COMPUTING FOR SCIENCE,ENGINEERING & ARTIFICIAL INTELLIGENCE
1st CSEAi 2023
California State University, USA
International Association of Academicians (IAASSE), USA
Lendi Institute of Engineering & Technology
Mother Theresa
Institute of Engineering and Technology

Certificate of Appreciation

This is to certify

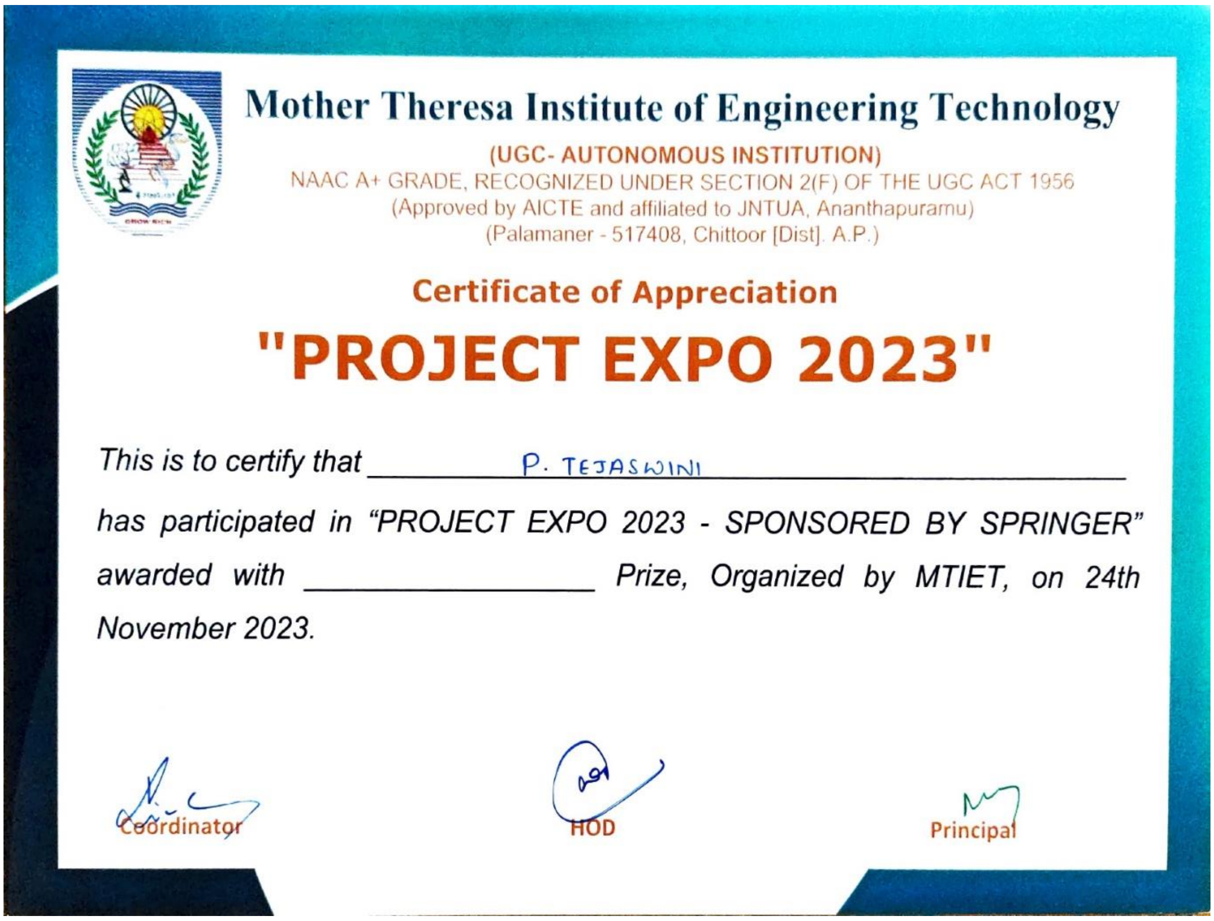
P. Praveen

B.Tech, CSE, Mother Theresa Institute of Engineering & Technology, JNTUA, Palamaner.

For Participating in **International & National Workshops/Seminars/Webinars/Technical Sessions**

At the 1 st International Conference on CSEAi 2023 organized by the California State University, IAASSE, LIET and Mother Theresa Institute of Engineering and Technology, India in collaboration with California State University USA , 24 - 25, November 2023 in Hybrid Mode.

Dr. M. Lakshmikantha Reddy	K. Krishna Reddy	Sri. M. Ravindra Babu	Bosubabu Sambana
Principal & Honorary Chair	Program Chair & Convenor	Secretary & Correspondent	Organizing & Program Chair . Editor - CSEAi 023 .



Mother Theresa Institute of Engineering Technology

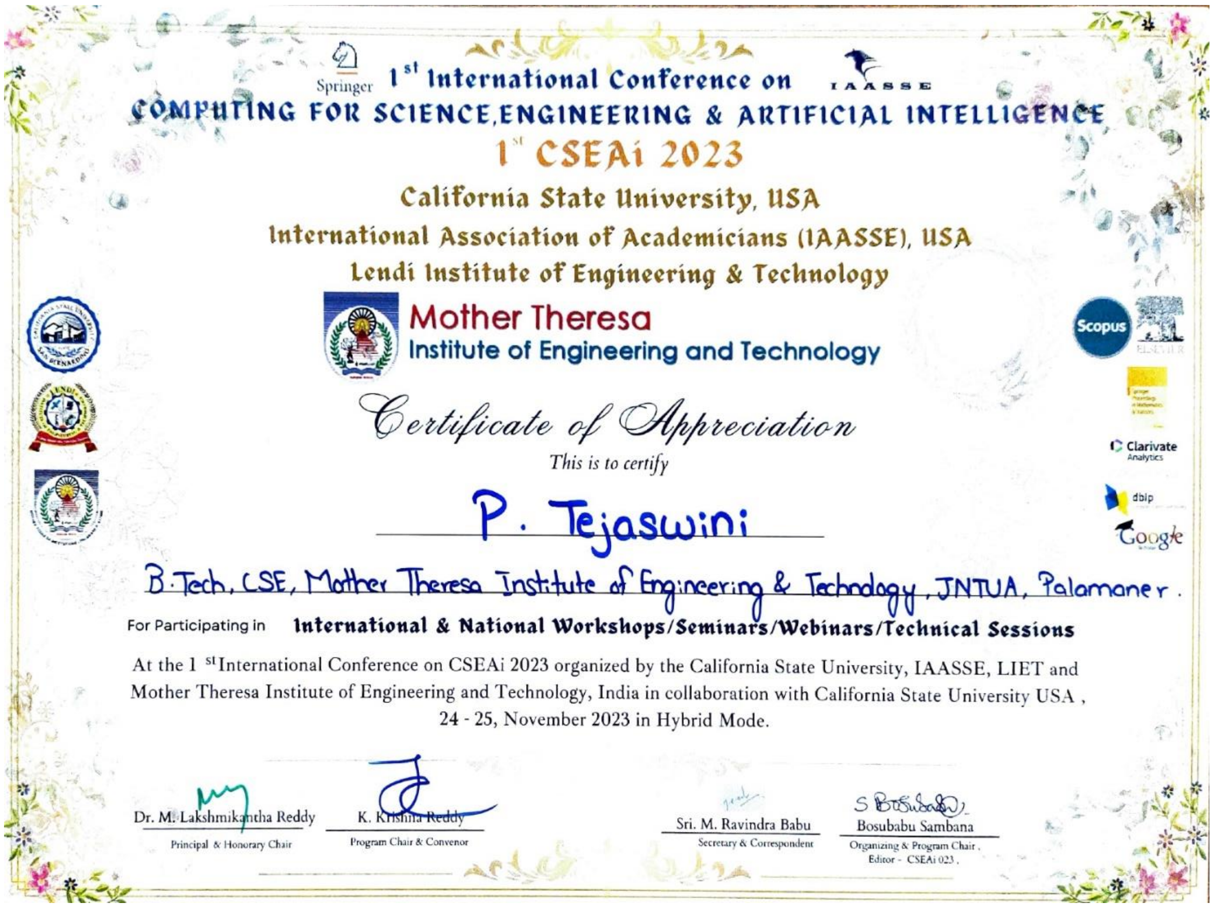
(UGC- AUTONOMOUS INSTITUTION)
NAAC A+ GRADE, RECOGNIZED UNDER SECTION 2(F) OF THE UGC ACT 1956
(Approved by AICTE and affiliated to JNTUA, Ananthapuramu)
(Palamaner - 517408, Chittoor [Dist]. A.P.)

Certificate of Appreciation

"PROJECT EXPO 2023"

This is to certify that P. TEJASWINI
has participated in "PROJECT EXPO 2023 - SPONSORED BY SPRINGER" awarded with ________________ *Prize, Organized by MTIET, on 24th November 2023.*

Coordinator HOD Principal



Springer **1st International Conference on** IAASSE

COMPUTING FOR SCIENCE,ENGINEERING & ARTIFICIAL INTELLIGENCE

1st CSEAi 2023

California State University, USA
International Association of Academicians (IAASSE), USA
Lendi Institute of Engineering & Technology

Mother Theresa
Institute of Engineering and Technology

Certificate of Appreciation

This is to certify

P. Tejaswini

B.Tech, CSE, Mother Theresa Institute of Engineering & Technology, JNTUA, Palamaner.

For Participating in **International & National Workshops/Seminars/Webinars/Technical Sessions**

At the 1 st International Conference on CSEAi 2023 organized by the California State University, IAASSE, LIET and Mother Theresa Institute of Engineering and Technology, India in collaboration with California State University USA , 24 - 25, November 2023 in Hybrid Mode.

Dr. M. Lakshmikantha Reddy
Principal & Honorary Chair

K. Krishna Reddy
Program Chair & Convenor

Sri. M. Ravindra Babu
Secretary & Correspondent

Bosubabu Sambana
Organizing & Program Chair , Editor - CSEAi 023 .