IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

VEHICLE COUNTING CLASSIFICATION AND DETECTION USING OPENCV

[1] Tarun Bondla, [2] Piyush Parate, [3] Shantanu Rapatwar,[4]Sagar Dhawale

[1.2.3]Student, Department of Electronics and Telecommunications, Ajeenkya D. Y. Patil School of Engineering, Lohegaon, Pune

[4] Professor, Department of Electronics and Telecommunication, Ajeenkya D. Y. Patil School of Engineering,
Lohegaon, Pune

Abstract: Vehicle counting, classification, and detection have become essential tasks in modern transportation systems and surveillance applications. This abstract presents a comprehensive overview of these critical functions using OpenCV, a popular computer vision library. Vehicle counting involves tracking and tallying the number of vehicles passing through a specific region. OpenCV provides tools for frame differencing, contour detection, and object tracking, allowing for real-time and accurate vehicle counting in various scenarios. Classification goes a step further by categorizing vehicles into different classes, such as cars, trucks, or motorcycles. OpenCV can leverage deep learning models like Convolutional Neural Networks (CNNs) to achieve high-precision vehicle classification based on image features. Vehicle detection, on the other hand, identifies and localizes vehicles within an image or video stream. OpenCV offers pre-trained models like Haar Cascades and YOLO (You Only Look Once) for robust vehicle detection. These models enable the extraction of bounding boxes around vehicles for further analysis.

***Index Terms* – OpenCV, YOLO, Object detection, Counting.**

I. INTRODUCTION

Vehicle counting and classification using OpenCV and YOLO (You Only Look Once) is an advanced computer vision project aimed at revolutionizing traffic management and surveillance systems. This innovative project leverages cutting-edge technology to detect, count, and classify vehicles in real-time from video feeds or camera streams. At its core, OpenCV (Open Source Computer Vision Library) serves as the foundational framework for image processing and computer vision tasks. It provides a robust set of tools for image and video analysis, making it an ideal choice for handling the complex visual data associated with vehicle tracking. The project's workflow begins with capturing video feeds from surveillance cameras or traffic sensors. OpenCV is then employed to preprocess the video frames, performing tasks like noise reduction, resizing, and frame differencing to isolate moving objects. These processed frames are then fed into the YOLO model, which identifies and classifies vehicles in real-time. Vehicle counting and classification are achieved through a combination of object detection and tracking. Once a vehicle is detected in a frame, it is assigned a unique identifier. The project keeps track of each vehicle across frames, counting their number and classifying them into categories such as cars, trucks, or motorcycles. This information can be invaluable for various applications, including traffic management, security, and urban planning. Advanced techniques such as deep learning and neural networks play a pivotal role in achieving accurate results. The YOLO model is pre-trained on a vast dataset, enabling it to recognize a wide range of vehicle types and attributes. Fine-tuning and retraining the model on specific datasets can further improve its accuracy and adapt it to different scenarios. This project can be extended to include features like license plate recognition, speed estimation, and even behaviour analysis, adding more layers of intelligence to the system.

II. AIM

The aim of a vehicle counting, classification, and detection project using OpenCV is to develop a computer vision system that can automatically:

Count Vehicles: Accurately count the number of vehicles passing through a particular area, such as a road or intersection.

Classify Vehicles: Identify and categorize vehicles into different types, such as cars, trucks, motorcycles, or buses.

Detect Vehicles: Detect the presence of vehicles within a given frame or image and track their movement over time.

III. Working Principle

Vehicle counting, classification, and detection using OpenCV involves several steps and techniques to analyse video or image data. Here's a overview of the working principles:
Frame Acquisition: The first step is to capture video frames or images using a camera or load pre-recorded video files. Preprocessing: To improve the quality of the input data, preprocessing techniques are applied, which may include resizing, noise reduction, and colour space conversion. Object Detection: OpenCV offers pre-trained models, like Haar Cascades or deep learning-based models (e.g., YOLO, SSD, Faster R-CNN), for detecting objects, including vehicles, in each frame. These models identify potential vehicle locations. Object Tracking: To count and classify vehicles accurately, you need to track them across frames. Various tracking algorithms (e.g., Kalman filter, Hungarian algorithm) are available in OpenCV for this purpose. Classification: If you need to classify vehicles (e.g., cars, trucks, motorcycles), you can use machine learning models trained on vehicle images. OpenCV can be used for feature extraction and interfacing with machine learning libraries like scikit-learn or TensorFlow. Counting and Data Collection: As vehicles are detected and tracked, you can count and collect data about their movement, type, and other attributes. Visualization and Output: OpenCV provides tools for drawing bounding boxes, labels, and other visual elements on the video frames to help visualize the results. You can also save data to a file or display it in real-time. Post-processing: To improve the accuracy of vehicle counting and classification, post-processing steps may be applied, such as filtering out false positives or smoothing the tracking results. Evaluation and Optimization: Continuous evaluation of the system's performance and fine-tuning of parameters and models are essential to achieve reliable results. OpenCV simplifies many of these steps through its extensive library of computer vision functions and pre-trained models.

IV. Proposed System

A proposed system for vehicle counting, classification, and detection leverages advanced computer vision algorithms to accurately identify and categorize vehicles in real-time. It employs deep learning models to detect and track vehicles, providing valuable data for traffic management and surveillance. The system is capable of distinguishing between different vehicle types, such as cars, trucks, and motorcycles, contributing to more detailed traffic analysis. This solution is designed to enhance traffic safety, optimize road usage, and aid law enforcement in monitoring traffic conditions efficiently.

V. Object Detection Algorithms

1. Convolutional Neural Network (CNN):

CNN is a deep learning architecture designed for processing grid-like data such as images. It comprises multiple layers of convolutional and pooling layers, which learn hierarchical features from the input data. In the context of object detection, CNNs are often used as feature extractors to capture relevant information from the input image.

2. Region-based Convolutional Neural Network (R-CNN):

R-CNN was an early attempt to combine deep learning with object detection. It first generates region proposals in an image and then uses a CNN to extract features from each proposal. These features are then passed to a support vector machine (SVM) for object classification and bounding box regression. R-CNN achieved good accuracy but was slow due to the need to process a large number of region proposals.

3.Single Shot MultiBox Detector (SSD):

SSD is a real-time object detection model that integrates the entire object detection pipeline into a single neural network. It combines feature extraction, bounding box prediction, and class prediction in one architecture. SSD uses a set of default bounding boxes at multiple scales and aspect ratios to predict objects, making it efficient and accurate for various object sizes and shapes.

4.You Only Look Once (YOLO):

YOLO is another real-time object detection algorithm that revolutionized the field. YOLO divides the input image into a grid and predicts bounding boxes and class probabilities for each grid cell simultaneously. This one-pass approach is highly efficient and allows YOLO to achieve real-time performance, making it ideal for applications like autonomous vehicles and real-time surveillance.

In summary, CNN is the fundamental building block for feature extraction in many object detection algorithms. R-CNN was an early approach that introduced region proposals, but it was computationally expensive. SSD and YOLO are two newer algorithms that are designed for real-time performance, with SSD providing a balance between speed and accuracy, while YOLO offers a remarkable balance between speed and high accuracy. The choice of which algorithm to use depends on the specific requirements of the application and the trade-off between accuracy and speed.

VI. METHODOLOGY

Performing vehicle counting, classification, and detection using OpenCV and YOLO typically involves several key steps and a structured methodology. In this project we have used the following technologies.

OpenCV (Open Source Computer Vision Library**): -**

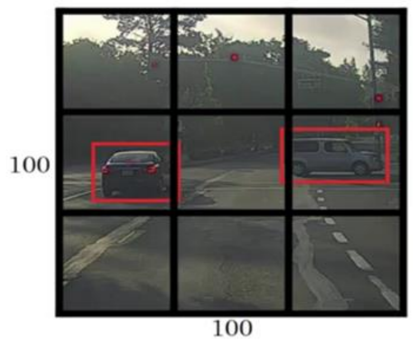
OpenCV is an open-source computer vision and machine learning software library. It provides a wide range of tools and algorithms for various computer vision tasks, including image and video analysis, object detection, image processing, and more. OpenCV is widely used in both academic and industrial applications and is known for its extensive set of functions, excellent community support, and cross-platform compatibility

YOLO(You Only Look Once): -

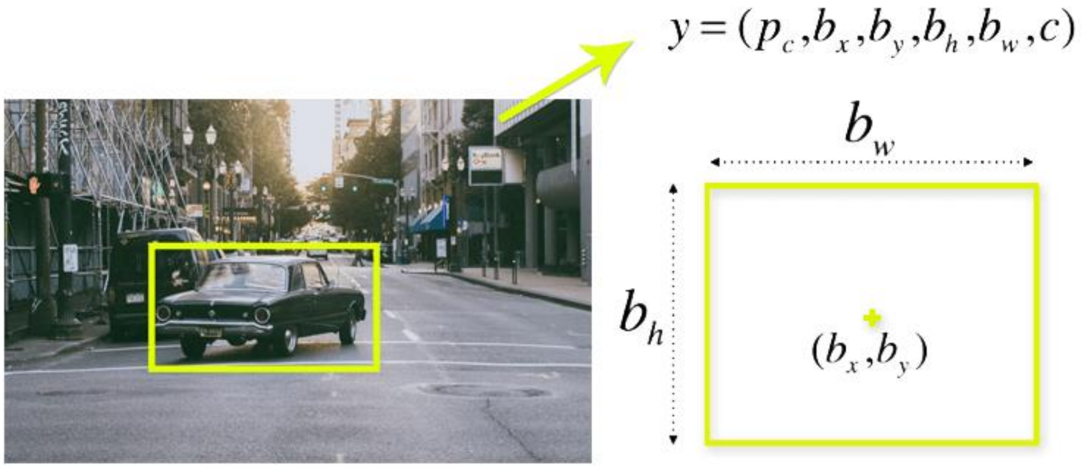
YOLO or You Only Look Once, is a popular real-time object detection algorithm. YOLO combines what was once a multi-step process, using a single neural network to perform both classification and prediction of bounding boxes for detected objects. As such, it is heavily optimized for detection performance and can run much faster than running two separate neural networks to detect and classify objects separately. It does this by repurposing traditional image classifiers to be used for the regression task of identifying bounding boxes for objects. This article will only look at YOLOv1, the first of the many iterations this architecture has gone through. Although the subsequent iterations feature numerous improvements, the basic idea behind the architecture stays the same. YOLOv1 referred to as just YOLO, can perform faster than real-time object detection at 45 frames per second, making it a great choice for applications that require real-time detection. It looks at the entire image at once, and only once — hence the name You Only Look Once — which allows it to capture the context of detected objects. This halves the number of false-positive detections it makes over R-CNNs which look at different parts of the image separately. Additionally, YOLO can generalize the representations of various objects, making it more applicable to a variety of new environments. Now that we have a general overview of YOLO. let's take a look at how it really works.

YOLO first takes an input image:

YOLO divides the resized image into a grid, typically with, for example, a 7x7 or 13x13 grid, depending on the YOLO version being used.



For each grid cell, YOLO predicts multiple bounding boxes (usually 2, 3, or 5 boxes per grid cell). These boxes are specified by coordinates (x, y) for the box's centre, width, and height.
p_c defines whether an object is present in the grid or not (it is the probability) b_x, b_y, b_h, b_w specify the bounding box if there is an object. Here c_1, c_2, c_3 represent the classes. So, if the object is a car, c_2 will be 1 and c_1 $_\&$ c_3 will be 0, and so on



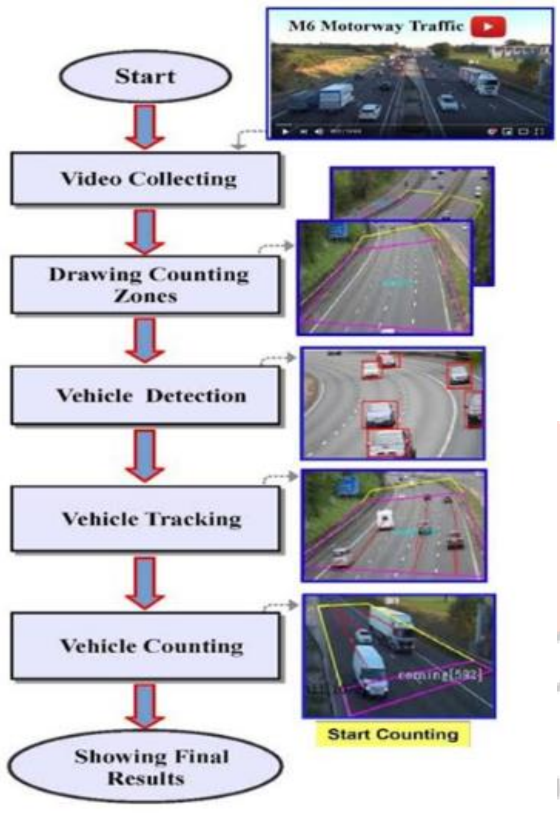
YOLO also predicts class probabilities for each object it identifies. This indicates the likelihood of the detected object belonging to a specific class (e.g., car, person).
Each bounding box has an associated confidence score that represents how confident YOLO is that the box contains an object. The confidence score takes into account both the presence of an object and the accuracy of the predicted bounding box.
YOLO combines the predictions from multiple grid cells, including their bounding boxes, class probabilities, and confidence scores. Non-maximum suppression (NMS) is applied to remove redundant and low-confidence detections. NMS retains only the most confident bounding boxes for each object.
YOLO's unique feature is its ability to perform all these steps in a single forward pass through a neural network, making it highly efficient and suitable for real-time object detection applications.

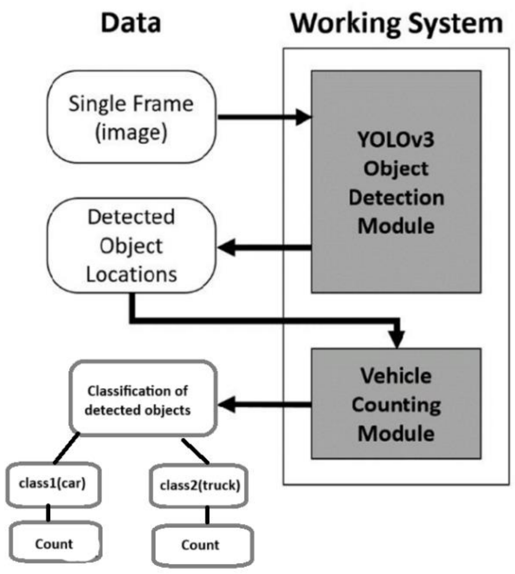
Step by Step Methodology process: -

- Data Collection: Collect or acquire a dataset of images or video footage that contains various vehicles. Annotate the dataset to label the vehicles and their categories (e.g., car, truck, bicycle).
- Setting Up the Environment: Install OpenCV and the YOLO model (e.g., YOLOv3 or YOLOv4). Ensure you have a compatible GPU (if using YOLO for real-time processing) and the required dependencies.
- Preprocessing: Preprocess the dataset and video frames (if applicable) using OpenCV. Resize, crop, and normalize the images or video frames to match the input requirements of the YOLO model.
- YOLO Model Configuration: Load the pre-trained YOLO model weights and configuration files. Define the classes for vehicle categories. Set up the YOLO model for object detection using the YOLO library or framework you've chosen.
- Object Detection: Use the YOLO model to perform object detection on the images or video frames. Detect and draw bounding boxes around vehicles. Extract detected vehicle information, including class labels and confidence scores.

- Post-processing: Apply non-maximum suppression (NMS) to remove duplicate or highly overlapping bounding boxes. Filter out detected objects with low confidence scores.
- Counting Vehicles: Implement logic to count the number of vehicles in each frame. Keep track of vehicle counts over time for video sequences.
- Classification: If you need to classify vehicles into subcategories (e.g., car, bike, truck), train a classification model using a separate dataset. Classify detected vehicles based on their characteristics and appearance.
- Data Analysis and Reporting: Analyse the vehicle counts and classifications over time. Generate reports or visualizations to present the results.

VII. FLOW CHART





VIII. Acknowledgment

First of all, we would like to thank Head of Electronics & Telecommunication Engineering Department, to give us the opportunity to work on proposed system.
We wish to express our sincere gratitude to our guide for his kind guidance and valuable suggestions without which this proposed work would not have been taken up.
We sincerely acknowledge the encouragement, timely help and guidance given to us by our beloved Guide carry out this proposed work within the stipulated time successfully.

References

[1] Karthik Srivathsa D S, Dr. Kamalraj R"Vehicle Detection and Counting of a vehicle using openCV", International Research Journal of Modernization in Engineering Technology and Science,2021

[2] Mr. P. Nagaraj, Raj Varshith Reddy, Mohammed Sabeehuddin, SriChandana Reddy V Santhosh "A Video based Vehicle Detection, Counting and Classification System",Alochana Chakra Journal,2020

[3] Sheeraz Memon,"A Video based Vehicle Detection, Counting and Classification System",2018.

[4] Prof. Anwarul Siddiqu ,Rohma Firdous , Shruti Reddy3 , Shariya Naaz4 , Aaliya Khan5 "Video-Based Detection, Counting and Classification of Vehicles Using OpenCV", 2022.

[5] Ravula Arun Kumar, D. Sai Tharun Kumar, K. Kalyan, B. Rohan Ram Reddy "Vehicle Counting and Detection", 2020.

[6] https://techvidvan.com/tutorials/opencv-vehicle-detection-classification-counting

[7] Vehicle Detection and Counting System using OpenCV - Analytics Vidhya/.