Intelligent Transportation System Based On Machine Learning For Vehicle Perception

Harshilsinh Rana[1], Dr. Dipesh Makwana[2]

[1]Student, Dept. of I.C Engineering, L.D college of Engineering, Gujarat, India
[2] Associate Professor, Dept. of I.C Engineering, L.D college of Engineering, Gujarat, India

---***---

Abstract - *The term "intelligent transportation systems," or ITS for short, refers to a group of services and applications that include, among others, driverless vehicles, traveler information systems, and management of public transit systems. Future smart cities and urban planning are predicted to heavily rely on ITS, which will improve transportation and transit efficiency, road and traffic safety, as well as energy efficiency and environmental pollution reduction .However, because of its scalability, a wide range of quality-of-service requirements, and the enormous amount of data it will produce, ITS poses a number of difficulties .In this study, we investigate how to make ITS possible using machine learning (ML), a field that has recently attracted a lot of attention. We offer a comprehensive analysis of the current state-of-the-art which covers many fold perspectives grouped into ITS ML-driven supporting tasks, namely perception, prediction and management of how ML technology has been used to a variety of ITS applications and services, like cooperative driving and road hazard warning, and determine future paths for how ITS might more fully utilize and profit from ML technology.*

Key Words: **Intelligent transportation system, Perception tasks, Machine Learning, Road safety, Privacy and security**

1.INTRODUCTION

The use of information, communication, and sensing technologies in transportation and transit systems is commonly referred to as "intelligent transportation systems," or ITS for short[1]. Future smart cities are anticipated to incorporate ITS as a key element, and it will likely contain a range of services and applications, including autonomous vehicles, traveler information systems, and management of public transit systems, to mention a few[2]. It is anticipated that ITS services will make a substantial contribution to higher energy efficiency, decreased environmental pollution, improved road and traffic safety, and transportation and transit efficiency. ITS applications have been made possible by remarkable developments in sensing, computation, and wireless communication technology; but because of their scalability and a range of quality-of-service requirements, they will present a number of obstacles, additionally to the enormous amounts of data that they will produce.

Parallel to this, cloud and edge computing, as well as other technologies, have helped machine learning (ML) techniques to gain a lot of popularity. A wide range of applications that, like ITS services, demand a variety of requirements have incorporated machine learning (ML). For prediction and precise decision making[3-5] in addition to vehicular cybersecurity, ML technologies like deep learning and reinforcement learning have been particularly helpful tools to discover patterns and underlying structures in massive data sets.[6] The number of research initiatives leveraging ML to enable and optimize ITS tasks has clearly increased over the past 10 years, according to statistics on scientific publications (see Figure 1).

The following figure explains about the number of publications that are done on ITS, which are included in machine learning approaches. Some different functions that are also used in the proposed system as vehicle detection, identification, classification, speed detection. Hence we will use the different algorithms for all the system. This study work intends to discuss a precise and useful method for counting moving vehicles that may be applied in the confusing traffic situation. To find a moving vehicle and get a foreground image, techniques like adaptive background reduction and morphological activities are utilized

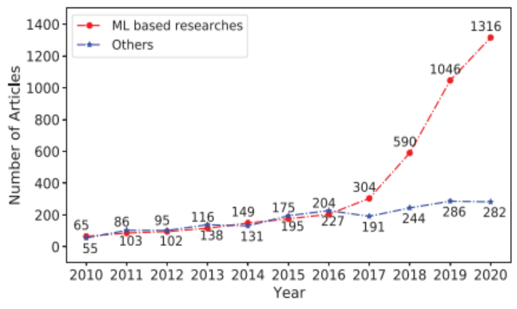

Chart -1: Number of publications on ITS, including ML-based approaches, from 2010 to 2020

Now we will have some background information about the computer vision and OpenCV. The goal of the branch of study known as computer vision is to make it possible for machines to analyze and comprehend visual data from the environment. It is a branch of artificial intelligence (AI) concerned with giving machines the ability to see and comprehend the visual world. Object identification and recognition, picture and video analysis, autonomous vehicles, medical image analysis, and many other uses are just a few of the many applications for computer vision.

For use in real-time computer vision applications, OpenCV (Open Source Computer Vision Library) is an open-source computer vision library. A multinational group of developers now maintains it after Intel initially built it. For a variety of computer vision applications, including processing images and videos, object detection and recognition, feature detection and matching, and machine learning, OpenCV offers a set of tools and functions. Python, C++, Java, and other programming languages are among those it supports.

For numerous computer vision applications, such as facial recognition, gesture recognition, autonomous driving, robotics, and many others, OpenCV is widely utilized in both research and industry. It offers a wide range of computer vision features, such as processing of images and videos, feature extraction and identification, object recognition and tracking, machine learning, and more. It also has a sizable user and developer community that contributes to the library and offers assistance and resources to others so they can utilize it efficiently.

2. RELATED WORK

We can now have a brief review of the existing literature on vehicle detection, counting, classification, identification, and speed detection using OpenCV, and compare the different techniques and approaches used by various researchers.

Vehicle detection is an essential problem in many applications, including autonomous driving, parking management, and traffic monitoring. The three primary methods for vehicle detection in OpenCV are YOLO, HOG + SVM, and Haar cascades. A quick method for detecting automobiles is to utilize a cascade classifier with Haar-like characteristics. However, it might not be effective in difficult scenes or when looking for small vehicles. On the other hand, HOG + SVM, which performs well in a variety of illumination circumstances, uses Histogram of Oriented Gradients (HOG) features and a Support Vector Machine (SVM) to recognize cars. However, the computational cost might be high. A neural network is used in the YOLO deep learning method to identify automobiles in a single pass. It can recognize small vehicles and performs well in real-time applications, but it could struggle to recognize obscured or dimly lit vehicles. The requirements and limitations of the application, such as the desired speed, accuracy, and complexity, determine the best course of action.

Another crucial job in many contexts, including traffic control and congestion management, is vehicle counting. The two primary methods for vehicle counting with OpenCV are optical flow and background subtraction. A simple and quick method, background subtraction uses a background model to find and count moving cars. However, it might not be effective for overlapping vehicles or complex scenes. On the other hand, optical flow counts the vehicles and estimates their motion, which is useful for real-time applications. It might not be effective for stationary or slowly moving automobiles, though. The application's particular requirements and constraints, such as the desired accuracy, speed, and resilience, must be taken into consideration while choosing the best strategy.

The practice of classifying vehicles into various kinds, such as cars, trucks, and buses, is known as vehicle classification. Using OpenCV, one can categorize vehicles using a variety of methods, such as color, shape, and texture features. Color features are a quick and easy way to categorize automobiles since they use color information. However, under different lighting conditions, it might not be precise. While more accurate than color features, shape features may be computationally expensive because they classify vehicles based on their geometric shape. Texture features categorize cars based on their texture patterns, which works well for different vehicle kinds but may not function well in complicated scenarios. Depending on the requirements of the particular application, the best method for vehicle classification can be chosen.

Aside from traffic management and law enforcement, other crucial tasks include vehicle identification and speed detection. License plate detection and recognition are the two primary methods for identifying vehicles using OpenCV. In order to identify license plates, license plate identification employs image processing methods like edge detection and morphological processes. It works well in various lighting situations but might not be effective with license plates that are damaged or concealed. On the other hand, license plate identification use optical character recognition (OCR) methods to identify the characters on the plate. It performs admirably for clearly readable license plates, but it might not be precise for characters that are deformed or blurry. The two major OpenCV methods for detecting vehicle speed are GPS tracking and optical flow. When used in real-time applications, optical flow evaluates the motion of vehicles and determines their speed. However, it might not be precise for stationary or slowly moving vehicles. However, GPS tracking, which is more precise than optical flow, uses GPS technology to track vehicles and determine their speed. It might not function properly in locations with weak GPS signals, though. Depending on the particular application needs and restrictions, such as the desired accuracy, speed, and availability of hardware and sensors, the best method for vehicle identification and speed detection can be chosen.

3. METHODOLOGY

In this section we will have a brief information about the vehicle detection, classification, identification, speed detection. Let's start with the vehicle detection.

3.1 Vehicle Detection

Background subtraction is used to first identify pixels that might be associated with a moving vehicle in the foreground. The current frame in the video is converted from a color (RGB) to a grayscale image, and the background image of the street has no cars in it [7]. At that time, the grey intensity of a background image is subtracted from that of a current frame for every pixel (x, y). In another image, also known as a different image, the absolutist result is stored in a position that is similar to that of the first image.

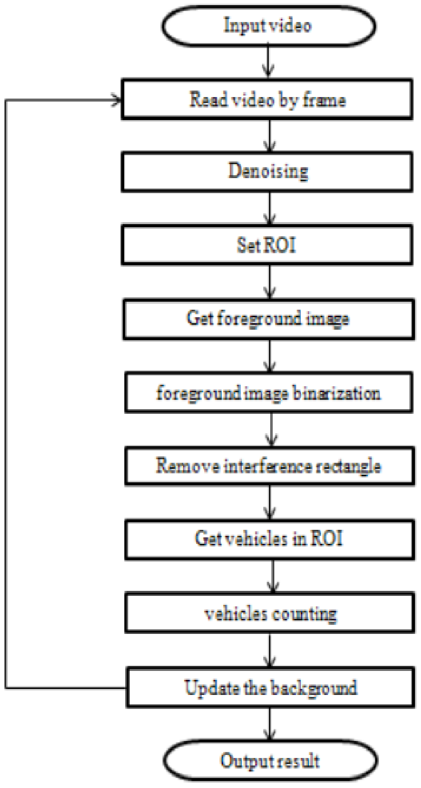

Fig -1: Flow Diagram of Vehicle Counting

A ROI that will be processed further. Zones are isolated within the ROI. Each will be treated differently in the steps that follow. Additionally, a second zone known as the virtual detection zone is also established for vehicle counting [8]. The zone of virtual recognition is contained inside zones. which serves as the foundation of ROI. The upper and lower bounds on the y-coordinate where the automobiles are anticipated to appear implicitly determine the ROI. Particularly, the variables up limit and down limit specify the top and bottom limits of the ROI, respectively. Contours found outside the ROI are filtered out using these values. The only contours that are taken into account as prospective vehicles and sent to the vehicle tracking module are those whose centroid falls inside the ROI's bounds.

In order to obtain the foreground mask, background subtraction which subtracts the current frame from the background frame is used for detection. This makes it easier to distinguish moving images in videos. To reduce noise and fill in gaps in the objects, the foreground mask is then treated using morphological operations including dilation and closing. The recovered contours from the processed mask are next examined to see if they match a vehicle.

A bounding box is drawn around a vehicle in the video frame if a contour resembles one (i.e., it meets certain established criteria like minimum width and height). The list of identified cars is then expanded to include the center of the bounding box.

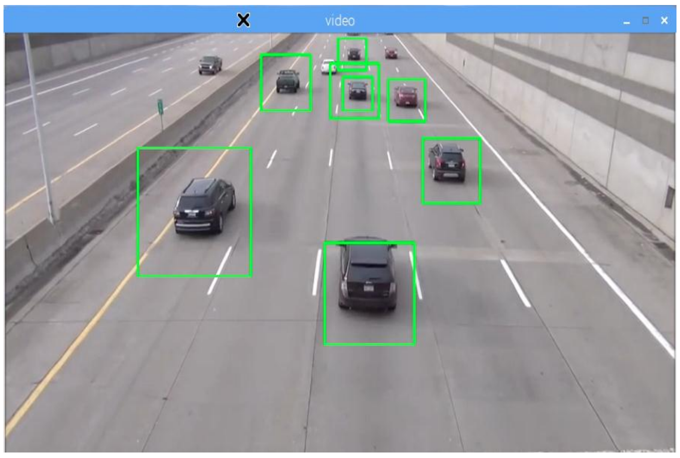

Fig -2: Detection of Vehicles

3.2 Vehicle Classification

Detecting, classifying, and counting the number of cars moving through a specific area of interest in a video stream is a computer vision job known as "vehicle classification and counting using OpenCV." To separate moving objects from the backdrop, the algorithm initially uses a background subtraction technique. The binary image that results is then subjected to morphological processes to eliminate noise and fill in gaps.

The binary image's contours are then retrieved, and each contour is examined to see if it matches a vehicle by examining its area. If a contour is recognized as belonging to a vehicle, its centroid is extracted and tracked over time to establish its course. The technology is also capable of categorizing vehicles based on their size and shape, for instance, separating cars from lorries. Last but not least, the number of vehicles going through a specific area of interest is determined by whether they cross an illegible line on the screen that is travelling in a specific direction.

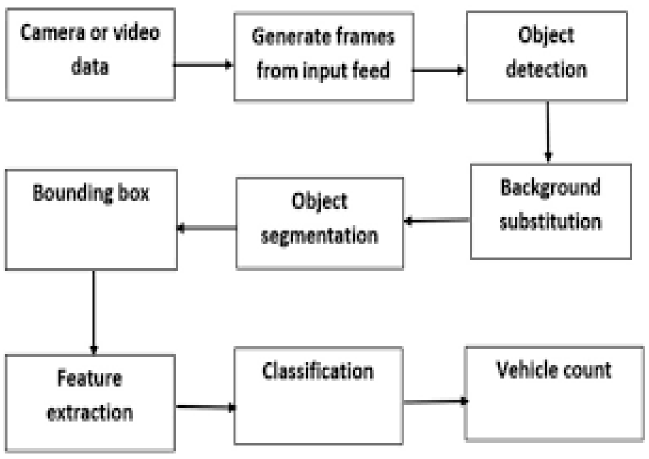

Fig -3: Proposed model of Vehicle Classification

3.3 Vehicle Identification

Using OpenCV, it is intended to find license plates in a video stream. An XML file is used to generate a trained classifier object, establish a minimum area for license plates, and specify a color for drawing rectangles around identified plates. Next, the code determines the desired width and height of the video frame.

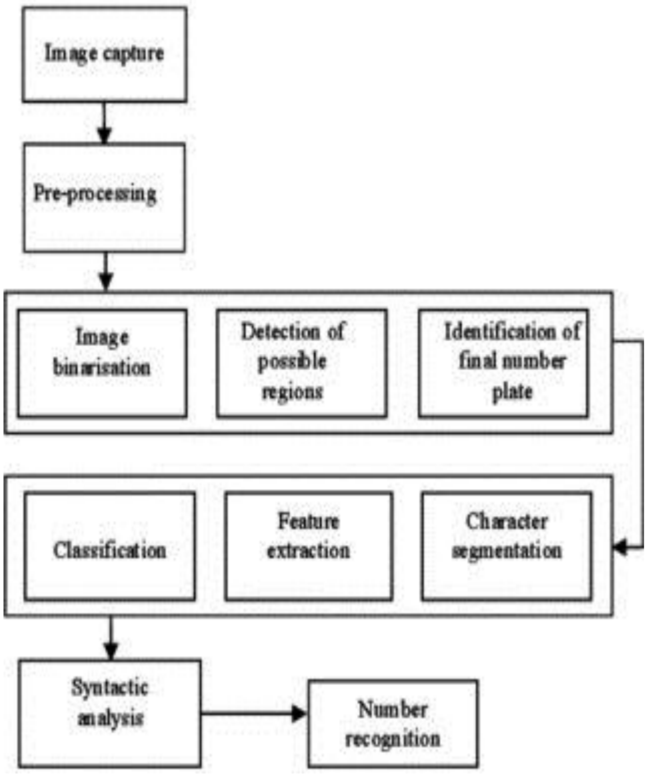

Fig -4: Vehicle Number Plate Identification model

Each frame of the video is read from the while loop and turned into grayscale. After that, the function is utilized to find license plates in the grayscale picture. This function uses a sliding window technique to find items of various sizes in an image, and it provides a list of bounding boxes that show where each detected license plate can be found. It determines if the rectangle's area exceeds the previously determined minimum area. The code surrounds the number plate with a rectangle and adds wording to identify it as a "Vehicle number Plate" if the region matches the minimal requirement.

3.4 Vehicle Speed Detection

It is an implementation of object tracking on video footage using OpenCV library. The code detects moving objects in a given region of interest (ROI) using background subtraction and morphology operations.

The detected objects are then tracked using the Euclidean distance tracker algorithm. The speed of the tracked object is calculated and used to determine if the object is moving above a certain speed limit. If it is, the object is marked as over speeding and the corresponding rectangle is drawn in red. The detected objects are also tracked across frames and their IDs are maintained to keep track of individual objects. The code also draws lines on the ROI to differentiate between different lanes on the road. It builds bounding boxes around the found and tracked objects using the MOG2 algorithm for object detection and the Euclidean distance technique for object tracking. Additionally, it keeps track of how many vehicles enter and exit the detection zone, which is helpful for monitoring and analyzing traffic.

The distance between two successive frames is determined using the distance formula to determine the tracked vehicles' speed. Based on the video's frame rate, the interval between each frame is fixed and predefined. The speed of the car can be calculated by dividing the distance travelled by the passing time.

The cars are then divided into two groups based on whether or not they are speeding using the estimated speed. The maximum speed restriction in the interest area serves as the basis for defining the classification threshold. A vehicle is considered to be speeding if its speed exceeds the threshold and is marked with a red bounding box; otherwise, it is marked with a green bounding box.

4. DETAILED ALGORITHMS

The vehicle detection, counting, identification, classification, speed detection algorithm involves several computer vision techniques and image processing operations. Here is a breakdown of the algorithms used:

4.1 Background Subtraction

In computer vision and image processing, the concept of background subtraction is used to distinguish foreground items from the background in an image or video. The fundamental concept is to take the foreground and subtract the backdrop from the current frame. The mathematical ideas and procedures involved in background subtraction are explained in full here. Calculating the average of the frames over a predetermined amount of time is a standard technique for building a background model. Considering B to be the background model at pixel (x, y);

$$B(x, y) = (1/N) * \Sigma\ i=1^N\ F(x, y, i)$$

where N is the total number of frames used to build the background model and F(x, y, i) is the value of the pixel at position (x, y) in frame i.

If we already have a backdrop image, such as an image of a building or a road, the background subtraction task can be simple. In the situations listed above, the background image could be eliminated and the foreground objects could be acquired, although this is not always the case. The original information about the setting might not be available, or the backgrounds might be dynamic.

Furthermore, because shadowed objects in videos move along with people and vehicles, making background subtraction more challenging because the shadows will be mistaken for foreground objects by conventional background subtraction.

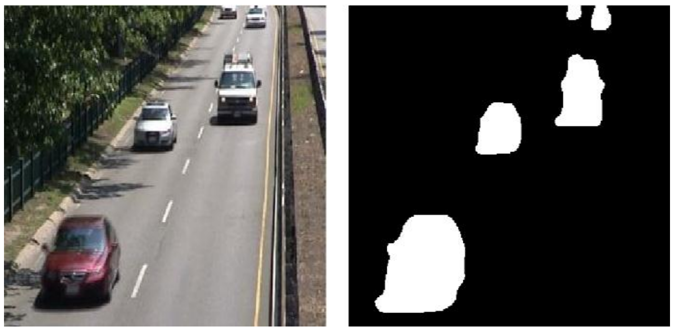

Fig -5: Frame of video before and after subtraction of background.

For scenarios like these, a number of algorithms have been developed; some of them are implemented in OpenCV, such as Background Subtraction MOG [12], which builds a model of the image's backdrop using Gaussian distributions. For this, it makes use of three to five Gaussian distributions. Based on and combining the background image estimating methodology with Bayesian segmentation,[13] is another background subtraction method that has been implemented in OpenCV.

4.2 Morphological Operations:

It is common practice to process binary images acquire from background subtraction or other image processing methods using morphological operations like dilation and closure. Here are the mathematical formulas for these operations along with an explanation.

A morphological process called dilation adds pixels to an object's edges in a picture. The basic concept is to scan the image with a structuring element (a tiny binary matrix) and set each pixel value to 1 if any of the structuring element's pixels overlap any pixels belonging to an object. The structuring element's size and form affect the dilation's extent.

$$M\ dilated\ (x, y) = \vee\_\{(i, j) \in B\}\ M\ (x + i, y + j)$$

Closing is a morphological process that fills in tiny gaps within an item in an image by combining dilatation and erosion. The main concept is to dilate the image with a structuring element, then degrade the outcome using the same structuring element. The object's overall shape is maintained while the little holes are successfully removed.

Mathematically, the following is the definition of the closing of a binary image M with a structuring element B:

$$M\ closed = (M\ dilated)\ eroded\ with\ B$$

where (M dilated) denotes the enlargement of M by the structuring element B and (eroded) denotes the erosion of the outcome by the same structuring element B.

When preprocessing binary images in preparation for additional analysis like connected component labelling or shape analysis, the closing operation is especially helpful. There may be noise and gaps in the binary mask produced by thresholding that need to be eliminated.

4.3 Contour Detections:

The limits of a shape are its contours, which are utilized to identify and recognize shapes. The deft edge detection carried out on a binary picture can be used to define the correctness of the contour finding procedure. To find the contours, use the cv2.findContours() method provided by OpenCV.

In order to create continuous curves or contours that outline the objects in the image, contour detection often requires locating portions of the image that have edges or intensity changes and linking such areas. Several algorithms, including the Laplacian of Gaussian (Log) filter, the Sobel filter, and the Canny edge detector, can be used to accomplish this.

In order to extract parameters like area, perimeter, and orientation or to identify certain items based on their shape or texture, the contours can be identified and then subjected to additional processing or analysis. A fundamental method in computer vision called contour detection serves as the foundation for many sophisticated algorithms utilized in areas including robotics, autonomous cars, and medical imaging.

4.4 Euclidean Distance Algorithm

A mathematical technique known as the Euclidean distance algorithm is used in computer vision to calculate the separation between two points in a two-dimensional space. In computer vision, this algorithm is commonly used to calculate the similarity between two images or to match features between two images.

$d = 2\sqrt{(a^2+b^2)}$. Hence, the distance between two points (a, b) and (-a, -b) is $2\sqrt{(a^2+b^2)}$.

The centroids of the bounding boxes are subjected to the Euclidean distance algorithm in a series of frames when an object is being tracked. The two bounding boxes are taken to represent the same object if the distance between two centroids is less than a predetermined threshold. The algorithm can determine the object's speed and direction of motion by comparing the position of the bounding boxes between frames.

5. SIMULATION RESULTS

The proposed system is implemented on python, using the OpenCV bindings. Moving objects are initially segmented using a background subtraction technique, after which noise is reduced and object shapes are improved using binary thresholding and morphological processes.

The segmented objects outlines are then extracted, and centroid tracking is used to track the segmented objects across frames. The code assigns an individual ID to each object and records its position and age.

The code then creates two fictitious lines on the screen, and counts the number of vehicles that cross them. An object is considered to be moving downhill or upward if it crosses the top line in that direction, and upward or downward if it crosses the bottom line.

The main loop reads frames from the video and uses morphological operations and background subtraction to recover the contours of moving objects. The process then repeats over the outlines to determine whether they satisfy the necessary conditions (such as area) to be classified as a vehicle. It determines if a detected vehicle is a brand-new one or an existing one and changes the position of any existing vehicles. Additionally, it counts the vehicles that cross the detection lines and changes the direction of movement.

The Euclidean Distance Tracker class, which keeps track of objects between frames, is created by the code as an object. Then, in order to find objects in the video, it creates a background subtractor object using the create Background Subtractor MOG2 method.

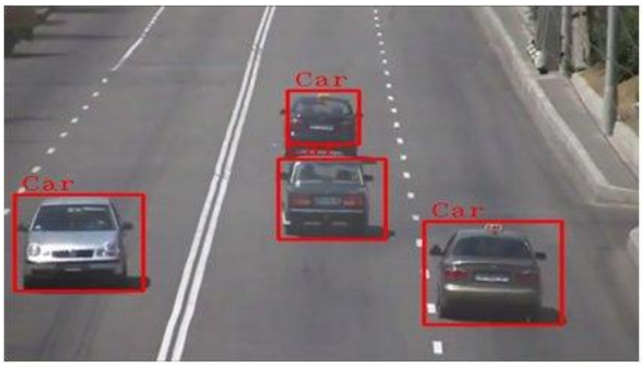

Fig -6: Vehicle Classification Results

In order to concentrate on the area of interest, which is the road where the vehicles are travelling, the code extracts a region of interest (ROI) from the frame. The ROI is then concealed using two distinct techniques. The ROI is subjected to the object detector in the first technique, which then thresholds the generated mask to reduce noise. The second technique applies a morphological opening and closing operation to the foreground mask before eroding it. This method takes a more complicated approach. The algorithm then tracks the discovered objects between successive frames using the Euclidean Distance Tracker object. Each object is given an ID, and based on its prior position, each object's position and speed are updated. Each item has a fixed speed limit of 30, and if an object goes over that limit, the code flags it as speeding by highlighting its ID in red.

It is essential to consider other methods and techniques that can also be used for vehicle speed detection. One such method is the optical flow method, which estimates the motion of objects in a sequence of images. Optical flow can be used to calculate the speed of vehicles by analyzing the change in position of the vehicle between consecutive frames. However, this method can be sensitive to lighting changes, object occlusion, and can suffer from inaccuracies in object detection.

Another technique that can be used for vehicle speed detection is the use of radar or LIDAR. These sensors use radio waves or laser pulses to measure the distance and speed of objects. These methods are highly accurate and can be used in various weather and lighting conditions. However, the cost of these sensors can be high, and they require specialized equipment and installation.

In comparison, the Euclidean distance algorithm is a simple and efficient method for vehicle speed detection. It uses object tracking to estimate the speed of the vehicle based on the change in position between consecutive frames. This method does not require specialized equipment and can be implemented using standard cameras. It is also less sensitive to changes in lighting and object occlusion compared to optical flow. However, the accuracy of this method can be affected by the accuracy of the object detection algorithm and the choice of distance metric used in the tracking algorithm. Overall, the Euclidean distance algorithm provides a cost-effective and practical solution for vehicle speed detection in various applications.

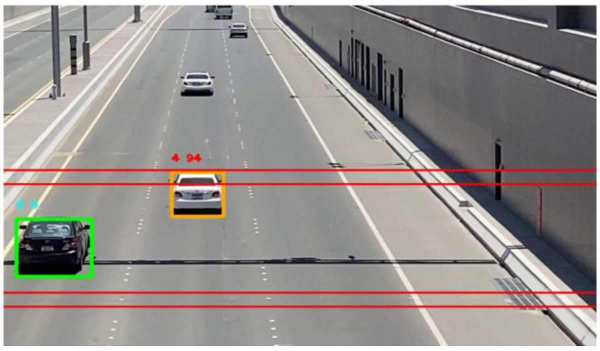

Fig -7: Results of vehicle speed detection

6. CONCLUSION

In this paper, we have implemented the proposed system on python , using the OpenCV. Computer vision techniques are utilized to detect the vehicle and counted the number of a vehicles that are passing on a particular street utilizing highway videos as input. At last, the vehicles were recognized and counted when they passed into the virtual detection zone. We have discussed the implementation of an object tracking method that uses Euclidean distance in particular. In order to improve tracking accuracy, we have also covered a variety of object identification and background reduction approaches. It shows how these strategies can be used in practice to track moving objects in a traffic video. In conclusion, the implementation of effective and precise tracking systems requires a solid understanding of computer vision concepts. Object tracking is a complex field with many methodologies and algorithms

7. REFERENCES

[1] Wang F-Y. Parallel control and management for intelligent transportation systems: concepts, architectures, and applications. IEEE 2010;11(3):630-638.

[2] Campolo C, Molinaro A, Scopigno R. From today's VANETs to tomorrow's planning and the bets for the day after.2015;2(3):158-171.

[3] Mao Q, Hu F, Hao Q. Deep learning for intelligent wireless networks: a comprehensive survey. IEEE 2018;20(4):2595-2621.

[4] Zhang C, Patras P, Haddadi H. Deep learning in mobile and wireless networking: a survey. IEEE 2019;21(3):2224-2287.

[5] Luong NC, Hoang DT, S. Gong, et al. Applications of deep reinforcement learning in communications and networking: a survey; 2018.1810.07862.

[6] Petho Z, Török Á, Szalay Z. A survey of new orientations in the field of vehicular cybersecurity, applying artificial intelligence based ˝ methods. 2021;e4325.

[7] Xiang, X., Zhai, M., Lv, N., & El Saddik, A. (2018). Vehicle counting based on vehicle detection and tracking from aerial videos. Sensors, 18(8), 2560

[8] Veni, S. S., Hiremath, A. S., Patil, M., Shinde, M., & Teli, A. (2021). Video-Based Detection, Counting and Classification of Vehicles Using OpenCV. Available at SSRN 3769139

[9] Kandalkar, P. A., & Dhok, G. P. (2017). Image Processing Based Vehicle Detection And Tracking System. IARJSET International Advanced Research Journal in Science, Engineering and Technology ISO 3297: 2007 Certified, 4(11).

[10] Hadi, R. A., Sulong, G., & George, L. E. (2014). Vehicle detection and tracking techniques: a concise review. arXiv preprint arXiv:1410.5894.

[11] Kamkar, S., & Safabakhsh, R. (2016). Vehicle detection, counting and classification in various conditions. IET Intelligent Transport Systems, 10(6), 406-413

[12]] 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.

[13] 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.