IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 8.102 ⁜ Peer-reviewed / Refereed journal ⁜ Vol. 12, Issue 5, May 2023

DOI: 10.17148/IJARCCE.2023.125189

Vehicle Speed Detection

Akanksha Kakde[1], Lavanya Sangode[2], Shivesh Kumar Singh[3], Yash Ladekar[4]

B.Tech 4th Year, Dept. of Computer Engineering, DYPSOEA, Pune, India[1-4]

Abstract: Vehicle Speed Detection and Estimation is an essential task for many traffic management and safety systems. In this research, we offer a novel computer vision-based method for real-time vehicle speed estimate and detection. To identify the cars and determine their speeds, our system first analyses video footage of the moving vehicles using image processing algorithms. On a dataset of actual traffic scenarios, we test the suggested algorithm, and the results show that it performs rather well in terms of accuracy. Potential uses for the suggested approach include traffic control, law enforcement, and intelligent transportation systems. Overall, this research helps to create methods for detecting and estimating vehicle speed that are accurate and efficient, which may greatly enhance traffic management and safety. The issue description, suggested solution, study methods, and findings are all briefly summarized in this abstract. It emphasizes the contribution to the area and draws attention to the research's importance and prospective uses. The abstract is succinct and informative, giving the reader enough details to comprehend the study and its significance to the subject.

Keywords: Image processing, computer vision, Nodejs, OpenCV, Image Processing, Moving Object Detection.

I. INTRODUCTION

Over the past few years, there has been a considerable increase in accidents due to the growth in the number of automobiles on the road. It is crucial to keep an eye on vehicle speed in specific zones or areas to increase.
Road safety. Although radar speed measuring methods are routinely employed for this purpose, they may have certain drawbacks, such as mistakes in identifying smaller automobiles with weaker echoes or detecting vehicles that changes speeds too quickly or frequently. As a result, there is a rising demand for systems that are more precise and effective for calculating the speed of moving vehicles.

In this paper, we suggest a unique method for quickly and correctly estimating the speed of the moving vehicle without the need of expensive sensors like radars. The suggested solution enters the moving vehicle video stream and filters it to calculate its speed.
There are four basics categorise. That may be used to group the components needed to complete this task:
1. Making a note of the subject area which is determined by the video.
2. Identifying car objects from the video frame.
3. Using a PPM (pixels per meters) algorithm to
estimate the velocity of a moving object.
4. Generating a precise image of the file to record the vehicle's speed and an image of the vehicle that was moving at that pace.

In this paper, we go into great details about the suggested methodology, including the methods and algorithms used to gauge the speed of moving vehicles. In section 2, we also give the experimental setup and assessment findings for the suggested strategy. Our findings show that the suggested method improves on conventional methods in terms of when it comes to determining the speed of moving vehicles.

The organization of this document is as follows. The suggested approach is thoroughly explained in **section 2**, along with the methods and algorithms employed to gauge the speed of the moving vehicles. The experimental setups and assessment findings for the suggested strategy are presented in **section 3**. they study is conclude in **section 4** with a review of the contributions and a discussion of the possible user and future development of the suggested approach.

- Implementation of the detection model is based on OpenCV.
- Implementation of website backend is done with the help of Python, Nodejs.

DOI: 10.17148/IJARCCE.2023.125189

- Databases

- MongoDB: for user sign up, and login.
- Supabase: for vehicle tracking id, and speed

II. METHODS AND MATERIAL

An The following section outlines the methods and materials utilized in the development of the vehicle speed detection is based on the python, OpenCV, NodeJS, Supabase and MongoDB. It describes the step-by-step process following to build the application, including the tools, frameworks, and Algorithms.

A. Requirement Analysis

- Conducted a thorough analysis of the requirements for the application, including features, functionalities, models that will be needed and security considerations.

B. Technology Selection:

- For detecting the object, which in this case is a vehicle from the video stream, OpenCV and dlib libraries were selected. This decision was made based on their efficiency and effectiveness in object detection tasks.
- Chose the NodeJS framework for its comprehensive end to end capability, suitable for web development, and efficient integration of frontend and backend technologies.

C. Database Design:

- MongoDB-(NOSQL) and Supabase -(PostgreSQL) are two different databases were used in the construction of a database schema to store user information and computed speed for respective cars. These databases were chosen because of their versatility in storing structured data and unstructured data as well as their aptitude for managing massive datasets.

D. Backend Development

- Implementation of the detection model is based on OpenCV.
- Implementation of website backend is done with the help of Python, Nodejs.
- Databases

- MongoDB: for user sign up, and login.
- Supabase: for vehicle tracking id, and speed.

E. Frontend Development

- Developed the user interface using html and JavaScript library for building user interfaces.
- Designed and implemented components for model window and user login/signup.

F. Algorithm of Vehicle Detection Model:

Algorithm for vehicle speed detection:

1.import the required libraries:

- Supabase
- Os
- Cv2
- Dlib
- Time
- Math
- Dotenv

2. Using the specified URL and key, establish a connection to the Supabase database.
3. load the cascade classifier for car detection.
4. open the video file for processing.
5.set the width and height for video frame processing.
6. Define a function “estimateSpeed” that calculates the seed of a vehicle based on its location.

DOI: 10.17148/IJARCCE.2023.125189

- The “estimateSpeed(location1, location2)” function uses the coordinate (x,y,and scale) of the item in two different frames to determine the estimated speed of the an object, such as an automobile , in a video. It utilizes the Euclidean distance formula to get the distance in pixels between the points, converts that distance to meters using a conversion factor (ppm) of 8.8 , determines the distance in meters, and uses a frame rate(fps) of 18. The distance in meters is then multiplied by the frame rate and a conversion factor(3.6) , and the result is the speed in kilometres per hour. The predicted speed is returned by the function.

7. Define a function “trackMultipleOjects” for tracking multiple vehicles in the video.

- trackMultipleOjects () is a function for tracking and estimating the speed of numerous objects like vehicle in a video. o process the video frames, identify objects, track their locations, and determine their speed based on those positions in successive frames, it makes use of the OpenCV and Dlib packages. The following are the code's primary steps:
  - Start by initialising the frame counter, the vehicle ID, the frame rate, the dictionaries used to store the data about the cars, and the output video file.
  - Continuously loop over the video input's frames.
  - Resize each frame and make a duplicate of it for presentation.
  - Update each car's tracking quality, and eliminate any that fall below a certain level.
  - Using 'carCascade.detectMultiScale()' on a grayscale version of the frame, detect cars every 10 frames.
  - Use Dlib's "correlation_tracker()" to compare the detected automobiles to the current tracked cars and build new trackers for mismatched cars.
  - In the resulting picture, draw rectangles around the tracked automobiles.
  - Using the 'estimateSpeed()' method, calculate each car's speed by comparing its location in the previous and current frames.
  - Using the Supabase library, insert or update the speed data in the database.
  - Show each car's estimated speed on the result image.
  - Write the output video file with the result image and display it in a window.
  - When the ESC key is used, the loop ends and the window is closed.
- Initialize variables for tracking cars, car numbers, and car locations.
- Create an empty list `speed` to store the speeds of vehicles.
- Optional Create a VideoWriter object to write the processed frames to an output video file.
- Start a loop to read frames from the video.
  - Read the current frame and resize it.
  - Increment the frame counter
  - Remove cars that are no longer tracked.
  - Every 10 frames, perform car detection and update the trackers.
  - Draw rectangles around the tracked cars.
  - Calculate the frames per second (fps).
  - For every car, update its location and estimate its speed.
  - Display the result image with the speed information.
  - Write the frame with rectangles to the output video file.
  - Check if the user pressed the Esc key to exit the loop.
- Release the video capture and destroy all windows.
- Release the VideoWriter object.
- Call the `trackMultipleObjects` function.

G. Model Architecture:

1. vehicle detection and speed detection model:

- Vehicle detection and classification have been
done by using Haar cascade.
- Vehicle speed prediction has been developed using OpenCV via image pixel manipulation and calculation.

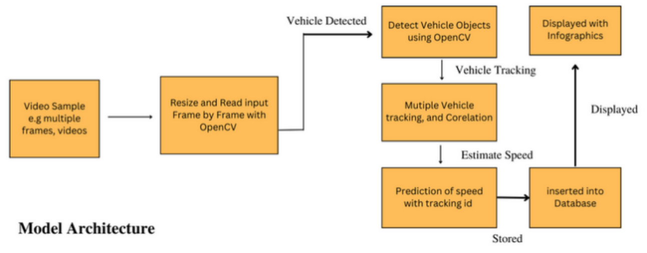
Video Sample e.g multiple frames, videos
Resize and Read input Frame by Frame with OpenCV
Vehicle Detected
Detect Vehicle Objects using OpenCV
Vehicle Tracking
Mutiple Vehicle tracking, and Corelation
Estimate Speed
Prediction of speed with tracking id
Stored
inserted into Database
Displayed
Displayed with Infographics
Model Architecture

Figure 1 : the figure shows how the video stream will be process in order to determine the speed of the vehicle.

2. use case diagram

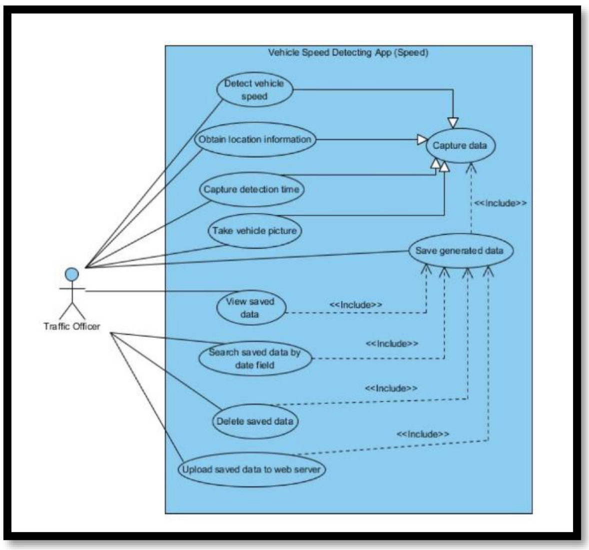

Figure 2: use case diagram of the system application

3.architectureal design:

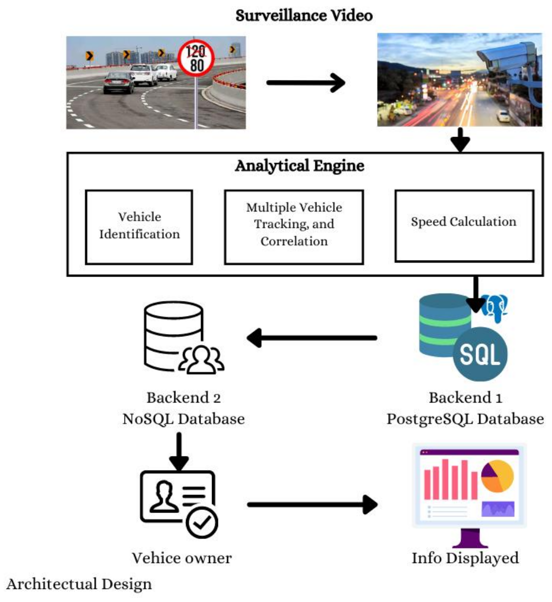

Figure 3:architectural design

H. Security implementation:

- Implementation security measures, such as data encryption, security user authentication, and protection against common vulnerabilities like cross-site scripting (XSS) and cross-site requires forgery (CSRF).

III. RESULTS AND DISCUSSION

A. Results

Using OpenCV, a well-known computer vision library, we created a vehicle speed detection system in this study. The technology was able to interpret video feeds. The system effectively detected and tracked cars in real-time by utilizing a variety of image processing techniques, such as background reduction, contour identification, and optical flow estimates.

After the cars were located, their speeds were determined by counting the distance they covered in a certain amount of time. After that, this data was saved in a database where it could be accessed for future monitoring and analysis. A set of standards, including detecting precision, processing speed, and computational effectiveness, were used to assess the system's performance.

B. Discussion

The results of the study show that the proposed vehicle speed detection system using OpenCV is effective in accurately detecting and tracking vehicles in real-time. The system demonstrated a high detection accuracy rate, proving its ability to correctly identify vehicles in various traffic conditions and lighting environments.

Moreover, the processing speed of the system proved to be sufficient for real-time applications, ensuring that the vehicle speed data could be accessed and monitored promptly. However, it is worth noting that the system's performance may be affected by factors such as camera angle, camera resolution, and weather conditions. Therefore, further research and optimization may be necessary to improve the system's robustness under these conditions.

One potential area of improvement is the integration of machine learning techniques, such as deep learning and convolutional neural networks, which could enhance the system's detection accuracy and adaptability to different scenarios. Additionally, incorporating more sophisticated tracking algorithms, such as Kalman filters or particle filters, could lead to better tracking performance and speed estimation.

In conclusion, the vehicle speed detection system developed using OpenCV demonstrates promising results in real-world applications, offering a valuable tool for traffic management and law enforcement. Further research and development can help optimize this system and broaden its applications across various traffic conditions and environments.

IV. CONCLUSION

In conclusion, traffic management and law enforcement organizations may greatly profit from the creation and application of a vehicle speed detecting system. The system can precisely identify and track vehicle speeds in real-time by utilizing computer vision methods, such as OpenCV, and potentially adding machine learning and sophisticated tracking algorithms. This makes it possible to spot speeding offences, offers useful information for traffic analysis, and eventually helps to increase overall road safety.

Even though the system now in use shows promising results, more investigation and optimization are required to guarantee its resilience and effectiveness in a variety of situations, including CAMERA ANGLES, weather, and lighting circumstances. Vehicle speed detection systems are projected to become even more useful tools for encouraging safe and responsible driving on our roads as computer vision and machine learning technology continue to progress.

V. REFERENCES

[1]. https://core.ac.uk/download/pdf/159180189.pdf

[2]. https://www.researchgate.net/publication/313844930_Vehicle_Speed_Detecting_App

[3]. https://www.researchgate.net/publication/342514938_Speed_Detection_using_IOT

[4]. https://ijariie.com/AdminUploadPdf/VEHICLE_DETECTION_AND_SPEED_TRACKING_SYSTEM_ijariie16695.pdf

[5]. T. Kumar, R. Sachan and D. S. Kushwaha, Smart City Traffic Management and Surveillance System for Indian Scenario, Proceedings of International Conference on Recent Advances in Mathematics, Statistics and Computer Science (ICRAMSCS), (2015).

[6]. International Conference on Recent Advances in Mathematics, Statistics and Computer Science (ICRAMSCS), (2015). [2] A. Mittal, A. Monnet and N. Paragios, Scene Modeling and Change Detection in Dynamic Scenes: A Subspace Approach, Computer Vision and Image Understanding, vol. 113(1), pp. 63–79, (2009).

[7] Zhou, X., Wang, D., Kr¨ahenb¨uhl, P.: Objects as points. arXiv preprint arXiv:1904.07850 (2019)

[8] O. Barnich and M. V. Droogenbroeck, ViBe: A Universal Background Subtraction Algorithm for Video Sequences, IEEE Transactions on Image Processing 2015, vol. 20, pp. 1709–1724, July (2011).

[9] S. Javed, U. D. L. Rochelle, S. K. Jung and L. Mia, OR-PCA with Dynamic Feature Selection for Robust Background Subtraction, Proceedings of the 30th Annual ACM Symposium on Applied Computing-SAC'15, pp. 86–91, (2015).