International Journal of Engineering Applied Sciences and Technology, 2023
Vol. 8, Issue 02, ISSN No. 2455-2143, Pages 70-73
Published Online June 2023 in IJEAST (http://www.ijeast.com)

VEHICLE DETECTION AND TRACKING SYSTEM

Abhay Lohani, Yash Tyagi, Anubhav Jindal
Student,
Computer Science and Engineering, HMRITM, Delhi, India

Veerendra Yadav
Professor,
Department of Computer Science and Engineering HMRITM, Delhi, India

Abstract: **The primary objective of this research is to design and implement an automated system for detecting vehicles from video footage, and estimating their speeds without the use of sensors. Object detection is a challenging task in computer vision that involves identifying and locating objects within an image or video. The proposed system begins by capturing the initial frame through a webcam, which serves as the reference frame. The system then calculates the phase difference between the reference frame and subsequent frames to detect movement. This refined frame is referred to as the threshold frame. Through the use of advanced image processing techniques such as shadow removal, dilation, and contouring, larger objects are identified within the threshold frame. This project also involves estimating the speed of vehicles using image processing techniques.**

Keywords: **Background Subtraction, roi Extraction, Thresholding, Morphological Operations, Contours, Object Tracking, Euclidean Distance.**

I. INTRODUCTION

1.1 Value of a Vehicle Detection System in the Market

The use of vehicle detection and tracking technology plays a crucial role in both civilian and military applications. These include highway traffic control, management, and urban traffic planning. The detection of vehicles on the road is crucial for tasks such as tracking, counting, determining average speed, traffic analysis, and categorizing vehicles. These processes can be implemented in different environments and under varying conditions. In this review, we provide a comprehensive summary of image processing techniques and analytical tools used in traffic monitoring. With the advancement of technology, monitoring traffic on busy roads is now less labor-intensive, cost-effective, and time-efficient. The use of object detection and tracking, behavioral analysis of traffic patterns, number plate recognition, and surveillance on video streams produced by traffic monitoring, can now be implemented on a large scale, which was not possible earlier due to technical limitations.

II. METHODOLOGY

2.1 Background Subtraction

Background subtraction is a technique used in computer vision and image processing to separate the foreground objects from the background in a video sequence. The main idea behind background subtraction is to model the background of the scene and then subtract it from the current frame to obtain the foreground objects.

We've used backgroundsubtractorMOG2 technique. MOG2 (Mixture of Gaussian Model version 2) is a background subtraction algorithm that uses a Gaussian Mixture Model (GMM) to model the background of a video sequence. The GMM is a probabilistic model that represents the background as a mixture of Gaussian distributions. The MOG2 algorithm uses a two-layer GMM to model the background, where the first layer represents the background model, and the second layer represents the foreground model. The MOG2 algorithm works by updating the GMM over time as new frames are processed. For each pixel in the current frame, the algorithm calculates the likelihood that the pixel belongs to the background or foreground model using a likelihood ratio test. The background and foreground models are then updated using a weighted average of the previous models and the new pixel.

MOG2 algorithm has some advantages over other background subtraction methods, such as: It is able to handle shadows and highlights in the background, which is a common problem with other background subtraction methods. It is able to adapt to changes in the background over time, which makes it more robust in dynamic environments. It is relatively fast and efficient, which makes it suitable for real-time applications.

2.2 ROI Extraction

Roi = frame[0:900,0:2000] This line of code defines a Region of Interest(ROI) from the given frame. This selects a rectangular region from the frame by specifying the starting and ending indices for both the rows and columns. In this case, it is selecting rows from index 0 to index 900 (0:900) and columns from index 0 to index 2000 (0:2000). The resulting region, roi, contains the pixels from the original frame that fall within the specified region, which is defined by the indices provided. It

International Journal of Engineering Applied Sciences and Technology, 2023
Vol. 8, Issue 02, ISSN No. 2455-2143, Pages 70-73
Published Online June 2023 in IJEAST (http://www.ijeast.com)

means that it is creating a new image, which is a sub-image of the original frame and the region of this sub-image is defined by the rows between 0 and 900 and columns between 0 and 2000.
fgbg.apply(roi) - The apply() method is used to apply the background subtraction algorithm on the region of interest (ROI) defined in the previous code. It means that the algorithm is applied on the ROI of the frame, which is defined by the rows between 0 and 900 and columns between 0 and 2000. The algorithm will update the background model and will identify the foreground objects in the given ROI.

2.3 Thresholding

cv2.threshold(fgmask,200, 255, cv2.THRESH_BINARY) It applies a threshold operation on the input image (fgmask) to create a binary image. It takes in the input image, a threshold value, and a maximum value, and returns a binary image where pixels with intensity values greater than the threshold are set to the maximum value. In this case, it is thresholding the fgmask image with a threshold value of 200, and setting all pixels greater than 200 to 255. The fourth argument cv2.THRESH_BINARY is a flag that specifies the type of thresholding to be applied. In this case, it is a binary thresholding, where the pixels are either 0 or 255, depending on whether their intensity value is above or below the threshold.
The function returns the threshold value that was used and the thresholded image. This thresholded image contains only black and white pixels, where black pixels represent the background and white pixels represent the foreground objects.

2.4 Morphological operations

Erosion: It is a morphological operation that involves eroding the boundaries of an object in an image. This is typically done by using a structuring element, which is a small matrix of pixels, to scan over the image. Pixels in the object that are not surrounded by pixels in the structuring element are removed, resulting in a shrinkage of the object. Erosion is often used in image processing to remove noise or to separate touching objects.
Dilation: It is a morphological operation that increases the size of the elements in a binary image. It is used to expand the boundaries of the objects in the image by adding pixels to the edges of the objects. The process is performed by moving a structuring element (a small matrix of pixels) over the image and replacing each pixel in the image with the maximum value of the pixels in the structuring element that overlaps with that pixel. This operation is typically used to fill in small holes or gaps in objects, or to connect separate objects that are close together.
cv2.morphology Ex(imBin, cv2.MORPH_OPEN, kernalOp) : It applies a morphological operation on the input image (imBin) using the structuring element (kernalOp). It is used to perform morphological operations such as erosion, dilation, opening, and closing on an image.
The first argument is the input image, the second argument is the morphological operation to be performed and the third argument is the structuring element, which is a matrix of 1s and 0s that defines the shape of the element used for the morphological operation. In this case, it is performing an opening operation on the input image "imBin" using the structuring element "kernel Op". Opening is an erosion operation followed by a dilation operation. It's used to remove noise from the image and to make small holes disappear. It means that the algorithm is performing opening operation on the binary image obtained by thresholding the foreground mask, which is the result of the background subtraction. This operation is used to remove noise and small holes from the image.
cv2.morphologyEx(mask1,cv2.MORPH_CLOSE, kernalCl) : It applies a morphological operation called "closing". The first argument, mask1, is the input binary image on which the operation is performed. The second argument, cv2. MORPH_CLOSE, specifies the type of operation to be performed. In this case, the operation is closing, which is a combination of dilation and erosion. The third argument, kernalCl, is the structuring element, or kernel, used for the operation. The kernel defines the shape of the neighborhood over which the operation is performed. In closing operation, it first applies dilation operation and then erosion operation. Together, they can be used to remove small holes or gaps in objects, or to connect separate objects that are close together.

2.5 Find Contours

cv2.findContours
(e_img,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
The first argument, e_img, is the input binary image, which is typically the result of a thresholding or edge detection operation. The second argument, cv2.RETR_TREE, specifies the contour retrieval mode. In this case, it is set to cv2. RETR_TREE, which retrieves all of the contours and organizes them into a full hierarchy. The third argument, cv2. CHAIN_APPROX_SIMPLE, specifies the contour approximation method. In this case, it is set to cv2. CHAIN_APPROX_SIMPLE, which compresses horizontal, diagonal, and vertical segments and leaves only their end points. The function returns two values, the first one is a list of contours, where each contour is represented by a list of (x, y) coordinates of the boundary points of the object. The second one is hierarchy, which describes the topological relationship between the contours. After this, we iterate through a list of contours, and for each contour, following actions are performed:
1. Calculating the area of the contour using the cv2. contour Area() function.
2. Checking if the area of the contour is greater than a certain threshold (in our case, 1000). If the area is greater than the threshold, it proceeds to the next step. If the area is not greater than the threshold, it moves on to the next contour.
3. Calculating the bounding rectangle of the contour using the cv2.boundingRect() function. The bounding rectangle is repre-

International Journal of Engineering Applied Sciences and Technology, 2023
Vol. 8, Issue 02, ISSN No. 2455-2143, Pages 70-73
Published Online June 2023 in IJEAST (http://www.ijeast.com)

sented by the coordinates of the top-left corner (x, y) and the width and height (w, h) of the rectangle.
4. Drawing a green rectangle around the object using the cv2. rectangle() function. The first argument is the image on which the rectangle is drawn (in this case, the "roi" image), the second argument is the top-left corner of the rectangle, the third argument is the bottom-right corner of the rectangle, the fourth argument is the color of the rectangle (in this case, green) and the last argument is the thickness of the rectangle lines.
5. Appending the bounding rectangle coordinates (x, y, w, h) to a list called "detections".

2.6 Object Tracking

In this, the tracker object is updated with new detection data and then it loops through the identified objects (boxes_ids) to draw a rectangle around them on the image (roi). It also checks the number of detections for each object, as well as the number of detections for each object, and it also uses a limit to check if the number of detections exceed the limit and it will change the color of the rectangle. It also calls the capture function of the tracker object with the object's current position and size, as well as the number of detections.

2.7 Euclidean Distance

It is used to track the objects across multiple frames, while the time difference between two lines is used to calculate the speed of the objects. This implementation is used to track the vehicles and calculate their speeds. We made an 'update' method which iterates through the rectangles of the detected objects in consecutive frames. For each rectangle, it finds the center point of the object and then calculates the Euclidean distance between the center point of the new object and the center point of the previously detected objects. If the distance is less than 70, it means the object has already been detected in the previous frame, and the center point of the object is updated. If the distance is greater than 70, it means the object is new and a new id is assigned to the object and the center point of the object is added to the 'center_points' dictionary.
As for speed calculation, we made a 'getsp' function which takes in one parameter 'id' and returns the speed of the object by dividing a constant value(214.15) by the time difference between the two lines. In the 'update' method, the time of the object crossing two specific lines (s1, s2) is stored in 's1' and 's2' arrays respectively. The time difference (s) between the two lines is calculated and stored in 's' array. By dividing a constant value by the time difference, the speed of the object is calculated.

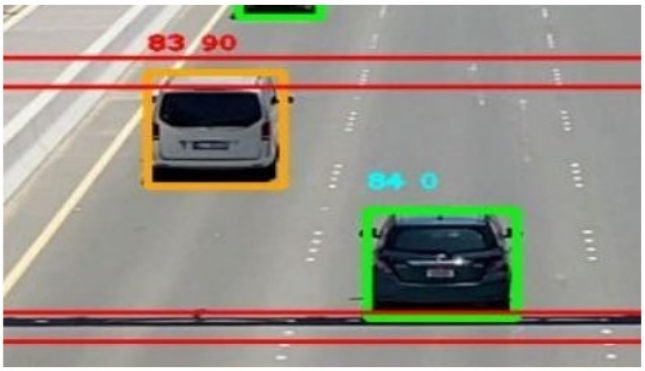

Fig. 1. Vehicle Speed Normal and Exceeded

The above image showing 2 vehicles with different speeds and color of their bounding box. The green one isn't calculated yet as the speed will get calculated after crossing the upper line. The orange box vehicle speed is more than the speed limit, hence, difference in color.

2.8 Saving Vehicle Data

For saving vehicle data i.e, vehicle id, speed and license plate number, we defined a method called "capture". It takes in several parameters:
self: the instance of the class that this method is being called on
img: an image (presumably in the form of a numpy array)
x, y, h, w: integers representing coordinates and dimensions of a rectangular region within the image
sp: an integer representing a speed value
id: an identifier for the object(vehicle)
The method first checks if a certain flag (self.capf[id]) is equal to 0. If it is, the flag is set to 1, another flag (self.f[id]) is set to 0, and a region of the image is cropped using the x, y, h, and w values.

Fig. 2. Saved Vehicle Images

This cropped image is then saved to a file (as you can see in the above image) with a specific naming format that includes the id and speed. Next, the method opens a file to write speed records, and checks if the speed (sp) is greater than a certain limit. If it is, the cropped image is saved to a different directory and the OCR(Optical Character Recognition) is applied to the image to extract the text from the image (license plate number) and the id, speed, and text are written to the file along with a message that the speed was exceeded. If the speed is not exceeded, only the id and speed are written to the file. Lastly, the method closes the file and increments a count of the total number of captures. It also increments a count of the number of captures where the speed was exceeded (if applicable).

2.9 Result

In this paper we have detected the vehicles with object detection algorithm and background subtraction and thresholding techniques and used mathematical calculation of Euclidean distance tracking algorithm for calculating speed of the vehicle and extracted the license plate number using OCR text recognition algorithm and saved the images of vehicles in the database with their speeds. Our method showed high accuracy and efficiency and can be used in traffic surveillance.

III. REFERENCES

[1]. Parin, P. and Pandi, G.Vehicle detection in traffic monitoring with machine learning - IJCRT. Available at: https://ijcrt.org/papers/IJCRT1807095.pdf. (2018).

[2]. Kandalkar, P.A. and Dhok, G.P. Review on Image Processing Based Vehicle Detection & tracking ... - IJSRST. Available at: https://ijsrst.com/paper/1709.pdf. (2017).

[3]. Marode, A. et al. Available at: https://www.irjmets.com/uploadedfiles/paper/volume3/issue_5_may_2 021/9973/1628083400.pdf. (2021).

[4]. Rad, A.G., Dehghani, A. and Karim, M.R. Vehicle speed detection in video image sequences using CVS method. Available at: https://www.researchgate.net/publication/266215176_Vehicle_speed_detection_in_video_image_sequences_using_CVS_method. (2010).

[5]. Cao, Y. et al. Research on vehicle detection and tracking algorithm based on the methods of frame difference and adaptive background subtraction difference| Atlantis Press. Available at: https://www.atlantis-press.com/proceedings/aiie-16/25866334.(2016).

[6]. Zhu, Q.; Li, H. and Guo, W. Research on Vehicle Detection and Direction Determinationbased on Deep Learning.Available at: https://www.scitepress.org/PublicationsDetail.aspx?ID=/o+VOkKTY Z8=&t=1.(2020).

[7]. Dwivedi, H. et al. Speed detection software, International Journal of Engineering Research & Technology. Available at: https://www.ijert.org/speed-detection-software.(2021).