International Journal for Modern Trends in Science and Technology, 6(12): 171-175, 2020

ISSN: 2455-3778 online
DOI: https://doi.org/10.46501/IJMTST061233
Available online at: http://www.ijmtst.com/vol6issue12.html

Traffic Light Optimization using OpenCV

Aditya Lahoty

B.Tech Scholar, Department of IT, Maharaja Agrasen Institute of Technology, Delhi, India

To Cite this Article
Aditya Lahoty, "Traffic Light Optimization using Open CV", *International Journal for Modern Trends in Science and Technology*, 6(12): 171-175, 2020.

Article Info
Received on 08-November-2020, Revised on 28-November-2020, Accepted on 02-December-2020, Published on 05-December-2020.

ABSTRACT

Traffic Light Optimization aims to find the solution for an increased amount of unnecessary waiting time on traffic signals. Traffic Signal Optimization is the process of changing the timing parameters relative to the length of the green light for each traffic movement and the timed relationship between signalized intersections using a computer software program. Our project aims to set the timer of green light based on real-time traffic congestion i.e. number of vehicles in a particular direction of the traffic light. To work in this project, we are using the OpenCV method to detect vehicles and then perform our calculation in the algorithm to predict the time for the green light to be in an active state.

KEYWORDS: *OpenCV, Vehicle Detection, Video Surveillance, Computer Vision*

I. INTRODUCTION

Traffic congestion is one of the worst problems in many countries. Traffic congestion wastes a huge portion of the national income for fuel and traffic-related environmental and socioeconomic problems [1]. Computer vision is a powerful tool for analysing complex and dynamic scenarios. It provides an appealing approach to analyse capturing processes. OpenCV helps decision-makers identify different possible options by analysing enormous amounts of data. Hence, computer vision can be used effectively to analyse traffic flow patterns and signal light timing. The proposed model is capable of optimizing signal light timing by the amount of traffic congestion at that particular signal. It also provides signal light timing for certain periods according to traffic demand. Video-based vehicle detection has played an important role in real-time traffic management systems over the past decade. Video-based traffic monitoring systems offer several advantages over traditional methods, such as loop detectors. In addition to vehicle counting, more traffic information can be obtained by video images, including vehicle classifications, lane changes, etc. Furthermore, video cameras can be easily installed and used in mobile environments [2]. Vehicle detection is fundamental data for a variety of transportation projects ranging from transportation planning to modern intelligent transportation systems and traffic optimization. Still 'Traffic Monitoring' and 'Information Systems' related to the classification of vehicles rely on sensors for estimating traffic parameters. Currently, magnetic loop detectors are often used to count vehicles passing over them [3]. Vision-based monitoring systems offer several advantages over earlier methods. In addition to detection, a much larger set of traffic parameters such as vehicle classifications, lane changes, parking areas, etc., can be measured in such types of systems [2]. In large metropolitan areas, there is a need for data about vehicle classes that should have a particular speed on a street. A detection system like the one proposed here can provide important data for a particular decision-making

agency. Our system uses a single camera mounted on a pole or other tall structure, looking down on the traffic scene. It can be used for detecting and classifying vehicles in a single lane and count the vehicles at act accordingly.

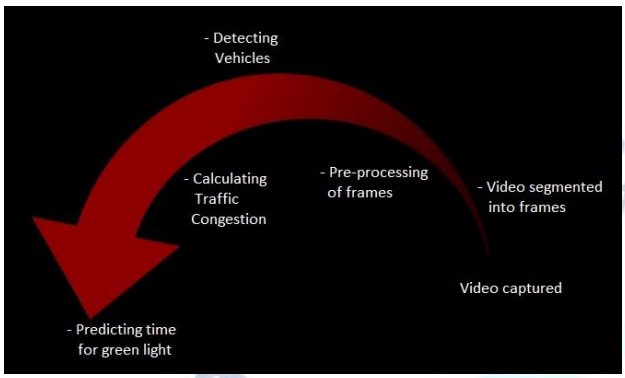

Figure 1: Flow Chart Traffic Light Optimization

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.

Nilakorn et al. [4], proposed a 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 a particular frame from a video. The proposed system provides limited functionality for the objects appearing in the detection zone if they are occluded or small.

A GMM was proposed by Friedman, N. and S. Russell.[5] and was refined for real-time tracking by Stauffer, C., and W.E.L. Grimson [6,7]. Gaussian Mixture Model relies on assumptions that the background is visible more frequently than any foreground regions. Elgammal proposed a Kernel density estimation based nonparametric background model [8]. Kernel density estimation method evaluates the samples of video data using kernel functions and it samples data that has maximal probability density as the background is selected.

Tursun, M and Amrulla, G [9] proposed a video-based real-time vehicle counting system using an 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 a virtual loop.

Bhushan et al. [10], proposed a vehicle detection method based on morphological operations including binarization, edge detection, and top-hat processing, masking operation. The proposed system fails to give good results in a cloudy environment.

Another video-based vehicle counting system was proposed by Lei, M., et al. [11]. In this system surveillance cameras were used and mounted at a relatively high place to acquire the traffic video stream, 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.

Detection of vehicles in a video-based traffic surveillance system is the 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 moving images.

Bas et al. proposed a video analysis method to count vehicles [13] based on an adaptive bounding box size to detect and track vehicles following the 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.

Habibu Rabiu proposed a vehicle detection and classification for cluttered urban intersections [14]. 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.

An improved frame differencing method was presented by Collins [15] which uses the difference between multiple frames to compute the foreground instead of just using the initial frame.

A. Suryatali and V.B. Dharmadhikari in [16], proposed a Computer Vision-based vehicle detection that counts and also classifies the vehicles into heavy and lightweight 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 the processed frame.

III. Proposed Methodology

In our model, we are first detecting the vehicles and then finding the area covered by these vehicles on the road i.e., traffic congestion based on which we are holding the green light active state.

There are certain key concepts in the vehicle detection system using OpenCV. Our objective is to capture the coordinates of the moving object and highlight that object in the video [17].
We would want our model to detect the moving object in a video as illustrated in the image above. The moving car is detected and a bounding box is created surrounding the car [17].

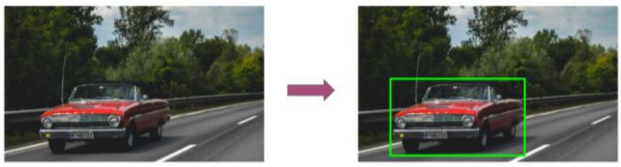

Figure 2: Vehicle Detection[17]

There are multiple techniques to solve this problem. In this model, we will focus on the unsupervised way of object detection in videos, i.e., object detection without using any labelled data[17].

The key concepts used are:

- Background Subtraction
- Frame Differencing
- Image Thresholding
- Morphological Transformation
- Contour Finding

3.1 Background Subtraction

There are many different algorithms for background subtraction, but the main idea of them is very simple [18].

foreground_objects = current_frame - background_layer

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 the 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 roadside, 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. The consists of three steps, background subtraction, image enhancement, and foreground extraction. The 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 [19].

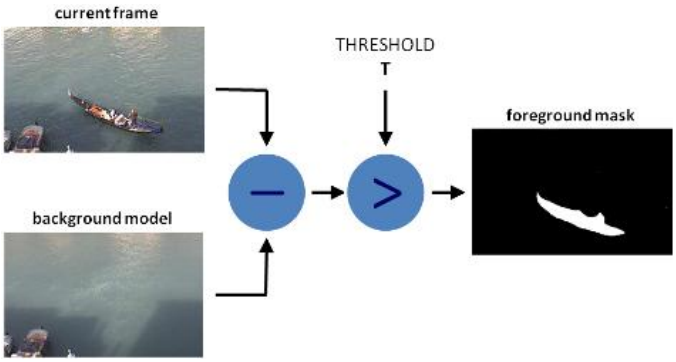

Figure 3: Background Substraction[18]

3.2 Frame Differencing

A video is a set of frames stacked together in the right sequence. So, when we see an object moving in a video, it means that the object is at a different location at every consecutive frame [17]. If we assume that apart from that object nothing else moved in a pair of consecutive frames, then the pixel difference of the first frame from the second frame will highlight the pixels of the moving object.

Now, we would have the pixels and the coordinates of the moving object. This is broadly how the frame differencing method works [17].

3.3 Image Thresholding

If the pixel value is greater than a threshold value, it is assigned one value (maybe white), else it is assigned another value (maybe black). The function used is cv2.threshold. The first argument is the source image, which should be grayscale. The second argument is the threshold value which is used to classify the pixel values. The third argument is the maxVal which represents the value to be given if the pixel value is more than (sometimes less than) the threshold value. OpenCV provides different styles of thresholding and it is decided by the fourth parameter of the function [20].

3.4 Morphological Transformation

Morphological transformations are some simple operations based on the image shape. It is normally performed on binary images. It needs two inputs, one is our original image, the second one is called the structuring element or kernel which decides the nature of the operation. Two basic morphological operators are Erosion and Dilation [21].

3.4.1 Erosion

The basic idea of erosion is just like soil erosion only, it erodes the boundaries of the foreground object and always tries to keep the foreground in white. The kernel slides through the image as in 2D convolution. A pixel in the original image either 1 or 0 will be considered 1 only if all the pixels under the kernel are 1, otherwise, it is eroded i.e., made to zero [21].

All the pixels near the boundary will be discarded depending upon the size of the kernel. So the thickness or size of the foreground object decreases or simply white region decreases in the image. It is useful for removing small white noises [21].

3.4.2 Dilation

Dilation is just the opposite of erosion. Here, a pixel element is '1' if at least one pixel under the kernel is '1'. So, it increases the white region in the image, or the size of the foreground object increases. Normally, in cases like noise removal, erosion is followed by dilation. Because, erosion removes white noises, but it also shrinks our object. So we dilate it. Since the noise is gone, they will not come back, but our object area increases. It is also useful in joining broken parts of an object [21].

3.5 Finding Contours

Contours can be explained simply as a curve joining all the continuous points (along the boundary), having the same colour or intensity. The contours are a useful tool for shape analysis and object detection and recognition [22].

There are three arguments in cv.findContours() function, the first one is the source image, the second is contour retrieval mode, third is the contour approximation method. And it outputs the contours and hierarchy. Contours is a Python list of all the contours in the image. Each contour is a Numpy array of (x,y) coordinates of boundary points of the object [22].

The contours are used to identify the shape of an area in the image having the same colour or intensity. Contours are like boundaries around regions of interest. The white regions have been surrounded by greyish boundaries which are nothing but contours. We can easily get the coordinates of these contours. This means we can get the locations of the highlighted regions[17].

Note that there are multiple highlighted regions and each region is encircled by a contour. In our case, the contour having the maximum area is the desired region. Hence, it is better to have as few contours as possible[17].

The above process is about the vehicle detection process.

Since our camera is just above the road on which the vehicles are getting passed, so to calculate the time for which the green light will be in an active state, we calculate the area of the frame covered by the contours and check that whether the area covered is greater than a particular threshold value. If the area is greater than the green light remains in the active state and when the area goes beyond the threshold value, it means there are very few vehicles on the road currently, so we change the signal to red.

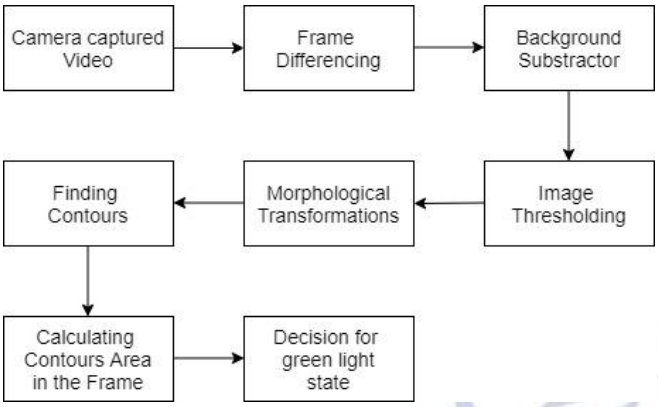

Figure 4: Block Diagram of the proposed algorithm

IV. Conclusion And Future Work

The proposed solution is implemented on python, using the OpenCV bindings. The traffic camera footages from a variety of sources are in implementation. Currently proposed system works with already captured videos but it can be modified to be used for processing live video streams by adding microcontrollers.

One of the limitations 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 the 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 a better view of the road and increasing the efficiency. The system is not capable of detecting vehicles at night as it needs the foreground objects to be visible for extraction of contour The system could also be improved for better accuracy using the more sophisticated image and artificial intelligence operations.

References

[1] Hewage, Kasun & Ruwanpura, Janaka. (2004). Optimization of Traffic Signal Light Timing Using Simulation. 1428-. 10.1109/WSC.2004.1371482.

[2] https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.90.700&rep=rep1&type=pdf

[3] Uke, Nilesh & Thool, Ravindra. (2012). Moving Vehicle Detection for Measuring Traffic Count Using OpenCV. Journal of Automation and Control Engineering. 10.12720/joace.1.4.349-352.

[4] 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.224-227, 2016

[5] 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

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

[7] 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

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

[9] 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.

[10] 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.

[11] M. Lei, D. Lefloch, P. Gouton, K. Madani, "A videobased 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

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

[14] 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.

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

[16] 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

[17] https://www.analyticsvidhya.com/blog/2020/04/vehicle-detection-opencv-python/

[18] https://medium.com/machine-learning-world/tutorial-making-road-traffic-counting-app-based-on-computer-vision-and-opencv-166937911660

[19] Sheeraz Memon, Sania Bhatti, Liaquat A. Thebo, Mir Muhammad B. Talpur, Mohsin A. Memon, " A Video based Vehicle Detection, Counting and Classification System", International Journal of Image, Graphics and Signal Processing(IJIGSP), Vol.10, No.9, pp. 34-41, 2018.DOI: 10.5815/ijigsp.2018.09.05

[20] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html

[21] https://docs.opencv.org/master/d9/d61/tutorial_py_morphological_ops.html

[22] https://docs.opencv.org/master/d4/d73/tutorial_py_contours_begin.html