**Vehicle detection and tracking in a video using Gaussian Mixture model**

Roopali Rajaram Sarode

Department of Computer Science

New Jersey Institute of Technology

Newark, NJ 07102

Contact: rs366@njit.edu

**Abstract**

The Gaussian Mixture Model has been popularly used for tracking and counting vehicles in the Intelligent Transport System (ITS). The GMM is basically responsible for extracting moving objects (input image) from stored background images (static image) or generated background frames from a video. The main idea behind the model is setting the background image and detecting the changes that occur in the foreground. The Mixture of Gaussians or MoG is a mixture of k Gaussians distribution models for each background pixel. Different distributions each represent the different background and foreground colors. The weight of each one of those used distributions on the model is proportional to the amount of time each color stays on that pixel. Therefore, when the weight of pixel distribution is low then that pixel is classified as a foreground pixel.

**I. INTRODUCTION**

Traffic surveillance remains one of the most significant applications of Video-based supervision systems. There have been several researches on Vision-based Intelligent Transportation System (ITS) for applications like transportation planning and traffic engineering applications to extract useful and precise traffic information for traffic image analysis and traffic flow control. Due to the increase in computing power of personal computers the creation of complex algorithms to detect foreground objects and even track them has been made possible. The camera used to obtain the video is stationary as it is easier to detect vehicles from stationary cameras which are also placed on a higher angle on the lane. The GMM method compares between the background and foreground objects.

In this paper, a detection and segmentation technique followed by a contour tracking method is used to detect and keep track of vehicles passing by in a traffic video. On the basis of this division, further segmentation and tracking techniques are applied to detect an object. Python has been used to implement object detection and tracking algorithms as Python is easy to understand. The arrangement of this paper is structured as follows : In section II Background subtraction model - Gaussian Mixture Model(MOG2) for image segmentation is explained. Section III explains morphological operations performed on the image to reduce noise and help for better detection of the object. Section IV explains the contour tracking method to detect the object. Section V gives the counting and tracking of object methodology recruited in the paper and Section VI provides an overview of the algorithm used as well as a flow diagram of the algorithm followed by Section VII - Conclusion, providing the summary of overall detection and tracking process.

**II. Background subtraction model- Gaussian mixture model (MOG)**

Every single frame of a video can be classified into two different areas : foreground and background. The foreground represents the objects of interest (vehicles, pedestrians etc). On the other hand, the background represents the static part of the frame such as the sky, trees, buildings etc.

Gaussian Mixture Model(GMM) and Mixture of Gaussians(MOG) are equivalent terms used to represent the same concept of background subtraction. Due to its reliability against the changes in light and condition during repeated object detection, this method works well for background extraction. GMM or MOG is a mixture of k Gaussian distributions models each background pixel with values within 3 to 5 for k. Each distribution’s weight is proportional to the amount of time each color stays on that pixel. Therefore, when the weight of the pixel distribution is low, that pixel is classified as foreground and pixels with similar values under standard deviation and high weight factor are considered as background. The MOG implementation on OpenCV has various input parameters to customize the behavior of the method. Below is the syntax for MOG background subtraction function.

*createBackgroundSubtractorMOG2([, history[, varThreshold[, detectShadows]]])*

The parameter “history” is responsible for the number of frames the method will use to gather weights on the model, throughout the entire processing period. The default value for this parameter is 500. varThreshold parameter correlates the pixel’s weight value on the current frame with values on the model. In other words it determines whether a pixel is well defined by its background model, the default value for this parameter is 16. Lastly, the parameter detectShadows enables or disables shadow detection. Whether shadows are given importance or not depends on this parameter. It can be set to either true or false. Enabling this parameter increases processing time.

Pixels are then assigned into categories : If the pixel color is categorized as a background model then the pixel will be given value zero (0) or black color. Whereas if a pixel is uncategorized in the background model then it will be assigned a value of one (1) or white color and will be considered as a foreground object. Foreground is a moving object and changes position throughout the video (dynamic) while background is an object which remains static in every frame of the video.



Figure 1: Image after background subtraction and gaussian blur filter application

**III. Filtering using morphological operations**

After the background subtraction or foreground object detection process, the image is converted into a grayscale image. Filtering is done on the image to fill in the holes on the detected object and filter the noise from the image. Morphology processes are performed on the detected object to clarify the object. A gaussian blur filter is applied on each frame extracted from the video. The filter smoothens the image by blurring it, in other words it removes outlier pixels that may be noise in the image. The filtered image is then convoluted with a structuring element. A three by three elliptical shape kernel is used as the structuring element.

The morphological operations help in increasing the accuracy of the object detection. Closing operation is performed on the object, which is basically dilation followed by erosion. Dilations increase the size of foreground objects and are especially useful for joining broken parts of an image together and erosion is useful for removing small blobs in an image or disconnecting two connected objects. It is useful in closing small holes inside foreground objects, or small black points on the object.

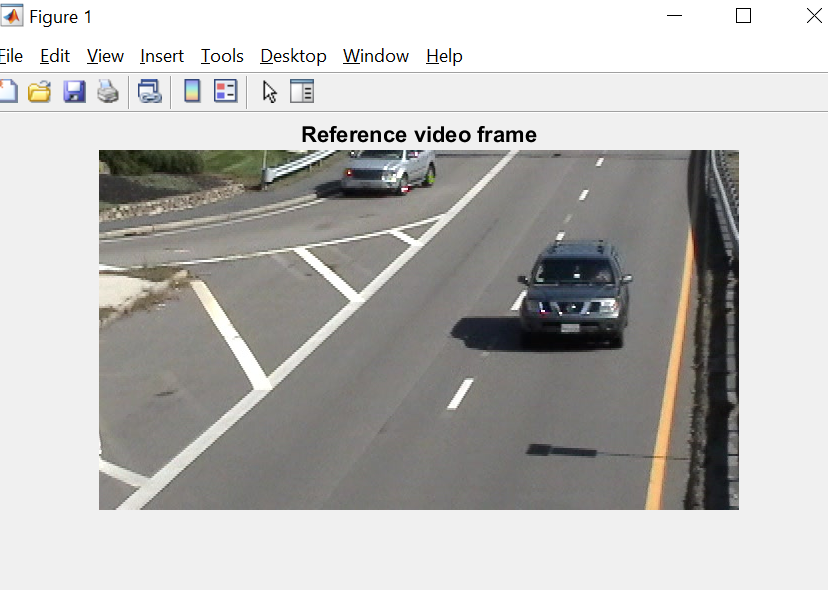
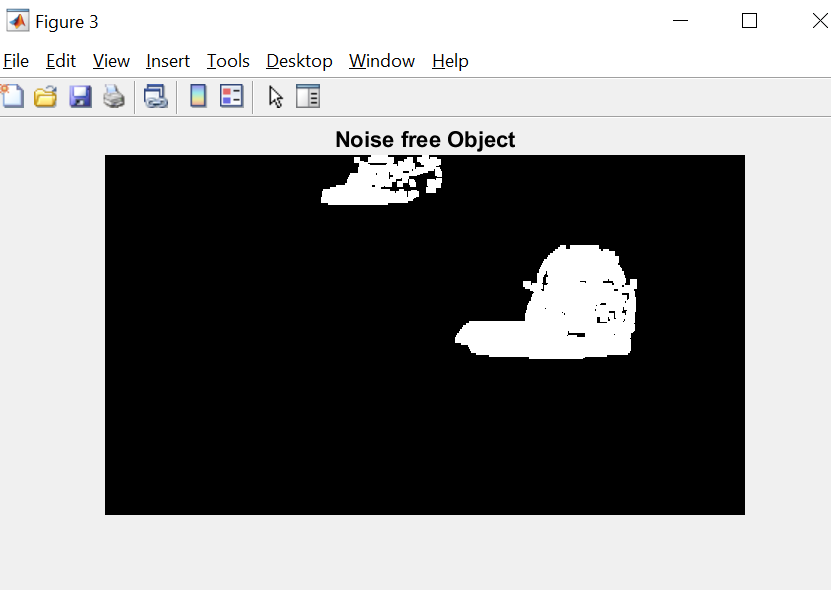
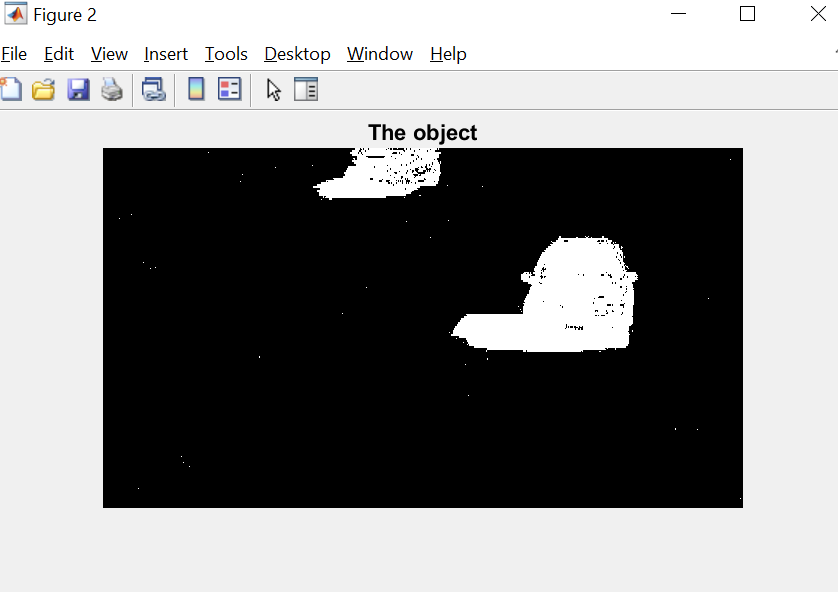


Figure 2 : A frame from the input video

 Figure 3: Image after background subtraction Figure 4: Image after noise removal

**IV. Contour Tracking Method**

Contours are curves joining all the continuous points (along the boundary), having some color or intensity. The findContours method from OpenCV simply detects change in the image color and marks it as contour. For better results this operation is performed on gray image or binary image. The syntax for the method is : cv2.findContours(src, contour\_retrieval, contours\_approximation). The method can be customized with the help of input parameters like contour retrieval mode, contour approximation method. The first parameter (src) takes the input image of an n dimensional array. The second parameter determines the retrieval mode of the contours of the object. The output of the retrieval mode is array hierarchy which shows how various contours are linked to each other, their relation with other contours, parent child relation. In this paper, cv2.RETR\_TREE has been used. It retrieves all of the contour pixel points and reconstructs a full hierarchy of nested contours. The last parameter is contours\_approximation which represents the method in which you want to store the contours. Instead of storing all the coordinates of the boundary points, cv2.CHAIN\_APPROX\_SIMPLE excludes all excessive points and compresses the contour, thereby saving memory. It compresses horizontal, vertical and diagonal segments and leaves only their end points.

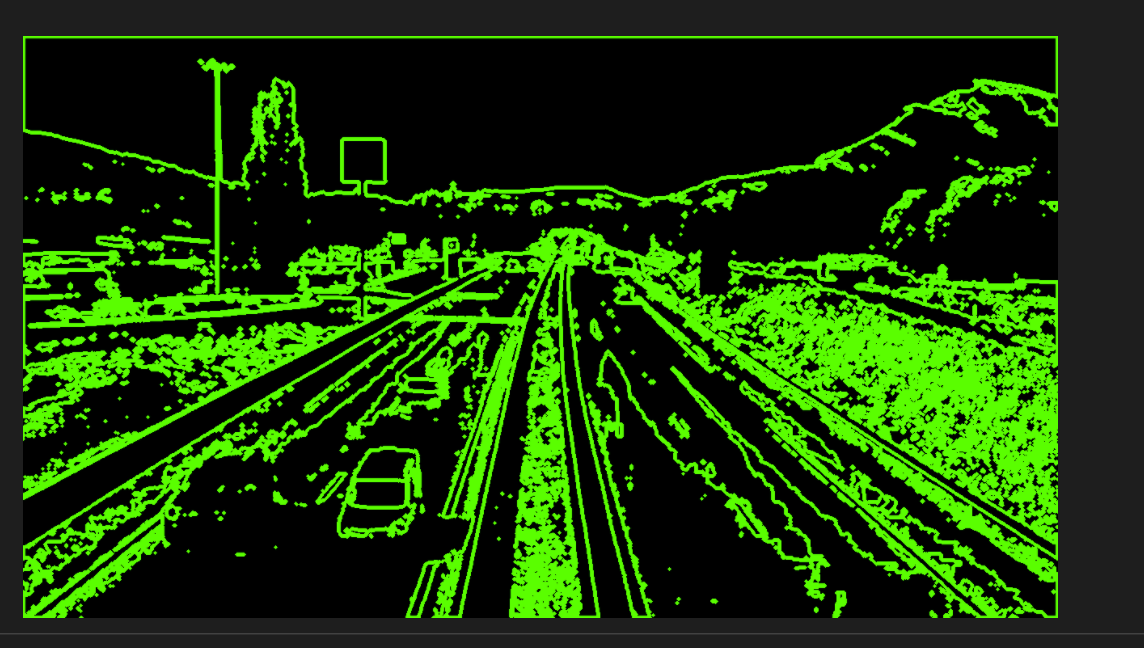
 

Figure 5: Original image Figure 6: Finding contours from an image

**V. Counting and Tracking of Objects**

Once the video frame (image) undergoes background subtraction, morphological operations like filtration for noise removal followed by contour tracking for the foreground object (vehicle), it is now time for counting the total number of vehicles in the frame. In order to achieve that a function which can highlight the area of interest after obtaining the object’s outer shape(contour) is used. The function used for this study is cv2.boundingRect(), it creates an approximate rectangle along the foreground object. The function returns a total of 4 points x,y,w,h which corresponds to X coordinate, Y coordinate, Width and Height of the rectangle.

After surrounding the vehicles with rectangles in the image, we then go on with counting them throughout the video. Two different counters are utilized here : one to keep track of the total number of vehicles in the video and another to enumerate each vehicle. The counters would be displayed on the output video to help keep track of the number of vehicles passing by. In addition to that, a red dot is placed in the center of the rectangle to further highlight the vehicle which is being counted.

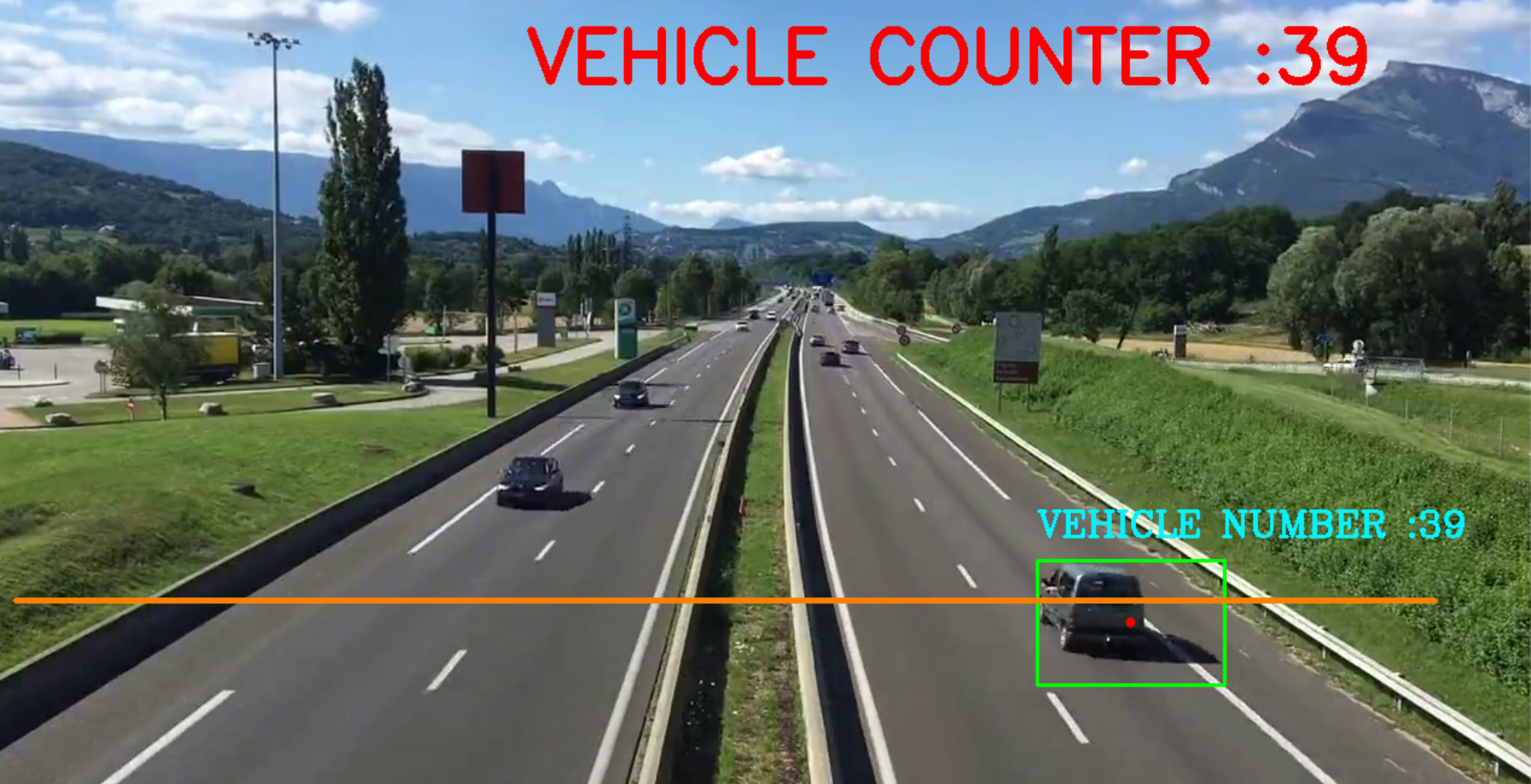


Figure 7: Output video snippet

**VI. ALGORITHM**

**Step 1**: Read the video and store in a variable

**Step 2**: Variable and counter initiations

1. provide a minimum width and height for the rectangle around the vehicles
2. create a counter for keeping count of object(vehicles) detected and a counter for individual vehicle number display on their respective rectangles
3. create a line in the video frame, passing the line would increment the counter for vehicle count
4. decide on the position of the line in the video
5. an offset value(allowable error between pixel) while the vehicles are counted when they cross the line
6. define a function for red dot on the vehicles while counting

**Step 3**: Preprocessing

1. Read frames from captured video
2. Convert the captured video frame to grayscale
3. Convolution of the gray image with blurring gaussian filter
4. Apply gaussian mixture based background segmentation algorithm on the convoluted image
5. Morphological operation: Dilating image from step 4 with 3X3 filter matrix of ones
6. Generate an elliptical structuring element
7. Perform closing(Dilation followed by erosion)for closing small holes inside the foreground objects, or small black points on the object.

**Step 4**: Find contours of the foreground object(vehicles) in the image with the help of contour retrieval mode and contour approximation method parameter

**Step 5**: Iterate through the counted objects(vehicles)to put a rectangle around if their width and height is greater than the minimum width and height given by user

**Step 6**: Set a text box on top of the rectangles of each vehicle to display its count

**Step 7**: Detect the center of the object, place a red dot on it while counting the object and append the count to the total vehicle counter

**Step 8**: Update the counter for counting total vehicles in the frame

**Step 9**: Display the total number of vehicles count on top of the output video

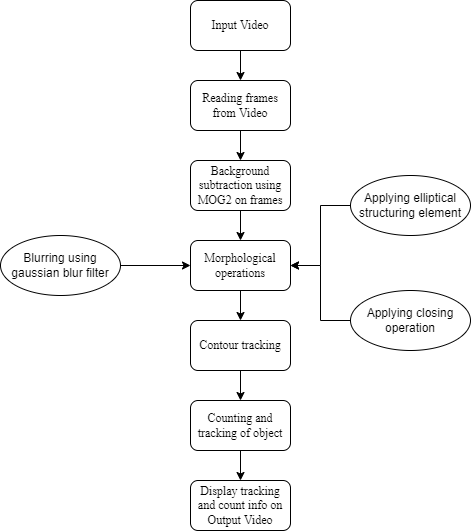


Figure 7: Flow chart of the algorithm

**VII. Conclusion**

This paper aims to conduct detection and tracking of vehicles from a video taken from a stationary camera using low-cost techniques. To detect the foreground object Mixture of Gaussian or Gaussian Mixture Model technique has been used. Morphological operations like smoothening, noise removal using a structural element, dilation and erosion have been performed on the video frame as preprocessing steps. After that contours of the foreground object (vehicle) in the video are identified by specifying the retrieval mode and pixel point approximation method. The background subtraction method which works well under proper illumination, moderate traffic conditions and a variety of camera angles. However, under poor lighting conditions, ambiguous background and heavy traffic it seems to slack off in performance.

**REFERENCES**

[1] Raad Ahmed Hadi, Ghazali Sulong and Loay Edwar George, “VEHICLE DETECTION AND TRACKING TECHNIQUES: A CONCISE REVIEW,” *Signal & Image Processing : An International Journal (SIPIJ) Vol.5, No.1, February 2014*

[2] Muhammad Moin Akhtar, Yong Li, Lei Zhong and Ayesha Ansari, Vehicle Detection, Tracking and Counting Using Gaussian Mixture Model and Optical Flow: *Journal of Engineering Research and Reports, 15(2): 19-27, 2020; Article no.JERR.59118ISSN: 2582-2926*

[3] Tamilarasu Viswanathan, N. Vinothkumar, “Detecting Vehicle Motion using Deep Gaussian Mixture Model with SCI-kit Learn,” *International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-8, Issue-6S3, September 2019.*

[4] H. Chung-Lin and L. Wen-Chieh, "A vision-based vehicle identification system," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on, 2004, pp. 364- 367 Vol.4*

[5] K. H. Lim, et al., "Lane-Vehicle Detection and Tracking," *Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS 2009), vol. 2, pp. 5–10, 2009.*