Real Time Tracking

We used the approach mentioned in the shared research papers for object tracking. The algorithm was implemented using basic matrix algebra and numpy.

Dataset - We are using a traffic IP camera video from YouTube of a length of 30 seconds. This is the Link to the dataset.

Methodology

1) The input video was processed using opencv2 to extract frames out of it.
2) For each image location, we stored pixel values across all the frames in an array and passed it as input to the Gaussian_Mixture function to represent the pixel values as a mixture of two Gaussians.
3) One of the Gaussians will represent the background and the other will represent the foreground of a particular pixel across all the frames.
4) As the background remains the same for most of the frames, the mean of the gaussian that has more weight and less standard deviation is used for constructing the background.
5) To track the object, we subtract this background from each frame.

Algorithm (Gaussian Mixture Model)

1) For this assignment, we are representing the pixel values across all the frames as a mixture of two Gaussians.
2) We start by initializing the values of mean and variance for both the gaussians.
3) We calculate the likelihood of each observation x_i using the initial values of mean and variance.
4) For both the gaussian clusters, we calculate our data's probability density (pdf) using the initial mean and variance values.

$$f(\mathbf{x}|\mu_k, \sigma_k^2) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp(-\frac{(\mathbf{x}-\mu_k)^2}{2\sigma_k^2})$$

5) Then, we can calculate the likelihood of a given example x_i to belong to the k^{th} cluster.

$$\mathbf{b}_k = \frac{f(\mathbf{x}|\mu_k, \sigma_k^2)\phi_k}{\sum_{k=1}^{K} f(\mathbf{x}|\mu_k, \sigma_k^2)\phi_k}$$

6) Using Bayes Theorem, we get the posterior probability of the kth Gaussian to explain the data. That is the likelihood that the observation x_i was generated by k^{th} Gaussian. We

have initialized the weights to 0.5 for both the gaussians, since we don't have any information to favor one gaussian over the other.

7) In the next step, we re-estimate our learning parameters as follows.

$$\mu_k = \frac{\sum \mathbf{b}_k \mathbf{x}}{\sum \mathbf{b}_k} \qquad \sigma_k^2 = \frac{\sum \mathbf{b}_k (\mathbf{x} - \mu_k)^2}{\sum \mathbf{b}_k} \qquad \phi_k = \frac{1}{N} \sum \mathbf{b}_k$$

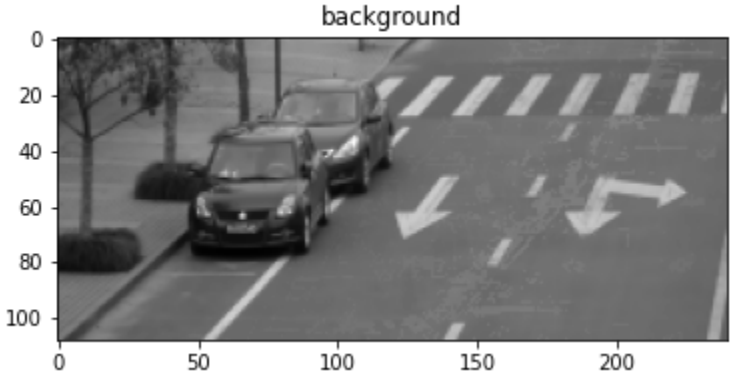
8) We will repeat the same process until the values of mean, variance and weights of boththe Gaussians converge.

Limitations

- The implementation is not an optimized one in terms of time-complexity and is only intended to deliver the concept.
- The approach can fail when the foreground objects are present in the video for a larger duration of time. However, in such scenarios, taking a video of a longer duration as input is expected to yield better results.

Results

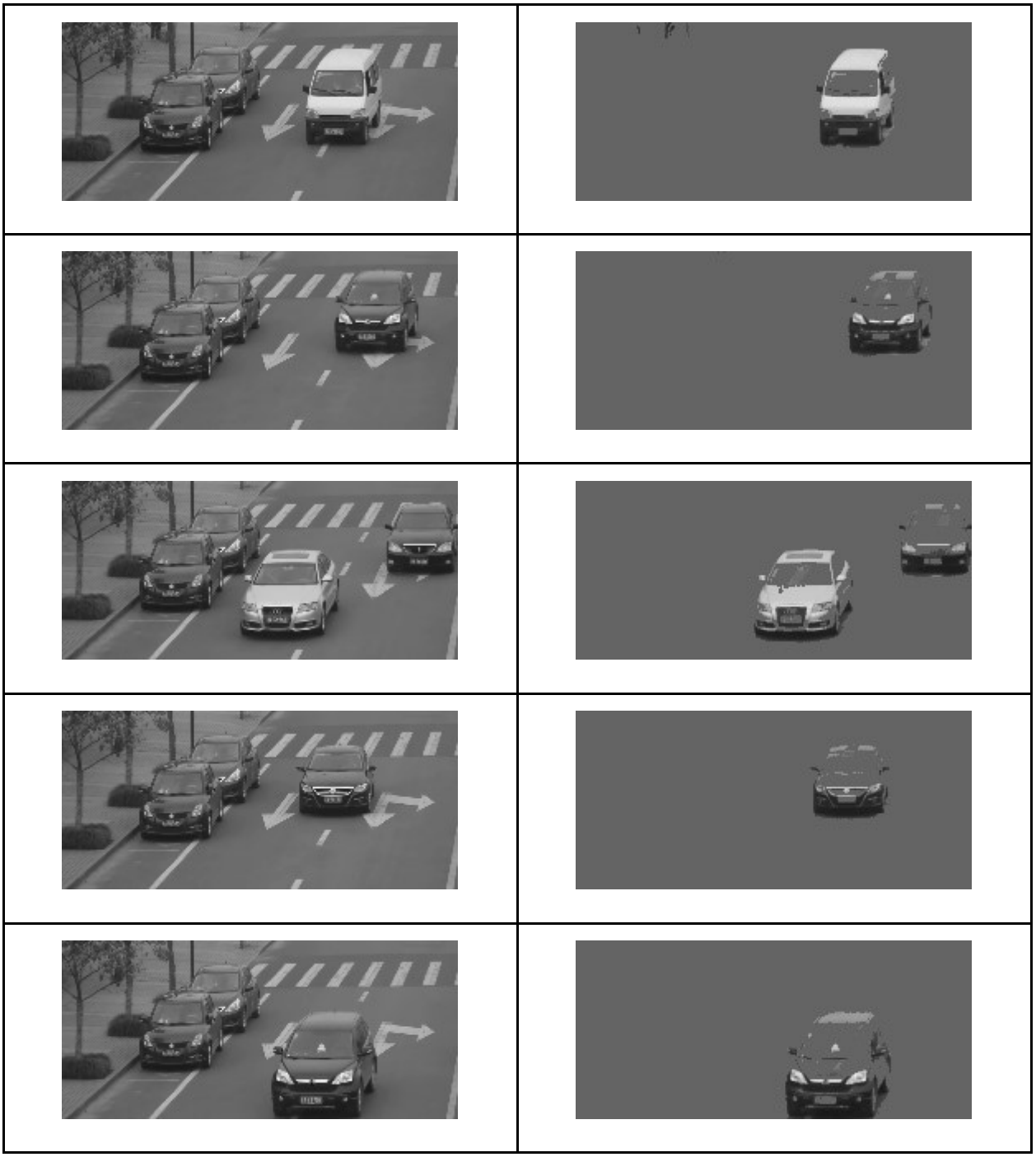
The following background image is obtained on the successful execution of the algorithm described above. Note that the image is devoid of any moving vehicles.



The final video that detects the moving object can be found here.It is also attached in the zip file.

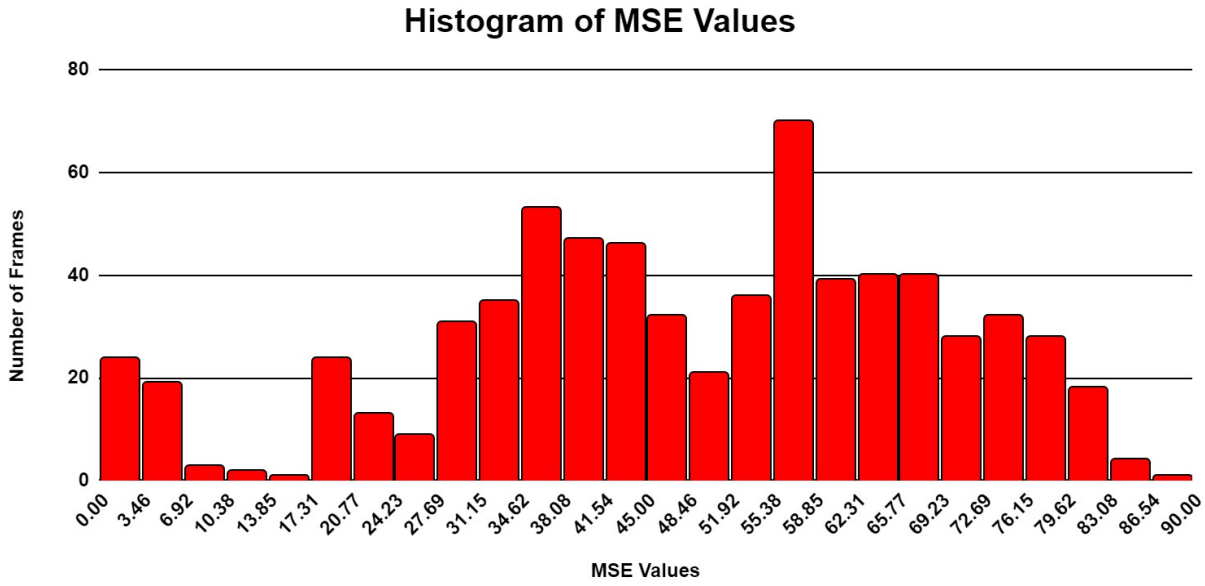
Some of the video frames and their detected foregrounds are shown below

Video Frame	Detected Foreground



Accuracy

The output frames computed by the algorithm were compared with the output frames obtained after using the in-built BackgroundSubtractorMOG2 function in OpenCV2. **The metric used for comparison was MSE**. The frames from my algorithm were converted to black and white only for fair comparison since the in-build function gives output frames in only black and white. The maximum RMSE obtained is **87.11** and minimum RMSE obtained is **1.58**

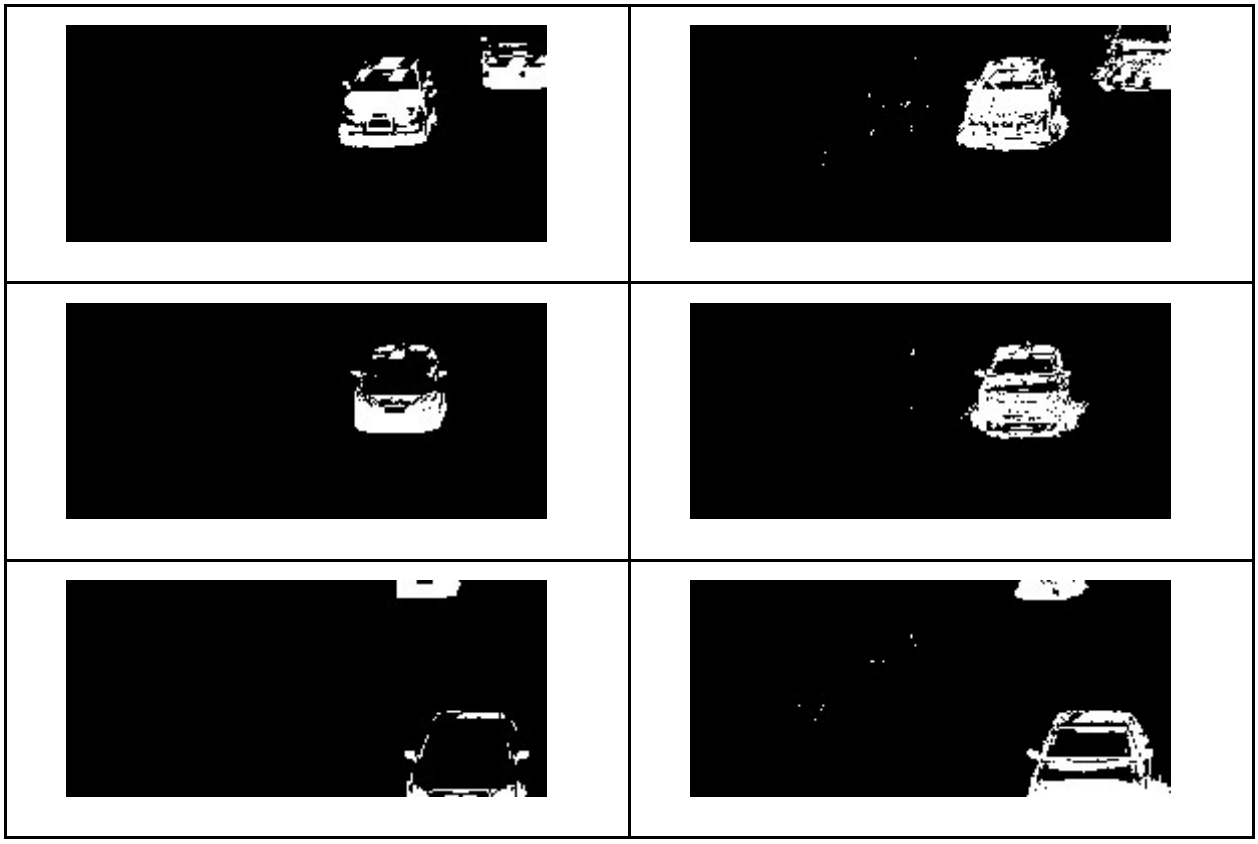


[Link to the spreadsheet containing the MSE values]

Comparison between the two algorithms

- The gaussian mixture algorithm implemented by me had a sharper foreground as compared to the BackgroundSubtractorMOG2 function output.
- However, the in-built function was also able to detect shadows which was not detected by my algorithm.

Output frame using my algorithm	Output frame using in-built function
	
	



References:

- https://towardsdatascience.com/how-to-code-gaussian-mixture-models-from-scratch-in-python-9e7975df5252
- https://ieeexplore.ieee.org/document/784637
- https://hal.archives-ouvertes.fr/hal-00338206/en/
- https://doi.org/10.1016/j.cosrev.2019.100204
- https://medium.com/@prantiksen4/background-extraction-from-videos-using-gaussian-mixture-models-6e11d743f932