Digital Video Processing: CS6870
Assignment 1 : Implementation of Pixel-wise Background Subtraction using GMM

Divya K Raman
EE15B085

7th February 2019

1 Overview

In this assignment, we implement the background (BG) subtraction algorithm using Gaussian Mixture Models as in the paper Stauer and Grimson et. al. The implementation is done in python. We analyse the results on two synthetic(and ideal) videos - Run and Jump and on three real-life vieos - canoe, highway and Pets. In this report, we explain the method, provide snapshots of the output video containing background subtraction results applied on the inputs and analyze various factors that affect the results. The codes as well as the output videos have been zipped and submitted for further reference.

2 Method

Each pixel is modelled as a weighted sum of Gaussians. We use 4 Gaussians for the case of our assignment. Mean, variance and the weights are initialized. The video is then loaded frame by frame. Learning rate alpha is a tunable parameter and has been tuned to be 0.001 for our experiments. The weights, mean and standard deviation are updated as per the update rule given in the paper. Pixels of different videos need to be modelled independently.

The probability of observing the current pixel value is
$P(X) = \Sigma w * \eta(X, \mu, \Sigma)$
where the parameters take the updated values as per the update rule. The summation is done over all the gaussians we use to model each pixel. η is the standard multivariate gaussian function. We take the standard deviation for R, G and B channels to be the same and we take their co variances to be zeroes. This gives us a simple covariance matrix which simplifies calculations.

Update rules are as follows:
$\mathrm{w}_{K,t} = (1 - \alpha)w_{K,t-1} + \alpha M_{K,t}$

where α is the learning rate[2]and $M_{k,t}$is 1 for the model which matched and 0 for the remaining models.

μt= (1ρ)μt1+ρXt
σ2t= (1ρ)σ2t1+$\rho(X_t\mu_t)^T(X_t\mu_t)$
$where\rho = \alpha\eta(X_t|\mu_k, \alpha_k)$

These update rules are applied on each frame of the video and we finally obtain a model for each pixel to predict if it is foreground or background. The first B(tunable parameter) which satisfy a threshold(tunable) are chosen as the background model. The probability threshold to predict if it is foreground or background is set at 0.6 for most of our experiments.

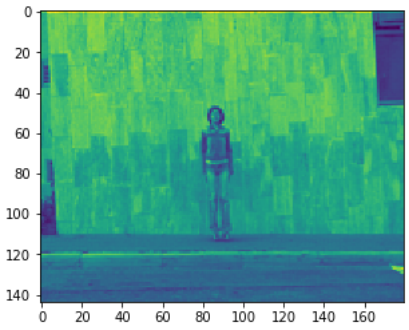
3 Experiments, Analysis and Results

We first code the paper for videos with RGB frames. The code for the same is in the jupyter notebook $\text{BGSub}_RGB.Wehavealsovisualisedaresultforthevideo Jump.avi.Wethencodethepaperf$

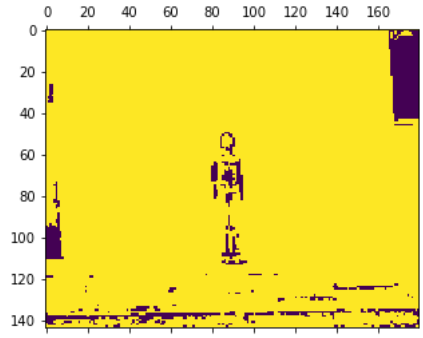
Standard deviation for all the gaussians was initialised to 20 pixels in all the cases. At initialisation time, equal weights were assigned to all the gaussians. The first four frames were used to assign the means of the gaussians in the last three cases. In the first two cases, means were initialized manually by inspecting the pixel values of the background.

3.0.1 Jump video

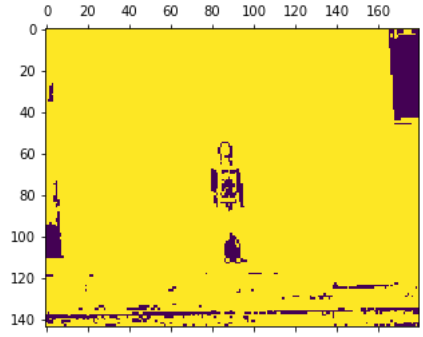
Refer to Jump.ipynb jupyter notebook for more results and details. The model has been run through all the frames of the video. For space constraints, we present the results of only a few frames here. The rest can be obtained from the jupyter notebook.
Jump grayscale frame 1



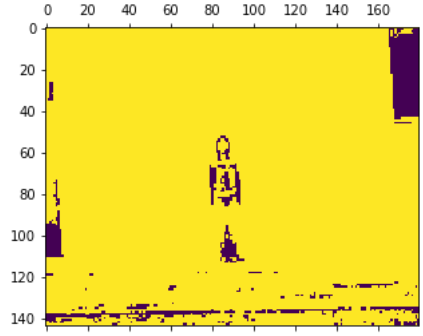
After frame 12



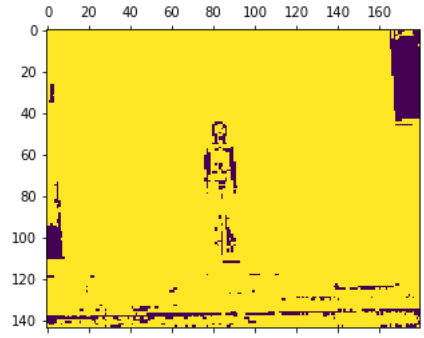
After frame 23



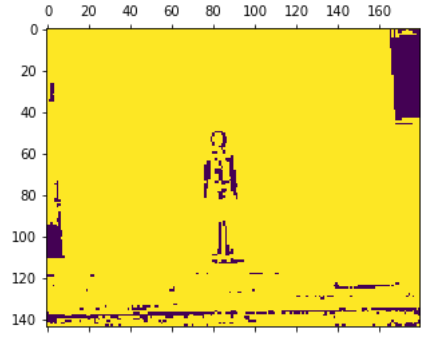
After frame 35



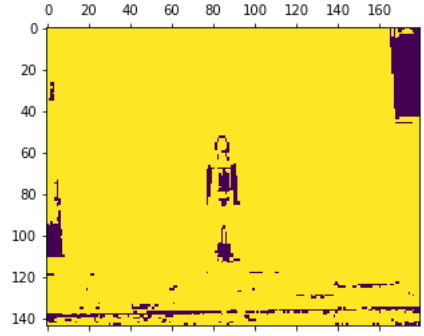
After frame 56



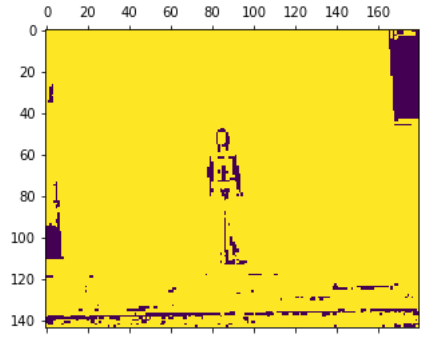
After frame 67



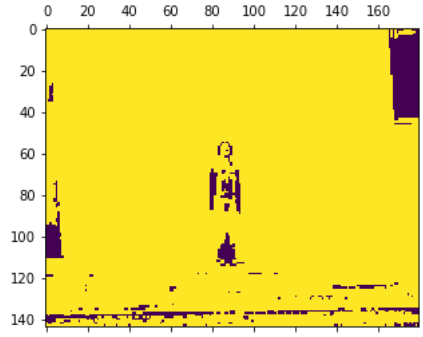
After frame 80



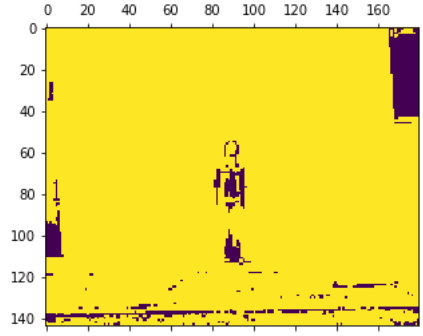
After frame 96



After frame 108

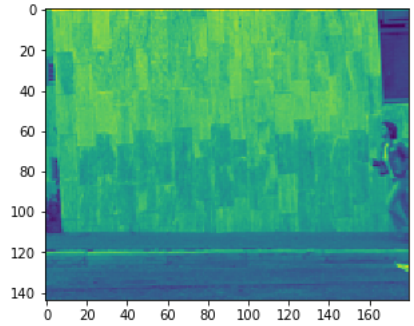


After frame 124

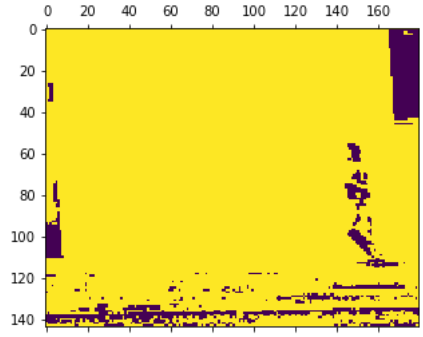


3.0.2 Run video

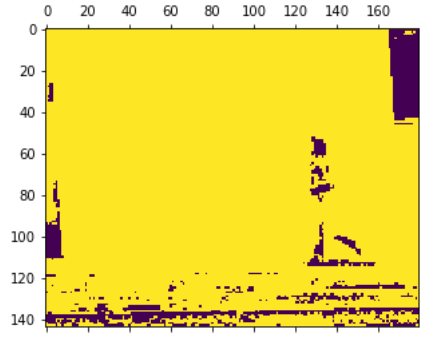
Refer to Run.ipynb jupyter notebook for more results and details. The model has been run through all the frames of the video. For space constraints, we present the results of only a few frames here. The rest can be obtained from the jupyter notebook.
Jump grayscale frame 1



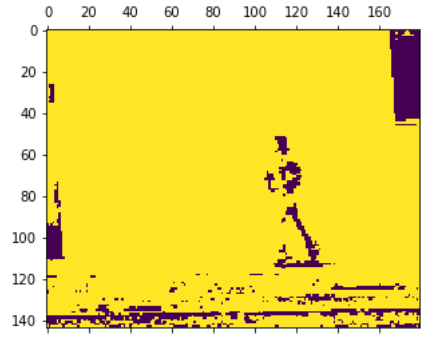
After frame 10



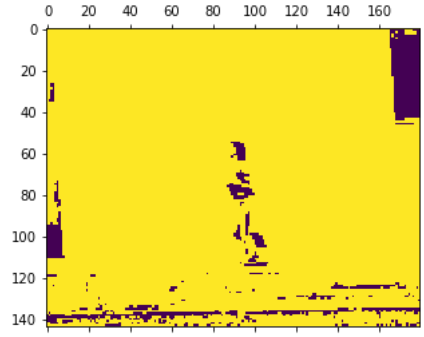
After frame 15



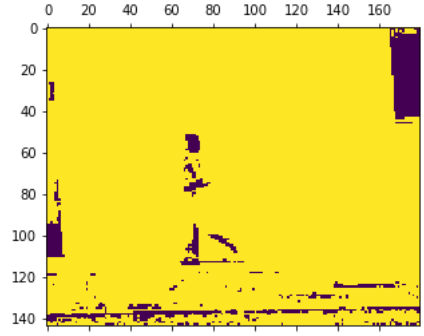
After frame 21



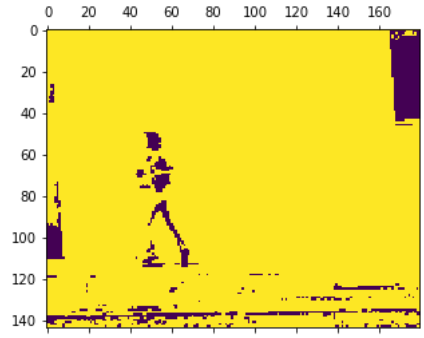
After frame 28



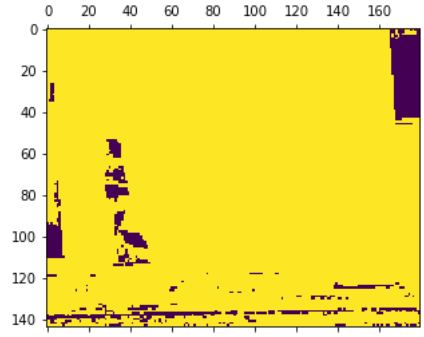
After frame 35



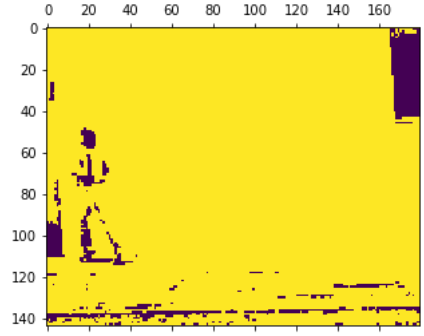
After frame 41



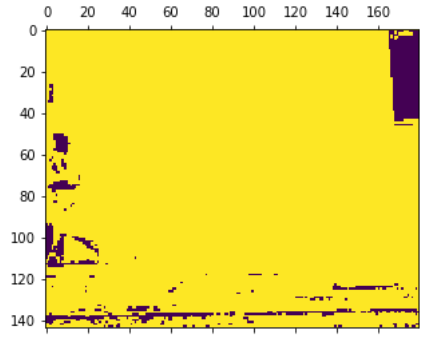
After frame 47



After frame 51



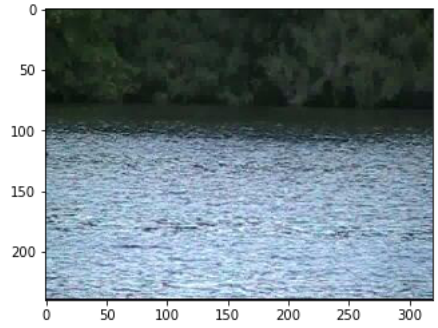
After frame 55



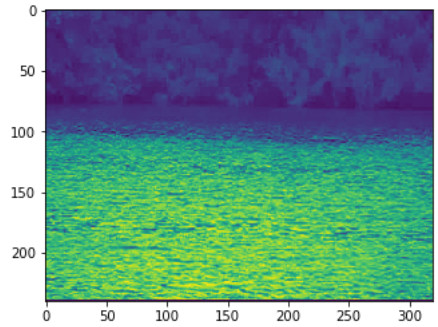
3.0.3 Canoe video

Refer to canoe.ipynb jupyter notebook for more results and details. The model has been run through all the frames of the video. For space constraints, we present the results of only a few frames here. The rest can be obtained from the jupyter notebook.

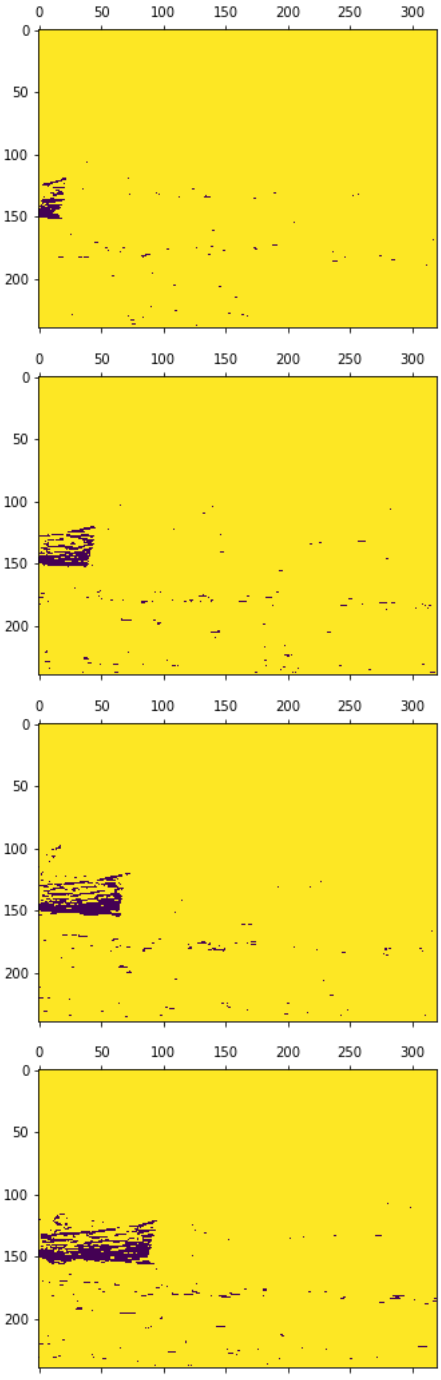
Canoe rgb

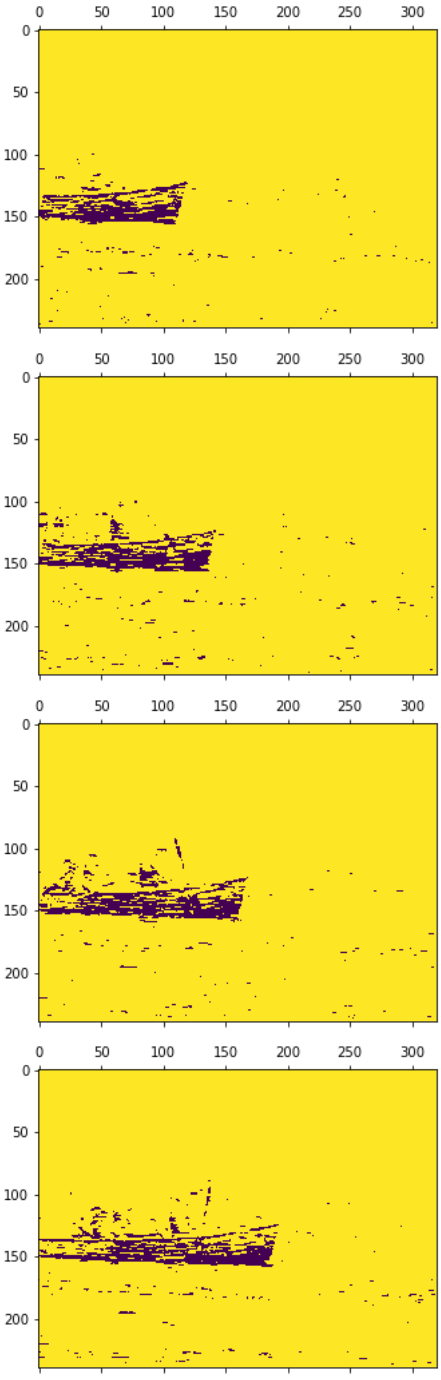


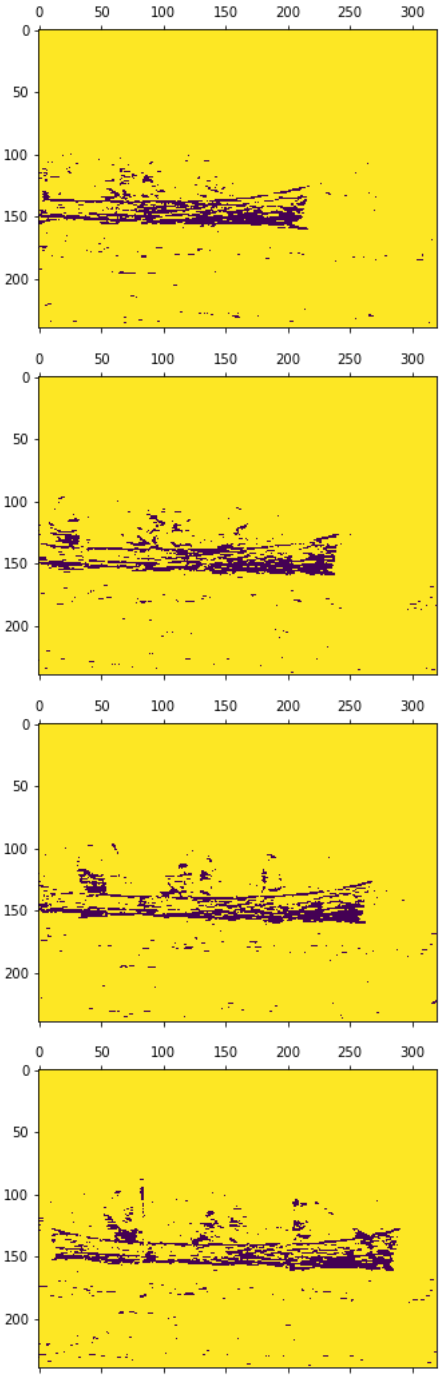
Canoe grayscale

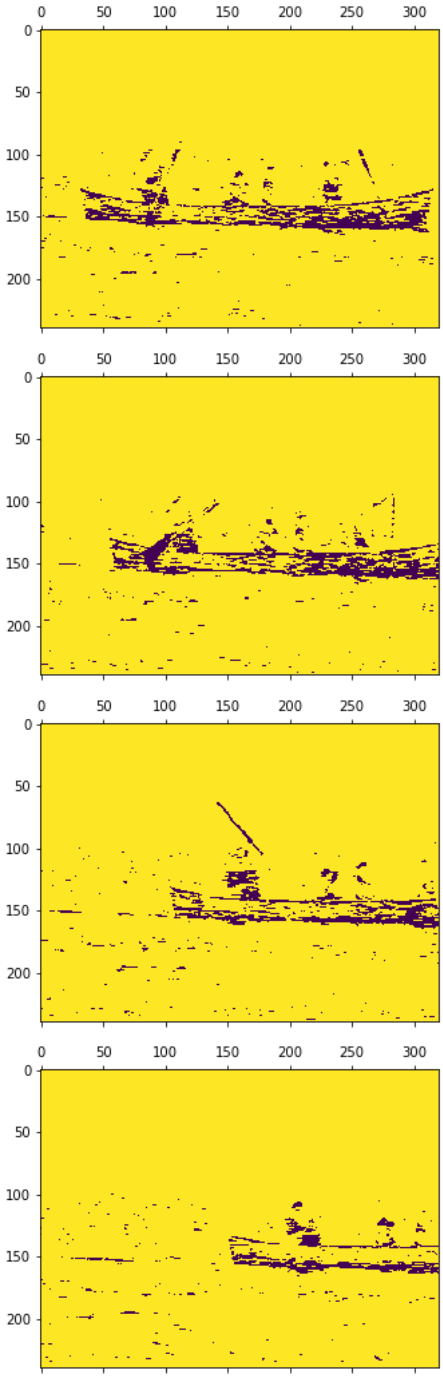


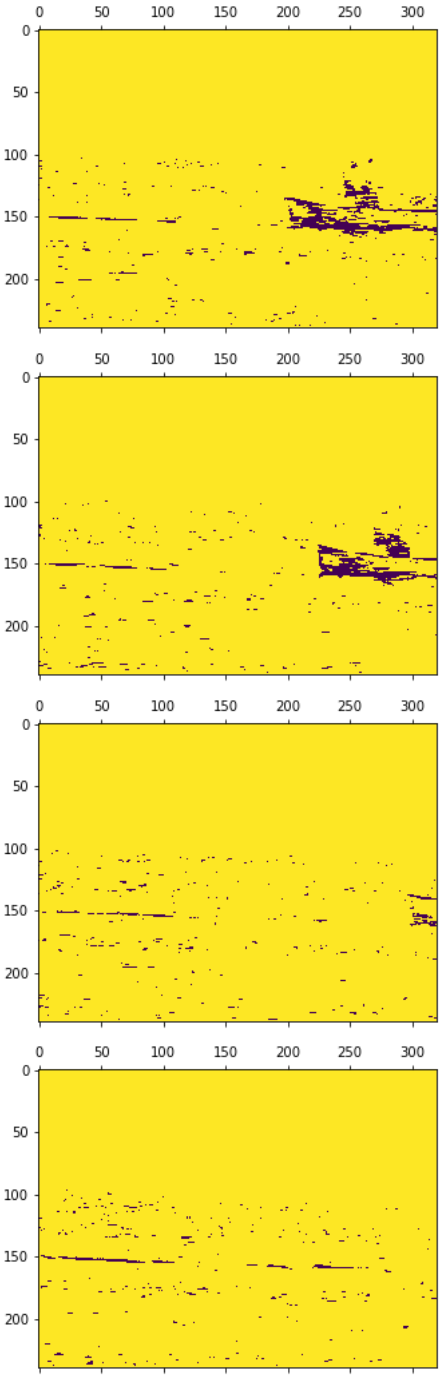
Background subtraction results:


0 50 100 150 200 250 300
0 50 100 150 200


0 50 100 150 200 250 300
0 50 100 150 200


0 50 100 150 200 250 300
0 50 100 150 200


0 50 100 150 200 250 300
0 50 100 150 200
0 50 100 150 200 250 300
0 50 100 150 200
0 50 100 150 200 250 300
0 50 100 150 200
0 50 100 150 200 250 300
0 50 100 150 200

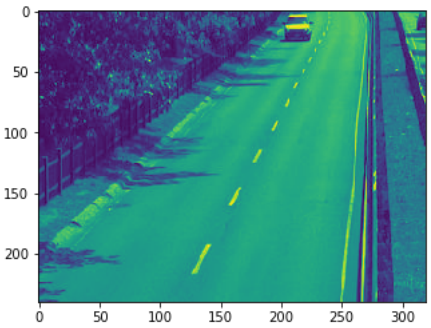

0
50
100
150
200
250
300

3.0.4 Highway video

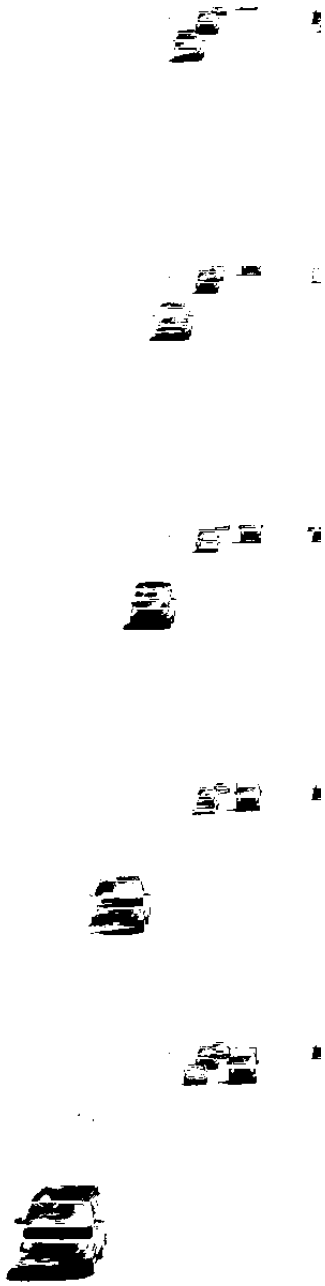
Refer to highway.ipynb jupyter notebook for more results and details. The model has been run through all the frames of the video. For space constraints, we present the results of only a few frames here. The rest can be obtained from the jupyter notebook.

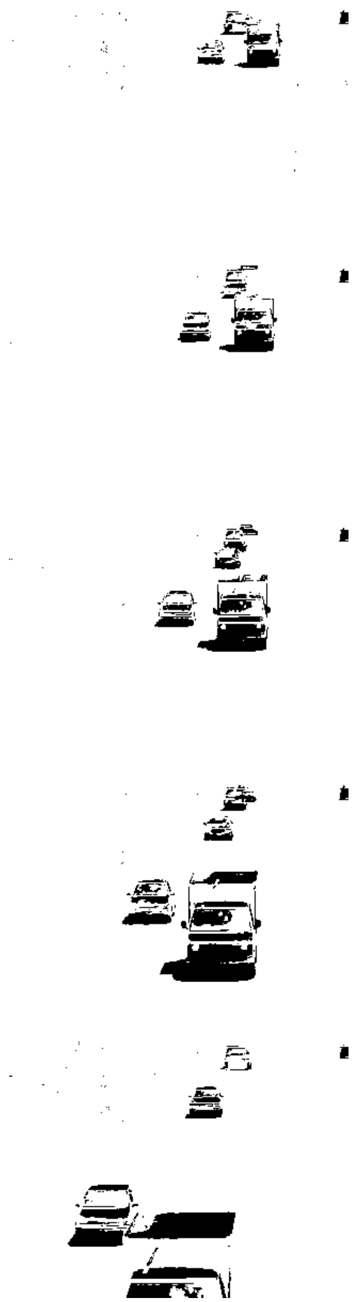
Highway rgb

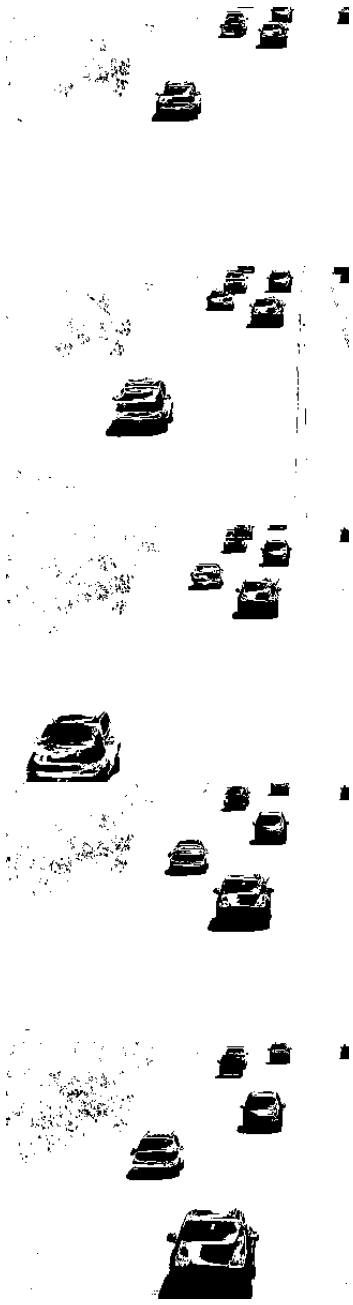
Highway grayscale

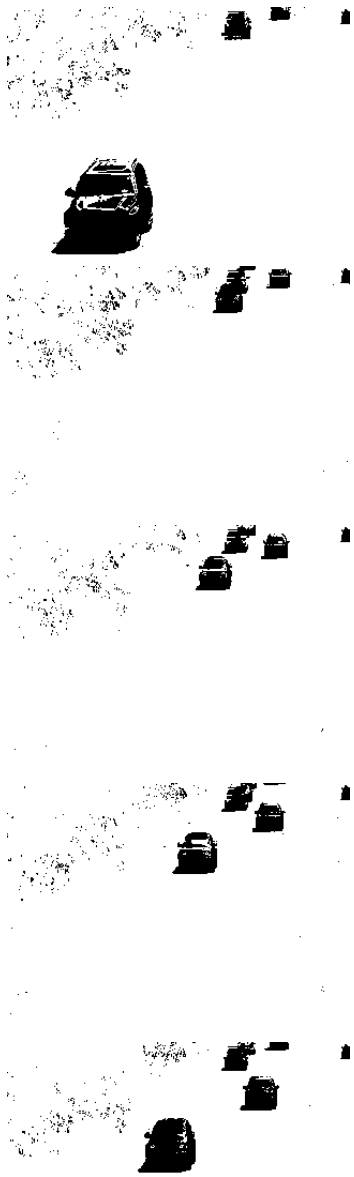


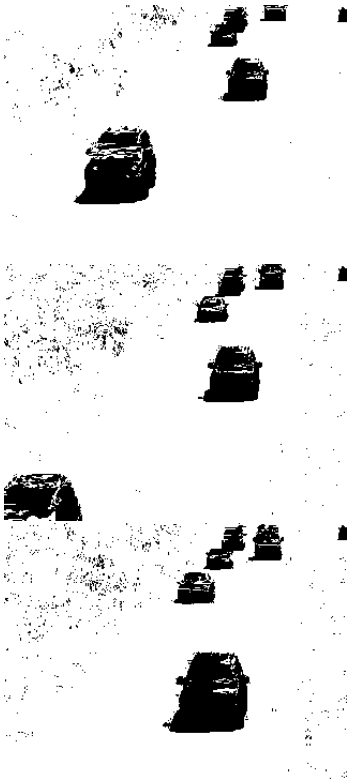
Background subtraction results







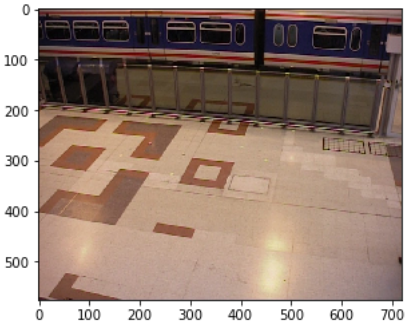




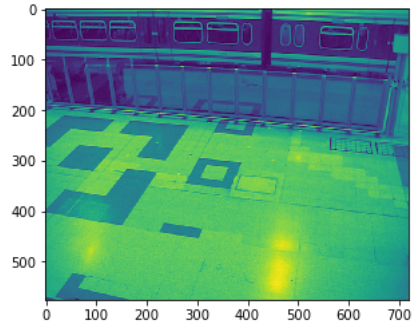
3.0.5 Pets video

Refer to pets.ipynb jupyter notebook for more results and details. The model has been run through all the frames of the video. For space constraints, we present the results of only a few frames here. The rest can be obtained from the jupyter notebook.

Pets rgb



Pets grayscale



Background subtraction results

4 Conclusion

Our results are good for both the ideal videos and the real-life videos. This proves the robustness of the code. Before any processing, pixel values need to be normalized from the 0 to 255 scale to 0 to 1 scale. This makes the code numerically stable. Also, parameters need to be initialized well and tuned carefully.