

# Implementation of Background Subtraction with Kernel-Density Based method with decaying weights

By Rittwick Bhabak (2022MCS2054) (X = 54, so Y = 54%4 = 2)

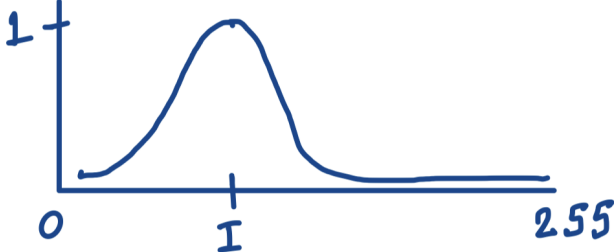
## 1. Algorithm

### Step 1: Initialise the Background Model:

Using the first frame the background model is initialised. For each pixel, (h, w), in the frame the KDE is calculated using the gaussian function.

$$p(x_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(\frac{x_i - x_0}{\sigma}\right)^2}$$

where  $x_i$  are 0 to 255  
and  $x_0 = I$ , which is the intensity of that pixel (h,w)



So, if the frame size is H x W, then as a background model we have an matrix of dimension H x W x 256.

To speed up the process we have divided 256 intensities to 64 bins, where each bin represents 4 consecutive intensities. And 1.5, 5.5, 9.5,...253.5 are the centre of those 64 bins. So, the background model size will be H x W x 64.

Now, such background models are created for red, green and blue channels.

### Step 2: Detecting Background/Foreground pixel and Updating the model

For each frame:

cnt, and  $G_t$  (which are hyperparameters [1]) are updated for this frame, cnt is incremented by one and  $G_t$  is dependent on cnt, so  $G_t$  is updated accordingly.

For each pixel:

#### Background/Foreground Detection:

To detect an pixel as foreground or background the distance  $Dist_d$  is calculated using the following formulae:

$$Dist_d = \min_{\forall k} (C_k^d - x^d), \text{ where } \hat{p}^d(C_k) > 1/N_d \quad d = 1, 2, 3$$

We can obtain the foreground by comparing  $Dist_d$  with  $B_d$  as follows:

$$\left\{ \begin{array}{l} \text{if} \left( \sum_{d=1}^3 (|Dist_d| / (1 + Grad_{t-1,d})) > \sum_{d=1}^3 B_d \times \gamma \right) \\ \text{then, For } G=1, \\ \quad Grad_{t,d} = (G_t - 1) \times Grad_{t-1,d} / G_t + w \times |Dist_d| / G_t \quad d = 1, 2, 3 \\ \text{Else, For } G=0, \\ \quad Grad_{t,d} = (G_t - 1) \times Grad_{t-1,d} / G_t + |Dist_d| / G_t \quad d = 1, 2, 3 \end{array} \right.$$

Where, ‘d’ denotes the colour channel and  $\text{Grad}_{t,d}$  is a hyperparameter which is initialised to 1 for each pixel and each channel and  $N_d$  is the number of bins, here  $N_d$  is 64. [\[1\]](#)

Updating the Model:

$$p_t(c_k) = \text{weight} * \hat{p}_{t-1}(c_k) + \frac{1}{G_t \sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{C_k - \mu_t}{\sigma}\right)^2}$$

this probability is calculated and where

$$\hat{p}_t(c_k) = p_t(c_k) / \sum_{k=0}^{64} p_t(c_k) \quad \text{and} \quad G_t = \text{Gain} \times \frac{2}{1 + e^{-\frac{\text{cnt} - \beta}{\lambda}}}$$

Where weight is 0.95 and the  $C_k$ ’s are the centre of each bins and  $G_t$  is learning rate and Gain, cnt, beta, lambda are hyperparameters [\[1\]](#)

Summary of Algorithm:

- 1. Initialise the background model using the first frame
- 2. For each frame
  - a. For each pixel
    - i. Calculate whether the pixel is a background/foreground pixel
    - ii. Update the background model:  
background\_model =  
current\_probability\_distribution\_of\_pixel + weight \* background\_model

2. Comparison

The comparison is done with the result of the Gaussian Mixture Model with exponentially decaying weights algorithm (Y=1), assignment done by Siddharth S (2022MCS2061).

Results generated by Siddharth S: [link](#)

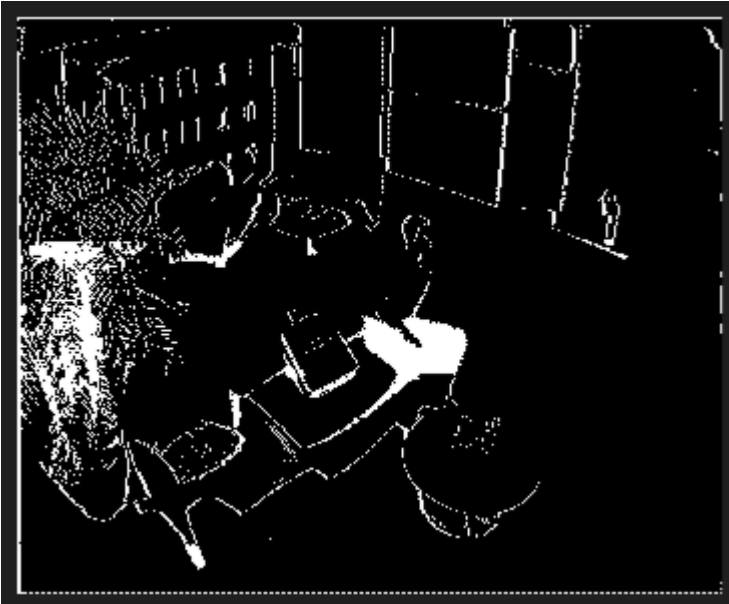
| Test Case      | Accuracy of results using KDE with decaying weights algorithm (implemented my myself) | Accuracy of results using GMM with decaying weights algorithm (implemented my Siddharth S) |
|----------------|---------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|
| Candela_m1.10  | 97%                                                                                   | 88%                                                                                        |
| CAVIAR1        | 97%                                                                                   | 94%                                                                                        |
| HallAndMonitor | 98%                                                                                   | 91%                                                                                        |
| HighwayI       | 89%                                                                                   | 85%                                                                                        |
| IBMtest2       | 96%                                                                                   | 91%                                                                                        |

[Please note, accuracy is taken as (true positive + true negative) / total]  
So, KDE with decaying weights algorithm is working better than that of Gaussian Mixture Models with decaying weights algorithm. The possible reason may be, GMM is trying to assume some

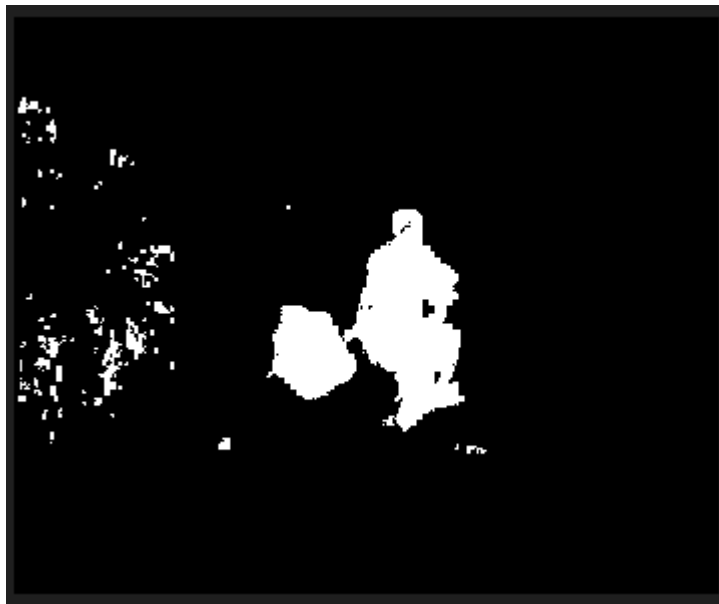
distribution initially and changing it accordingly later but KDE is not assuming any function, rather it is following what input is being given to it.

Failure Cases:

1.



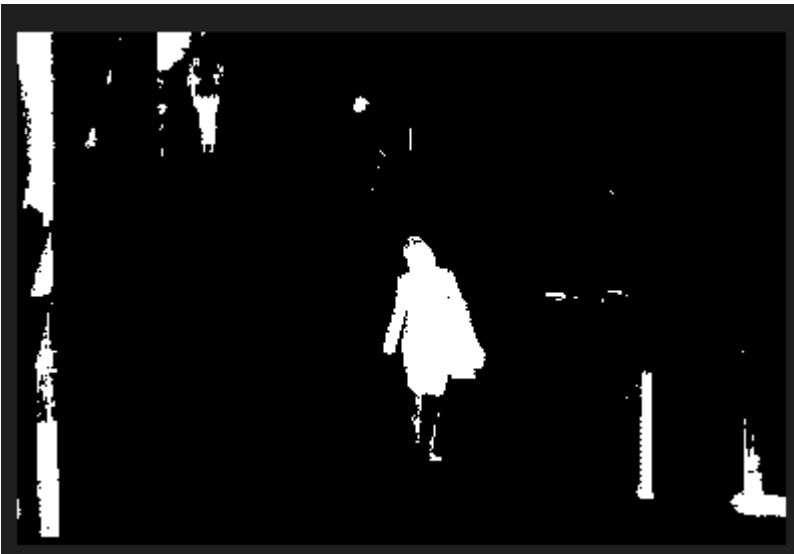
GMM



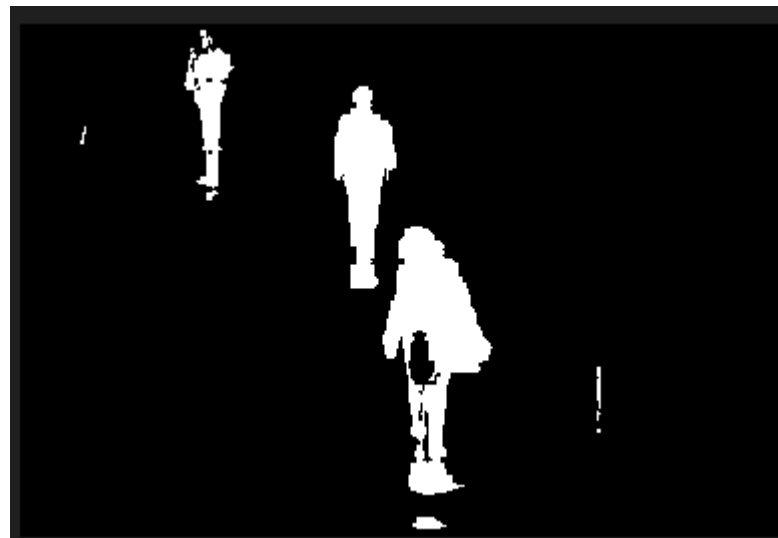
KDE

In GMM the outlines are visible and in KDE the change is correctly detected. Although in KDE the waving of trees is also detected as a change in the frame.

2.



GMM



KDE

In GMM the man is not detected but in KDE the moving man is correctly detected as a change. This happens due to the fast learning rate.

### 3. Image Cleaning using Integral Images

To clean the noise in the result of the algorithm, Integral Image is used.

1. A mask of the image is created i.e. where the pixel is 0, the mask is 0, and where the image pixel is 255, the mask is made 1 there.
2. The mask is divided into 3x3 pixel patches
3. The sum of the patches are calculated using integral images.
4. If the sum is less than 2.25 then all of the pixels of the patch are made foreground in the output image.
5. If the sum is greater than 6.75 then all of the pixels of the patch are made background in the output image.
6. Otherwise the pixels are left as they are.

After image cleaning the result of the algorithm in different test cases are

| Testcase       | Accuracy | Link of the restaurant video<br><a href="#">[The complete directory]</a> |
|----------------|----------|--------------------------------------------------------------------------|
| Candela_m1.10  | 97%      | <a href="#">[Video link]</a>                                             |
| CAVIAR1        | 97%      | <a href="#">[Video link]</a>                                             |
| HallAndMonitor | 98%      | <a href="#">[Video link]</a>                                             |
| HighwayI       | 89%      | <a href="#">[Video link]</a>                                             |
| IBMtest2       | 96%      | <a href="#">[Video link]</a>                                             |

References:

[1] Lee, Jeisung, and Mignon Park. 2012. "An Adaptive Background Subtraction Method Based on Kernel Density Estimation" *Sensors* 12, no. 9: 12279-12300. <https://doi.org/10.3390/s120912279>

[2] Elgammal, A., Harwood, D., Davis, L. (2000). Non-parametric Model for Background Subtraction. In: Vernon, D. (eds) Computer Vision – ECCV 2000. ECCV 2000. Lecture Notes in Computer Science, vol 1843. Springer, Berlin, Heidelberg. [https://doi.org/10.1007/3-540-45053-X\\_48](https://doi.org/10.1007/3-540-45053-X_48)

[3] To generate video from frames the following article is used:  
<https://theailearner.com/2018/10/15/creating-video-from-images-using-opencv-python/>

The way to run the project:  
There must be an "input" directory [The name of the directory is "input", which contains the frames].

