Programming Assignment 2
Gaussian mixture model to achieve real-time visual tracking of a dynamic object present in a given video

Abhavya Chandra
16110001

Overview →

GMM models each pixel as a mixture of Gaussians and use an on-line approximation to update the model. The Gaussian distributions of the adaptive mixture model are evaluated to determine which are most likely to result from a background process. Each pixel is classified based on whether the Gaussian distribution which represents it most effectively is considered part of the background model. All the other pixels are classified as foreground.
The value of each pixel over time is a “pixel process”. Pixel process is a time series of pixel values. Scalar for gray values and vector for color image

- For a particular pixel (x_0, y_0) the random process is

$$\{X_1, ..., X_t\} = \{I(x_0, y_0, i) : 1 \leq i \leq t\}$$

- Each pixel at time t is modeled by a mixture of K Gaussian distributions.

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t})$$

Training →

The algorithm is broken down into two parts:

Modeling process →

K Gaussian distributions are defined as part of GMM. The weights, mean and std are defined as follows →

```
def init_GMM(self):
    """
    initialize the GMM model
    """
    for model_k in range(self.K):
        #see the hardcoded values
        new_model = Gaussian(0, 1)#mean ==0, std==1
        self.Models.append(new_model)

    self.weights = [0.7, 0.11, 0.1, 0.09]
```

For all the normalized pixel (value/vector) in all the frames, the following task was performed →

- Check if the current pixel value belongs to any of the pre-existing Gaussian, based on the threshold value passed (2.5 in my case) →
    - If yes then
        - Mean and std of all the matched Gaussian distributions are updated →

$$\mu_t = (1-\rho)\mu_{t-1} + \rho X_t$$
$$\sigma_t^2 = (1-\rho)\sigma_{t-1}^2 + \rho(X_t-\mu_t)^T(X_t-\mu_t)$$

Where μ and σ are mean and standard deviation, **X** is the current pixel vector ρ is the second learning rate defined as →

$$\rho = \alpha\eta(X_t|\mu_k,\sigma_k)$$

Weight corresponding to all the weights are updated based on the following equation→

$$\omega_{k,t} = (1-\alpha)\omega_{k,t-1} + \alpha(M_{k,t})$$

Where M is 1 if current pixel matches gaussian k else it is taken as 0.

Subtraction process →
After the modeling process, the Gaussians are ordered by the value of **ω/ σ**. And the first B Gaussians are choosen as the background model.

$$B = argmin_b\left(\sum_{k=1}^{b}\omega_k > T\right)$$

Where T is the threshold we apply. ω is the weight and σ is the standard deviation for each of the Gaussian. Based on this model the pixels of each of the frames are further classified as background(black) or foreground(white).

Hyperparameters used →

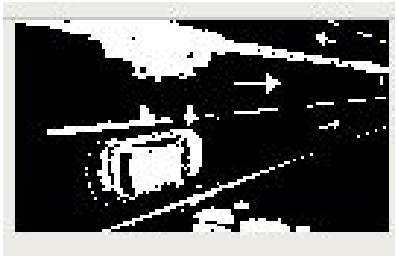
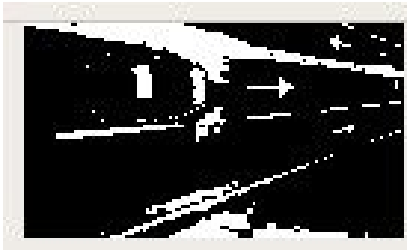
```
#hyper parameters
train_frame_cnt  = 50  [NUMBER OF FRAMES USED TO TRAIN GMM]
K = 4  [NUMBER OF GAUSSIANS USED IN THE GMM]
alpha = 0.05  [LEARNING RATE]
T_b = 0.90  [THRESHOLD TO CLASSIFY GAUSSIAN AS BACKGROUND]
threshVar=2.5 [THRESHOLD TO CHECK BELONGING OF A PIXEL TO A
GAUSSINA]
```

Model API →

```
gmm = GMM(K, alpha, T_b, threshVar)
#train
gmm.perform(PATH, train_frame_cnt, task = "train")
#test
gmm.perform(PATH_test, train_frame_cnt, task="test")
```

Results →

The model was applied on a video that was captured using a camera that was stationary and five cars passed at different time intervals→

All the five cars that pass in the video were perfectly classified as foreground by the model. Since noise was not removed from the frame before applying the model some of the background pixels like the arrow and the divider on the road were classified as foreground. The model performed fairly well in tracking all the four cars when trained on the first 50 frames.

Limitations →

- The model is heavily dependent on the modeled background. If the background is changed or changes rapidly then the model can not work better there.
- A lot of hyperparameters involved in the model that makes it difficult to choose the idea hyperparameters.
- For the algorithm to work the camera is required to be stationary.

Refference →

- https://www.youtube.com/watch?v=g_ve2txPkSc
- http://www.ai.mit.edu/projects/vsam/Publications/stauffer_cvpr98_track.pdf
- https://ieeexplore.ieee.org/abstract/document/784637