COL780 Assignment 1

Adaptive Gaussian Mixture Model for Background Subtraction (Y=0)

Samarth Bhatia
2019CH10124

Results OneDrive

Details/Formulation

I have implemented an adaptive GMM taking as input the number of gaussians to learn per pixel and the learning rate α which also **exponentially decays** the weights. I store the means(μ_k), stds(σ_k), and weights(ω_k) of each gaussian for each pixel.

I assume that the covariance matrix for each gaussian is diagonal, i.e.,

$$\Sigma = \sigma_k^2 I$$

where σ_k is the vector of standard deviations of the 3 channels for the k^{th} gaussian.

For all pixels in the t^{th} frame, the k^{th} gaussian is said to be a match if:

$$|X_t - \mu_k| < 2.5\sigma_k$$

(The first corresponding match is taken in case of multiple matches).

I have implemented all these operations in a vectorized manner so that the computation of each pixel is done simultaneously and we dont need to loop over the total number of pixels.

The update rule for a pixel matching to the k^{th} gaussian is:

$$\begin{aligned}\omega_k &= (1-\alpha)\omega_k + \alpha\mathbb{1}_k \\ \mu_k &= (1-\rho)\mu_k + \rho X \\ \sigma_k &= (1-\rho)\sigma_k + \rho(X-\mu_k)^2 \\ where,& \\ \rho &= \alpha\eta(X,\mu_k,\sigma_k), \\ &\eta\ is\ the\ PDF\ of\ the\ normal\ distribution\end{aligned}$$

The value of $\eta(X, \mu_k, \sigma_k)$ is also called the **likelihood** of the pixel belonging to k^{th} gaussian.

For all pixels that did not find a matching gaussian (new pixel/ foreground), we set the mean of the last gaussian to X and set a high standard deviation.

This process is done for each frame as they arrive, thus the name "adaptive" GMM.
The parameter `alpha` controls how fast the model *reacts* to a new object. A very high value would make it not fluctuate very high and not converge at all, whereas a very low value would result in nothing being learnt from the new frame. It is taken as 0.05 from previous literature and this value works fine here too.
The parameter `num_gaussians` controls how many gaussians to fit. Too less and we will get a very general fit, too many and we will overfit the data and introduce large noise.

Implementation

The python file `gmm.py` contains the class `OnlineGMM` which implements the adaptive/online GMM as described above. I have also made bounding boxes using the `findContours` function in `cv2` and have implemented a vectorized version of Non Maximal Suppression to reject very similar bounding boxes, based on their **IoU**(Intersection over Union) value.
The code to read from the dataset and the driver code is in the `main.py` file.

```
usage: main.py [-h] [--dataset DATASET] [--seed SEED] [--num_gaussians
NUM_GAUSSIANS] [--show SHOW] [--alpha ALPHA]

optional arguments:
  -h, --help            show this help message and exit
  --dataset DATASET     0='Candela_m1.10', 1='CAVIAR1', 2='HallAndMonitor',
3='HighwayI', 4='IBMtest2'
  --seed SEED           Random Seed for initialization of gaussians
  --num_gaussians NUM_GAUSSIANS
                        Number of gaussians
  --show SHOW           0/1/2
  --alpha ALPHA         The exponentially decaying weight alpha
```

I have found the good values of `seeds` and `num_gaussians` and preset them for each dataset, so you only need to use the `dataset` argument.

Problems

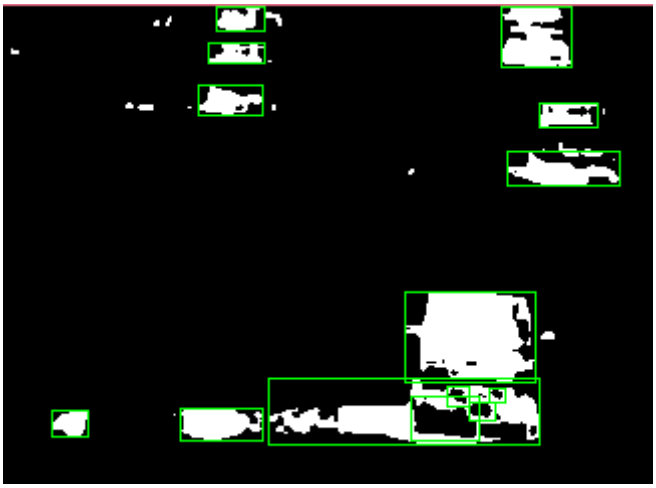
We see from the outputs that at ideal conditions with an easy task (the `HighwayI` dataset), the results are good, except that it also detects shadows, but shadows are part of the ground truth images given.
However, performance degrades severely when there is a complex scene with a lot of change in lighting in the background as well. For example, the `IBMtest` and especially the `Candela` datasets are very noisy, even after using a gaussian kernel to smooth out the image.

Overall, there are small islands of noise detected in the mask. I have tried to overcome that by using various pixel aggregation methods like the closing transform (dilation followed by erosion), by using connected components analysis, and by using a simple area threshold on bounding boxes. However, some noise still remains in the image, partly because of the GMM itself.

Detailed Analysis: `HighwayI` dataset (Success case)

This dataset contains images like this

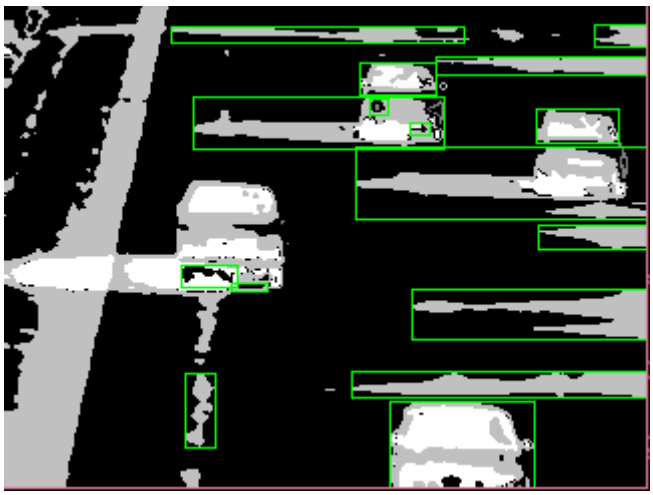
and its prediction from the model: (with `seed=3` and `num_gaussians=3`)



We can see that:

1. It is detecting shadows from all cars. However, raw mask of shadows were very patchy (because of the resemblance to the road) and most shadows are trimmed in the post processing done (i.e., closing transform, connected components method, bbox thresholding and NMS).
2. It is not able to detect the whole car, instead all cars are detected in 2 pieces, being separated at the windshield. I think this is reasonable considering that some of the windows are dark grey and are very similar to the pixels of the road.

We see that on increasing the `num_gaussians`, it is able to detect the cars and their shadows as 2 separate gaussians, and even detects the windshield as a different from gaussian (not BG this time).
With `num_gaussians=5`, we get this:



We can see that:

1. It is now also detecting shadows and windshields and the bumpers of cars. However, there is a lot of noise as these are fast moving foreground objects and each car remains on screen for about ~10 frames, which is enough to know that there is a car, but not enough to learn what is what part of the car.
2. We can see that it is also detecting the *tire trails* on the road, which is undesired as it is also part of the BG.

Most detection problems lie not in the GMM but in the post-processing, as we have not been taught yet what method is ideal in which scenario.

Overall, it does a great job in detecting the foreground of this dataset.

Detailed Analysis: `IBMtest2` dataset (Failure case)

Images from this dataset look like this:

Note that this is an especially challenging scene because:

1. The BG is not consistent in color and has a checkerboard pattern, that too with some regions colored in light brown/dark brown and some colored in light grey/dark grey. This is undesirable as it would not be able to discern that the background changes in color within a local region.
2. The foreground (the person) is also wearing a checkerboarded shirt of similar colors.
3. (Not noticeable from a still) The light being reflected on the wall in the center and that on the front left fluctuates quite a lot and the lighting conditions of the shot change a lot. This

effects all the colors in the scene and it makes the model less accurate about the BG it learned from previous frames.

As expected most of the results are bad:

The model is clearly getting confused in the background tiles itself because of the large variation in color. Also, it is unsure about what part is the person (foreground) too.

One important thing to notice is that the model classifies the walls as foreground as well because the color intensities of the walls change in each frame as well. So, it cannot be sure that it is a background. Similarly, the lighter colored tiles have more variation because of the changing light condition than the darker colored tiles, and so darker colored tiles are being detected as BG whereas lighter ones are FG.

Conclusion

So, we can conclude that just the raw color intensities alone are not a reliable feature of detection. We need more complex features like brightness invariant colors (r = R/R+G+B, g = G/R+G+B) and more complex models (deep models come to mind) to learn these difficult scenes.

After all, GMMs are intrinsically very simple models and operate only on the color information of pixels, and do not take the context of the surroundings into account. For examples, edges and corners are very important features which can help us track objects and very accurately tell what is foreground and what is background. We also notice that in the datasets like `HallAndMonitor`, where the person is wearing shirt and pants of different color, the model cannot infer just from colors of the clothes alone that the shirt and pants "form" a singular human object, and instead it detects them as two foreground objects with different gaussians (again showing the limitation of using only color as features).

Also, there is a lot of visual noise in these scenes, probably because they were recorded quite long ago when cameras were not as great. I was not able to get rid of the jittering noise in most of the indoor datasets using preprocessing methods.