TECHNICAL UNIVERSITY OF MUNICH

Computer Vision Challenge

Report - Group 58

Authors Iheb Belgacem, Hassen Ben Ammar,
Achraf El Euch, Andreas Raupach
Abdelhakim Souit

Jul 2020

Contents

1 User manual for the created program

According to the challenge, our program consists of saving given frames as a matrix, concatenating them into a tensor, and then using segmentation to separate the people in the pictures from the original background by creating a mask and finally using a rendering function to get a result based on which Mode we choose (foreground, substitute, overlay, background).
The program first starts with the pop-up of the GUI which would ask the user for the following information:

- Path for the frames, background picture and where the rendered frames should be saved
- Starting frame
- Rendering mode
- Which camera should be used

Afterwards the function *Image reader* is called which would get the path of every existing frame. N+1 frames would then be concatenated thanks to the function *ir.next()* and be saved into 2 tensors called *left* and *right*.
These frames would then get processed with the *segmentation* function to have masks created explicitly for them.
Afterwards the render function would process them to give us the expected result. Finally the variable *i.start* would increase with N+1 so that in the next iteration of the loop the program would move on to the next frames found in the folder.
When every frame has been processed and the program has exceeded the number of frames found in the folder, the loop will stop. Thereupon the generated frames would start playing and be saved as a video.

2 Segmentation methodology

2.1 Gaussian Mixture Models

To achieve our goal, we decided to use the Gaussian Mixture Model method, which is a very communly used approach for this type of tasks, introduced by Stauffer and Grimson in 1999 [1].
The idea behind the method we used is to fit Gaussian Mixture Models for 600×800 time series, each corresponding to a specific image pixel.
We have as input a certain number of frames corresponding to a scene, taken with a fixed camera.

To begin with, let's consider a pixel (x, y) , $x \in \{1..600\}, y \in \{1..800\}$. The intensity values corresponding to (x, y) at time t are represented by $I_t(x, y)$. For RGB images, $I_t(x, y) \in \{0...255\}^3$, $\forall t \geq 0$, $x \in \{1..600\}$, $y \in \{1..800\}$.
At any time t, we know the previous intensity values corresponding to the pixel (x, y): $\{X_1, ..., X_t\} = \{I_i(x, y); 1 \leq i \leq t\}$.We can now use this time series to estimate the

likelihood of each intensity value for this specific pixel. We model the distribution of intensity values at (x, y) and at time t as a mixure of N Gaussians, using the historical values of intenisty, and we estimate the probability of the new pixel value using this model. The probability of observing the current pixel value is :

$$P(X) = \sum_{I=1}^{K} w_{i,t} * \eta(X, \mu, \Sigma_{i,t}) \tag{1}$$

K = number of distributions
$w_{i,t}$ = an estimate of the weight (what portion of the data is accounted for by this Gaussian) of the i-th Gaussian in the mixture at time t
μ_{it} = is the mean value of the i-th Gaussian in the mixture at time t
$\Sigma_{i,t}$ = the covariance matrix of the ith gaussian in the mixture at time t
η = the probability density function of a Gaussian

The probability density function of a Gaussian can be written as :

$$\eta(X, \mu_{i,t}, \Sigma_{i,t}) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(X-\mu_t)^T \Sigma^{-1} (X-\mu_t)} \tag{2}$$

2.2 Background Model estimation

For each of the 600×800 pixels of the frame at time t, we have a mixure of N Gaussians that captures the distribution of the intensity values at this pixel in the recent frames. The foreground detector from the computer vision toolbox[2], that we have used in this project, uses this modelization to estimate whether the new intensity value at time t should be associated with the background or with the foreground.
Some Gaussians are more likely to belong to the background zone than others. The criterium used to classify the Gaussians is quiet simple, the intensity values corresponding to the background areas are normally more represented in the overall distribution of the intensity values. Furthermore, we expect the intensity values corresponding to the background if the video is taken from a static camera not to vary significantly accross time. These two simple properties will be the basis for the process of classification of the Gaussians.
The Gaussians that simultaneously have a big weight $w_{i,t}$ in the model and a small variance are associated with the background, the other Gaussians represent the foreground.
Once we have identified the Gaussians that describe background elements, we can assign new pixel values to its corresponding zone by calculating its likelihood to belong to one of the background Gaussians.
To illustrate this approach, we will apply it on the first scene from the dataset of the challenge, taken with the right camera. We will study the pixel (500,500), marked by a red star in Fig 1 . For visualization purposes, we transform the images from RBG to grayscale. When we apply the GMM method, we obtain the results shown in fig 2 and fig 3. We have used for this illustration the foreground detector algorithm from the computer vision toolbox. We chose a model with N=5 Gaussians. And in accordance with the

Figure 1: Scene taken with the right camera

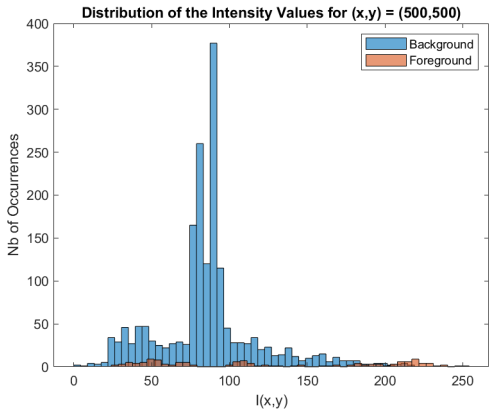

Figure 2: Distribution of the Intensity Values

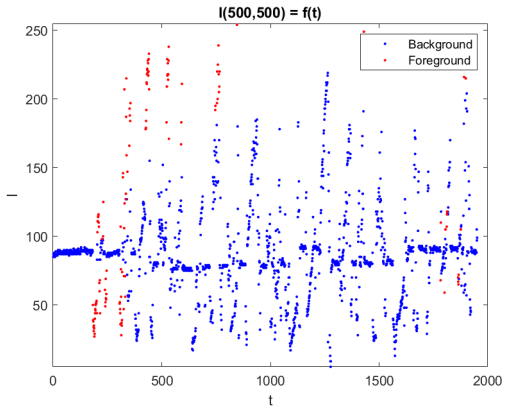

Figure 3: Itensity over time

theory, we found that the distribution of the intensity values classified as corresponding to the background can be modeled as a mixure of two Gaussians with means 40 and 90, and the foreground is modeled as a mixure of three Gaussians with means 50 , 120 and 220. The plots show that the Gaussians associated with the background have large weights and relatively small variances. The weight/variance rate is as expected significantly higher for the background compared to the foreground.

2.3 Hyperparameter tuning

The GMM algorithm has several hyperparameters that affect significantely the quality of the segmentation.

Figure 4: Segmentation with LR = 0.1

Figure 5: Segmentation with LR = 0.01

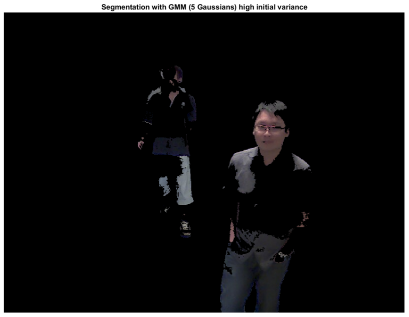

Figure 6: Segmentation with $V_0 = 60 \times 60$

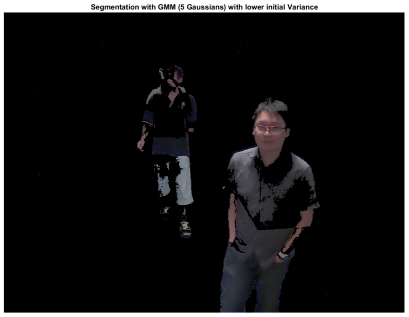

Figure 7: Segmentation with $V_0 = 35 \times 35$

2.3.1 Learning Rate

The learning rate controls how quickly the GMM models adapt to changes in the background. The figures 4 and 5, show that if we choose a too high a learning rate, people who are moving slowly are classified as being part of the background.

2.3.2 Initial Variance

In the Foreground Detector Algorithm, we cannot afford, at each timestep, fitting the Gaussians for all the pixels from scratch. This would be inefficient and computationally very expensive. Instead of that, at each timestep, the algorithm updates the parameters (means and variances) and weights of the Gaussians corresponding to the previous timestep. If the new intensity value is unlikely to correspond to any of the Gaussians from the previous timestep, the Gaussians with the smallest weight is replaced with a Gaussians centered at this new intensity value and with an inital variance V. The initial variance V is a hyperparameter that influences heavily the quality of the segmentation. The figures 6 and 7 show that, if we choose for the initial varience a high value, the intensity values are more likely to be classified as background.

2.3.3 Choice of the hyperparameters

After trying out several combinations of hyperparameter values, we had the best segmentation quality with models with 5 Gaussians, a low learning rate (l = 0.01) and a low

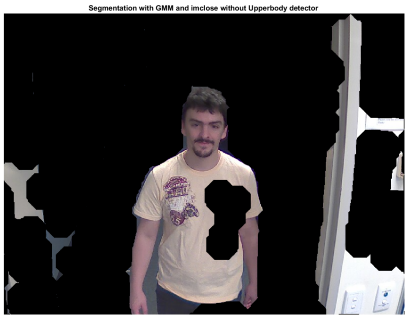

Figure 8: Segmentation without Upper Body Detection

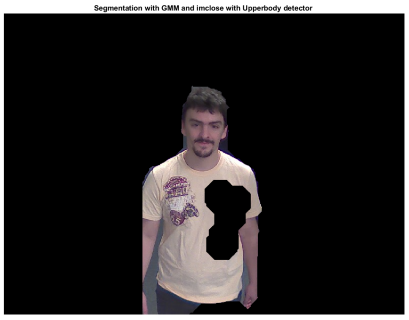

Figure 9: Segmentation with Upper Body Detection

value for the initial variance (35×35). The model with 5 Gaussians manages to capture well the complexity of the different scenes that we have as input.

3 Optimisation of the segmentation method

Here we will explain how we dealt with the biases of the foreground detection algorithm.

3.1 Upper Body detection

To avoid the detection of pixels that are in the foreground but are not our target, like for example the wall in figure 8 , we used the UpperBody detection algorithm [3] from the Computer Vision toolbox of Matlab. We created a mask that extracts the regions desginated by the Upper Body Detector and combined it with the foreground detecter. This approach has improved significatly the quality of the segmentation as in 9.

3.2 Morphological Closing

As we can see in figure 10, below the foreground detector is not detecting the whole body since its working on each pixel separately and does not consider the connection between the pixels in its assumption. To achieve a detection of the whole bodies, it is therefore reasonable to use a function that closes the detected image and gives a homogeneous surface using the connection of the detected pixels. Using imclose function of Matlab [4], we make the inner pixels of the body visible. We used as structuring element a disk with radius 30. We chose this shape to preserve the circular nature of the body. The imclose function performs dilation followed with an erosion using the same structuring element. The use of imclose has significantly improved the segmentation quality. This can be seen in the figure 11.

4 The Bonus task

To achieve our goal of having also the option of a video in the background, we added a function called *nextvideo* that would transform any video it receives as input it into an

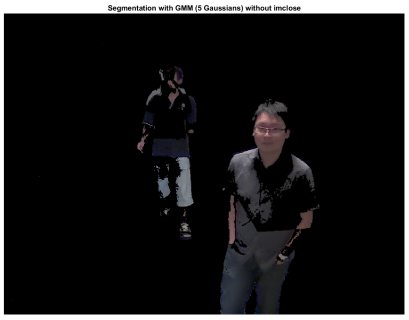

Figure 10: Segmentation without Imclose

Figure 11: Segmentation with Imclose

Image tensor. The function would then compare the size of the video with the number of frames we got in the folder. If the size of the video is smaller than the number of those frames then it would loop it a number of time until it exceeds the number of the frames. When inputing the backgrounds path, its last characters are used to determine whether the file is a video or an image. So the program automatically knows how to act depending on that.

5 Results and Outlook

In the figure 12, we can see an example of the results that we obtained

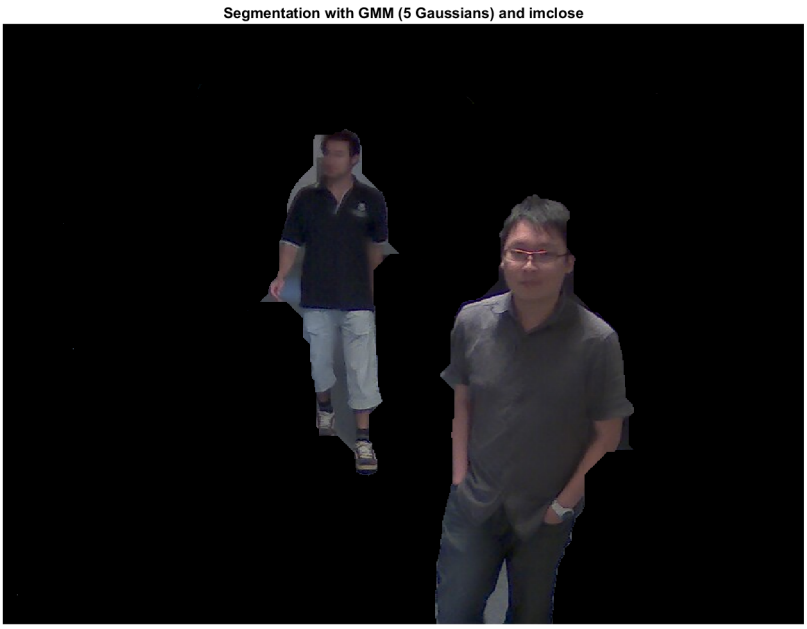

Figure 12: Segmentation result for a scene with two persons

We can see that it is still containing false negatives (Pixels from the background classified as foreground). Those could be explained by the fact that the brightness of the background

is changing more than the predefined threshold, so that the too dark pixels are seen as a new detected object. This happens especially when the object is too close to the camera, as we can see in the figure 13 where this problem is exaggeratedly happening.

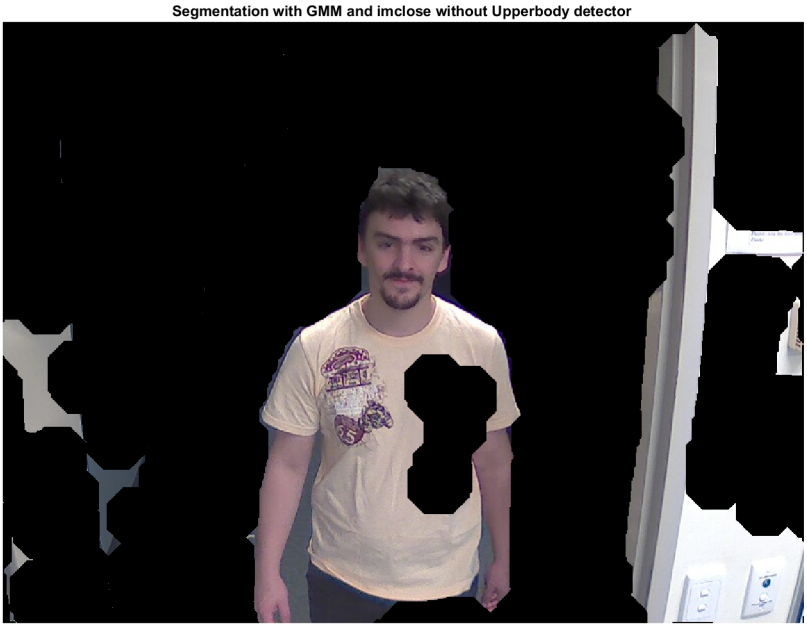

Figure 13: Segmentation result for a scene with a person too close to the camera

This problem could be possibly partly solved by a normalization of the brightness of the picture. To achieve this, it is maybe required to have a reference spot or pixel to compare with and then regulate the brightness of the picture depending on this reference pixel. An example of this reference pixel could be a spot where it is the less possible that a new object could be detected , for example in the upper right corner of the picture in this case.But since our program is supposed to work for different scenes, it is difficult to generalize the predefinition of this reference point.

Bibliography

[1] Stauffer Chris Grimson W.E.L , *Adaptive background mixture models for real-time tracking.*, Proceedings of IEEE Conf. Computer Vision Patt. Recog, vol. 2. 2.

[2] MathWorks *Foreground detection using Gaussian mixture models*, https://www.mathworks.com/help/vision/ref/vision.foregrounddetector-system-object.html

[3] MathWorks *Morphologically close image*, https://www.mathworks.com/help/images/ref/imclose.html

[4] MathWorks *Train a Cascade Object Detector*, https://www.mathworks.com/help/vision/ug/train-a-cascade-object-detector.html

[5] Soellinger Dominik *Adaptive foreground-background segmentation using GMMs*, https://github.com/dsoellinger/Adaptive-foreground-background-segmentation-using-GMMs