EE243: Advanced Computer Vision Assignment #3

Ali Jahanshahi (862029266)

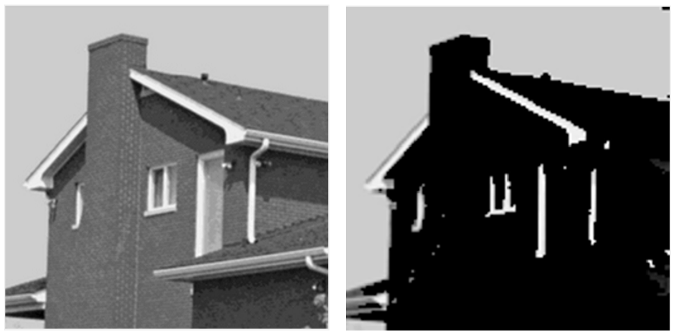
May 2019

1 Problem 1

In this question we are asked to implement and apply a basic version of normalized cuts for segmenting the image on $house.tif$ and $peppers\_color.tif$ images. In [1], the basic version of normalized cuts algorithm for image segmentation is stated as follows:

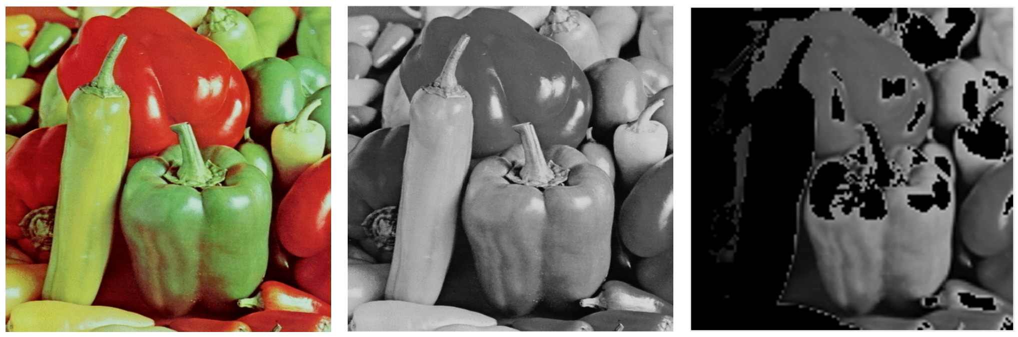
1. Given an image or image sequence, set up a weighted graph $G(V, E)$ and set the weight on the edge connecting two nodes to be a measure of the similarity between the two node.
2. Solve $(\mathbf{D} - \mathbf{W})\mathbf{x} = \lambda \mathbf{D} x$ for eigenvectors with the smallest eigenvalues.
3. Use the eigenvector with the second smallest eigenvalue to bi-partitioning the graph.
4. Decide if the current partition should be subdivided and recursively re-partition the segmented parts if necessary.

We implemented the algorithm and applied normalized cuts on the images. Since solving the eigenvalue problem for all eigenvectors is of a high complexity, we re-sized images from 512×512 to 128×128, performed normalized cuts, and finally re-sized them to their original size. Figure 1 and Figure 2 show the result of applying normalized cuts on $house.tif$ and $peppers\_color.tif$ images, respectively.



(a) Original image. (b) Segmented image.

Figure 1: Result of applying normalized cuts algorithm on *House* image.



(a) Original image. (b) Gray scale image. (c) Segmented image.

Figure 2: Result of applying normalized cuts algorithm on *pepper* image.

Comments: As we can see visually, for $house.tif$ image, the segmentation is more accurate than that of $peppers\_color.tif$ image. The reason behind the accuracy difference is that pixel's brightness plays an important role in the output of the algorithm. Since $house.tif$ image is more smooth from a brightness perspective, the segmentation results tend to be more accurate. On the other hand, $peppers\_color.tif$ image has more brightness levels ans since the algorithm is performs segmentation using bi-partitioning, it maps pixels with brightness above a certain threshold to the same class. In addition, since $peppers\_color.tif$ image is converted to gray scale, it had lost some of its information that is useful in segmentation. In addition to smoothness and brightness, normalized cut algorithm does not work well with objects that do not have well defined boundaries, as we can see in our test case images that have different boundaries.

2 Problem 2

In this problem, we are asked to implement the Expectation Maximization algorithm for mixture of Gaussian model based on color features for segmenting images. It is asked to apply the algorithm on $peppers\_color.tif$ image,, which is shown in Figure 3.

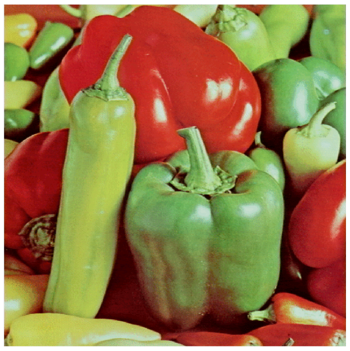


Figure 3: $peppers\_color.tif$ image.

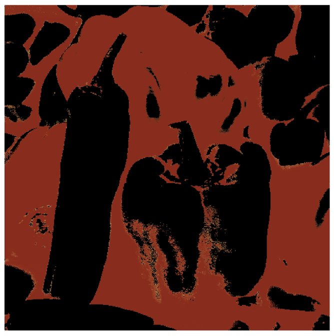
The Expectation-Maximization (EM) algorithm is an iterative way to find maximum-likelihood estimates for model parameters when the data is incomplete or has some missing data points or has some hidden variables. EM chooses some random values for the missing data points and estimates a new set of data. These new values are then recursively used to estimate a better first data, by filling up missing points, until the values get fixed. It is mainly consists of two phases:

1. **E-step** or expectation step: Calculate the estimate of the hidden data from the observed data and and current parameter estimate.

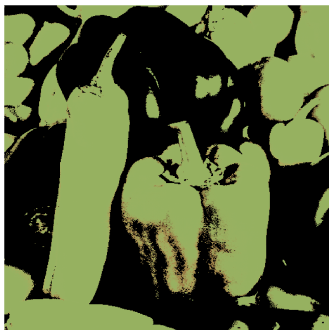
2. **M-step** or maximization step: Calculate the maximum likelihood parameters for the current estimate of the complete data.

EM algorithm iterates between E-step and M-step and converges to maximum likelihood parameter estimate for observed data under certain conditions.

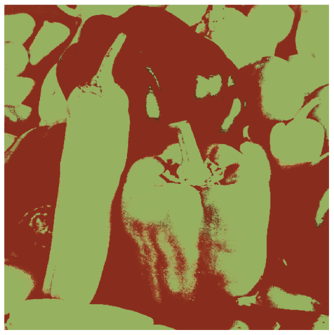
We implemented EM in Matlab and applied on $peppers\_color.tif$ image. We applied the algorithm with 2, 3, and 5 Gaussian mixtures that resulting segments are illustrated in Figure 4, Figure 5, are Figure 6, respectively. Figure 4c, Figure 5d, and Figure 6f show the combination of all the algorithm output segments for their corresponding Gaussian mixtures.



(a) First segment.

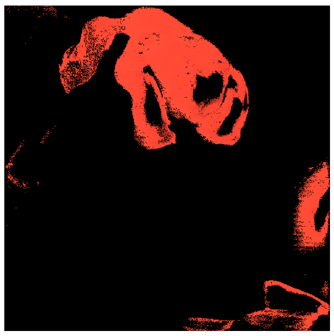


(b) Second segment.

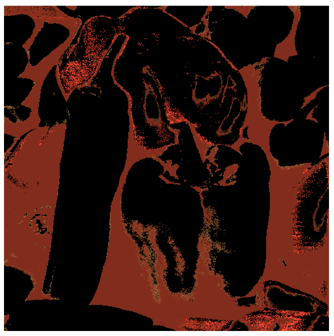


(c) Combined segments.

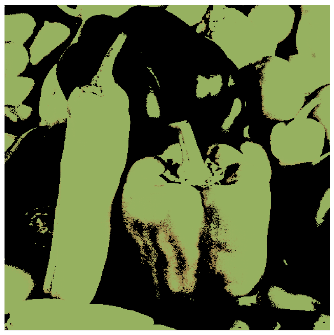
Figure 4: Result of applying EM algorithm on $peppers\_color.tif$ image with 2 Gaussian mixtures.



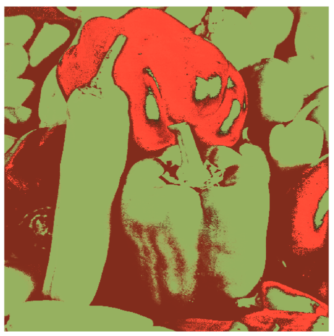
(a) First segment.



(b) Second segment.

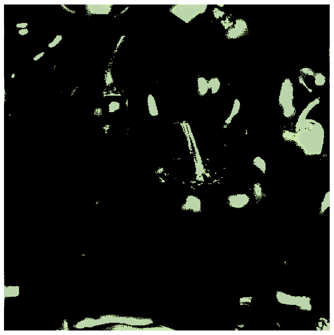


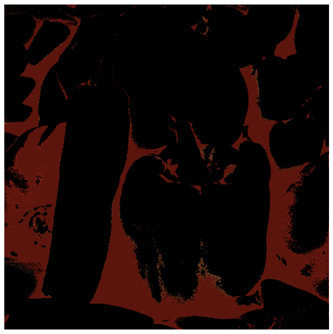
(c) Third segment.

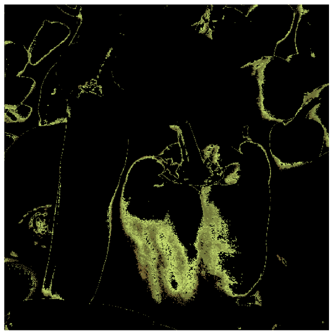


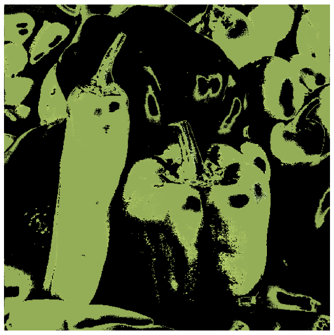
(d) Combined segments.

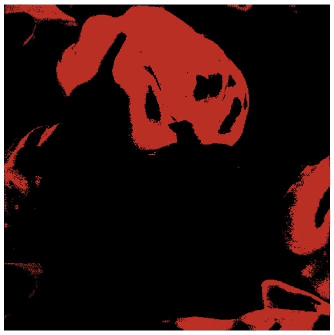
Figure 5: Result of applying EM algorithm on $peppers\_color.tif$ image with 3 Gaussian mixtures.


(a) First segment.


(b) Second segment.


(c) Third segment.


(d) Fourth segment.


(e) Fifth segment.

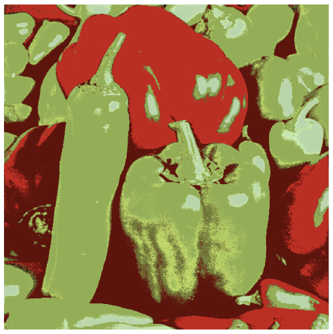
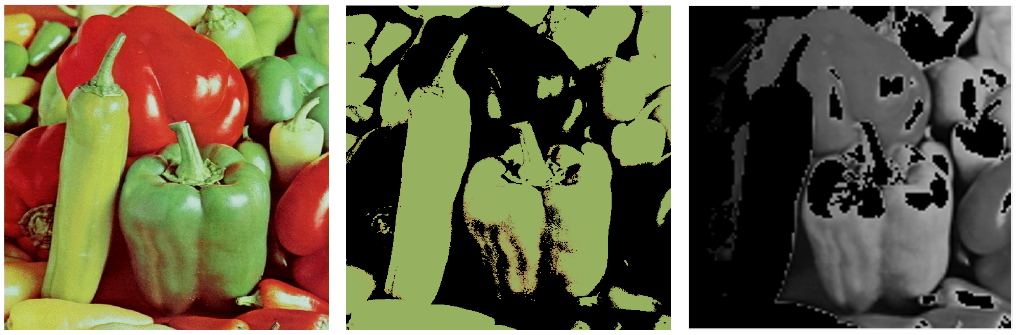

(f) Combined segments.

Figure 6: Result of applying EM algorithm on $peppers\_color.tif$ image with 5 Gaussian mixtures.

Comments: Gaussian mixture model (GMM) addresses segmentation problem by a $clustering$ approach. As we can see in the implementation, GMM is fairly simple and efficient. Due to its nature, GMM is able to model complex distributions. However, on the other hand, we need to know number of image components in advance.

As we can see visually, by increasing the number of Gaussian mixtures, the accuracy of the segmentation increases, which completely makes sense. In order to compare the accuracy of EM for GMM algorithm with that of the other approach in Problem 1, we put their results next to each other in Figure 7. Figure 7b shows the output of applying $NormalizedCuts$ (NCut) which segments the image to two parts. In order to be fair, we illustrated EM-GMM output with 2 Gaussian mixtures in Figure 7c.



(a) Original image. (b) EM-GMM. (c) NCut.

Figure 7: Comparing the result of applying EM-GMM with 2 Gaussian mixtures and NCut on $peppers\_color.tif$ image.

As we can see in Figure 7, the objects in the left part of the image is segmented more accurate by applying EM-GMM than NCut. As mentioned in Problem 1 analysis part, the reason that NCut has a lower accuracy might be because of losing color information after converting the image to gray scale.

3 Problem 3

For this problem, we are asked to detect the people, extract features for the detected regions and then apply data association to obtain correspondence between persons from frame t to $t+1$. $dataAssociationTracking.m$ is provided that calls the following functions, which we need to implement:

1. It first loads the video and extracts the frames from it. For each frame, it calls the following functions.

2. As the video is recorded from a static camera, a simple sum of difference between N frames will be able to highlight the moving objects. $getSumOfDiff.m$ function is called for this purpose with $N = 3$. Given a stack of N frames, it returns the following:

$$D(m,n) = \frac{1}{\binom{N}{2}} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N-1} |I_i(m,n) - I_j(m,n)| \tag{1}$$

where I_i represent the i^{th} image in the stack. This function should be implemented.

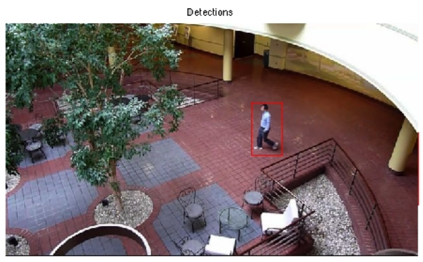
3. The sum of difference image is then used by the $getDetections.m$ function to obtain the detections. This function should segment the blobs which are highlighted in the sum of difference image. We need to implement this function. It returns the bounding box details [$topleft\_x$, $topleft\_y$, $width$, $height$] of the segmented blobs.

4. The features are extracted from the bounding box regions using the $getFeatures.m$ function. We used the HoG features function.

5. All the above steps are repeated for the next frame. Then, the $getMatches.m$ code is used to obtain the correspondences between the detected regions in the two frames.

6. Finally, the detections and the matched bounding box centroids are displayed in the started code.

After completing all the above required functions, we observed the detections and data associations as shown in Figure 8, Figure 9, and Figure 10 that show frames in which one, two, three persons are detected. Figure 11 shows a situation in which the object (person) is mostly covered with a tree but the algorithm has detected that.



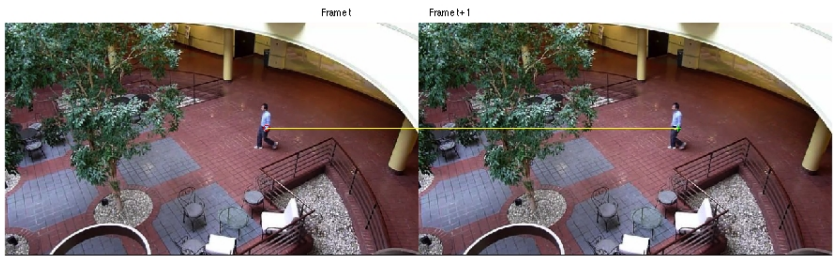


Figure 8: Detecting one person in the video.

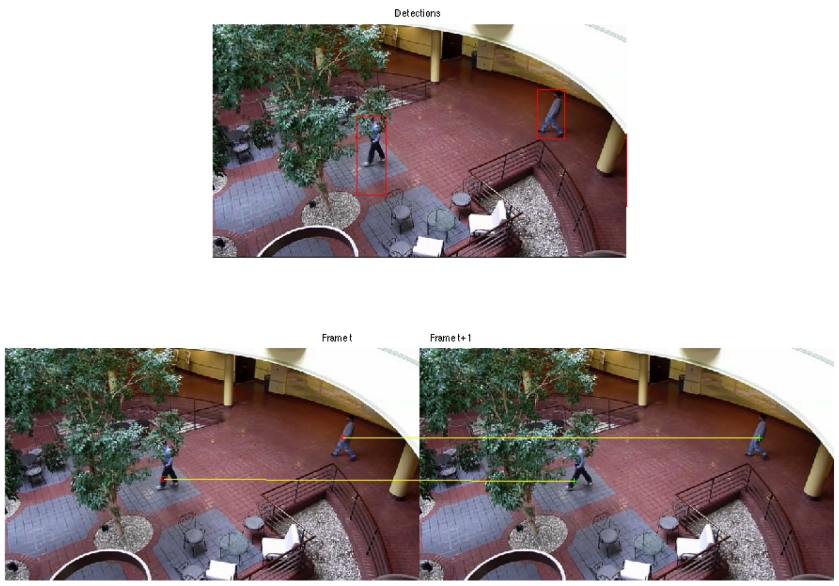


Figure 9: Detecting two persons in the video.

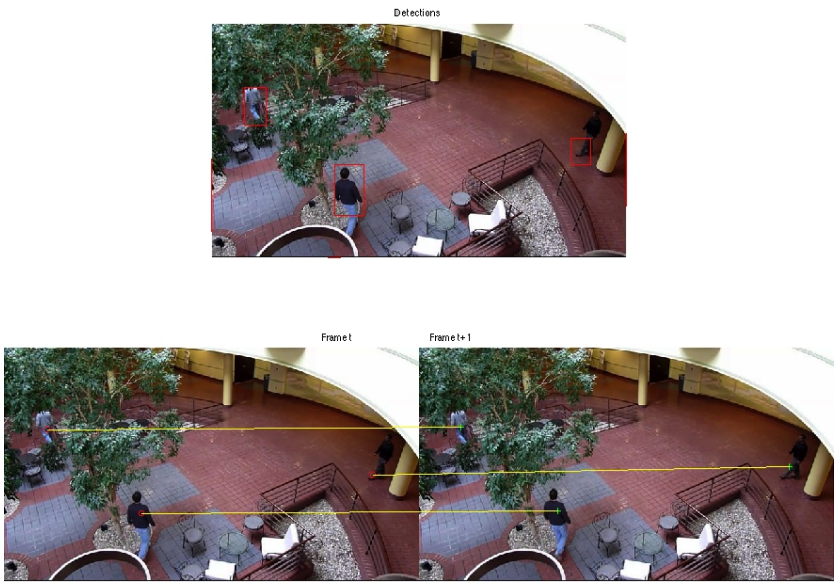


Figure 10: Detecting three persons in the video.

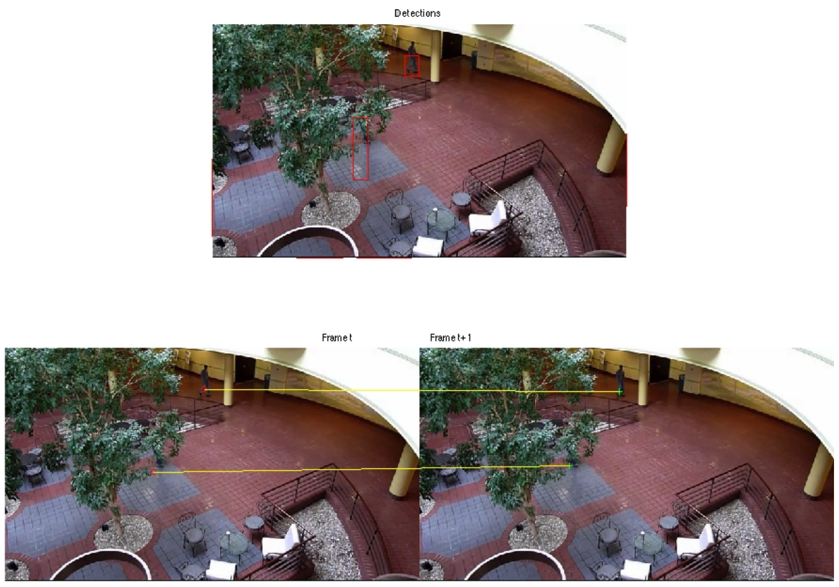


Figure 11: Detection when the object is mostly covered by a tree.

References

[1] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Departmental Papers (CIS)*, page 107, 2000.