CSE152a – Computer Vision – Assignment 2 WI14
Instructor: Prof. David Kriegman.
Revision 1

Instructions:

- This assignment should be solved, *and written up* in groups of 2. Work alone *only* if you can not find a partner. No groups of 3 are allowed.
- Submit your assignment electronically by email to obeijbom@cs.ucsd.edu with the subject line *CSE152 Assignment 2*. The email should have two files attached.

  1. A pdf file with your writeup. This should have all code attached in the appendix. Name this file: CSE_152_hw2_writeup_lastname1_lastname2.pdf.
  2. A compressed archive with all your MATLAB code files. Name this file: CSE_152_hw2_code_lastname1_lastname2.zip.

  The code is thus attached *both* as text in the writeup appendix and as m-files in the compressed archive.
- Please make this a proper report, with methods, thoughts, comments and discussions. All code should be tucked away in an appendix. Include name and student ID in your report and always use figure legends.
- No physical hand-in for this assignment.
- You may do problems on pen an paper, just scan and include in the writeup pdf file.
- In general, MATLAB code does not have to be efficient. Focus on clarity, correctness and function here, and we can worry about speed in another course.
- The following matlab commands are useful for formatting your plots nicely

  1. subplot.m allows your to grid your figures
  2. set(gcf, 'Color', [111]) makes the figure background white.
  3. set(0,'defaultlinelinewidth',2) increases line thickness.

1 Binarization [10 points]

Write a MATLAB function to implement the 'peakiness' detection algorithm described in class (`http://cseweb.ucsd.edu/classes/wi14/cse152-a/lec7.pdf`). This algorithm should automatically determine an intensity level to threshold an image to segment out the foreground from the background. The output of your function should be a binary image that is 0 for all background pixels and 1 for all foreground pixels. Apply this function to the image *can.jpg* and turn in the output image in your report.

Notes:

- Load in an image as a grayscale, double-precision image, e.g.
  `img = im2double(rgb2gray(imread('can.jpg')));`
- You can use the MATLAB function `hist(img(:),numBins)` to create a histogram of pixel intensities as a first step to the peakiness detection algorithm. Using `numBins=15` and a minimum distance of 3 bins between peaks should work.

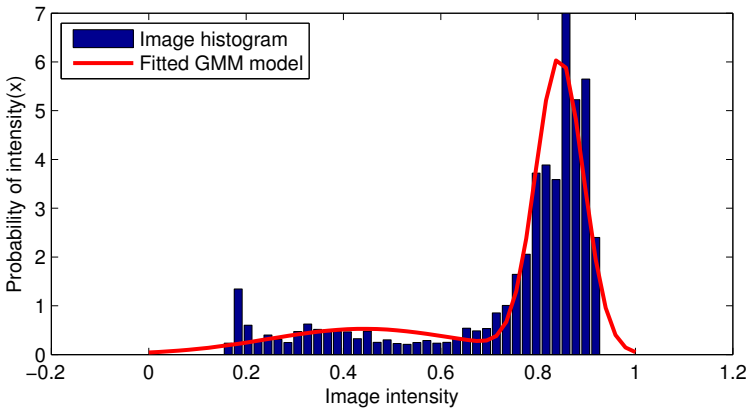

Figure 1: Fitted Gaussian Mixture Model to a sample image histogram

1.1 BONUS: Gaussian Mixture Models [5 points]

Instead of the peakiness algorithm, try fitting a Gaussian Mixture Model to the grayscale image. Let one of the modes represent the foreground and the other the background. You can use the MATLAB function gmdistribution.fit to fit the model to the data. Include the image histogram along with the fitted distribution overlaid, as shown in fig. 1. Note that it will suffice to use 2 mixture models. Once you have fitted your distribution, a segmentation can be achieved by

$$B(x,y) = \arg\max_c P(I(x,y)|c)P(c) \tag{1}$$

where $P(I(x,y)|c)$ is the probability of intensity at pixel (x,y) for class $c \in 0,1$, and $P(c)$ is the (prior) probability of class c, and $B(x,y)$ is the output binary image. Compare and contrast your results using this method to the peakiness method above.

In addition to the class slides, some more details on gaussian mixture models can be seen at Professor Vasconcelo's slides (`http://www.svcl.ucsd.edu/courses/ece271A/handouts/EM.pdf`), which also contain some information about the EM-algorithm used by gmdistribution.fit to fit the model.

2 Connected Components [15 points]

(a) Write MATLAB code to implement the connected component labeling algorithm discussed in class, assuming 8-connectedness. Your function should take as input a binary image (computed using your algorithm from question 1) and output a 2D matrix of the same size where each connected region is marked with a distinct positive number (e.g. 1, 2, 3). On the image *can.jpg*, display an image of detected connected components where connected region is mapped to a distinct color (using the function imagesc in MATLAB).

(b) How many components do you think are in the image *coins.png*? What does your connected component algorithm give as output for this image? Include your output on this image in your report.

Notes:

- To avoid MATLAB recursion limits, you may find it necessary to increase the recursion limit: set(0,'RecursionLimit',1000);

Figure 2: Sample images

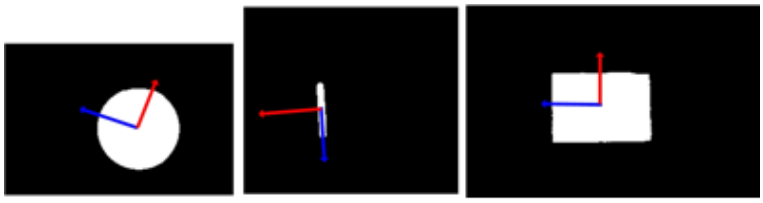

Figure 3: Sample images with principle directions overlaid

- For the coin image, it may help to reduce the size of the image before running the connected components algorithm but after converting the image to a binary image:
  binarySmall = imresize(binary, 0.25, 'bilinear');

3 Take your own images [5 points]

For the next parts, you will be using images you take with your own camera. Choose three objects of different shapes, preferably with non-shiny surfaces. Possibilities include paper/cardboard cut-outs, bottle tops, pencils, Lego pieces, etc. Take 3 pictures of these objects individually with a solid background. The object should be clearly distinguishable from the background (e.g. bright object & dark background). Include these three images in your report as in Figure 2. In addition, show similar output images as in Problem 2 for each of your new images (there is only 1 connected component in this case).

4 Image moments and rectification [10 points]

Write three functions which compute the moments, central moments, and normalized moments of a marked region (`http://cseweb.ucsd.edu/classes/wi14/cse152-a/lec7.pdf`). Each function should take as input a 2D matrix (output of part 2) and 3 numbers j, k, and d. The output should be the (i, k) moment $M_{j,k}$, central moment $\mu_{j,k}$, normalized moment $m_{j,k}$ of the region marked with positive number d in the input matrix.

Using these functions, on each of the three images, draw the centroid of each object. Also compute the eigenvectors of the centralized second moment matrix (`http://cseweb.ucsd.edu/classes/wi14/cse152-a/lec7.pdf`) and draw the two eigenvectors on the centroid. This should indicate the orientation of each object, see Figure 3 for the results on the example images. Turn in the outputs for your own images.

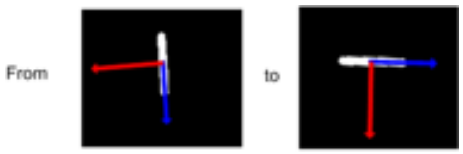

Figure 4: Aligning one of the sample images

5 Image alignment [10 points]

We have seen that the orientation computed from Problem 4 can be used to roughly align the orientation of the region (i.e. in-plane rotation). Write a function to rotate the region around its centroid so that the eigenvector corresponding to the largest eigenvalue (i.e. the blue vector in Figure 3) will be horizontal (aligned with $[1, 0]^T$). This might look like in Figure 4.

Your function should take as input a 2D matrix (output of Problem 2) and output a matrix of the same size in which all marked regions are aligned. Turn in the aligned outputs for your images.

Note: After finding the rotation matrix R to rotate the largest eigenvector to $[1, 0]^T$, we rotate all points (x, y) belonging to that region using the following transformation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = R \begin{bmatrix} x - \hat{x} \\ y - \hat{y} \end{bmatrix} + \begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix}$$

where $[\hat{x}, \hat{y}]^T$ are the centroid coordinates of the region For simplicity, just ignore the cases when part of the aligned region falls outside of the image border or is overlapped with other regions. You can avoid these issues when capturing your images (e.g. put your objects a bit far apart). Finally, note that the rotation matrix can be created trivially from the eigenvectors.

6 Full recognition pipeline [20 points]

Now you will put all the pieces together. Take an additional picture with all objects present in the image (e.g Figure 5).

Do the following:

- Use your segmentation routine (Problem 1) to produce a binary image.
- Use your connected components routine (Problem 2) to produce an image of connected regions (should be three). Display your result for this step.
- Use your alignment routine (Problem 4) to produce an image of aligned regions. Display your result.
- Extract a feature vector fi of the moments $(\mu_{2,0}, \mu_{0,2}, m_{2,0}, m_{0,2})^T$ for each aligned regions. Label each region in the 4th image by comparing its feature vector to the feature vectors extracted from the aligned objects in the first three images (images with individual objects). Choose the label for which the Euclidean distance between feature vectors is minimized: $\arg\min_i ||z - x_i||_2^2$, where x_i is the feature vector for object i, and z is the feature vector for the 'unknown' object that you want to classify.

Note: Use MATLAB text function to label the regions with a text at their centroids (e.g. Circle, Stick). Depending on the shape of your object, you may not get perfect results. Though it is not required, you can play with different set of moments to see which moments are most discriminative for your objects.

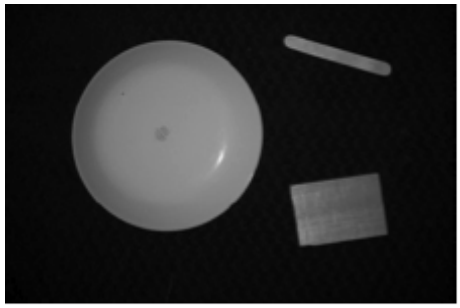

Figure 5: Picture of all objects together