Computer Vision HW-NN See Dropbox for due date.

Segmentation Using Mahalanobis Distance

1. The usual rubric applies. Comments are important. If you are provided with code that has minimal comments, you should add to those comments to show understanding.

2. Put all of your files into ONE DIRECTORY named HWNN_Lastname_Firstname_DIR

3. For easy grading, write one driver function that runs your entire homework.
   a. The function is to be in a file named HWNN_Lastname_Firstname_MAIN.m
   b. This function takes no parameters, but runs three subroutines:
      i) HWNN_Email1_Cluster_Colors('Img431_SCAN275__Secret_Mess.jpg')
      ii) HWNN_Email1_FIND_RASPBERRIES('Raspberry_Bush_image.jpg');
      iii) HWNN_Email1_FIND_ORAGES('Orange_Tree_Image_3305.jpg');

   Copy all produced images into your final write-up.
   Describe in your write-up, how you achieved the color segmentation.

4. HWNN_Email1_Cluster_Colors (' Img_Some__Secret_Mess.jpg '); (3 pts)
   This reads in any one of the secret messages, and using k-Means, segments them into different colors.

   Render each color separation that your program finds as white, on a black background.
   Use kmeans to find the cluster centers, and then minimum ~~Mahalanobis~~ distance to find which cluster each pixel belongs in.
   Describe in your write-up, how you achieved this separation. What did you do, what issues did you face?

   Update: the Mahalanobis distance is no longer an option for Matlab's kmeans( ) routine.
   I have found that using the Squared Euclidean distance works fine. Minimizing the squared Euclidean distance is the same as minimizing the Euclidean distance, because squaring is a monotonic function – it does not change the relative order.

   You will find that using the Squared Euclidean distances works especially well in a color space that is nearly perceptually uniform. (*hint*) **Do not be surprised if kmeans takes a long time to run on a large image. It will run faster on a sub-sampled smaller image. You might want to test your technique on a smaller image.**

   For increased speed, it is possible to sub-sample the image significantly, then find the cluster centers for kmeans, and then use the cluster centers from the sub-sampled image as seed points for the original larger image. This makes the final run much faster since it is starting with centers that are nearly the final cluster centers.

   Remember: for kmeans, you need to tell it ahead of time how many clusters to create. You might want to create an extra "catch all" cluster for noise points.

   I have found that telling the code to generate 8 clusters for 6 colored pens seems to work well. Your results will be different.

   If you want to cheat, it is perfectly legal for you to pre-seed kMeans with the initial seeds that you think might be good starting points. For example, black ink would be [0, 0, 0] and white paper would be [1,1,1] in RGB color space. In CIELab (lab) color space, the colors are different. For example, the purple pen might be [45, 35, -45], or something, and green might be [50, -40, 10]. See the documentation for kmeans on how to pre-seed the clusters with initial "centers" even if the colors are not the centers of the clusters.

   Using End Members: You can seed clusters with "end members." End members are idealized points that are far from other colors. For example, the color [40, 0, -45] in CIELab color space might not be the color of the blue ink that was used. However, this might be a great starting point for blue because it is far from the other colors used, and it will shift towards the true color of the blue ink.

   If you seed your clusters with a cluster center, describe how you seeded them, and how you decided on your seeds. AND, in your conclusion point out that you did this.

(continued)

5. HWNN_names_FIND_RASPBERRIES(‘Img431_Example__Raspberry_Image.jpg‘); (3 pts)

   This reads in one image.
   Based on a model that you have written for classifying pixels as raspberries (based on the color),
   this classifies the pixels in the image as raspberry (1) or not-raspberry (0).

   Design and use a Mahalanobis Distance Classifier.

   You will develop this classifier by collecting pixels on raspberries from the provided images. The test images will be very similar. (Use ginput() to click on pixels.)

   Your program may change the color space of the image to any other color space it wants to use.
   It then uses a pre-built classifier to determine which pixels are raspberry pixels, and returns 1 if the pixel is a raspberry and it returns a 0 if the pixel is not.

   The key here is that you need to create a cluster center in some color space. Then if the pixel is close enough to that color, classify it as being belonging to that class. For example, if the pixel is close enough to some red center, then it is a raspberry. You will have to decide what distance works well.

   In your write-up for this question, you should describe what distance you used, what color space you used, and how you made these design decisions.

   It should create a uint8 image, with white pixels where there are raspberries in the original image.

6. HWNN_your_names_FIND_ORANGES(‘Orange_Tree_Image_3305.jpg’) (3 pts)
   Repeat the process of separating the fruit from the background, but instead of raspberries, use one of the images of oranges provided.

   For this code, you will need a different cluster center. Oranges are not red.

7. Conclusion, write-up your overall learning here. (1 pt)

   Give an overview, describe any problems you ran into, and give strong evidence of what you both learning.

   If you seeded your kmeans clusters, describe that you did that.

Do not re-submit the original images. The graders will have their own copy.
However, in your write-up, show your resulting images.