CS4640 A<7>

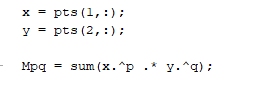
<Yingjie Lian>

November 8th , 2019

**Question 1 Answer:**

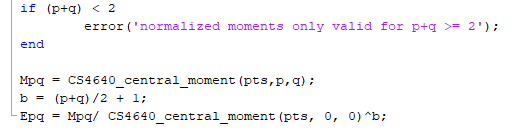
For CS4640\_central\_moment(pts,p,q), I have followed the equation Mpq = sum sum (xˆp\*yˆq).

First, I take out x and y from points. Then, I have applied to the equation.

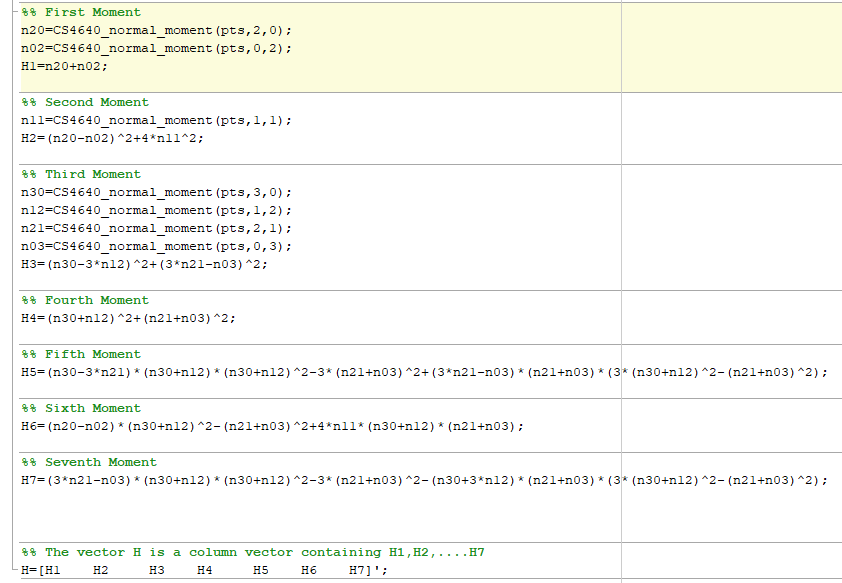


For CS4640\_normal\_moment(pts,p,q), I have followed the equation Epq = Mpq/M00ˆb where b = 1+(p+q)/2.

There is a trick, p+q must be greater or equal to 2.



For CS4640\_Hu\_moments(pts), I have followed the equation to A1, A2, ...A7.

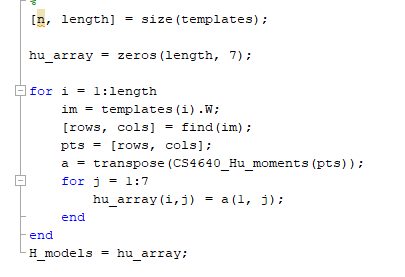
Then, I have created an 7x1 array to store them.

For H\_models = CS4640\_Hu\_build(templates).

Since the input is a n element struct. I first take out its length, so I can use it in the for loop later.

Then, I have created a nx7 array, so I can store the hu moments value for each template image.

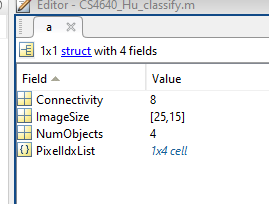
Then, I have used the for loop twice to put each value in the array I have created.



For classes = CS4640\_Hu\_classify(im,H\_models).

First, I have used bwconncomp() function to see how many connected components the image has.

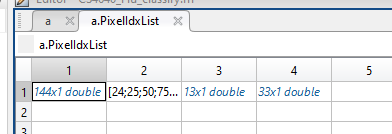
When you use it, it will generate information below:



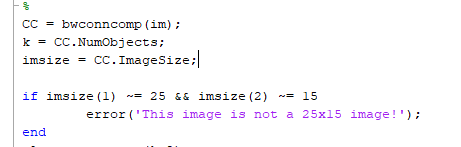
The connectivity is defaulted as 8. It is the same when we use bwlabel() as 8-connected objects.

ImageSize is mxn size image. This example is a 25x15 image.

NumObjects is the number of how many connected components this image has.

PixelIdxList is the 4 different connected components points cell. Here is that:

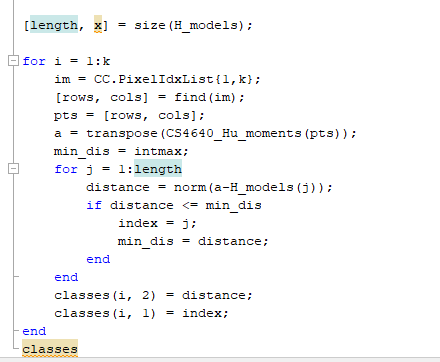
To look it, we can simply call a.PixelIdxList(i). i is the index of the points.



Then I have checked if the image is a 25x15 image.

Then, I have generated a kx2 array. k is the number of how many connected components the input image has.

Since the truth.mat has all 25x15 image, I used this as my templates.



I have used two for loops to get the distance and index of char.

First, I computed the Hu moments for each connected component.

Then, inside the second for loop, I used norm() to calculate the Euclidean distance.

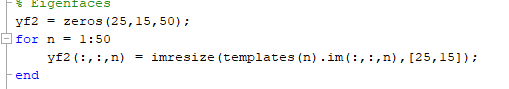
Then, I did a comparison in order to keep the smallest distance and its index.

Finally, I got the smallest distance and its index, put them into my classes array.

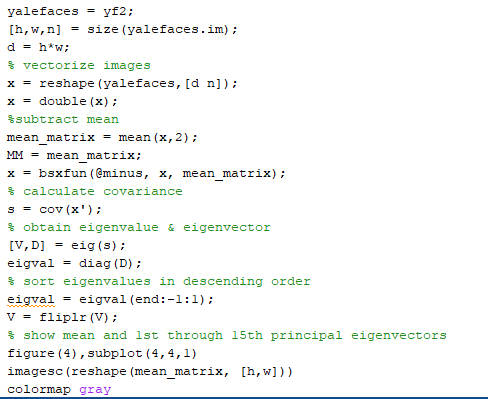
**Question 2 Answer:**

For [V,MM,PCA\_models] = CS4640\_PCA\_model(templates), I have used the week\_10 code.

Here are something that I have modified and why it works:



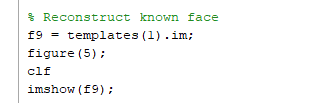
First, I have modified the yf2 since our template is 25x15, so we need a 25x15x50 array.



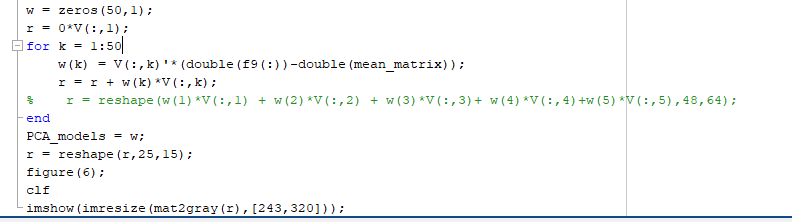
I still used variable name as yalefaces since it would be convenient to use the rest of codes.

We got our first output MM here, it equals to mean\_matrix.

By calling eig() function, we can get our second output V.



Then I have changed to f9 = templates(1).im since our input is templates not yalesface.



Then I have created w array, since we know we have 50 characters.

By using for loop, we can put weight values for first k eigenvectors into w.

Then we got out last output here PCA\_models, it equals to w.

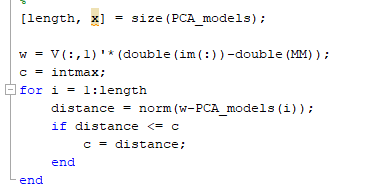
For c = CS4640\_PCA\_classify(im,V,MM,PCA\_models).

I first computed the w for the input im. I assume that the input V and MM are given for the input image.



I have used the equation given in CS4640\_PCA\_model(templates) to compute w.

Then I have used a for loop to compare each Euclidean distance I got, I only want to keep the smallest Euclidean distance.



I first got the length of the input PCA\_models.

Then I set c to the biggest int number we can set to make sure there is no number bigger than that, then I can do the comparison to store the smallest Euclidean distance to c, which would be my output.



Since the type of c is an integer, the last step is to set c to int.