Computer Vision for HCI Abhinav Mahalingam (.15)

CSE 5524

Homework #4

1. Use the regionprops Matlab function on boxIm1.bmp (provided on the class website) to compute its 'Area', 'Centroid', and 'BoundingBox'. Convert this [0, 255] image to be [0, 1] binary before calling this function. Plot/mark/identify the centroid and bounding box on the image. [2 pts]

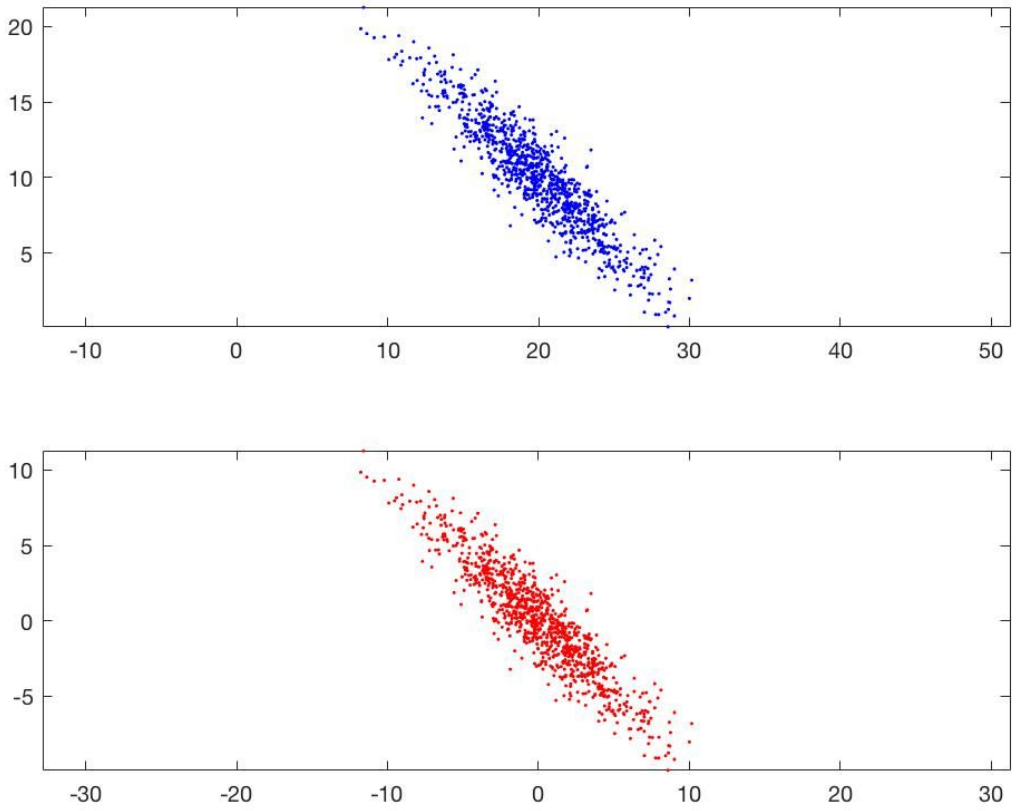
2. Write a function to compute the seven similitude moment shape descriptors. Test and compare results on the rectangle box images 'boxIm[1-4].bmp' on the website. How do they change across the box images? Please make sure your function will work with non-binary (grayscale) imagery (you will need this for later assignments), i.e., do not use regionprops for the mean or area. [3 pts]

boxIm1.bmp	0.0422	0	0	0	0.1646	0	0
boxIm2.bmp	0.0422	0	0	0	0.1646	0	0

boxIm3.bmp	0.0423	0	0	0	0.1641	0	0
boxIm4.bmp	0.1646	0	0	0	0.0422	0	0

We can see from the above images and results that the similitude moments are not affected by scaling or translation operations (as observed for fig 1,2 and 3). However the similitude moments do change when the object undergoes rotational transformation.

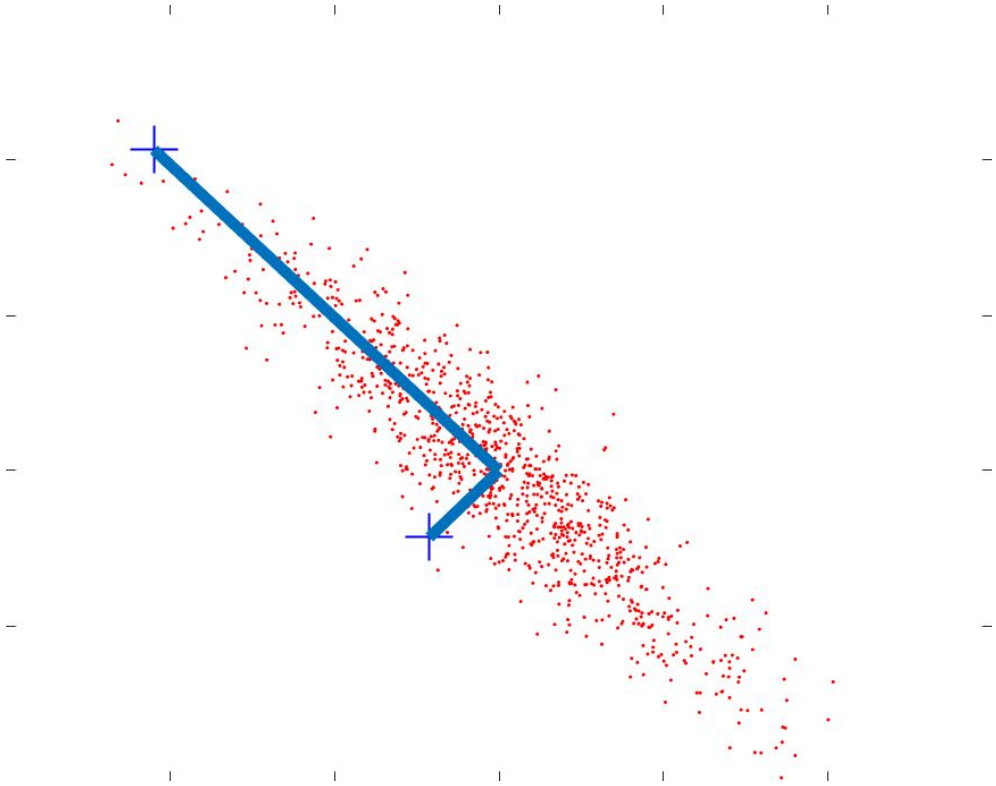
3. Using the datafile (eigdata.txt) provided on the WWW site, perform the following MATLAB commands [1 pt]:



The plot of the actual data and mean subtracted data can be seen above.

4. Compute the eigenvalues (V) and eigenvectors (U) of the data (stored in Y) using the function eig() in Matlab (*recall that you use either the covariance matrix or the Inverse-covariance matrix of the data – see class notes*). Plot the mean-subtracted data Y and the 2-D Gaussian ellipse axes for given the eigenvectors in U (*you can use the plot command in Matlab for*

this). Use the eigenvalues in V to give the appropriate 3σ (*standard deviation - not variance!*) length to each axis (*did you compute the eigenvalues from the covariance or inverse covariance of Y? The eigenvalues will be related but different! See class notes*). [4 pts]



eigen_val =

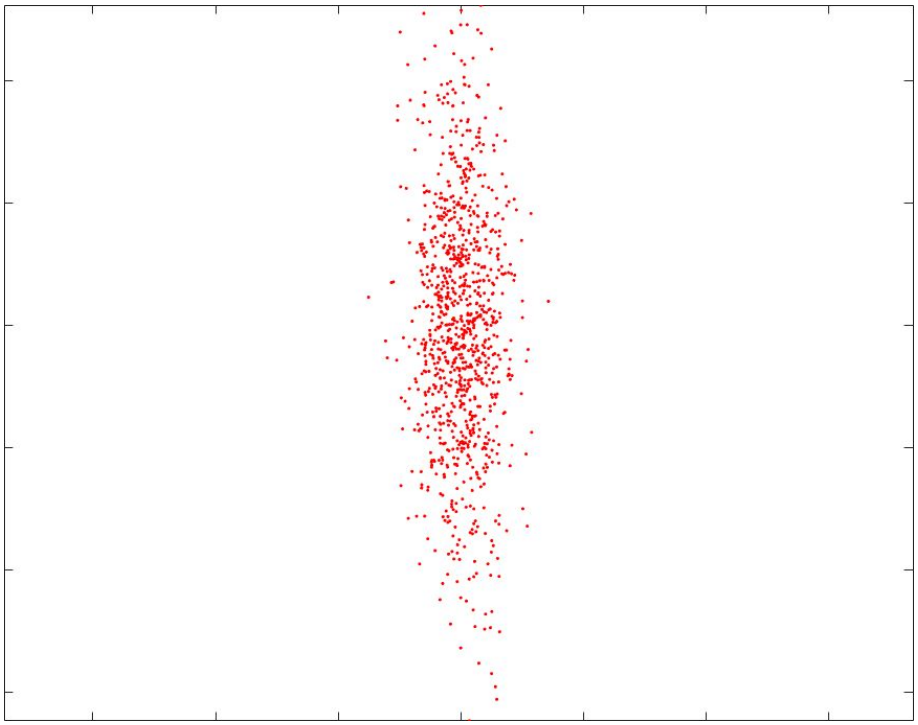
```
  24.1385         0
       0    1.0142
```

eigen_vec =

```
  -0.7126   -0.7016
   0.7016   -0.7126
```

We can see that the eigen vectors are orthogonal to each other. The primary eigen vector in direction of the largest variance and the secondary is in the direction of the next largest.

5. Rotate Y using the eigenvectors to be uncorrelated (i.e., project data Y onto the eigenvectors – see class slides). Plot the results. [2 pts]



```
% Abhinav Mahalingam
% CSE5524 - HW4
% 9/16/2017

% HW4.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem 1
Im = double(imread('input/boxIm1.bmp'))/255;
colormap('gray');
imagesc(Im);
st = regionprops('table',Im,'Area','Centroid','BoundingBox');
objectRegions = find(st.Area > 0);
hold on;
% Get the area, boundary and centroid for all regions in the image
for i=1:size(objectRegions)
   objReg = st(objectRegions(i),:);
   plot(objReg.Centroid(1),objReg.Centroid(2),'r.');

rectangle('Position',[objReg.BoundingBox(1),objReg.BoundingBox(2),objReg.BoundingBox(3),ob
jReg.BoundingBox(4)],'EdgeColor','r','LineWidth',2 );
   text(objReg.BoundingBox(1)+1,objReg.BoundingBox(2)+5,sprintf('Area: %g',objReg.Area));
                                   text(objReg.Centroid(1)+1,objReg.Centroid(2)+5,sprintf('Centroid:
(%g,%g)',objReg.Centroid(1),objReg.Centroid(2)));
end
saveFrame('results/regionProperties.png');
hold off;
pause;
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem 2
clear; close all;
colormap('gray');
for i=1:4
   Im = double(imread(sprintf('input/boxIm%d.bmp',i)))/255;
   simMoments = similitudeMoments(Im);
   imagesc(Im);
   title(sprintf('boxIm%d.bmp',i));
   text(10,10,sprintf('Similitude moments: %s',mat2str(simMoments,2)),'Color','Yellow');
   saveFrame(sprintf('results/SimilBoxIm%d.png',i));
   fprintf('boxIm%d.bmp',i);
```

```
    disp(simMoments);
    pause;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem 3
% Load the data
clear; close all;
load eigdata.txt;
X = eigdata;
subplot(2,1,1);
plot(X(:,1),X(:,2),'b.');
axis('equal');
m = mean(X);
Y = X - ones(size(X,1),1)*m;
subplot(2,1,2);
plot(Y(:,1),Y(:,2),'r.');
axis('equal');
saveFrame('results/dataPlot.png');
pause;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem 4
close all;
covariance = cov(Y);
[eigen_vec,eigen_val] = eig(covariance);
[eigen_vec,eigen_val] = sortDescending(eigen_vec,eigen_val);
C=9;
maj_axis_len = sqrt(C*eigen_val(1,1));
scaled_maj_axis = eigen_vec(:,1)* maj_axis_len;
min_axis_len = sqrt(C*eigen_val(2,2));
scaled_min_axis = eigen_vec(:,2)* min_axis_len;
plot(Y(:,1),Y(:,2),'r.');
hold on;
plot(scaled_maj_axis(1,1),scaled_maj_axis(2,1),'b+','MarkerSize',20);
plot(scaled_min_axis(1,1),scaled_min_axis(2,1),'b+','MarkerSize',20);
line([ 0;scaled_maj_axis(1,1)],[ 0;scaled_maj_axis(2,1)],'LineWidth',5);
line([ 0;scaled_min_axis(1,1)],[ 0;scaled_min_axis(2,1)],'LineWidth',5);
saveFrame('results/ellipseAxis.png');
hold off;
pause;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem 5
rotate_val = (eigen_vec*Y')';
plot(rotate_val(:,1),rotate_val(:,2),'r.');
axis('equal');
saveFrame('results/rotatedellipseAxis.png');
```

computeMoments.m

```
function [m] = computeMoments(Img,x,y,p,q)
  m=0;
  for r = 1:size(Img,1)
    for c = 1:size(Img,2)
        m = m + (Img(r,c)*((c-x)^p)*((r-y)^q));
    end
  end
end
```

getCentroid.m

```
function [centroidX,centroidY] = getCentroid(Img)
   m00 = computeMoments(Img,0,0,0,0);
   m10 = computeMoments(Img,0,0,1,0);
   m01 = computeMoments(Img,0,0,0,1);
   centroidX=m10/m00;
   centroidY=m01/m00;
end
```

saveFrame.m

```
function[] = saveFrame(fileName)
   frame = getframe;
   imwrite(frame.cdata,fileName);
end
```

similitudeMoments.m

```
function[N] = similitudeMoments(Img)
   [centroidX,centroidY] = getCentroid(Img);
```

```
   m00 = computeMoments(Img,0,0,0,0);
   moments = [0 2; 0 3; 1 1; 1 2; 2 0; 2 1; 3 0];
   N = zeros(1,size(moments,1));
   for r=1:size(moments,1)
      p = moments(r,1);
      q = moments(r,2);
      momentVal = computeMoments(Img,centroidX,centroidY,p,q);
      N(1,r) = momentVal/(m00^(((p+q)/2)+1));
   end
end
```

sortDescending.m

```
function [sorted_eigen_vec,sorted_eigen_val] = sortDescending(eigen_vec,eigen_val)
   [temp,ind_mat]=sort(diag(eigen_val),'descend');
   sorted_eigen_val=diag(temp);
   sorted_eigen_vec=eigen_vec(:,ind_mat);
end
```