**Programming Assignment 2**

**(1) 2D Rotation:** Write a MATLAB function(rotate.m) that takes a set of points in 2D and an angle (in degrees) and returns a new set of points which have been rotated **counter-clockwise** by that angle. Your function header should have the following format:

function Xrot = rotate(X,angle)

% function Xrot = rotate(X,angle)

%

% This function takes a set of points stored in X and

% applies a rotation specified by angle.

%

% arguments:

%

% X : a 2xN matrix of points where the first row gives the x coordinate

% of each point and the second row gives the y coordinate of each point

%

% angle : the amount to rotate counter-clockwise, in degrees

%

%

% return value:

%

% Xrot : a 2xN matrix containing the rotated points.

%

There are two ways to go about this problem. The preferred way (assuming you remember how rotation matrices work) is to build the appropriate 2x2 rotation matrix and multiply the vectors in X by the matrix to get the rotated vectors.  If you don't remember how to build a rotation matrix, another approach is to convert the (x,y) coordinates to polar coordinates (r,theta), add the appropriate angle to theta, and then convert back into euclidian (x,y) coordinates.

Hint: MATLAB's cos() and sin() functions expect angles specified in radians so you will need to convert from degrees to radians.

**(2) Image Rotation:** Write a MATLAB function (rotate\_image.m) that takes an image and an angle (in degrees) and returns a new image which has been rotated **counter-clockwise** around the center of the image by that angle.  Your function header should have the following format:

function Irot = rotate\_image(I,angle)

% function Irot = rotate\_image(I,angle)

%

% This function takes an image I and creates a new version of the image

% which is rotated by amount angle

%

% arguments:

%

% I - the original grayscale image, stored as a matrix

% angle - the amount by which to rotate the image

%

% return value:

%

% Irot - an image which containing the rotated original

%

To accomplish this, your code will need to carry out the following steps:  
  
1. Compute the x,y coordinates of the pixels in the image relative to the center of the image.  For example, if the image is 101 pixels wide, the x coordinate should range from -50 to 50 pixels. You may find the MATLAB function **meshgrid** useful. **[xcoord,ycoord] = meshgrid(-5:5,-5:5)** should give you arrays ycoord and xcoord containing the pixel coordinates for an 11x11 image.

2. Put those (x,y) coordinates into a 2xN matrix X and use your rotate function from the previous exercise to rotate them.  To accomplish this, you will need to choose an ordering for the pixels (e.g. start with the coordinates for the first row of the image, followed by the second, etc.).  If you used **meshgrid** to generate the coordinates then you will need to reshape each of the arrays of coordinates into a 1xN vector and then concatenate those vectors into the 2xN matrix.  You can accomplish this using the colon operator, e.g. **X = [xcoord(:) ycoord(:)]'**.  You also need to reshape the image brightness values into a 1xN vector: **Ivector = I(:)'**

3. Now figure out the dimensions of a new image that will include all of the rotated points.  You can do this by finding the maximum and minimum values of the x and y coordinates stored in Xrot.

4. Generate coordinates for the pixels in the new image, **[newxcoord,newycoord] = meshgrid(xmin:xmax,ymin:ymax)**.  Note that in general, this grid will be bigger than your original image.  You will also want to round xmin, xmax, etc to integers before generating the grid.

5. Now we have the coordinates for the rotated pixels and the new pixel grid we want to map them onto.  In order to do the interpolation of brightness values from the old grid to the new grid, you'll want to use the builtin MATLAB function**griddata**. Read the help file for griddata.  The way you will call it will look like: **Irotated = griddata(Xrot(1,:),Xrot(2,:),Ivector,newxcoord(:),newycoord(:));**

6. This gives you the intensity for all the new pixels, however, they will also be in the form of a Nx1 vector.  To get back an image, you will need to reshape this vector to be the same size as **newxcoord**, the grid of points generated by meshgrid using the **reshape**function.

**(3) User Interaction:** Write a MATLAB script (stored as an ascii text file, with the name straighten.m) that loads in an image, allows the user to click on two points in the image, and then rotates the image so that the line connecting these two points is horizontal. This script will use your  **rotate\_image.m** function as a subroutine to generate the rotated image.

Load and display the image as in test\_rotate\_image.m from the previous exercise.  In order to get user mouse clicks, you script can call the MATLAB function **ginput** to get the image coordinates of two mouse clicks.  Your function should plot the points the user clicked on top of the image by calling **hold on** and then **plot().** Given these two mouse clicks, you will need to do a little trigonometry to find the angle by which you'll need to rotate the image in order to make those two points lie on a horizontal line.  Rotate the image by that amount and display the result.

**(4) 提交报告内容**：包括实现原理，程序输入与输出对比，结果分析，及代码。上面以matlab为例进行说明，但不限编程语言。