Instructions:

You will need to install MATLAB to perform some of these tasks. You can see the code for doing some basic MATLAB operations with images in the Implementation Notes section.
If you do not have MATLAB installed, you can also use the free GNU Octave with same syntax as MATLAB. Most of the commands given will work on Octave without modification. You can also use Python(with numpy library) to do some of these tasks, however, no documentation will be provided for Python programming.

Each student is required to submit an individual report of the project.

For each problem, you have only a fixed number of pages to write your answer. You have to fit all derivations, images and calculations(everything except code) within the set number of pages. Code for each section will be submitted separately. Any answers outside the number of pages alloted will not be considered.

Each student will submit a PDF report named `P1-1YourRollNo.PDF` and a zipped folder containing code for each section named `P1-1YourRollNo.zip`. The last page of PDF file should also explain how to execute your code. All submissions are to be made to TA before the deadline at soofia.mirza@hotmail.com. After the deadline, 30 percent of the project's points will be lost with each passing day, for up to 3 days.

All announcements related to this project, any changes, corrections or updates will be posted on Piazza post. Please keep yourself updated with these changes.

Problem 1:(30 points, 2 pages max)

The objective of this part of the project is to understand and apply image filtering and edge detection

Instructions on how to read image files into MATLAB are given in Implementation Notes section.

P1.1: Read the image `'cameraman.tif'`. This image is copied in your MATLAB directory when you install MATLAB. Find gradient of this image using imgradient MATLAB function. Display angles of gradient as a 2D map.

P1.2: Read the image `'lines.png'` and display it. This image has been provided along with the handout for this project.

This image has several lines drawn at different angles. We are interested in finding a 3x3 convolution filter, which when applied to the image, only returns the line(s) at 45 deg from the x-axis.

In other words, the convolution filter should return high values when applied on a line drawn at 45 deg from the x-axis, and return low values otherwise. Using your filter, locate the line along 45 deg.

P1.3: Read the cameraman image again, denoted as X, this time apply the simplest edge detector $F = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$ on it to find Y.
Is it possible to go back to retrieve X from Y? Given that $y_n = x_{n-1} - x_{n+1}$, can you express X in terms of Y. Can you design a 3x3 filter G that performs the opposite of F? If yes, provide the 3x3 filter. If no, provide a proof showing such a filter does not exist.

Implementation Notes

MATLAB has an extensive help available online. If some MATLAB command is missing here, or for more details on any of these commands, you can use MATLAB's help at www.mathworks.com

```
f=imread('MyImageFileIn.tif');% Read an image file

imwrite(f,'MyImageFileOut.tif');% Write an image file

imagesc(f);colormap(gray);colorbar;% Display an image
imshow(f);% Display an image

hist(f(:),[0:255]);% Display histogram of the read image

grayimg=rgb2gray(colorimg);% Convert a color image to gray
scale

Filt=[0,1,0;1,0,1;0,1,0];
g=imfilter(f,Filt);% Filter an image f to produce g
[camMag,camAng]=imgradient(I);% Compute the magnitude and angle
of the gradient of an image

F=fft2(f);
F=fftshift(F);
imagesc(abs(F))% Find 2D FFT F(u,v) of an image f(x,y)
 % Sometimes the dynamic range of FFT is too large, so you may
see one or two impulses only while actually there is more.  To
see such a FFT, use imagesc(log(abs(F))) to squeeze the dynamic
range.

fr=ifft2(ifftshift(F));
imagesc(abs(fr))% Find Inverse 2D FFT fr(x,y) of a 2D FFT
F(u,v)

surf(X,Y,Z)% Plot a surface Z = f(X,Y)

imhist(f) or hist(f(:))% Plot histogram of image f
```

[1]

[1]

Assistant Professor, FAST-NU
sadiq@purdue.edu