UNIVERSITY OF CALIFORNIA, SANTA BARBARA
Department of Electrical and Computer Engineering

Prof. P. Sen **ECE 178 Digital Image Processing** Fall 2017

Homework 3

Due Monday, October 23, at 9pm. MATLAB portion due on GauchoSpace.
Written portion can be submitted either in the homework box or on GauchoSpace.
(please write your name and discussion section number clearly at top of first page)

1 Book Problems [9 Points]

1. 3.15
2. 3.19
3. 3.22

2 MATLAB [16 Points]

This MATLAB assignment consists of two parts. The first part is about 2D convolution and the second part is about image transformation.

There is a wrapper code that takes care of generating input test images, calling your functions and displaying the output images. The wrapper code along with the input images are available on GauchoSpace inside **HW3_MATLAB.zip**.

PART 1:

In the first part, we are going to implement 2D convolution with three kinds of padding methods, i.e., **zero-padding**, **mirroring**, and **repeating**, to handle region outside the boundary. In addition, we are going to implement **bilateral filter** and **Gaussian filter**. These filters are the easiest implementations for recovering a noisy image. For part one, you have to write three functions (*myGaussianFilter.m*, *myBilateralFilter.m* and *myConv2.m*). It's **not allowed** to use the built-in *conv2* function for this HW.

For the **myGaussianFilter** function, you have to specify the size (width and height) of the filter, and the mean and the standard deviation of the Gaussian distribution as inputs. Your output will be a Guassian filter kernel.

We will then use the generated Gaussian filter kernel to de-noise the noisy input by performing the convolution operation. To perform this convolution operation you need to implement **myConv2** function. The function takes an image, a spatial filter kernel, and a padding method as inputs. Please refer to **myConv2.m** in the provided zip file for more guidelines to implement this function.

Another popular way of de-noising an input image is bilateral filtering. The **myBilateralFilter** function takes an image, a window size, $sigma\_s$ and $sigma\_r$ as inputs, where $sigma\_s$ is the the standard deviation of Gaussian function to smoothing differences in coordinates, and $sigma\_r$ is the the standard

deviation of Gaussian function to smoothing differences in intensities, i.e. we use zero mean Gaussian functions to do the bilateral filtering.

PART 2:

In the second part, we are going to implement DCT and IDCT in *myDCT_Transform.m* and *myIDCT_Transform.m* respectively. It's **not allowed** to use the built-in *dct2*, *idct2*, *dct*, and *idct* functions for this HW.

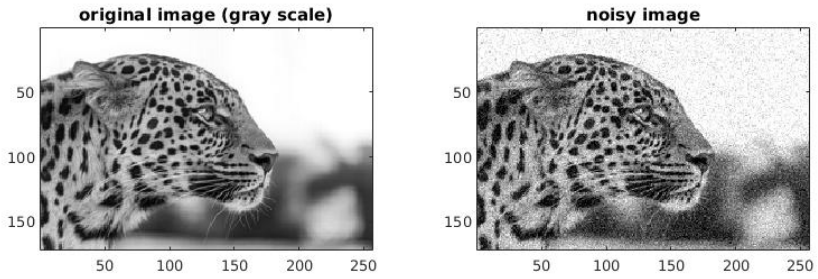
For the **myDCT_Transform.m**, it only takes an image as an input and produces a transformed image.

For the **myIDCT_Transform.m**, it only takes a transformed image as an input and produces a recovered image.

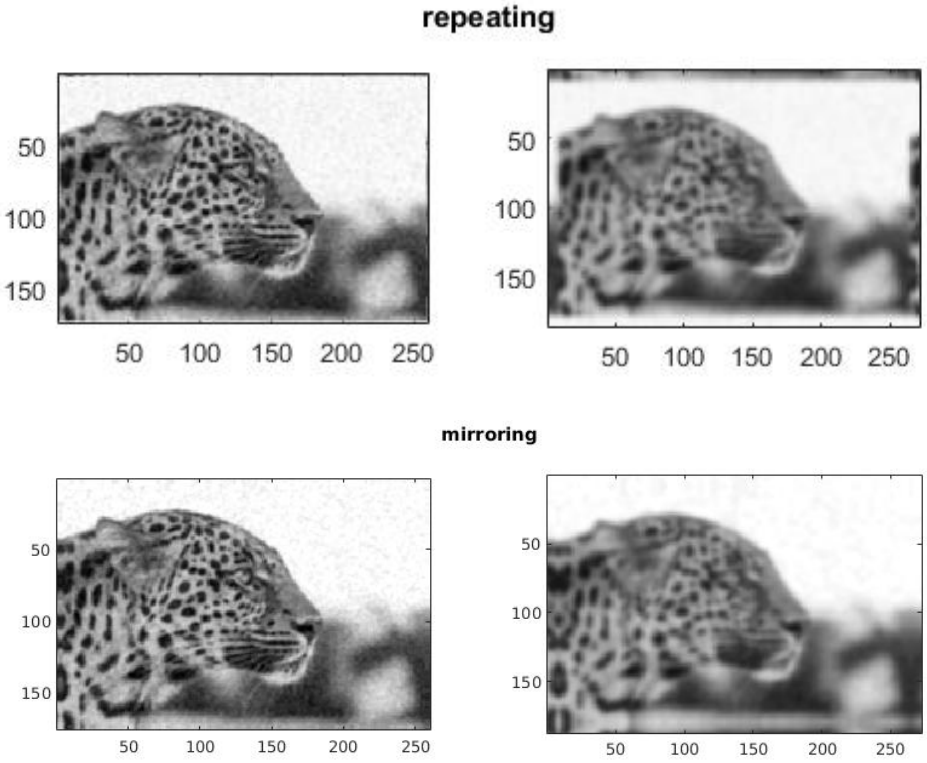
ADDITIONAL DETAILS:

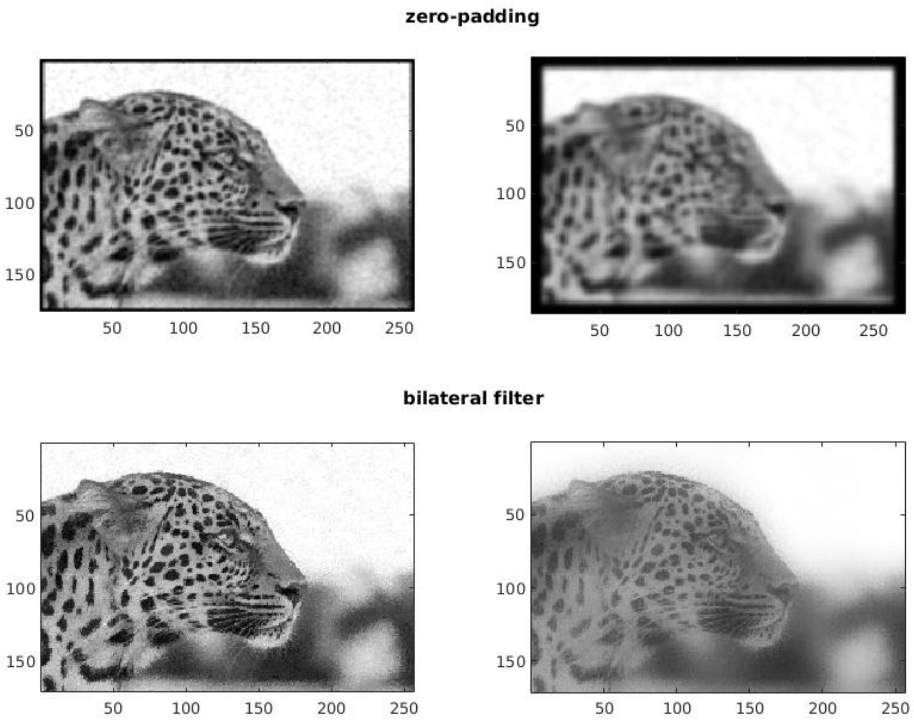
The main function is **HW3_main.m**. It processes the input image as following:

It loads the original image, resizes it so the computations will be faster, converts it into grayscale, and adds noise.

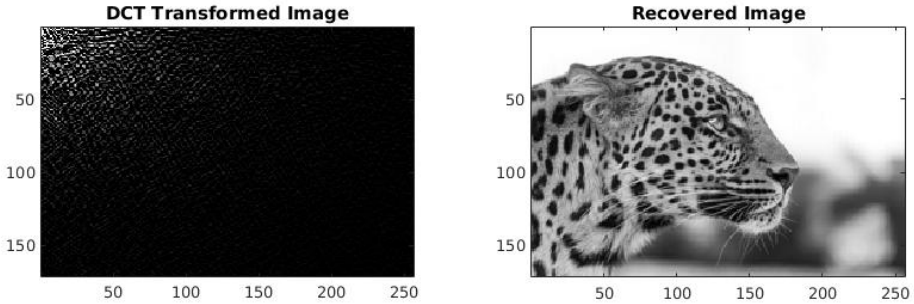


Once you get your code done, the main file will display the denoised images. The results should look like these, notice how the boundary of the images change as you change the padding method:





And the DCT transformed image and the recovered image:



SUBMITTING YOUR CODE:

For submitting your code, please upload one zipped file includes 8 files:
HW3_main.m
display2images.m
Leopard.jpg
myGaussianFilter.m
myBilateralFilter.m
myConv2.m
myDCT_Transform.m
myIDCT_Transform

You **don't have to** attach your result images. Please use the following convention to name the zip/tar files: <Perm number>_<First name>_<Last name>_HW3. Good luck!