Homework 3

ECE 253
Digital Image Processing

October 31, 2015

Instructions :

- Homework 3 is due by 11:59 PM, November 8, 2015.
- Submit your homework electronically by email to arangesh@ucsd.edu with the subject line *ECE 253 HW3*.
- The email should have one PDF file of your writeup attached. Make sure it includes your full name, PID, and email. This file must be named ECE_253_hw3_lastname_studentid.pdf.
- All problems are to be solved using MATLAB unless mentioned otherwise.
- You should avoid using loops in your MATLAB code unless you are explicitly permitted to do so.
- Append source code, floats, matrices, and image outputs, to your writeup where applicable. Simply pasting your code in the report should suffice, but make sure it is indented correctly.

Problem 1. Median versus Mean Filtering (10 points)

In the data appended, find an image named *Helmet.jpg*. Convert it to a grey-scale image.

(i) Add noise to this image by using the following commands:

```
noise = rand(size(Helmet));
noisy_image = uint8(double(Helmet) .* (noise > 0.2) + 255 .* (noise < 0.1));
```

where the grey-scale image is stored in *Helmet*. What kind of noise is this?

(ii) Write a function that performs two-dimensional median filtering with a 5-point cross-shaped kernel (3x3 in size). The function takes a noisy grey-scale image as an input, and returns the filtered image as an output. You may use loops if necessary.

Use the above function to clean up the noisy image generated above. In addition, use the command *medfilt2()* to do median filtering with different sized kernels. In addition to your cross-shape, try the following sizes: 1x2, 1x3, 2x2, 3x3, 3x4, 4x4, 4x5, 5x5 and 7x7.

If a median filter were able to perfectly clean up the noisy image, then the mean-squared error (MSE) between the cleaned-up version and the original image would be zero. In practice, this doesn't happen. We can use the MSE as a simple measure of how well a given median filter is doing. Plot the MSE corresponding to each kernel above (including the cross-shaped kernel). Include the MSE between the original image and the noisy image in your plot as well, as being unfiltered case (a one-point median filter). In conclusion, the plot should contain the size (or type) of the kernel along the X-axis, and the corresponding MSE on the Y-axis.

(iii) Apply a mean (smoothing) filter with kernel sizes 3x3, 5x5 and 7x7 on the same noisy image as above. One way to do this is to set up a kernel as per your need (make sure you normalize it) and convolve it with the noisy image as follows:

```
% Convolve keeping size of noisy_image
im_out = uint8(conv2(double(noisy_image), kernel, 'same'));
```

Plot the MSE corresponding to these three kernels in the same manner as above.

In your report, include the original grey-scale image, the noisy image, and the images after filtering (use of *subplot()* is recommended). Also, include the two MSE plots corresponding to the mean and median filter. Additionally, include your answers (no more than three sentences each) to the following questions:

- On an average, which of the two filtering techniques (mean or median) works better. Why do you think this happens?
- For the median filter, which kernel produces the best result and why?

Problem 2. Edge Sharpening (10 points)

In general, the following relation holds:

$$HPF + LPF = IF \tag{1}$$

where **LPF**, **HPF** and **IF** denote a *low pass*, *high pass*, and *identity* filter respectively.

An *identity* filter is an NxN mask with a 1 at the center. Filtering an image with an IF will leave the image unchanged. If we use an LPF on our image, then the lines/edges will be blurred. To sharpen images (or detect edges), we use a HPF. However if we high pass filter our image, then everything is sharpened up. This might not be what we want if the original image itself consists of smooth and slow transition area. Therefore we will be creating a *high boost* filter (**HBF**) defined as:

$$HBF = IF + \alpha HPF \tag{2}$$

where α is a positive constant. The high boost filter amplifies the high frequency components and it keeps the low frequency components (almost) intact. In general, $\alpha \geq 1$ for high boost filtering. In the special case when $\alpha = 1$, the process is called *unsharp masking*.

(i) Using equations (1) and (2), show that:

$$HBF = (1 + \alpha)IF - \alpha LPF$$

(ii) Consider the following MATLAB function used to sharpen an image:

```
function im_out = unsharp( im_in, maskA, weight )
[a,b] = size(maskA)
maskB = zeros( size(maskA) )
maskB(ceil(a/2), ceil(b/2)) = 1
maskC = maskB - maskA
maskD = maskB + weight * maskC
im_out = conv2(im_in, maskD, 'same')
```

Here *im_in* is the input image and *im_out* is the output image. Suppose *maskA* is a small odd-sized low-pass filter mask, and *weight* is a positive number greater than 1. What kind of masks are masks B, C and D? Using the background above on separating an image into low-pass and high-pass components, explain how this function performs edge sharpening.

(iii) In HBF, there are basically two things you can choose: which low-pass filter to use and the weight α to be assigned. We will investigate the effect of each on the resulting image. First, create a synthetic image of size 128x128 that consists of a ramp and simple step function, as follows:

```
image_synth = ones(128,1)*[64*ones(1,32) (64:4:188) 192*ones(1,32) 64*ones(1,32)];
```

This has four equal sized areas (from left to right): first, 32 columns with value 64, then 32 columns of ramp going from 64 to 192, 32 columns with value 192 and finally 32 columns with value 64. Try a few combinations of low pass filters and weights. Vary the size of the low pass kernel(e.g. 3x3, 5x5, maybe even 7x7), the type of low pass kernel and the weight α. Also, look at a horizontal slice of the filtered images (for example, $plot(image\_synth(64, :))$) and compare it with the same slice in the original image.

In your report, include three plots, found by varying one of each of the above parameters (i.e. low-pass filter size, low-pass filter type and α) while keeping the other two constant. Also include the slice from the original image in each plot to serve as a baseline. Discuss how each parameter affects the image output and any trends that you may have noticed.

(iv) Finally, let's look at a real world example. Read the image *xray.tif* into MATLAB. Find a good combination of low-pass filter and weight so that the enhanced image ”looks good”. What has changed about the image? What can you say about the noise level? You may find that the values in the processed image exceed the range of [0,255], and so you may need to truncate or re-scale the values. If you need to, does truncation or re-scaling look better? With the right combination of masks and weights, make sure that you can discover what numbers are written on the vertebrae.

In your report, include the original and enhanced image. Mention the filter type, size and weight you used to obtain this image. Additionally, answer the above questions in no more than two sentences each.

Problem 3. Binary Morphology (10 points)

In this problem, you will perform binary morphological operations on noisy versions of the image *eyechart.tif*. Along with the clean image, you will find three noisy versions : *eyechart_1.tif*, *eyechart_5.tif*, *eyechart_20.tif* with 1%, 5% and 20% salt and pepper noise respectively. Note that the images will be stored in *uint8* format, and need to be converted into binary double arrays which is the format required by MATLAB for binary image processing. To do so, use:

```
binary_im = 1 - round(double(image)/255);
```

Now, the binary image has values of 0 and 1, and the foreground pixels (letters) have value 1.

(i) **Isolated Pixel Cleaning :** We are interested in cleaning up these noisy versions. Write a function that does isolated pixel cleaning. That is, a black pixel entirely surrounded by white pixels should be flipped, as should a white pixel entirely surrounded by black pixels. The function should take the noisy binary image as an input and return the clean image as an output. You are not allowed to use the *bwmorph()* routine. You may use loops if necessary. Apply this function to the three noisy images.

In your report, include the binary images before and after cleaning (6 in total).

(ii) **Cleaning using opening and closing :** Opening followed by closing can also be used for cleaning. Try it on the three noisy images (use *bwmorph()* routine with proper arguments). Now, repeat the experiment by first closing and then opening the noisy images. Do both techniques produce the same result? Why or why not? How do they compare to isolated pixel cleaning?

In your report, include the binary images after cleaning (3 for each technique), and the answers to the questions above in no more than three sentences.

Finally, tabulate the MSE values before and after cleaning, for all three noisy images, and for all three cleaning techniques (isolated pixel, opening after closing and closing after opening). Note that all MSE values must be calculated with respect to the original binary image obtained from *eyechart.tif*.