**polyu_logo_2015 eie_logo**

**EIE 529 Digital Image Processing**

**Laboratory Exercise 2**

**Equipment**

1) PC with Windows 7 or above;

2) Matlab R2009b or above with image processing toolbox;

3) Microsoft Visual C++ 2008 SP1 Redistributable Package.

**Objectives**

This laboratory exercise is divided into two parts. In part A, different image denoising methods will be compared. In part B, the Wiener filter and regularized minimization methods are studied for image deblurring.

**Part A. Image denoising**

Follow the procedures below.

1. Download the files “Bilateral.zip”, “BM3D.zip” and the image “Lena512Gray.bmp” from the Blackboard to your home directory.
2. Unzip “Bilateral.zip” and “BM3D.zip” to your home directory with the paths “*home* *directory*\Bilateral\” and “*home directory*\BM3D\, respectively.

**(a) Denoising with bilateral filters**

1. Create a program called “CompareDenoise.m” in your home directory.
2. In that program, add the following paths:

addpath('Bilateral')

addpath('BM3D')

1. Load the image “Lena512Gray.bmp”, add noise to it and then show the noisy image as follows:

y = im2double(imread('Lena512Gray.bmp')); % y is in the interval [0,1]

sigma = 25;

n = (sigma/255)\*randn(size(y)); % So the normalized noise standard deviation is sigma/255

z = y + n;

figure(1)

imshow(z);

Capture the screen that shows the noisy image. Paste the screen in the box below.

1. Give the PSNR of the image.

1. In your program “CompareDenoise.m”, add the following statements to denoise the image using the bilateral filter.

Gaussian\_sigma = [3 0.1];

dx\_bi = RUNBFILTER(z, Gaussian\_sigma);

% Output – dx\_bi: denoised image in the interval [0,1]

% Input – z: noisy image in the interval [0,1]

% Gaussian\_sigma: controls the shape of the Gaussian filters

figure(2)

imshow(dx\_bi)

Capture the screen that shows the denoised image. Paste the screen in the box below.

1. Give the PSNR of the image.

1. The input parameter Gaussian\_sigma controls the shape of the Gaussian filters used in the bilateral filter. The current values are [3 0.1]. Change the values to [10 0.1] and [3 1]. Run the program to see the differences in the resulting image when different sigma values are used. Capture the screen that show the resulting denoised image in each case. Paste the screens in the box below.
2. From the results in (7) above, explain how the two sigma values affect the resulting denoised images.

**(b) Denoising with BM3D**

1. In your program “CompareDenoise.m”, add the following statements to denoise the image using the BM3D algorithm.

sigma = 25;

dx\_bm = BM3D(1,z,sigma);

% Output – dx\_bm: denoised image in the interval [0,1]

% Input – z: noisy images in the interval [0,1]

% sigma: Noise standard deviation (the true one, not the normalized one)

figure(3)

imshow(dx\_bm)

Capture the screen that shows the denoised image. Paste the screen in the box below.

1. Give the PSNR of the image.

1. BM3D is based on the Wiener filter. Thus, it requires the knowledge of the noise variance. The current sigma is set to 25, which is exactly the standard deviation of the noise. Change this input parameter to 10 and run the program again. Describe the difference of the resulting image from that in (1).

1. In practice, the noise variance is not known in advance. We can only estimate the noise variance based on the image on hand. One of the approaches is based on the robust statistics. For zero mean white Gaussian noise, we can estimate the standard deviation of noise in a noisy image by the following equation:

where is the sparse transform (e.g. wavelet transform, discrete cosine transform (DCT), etc.) of the image . In your program “CompareDenoise.m”, add the statements to estimate the standard deviation of the noise based on the above equation. You may want to use the Matlab functions dct, median and mean. Type help dct, help median and help mean on the Matlab console to understand how to use these functions. Hint: it is more straightforward to convert the 2-D transform coefficients into a 1-D vector before you compute the mean and median, since they are 1-D functions. A 2-D matrix can be converted to a 1-D vector by the following statement:

y1d = y(:) % y1d is a 1-D vector created by stacking every column of y

Give the statements you have used, and the noise standard deviation estimated by your statements below.

**Part B. Image deblurring**

**(a) Deblurring by inverse filtering**

1. Create a program called “ImageDeblur.m” in your home directory.
2. Load the image “Lena512Gray.bmp”, blur and add noise to it. Then show the blurry image as follows:

%%% Image deblurring by different methods %%%

%

f = im2double(imread('Lena512Gray.bmp')); % f is in the interval [0,1]

h = [1 1 1 1 1 1 1; 1 1 1 2 1 1 1; 1 1 2 4 2 1 1; 1 2 4 8 4 2 1; ...

1 1 2 4 2 1 1; 1 1 1 2 1 1 1; 1 1 1 1 1 1 1];

h = h/(sum(sum(h))); % Assume the PSF is known

r = conv2(h,f); % The blurring process is modelled by a 2-D convolution of the PSF and the image f

sigma = 3;

n = (sigma/255)\*randn(size(r)); % So the normalized noise standard deviation is sigma/255

g = r + n;

figure(4)

imshow(g);

Capture the screen that shows the blurry image. Paste the screen in the box below.

1. Give the PSNR of the image. Note that the size of g is a bit bigger than f. You need to chop out the relevant part of g to compare with f when computing the PSNR.

1. Add the following statements to the program “ImageDeblur.m” to deblur the image using the inverse filtering method.

%%% Deblur by inverse filtering

%

N1 = size(g,1);

N2 = size(g,2);

G = fft2(g); % fft2 performs the 2-D DFT

H = fft2(h,N1,N2);

FH = G./H;

fh = real(ifft2(FH)); % ifft2 performs the inverse 2-D DFT

% fh has to be a real function

figure(5)

imshow(fh);

Can the program generate the deblurred image? If yes, capture the screen that shows the deblurred image. Paste the screen in the box below. If no, just paste what the screen shows.

1. If the program can generate the deblurred image, give the PSNR of the image. Remember to chop out the relevant part of fh to compare with f when computing the PSNR. If not, explain why the program cannot generate the deblurred image.

**(b) Deblurring by Wiener filter**

1. Add the following statements to the program “ImageDeblur.m” to deblur the image using the Wiener filter.

%%% Deblur by Wiener filter %%%

%

Pn = (abs(fft2(n,N1,N2)).^2)/(N1\*N2);

Pnn = mean(Pn(:)); % Assume the noise psd is known

Pgg = (abs(fft2(g,N1,N2)).^2)/(N1\*N2);

% The signal psd is approximated by subtracting Pnn from Pgg

% Pgg is approximated by its periodogram, since we have only one observation of g

Pff = max(Pgg-Pnn,0.000001); % To give a lower bound (greater than 0) for Pff to avoid data instability

FH = G.\*conj(H)./((abs(H)).^2+Pnn./Pff); % Wiener filtering

fh = min(max(real(ifft2(FH)),0),1); % fh must be real and in the interval [0,1]

figure(6)

imshow(fh)

Capture the screen that shows the deblurred image. Paste the screen in the box below.

1. Give the PSNR of the image. Remember to chop out the relevant part of fh to compare with f when computing the PSNR.

1. Comment on the quality of the deblurred image as compared with the original one.

**(c) Deblurring by regularized minimization**

1. We can also deblur an image using the regularized minimization method. With this method, the blurry image is multiplied with the following gain function to obtain an estimate of the original image in the Fourier domain.

where , and are the Fourier transform of the PSF , blurry image and the Laplacian function . is a constant which balances the two terms of the denominator. To implement this method, first add the following statements to “ImageDeblur.m” to define the Laplacian function.

%%% Deblur by regularized minimization %%%

%

lap = [0 -1 0; -1 4 -1; 0 -1 0]; % The Laplacian function

Lap = fft2(lap,N1,N2);

Then, modify the statements you used for the implementation of the Wiener filter in (b) to implement the above equation. At the moment, you may arbitrarily select a value for (say, ). We will look for a better later.

Give the statements you used for the implementation of the regularized minimization method in the box below.

Capture the screen that shows the deblurred image. Paste the screen in the box below.

1. Give the PSNR of the image. Remember to chop out the relevant part of fh to compare with f when computing the PSNR.

1. Comment on the quality of the deblurred image as compared with that given by the Wiener filter.

**(d) Finding the best**

1. The value of can affect the quality of the deblurred image. Define , where is the deblurred image. If is close to the original image , should be close to , where is the Fourier transform of the noise . Consequently, the best should be the one that minimizes the cost function .

Modify your program in (c) such that it will search for the best . Your program should have a for loop such that it will compute the deblurred image using the regularized minimization method with ranging from 0.001 to 1 with a step size of 0.001. For each value of , compute the value of . The that gives the minimum will be the best .

Give the statements you used for the implementation of the regularized minimization method with the searching of the best in the box below.

**Run the program and show the above result to your tutor.**

Capture the screen that shows the deblurred image with the best . Paste the screen in the box below.

1. Give the PSNR of the image. Remember to chop out the relevant part of fh to compare with f when computing the PSNR.

1. Comment on the quality of the deblurred image as compared with that in (c).

1. Explain the advantage of the regularized minimization method over the Wiener filter method for deblurring an image.

**References**

[1] *Matlab Reference Guide*, The Math Works Inc., 1992.

[2] R.C. Gonzalez and R.E. Woods, “Image Restoration and Reconstruction”, Digital Image Processing 4th Ed., Prentice Hall, Chapter 5, 2018.

End