University of Waterloo
Department of Electrical and Computer Engineering
Image Processing and Visual Communications

Instructor: Prof. Zhou Wang

Homework 2

- In this Homework, you will program with MATLAB to solve some image processing problems, and make comments on your observations. If you have not worked with MATLAB before, there are many excellent online resources for beginners, one of which is by MathWorks:

  https://www.mathworks.com/help/matlab/getting-started-with-matlab.html
- Type *demo* at the MATLAB command prompt to learn some basics about MATLAB. You will usually find the *help* & *lookfor* commands pretty useful.
- It is a good idea to always clear your workspace variables in MATLAB using the *clear all* command before running a new program. You may also close all figures by using the *close all* command.
- *for/while* loops are very inefficient in MATLAB. In most cases, they can be replaced by properly arranging your data with vector and matrix forms, which are much more efficient. Unnecessary use of *for/while* is discouraged.
- Label and scale all your plots properly (use *xlabel, ylabel, title, axis*, etc).
- Use the *subplot* function when you need to draw multiple sub-figures in one figure.
- You will find the *imread*, *imwrite*, and *imshow* functions very useful when working with image processing problems.
- Work on the Homework independently and create your own implementations of the algorithms. Group work or group homework submissions are not allowed. In addition to the basic MATLAB operations, follow instructions to utilize the "useful MATLAB functions" (or similar functions) indicated in each problem. Do not use existing open source code. Do not use the built-in functions in MATLAB or its toolboxes that may directly solve the problems.
- Submit your Homework 2 in one single file that includes your answers to all questions. For each question, include your MATLAB source code, and the figures you generated, followed by brief descriptions of your observations. Repeat the same sequence for all questions.

---

1. Extraction Bitplanes from Image and Image Reconstruction from Bitplanes

Read the original image ‘slope.tif’, which is an 8 bits/pixel gray scale image.

Extract its 8 bitplanes as 8 binary images. Name the most significant (upper) bitplane as Bitplane 1, the second most significant as Bitplane 2, and so on.

Reconstruct 8 versions of the original image, the first using the most significant (upper) bitplane only, the second using the upper 2 bitplanes only, …, and the eighth using all 8 bitplanes.

Create 4 figures, each with 4 sub-figures of 2x2 layout. In Figure 1, show the upper 4 bitplanes in 4 sub-figures. In Figure 2, show the lower 4 bitplanes in 4 sub-figures. In Figure 3, show the first 4 reconstructed images in 4 sub-figures, corresponding to using the upper 1, 2, 3, and 4 bitplanes, respectively. In Figure 4, show the remaining 4 reconstructed images in 4 sub-figures, corresponding to using the upper 5, 6, 7 and 8 bitplanes, respectively.

In a few sentences, briefly describe your observations of the results you obtained.

Useful MATLAB functions: *imread, figure, subplot, imshow, title, mod, bitshift*

2. Intensity Transformations: Gamma Mapping, Full-scale Contrast Stretch, and Histogram Equalization

Read the original image ‘books.tif’, which is an 8 bits/pixel gray scale image.

Create a Gamma mapped image by applying the power law (Gamma) intensity transformation to the original image. Specifically, scale the pixel intensity from [0, 255] to [0, 1], apply a power law to all pixels for Gamma = 0.5, and then scale the pixel intensity back from [0, 1] to [0, 255].

Create a full-scale contrast stretched image from the original image.

Create a histogram equalized image from the original image.

Create 2 figures, each with 4 sub-figures of 2x2 layout. Figure 1 shows the original, Gamma-mapped, full-scale contrast stretched, and histogram equalized images, respectively. Figure 2 shows their corresponding histograms of 256 histogram bins representing the pixel values between 0 and 255.

In a few sentences, briefly describe your observations of the results you obtained.

Useful MATLAB functions: *imread, figure, subplot, imshow, title, bar, hist, cumsum*

3. Frequency Domain Low-pass, Band-pass and High-pass Filtering

Read the original image 'bridge.tif', which is an 8 bits/pixel gray scale image.

Apply 2D-DFT to the original image. The DC component will be at the top-left corner. Shift the frequency domain spectrum by half of the image size such that the DC component (0 frequency) moves to the center.

Apply an ideal lowpass, an ideal bandpass, and an ideal highpass filters in the 2D-DFT domain. The frequency response of the ideal lowpass filter equals 1 within a circular region of a radius of $\pi/8$ (i.e., one-eighth of the distance from the center to the horizontal or vertical edge), and 0 otherwise. The frequency response of the ideal bandpass filter equals 1 between radius $\pi/8$ and $\pi/2$, and 0 otherwise. The frequency response of the ideal highpass filter equals 1 outside of the circular region of $\pi/2$, and 0 otherwise.

Shift the ideal lowpass, bandpass and highpass filtered images in the frequency domain back by half of the image size such that the DC component moves back to the top-left corner. Apply inverse 2D-DFT and take the real-parts to obtain their spatial domain images (the imaginary parts of these images are theoretically 0, but may not be completely 0 due to numerical errors).

Create 2 figures, each with 4 sub-figures of 2x2 layout. Figure 1 shows the original, lowpass filtered, bandpass filtered, and highpass filtered images in the spatial domain. Since the bandpass and highpass filtered images may have both positive and negative values, add the mid-gray value (128 for 8 bits/pixel images) to all pixels to these two images, such that the negative values are darker than the mid-gray and the positive values are brighter than the mid-gray.

Figure 2 shows the amplitude spectra of the original, lowpass filtered, bandpass filtered, and highpass filtered images in the 2D-DFT domain with DC at the center. For better visualization, apply an intensity transformation of $\ln(1+x)$ to all four amplitude response images. Also define the brightest point in the original image amplitude spectrum as the white point, and normalize all amplitude response values in all 4 amplitude spectra using the white point, so that the white point maps to the brightest value 1 and all other values are between 0 and 1.

In a few sentences, briefly describe your observations of the results you obtained.

Useful MATLAB functions: *imread, figure, subplot, imshow, title, fft2, ifft2, fftshift, abs, log, real, max*

4. Image Deblurring by Frequency Domain Inverse Filter Design

Read the original image ‘text.tif’, which is an 8 bits/pixel gray scale image.

Create a spatial domain Gaussian filter of size 21x21 and with the standard deviation of the Gaussian profile equaling 1 pixel. Normalize the Gaussian filter such that all coefficients sum to 1.

Apply 2D convolution between the original image and the Gaussian filter to create a blurred image. Crop the central part of the result, so that the Gaussian filtered or blurred image has the same size of the original image.

Apply 2D-DFT to the spatial domain Gaussian filter, resulting in a frequency domain Gaussian filter. Create a frequency domain inverse Gaussian filter by taking the reciprocal value of all frequency domain Gaussian filter coefficients. Apply inverse 2D-DFT and take the real-parts to create a spatial domain inverse Gaussian filter (the imaginary parts are theoretically 0, but may not be completely 0 due to numerical errors).

Apply 2D convolution between the spatial domain Gaussian filtered image and the spatial domain inverse Gaussian filter. The result is a spatial domain deblurred image. Crop the central part of the result, so that the deblurred image has the same size of the original image.

Create 2 figures. Figure 1 contains 6 sub-figures of 3x2 layout. In the first row, show the original, blurred and deblurred images. In the second row, show their corresponding Fourier domain magnitude spectra. For better visualization, apply an intensity transformation of ln(1+ x) to all Fourier amplitude, normalize them so that the brightest point maps to value 1, and shift the spectra such that the DC component locates at the center of the spectera.

Figure 2 includes 4 sub-figures of 2x2 layout. In the first row, use the *mesh* function to show the spatial domain Gaussian blur filter its Fourier domain magnitude spectrum, respectively, where DC component of the Fourier magnitude spectrum should locate at the center. In the second row, use the *mesh* function to show the spatial domain inverse Gaussian deblur filter its Fourier domain magnitude spectrum, respectively, where DC component of the Fourier magnitude spectrum should locate at the center.

In a few sentences, briefly describe your observations of the results you obtained.

Useful MATLAB functions: *imread, figure, subplot, imshow, title, linespace, meshgrid, exp, conv2, fft2, ifft2, fftshift, real, abs, log, mesh*

5. Image Interpolation: Nearest Neighbor and Bilinear Interpolation

You will downsample an image, and then reconstruct the image to its original size from the samples using two interpolation methods, namely “nearest neighbor” and “bilinear interpolation”. The first method fill a missing pixel by the intensity value of the sampling point that is closest to the pixel. The second method connects the sampling points with straight lines along one direction (e.g., horizontally), and then along the second direction (e.g., vertically). There are different ways to implement these methods. A convenient way is to first upsample the image by filling zeros at the missing pixels and then convolve the upsampled image with a 1D interpolation function along one direction (e.g., horizontally) and then the other (e.g., vertically).

Read the original image ‘einstein.tif’, which is an 8 bits/pixel gray scale image.

Given a downsampling factor D (assume D is an odd number). Downsample the original image with factor D both horizontally and vertically, where the top-left sample should be at the location of ((D+1)/2, (D+1)/2) in the original image. This will result in a downsampled image of approximately 1/D x 1/D size of the original image.

Upsample the downsampled image to have exactly the size of the original image by filling zeros in the missing pixels. In the resulting upsampled image, the pixel values in the (non-zero) sampling points should match exactly those in the original image.

Convolve each row of the upsampled image with a 1D filter of a block shape and size D. After all row convolutions, convolve each column of the resulting image with the same 1D filter of a block shape and size D. This leads to an interpolated image based on the nearest neighbor method.

Convolve each row of the upsampled image with a 1D filter of triangular shape and size 2D-1, where the height of the central point of the triangular shape equals 1, and the height decays at a speed of 1/D per pixel on both sides. After all row convolutions, convolve each column of the resulting image with the same 1D triangular filter. This leads to an interpolated image based on the bilinear interpolation method.

Repeat all processes above for D=3 and D=7 cases.

Create 2 figures for D=3 and D=7 cases, respectively. Each figure contains 4 sub-figures of 2x2 layout, and shows the original image, upsampled image, interpolated image using nearest neighbor method, and interpolated image using bilinear interpolation, respectively.

In a few sentences, briefly describe your observations of the results you obtained.

Useful MATLAB functions: *imread, figure, subplot, imshow, title, conv*