The Hong Kong University of Science and Technology
Department of Computer Science and Engineering
COMP4421 (Fall 2018)

Assignment 1

Total = 100 marks
Due: 11:55pm, Oct. 12, 2018
Assignments must be submitted via Canvas
Late Policy: 10% reduction; only one day late is allowed, i.e., 11:55pm, Oct. 13.
Bonus Policy: Bonus points cannot be transferred to the next assignments and the total score will not exceed 100 marks.

Overview

Topics: Image Enhancement in the Spatial and Frequency Domain

In this assignment, you need to finish two parts. The first one is to answer a simple question, while the second part requires certain programming works.
Please submit your answers of the first part in PDF format, and all relevant codes and results of your second part together in a folder. You also need to **paste your result images of the second part in the PDF file**. **Please follow the structure of "Sample" folder to place your codes and result images (otherwise marks will be deducted)**. Then you should rename the "Sample" folder by your student ID and the programming language you use and compress it into a zip file. For example, if your student ID is "12345678" and you choose to use Matlab, then your folder should be "12345678_matlab".

This assignment should be submitted via the Canvas system on or before the due date.

1. Exercises

Answer the following question.

1.1 Fourier Transform (10%)

The Fourier spectrum (Fourier Representation) in Fig.(b) corresponds to the original image Fig.(a) and the Fourier spectrum in Fig.(d) was obtained after the image Fig.(a) was padded with zeros (Shown in the Fig.(c)).

a) Explain the significant increase in signal strength along the vertical and horizontal axes of Fig.(d) compared with Fig.(b).
b) Explain the significant increase in signal strength in the low frequency region of Fig.(d) compared with Fig.(b).

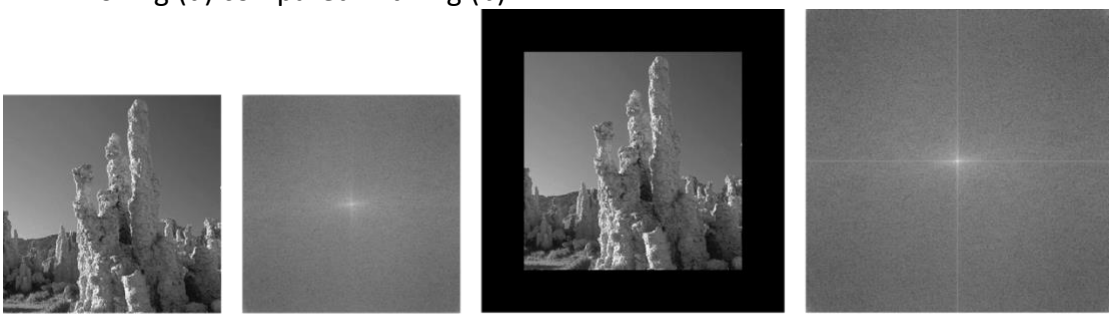


Figure: Fourier Transform

2. Programming Tasks

Write programs to finish the following tasks.

2.1 Pre-requirement

2.1.1 Input: Please download “Assignment_1.zip”, unzip it and choose the image corresponding to the last two digits of your student ID as your input image. Make sure that you have selected the correct image, otherwise your result images will obtain no scores.

2.1.2 Language: Matlab/Octave/**Python 3.6**. For the students who choose to use Python, you need to transform our provided simple MAIN function into Python so that we can run your code.

2.1.3 Functions to be used: You can use build-in functions of Matlab/Octave/Python for handling images, but you should finish your programming tasks with your own codes. For example, you can use “imread” to load an image, “imwrite” to save your results, but functions for spatial filtering like “conv2”, “imfilter”, “fspecial”, “medfilt2” or so forth are not allowed to use.

2.2 Spatial Linear Filtering (15%)

Write a function that performs spatial filtering on a gray level image. The function prototype should be “filter_spa(img_input, filter) -> img_result”, where “filter” is the required filter, for example, “filter” can be [1,1,1;0,0,0;-1,-1,-1]. The function should be able to:

2.2.1 Smooth an image with different sizes of averaging filters, for example, 3×3, 5×5, 11×11 and so on. Please upload your smoothing image with 5×5 averaging filter, named “img_ave.png”. (5%)

2.2.2 Compute x-gradient, y-gradient of an image with Sobel operator. Please upload your x-gradient image and y-gradient image, named “img_dx.png” and “img_dy.png” respectively. (5%)

2.2.3 Sharpen an image with a 3×3 Laplacian filter. As for the Laplacian filters, you can choose any one of the four variants, as shown in the Reference. Please upload your result image, named “img_sharpen.png”. (5%)

Bonus (10%) The naïve approach to convolute an image with a filter mask is to create two loops to retrieve each pixel and do the convolution. Please try to use at most one loop to replace the original two-loop image retrieval. You can take the LBP code (lbp.m) in Canvas as a reference for one-loop image retrieval.

2.3 Spatial Non-linear Filtering (10%)

Write a function that performs median filtering on a gray level image. The function prototype should be “medfilt2d(img_input, size) -> img_result”, where “size” is the window size of median filter, for example, “size” can be 3, 5, and so on. Please use your function to finish the following tasks (you can use the noise generator to generate Gaussian noise or salt-and-pepper noise, the code for noise generator is available in Canvas):

2.3.1 Add Gaussian noise to your input image with mean 0 and standard variance 30. Then try to use your median filter to denoise it. Please upload your noisy image and filtering result, named “img_gaussian.png” and “med_gaussian.png” respectively. (5%)

2.3.2 Add salt-and-pepper noise to your input image with the probabilities of the two noise components 0.3. Then try to use your median filter to denoise it. Please upload your noisy image and filtering result, named “img_sp.png” and “med_sp.png” respectively. (5%)

2.4 Discrete Fourier Transform (20%)

Write a function that performs 2-D Discrete Fourier Transform (DFT) or Inverse Discrete Fourier Transform (IDFT). The function prototype is "dft_2d(img_input, flag) -> img_result", returning the DFT / IDFT result of the given input. "flags" is a parameter that specifies when the function should perform DFT (flag == 'DFT') and when it should perform IDFT (flag == 'IDFT'). Please use your function to finish the following sub-tasks:

2.4.1 Perform DFT and compute the Fourier spectrum. Note that you should shift the values such that F(0,0) is at F(M/2, N/2), where M is the height of the input image and N is the width of the input image. Please upload the resulting spectrum image, named 'dft_spectrum.png'. (10%)

2.4.2 Perform IDFT on the DFT result in 2.4.1. Note that the result of IDFT is a complex-value matrix and you need to transform it into a real-value matrix, for example, by computing the absolute values or simply collect the real part. (10%)

Hints:

1) For better visualization of the Fourier spectrum, we can apply the function $f(t) = \log(t+1)$ on the computed spectrum;
2) To shift the coordinate origin, we can multiply the input image $f(x,y)$ by $(-1)^{x+y}$. Note: **you need to include this part in your own DFT/IDFT function**;
3) The formulas for DFT and IDFT are as follows,

$$DFT: F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

$$IDFT: f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

2.5 Filtering in the Frequency Domain (10%)

Write a function that performs filtering in the frequency domain. The function prototype is "filter2d_freq(img_input, filter) -> img_result", where "filter" is the required filter. According to Slide 23 in the Lecture 03: image-enhancement-freq, convolution/filtering in the spatial domain is equivalent to element-by-element multiplication in the frequency domain. Thus in this task, you are required to apply DFT to the given image and the given filter, and then multiply them element by element, followed by IDFT to get the filtered result. Hence, it should be easy to implement "filter2d_freq" based on "dft_2d". Use your "filter2d_freq" to finish the following sub-tasks:

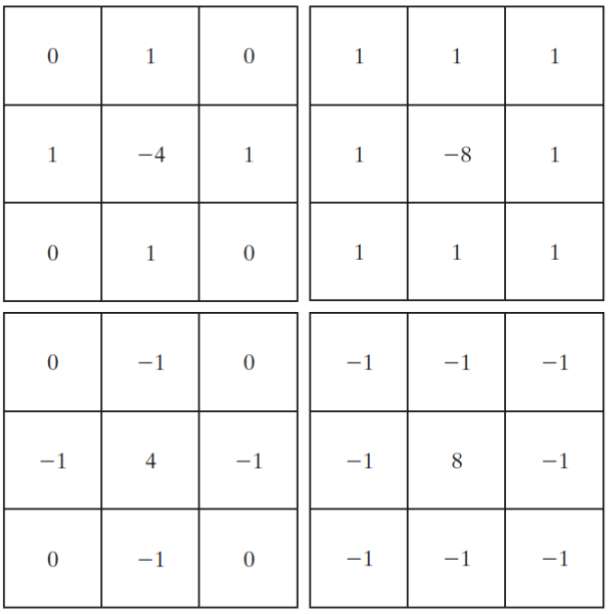
2.5.1 Smooth your input image with 5×5 averaging filter, named "img_ave_freq.png". (5%)

2.5.2 Sharpen an image with a 3×3 Laplacian filter. As for the Laplacian filters, you can choose any one of the four variants, as shown in the Reference. Please upload your result image, named "img_sharpen_freq.png". (5%)

2.6 High-Frequency Emphasis (35%)

Write a function that performs high-frequency emphasis in frequency domain. The function prototype should be "high_freq_emphasis(img_input, a, b, type) -> img_result", where "a", "b" are the parameters in the high-frequency Emphasis formulation and "type" indicates the filter used in frequency domain whose values are from {"butterworth", "gaussian"}. You need to complete the high-pass Butterworth filter and high-pass Gaussian filter in butterworth.m and gaussian.m respectively. The function prototypes are "butterworth(size, cutoff, n) -> f" and "gaussian(size, cutoff) -> f", where "size" is a 1×2 matrix indicating the size of the filter. Let $cutoff = 0.1$, which is the normalized cutoff frequency, and $n = 1$. Please upload your result images filtered by Butterworth and Gaussian, named "butter_emphasis_a_b.png" and "gaussian_emphasis_a_b.png" respectively, where "a" and "b" are the actual values you used.

3. Reference



0	1	0	1	1	1
1	−4	1	1	−8	1
0	1	0	1	1	1
0	−1	0	−1	−1	−1
−1	4	−1	−1	8	−1
0	−1	0	−1	−1	−1

a b
c d

FIGURE 3.37
(a) Filter mask used to implement Eq. (3.6-6). (b) Mask used to implement an extension of this equation that includes the diagonal terms. (c) and (d) Two other implementations of the Laplacian found frequently in practice.