Assignment 5: CS 663

Due: 29th October before 11:55 pm

Remember the honor code while submitting this (and every other) assignment. You may discuss broad ideas with other students or ask me for any difficulties, but the code you implement and the answers you write must be your own. We will adopt a zero-tolerance policy against any violation.

Submission instructions: Follow the instructions for the submission format and the naming convention of your files from the submission guidelines file in the homework folder. Please see assignment5_DFT.rar. Upload the file on moodle before 11:55 pm on 29th October. Only one student per group needs to upload the assignment. We will not penalize submission of the files till 7 am on 30th October. No late assignments will be accepted after this time. Please preserve a copy of all your work until the end of the semester.

1. Suppose you are standing in a well-illuminated room with a large window, and you take a picture of the scene outside. The window undesirably acts as a semi-reflecting surface, and hence the picture will contain a reflection of the scene inside the room, besides the scene outside. While solutions exist for separating the two components from a single picture, here you will look at a simpler-to-solve version of this problem where you would take two pictures. The first picture g_1 is taken by adjusting your camera lens so that the scene outside (f_1) is in focus (we will assume that the scene outside has negligible depth variation when compared to the distance from the camera, and so it makes sense to say that the entire scene outside is in focus), and the reflection off the window surface (f_2) will now be defocussed or blurred. This can be written as $g_1 = f_1 + h_2 * f_2$ where h_2 stands for the blur kernel that acted on f_2. The second picture g_2 is taken by focusing the camera onto the surface of the window, with the scene outside being defocussed. This can be written as $g_2 = h_1 * f_1 + f_2$ where h_1 is the blur kernel acting on f_1. Given g_1 and g_2, and assuming h_1 and h_2 are known, your task is to derive a formula to determine f_1 and f_2. Note that we are making the simplifying assumption that there was no relative motion between the camera and the scene outside while the two pictures were being acquired, and that there were no changes whatsoever to the scene outside or inside. Even with all these assumptions, you will notice something inherently problematic about the formula you will derive. What is it? [8+7 = 15 points]
   ANSWER:
   See model code ReflectionSeparation.m in the solutions folder, though no code was expected from you for this question. Taking Fourier Transforms, we have $G_1(\mu) = F_1(\mu) + H_2(\mu)F_2(\mu)$ and $G_2(\mu) = H_1(\mu)F_1(\mu) + F_2(\mu)$. Solving these equations simultaneously for $F_1(\mu)$ and $F_2(\mu)$, we get $F_2(\mu) = \frac{G_2(\mu)-H_1(\mu)G_1(\mu)}{1-H_1(\mu)H_2(\mu)}$ and $F_1(\mu) = \frac{G_1(\mu)-H_2(\mu)G_2(\mu)}{1-H_1(\mu)H_2(\mu)}$. This solution is well-defined for all values of μ, except if $H_1(\mu)H_2(\mu) = 1$. Now, remember that h_1 and h_2 are low-pass filter kernels, due to the defocussing of the image. These blur kernels will always integrate to 1, i.e. $\int_{-\infty}^{+\infty} h_1(x)dx = 1$ or $\int_a^b h_1(x)dx = 1$ for a blur kernel defined on the interval $[a, b]$. Therefore, we can conclude that $H_1(0) = H_2(0) = 1$ (why? Look at the Fourier transform formula and plug in $\mu = 0$). For all other values of μ, we will have $|H_1(\mu)| \leq 1$ and $|H_2(\mu)| \leq 1$ as this is a low pass filter. This therefore means, that our solution is undefined for those low frequencies, where $H_1(\mu)H_2(\mu) = 1$ (such as $\mu = 0$). Here is a (somewhat rare) situation where reconstruction of higher frequency components is fine, but where lower frequencies (especially the DC component) cannot be recovered robustly. In the case of image denoising, the situation is exactly opposite - the lower frequencies are easy to reconstruct, and hence smooth regions look fine. But the higher frequencies are difficult to reconstruct and hence the loss of finer edges and textures in image denoising. In practice, we would need to add in a small ϵ to the denominators, i.e. we would have $F_2(\mu) = \frac{G_2(\mu)-H_1(\mu)G_1(\mu)}{1-H_1(\mu)H_2(\mu)+\epsilon}$ and $F_1(\mu) = \frac{G_1(\mu)-H_2(\mu)G_2(\mu)}{1-H_1(\mu)H_2(\mu)+\epsilon}$. The resultant image would look somewhat artificial due to incorrect reconstruction of lower frequencies, such as what you see below in Figure 1.

   Now, if there is image noise, i.e. $g_1 = f_1 + h_2 * f_2 + \eta_1$ and $g_2 = h_1 * f_1 + f_2 + \eta_2$, we incur errors proportional to $\frac{N_1(\mu)-H2(\mu)N_2(\mu)}{1-H_1(\mu)H_2(\mu)}$ and $\frac{N_2(\mu)-H_1(\mu)N_1(\mu)}{1-H_1(\mu)H_2(\mu)}$. For higher frequencies, the denominator is large (i.e. close to 1)

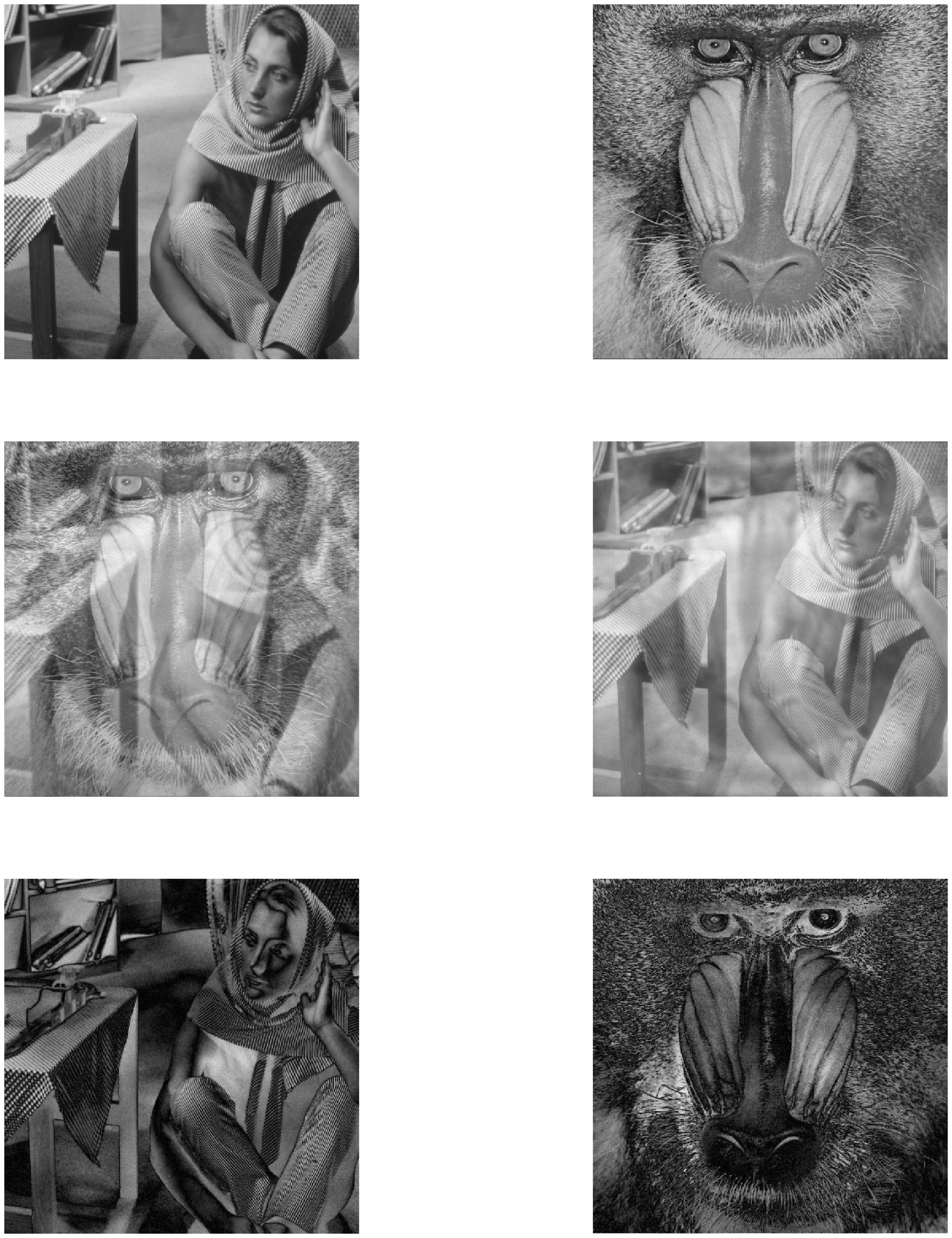

Figure 1: Left to right, top to bottom: barbara, mandrill, mixture 1, mixture 2, reconstructed barbara and reconstructed mandrill. Notice that the features of barbara and mandrill are correctly reconstructed. But the image looks as if it were sent through a high pass filter - which is due to error in the reconstruction of the DC component.

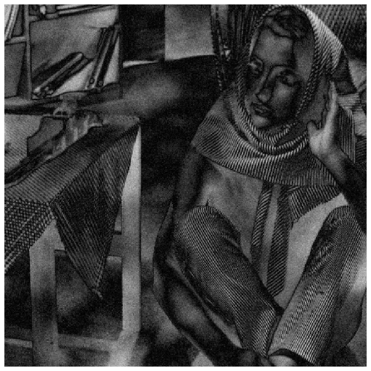
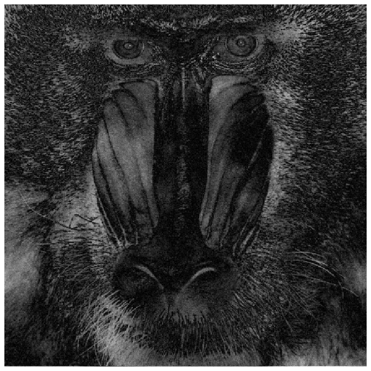

Figure 2: Left to right: reconstructed barbara and reconstructed mandrill when the mixtures had noise. Notice that the features of barbara and mandrill are correctly reconstructed. The reconstructions are not totally different from the earlier results despite the noise. The errors in the DC component, however, remain as before.

and hence there is no amplification of the noise, unlike the case with the inverse filter under noise. For lower frequencies, the noise will get amplified especially as $H_1(\mu)H_2(\mu)$ gets closer to 1, but the signal strength is also very high in these frequencies, so the relative error will not totally blow off, unlike the case with the inverse low-pass filter under noise. See below a reconstruction result under noise in Figure 2.

Also take note that the blurring of the images was actually useful over here (somewhat counter-intuitively). Without the blur, we would simply have two identical images and the separation would have been impossible.

MARKING SCHEME: 8 points for correct derivation of formula. 7 points for pointing out what is the problem with the solution. While describing the problem with this approach, you must argue that $H_1(0) = H_2(0) = 1$ and hence the lower frequency components (particularly the DC component) are not properly reconstructed. Merely saying that there is a division by 0 when $H_1(\mu) = H_2(\mu) = 1$ will fetch you only 4 points out of 7.

2. Consider a 1D image (for example, a single row from a 2D image). You know that given such an image, computing its gradients is trivial. An inquisitive student frames this as a convolution problem to yield $g = h * f$ where g is the gradient image (in 1D), h is the convolution kernel to represent the gradient operation, and f is the original 1D image. The student tries to develop a method to determine f given g and h. What are the fundamental difficulties he/she will face in this task? Justify your answer. You may assume appropriate boundary conditions. Now consider that you are given the gradients of a 2D image in the X and Y directions, and you wish to determine the original image. What are the difficulties you will face in this task? Justify your answer. Again, you may assume appropriate boundary conditions. [10+10 = 20 points]
ANSWER:
The convolution kernel can be typically given as $[-1, 1]$, yielding $g(x) = f(x+1) - f(x)$ for $1 \leq x \leq N$. We know that the DFT of g is given by $G(u) = (e^{\frac{j2\pi u}{N}} - 1)F(u)$, which yields $F(u) = \dfrac{G(u)}{e^{\frac{j2\pi u}{N}} - 1}$. The problem with using this formula to estimate $F(u)$ is that for $u = 0$ we have zero in the denominator. This leads to problems in estimating the DC component of F. So you have to make some assumption about what you expect the DC component to be. This is equivalent to assuming a boundary condition for the gradients, i.e. if f has N elements, we will assume $f(N+1) = 0$. An alternative method is to assume a boundary condition and simply integrate the image from the rightmost pixel leftwards.
Given a 2D image, it is trivial to compute the gradients. Given the gradients in the X and Y directions, it is not easy to get back the image. The DFT of I_x is $\hat{F_x}(u,v) = (e^{\frac{j2\pi u}{N}} - 1)\hat{F}(u,v)$ where $F(u,v)$ is the FT of image f. To estimate $F(u,v)$ from $\hat{F_x}(u,v)$, the problem again occurs for the frequency components with $u = 0$. Similarly, the DFT of f_y is $\hat{F_y}(u,v) = (e^{\frac{j2\pi v}{N}} - 1)F(u,v)$. To estimate $F(u,v)$ from $\hat{F_y}(u,v)$, the problem occurs for the frequency components with $v = 0$. If you knew both $\hat{F_y}(u,v)$ and $\hat{F_x}(u,v)$, the problem still occurs for the component $u = v = 0$, which is the DC component. Again you need to make some assumptions on what this DC component is. If you use the integration based method using boundary conditions, there may

be inconsistent answers when you use f_x and f_y.

(Note: Here we have assumed that we had accurate estimates of the image gradients f_x and f_y. But sometimes, you may get noisy estimates of the image gradients either as the output of some computational procedure, or else (rarely) using gradient cameras such as those in `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.415.6700&rep=rep1&type=pdf` (the title of the article is 'Why I want a gradient camera?'.) This makes this problem much more complicated than what we have seen so far. Want to figure out what that is?).

Marking scheme: 10 points for the first part with the 1D image for either the Fourier based solution or the simple computational solution using digital integration (i.e. addition). In each case, the problem faced must be properly outlined. 10 points for the second part - where the analysis must show that the problem now exists for $u = v = 0$ only. For the second part, if you write a solution using integration across rows or columns, you lose 6 points if you don't mention that the solution using integration across rows will be inconsistent with the solution using integration across columns.

3. Consider the image with the low frequency noise pattern shared in the homework folder in the form of a .mat file. Your task is to (a) write MATLAB code to display the log magnitude of its Fourier transform, (b) to determine the frequency of the noise pattern by observing the log magnitude of the Fourier transform and guessing the interfering frequencies, and (c) to design and implement (in MATLAB) an ideal notch filter to remove the interference(s) and display the restored image. To this end, you may use the fft2, ifft2, fftshift and ifftshift routines in MATLAB. [15 points]
ANSWER: The code is in the file notchfilter.m in the homework folder. The interfering pattern has the equation $Z = 50\sin(X/8+2*Y/8)$, leading to some unnatural peaks in the frequency domain at two frequencies. These peaks can be determined by inspection and removed using an ideal band-reject filter. (A Gaussian band-reject filter could also have been used).
MARKING SCHEME: 8 points for identifying the peaks and 7 points for implementing the notch filter.

4. Consider the barbara256.png image from the homework folder. Implement the following in MATLAB: (a) an ideal low pass filter with cutoff frequency $D \in \{40, 80\}$, (b) a Gaussian low pass filter with $\sigma \in \{40, 80\}$. Show the effect of these on the image, and display all images in your report. Display the frequency response (in log Fourier format) of all filters in your report as well. Comment on the differences in the outputs. Make sure you perform appropriate zero-padding! [20 points]
ANSWER: The code for the ILPF and GLPF is lpf.m (with appropriate zero-padding) and lpf2.m (without appropriate zero padding).
MARKING SCHEME: 10 points for ILPF and 10 points for GLPF. Deduct 2 points for ILPF and 2 points for GLPF if appropriate zero-padding was not performed. To convolve two signals of size $N_1 \times N_2$ with each other, both need to be zero-padded to form signals of size $(2N_1 - 1) \times (2N_2 - 1)$. Hence the frequency domain filters must be of size $(2N_1 - 1) \times (2N_2 - 1)$ to avoid aliasing.

5. Read Section 1 of the paper 'An FFT-Based Technique for Translation, Rotation, and Scale-Invariant Image Registration' published in the IEEE Transactions on Image Processing in August 1996. A copy of this paper is available in the homework folder. Implement the technique in Equation 3 of the paper to align two images which are related to each other by a 2D in-plane translation. Test your implementation on images I and J as follows. I is a 300×300 image containing a 50×70 white rectangle (intensity 255) whose top-left corner lies at pixel $(50, 50)$. All other pixels of I have intensity 0. The image J is obtained from a translation of I by values $(t_x = -30, t_y = 70)$. Verify carefully that the predicted translation agrees with the ground truth translation values. Repeat the exercise if I and J were treated with iid Gaussian noise with mean 0 and standard deviation 20. In both cases, display the logarithm of the Fourier magnitude of the cross-power spectrum in Equation 3 of the paper. What is the time complexity of this procedure to predict translation if the images were of size $N \times N$? How does it compare with a pixel-wise image comparison procedure for predicting the translation? Also, briefly explain the approach for correcting for rotation between two images, as proposed in this paper in Section II. Write down an equation or two to illustrate your point. [8+7=15 points]
ANSWER: The inverse Fourier transform of the cross-spectrum should peak at the correct value of the translation, which in this case is $(270, 70)$, since $300 - 30 = 270$. In MATLAB, the peak will be at $(271, 71)$. The peak is preserved even under addition of noise, though the inverse Fourier transform now contains a great deal of noisy artifacts.

The time complexity of this procedure is $O(N^2 \log N)$ for an $N \times N$ image. A pixel-wise translation prediction will have time complexity $O(N^2 W^2)$ wheer $W \times W$ is the window size for the range of translations.
Marking scheme: 3 points for time complexity, 4 points for code implementation and 3 points for correct prediction of the translation (with justification - you need to check that the translation is indeed predicted by the code). 5 points for a correct description of the rotation correction procedure with equations.

6. Consider two different Laplacian filter kernels $k_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$ and $k_2 = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$. Write down the formulae for their N, N-point Discrete Fourier Transforms in the report. Compute their N, N-point Discrete Fourier Transforms in MATLAB with $N = 201$, i.e. where the spatial and frequency indices range from -100 to 100 in both canonical directions. Display the magnitude of the DFT on a log scale using the `imshow` and `surf` functions in MATLAB along with a colorbar. Besides the plots, include the code snippet for the DFT computation and display in the report. Comment on the difference in the Fourier transforms of the two kernels. [10+5=15 points]
Solution: Let $g_1(x,y) = (f * k_1)(x,y)$ and $g_2(x,y) = (f * k_2)(x,y)$. We have $g_1(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$ and $g_2(x,y) = 8f(x,y) - [f(x+1,y+1) + f(x-1,y-1) + f(x-1,y+1) + f(x+1,y-1) + f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)]$.

Taking Fourier transforms, we have $G_1(u,v) = F(u,v)\Big[e^{j2\pi u/N} + e^{-j2\pi u/N} + e^{j2\pi v/N} + e^{-j2\pi v/N} - 4\Big]$. Thus $K_1(u,v) = \Big[e^{j2\pi u/N} + e^{-j2\pi u/N} + e^{j2\pi v/N} + e^{-j2\pi v/N} - 4\Big]$.

Likewise, we have $G_2(u,v) = F(u,v)\Bigg(8 - \Big(e^{j2\pi(u+v)/N} + e^{-j2\pi(u+v)/N} + e^{j2\pi(u-v)/N} + e^{j2\pi(v-u)/N} + e^{j2\pi u/N} + e^{-j2\pi u/N} + e^{j2\pi v/N} + e^{-j2\pi v/N}\Big)\Bigg)$. Thus $K_2(u,v) = \Bigg(8 - \Big(e^{j2\pi(u+v)/N} + e^{-j2\pi(u+v)/N} + e^{j2\pi(u-v)/N} + e^{j2\pi(v-u)/N} + e^{j2\pi u/N} + e^{-j2\pi u/N} + e^{j2\pi v/N} + e^{-j2\pi v/N}\Big)\Bigg)$.

From the plots in `laplace_filter.m`, one can see that k_2 boosts a wider range of high frequencies as compared to k_1. This is evident from the image as well as surface plots.
Marking scheme: 5 points each for the fourier transform of k_1 and k_2. 5 points for the difference in their behaviour in boosting higher frequencies.