Assignment 5: CS 663

Due: 26th 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 `http://www.cse.iitb.ac.in/~suyash/cs663/submissionStyle.pdf`. However, please do not submit the face image databases in your zip file that you will upload on moodle. Please see `http://www.cse.iitb.ac.in/~ajitvr/CS663_Fall2017/HW5/assignment5_DFT.rar`. Upload the file on moodle before 11:55 pm on 26th October. Policy for late submissions will be the same as in the aforementioned guidelines document. 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? [7+8 = 15 points]

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. [5+10 = 15 points]

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. [10 points]

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! [15 points]

5. Consider a set of N vectors $\mathcal{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_N}\}$ each in $\mathbb{R}^d$ ($N > d$). Assume their mean vector is $\mathbf{0}$. Let $\mathbf{V} \in \mathbb{R}^{d \times d}$ be the orthonormal matrix containing the principal components of this dataset arranged in descending order of the eigenvalues (assume all eigenvalues are distinct). Let us denote the order k ($k < d$) linear approximation of vector $\boldsymbol{x_i}$ using $\mathbf{V}$ as $L(\boldsymbol{x_i}^{(k)}; \mathbf{V}) = \boldsymbol{V_k}\boldsymbol{\alpha_i^{(k)}}$ where $\boldsymbol{V_k}$ is a $d \times k$ matrix containing the first k columns of $\mathbf{V}$, and $\boldsymbol{\alpha_i^{(k)}} = \boldsymbol{V_k}^t\boldsymbol{x_i}$. Let us denote the order k ($k < d$) non-linear approximation of vector $\boldsymbol{x_i}$ using $\mathbf{V}$ as $N(\boldsymbol{x_i}^{(k)}; \mathbf{V}) = \mathbf{V}\boldsymbol{\alpha_i}$ where $\boldsymbol{\alpha_i} = \arg\min_{\boldsymbol{c_i}} \|\boldsymbol{x_i} - \mathbf{V}\boldsymbol{c_i}\|^2$ subject to the constraint that vector $\boldsymbol{c_i}$ has at the most k non-zero elements. The total reconstruction errors for the linear and non-linear approximations are respectively $E_L(\mathbf{V}) = \sum_{i=1}^N \|\boldsymbol{x_i} - L(\boldsymbol{x_i}^{(k)}; \mathbf{V})\|^2$ and $E_N(\mathbf{V}) = \sum_{i=1}^N \|\boldsymbol{x_i} - N(\boldsymbol{x_i}^{(k)}; \mathbf{V})\|^2$. Which of the following statements is true and <u>why</u>:

   (a) $E_L(\mathbf{V}) \leq E_N(\mathbf{V})$

   (b) $E_L(\mathbf{V}) \geq E_N(\mathbf{V})$

   (c) $E_L(\mathbf{V}) = E_N(\mathbf{V})$

   (d) One cannot make a conclusion about which error is greater.

   Also devise an efficient algorithm to obtain the order k non-linear approximation of $\boldsymbol{x_i}$ given $\mathbf{V}$, and state its time complexity. Argue <u>why</u> your algorithm is correct. [5+5+5 = 15 points]

6. In this part, we will apply the PCA technique for the task of image denoising. Take the barbara256.png image present in the corresponding data/ subfolder - this image has gray-levels in the range from 0 to 255. Add zero mean Gaussian noise of $\sigma = 20$ to it using the MATLAB code 'im1 = im + randn(size(im))*20'. (Do not clamp the values in im1 to the [0,255] range as that alters the noise statistics). Note that this noise is image-independent. If during the course of your implementation, your program takes too long, you can instead work with the file barbara256-part.png which has size 128 by 128 instead of 256 by 256.

   (a) In the first part, you will divide the entire noisy image im1 into overlapping patches of size 7 by 7, and create a matrix $\mathbf{P}$ of size $49 \times N$ where N is the total number of image patches. Each column of $\mathbf{P}$ is a single patch reshaped to form a vector. Compute eigenvectors of the matrix $\mathbf{P}\mathbf{P}^T$, and the eigen-coefficients of each noisy patch. Let us denote the j^{th} eigen-coefficient of the i^{th} (noisy) patch (i.e. $\mathbf{P}_i$) by α_{ij}. Define $\bar{\alpha}_j^2 = \max(0, \frac{1}{N}[\sum_{i=1}^N \alpha_{ij}^2] - \sigma^2)$, which is basically an estimate of the average squared eigen-coefficients of the 'original (clean) patches'. Now, your task is to manipulate the noisy coefficients $\{\alpha_{ij}\}$ using the following rule, which is along the lines of the Wiener filter update that we studied in class: $\alpha_{ij}^{\text{denoised}} = \dfrac{\alpha_{ij}}{1 + \frac{\sigma^2}{\bar{\alpha}_j^2}}$. Here, $\alpha_{ij}^{\text{denoised}}$ stands for the j^{th} eigencoefficient of the i^{th} denoised patch. Note that $\frac{\sigma^2}{\bar{\alpha}_j^2}$ is an estimate of the ISNR, which we absolutely need for any practical implementation of a Wiener filter update. After updating the coefficients by the Wiener filter rule, you should reconstruct the denoised patches and re-assemble them to produce the final denoised image which we will call 'im2'. Since you chose overlapping patches, there will be multiple values that appear at any pixel. You take care of this situation using simple averaging. Write a function `myPCADenoising1.m` to implement this. Display the final image 'im2' in your report and state its mean squared error with respect to 'im'.

   (b) In the second part, you will modify this technique. Given any patch $\mathbf{P}_i$ in the noisy image, you should collect $K = 200$ most similar patches (in a mean-squared error sense) from within a 31×31 neighborhood centered at the top left corner of $\mathbf{P}_i$. We will call this set of similar patches as Q_i (this set will of course include $\mathbf{P}_i$). Build an eigen-space given Q_i and denoise the eigen-coefficients corresponding to **only** P_i using the Wiener update mentioned earlier. The only change will be that $\bar{\alpha}_j^2$ will now be defined using only the patches from Q_i (as opposed to patches from all over the image). Reconstruct the denoised version of P_i. Repeat this for every patch from the noisy image (i.e. create a fresh eigen-space each time). At any pixel, there will be multiple values due to overlapping patches - simply average them. Write a function `myPCADenoising2.m` to implement this. Reconstruct the final denoised image, display it in your report and state the mean squared error value.

(c) Now run your bilateral filter code `myBilateralFiltering.m` from Homework 2 on the noisy version of the barbara image. Compare the denoised result with the result of the previous two steps. What differences do you observe? What are the differences between this PCA based approach and the bilateral filter? [10 + 10 + 10 = 30 points]