Assignment 5: CS 663

Due: 5th November 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 5th November **and this is also the cutoff deadline. This is different from other assignments where you had time till the next day 10 am. But 5th November is the last day of classes**. Only one student per group needs to upload the assignment. No late assignments will be accepted after this time. Please preserve a copy of all your work until the end of the semester.

1. In this part, we will apply the PCA technique for the task of image denoising. Consider the images `barbara256.png` and `stream.png` present in the corresponding data/ subfolder - this image has gray-levels in the range from 0 to 255. For the latter image, you should extract the top-left portion of size 256 by 256. Add zero mean Gaussian noise of $\sigma = 20$ to one of these images using the MATLAB code `im1 = im + randn(size(im))*20`. 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. You can likewise extract the top-left 128 by 128 part of the `stream.png` image. You will not be penalized for working on these image parts.)

   (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 RMSE computed as $\dfrac{\|\text{im2}_{\text{denoised}} - \text{im2}_{\text{orig}}\|_2}{\|\text{im2}_{\text{orig}}\|_2}$.

   (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 RMSE value. *Do so for both barbara as well as stream.*

(c) Now run your bilateral filter code from Homework 2 on the noisy version of the barbara image. Compare the denoised result with the result of the previous two steps for both images. What differences do you observe? What are the differences between this PCA based approach and the bilateral filter?

(d) Consider that a student clamps the values in the noisy image'im1' to the [0,255] range, and then denoises it using the aforementioned PCA-based filtering technique which assumes Gaussian noise. Is this approach correct? Why (not)? [10 + 20 + 5 + 5 = 40 points]

2. 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.

   (a) Describe the procedure in the paper to determine translation between two given images. 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 the time complexity of pixel-wise image comparison procedure for predicting the translation?

   (b) 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.

   [15+15=30 points]

3. Consider a matrix $\boldsymbol{A}$ of size $m \times n, m \leq n$. Define $\boldsymbol{P} = \boldsymbol{A}^T\boldsymbol{A}$ and $\boldsymbol{Q} = \boldsymbol{A}\boldsymbol{A}^T$. (Note: all matrices, vectors and scalars involved in this question are real-valued).

   (a) Prove that for any vector $\boldsymbol{y}$ with appropriate number of elements, we have $\boldsymbol{y}^t\boldsymbol{P}\boldsymbol{y} \geq 0$. Similarly show that $\boldsymbol{z}^t\boldsymbol{Q}\boldsymbol{z} \geq 0$ for a vector $\boldsymbol{z}$ with appropriate number of elements. Why are the eigenvalues of $\boldsymbol{P}$ and $\boldsymbol{Q}$ non-negative?

   (b) If $\boldsymbol{u}$ is an eigenvector of $\boldsymbol{P}$ with eigenvalue λ, show that $\boldsymbol{A}\boldsymbol{u}$ is an eigenvector of $\boldsymbol{Q}$ with eigenvalue λ. If $\boldsymbol{v}$ is an eigenvector of $\boldsymbol{Q}$ with eigenvalue μ, show that $\boldsymbol{A}^T\boldsymbol{v}$ is an eigenvector of $\boldsymbol{P}$ with eigenvalue μ. What will be the number of elements in $\boldsymbol{u}$ and $\boldsymbol{v}$?

   (c) If $\boldsymbol{v}_i$ is an eigenvector of $\boldsymbol{Q}$ and we define $\boldsymbol{u}_i \triangleq \dfrac{\boldsymbol{A}^T\boldsymbol{v}_i}{\|\boldsymbol{A}^T\boldsymbol{v}_i\|_2}$. Then prove that there will exist some real, non-negative γ_i such that $\boldsymbol{A}\boldsymbol{u}_i = \gamma_i\boldsymbol{v}_i$.

   (d) It can be shown that $\boldsymbol{u}_i^T\boldsymbol{u}_j = 0$ for $i \neq j$ and likewise $\boldsymbol{v}_i^T\boldsymbol{v}_j = 0$ for $i \neq j$ for correspondingly distinct eigenvalues.[1]. Now, define $\boldsymbol{U} = [\boldsymbol{v}_1|\boldsymbol{v}_2|\boldsymbol{v}_3|...|\boldsymbol{v}_m]$ and $\boldsymbol{V} = [\boldsymbol{u}_1|\boldsymbol{u}_2|\boldsymbol{u}_3|...|\boldsymbol{u}_m]$. Now show that $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Gamma}\boldsymbol{V}^T$ where $\boldsymbol{\Gamma}$ is a diagonal matrix containing the non-negative values $\gamma_1, \gamma_2, ..., \gamma_m$. With this, you have just established the existence of the singular value decomposition of any matrix $\boldsymbol{A}$. This is a key result in linear algebra and it is widely used in image processing, computer vision, computer graphics, statistics, machine learning, numerical analysis, natural language processing and data mining. [7.5 + 7.5 + 7.5 + 7.5 = 30 points]

[1]This follows because $\boldsymbol{P}$ and $\boldsymbol{Q}$ are symmetric matrices. Consider $\boldsymbol{P}\boldsymbol{u}_1 = \lambda_1\boldsymbol{u}_1$ and $\boldsymbol{P}\boldsymbol{u}_2 = \lambda_2\boldsymbol{u}_2$. Then $\boldsymbol{u}_2^T\boldsymbol{P}\boldsymbol{u}_1 = \lambda_1\boldsymbol{u}_2^T\boldsymbol{u}_1$. But $\boldsymbol{u}_2^T\boldsymbol{P}\boldsymbol{u}_1$ also equal to $(\boldsymbol{P}^T\boldsymbol{u}_2)^T\boldsymbol{u}_1 = (\boldsymbol{P}\boldsymbol{u}_2)^T\boldsymbol{u}_1 = (\lambda_2\boldsymbol{u}_2)^T\boldsymbol{u}_1 = \lambda_2\boldsymbol{u}_2^T\boldsymbol{u}_1$. Hence $\lambda_2\boldsymbol{u}_2^T\boldsymbol{u}_1 = \lambda_1\boldsymbol{u}_2^T\boldsymbol{u}_1$. Since $\lambda_2 \neq \lambda_1$, we must have $\boldsymbol{u}_2^T\boldsymbol{u}_1 = 0$.