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]

ANSWER: See model code in the homework solutions folder. The reconstruction results are given below in Figure 1. For part (a), for barbara, the MSE is around 94, and the image looks extremely blurred even though the noise is removed. For part (b), the results are given below for two case $K = 25$ and $K = 200$. The respective MSEs are 190 and 60, and the result with $K = 200$ is far superior. This PCA-based method in part 3(b) is one of the much better denoising methods available today. Compare the results with the output of the bilateral filter to see how much better the texture preservation is.

So what is going on? When you apply a filter, there is an implicit assumption you make about the image. This assumption is often ignored in the literature! If you did a simple mean filter, the assumption is that the underlying image had a constant intensity. If that is violated, i.e. there are edges separating distinct regions, the edges will get blurred away too! The next best is to do a piecewise mean filter, or a weighted piecewise mean filter, which is what the bilteral filter is. The underlying assumption is that the image is piecewise constant. The bilateral filter preserves significant edges well, but obliterates subtle textures. In fact, if you had an image that was piecewise linear instead of piecewise constant (i.e. within a single region, the image intensity was of the form $I(x, y) = ax + by + c$ for some constants a, b and c - this looks like a shading with constant slope), you would lose them depending on the parameters of the bilateral filter.

The PCA filter makes no such assumption - be it piecewise linear or piecewise constant. Instead it assumes that given any patch P_r of small size such as 8×8, there exist patches elsewhere in the image which are structurally very similar to it. This is true for many natural images (see barbara for instance, there is so much repetition of patterns on her hair, her jeans, the tablecloth, the bookshelf and the background on the top right corner!). For noise levels whose standard deviation is much less than the average image intensity, this sort of self-similarity is still preserved even in the noisy image. So the method in part (b) says - why not collect these similar patches, denote them as a group $\{P_i\}_{i=1}^{K}$, and build an eigenspace from them?
For part (d), you need to realize that such a clamping to [0,255] range essentially violates the Gaussian noise assumption. But the PCA based algorithm is based on the assumption of Gaussian noise. Hence the procedure is not kosher, and may produce artifacts. I have observed that it tends to produce lower PSNR values than without clamping.

MARKING SCHEME: 10 points for correct implementation of part (a) and 10 points for part (b). part (c): 5 points for a sensible explanation of the difference between PCA and bilateral filtering. In particular, you should state that bilateral filtering performs local averaging with parameters not immediately tied to noise level whereas PCA is specifically designed for a particular noise model. Part (d): 6 points for proper implementation and 4 points for the NSR argument. Also, for all four parts from (a) to (c): For every RMSE value not mentioned in the report and not directly printed by the code, deduct 0.5 points. For part (d), there are 5 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]
Solution:

Figure 1: Left to right, top to bottom:Original image; noisy image (under zero mean, iid Gaussian noise with $\sigma = 20$); image reconstructed usign global PCA, i.e. part (a); image reconstructed using spatially vaying PCA, i.e. part (b), with $K = 25$; image reconstructed using spatially vaying PCA, i.e. part (b), with $K = 200$; result with bilateral filtering for 20 iterations with $\sigma_{spatial} = 8$ and $\sigma_{intensity} = 8$; result with bilateral filtering for 20 iterations with $\sigma_{spatial} = 8$ and $\sigma_{intensity} = 5$. Second last row: left - Poisson noisy image when intensities were divided by 20, right: denoised image using the PCA based method (RMSE: 0.12). Last row: left - Poisson noisy image when intensities were divided by 1, right: denoised image using the PCA based method (RMSE: 0.038).

The translation is computed as follows: Let $f_2(x, y) = f_1(x - x_0, y - y_0)$ where (x_0, y_0) is the pixel-space shift. Then, we have $F_2(\mu, \nu) = F_1(\mu, \nu)e^{-j2\pi(ux_0+vy_0)/N}$ where images f_2, f_1 have size $N \times N$. The paper proposes to compute the cross power spectrum of the two images given as:

$$C(\mu, \nu) = \frac{F_2^*(\mu, \nu)F_1(\mu, \nu)}{|F_2(\mu, \nu)||F_1(\mu, \nu)|} = e^{-j2\pi(ux_0+vy_0)/N}. \tag{1}$$

The inverse fourier transform of C will yield a peak at (x_0, y_0). The student may also only refer to equations in the paper for the answer.

For the part on rotation, consider equation 4 of the paper and also equation 5 which is obtained by applying the rotation theorem to both sides of equation 4. The value of (x_0, y_0) can be obtained using the cross-power spectrum. To find the angle of rotation, consider equation 6 which is obtained by considering the magnitude of the Fourier transform on both sides of equation 5 and converting Cartesian coordinates to polar coordinates. Rotation in Cartesian coordinates is equivalent to a shift in the angle θ in the polar coordinates for which the cross-power spectrum method can again be used.

For the part on translation, 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 $O(N^2W^2)$ where $W \times W$ is the window size for the range of translations.

marking scheme: Description of procedure to determine translation: 10 points, time complexity for translatio: 5 points. Procedure for rotation: 15 points (application of rotation theorem: 5 points, determining (x_0, y_0): 2.5 points, taking magnitudes: 2.5 points, conversion to polar coordinates: 3 points and 2 points for final application of cross-power spectrum.)

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?
   ANSWER:
   $\boldsymbol{y}^t\boldsymbol{P}\boldsymbol{y} = \boldsymbol{y}^t\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{y} = (\boldsymbol{A}\boldsymbol{y})^t(\boldsymbol{A}\boldsymbol{y})$, which is the squared magnitude of the vector $\boldsymbol{A}\boldsymbol{y}$. Hence $\boldsymbol{y}^t\boldsymbol{P}\boldsymbol{y} \geq 0$. Similarly, $\boldsymbol{z}^t\boldsymbol{Q}\boldsymbol{z} \geq 0$. Now suppose $\boldsymbol{u}$ is an eigenvector of $\boldsymbol{P}$ with eigenvalue λ, then $\boldsymbol{P}\boldsymbol{u} = \lambda\boldsymbol{u}$, and hence $\boldsymbol{u}^t\boldsymbol{P}\boldsymbol{u} = \lambda\boldsymbol{u}^t\boldsymbol{u} = \lambda$. As $\boldsymbol{u}^t\boldsymbol{P}\boldsymbol{u} \geq 0$, we must have $\lambda \geq 0$. Likewise for $\boldsymbol{Q}$. Thus $\boldsymbol{P}$ and $\boldsymbol{Q}$ are positive semi-definite matrices.

   (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 $\boldsymbol{v}$?
   ANSWER:

$$\begin{aligned} \boldsymbol{P}\boldsymbol{u} &= \lambda\boldsymbol{u} \\ \therefore \boldsymbol{A}\boldsymbol{P}\boldsymbol{u} &= \lambda\boldsymbol{A}\boldsymbol{u} \\ \therefore \boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{A})\boldsymbol{u} &= \lambda\boldsymbol{A}\boldsymbol{u} \\ \therefore (\boldsymbol{A}\boldsymbol{A}^T)(\boldsymbol{A}\boldsymbol{u}) &= \lambda(\boldsymbol{A}\boldsymbol{u}) \\ \therefore \boldsymbol{Q}(\boldsymbol{A}\boldsymbol{u}) &= \lambda(\boldsymbol{A}\boldsymbol{u}) \end{aligned} \tag{2}$$

$$\begin{aligned} \boldsymbol{Q}\boldsymbol{v} &= \mu\boldsymbol{v} \\ \therefore \boldsymbol{A}^T\boldsymbol{Q}\boldsymbol{v} &= \mu\boldsymbol{A}^T\boldsymbol{v} \\ \therefore \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{A}^T)\boldsymbol{v} &= \mu\boldsymbol{A}^T\boldsymbol{v} \\ \therefore (\boldsymbol{A}^T\boldsymbol{A})(\boldsymbol{A}^T\boldsymbol{v}) &= \mu(\boldsymbol{A}^T\boldsymbol{v}) \\ \therefore \boldsymbol{P}(\boldsymbol{A}^T\boldsymbol{v}) &= \mu(\boldsymbol{A}^T\boldsymbol{v}) \end{aligned} \tag{3}$$

   Note that $\boldsymbol{u} \in \mathbb{R}^n, \boldsymbol{v} \in \mathbb{R}^m$.

(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 $\boldsymbol{A}\boldsymbol{u}_i = \gamma_i\boldsymbol{v}_i$. **ANSWER:**

$$\begin{aligned}
\boldsymbol{Q}\boldsymbol{v_i} &= \mu_i\boldsymbol{v_i} \\
\therefore \boldsymbol{A}\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{v_i} &= \mu_i\boldsymbol{v_i} \\
\therefore \boldsymbol{A}(\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{v_i}) &= \mu_i\boldsymbol{v_i} \\
\therefore \boldsymbol{A}(\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{v_i})/\|\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{v_i}\|_{\boldsymbol{2}} &= \mu_i\boldsymbol{v_i}/\|\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{v_i}\|_2 \\
\therefore \boldsymbol{A}\boldsymbol{u_i} &= (\mu_i/\|\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{v_i}\|_2)\boldsymbol{v_i} \\
\therefore \boldsymbol{A}\boldsymbol{u_i} &= \gamma_i\boldsymbol{v_i}
\end{aligned} \tag{4}$$

where we define the scalar $\gamma_i = \mu_i/\|\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{v_i}\|_2$. Note that γ_i is non-negative as μ_i is non-negative (see part (a)) and $\|\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{v_i}\|_2$ is non-negative since it is the magnitude of a vector.

(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]

ANSWER: We saw that $\boldsymbol{A}\boldsymbol{u_i} = \gamma_i\boldsymbol{v_i}$. Assume $m \leq n$ without loss of generality. Then we can write $\forall i, 1 \leq i \leq m, \boldsymbol{A}\boldsymbol{u_i} = \gamma_i\boldsymbol{v_i}$ and $\forall i, m+1 \leq i \leq n, \boldsymbol{A}\boldsymbol{u_i} = \boldsymbol{0}$. So we can write $\boldsymbol{A}\boldsymbol{V} = \boldsymbol{U}\boldsymbol{\Gamma}$, where $\boldsymbol{V}$ is an orthonormal matrix of size $n \times n$, $\boldsymbol{\Gamma}$ is a diagonal matrix of size $m \times n$ containing at the most m non-zero values along its diagonal, and $\boldsymbol{U}$ is an orthonormal matrix of size $m \times m$. Hence we can say $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Gamma}\boldsymbol{V}^{\boldsymbol{T}}$. Note that the footnote in the question argues why $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthonormal, and this is an important step. It is also important to realize that values in $\boldsymbol{\Gamma}$ are non-negative, and we have made that argument previously. **MARKING SCHEME:** 7.5 points for each part. In the last part, you must argue for the case when $m \neq n$, else you lose 4 points. In the second last part, you must argue why γ_i is not negative, else you lose 4 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 $\boldsymbol{u}_2^T\boldsymbol{u}_1 = 0$.