Final exam: CS 663, Digital Image Processing, 11th November

Instructions: There are 180 minutes for this exam (9:30 am to 12:30 pm). Answer all 8 questions. This exam is worth 20% of the final grade. Some formulae are listed at the end of the paper. Think carefully and write **brief, concise** answers.

1. **Image Restoration:** A professor of history needs your help as an image processing expert to solve a problem. He had ten ancient manuscripts (all similar to each other and of same size), five of which have been permanently misplaced. The rest are still in his possession. Luckily he had acquired pictures of each of the ten manuscripts, using a camera on a tripod looking vertically down on the manuscript which was placed at a fixed position on a flat table (all under the same lighting conditions, manuscript position and camera viewpoint). However unfortunately, all the pictures appear blurred because the camera was out of focus during the acquisition, due to which the professor is unable to decipher the contents of the five misplaced manuscripts. The professor can give you access to the exact same camera that was used for acquiring the pictures. He knows the distance between the camera and the manuscript during the acquisition of the pictures and the position of the manuscript on the table, and can reproduce the exact same lighting conditions. Your task is to devise a method to deblur the images, making use of the information/equipment he has provided and your knowledge of image processing. Write down a concise, step by step procedure for this. Include mathematical equations if necessary. [20 points]
Answer:

   - The answer lies in Wiener filtering. The DFT of the deblurred image will be given as $\hat{F}(u,v) = \dfrac{|H(u,v)|^2 G(u,v)}{H(u,v)[|H(u,v)|^2 + Sn(u,v)/Sf(u,v)]}$ where $G(u,v)$ is the DFT of the blurred image and is already available.
   - $Sf(u,v)$ is an estimate of the power spectrum of the unknown (to-be-estimated) image and this can be obtained by taking in-focus pictures of one of the five available manuscripts since we know they were similar (but not exact same) as the ones which have been misplaced. Yes, these images will be corrupted by noise, by under the assumption that the noise magnitude is small, the net effect on the Wiener deblurring will not be significant.
   - How will you obtain $H(u,v)$? We have $H(u,v) = F_{defocussed}(u,v)/F_{infocus}(u,v)$ but this assumes negligible camera noise. A better solution is to find $H(u,v)$ that will minimize $\sum_{i=1}^{5} \|F^i_{infocus}(u,v) - H(u,v)F^i_{defocussed}(u,v)\|^2$. In fact, one can take as many pictures of the available images to get an even more reliable estimate of $H(u,v)$. Another method is to create a white poster with a small dot in it, and take pictures from the same distance as before. The noise in such a picture can be easily smoothed out, and the blur kernel can be estimated.
   - That leaves behind just $Sn(u,v)$. Since noise is spread-spectrum, you can make an assumption that $Sn(u,v) = K$ where K is some constant. You can produce a deblurred image for different values of K and let the professor decide which image is the best result for him. An even better method is to take a picture of a white poster under the same acquisition settings. The variance of the captured image will equal the noise variance and that will give you an estimate of $Sn(u,v)$ as well.

   Marking scheme: 5 points for identifying that you need a Wiener filter! 5 points for specifying how to estimate the blur kernel. 5 points for specifying how you can use five available manuscripts as the estimate for the power spectrum of the unknown ones. 5 points for taking into account the issue of noise and specifying a method to deal with it.

2. **SVD:** Consider a matrix $\mathbf{A}$ of size $m \times n$ where m is much larger than n. Explain how you will **efficiently** compute the SVD of $\mathbf{A}$ (*i.e.*, the decomposition $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal, and $\mathbf{S}$ is

a diagonal matrix with non-negative values), if you had access to a software routine that computed the eigenvectors and eigenvalues of a square matrix, and assuming you had no access to any software routine that directly computed the SVD. State the time complexity of your procedure. [10 points]
Answer: Compute an eigendecomposition of $\mathbf{A}^T\mathbf{A} = \mathbf{V_1}\mathbf{\Lambda}\mathbf{V_1}^T$. This is an $O(n^3)$ operation and computing $\mathbf{A}^T\mathbf{A}$ takes $O(mn^2)$ time. The eigenvectors in $\mathbf{V_1}$ are the right singular vectors of $\mathbf{A}$. The singular values of $\mathbf{A}$ will be the positive square-roots of the eigenvalues in $\mathbf{\Lambda}$. Now we need to compute the left singular vectors of $\mathbf{A}$. This can be done by computing $\mathbf{AV}$ and then unit-normalizing each column of $\mathbf{AV}$. This will taken another $O(mn^2)$ time for a total of $O(n^2(m+n))$ time.
Marking scheme: Only 3 points if you independently estimated $\mathbf{U}$ and $\mathbf{V}$ due to lack of sign consistency. Correct estimation of $\mathbf{V_1}$ (right singular vectors) fetches 3 points, 2 points for the singular values and 3 points for the method to update the left singular vectors using $\mathbf{V_1}$. 2 points for correct time complexity (1 point if the student did not know eigenvector computation is $O(n^3)$).

✓3. **Fourier transforms:** Explain qualitatively what effect the following transformations will have on the Discrete Fourier Transform of an image:

(a) random shuffling of the intensity values of the image pixels,
Answer: will significantly increase the high frequency content due to considerable fluctuation between neighboring intensity values, and lower frequency components (except for DC value) will reduce in magnitude.

(b) convolution of the image with a Gaussian kernel of very tiny (nearly zero) standard deviation,
Answer: will have nearly no effect on the DFT as the image is being convolved with just a delta function.

(c) addition of zero mean Gaussian noise with some standard deviation σ to the image,
Answer: will increase the high frequency content of the image significantly, especially since the image is weaker in the higher frequencies

(d) addition of a sinusoidal pattern $z = A\sin(ax+by)$ to the image where a and b are known constants, and x, y are spatial coordinates,
Answer: will add to the DFT of the image at frequencies (Ma, Nb) and $(-Ma, -Nb)$ for an $M \times N$ sized image. The amount to be added will be of magnitude A. Sufficient to say that two Fourier transform magnitude at two frequency values directly related to (a, b) will get affected by additive factor A.

(e) point-wise multiplication of the image with a complex exponential pattern $\exp(j2\pi(ax+by))$ where a and b are known constants, and x, y are spatial coordinates,
Answer: Multiplication in spatial domain is convolution in Fourier domain. The image DFT will be convolved with a delta function peaked at (Ma, Nb) and hence the Fourier transform will undergo a frequency shift of (a, b), i.e. $F_{new}(u, v) = F(u-a, v-b)$. Another way to do this is to look at a twisted version of the Fourier shift theorem which says that $F(f(x, y)\exp(j(ax+by)) = F(u-a, v-b)$.)

(f) zero-padding of the image on all four sides (for this part, the DFT is computed on the larger zero-padded image. Your answer should be something over and above the obvious statement that the DFT will be a larger array). [6 × 4 = 24 points]
Answer: This will significantly increase the Fourier transform magnitudes along the u and v axes because the zero-padding produces strong edges parallel to the X and Y axes respectively. See problem 4.22 in the textbook.

✓4. **Color Image Processing:** A student decides to define the gradient vector of a color image in RGB format as $\nabla I(x, y) = \nabla R(x, y) + \nabla G(x, y) + \nabla B(x, y)$ at pixel (x, y). However this may falsely yield $\nabla I(x, y) = (0, 0)$ even at a color edge. Give an example to illustrate this. [5 points]
Answer: Consider an image consisting of two regions separated by a vertical edge. Let the pixels in the two regions be colored $(255, 0, 0)$ and $(0, 0, 255)$ respectively. We will have $\nabla I(x, y) = (-255, 0) + (0, 0) + (0, 255) = (0, 0)$ at the color edge.

✓5. **Color Image Compression and PCA:** Given a color image with N pixels in RGB format, suppose you perform PCA on the N vectors (all 3D vectors) and produce N corresponding eigencoefficient vectors (with

3 values in each vector). Prove that the eigencoefficients are decorrelated. What is the practical application of this property in compression of the color image? [6 + 4 = 10 points]
Answer: To perform PCA, we compute the eigenvectors of the covariance or correlation matrix $\mathbf{C} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T$ where $C = \sum_{i=1}^{N} \mathbf{x_i}\mathbf{x_i}^t$ where $\mathbf{x_i}$ is the vector of values at the i^{th} pixel. The eigencoefficients for the i^{th} pixel are $\alpha_\mathbf{i} = \mathbf{V}^T\mathbf{x_i}$. The correlation matrix of the eigencoefficients is $\sum_i \alpha_\mathbf{i}\alpha_\mathbf{i}{}^t = \sum_i \mathbf{V}^t\mathbf{x_i}\mathbf{x_i}{}^t\mathbf{V} = \mathbf{V}^T\mathbf{C}\mathbf{V} = \boldsymbol{\Lambda}$ which is a diagonal matrix. This proves that the eigencoefficients are decorrelated. This has applications in image compression. You can now independently JPEG-compress each of the individual eigencoefficient images. The quantization step will affect only the concerned eigencoefficient, without affecting other values. Furthermore, one can even downsample the eigencoefficient images for those eigencoefficients which have a small variance.
Marking scheme: 6 points for the proof - partial credit for sensible steps. 4 points for the usefuleness in compression - 2 points even for a mention of YCbCr as a decorrelated color space.

6. **Compression:** An inquisitive student has acquired a video of an object moving very slowly against a stationary background, and wants to compress the video using his own ideas (described here). Consider that there are T frames in all, and each frame has size $H \times W$. The frame at time instant t is denoted as I_t ($0 \leq t \leq T-1$). The student computes the gradients in the X and Y direction for each frame of the video except the last frame. Let the pixel coordinates be given by (x, y) where $0 \leq x \leq W-1, 0 \leq y \leq H-1$. Thus, for a frame I_t, he computes $\frac{\partial I_t(x,y)}{\partial x} = I_t(x+1, y) - I_t(x, y)$ (if $x = W-1$, he sets $\frac{\partial I_t(x,y)}{\partial x}$ to 0) and $\frac{\partial I_t(x,y)}{\partial y} = I_t(x, y+1) - I_t(x, y)$ (if $y = H-1$, he sets $\frac{\partial I_t(x,y)}{\partial y}$ to 0). In any frame, he observes that most of the gradient values are very small, so he stores only those values that are above some threshold τ, along with their location. He stores the last frame I_{T-1} entirely as it is. Given the stored gradient values in each frame and the entire last frame, explain how the student can reconstruct the video sequence. Of course, this will be a lossy reconstruction. [15 points]
ANSWER: Given an image, it is trivial to compute the gradients. Given the gradients, it is not easy to get back the image. The FT of I_x is $\hat{I}_x(u, v) = \hat{I}(u, v)(e^{j2\pi u} - 1)$ where $\hat{I}(u, v)$ is the FT of image I. To estimate $\hat{I}(u, v)$ from $\hat{I}_x(u, v)$, the problem occurs for the frequency components with $u = 0$. Similarly, the FT of I_y is $\hat{I}_y(u, v) = \hat{I}(u, v)(e^{j2\pi v} - 1)$ where $\hat{I}(u, v)$ is the FT of image I. To estimate $\hat{I}(u, v)$ from $\hat{I}_y(u, v)$, the problem occurs for the frequency components with $v = 0$. If you knew both $\hat{I}_y(u, v)$ and $\hat{I}_x(u, v)$, the problem still occurs for the component $u = v = 0$, which is the DC component. As the object was moving slowly and the last frame is known in its entirety, you can make the crude assumption that the DC component of each frame is equal to the DC component of the last frame, and therefore reconstruct the entire video. [NOTE: The same issue will be encountered even if you attempt to solve this entirely in the spatial domain, by doing digital integration (instead of differentiation). In integration, you always have the problem of unknown constant of integration. Not knowing what to do at the $u = 0$ component, is a manifestation of this same problem, when you integrate across rows. Likewise when you integrate across columns, the unknown constant of integration manifests itself as the indeterminate $v = 0$ component.]

But this problem is more complicated than this! Note that we have nullified gradients whose absolute value fell below τ - for the sake of compression. Due to this, the reconstruction problem becomes more complicated. The estimates of $\hat{I}(u, v)$ using (1) $\hat{I}_x(u, v)$ and (2) using $\hat{I}_y(u, v)$ will not be consistent (i.e. for the same (u, v) you will get two different values of $\hat{I}(u, v)$, and hence two different images!). One (very) crude solution is to take an average of the two, and that is a reasonable answer. There exist much better ways of solving this problem, for instance solving a least squares problem $E(J) = \sum_{x,y}(I_x(x, y) - J_x(x, y))^2 + (I_y(x, y) - J_y(x, y))^2$.
Marking scheme: Pointing out problems with the integration process is very important and takes 9 points for a sensible answer. 3 points for stating that you can get a rough estimate of the constant of integration from the last frame. 3 points for stating the additional issues due to noise. Many students sought to mimic MPEG by searching for matching blocks and copying them backwards from the last frame. However this ignores many issues. First, matching in gradient space is risky, as the DC component is removed during gradient computation and this can lead to false matches. Second, the portion of the background occluded by the object in the last frame will never be reconstructed properly by this 'patch-copying method'. Third, if the moving object undergoes any geometric transformations or occlusions, you have another set of problems!

✓7. **Color Image Processing:** In color images, the hue θ at a pixel is calculated from the R,G,B values at that pixel, using the following method. Define $h \triangleq \cos^{-1}\left(\frac{0.5(2R-G-B)}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right)$. If $B \leq G$, set $\theta = h$ otherwise $\theta = 2\pi - h$ (all calculations in radians). What are the advantages and disadvantages of using hue in color image applications? [8 points]

Answer: Hue is invariant to scaling of RGB by any positive constant or addition of any positive constant to the RGB values. Hence it is invariant to scaling of the intensity of a white light source or change in light direction, or specular reflections or white ambient lighting (additive). It also gives you the true color. However, it is sensitive to noise, especially near the $R = G = B$ axis (i.e. small changes in RGB can lead to very large changes in hue). Also, hue is not directly used for display of the color values.

Marking scheme: 3 points for mentioning it is a multiplicative scaling invariant, 3 points for mentioning that it is invariant to additive offsets. 2 points for any one disadvantage.

✓8. **Spatial Filters:** Briefly explain (in up to 3-4 sentences each) any two uses of the image Laplacian in image processing. The Laplacian is given by $\triangle f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$. [8 points]

Answer: The Laplacian is the simplest rotationally invariant double derivative operator for 2D (or higher dimensions). The zero crossings of the Laplacian occur at image edges, so it is used for edge detection. It can be used for image sharpening, for which you need to deduct (not add!) the Laplacian from the original image. You can use the Laplacian for image smoothing, for which you add (not subtract) the Laplacian to the image - which is the isotropic heat equation (equivalent to Gaussian smoothing). Some of you wrote that convolution of an image with a Laplacian kernel smoothes the image, which is absolutely incorrect - any derivative operator is a high pass filter! Some of you wrote that the Laplacian is used in the Perona Malik equation, which is imprecise, unless you specify how ($div(g\Delta I) = g_x I_x + gI_{xx} + g_y I_y + gI_{yy}$). Another application is to use the Laplacian as a regularizer in regularized restoration (see lecture slides on image restoration).

Marking scheme: 4 points each for an application. Merely mentioning the application (eg: image smoothing) without specifying the role of the Laplacian would yield only 1.5 points out of 4.

LIST OF FORMULAE:

1. Gaussian pdf in 1D centered at μ and having standard deviation σ: $p(x) = \frac{1}{\sqrt{2\pi}\sigma}e^{-(x-\mu)^2/(2\sigma^2)}$.
2. 1D Fourier transform and inverse Fourier transform:
$F(u) = \int_{-\infty}^{+\infty} f(x)e^{-j2\pi ux}dx, f(x) = \int_{-\infty}^{+\infty} F(u)e^{j2\pi ux}du$
3. 2D Fourier transform and inverse Fourier transform:
$F(u,v) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x,y)e^{-j2\pi(ux+vy)}dxdy, f(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} F(u,v)e^{j2\pi(ux+vy)}dudv$
4. Convolution theorem: $\mathcal{F}(f(x) * g(x))(u) = F(u)G(u); \mathcal{F}(f(x)g(x))(u) = F(u) * G(u)$
5. Fourier transform of $g(x-a)$ is $e^{-j2\pi ua}G(u)$. Fourier transform of $\frac{df^n(x)}{dx^n} = (j2\pi u)^n F(u)$ ($n > 0$ is an integer).
6. 1D DFT: $F(u) = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1} f(x)e^{-j2\pi ux/N}, f(x) = \frac{1}{\sqrt{N}}\sum_{u=0}^{N-1} F(u)e^{j2\pi ux/N}$
7. 2D DFT: $F(u,v) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux+vy)/N}, f(x,y) = \frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(ux+vy)/N}$