EE368/CS232 - Midterm Exam

24-hour take-home exam, 3 slots in Feb 13 – 16.
Each slot begins at 5:00pm and ends at 5:00pm the next day.

Please respect the Stanford Honor Code. There are a total of 4 problems in this midterm exam. You may use the lecture notes, your own notes, or any book to solve the exam. However, you may not discuss the exam with anyone else, including on Piazza or any other online forum, until February 16 at 5:00pm, when everyone in the class will have taken the exam. The only exception is that you can ask the course staff for clarification by emailing the staff email address (ee368-win1819-staff@lists.stanford.edu). Emails sent to other email addresses may not be answered. We have tried hard to make the exam unambiguous and clear, so we are unlikely to say much.

Submission guidelines:
• All the files needed for the problems are located in the archive midterm_data.zip that you can download at: https://web.stanford.edu/class/ee368/rrQNV3zBsD/midterm_data.zip
• Submission must be made electronically on Gradescope. Please submit your answers as a single pdf file to the “Exam MM/DD” assignment, with relevant Matlab code and figures, where MM/DD corresponds to the date you received the exam. The assignment for a given date will close at 5pm the next day and late submissions will not be accepted.
• If you experience issues submitting to Gradescope, please email your solution to the staff email address (ee368-win1819-staff@lists.stanford.edu).
Problem 1: Multiple Choice Questions (20 points)

Each part of this problem is a multiple-choice question. You do not have to explain your choice to receive full credit, as long as your answer is correct. If your answer is not correct, we will consider an explanation of your reasoning (no longer than 100 words) for possible partial credit.

Part A (4 points):

We consider a tiny grayscale image of bit depth 3 (i.e., 8 gray levels) with the following histogram:

<table>
<thead>
<tr>
<th>gray level</th>
<th>bin count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>

We apply histogram equalization to this image, yielding an image with same bit depth. Among the following options, what is the histogram of this new image?

(A)

<table>
<thead>
<tr>
<th>gray level</th>
<th>bin count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
</tr>
</tbody>
</table>

(B)

<table>
<thead>
<tr>
<th>gray level</th>
<th>bin count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>

(C)

<table>
<thead>
<tr>
<th>gray level</th>
<th>bin count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>11</td>
</tr>
</tbody>
</table>
Part B (4 points):

S is a horizontal line-shaped structural element of length 5. We consider an image processing algorithm that performs the following series of operations on a binary image:

1. dilation by S
2. clockwise rotation of the image by 90°
3. dilation by S
4. bit inversion (logical NOT)
5. erosion by S
6. counter-clockwise rotation of the image by 90°
7. bit inversion (logical NOT)

What is this algorithm equivalent to?

(A) dilation with a rectangular structuring element of width 5 and height 9
(B) dilation with a rectangular structuring element of width 9 and height 5
(C) dilation with a horizontal line-shaped structuring element of length 5
(D) dilation with a vertical line-shaped structuring element of length 5
Part C (7 points):

The input image shown below is processed by 4 different linear shift-invariant systems. Match the output images (1-4) to frequency responses (A-D). Note: \( \omega_x \) and \( \omega_y \) are horizontal and vertical frequencies respectively. When the output image has negative values, 0 is mapped to mid-gray and the image is scaled appropriately to occupy the entire range from black to white.

![Input Image](image)

Filter frequency responses \( F(\omega_x, \omega_y) \) below:

(A) \[ F(\omega_x, \omega_y) = \begin{cases} \exp(jX \omega_x) & \text{if } (\omega_x^2 + \omega_y^2) < T \\ 1 & \text{else} \end{cases} \] where \( X, \sigma, T \) are constants

(B) \[ F(\omega_x, \omega_y) = \exp\left(-\frac{\omega_x^2}{(\sigma_0+\Delta\sigma)^2}\right) - \exp\left(-\frac{\omega_x^2}{(\sigma_0-\Delta\sigma)^2}\right) \] where \( \sigma_0, \Delta\sigma \) are constants

(C) \[ F(\omega_x, \omega_y) = \cos\left(A(\omega_x + \omega_y)\right) \] where \( A \) is a constant

(D) \[ F(\omega_x, \omega_y) = \exp\left(-\frac{\omega_y^2}{\sigma^2}\right) \] where \( \sigma \) is a constant

Outputs:

![Output Images](image)

(1) (2) (3) (4)
Part D (5 points):

A linear shift-invariant digital filter that approximates the mixed 2nd derivative of an image smoothed by a Gaussian has the 2-d impulse response

\[
\begin{bmatrix}
1 & 4 & 5 & 0 & -5 & -4 & -1 \\
4 & 16 & 20 & 0 & -20 & -16 & -4 \\
5 & 20 & 25 & 0 & -25 & -20 & -5 \\
0 & 0 & 0 & [0] & 0 & 0 & 0 \\
-5 & -20 & -25 & 0 & 25 & 20 & 5 \\
-4 & -16 & -20 & 0 & 20 & 16 & 4 \\
-1 & -4 & -5 & 0 & 5 & 4 & 1
\end{bmatrix}
\]

It can be implemented very efficiently with only 12 additions/subtractions per output pixel by convolving the image with an impulse response of [-1 1] and five times with [1 1] horizontally and vertically. What is the approximate variance \( \sigma^2 \) of the Gaussian?

(A) \( \sigma^2 \approx 1.5 \) \quad (B) \( \sigma^2 \approx 6 \) \quad (C) \( \sigma^2 \approx 12 \) \quad (D) \( \sigma^2 \approx 36 \)
Problem 2: Reverse Engineering (20 points)

In this problem, we will perform reverse engineering to discover two different image processing algorithms. The data from this problem include the following two images:

- zoneplate.tif
- man_noisy.tif

These two images are the inputs to two unknown image processing algorithms.

**Part A (10 points):**
A first image processing algorithm IPA1 is applied to both input images. The resulting images are:

- zoneplate_IPA1.tif
- man_noisy_IPA1.tif

Please describe as clearly as possible what processing steps are contained in IPA1. Justify your conclusions.

**Part B (10 points):**
A second image processing algorithm IPA2 is applied to both input images. The resulting images are:

- zoneplate_IPA2.tif
- man_noisy_IPA2.tif

Please describe as clearly as possible what processing steps are contained in IPA2. Justify your conclusions.

Note: If you use MATLAB to support your conclusions for Part A or Part B, please include the relevant MATLAB code.
**Problem 3: Gamma and Color (30 points)**

The contrast of a grayscale image can be adjusted by changing the gamma of the display. In this problem, we study the effects of adjusting the contrast of color images by changing the gamma for each of the R, G, B channels. More precisely, the values (R, G, B) are each raised to the power of $\gamma > 0$ according to $(R', G', B') = (R^\gamma, G^\gamma, B^\gamma)$ for all colors $0 \leq R, G, B \leq 1$, where $(R', G', B')$ are the contrast-adjusted values.

**Part A (10 points):**

In order to study the effects of changing the RGB gamma in the chromaticity plane, we need the forward and the inverse mapping from $(R, G, B)$ to $(x, y, Y)$. These formulas will be handy for solving Parts B to D.

Our RGB space is defined by the NTSC primaries, given by the following chromaticity coordinates:

\[
\begin{align*}
    x_R &= 0.67 & y_R &= 0.33 \\
    x_G &= 0.21 & y_G &= 0.71 \\
    x_B &= 0.14 & y_B &= 0.08
\end{align*}
\]

The reference white is illuminant C with the chromaticity coordinates:

\[
(\bar{x}_W = 0.31005, \bar{y}_W = 0.31616)
\]

Find closed-form expressions that map $(R, G, B)$ to $(x, y, Y)$ and that conversely map $(x, y, Y)$ to $(R, G, B)$, where $x, y$ are the CIE chromaticity coordinates and $Y$ is luminance. Your mappings should be scaled such that the brightest white $(R, G, B) = (1, 1, 1)$ maps to $(x, y, Y) = (\bar{x}_W, \bar{y}_W, 1)$.

The closed form expressions should be explicit functions like this:

\[
\begin{align*}
    x &= f_x(R, G, B) & R &= f_R(x, y, Y) \\
    y &= f_y(R, G, B) & G &= f_G(x, y, Y) \\
    Y &= f_Y(R, G, B) & B &= f_B(x, y, Y)
\end{align*}
\]

These functions should incorporate the chromaticity values for the primaries and the reference white given above. In order to keep the expressions compact, it is recommended to evaluate the constants (e.g., using MATLAB) and to include numerical values in your expressions.

Note: Please include any relevant MATLAB code.

**Part B (5 points):**

Consider a color with coordinates $(R, G, B)$ and corresponding $(x, y, Y)$ that is adjusted according to $(R', G', B') = (R^\gamma, G^\gamma, B^\gamma)$. Let $(x', y', Y')$ denote the chromaticity and luminance corresponding to $(R', G', B')$. Mathematically show that $(x', y')$ depends only on $(x, y)$, but not on $Y$. 

Part C (10 points):

In the file `colors_xy.mat` we provide the (x, y) chromaticity coordinates of 210 colors: each column corresponds to a color; the first row lists the x values and the second row lists the y values. Their locations in the chromaticity diagram are shown below.

For the adjustment given in Part B, we want you to find the (x', y') coordinates of the corresponding gamma-mapped colors. Recall that (x', y') is independent from the luminance Y. The file `chroma_mask.mat` contains a binary image marking the interior of the CIE chromaticity diagram in xy space. Display that mask and, for each color, draw an arrow on this mask that points from the original (x, y) coordinates to the gamma-mapped (x', y') coordinates. We want you to submit two plots: one for γ = 0.8, and one for γ = 1.2.

Notes:
- You can use the following code to display your mask with the axes in the [0,1] range:
  ```matlab
  RI = imref2d(size(chroma_mask));
  RI.XWorldLimits = [0 1];
  RI.YWorldLimits = [0 1];
  imshow(chroma_mask, RI)
  set(gca, 'YDir', 'Normal')
  ```
- If xy are the original coordinates and xy_gamma the gamma-mapped coordinates, you can use the following code to plot the arrows:
  ```matlab
  xy_delta = xy_gamma - xy;
  quiver(xy(1,:),xy(2,:),xy_delta(1,:),xy_delta(2,:),0)
  ```

Note: Please include any relevant MATLAB code.

Part D (5 points):

Interpret the plots you generated in Part C. What happens to the displayed colors if contrast is increased or decreased by changing RGB gamma? Comment on both saturation and hue. Would you recommend RGB gamma change as a way to adjust the contrast of color images? Your answer should be between 50 and 100 words long.
Problem 4: Shredder Woes (30 points)

Law enforcement sometimes faces the challenge of recovering shredded documents. In this problem, you are provided with scanned images of the paper strips collected from a shredder. Your task is to use the image processing knowledge gained from EE368 to design and implement an algorithm that recovers each of the original documents.

For your convenience, the shreds have been sorted into 2 distinct sets – each corresponding to one original document. In your data folder, you can find the shreds stored as shreds_{1, 2}/shred_{XX}.png, where XX goes from 0 to 99. The shreds are in random order and random orientation. Each set is complete, i.e., no shreds are missing.

Please submit the recovered images, all relevant MATLAB code, as well a brief description how your algorithm works. You will be evaluated based on the following criteria.

1. The largest, continuous portion of the document that you have been able to recover from each set of shreds
2. Correct alignment of individual strips within the reconstructed result
3. The time that your algorithm takes to compute. Please use the MATLAB functions tic and toc and report the overall runtime of your code for each document.
4. Postprocessing to remove or reduce any visible seams or artifacts in the recovered documents
5. Proper functioning of the MATLAB code you submit

Note: Your code must run on both sets of shreds without tweaking. If your code contains set-specific parameters, as a penalty the parameters for shreds_1 will be used to evaluate the performance on shreds_2 and vice-versa.

Note: If your code returns a perfectly stitched document rotated by 180 degrees, that still counts as a correct result.