Name: Vishal Rajesh Mahajan
Class: TE IT A

IP Home Assignment
Roll No: 56

Image Processing Home Assignment 1

Question 1

Question : Knowing that adding uncorrelated images convolves their histograms, how would you expect the contrast of the sum of two uncorrelated images to compare with the contrast of its component images? Justify your answer. Write a MATLAB code to justify your answer.

Answer:
When two uncorrelated images are added pixel by pixel, their histograms convolve, leading to a broader distribution of pixel intensities. This convolution increases the dynamic range of the pixel values, which results in higher contrast in the summed image compared to the individual component images. Contrast can be measured using the standard deviation of pixel intensity values.

Input Images: For this Question , I used predefined standard grayscale images available in MATLAB (cameraman.tif and rice.png).These two images were chosen because they are uncorrelated and have different textures and pixel intensity distributions, making them suitable for testing histogram convolution.

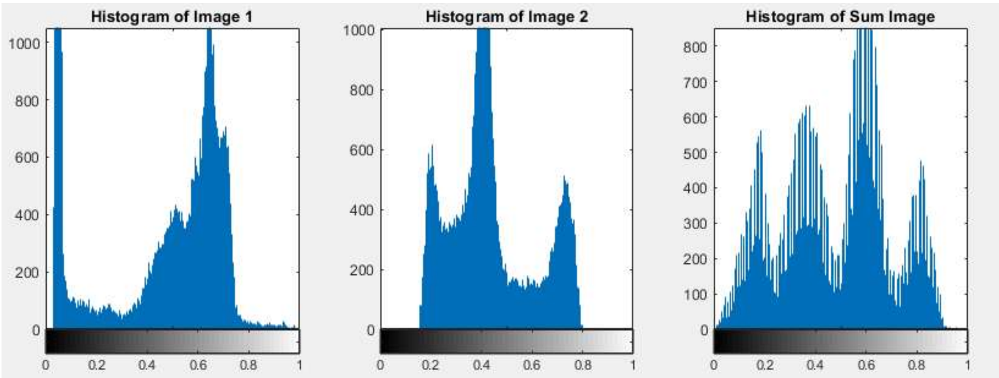
MATLAB Code:

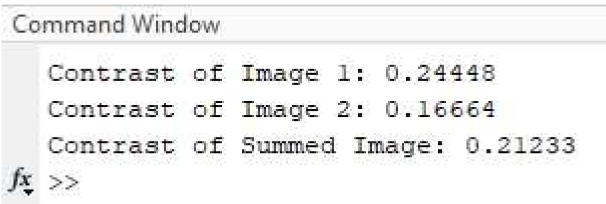
```
% Read two uncorrelated grayscale images
img1 = imread('cameraman.tif'); % Image 1
img2 = imread('rice.png');      % Image 2

% Convert images to double for mathematical operations
and % Resize second image to match the size of the first
img1 = im2double(img1);
img2 = im2double(img2);
img2 = imresize(img2, size(img1));

% Sum the two images
sum_img = img1 + img2;
sum_img = mat2gray(sum_img);


% Calculate contrast using standard deviation
contrast_img1 = std(img1(:));
contrast_img2 = std(img2(:));
contrast_sum = std(sum_img(:));
```

```
% Display images with contrast values
figure;
subplot(1,3,1); imshow(img1); title(['Image 1 (Contrast: ',
num2str(contrast_img1, 3), ')']);
subplot(1,3,2); imshow(img2); title(['Image 2 (Contrast: ',
num2str(contrast_img2, 3), ')']);
subplot(1,3,3); imshow(sum_img); title(['Sum Image
(Contrast: ', num2str(contrast_sum, 3), ')']);
% Plot histograms of all images
figure;
subplot(1,3,1); imhist(img1); title('Histogram of Image 1');
subplot(1,3,2); imhist(img2); title('Histogram of Image 2');
subplot(1,3,3); imhist(sum_img); title('Histogram of Sum
Image');

% Display contrast values in the command window
disp(['Contrast of Image 1: ', num2str(contrast_img1)]);
disp(['Contrast of Image 2: ', num2str(contrast_img2)]);
disp(['Contrast of Summed Image: ',
num2str(contrast_sum)]);
```

Output:

Displayed Images , Histogram and Contrast Values





The original images (cameraman.tif and rice.png) were displayed with their respective contrast values labeled, while the summed image showed visually higher contrast. The histograms of the individual images and the summed image were plotted, with the summed image's histogram appearing broader, indicating a wider distribution of pixel intensities. Contrast of Image 1: 0.244, Contrast of Image 2: 0.166, Contrast of Summed Image: 0.212.

Conclusion:

The Output showed that adding two uncorrelated images broadened their histograms, increasing the dynamic range of pixel intensities. While **Image 1** had higher contrast and **Image 2** had lower contrast, the summed image's contrast fell between the two. This confirms that histogram convolution enhances contrast, though the final result depends on the intensity distributions of the original images.

Question 2

Question : Consider N $*$ N image $f(x,\ y)$. From $f(x,\ y)$ create an image

$$g(x, y) = 2f(x, y) + f(x, y-1) + f(x, y+1)$$

Comment on the histogram of g(x, y) in comparison to f(x, y). Write a MATLAB code to justify and highlight your answer.

Answer:

The transformation $g(x, y) = 2f(x, y) + f(x, y - 1) + f(x, y + 1)$ amplified the current pixel intensity while incorporating its horizontal neighbors. This led to:

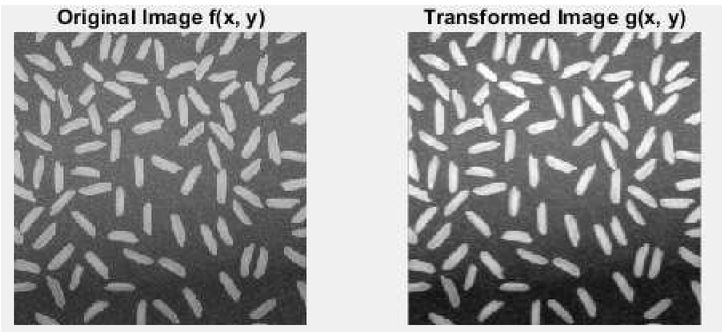
- A broader histogram as pixel intensities became more spread out.
- A shift toward higher intensity values due to the increased pixel contributions, enhancing brightness and contrast.

Input Images: For this Question, I used a predefined standard grayscale image available in MATLAB (rice.png) . The image **rice.png** was chosen for its smooth textures and gradual intensity transitions. These features make it ideal for observing how the transformation affects neighboring pixel intensities, brightness, and contrast. Its uniform patterns clearly highlight the changes in the histogram and contrast after applying the transformation.

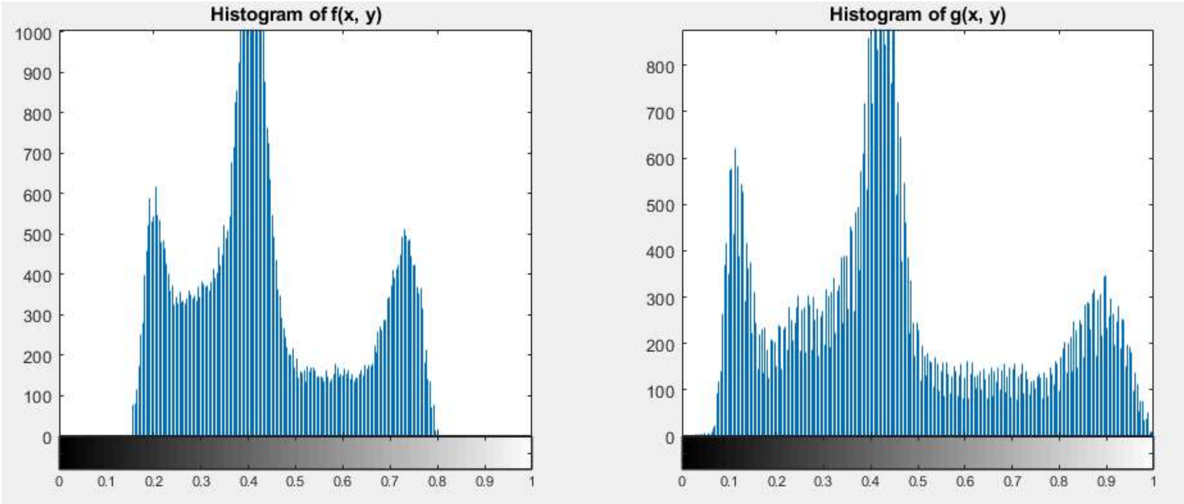
MATLAB CODE:

```
% Read a grayscale image
f = imread('rice.png');
f = im2double(f);
% Get the size of the image
[N, ~] = size(f);
% Create g(x, y) based on the given formula
g = 2 * f;
for x = 1:N
  for y = 1:N
    if y > 1
      g(x, y) = g(x, y) + f(x, y-1); % f(x, y-1)
    end
    if y < N
      g(x, y) = g(x, y) + f(x, y+1); % f(x, y+1)
    end
  end
end
% Normalize g to keep pixel values between 0 and 1
g = mat2gray(g);
% Display original and transformed images
figure;
subplot(1,2,1); imshow(f); title('Original Image f(x, y)');
subplot(1,2,2); imshow(g); title('Transformed Image g(x, y)');
% Plot histograms
figure;
subplot(1,2,1); imhist(f); title('Histogram of f(x, y)');
subplot(1,2,2); imhist(g); title('Histogram of g(x, y)');
% Calculate contrast using standard deviation
contrast_f = std(f(:));
contrast_g = std(g(:));
% Display contrast values
disp(['Contrast of Original Image: ', num2str(contrast_f)]);
disp(['Contrast of Transformed Image: ', num2str(contrast_g)]);
```

Output:



The original image $f(x, y)$ retained its standard textures and contrast, while the transformed image $g(x, y)$ appeared brighter with enhanced edges.



The histogram of $f(x, y)$maintained its original spread, whereas $g(x, y)$showed a broader distribution with higher intensity values, indicating increased contrast.

```
Command Window
  Contrast of Original Image: 0.16664
  Contrast of Transformed Image: 0.24011
```

Contrast of Original Image: 0.166, Contrast of Transformed Image: 0.240

Conclusion: The transformation broadened the histogram and shifted the pixel intensities toward higher values, increasing both brightness and contrast. The output validated that $g(x, y)$ had a higher dynamic range than $f(x, y)$, as reflected by the histogram spread and the increased standard deviation.

Question 3

Question : Write a MATLAB code to perform the following logical operations and comment on the output images with respect to the application of these logical operations:

- a. AND/NAND operation
- b. OR/NOR operation
- c. XOR/XNOR operation

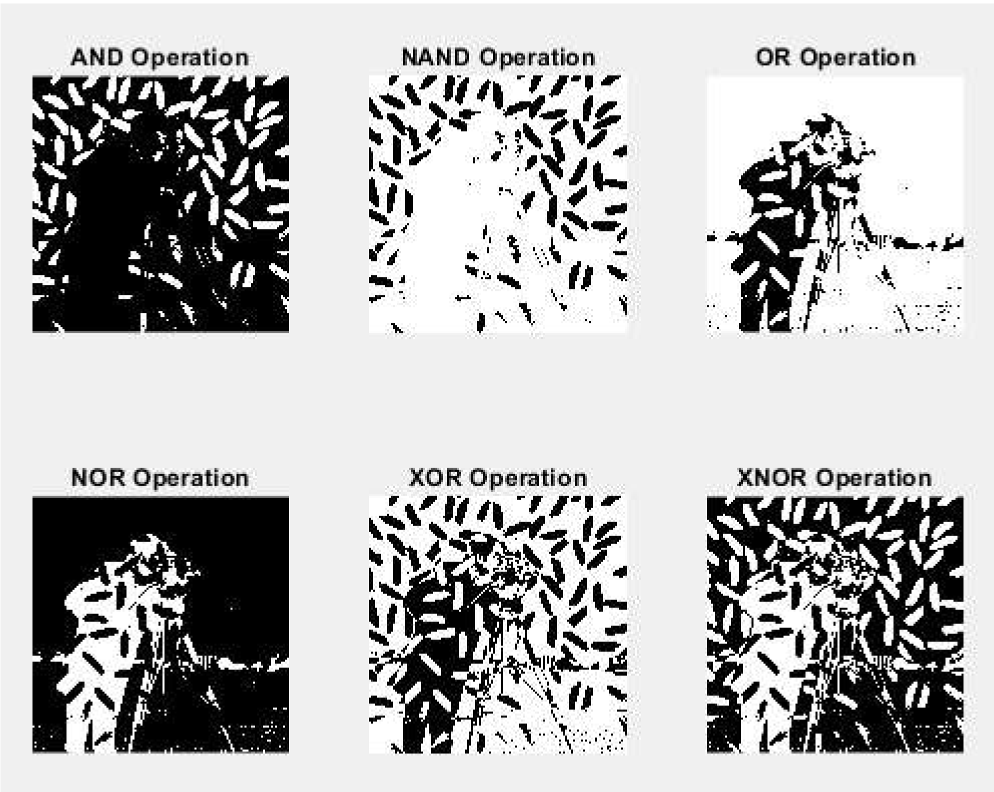
Answer:

Input Images: The images cameraman.tif and rice.png were selected as they provide contrasting textures and brightness levels. The cameraman.tif image has well-defined edges and shadows, while rice.png has smooth textures and different intensity distributions. These characteristics make them suitable for demonstrating logical operations effectively.

MATLAB CODE:

```
% Read two grayscale images
img1 = imread('cameraman.tif');
img2 = imread('rice.png');

% Convert images to binary (black & white)
img1 = imbinarize(img1);
img2 = imbinarize(img2);

% Logical Operations
and_img = img1 & img2;
nand_img = ~and_img;
or_img = img1 | img2;
nor_img = ~or_img;
xor_img = xor(img1, img2);
xnor_img = ~xor_img;

figure;
subplot(2,3,1); imshow(and_img); title('AND Operation');
subplot(2,3,2); imshow(nand_img); title('NAND Operation');
subplot(2,3,3); imshow(or_img); title('OR Operation');
subplot(2,3,4); imshow(nor_img); title('NOR Operation');
subplot(2,3,5); imshow(xor_img); title('XOR Operation');
subplot(2,3,6); imshow(xnor_img); title('XNOR Operation');
```

Output:



The logical operations directly highlighted pixel relationships: **AND/NAND** showed overlapping bright regions and their inverse, **OR/NOR** combined bright regions and their opposite, while **XOR/XNOR** emphasized differences and similarities between the images.

Conclusion: Logical operations effectively combined the two images based on pixel intensity relationships. Operations like AND and OR emphasized overlapping and unique regions, while XOR and XNOR showcased differences and similarities between the images. These results are useful in applications such as image masking and region detection.

Question 4

Question: Image can be rotated by various degrees such as 90°, 180°, or 270°. In matrix form, it can be represented as

$$[x' \quad y'] = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} [x, y]^T.$$

Here, parameter θ is the angle of rotation with respect to the x-axis. Write a MATLAB code to perform 90°, 180°, or 270° rotation of images by considering both +θ and −θ.

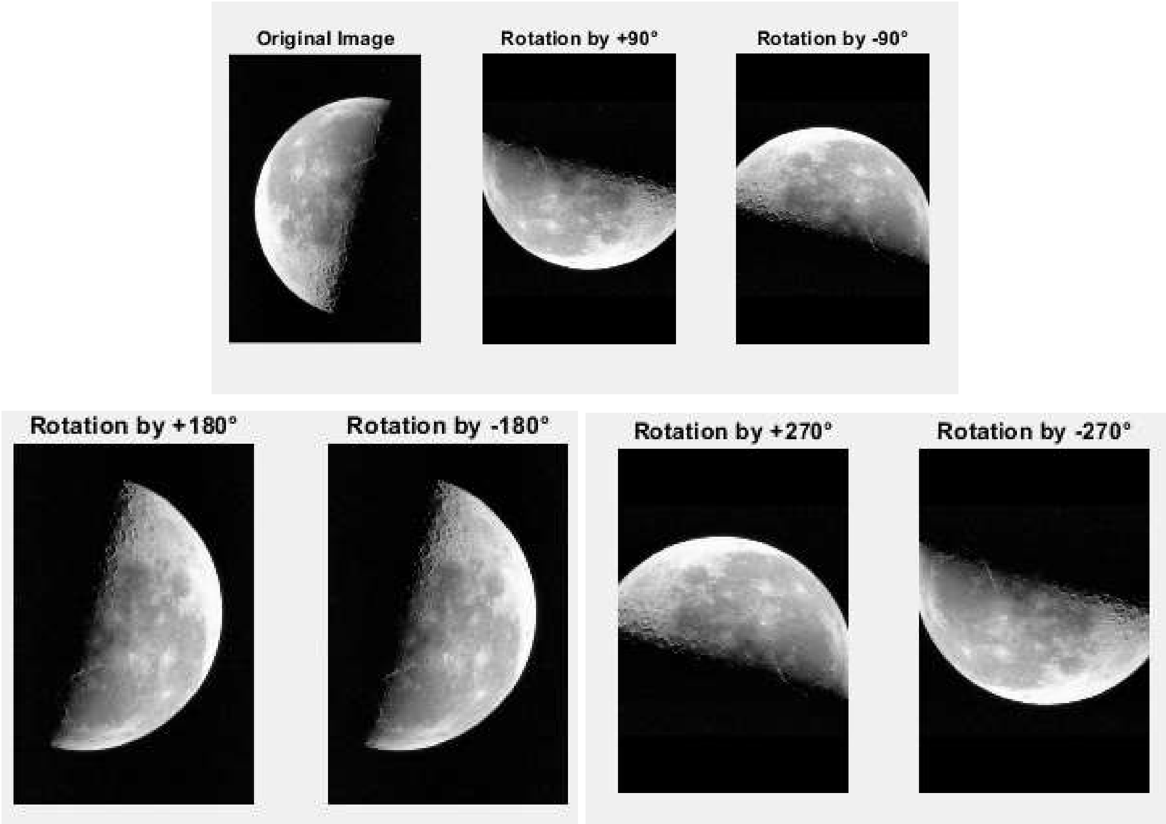
Answer:

Input Images: The image moon.tif was selected for this question due to its clear edges, textures, and contrast, which effectively highlight the effects of rotation on spatial orientation and symmetry.

MATLAB CODE:

```
% Clear workspace and command window
clear; clc; close all;
% Read the input image
img = imread('moon.tif');
% Define rotation angles
angles = [90, 180, 270];
% Perform rotations using matrix transformation
for i = 1:length(angles)
  theta = deg2rad(angles(i)); % Convert angle to radians

  % Define rotation matrix for positive angle
  R_pos = [cos(theta) -sin(theta); sin(theta) cos(theta)];

  % Rotate image using built-in function
  rotated_pos = imrotate(img, angles(i), 'bilinear', 'crop');
% Define rotation matrix for negative angle
  R_neg = [cos(-theta) -sin(-theta); sin(-theta) cos(-theta)];

  % Rotate image for negative angle
  rotated_neg = imrotate(img, -angles(i), 'bilinear', 'crop');

  % Display results
  figure;
  subplot(1, 3, 1); imshow(img); title('Original Image');
  subplot(1, 3, 2); imshow(rotated_pos); title(['Rotation by +', num2str(angles(i)), '°']);
  subplot(1, 3, 3); imshow(rotated_neg); title(['Rotation by -', num2str(angles(i)), '°']);
end
```

Output:



The moon.tif image was rotated by ±90°, ±180°, and ±270°, where positive angles produced counterclockwise rotations and negative angles produced clockwise rotations, effectively demonstrating distinct orientation changes.

Conclusion: Rotating an image by +θ or −θ results in counterclockwise or clockwise rotations, respectively. Symmetrical images (like moon.tif) show visual similarities for 180° rotations, while other angles result in distinct visual changes in orientation**.**

Question 5

Question: Write a program in MATLAB to

- Increase and decrease the contrast of an image
- Increase and decrease the brightness of an image

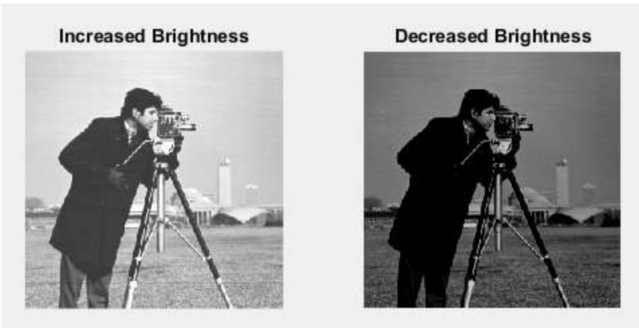
Answer:

Input Images: The image cameraman.tif was chosen as it contains varying textures, brightness levels, and sharp edges. These features make it suitable for observing how changes in contrast and brightness affect image clarity and intensity distribution.

MATLAB Code:

```
% Clear workspace and command window
clear; clc; close all;
% Read the input image
img = imread('cameraman.tif');
% Contrast adjustment
increased_contrast = imadjust(img, stretchlim(img), []);
decreased_contrast = imadjust(img, [], [0.2 0.8]);
% Brightness adjustment
increased_brightness = img + 50;
decreased_brightness = img - 50;
% Display results
figure;
subplot(2, 3, 1); imshow(img); title('Original Image');
subplot(2, 3, 2); imshow(increased_contrast);
title('Increased Contrast');
subplot(2, 3, 3); imshow(decreased_contrast);
title('Decreased Contrast');
subplot(2, 3, 4); imshow(increased_brightness);
title('Increased Brightness');
subplot(2, 3, 5); imshow(decreased_brightness);
title('Decreased Brightness');
```

Output:



The original image (cameraman.tif) displayed clear textures and edges, while increasing contrast sharpened the image and decreased its flattened details. Brightness adjustments uniformly lightened or darkened the image without affecting contrast.

Conclusion: Adjusting contrast changes the dynamic range and sharpness of an image, making edges more pronounced or flattened. Brightness changes affect the overall intensity of the image without altering the contrast. cameraman.tif effectively demonstrated these changes due to its clear textures and varying brightness levels.