RYERSON UNIVERSITY

Faculty of Engineering, Architecture and Science

Department of Electrical and Computer Engineering

Course Number	COE848
Course Title	Introduction to Computer Vision
Semester/Year	F2023

Instructor	Guanghui Richard Wang

ASSIGNMENT No.	1

Assignment Title	Problem Set #1

Submission Date	October 9th, 2023
Due Date	October 9th, 2023

Student Name	Hamza Iqbal
Student ID	500973673
Signature*	H.I

**By signing above you attest that you have contributed to this written lab report and confirm that all work you have swung the lab contributed to this lab report is your own work.*

Part 1

Problem 1.

Log Transformation

The log transformation stretches the dynamic range of low-intensity levels (darker regions) while compressing the dynamic range of higher-intensity levels (brighter regions).

Its equation is defined as:

$$s = c \log (1 + r) \qquad r \geq 0$$

Where s represents the new pixel intensity value, c as the scaling constant, and r as the original pixel intensity value, respectively.

Inverse Log Transformation

The inverse log transformation stretches the dynamic range of higher-intensity levels (brighter regions) while compressing the dynamic range of lower-intensity levels (darker regions). It is the opposite of the log transformation.

Its equation is defined as:

$$s = c \ log^{-1} (r)$$

Where s represents the new pixel intensity value, c is the scaling constant, and r is the original pixel intensity value, respectively.

Power-Law Transformation

The power law transformation is similar to the log transformation but it's more versatile, using gamma correction. It uses a lookup table.

Its equation is defined as:

$$s = cr^{y}$$

Where s represents the new pixel intensity value, c is the scaling constant, r is the original pixel intensity value, and y is the gamma parameter respectively.

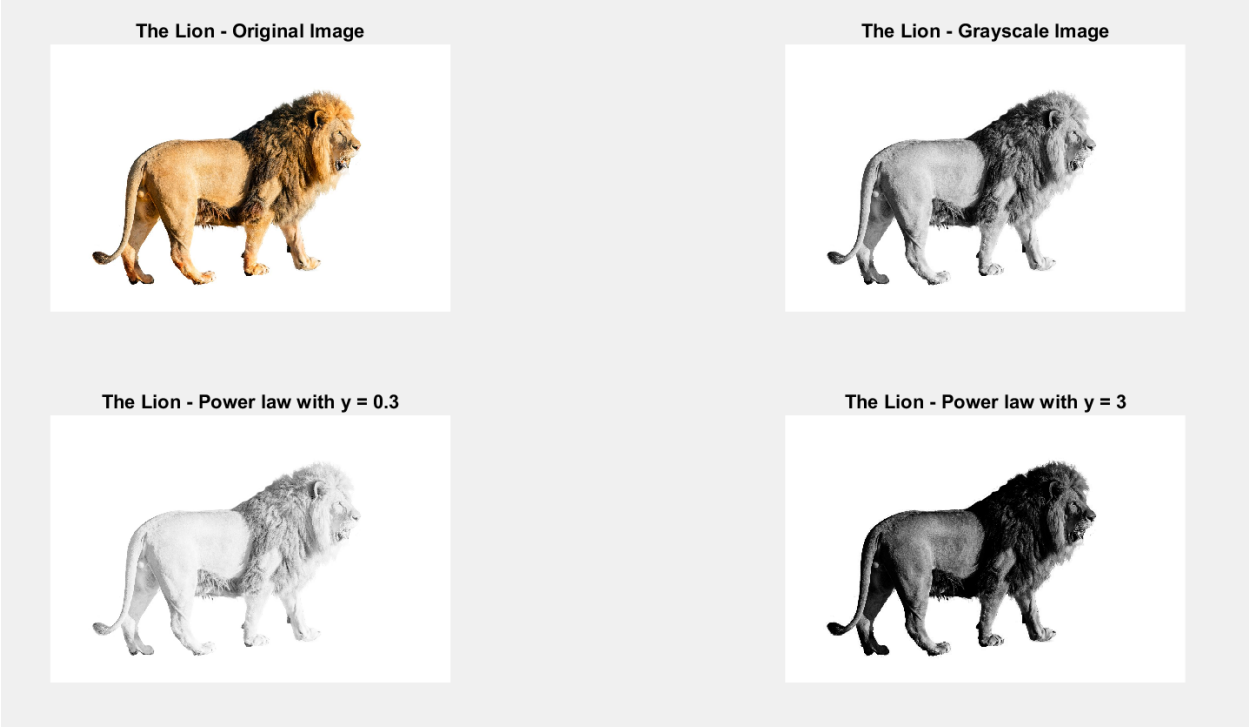

Figure 1 : Original, Greyscale, Power law (y=0.3), Power law (y=3)

```
%% Problem 1.
%% Upload Image


subplot (2,2,1);
img = imread ('lion.jpg');
imshow(img);
title ('The Lion - Original Image');


% Grayscale

subplot (2,2,2);
imggray = rgb2gray (img);
imshow(imggray);
title ('The Lion - Grayscale Image');


% Power-Law transformation with y = 0.3

subplot (2,2,3);
gammalow = imadjust(imggray,[],[],0.3);
imshow(gammalow);
title ('The Lion - Power law with y = 0.3');


% Power-Law transformation with y = 3

subplot (2,2,4);
gammahigh = imadjust(imggray,[],[],3);
imshow(gammahigh);
title ('The Lion - Power law with y = 3');
```

Code 1: Importing Image, converting RGB to grayscale, Power law transformations on grayscale image

In analyzing the results before and after the transformation of the images, it's apparent that increasing the gamma value causes the image to increase in contrast, that is, the dark regions become more apparent. This is apparent in the image where y=3, larger than the default value of 1. When decreasing the gamma value, the image loses contrast, appearing more washed out. This is apparent in the image where y=0.3, smaller than the default value of 1.

Problem 2.

The following showcases the 8-bit slicing done on the images:

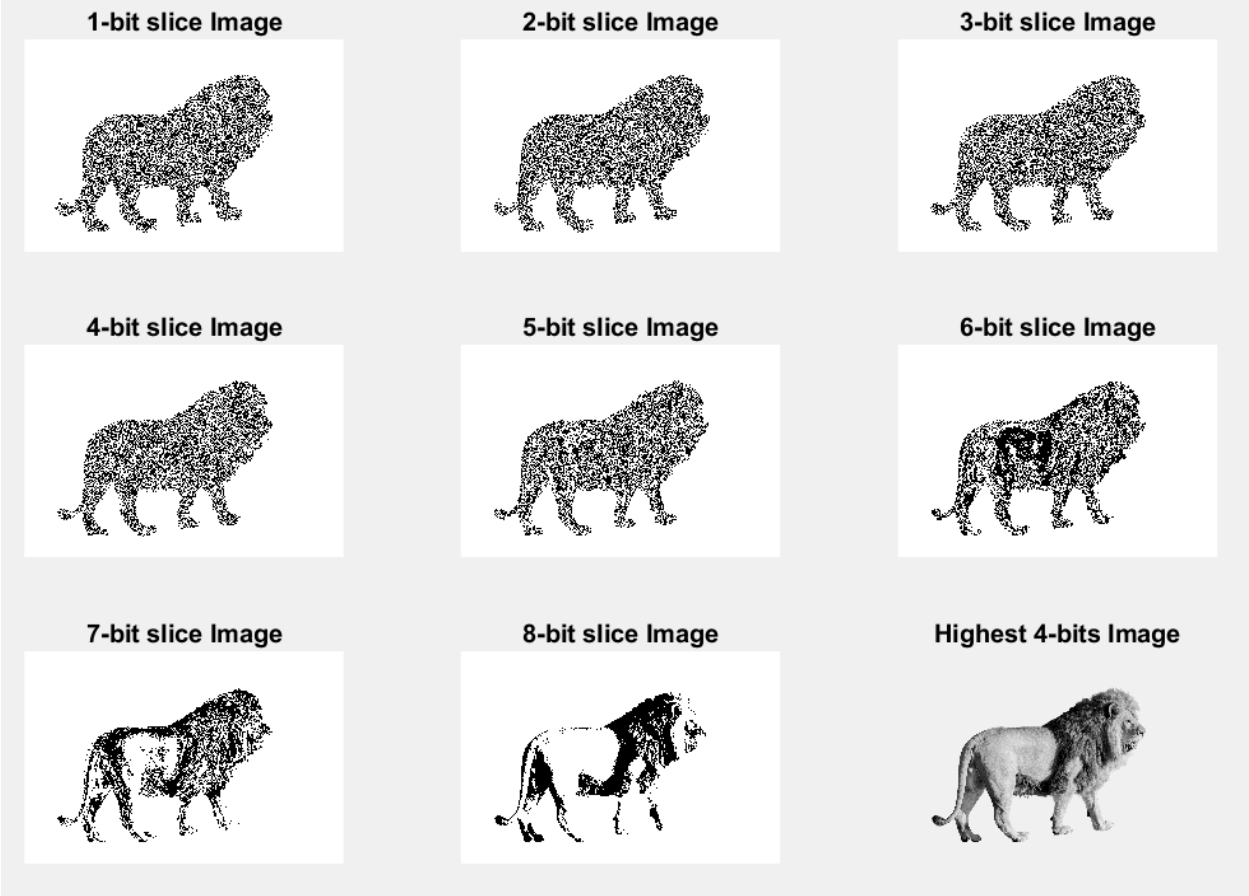

Figure 2: 8 bit slice image, highest 4-bits image

```
%% Problem 2.

% Grayscale Image
figure;
imshow(imggray);
title ('The Lion - Grayscale Image');

% 8-Bit Plane Slicing
bit1 = double (bitget(imggray,1));
bit2 = double (bitget(imggray,2));
bit3 = double (bitget(imggray,3));
bit4 = double (bitget(imggray,4));
bit5 = double (bitget(imggray,5));
bit6 = double (bitget(imggray,6));
bit7 = double (bitget(imggray,7));
bit8 = double (bitget(imggray,8));

subplot (3,3,1);
imshow (bit1);
title ('1-bit slice Image');

subplot (3,3,2);
imshow (bit2);
title ('2-bit slice Image');

subplot (3,3,3);
imshow (bit3);
title ('3-bit slice Image');
```

Code 2: 8-bit Slicing Code

```
subplot (3,3,4);
imshow (bit4);
title ('4-bit slice Image');

subplot (3,3,5);
imshow (bit5);
title ('5-bit slice Image');

subplot (3,3,6);
imshow (bit6);
title ('6-bit slice Image');

subplot (3,3,7);
imshow (bit7);
title ('7-bit slice Image');

subplot (3,3,8);
imshow (bit8);
title ('8-bit slice Image');

% Reconstruct using highest 4 bits
c5 = bit5 * 2^4;
c6 = bit6 * 2^5;
c7 = bit7 * 2^6;
c8 = bit8 * 2^7;

high4 = uint8 (c5+c6+c7+c8);

subplot (3,3,9);
imshow (high4);
title ('Highest 4-bits Image');
```

Code 2: 8-bit Slicing Code continued & reconstuction using highest 4-bits

In analyzing the reconstructed images, the lower bits exhibit increased noise. As you increase in the bit-plane, more information is preserved in the image, eventually reconstructing most of the information of the original image with the high 4-bit reconstructed image.

Problem 3.

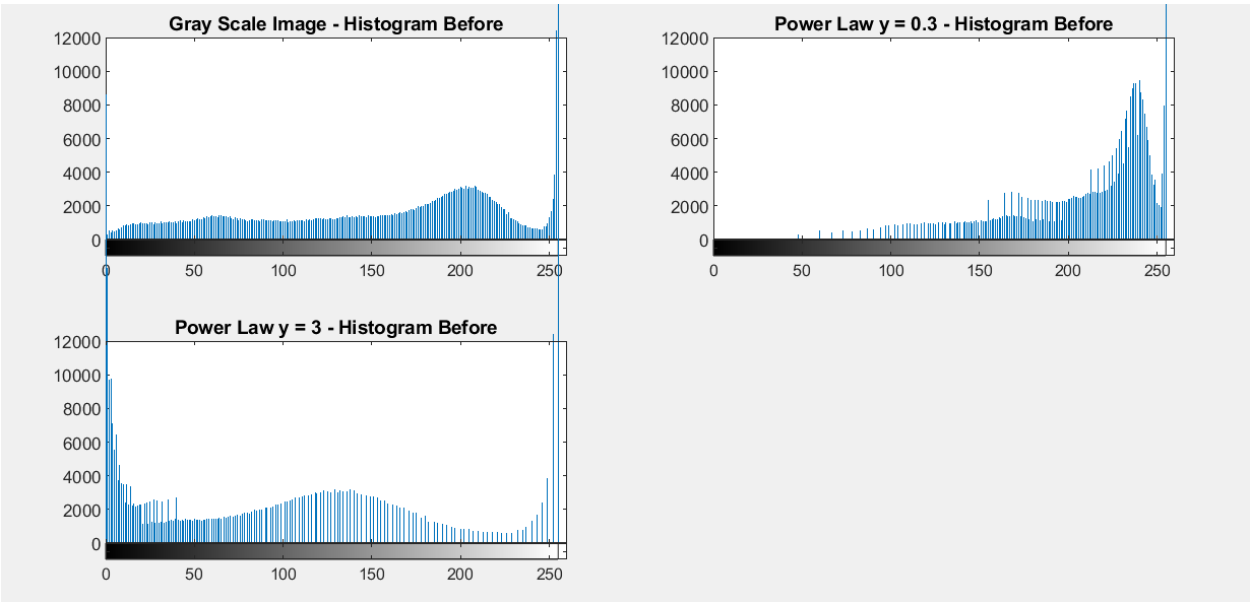

Figure 3: Histogram Before (Gray Scale, Power law (y = 0.3), Power law (y = 3))

```
%% Problem 3.

% Grayscale Histogram Before
figure;
subplot (2,2,1);
imhist(imggray,256);
axis ([0 260 0 12000]);
title ('Gray Scale Image - Histogram Before');

% Power-Law (y = 0.3) Histogram Before
subplot (2,2,2);
imhist(gammalow,256);
axis ([0 260 0 12000]);
title ('Power Law y = 0.3 - Histogram Before');

% Power-Law (y = 3) Histogram Before
subplot (2,2,3);
imhist(gammahigh,256);
axis ([0 260 0 12000]);
title ('Power Law y = 3 - Histogram Before');
```

Code 3: Histogram Before Code Snippet

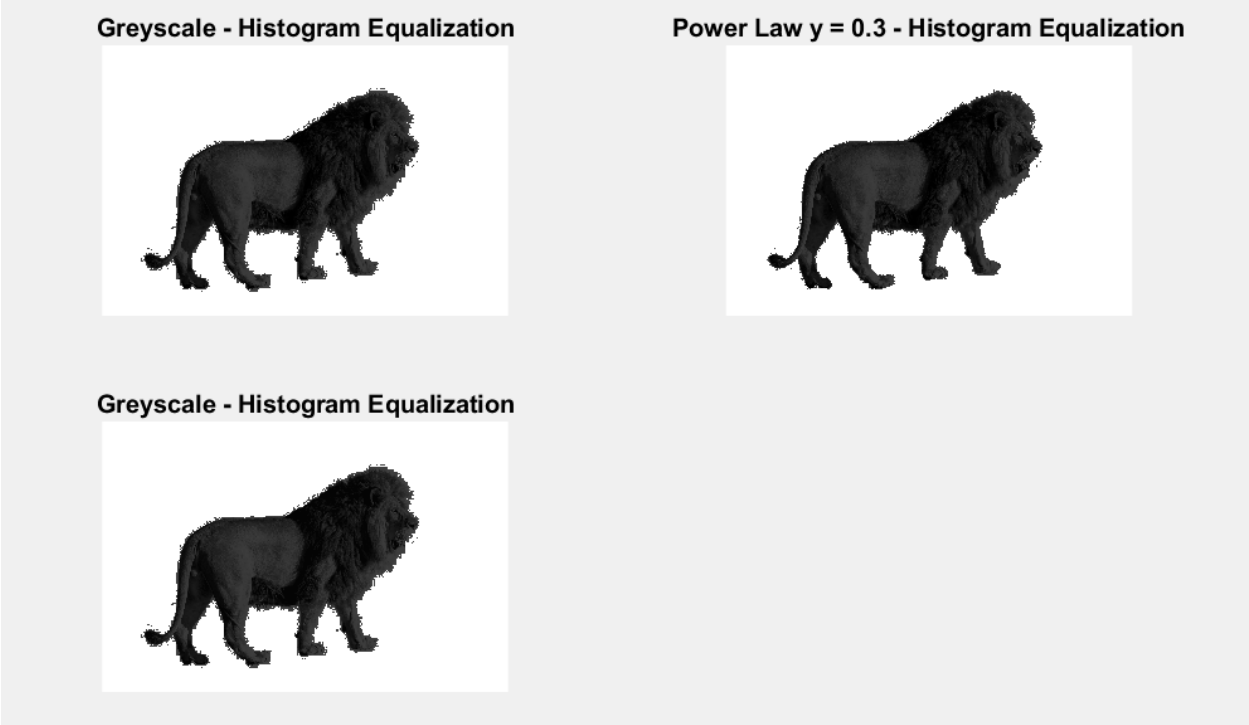

Figure 4: Histogram Equalization Images (Gray Scale, Power law (y = 0.3), Power law (y = 3))

```
% Histogram Equalization of GreyScale
figure;
subplot (2,2,1);
axis ([0 260 0 12000]);
histeq (imggray, 256);
title ('Greyscale - Histogram Equalization');

% Histogram Equalization of Power-Law (y = 0.3)
subplot (2,2,2);
axis ([0 260 0 12000]);
histeq (gammalow,256);
title ('Power Law y = 0.3 - Histogram Equalization');

% Histogram Equalization of Power-Law (y = 3)
subplot (2,2,3);
axis ([0 260 0 12000]);
histeq (gammahigh,256);
title ('Greyscale - Histogram Equalization');
```

Code 4: Histogram Equalization Snippet

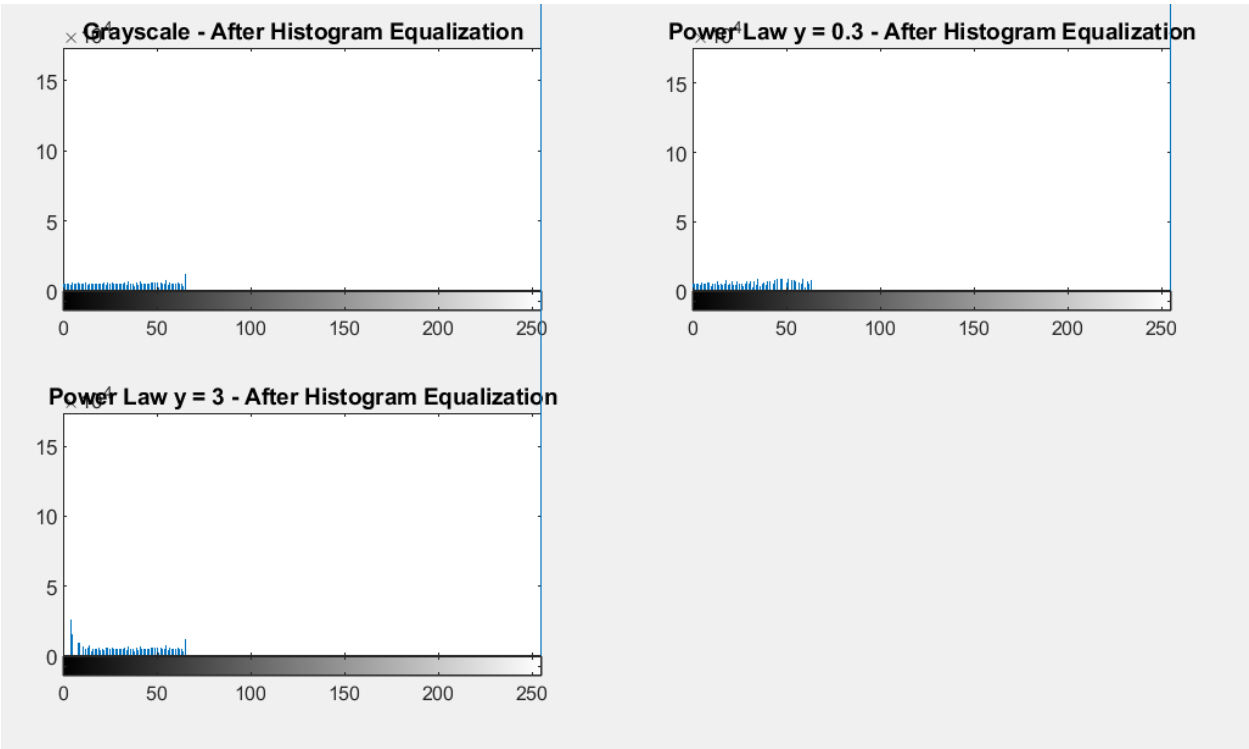

Figure 5: Histogram After (Gray Scale, Power law (y = 0.3), Power law (y = 3))

```
% After Histogram Equalization of Greyscale Image
figure;
subplot (2,2,1);
imhist(histeq(imggray,256),256);
title ('Grayscale - After Histogram Equalization');

% After Histogram Equalization of Power-Law (y = 0.3)
subplot (2,2,2);
imhist(histeq(gammalow,256),256);
title ('Power Law y = 0.3 - After Histogram Equalization');

% After Histogram Equalization of Power-Law (y = 3)
subplot (2,2,3);
imhist(histeq(gammahigh,256),256);
title ('Power Law y = 3 - After Histogram Equalization');
```

Code 5: Histogram After (Grayscale, Power law) Code Snippet

After analyzing the results from before and after histogram equalization, it is apparent that the distribution is more evenly throughout the range of intensities after the equalization process.

Problem 4.

We can find the PDF using the given nk, M, N values from the image and the formula:

$p_r(r_k) = \frac{n_k}{MN}$

$p_r(r_0)$: 790/((64*64)(1)) = 0.19

$p_r(r_1)$: 1023/((64*64)(1))= 0.25

$p_r(r_2)$: 850/((64*64)(1)) = 0.21

$p_r(r_3)$:656/((64*64)(1)) = 0.16

$p_r(r_4)$:329/((64*64)(1)) = 0.08

$p_r(r_5)$:245/((64*64)(1)) = 0.06

$p_r(r_6)$:122/((64*64)(1)) = 0.03

$p_r(r_7)$:81/((64*64)(1)) = 0.02

Histogram equalization allows for a more uniform distribution of intensities to improve visibility and contrast. The following is the histogram image before equalization:

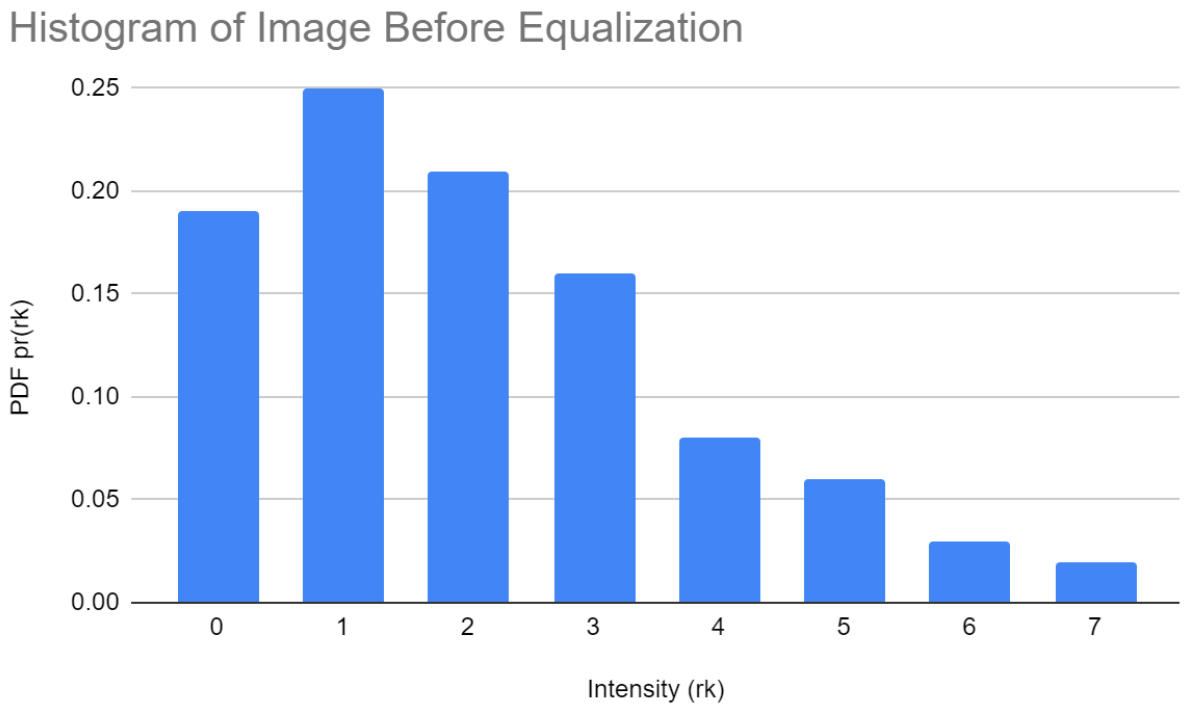

Figure 6: Histogram Before Equalization

Using the probability distribution function of input data $p_r(r_k)$, The histogram equalization process of the image for a discrete case can be computed with the following formula:

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j)$$

$$s_0 = T(r_0) = 7\sum_{j=0}^{0} p_r(r_j) = 7p_r(r_0) = 1.33$$

$$s_1 = T(r_1) = 7\sum_{j=0}^{1} p_r(r_j) = 7p_r(r_0) + 7p_r(r_1) = 3.08$$

$$s_2 = T(r_2) = 7\sum_{j=0}^{2} p_r(r_j) = 7p_r(r_0) + 7p_r(r_1) + 7p_r(r_2) = 4.55$$

$$s_3 = T(r_3) = 7\sum_{j=0}^{3} p_r(r_j) = 7p_r(r_0) + 7p_r(r_1) + 7p_r(r_2) + 7p_r(r_3) = 5.67$$

$$s_4 = T(r_4) = 7\sum_{j=0}^{4} p_r(r_j) = 7p_r(r_0) + 7p_r(r_1) + 7p_r(r_2) + 7p_r(r_3) + 7p_r(r_4) = 6.23$$

$$s_5 = T(r_5) = 7\sum_{j=0}^{5} p_r(r_j) + 7p_r(r_0) + 7p_r(r_1) + 7p_r(r_2) + 7p_r(r_3) + 7p_r(r_4) + 7p_r(r_5) = 6.65$$

Similarly,

$$s_6 = T(r_6) = 7\sum_{j=0}^{5} p_r(r_j) + 7p_r(r_0) + 7p_r(r_1) + 7p_r(r_2) + \ldots\ldots + 7p_r(r_6) = 6.86$$

$$s_7 = T(r_7) = 7\sum_{j=0}^{5} p_r(r_j) + 7p_r(r_0) + 7p_r(r_1) + 7p_r(r_2) + \ldots\ldots + 7p_r(r_7) = 7$$

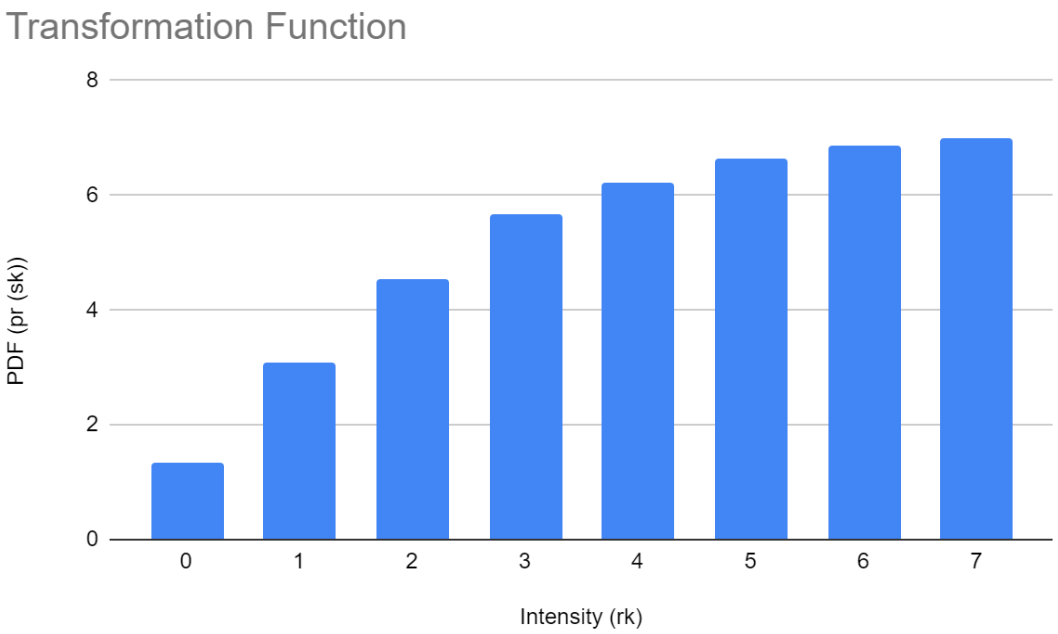

Figure 7: Transformation Function

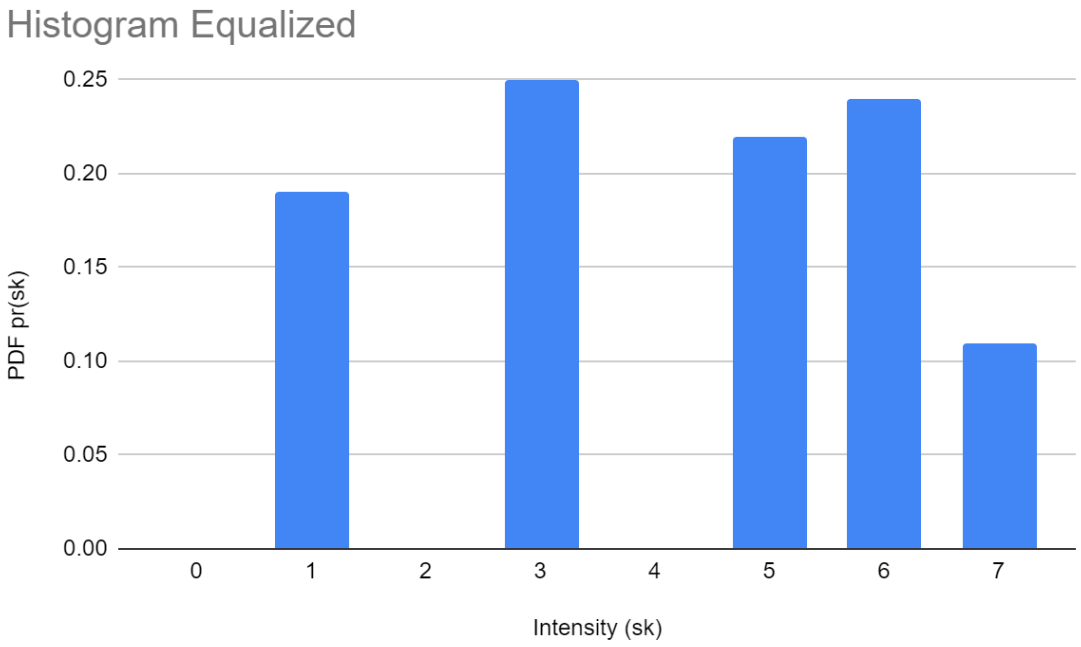

Figure 8: Histogram of Image after Equalization

Problem 5.

From the given image it's apparent that there are 20 points. The PDF of the numbers from the image are as follows:

PDF

$$p_r(r_k) = \frac{n_k}{MN}$$

$p_r(r_k) = 7/20 = 0.35$

$p_r(r_k) = 3/20 = 0.15$

$p_r(r_k) = 2/20 = 0.10$

$p_r(r_k) = 3/20 = 0.15$

$p_r(r_k) = 1/20 = 0.05$

$p_r(r_k) = 1/20 = 0.05$

$p_r(r_k) = 3/20 = 0.15$

The histogram graph is therefore:

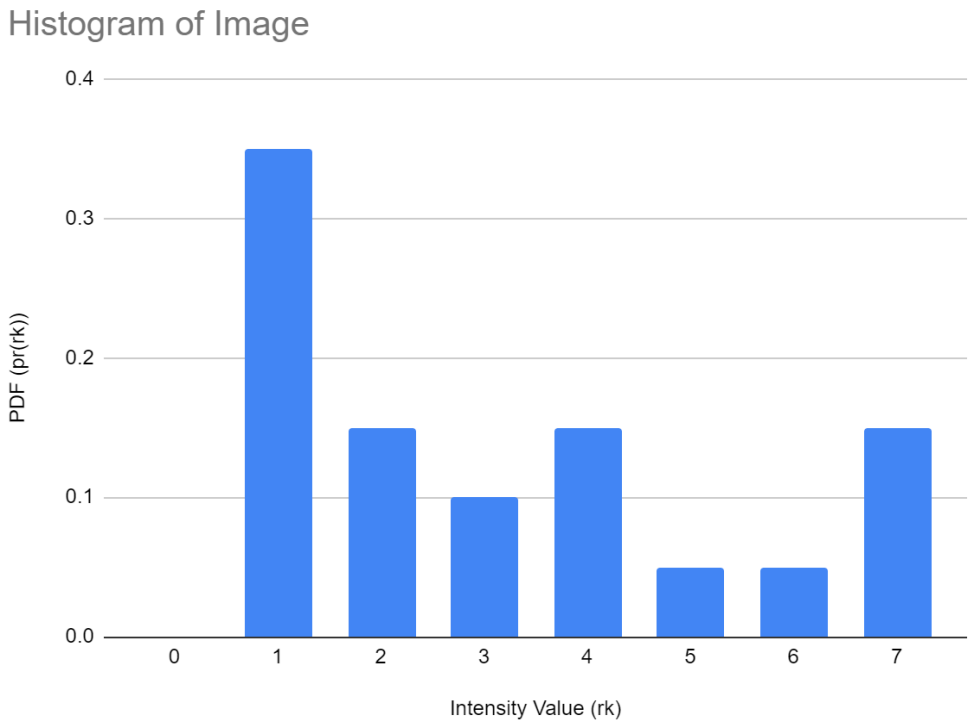

Figure 9: Histogram Before Equalization

Next, we will perform histogram equalization using the equation:

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_k)$$

0

$$s_0 = T(r_k) = 0$$

1

$$s_1 = T(r_k) = 7\sum_{j=0}^{1} p_r(r_1) = 7(7/20) = 2.45$$

2

$$s_2 = T(r_k) = 7\sum_{j=0}^{2} p_r(r_2) = 7(7/20 + 3/20) = 3.5$$

3

$$s_3 = T(r_k) = 7\sum_{j=0}^{1} p_r(r_1) = 7(7/20 + 3/20 + 2/20) = 4.2$$

4

$$s_4 = T(r_k) = 7\sum_{j=0}^{1} p_r(r_1) = 7(7/20 + 3/20 + 2/20 + 3/30) = 5.25$$

5

$$s_5 = T(r_k) = 7\sum_{j=0}^{1} p_r(r_1) = 7(7/20 + 3/20 + 2/20 + 3/30 + 1/20) = 5.6$$

6

$$s_6 = T(r_k) = 7\sum_{j=0}^{1} p_r(r_1) = 7(7/20 + 3/20 + 2/20 + 3/30 + 1/20 + 1/20) = 5.95$$

7

$$s_7 = T(r_k) = 7\sum_{j=0}^{1} p_r(r_1) = 7(7/20 + 3/20 + 2/20 + 3/30 + 1/20 + 1/20 + 3/20) = 7$$

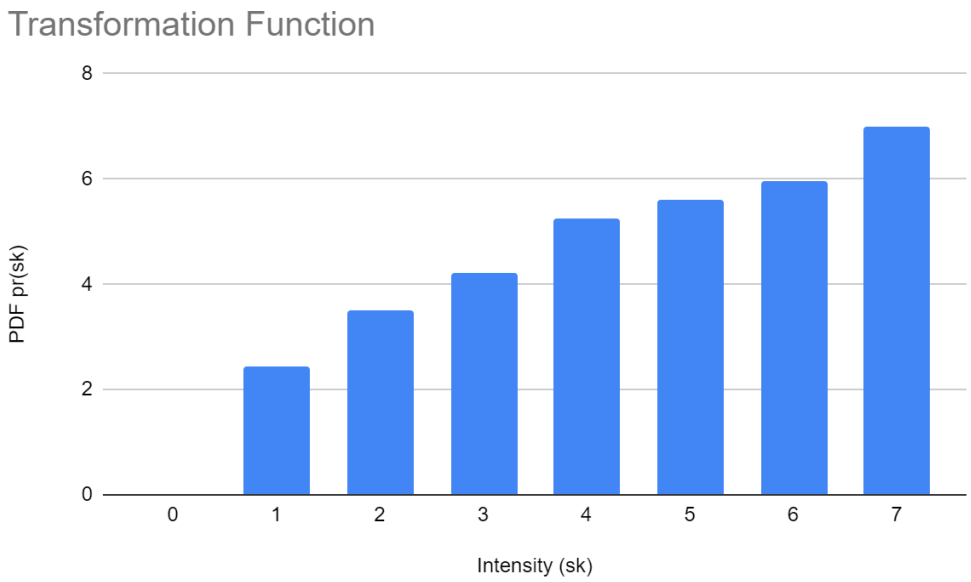

Figure 10: Transformation Function

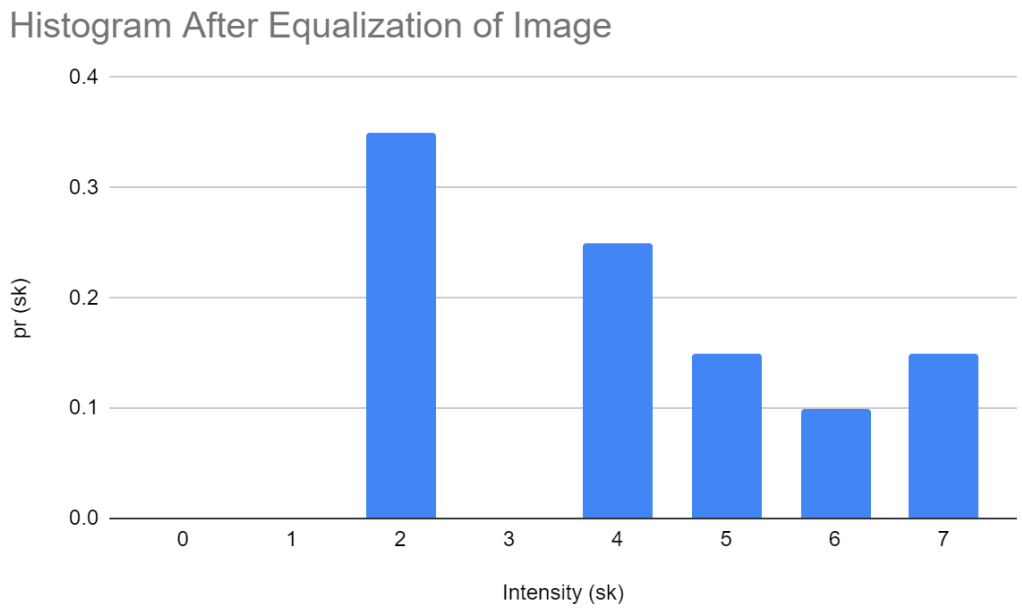

Figure 11: Histogram After Equalization

Part 2.

Step 1:

```
%% Part 2.

% Step 1: Transform Image with Simple Shear

a = 0.56;
T = maketform('affine', [1 0 0; a 1 0; 0 0 1]);
A = imread('yandhi.jpg');
h1 = figure;

imshow(A);
title('Original Image');

blue = [0 0 255]';
R = makeresampler({'cubic','nearest'},'fill');
B = imtransform(A,T,R,'FillValues',blue);
h2 = figure;
imshow(B);
title('Sheared Image');
```

Code 6: Original & Shearing Image

Figure 12: Original Image

Figure 13: Sheared Image

Step 2:

```
%% Step 2: Explore Transformation

[U,V] = meshgrid(0:64:1510,0:64:1510);
[X,Y] = tformfwd(T,U,V);
gray = 0.65 * [1 1 1];

figure(h1);
hold on;
line(U, V, 'Color',gray);
line(U',V','Color',gray);

figure(h2);
hold on;
line(X, Y, 'Color',gray);
line(X',Y','Color',gray);
```

Code 7: Adding Grid lines to Original and Sheared Image

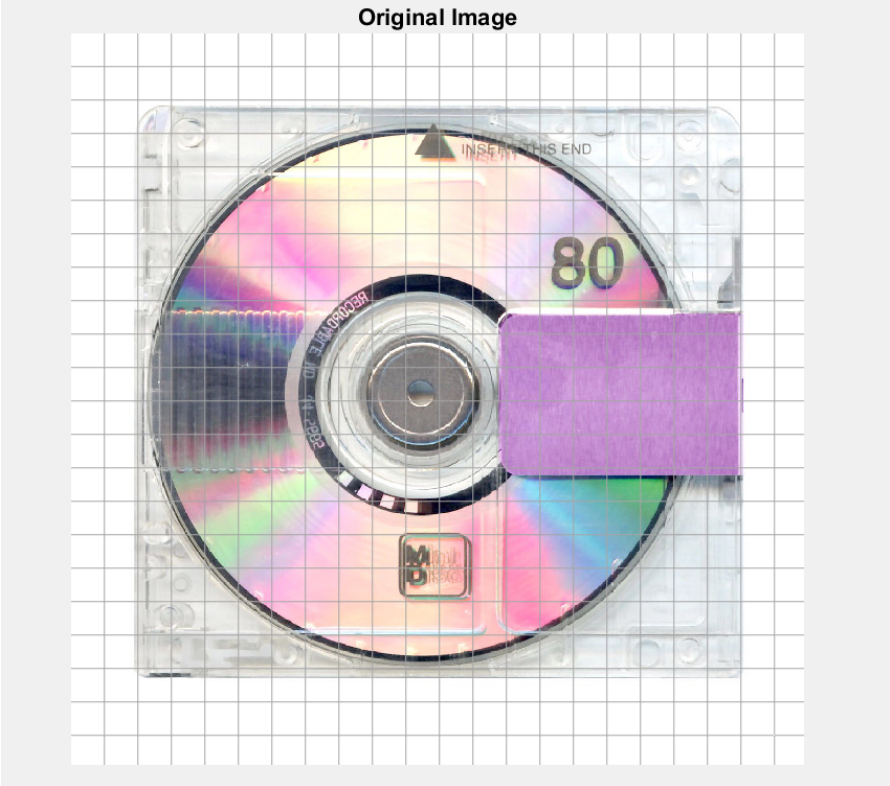

Figure 14: Original Image With Grid Lines

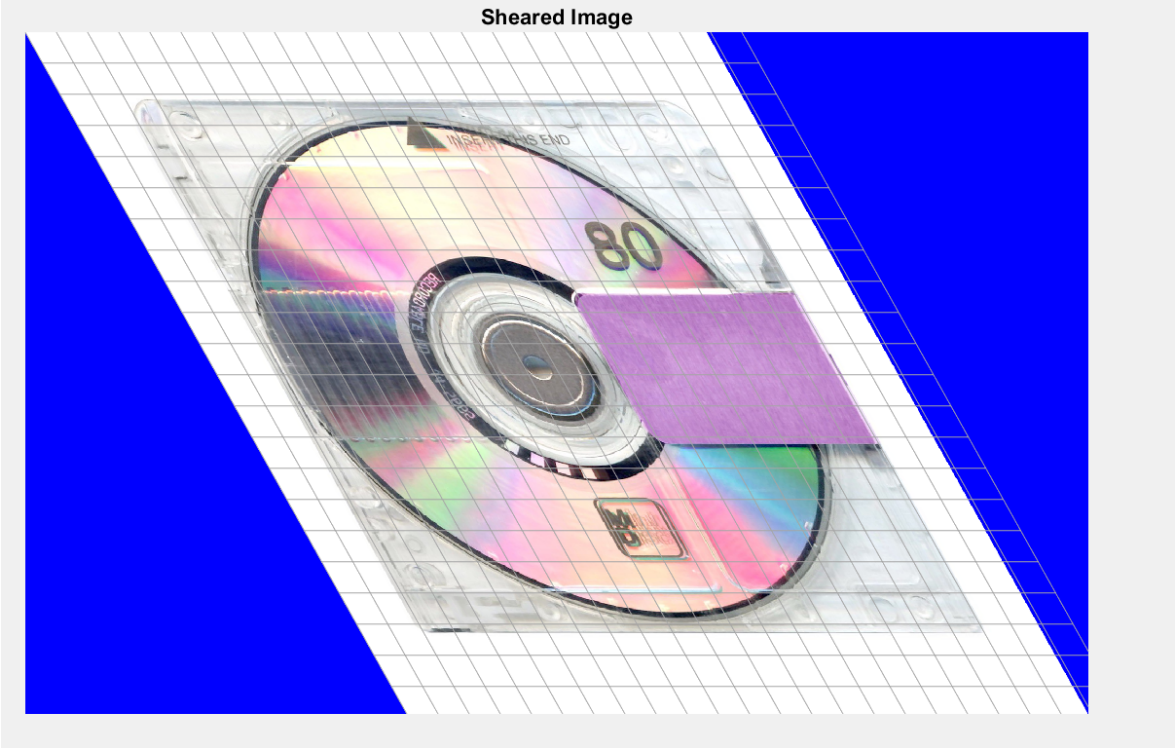

Figure 15: Sheared Image With Grid Lines

Step 3

```
% Step 3: Comparing fill, replicate & bound

% Fill
 R = makeresampler({'cubic','nearest'},'fill');
Bf = imtransform(A,T,R,'XData',[-149 1500],'YData',[-149 1400],...
                 'FillValues',blue);

figure, imshow(Bf);
title('Pad Method = ''fill''');

% Replicate
R = makeresampler({'cubic','nearest'},'replicate');
Br = imtransform(A,T,R,'XData',[-149 1500],'YData', [-149 1400]);

figure, imshow(Br);
title('Pad Method = ''replicate''');

% Bound
R = makeresampler({'cubic','nearest'}, 'bound');
Bb = imtransform(A,T,R,'XData',[-149 1500],'YData',[-149 1400],...
                 'FillValues',blue);
figure, imshow(Bb);
title('Pad Method = ''bound''');
```

Code 8: Applying various Pad Methods to Sheared Image (Fill, Replicate, Bound)

Figure 16: Sheared Image With ‘Fill’ Pad Method

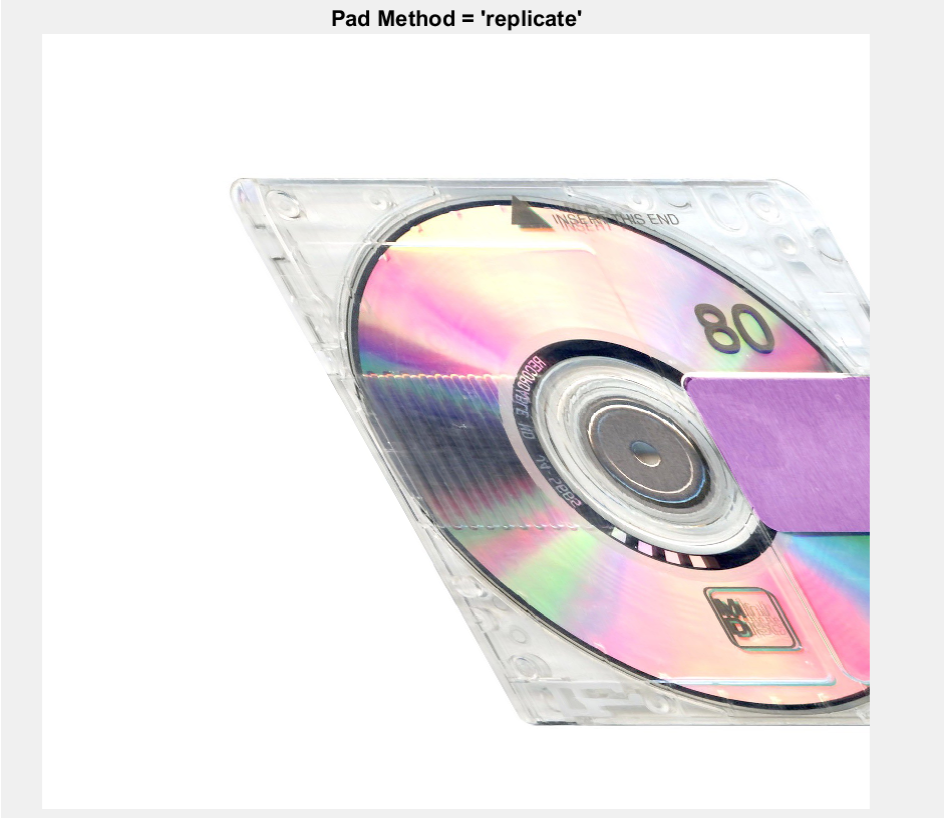

Figure 17: Sheared Image With ‘Replicate’ Pad Method

Figure 18: Sheared Image With ‘Bound’ Pad Method

Step 4

```
% Step 4: Circular & Symmetric

Thalf = maketform('affine',[1 0; a 1; 0 0]/2);

R = makeresampler({'cubic','nearest'},'circular');
Bc = imtransform(A,Thalf,R,'XData',[-149 1500],'YData',[-149 1400],...
                 'FillValues',blue);
figure, imshow(Bc);
title('Pad Method = ''circular''');


R = makeresampler({'cubic','nearest'},'symmetric');
Bs = imtransform(A,Thalf,R,'XData',[-149 1500],'YData',[-149 1400],...
                 'FillValues',blue);
figure, imshow(Bs);
title('Pad Method = ''symmetric''');
```

Code 9: Applying additional Pad Methods to Sheared Image (Circular, Symmetric)

Figure 19: Sheared Image With ‘Circular’ Pad Method

Figure 20: Sheared Image With ‘Circular’ Pad Method

References

MathWorks. (n.d.). Image Processing Toolbox Image Processing Toolbox - MATLAB (mathworks.com)

MathWorks. (n.d.). Padding and Shearing an Image Simultaneously. In Image Processing Toolbox User's Guide (Version R2023b). MathWorks. Padding and Shearing an Image Simultaneously - MATLAB & Simulink Example (mathworks.com)

Toronto Metropolitan University. 2023. Course Content: Week 1. In Computer Vision, CPS 843. Toronto Metropolitan University's Learning Management System. wk1_Introduction - CP8307/CPS843 - Intro to Computer Vision - F2023 (torontomu.ca)

Toronto Metropolitan University. 2023. Course Content: Week 2. In Computer Vision, CPS 843. Toronto Metropolitan University's Learning Management System. wk2 - CP8307/CPS843 - Intro to Computer Vision - F2023 (torontomu.ca)

Toronto Metropolitan University. 2023. Course Content: Week 3. In Computer Vision, CPS 843. Toronto Metropolitan University's Learning Management System. wk3 - CP8307/CPS843 - Intro to Computer Vision - F2023 (torontomu.ca)