SYDE 575 Lab 1
Fundamentals of Image Processing

Group 23: Laura Chambers, ID: 20655640

1 Introduction

The purpose of this lab is to gain experience with fundamental image processing concepts and techniques including Peak Signal to Noise Ratio (PSNR), digital zooming, and point operations.

PSNR is a measure of image quality that compares the relative influence of the desired signal and undesirable noise in an image. It is used to compare the performance of image processing algorithms.

Digital zooming is a method for resizing an image by changing the number of pixel rows and columns. Zooming techniques include nearest neighbour interpolation, bilinear interpolation, and bicubic interpolation, which produce varying results in terms of image quality.

Point operations are a way to transform an image by applying a function to each pixel. Point operations can be used to enhance the histogram and resulting appearance of an image.

The MATLAB code for this lab can be found in the Appendix.

2 Image Quality Measures

The following function is saved in PSNR.m:

```
function PSNR_out = PSNR(f,g)
%Peak Signal to Noise Ratio (PSNR)
%   Outputs PSNR given a reference image f and a test image g.
    [m n] = size(f);
    MAX_f = 255.0;
    error_fg = (double(f)-double(g)).^2;
    MSE = sum(error_fg(:))/(m*n);
    PSNR_out = 10*log10(MAX_f^2/MSE);
end
```

3 Digital Zooming

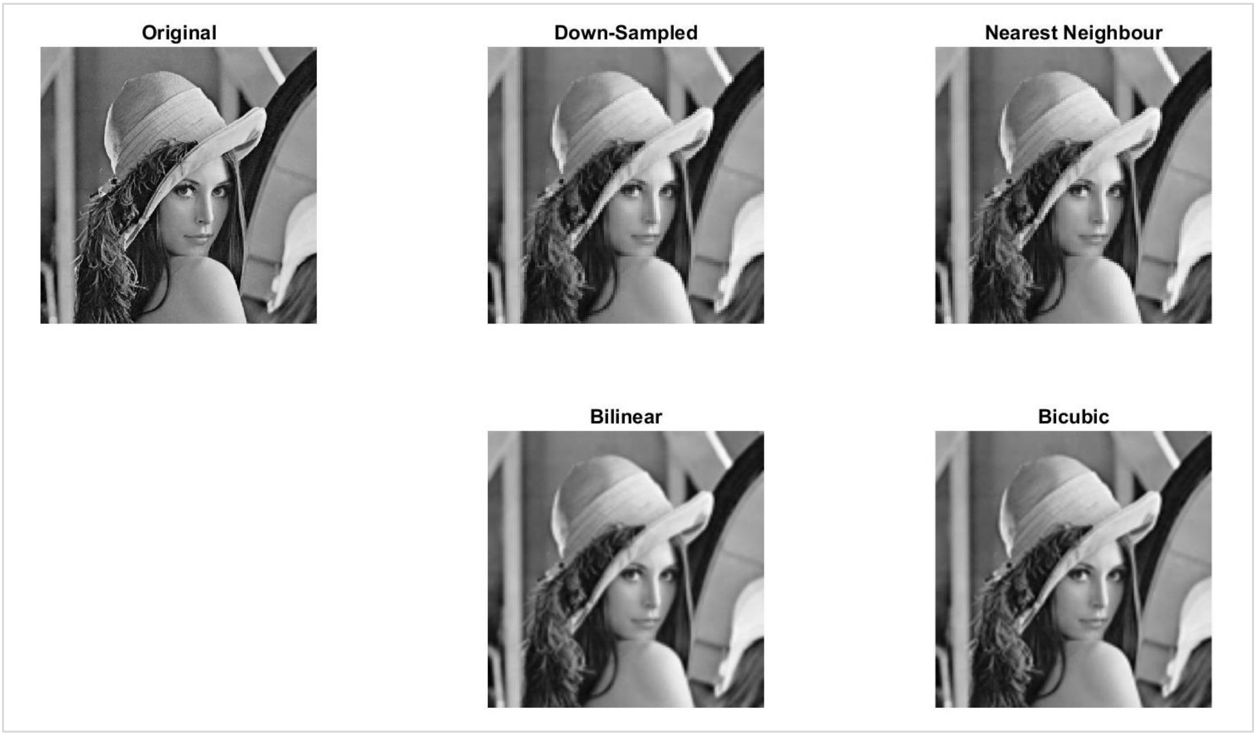

Figure 3-1 Digital zooming results - Lena image

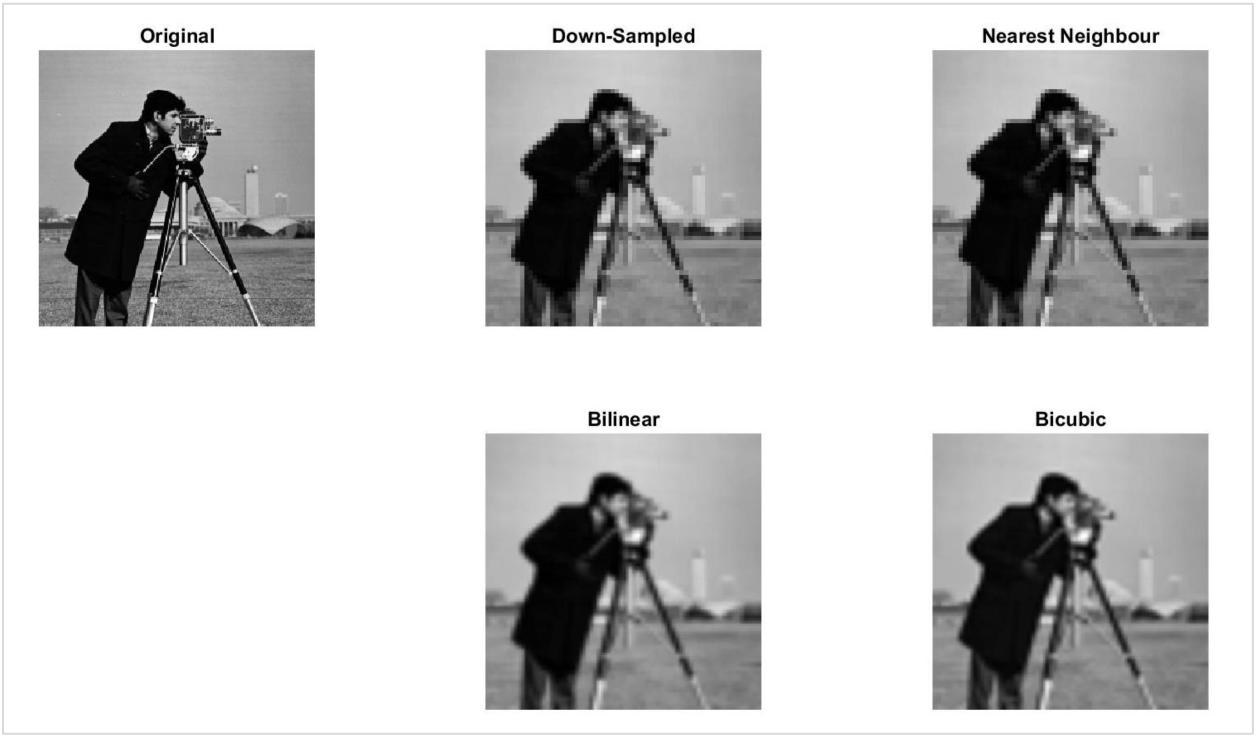

Figure 3-2 Digital zooming results - Cameraman image

Table 3-1 PSNR values between original and up-sampled images

Up-Sampling Method	**PSNR for Lena**	**PSNR for Cameraman**
Nearest Neighbour	26.67	21.54
Bilinear	27.30	21.82
Bicubic	28.08	22.27

1. The up-sampled images for all up-sampling methods appear less sharp than the original images, as can be observed in Figures 3-1 and 3-2. The up-sampled images are less clear and detailed than the originals. The image quality appears to be highest for the bicubic interpolation method and lowest for nearest neighbour interpolation. For example, the nearest neighbour Lena (Figure 3-1) and Cameraman (Figure 3-2) images look pixelated.

2. In order from lowest to highest PSNR values, the up-sampling methods performed as follows: nearest neighbour, bilinear, then bicubic interpolation (Table 3-1). This matches the observed visual quality of the images. The quality is poorest for nearest neighbour, moderate for bilinear, and highest for bicubic interpolation.

   The nearest neighbour method determines the pixel value based on the closest pixel in the original image. It is the fastest method but can result in an undesirable checkerboard effect, seen clearly in the Cameraman image.

   Bilinear interpolation determines the pixel value based on the four closest pixels in the original image. It requires more calculation but avoids the checkerboard effect; however, images may appear blurred.

   Bicubic interpolation determines the pixel value based on 16 surrounding pixels. It is the most intensive and the slowest method of the three, but avoids the checkerboard effect, is less blurry, and preserves the most detail in the image.

3. These digital zooming methods work well for areas of the image that do not have fine detail or straight lines. For example, in the Cameraman image, the less detailed areas of the background sky and the man’s jacket appear acceptable. The more detailed areas of the man’s face and camera, and the straight lines of the tripod look poorer.

4. The up-sampled images of Lena look better than the up-sampled images of Cameraman for all methods. The images of Lena appear less blurry and pixelated. This is reflected in the PSNR values, which are higher for the Lena images than the Cameraman images for all methods. The Lena images look better because the features of the images are larger and curved. For example, Lena’s facial features are closer and larger than the Cameraman’s facial features.

5. The PSNR values show that the bicubic interpolation method produces the clearest results, i.e., the signal is relatively higher than the noise. In contrast, the nearest neighbour interpolation has the lowest PSNR values and produces the lowest quality results. This matches the observed visual quality of the images.

4 Point Operations for Image Enhancement

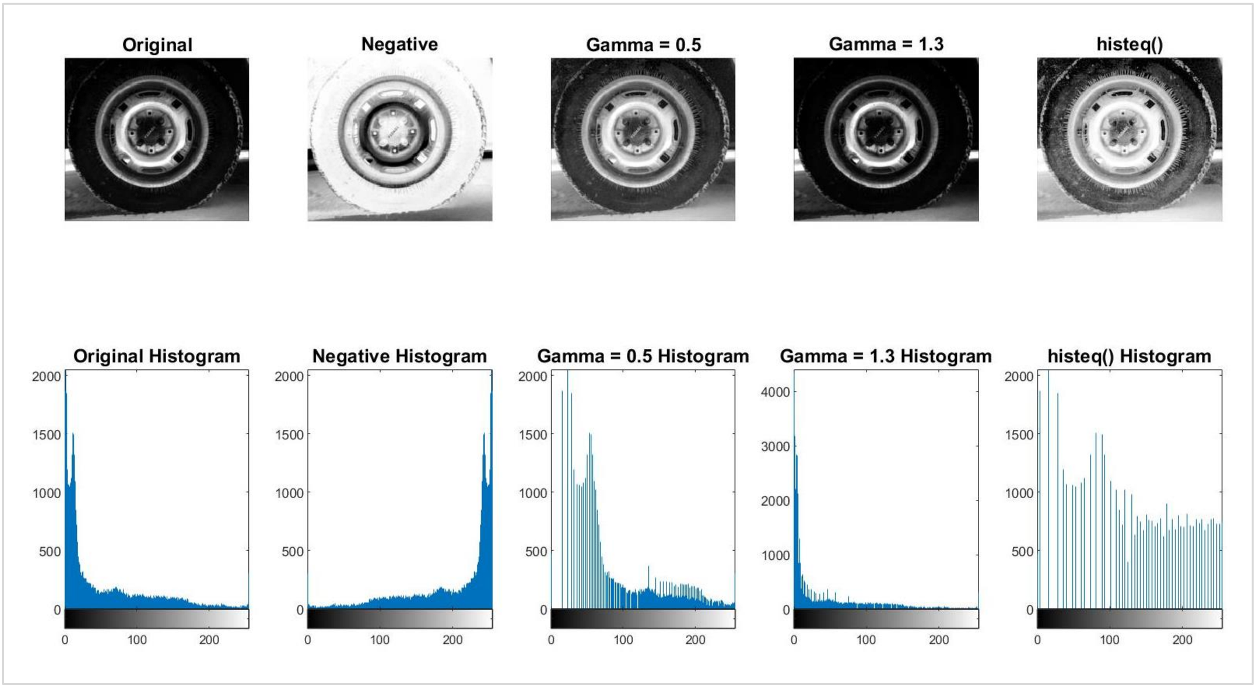

Figure 4-1 Image enhancement results – Tire image

6. The histogram of an image represents the distribution of gray levels in the image. The horizontal axis of the histograms shown in Figure 4-1 is gray levels (0-255) and the vertical axis is the pixel count. Histograms are useful because they enable point processing for enhancing images.

7. A gray level of 0 represents the darkest gray level (i.e., black) and 255 represents the lightest gray level (e.g., white). The histogram of an image shows the intensity of the gray levels within the image. The tire image contains primarily dark gray levels. This is reflected in the histogram, which is skewed toward 0. Most pixels in the image are dark gray level values.

8. The histogram of the negative image is inverted compared to the original image. Pixel values of 0 and 255 in the original image became 255 and 0 in the negative image, respectively. Said differently, all the dark values in the original image became bright in the negative, and all of the bright values became dark. In MATLAB, this is done in either of the following ways:

```
i.    negative = imcomplement(original);
ii.   negative = original - 255;
```

9. The gamma = 0.5 image appears brighter than the original image and the gamma = 1.3 image appears darker than the original image. This is because the transformations changed the distribution of gray levels in the image, as further described in question 10.

10. For gamma values less than 1, the low gray levels are stretched, and the high gray levels are compressed such that the histogram distribution moves to the right. As a result, the transformed image

appears brighter than the original. This can be observed in the Gamma = 0.5 image and histogram in Figure 4-1.

For gamma values greater than 1, the low gray levels are compressed, and the high gray levels are stretched such that the histogram distribution moves to the left. As a result, the transformed mage appears darker than the original, as can be observed in the Gamma = 1.3 image and histogram in Figure 4-1.

11. The original image is already quite dark; therefore, the gamma = 0.5 transformation should be applied to enhance the image. This transformation results in a more balanced distribution of gray levels, which improves the contrast and visibility of detail in the tire.

12. The equalized image appears noticeably brighter than the original image.

13. The histogram of the equalized image is the most evenly distributed of all the transformations. Compared to the original, the number of pixels at each gray level is more similar. The difference between the number of pixels at 0 and 255 is smaller. This can be observed in the image, which shows more mid-level intensities compared to the original.

5 Conclusion

This lab provided hands-on experience with fundamental image processing concepts and techniques and practice using image processing tools in MATLAB.

From this lab, it is evident that PSNR is a measure of image quality, with higher PSNR values corresponding to better looking images.

It was found that different digital zooming techniques result in varying image quality. Nearest neighbour interpolation resulted in the poorest quality image while bicubic interpolation produced the best image.

Finally, it was observed that the histogram of an image reflects its appearance in terms of brightness and darkness. Point operations alter the distribution of gray levels in an image. Images with more evenly distributed gray levels tend to look better with more discernable detail.

Appendix – MATLAB Code

File: lab1_pt3_digital_zooming.m

```
clc;
close all;
clear;
fontSize = 12;

% Load images
lena = imread('lena.tiff');
cameraman = imread('cameraman.tif');

% Convert images to grayscale if not already grayscale
lena_gray = convert2grayscale(lena);
cameraman_gray = convert2grayscale(cameraman);

% DOWN-SAMPLE
% Reduce resolution of images using bilinear interpolation
lena_gray_down = imresize(lena_gray, 0.25, 'bilinear');
cameraman_gray_down = imresize(cameraman_gray, 0.25, 'bilinear');

% UP-SAMPLE
% Perform digital zooming via nearest neighbour interpolation
lena_gray_up_nearest = imresize(lena_gray_down, 4, 'nearest');
cameraman_gray_up_nearest = imresize(cameraman_gray_down, 4,
'nearest');

% Perform digital zooming via bilinear interpolation
lena_gray_up_bilinear = imresize(lena_gray_down, 4, 'bilinear');
cameraman_gray_up_bilinear = imresize(cameraman_gray_down, 4,
'bilinear');

% Perform digital zooming via bicubic interpolation
lena_gray_up_bicubic = imresize(lena_gray_down, 4, 'bicubic');
cameraman_gray_up_bicubic = imresize(cameraman_gray_down, 4,
'bicubic');

% Plot images
figure;
set(gcf, 'Position', get(0,'Screensize'));
subplot(2, 3, 1);
imshow(lena_gray, []);
title('Original', 'FontSize', fontSize);
subplot(2, 3, 2);
imshow(lena_gray_down, []);
title('Down-Sampled', 'FontSize', fontSize);
subplot(2, 3, 3);
imshow(lena_gray_up_nearest, []);
title('Nearest Neighbour', 'FontSize', fontSize);
subplot(2, 3, 5);
imshow(lena_gray_up_bilinear, []);
title('Bilinear', 'FontSize', fontSize);
subplot(2, 3, 6);
imshow(lena_gray_up_bicubic, []);
title('Bicubic', 'FontSize', fontSize);

figure;
set(gcf, 'Position', get(0,'Screensize'));
subplot(2, 3, 1);
imshow(cameraman_gray, []);
title('Original', 'FontSize', fontSize);
subplot(2, 3, 2);
imshow(cameraman_gray_down, []);
title('Down-Sampled', 'FontSize', fontSize);
subplot(2, 3, 3);
imshow(cameraman_gray_up_nearest, []);
title('Nearest Neighbour', 'FontSize', fontSize);
subplot(2, 3, 5);
imshow(cameraman_gray_up_bilinear, []);
```

```
title('Bilinear', 'FontSize', fontSize);
subplot(2, 3, 6);
imshow(cameraman_gray_up_bicubic, []);
title('Bicubic', 'FontSize', fontSize);

% Compute PSNR between original and up-sampled images
PSNR_lena_nearest = PSNR(lena_gray, lena_gray_up_nearest)
PSNR_lena_bilinear = PSNR(lena_gray, lena_gray_up_bilinear)
PSNR_lena_bicubic = PSNR(lena_gray, lena_gray_up_bicubic)

PSNR_cameraman_nearest = PSNR(cameraman_gray,
cameraman_gray_up_nearest)
PSNR_cameraman_bilinear = PSNR(cameraman_gray,
cameraman_gray_up_bilinear)
PSNR_cameraman_bicubic = PSNR(cameraman_gray,
cameraman_gray_up_bicubic)
```

File: lab1_pt4_point_operations.m

```
clc;
close all;
clear;
fontSize = 12;

% Load image
tire = imread('tire.tif');

% Plot image and histogram
figure;
set(gcf, 'Position', get(0,'Screensize'));
subplot(2,5,1);
imshow(tire);
title('Original', 'FontSize', fontSize);
subplot(2,5,6);
imhist(tire);
title('Original Histogram', 'FontSize', fontSize);

% Plot negative image and histogram
tire_negative = imcomplement(tire);
subplot(2,5,2);
imshow(tire_negative);
title('Negative', 'FontSize', fontSize);
subplot(2,5,7);
imhist(tire_negative);
title('Negative Histogram', 'FontSize', fontSize);

% Power law transformation gamma = 0.5
gamma = 0.5;
tire_gamma1 = imadjust(tire, [], [], gamma);
subplot(2,5,3);
```

```
imshow(tire_gamma1);
title('Gamma = 0.5', 'FontSize', fontSize);
subplot(2,5,8);
imhist(tire_gamma1);
title('Gamma = 0.5 Histogram', 'FontSize', fontSize);

% Power law transformation gamma = 1.3
gamma = 1.3;
tire_gamma2 = imadjust(tire, [], [], gamma);
subplot(2,5,4);
imshow(tire_gamma2);
title('Gamma = 1.3', 'FontSize', fontSize);
subplot(2,5,9);
imhist(tire_gamma2);
title('Gamma = 1.3 Histogram', 'FontSize', fontSize);

% Histogram equalization
tire_eq = histeq(tire);
subplot(2,5,5);
imshow(tire_eq);
title('histeq()', 'FontSize', fontSize);
subplot(2,5,10);
imhist(tire_eq);
title('histeq() Histogram', 'FontSize', fontSize);
```

File: PSNR.m

```
function PSNR_out = PSNR(f,g)
%Peak Signal to Noise Ratio (PSNR)
%   Outputs PSNR given a reference image f and a test image g
    [m n] = size(f);
    MAX_f = 255.0;
    error_fg = (double(f)-double(g)).^2;
    MSE = sum(error_fg(:))/(m*n);
    PSNR_out = 10*log10(MAX_f^2/MSE);
end
```

File: convert2grayscale.m

```
function image_gray = convert2grayscale(image)
% Check if the image is colour and convert to grayscale if it is
    [m, n, numberOfColorChannels] = size(image);
    if numberOfColorChannels > 1
        % If it's a colour image, convert to gray
        image_gray = rgb2gray(image);
    else
        % Already gray, don't convert
        image_gray = image;
end
```