Digital Image Processing Laboratory

Experiment Report

Experiment Title Basic Image Operation & Image Transformation

Student's Name _____

Student's ID _____

Class _____

Date handed in _____

International school, Jinan University

A. Objectives

(1) To know how to manipulate images.
(2) To be able to implement basic image transformations in Matlab.
(3) To be able to use intensity transformations to enhance an image.

B. Technique

In this project, the image **tire.tif**, **rice.png**, **kids.tif** and **spine.tif** will be used.
(1) Do the basic operation on the **tire** and **rice** image.
(2) Zooming and shrinking images.
(3) Simple intensity transformations on images.

C. Experiment Content

(1) Basic Image Operation

1. Read the images **tire.tif**, **rice.png** given in the folder, and show them in one and two figures respectively. (referenced function: imread, imshow, figure, subplot)

Figure1: Original **tire.tif**

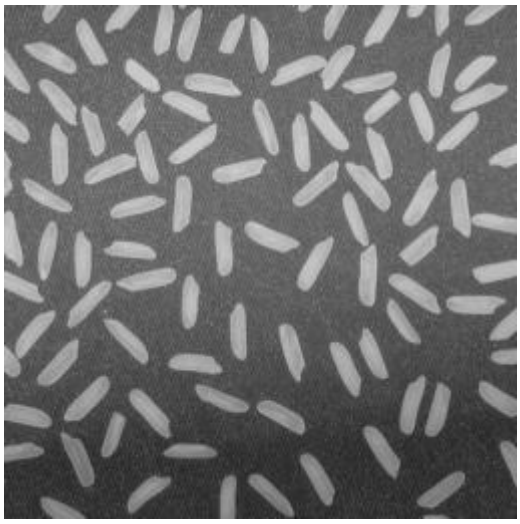

Figure 2: Original **rice.png**

2. In matlab, observe the images information from the workspace Panel
3. Use size, imfinfo, whos, etc functions to obtain image information respectively.
4. Add title to the images. (referenced function: title)
5. Implement the following codes in the M-file Editor

```
% Read the images

tire_img = imread('tire.tif');

rice_img = imread('rice.png');


% Display the tire image in one figure

figure; imshow(tire_img); title('Tire Image');

figure; imshow(rice_img); title('Rice Image');
```

```
% Display both images in a separate figure using subplots
figure;subplot(1,2,1); imshow(tire_img); title('Tire Image');subplot(1,2,2);
imshow(rice_img); title('Rice Image');

% Using size
tire_size = size(tire_img);
rice_size = size(rice_img);

% Using imfinfo
tire_info = imfinfo('tire.tif');
rice_info = imfinfo('rice.png');

% Using whos
whos tire_img
whos rice_img

f = imread('tire.tif');
figure; imshow(f);set(figure(4),'NumberTitle','off','Name','my first image')

% Create the folder if it doesn't exist
folder_path = 'imagetest';
if ~exist(folder_path, 'dir')
    mkdir(folder_path);
end

% Save the image
imwrite(tire_img, fullfile(folder_path, 'tire.tif'));
```

observe the result, and determine the function of set()

6. Save **tire.tif** to the directory "./imagetest". If the folder does not exist, please create this new folder first. (referenced function: imwrite)

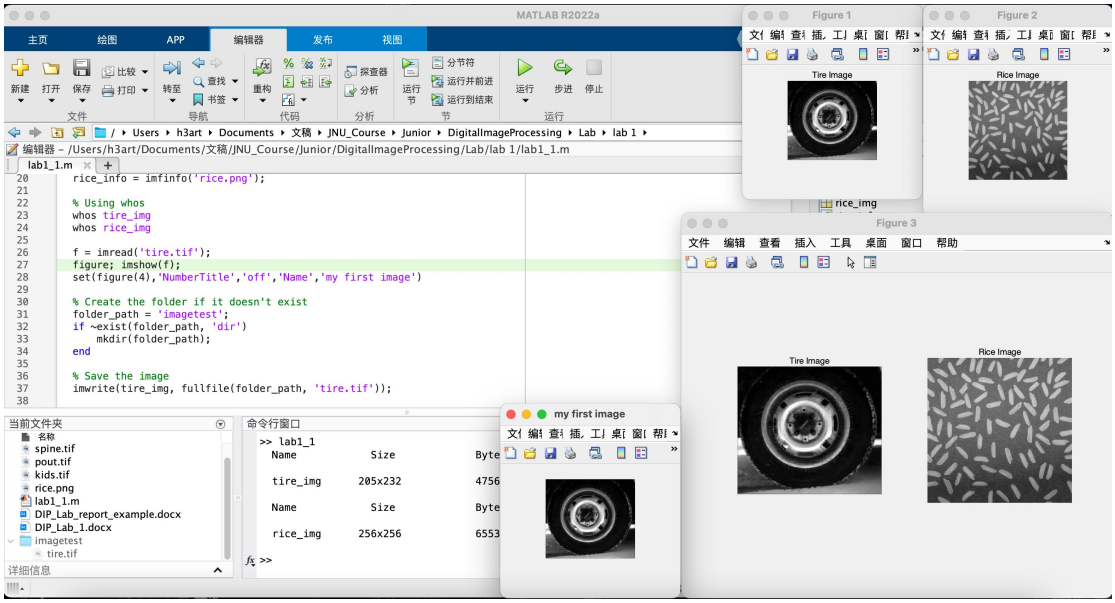

Figure 3: Final Result

(2) **Zooming and Shrinking Images**

1. Write a program capable of shrinking(缩放) the image **kids.tif** by nearest, bilinear, and bicubic interpolation respectively. (referenced function: imresize)

Figure 4: Shrinking the **kids.tif** in different methods

2. Use your program to zoom the images in step 1 back to original size. Explain the reasons for their differences. (referenced function: imresize)

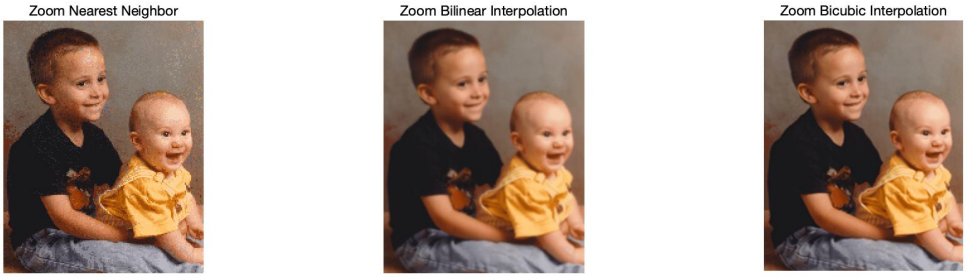

Figure 5: Zooming the the images in step 1 back to original size

The differences in the zoomed images are due to the interpolation method used.

Nearest neighbor interpolation can lead to a blocky and jagged appearance, especially when zooming back to the original size. Bilinear and bicubic interpolations provide smoother transitions between pixels, with bicubic providing the best quality at the cost of computational efficiency.

3. Crop the image **kids.tif** to the size specified by you.

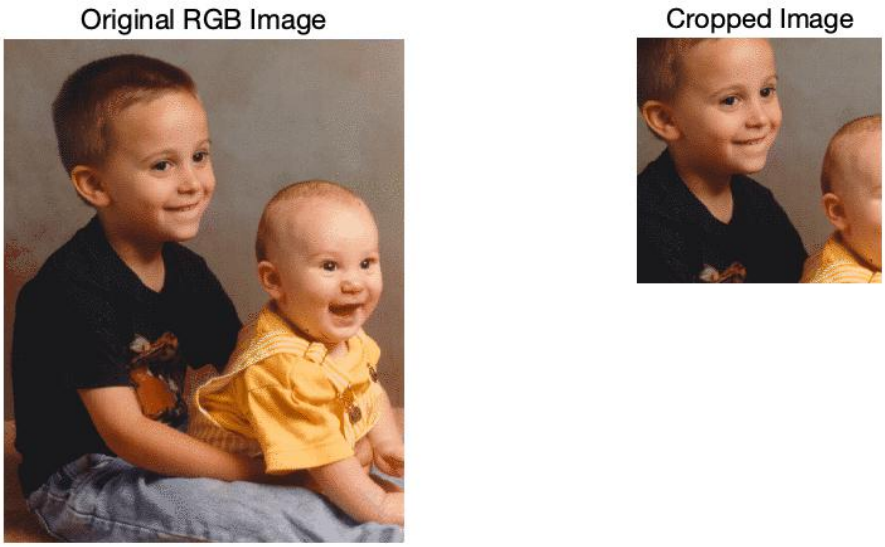

Figure 6: Crop the image **kids.tif** to the size [x, y, width, height] = [50, 50, 200, 200]

4. Rotate the image **kids.tif** by the degree specified by you.

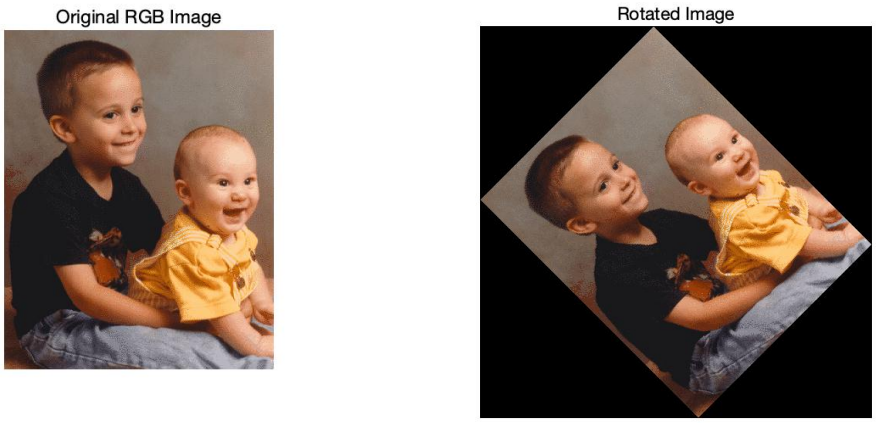

Figure 7: Rotate the image **kids.tif** by 45°

The MATLAB code in this experiment shows as follows:

```
% Read the image
[originalImage, colormap] = imread('kids.tif');
originalRGBImage = ind2rgb(originalImage, colormap);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%SHRINK%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Specify the scale factor for shrinking
scaleFactor = 0.5;

% Perform nearest neighbor interpolation
shrinkNearest = imresize(originalRGBImage, scaleFactor, 'nearest');

% Perform bilinear interpolation
shrinkBilinear = imresize(originalRGBImage, scaleFactor, 'bilinear');

% Perform bicubic interpolation
shrinkBicubic = imresize(originalRGBImage, scaleFactor, 'bicubic');

% Display the images
figure, imshow(originalRGBImage), title('Original RGB Image');
figure, imshow(shrinkNearest), title('Shrink using Nearest Neighbor');
figure, imshow(shrinkBilinear), title('Shrink using Bilinear Interpolation');
figure, imshow(shrinkBicubic), title('Shrink using Bicubic Interpolation');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%ZOOM BACK%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Zoom the images back to original size
zoomNearest = imresize(shrinkNearest, 1/scaleFactor, 'nearest');
zoomBilinear = imresize(shrinkBilinear, 1/scaleFactor, 'bilinear');
zoomBicubic = imresize(shrinkBicubic, 1/scaleFactor, 'bicubic');

% Display the zoomed images
```

```
figure, imshow(zoomNearest), title('Zoom Nearest Neighbor');

figure, imshow(zoomBilinear), title('Zoom Bilinear Interpolation');

figure, imshow(zoomBicubic), title('Zoom Bicubic Interpolation');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%CROP%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Define the crop rectangle [x, y, width, height]

cropRect = [50, 50, 200, 200];


% Crop the image

croppedImage = imcrop(originalRGBImage, cropRect);


% Display the cropped image

figure, imshow(croppedImage), title('Cropped Image');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%ROTATE%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Specify the rotation angle in degrees

rotationAngle = 45;


% Rotate the image

rotatedImage = imrotate(originalRGBImage, rotationAngle);


% Display the rotated image

figure, imshow(rotatedImage), title('Rotated Image');
```

(3) **Simple Intensity Transformations**

1. Use the log transformation to enhance the image **spine.tif**(reference function: log):

$$s = c \log(1 + r) \quad (1)$$

2. Use a power-law transformation of the form shown as follows to enhance the image **spine.tif** (reference algorithm operator: ^)

$$s = cr^{\gamma} \quad (2)$$

The only free parameter is c in equation (1), but in equation (2) there are two parameters, c and γ for which values have to be selected. As in most enhancement tasks, experimentation is a must.

The objective of this experiment is to obtain the best visual enhancement possible with the methods in 1 and 2. Once (according to your judgment) you have the best

visual result for each transformation, explain the reasons for the major differences between them.

3. I observed the impact of parameters on image processing by setting different parameters and uniformly outputting the processing results:

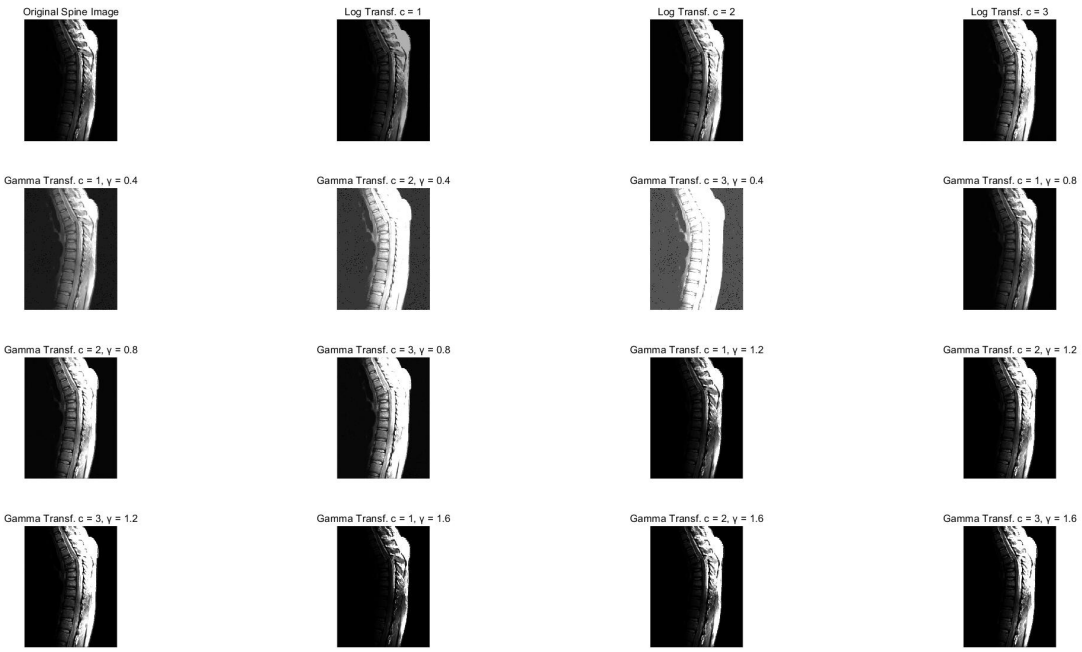

Figure 8: All results using log and Gamma transformation with different parameters

Therefore, I chose the best visual enhancement result with the method 1 and 2, this is a comparison between them and the original image:

Figure 9: Comparison between the transformed images and the original image

The selection criteria for the best visual enhancement effects are based on whether the image is more balanced in light and dark and can see more details.

It can be seen that in method 1, when the parameter c changes, the overall brightness of the image increases as c increases, but compared to the case of c = 3, the image

with c = 2 retains more bright parts. details, so I choose c = 2 as the best image enhancement.

The two parameters introduced in method 2, γ and c, both have a certain impact on image enhancement. Through formula analysis and experimental judgment, it can be found that γ has an impact on the contrast of the image. The larger the γ, the higher the contrast of the image, while c still It transforms the brightness of the image. The larger c, the brighter the image. Therefore, I finally chose the result of γ = 0.4, c = 1, which greatly enhanced the visible details of the image, showing much more details than the original image.

The code to complete this experiment is as follows:

```
% Read the image
spineImage = imread('spine.tif');
spineImageDouble = im2double(spineImage); % Convert to double for processing

% Initialize figure
figure;

% Display original image
subplot(4, 4, 1);
imshow(spineImage);
title('Original Spine Image');

% Use the log transformation to enhance
for c = 1:3
    logTransformed = c * log(1 + spineImageDouble);

    % Display the log transformed images
    subplot(4, 4, c+1);
    imshow(logTransformed);
    title(['Log Transf. c = ', num2str(c)]);
end
```

```
% Use a power-law transformation

gammaValues = 0.4:0.4:1.6;

for gammaIndex = 1:length(gammaValues)

    gamma = gammaValues(gammaIndex);

    for c = 1:3

        gammaTransformed = c * (spineImageDouble .^ gamma);


        % Display the gamma transformed images

        subplotIndex = 4 + (gammaIndex - 1) * 3 + c;

        subplot(4, 4, subplotIndex);

        imshow(gammaTransformed);

        title(['Gamma Transf. c = ', num2str(c), ', γ = ', num2str(gamma)]);

    end

end
```

D. Conclusions

Engaging in basic image operations, I mastered the art of reading, displaying, and extracting vital information from images. I learned the significance of adding titles and annotations, which not only made the results more interpretable but also enriched my understanding of the images' characteristics.

The exploration of image zooming and shrinking techniques was particularly fascinating. Experimenting with different interpolation methods, I observed their unique impacts on the image quality. I realized that while nearest neighbor interpolation might result in a blocky appearance, bilinear and bicubic interpolations provide smoother transitions, preserving the image's integrity even after transformation.

The intensity transformations were a highlight of this lab, as I manipulated image brightness and contrast through log and power-law transformations. It was intriguing to see how different parameter values could enhance the visual appeal of an image. Through trial and error, I developed a keen sense of how to choose optimal parameters to achieve the desired image enhancement.