**ELEC319 Image Processing**

**Assignment 2**

**Appendix: Submission Template:Student name:** Yucheng Lin

**Student ID:** 201447973

**Question 1:**

Part I (MATLAB code (copied and pasted as text) with comments (%)):

% Question 1: Image Noise Filtering

clear;

% Read image cameraman.tif

image = imread('cameraman.tif');

% Convert uint8 type to double type, otherwise the statistics cannot be calculated

im = im2double(image);

% 1 -Add 'Poisson' noise on the image 'image'

im\_np = imnoise(image,'poisson');

% Convert uint8 type to double type, otherwise the statistics cannot be calculated

d\_np = im2double(im\_np);

% 2 -Calculate the signal-to-noise ratio (SNR) of the noisy image in dB

% The signal-to-noise ratio (SNR) of the image is divided by the variance of the original image by the variance of the noise

% Image with only noise part

noise\_im\_np = imsubtract(d\_np,im);

% Find the image mean

avg\_im = mean2(im);

[m,n] = size(im);

% Find the sum of squares of all pixels and the mean

s\_im = 0;

for x = 1:m

for y = 1:n

s\_im = s\_im + (im(x,y)-avg\_im)^2;

end

end

% Find the variance of the original image

v\_im = s\_im/(m\*n);

% Find the noise mean

avg\_np = mean2(noise\_im\_np);

[m,n] = size(noise\_im\_np);

% Find the sum of squares of all pixels and the mean

s\_np = 0;

for x = 1:m

for y = 1:n

s\_np = s\_np + (noise\_im\_np(x,y)-avg\_np)^2;

end

end

% Find the variance of the noise

v\_np = s\_np/(m\*n);

% Find the signal-to-noise ratio (SNR) of the noise image

SNR\_np = 10\*log10(double(v\_im/v\_np));

% 3 -Use Gaussian filtering to denoise the image

im\_g = imgaussfilt(im\_np);

% Convert uint8 type to double type, otherwise the statistics cannot be calculated

d\_g = im2double(im\_g);

% Image with only noise part

noise\_im\_g = imsubtract(d\_g,im);

% Find the noise mean

avg\_g = mean2(noise\_im\_g);

[m,n] = size(noise\_im\_g);

% Find the sum of squares of all pixels and the mean

s\_g = 0;

for x = 1:m

for y = 1:n

s\_g = s\_g + (noise\_im\_g(x,y)-avg\_g)^2;

end

end

% Find the variance of the noise

v\_g = s\_g/(m\*n);

% Find the signal-to-noise ratio (SNR) of the noise image

SNR\_g = 10\*log10(double(v\_im/v\_g));

% 4 -Use Wiener filter to denoise the image

im\_w = wiener2(im\_np,[3,3]);

% Convert uint8 type to double type, otherwise the statistics cannot be calculated

d\_w = im2double(im\_w);

% Image with only noise part

noise\_im\_w = imsubtract(d\_w,im);

% Find the noise mean

avg\_w = mean2(noise\_im\_w);

[m,n] = size(noise\_im\_w);

% Find the sum of squares of all pixels and the mean

s\_w = 0;

for x = 1:m

for y = 1:n

s\_w = s\_w + (noise\_im\_w(x,y)-avg\_w)^2;

end

end

% Find the variance of the noise

v\_w = s\_w/(m\*n);

% Find the signal-to-noise ratio (SNR) of the noise image

SNR\_w = 10\*log10(double(v\_im/v\_w));

% 5 -Plotting Results

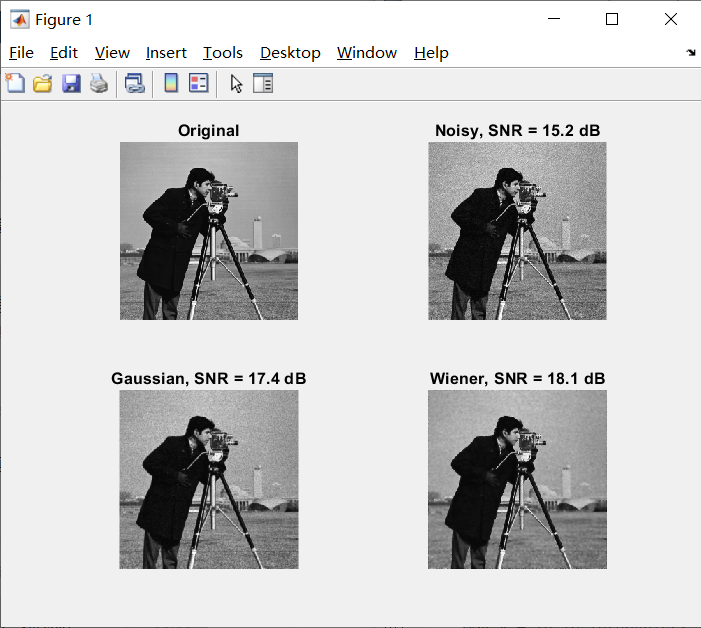
figure, subplot(2,2,1), imshow(image); title('Original');

subplot(2,2,2), imshow(im\_np); title(['Noisy, SNR = ',num2str(SNR\_np,3),' dB']);

subplot(2,2,3), imshow(im\_g); title(['Gaussian, SNR = ',num2str(SNR\_g,3),' dB']);

subplot(2,2,4), imshow(im\_w); title(['Wiener, SNR = ',num2str(SNR\_w,3),' dB']);

Part II (results and comments):



**Figure 1:** The result of Q1

In Fig. 1, Gaussian filtering and Wiener filtering are respectively used to denoise the noisy image. The higher the signal-to-noise ratio, the less noise. Therefore, Wiener filtering is better than Gaussian filtering.

**Question 2:**

Part I (MATLAB code (copied and pasted as text) with comments (%)):

% Question 2: Image Enhancement with histogram equalisation

clear;

% 1 -Read image board.tif

image = imread('board.tif');

% 2 -Convert the image from RGB to a format that separates intensity values from color components

% Convert RGB image to YCbCr image

% Extract y, cb and cr image components

ycc = rgb2ycbcr(image);

y = ycc(:, :, 1);

cb = ycc(:, :, 2);

cr = ycc(:, :, 3);

% 3 -Histogram equalization of the converted image

% Perform histogram equalization on the intensity plane y

h\_y = histeq(y);

% Combine h\_y, cb and cr components into an RGB image

h\_ycc = cat(3,h\_y,cb,cr);

% 4 -Convert the generated image back to RGB

RGB = ycbcr2rgb(h\_ycc);

% 5 -Plotting Results

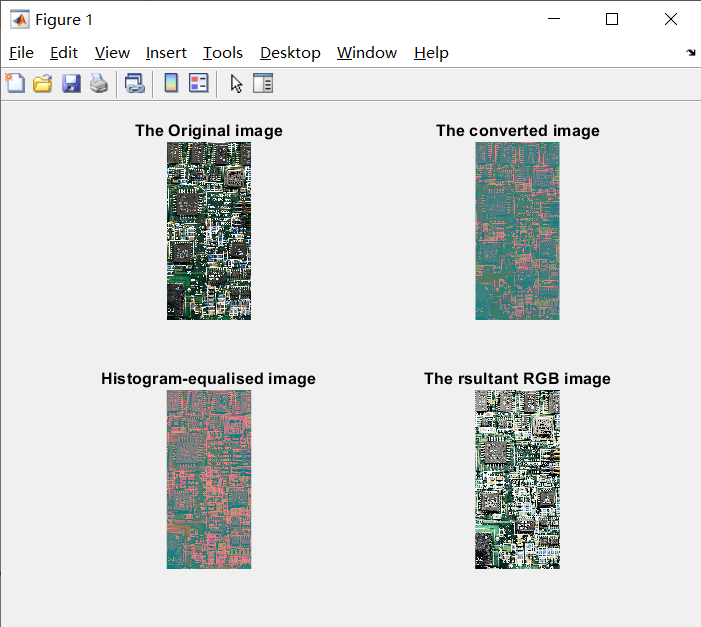
figure, subplot(2,2,1), imshow(image); title('The Original image');

subplot(2,2,2), imshow(ycc); title('The converted image');

subplot(2,2,3), imshow(h\_ycc); title('Histogram-equalised image');

subplot(2,2,4), imshow(RGB); title('The rsultant RGB image');

Part II (results and comments):



**Figure 2:** The result of Q2

Figure 2 is to convert the RGB image into a YCbCr image, and then perform histogram equalization on the Y layer, because the histogram equalization is only effective for grayscale images, and the Y layer is the intensity component and grayscale image. Finally, the histogram is converted back to the RGB image for comparison. The result is to equalize the number of pixels in each gray level and improve the overall contrast.

**Question 3:**

Part I (MATLAB code (copied and pasted as text) with comments (%)):

% Question 3: Image segmentation, edge detection and edge enhancement

clear;

% 1 -Read image building.tif

image = imread('building.tif');

im = image;

% Use Gaussian filtering to the image

im\_g = imgaussfilt(image);

% 2 -Detect (segment) straight edges in images

% Keep all edges within the edge intensity threshold

thresh = [0.04,0.10];

% Standard deviation of the filter

sigma = 1.49;

% Canny method extracts the image boundary and returns a binary image (boundary 1, otherwise 0)

im\_e = edge(double(im\_g),'canny',thresh,sigma);

% Calculate the standard Hough transform of a binary image

% H is the Hough transform matrix, I and R are the angle and radius values for calculating the Hough transform

[H,T,R] = hough(im\_e);

% Find the peak of the Hough transform

P = houghpeaks(H,50);

x = T(P(:,2));

y = R(P(:,1));

% Extract line segment

lines=houghlines(im\_e,T,R,P);

% 3 -Plotting Results

figure, subplot(1,2,1), imshow(image);

title('The Original image');

subplot(1,2,2), imshow(im\_e),hold on;

for k=1:length(lines)

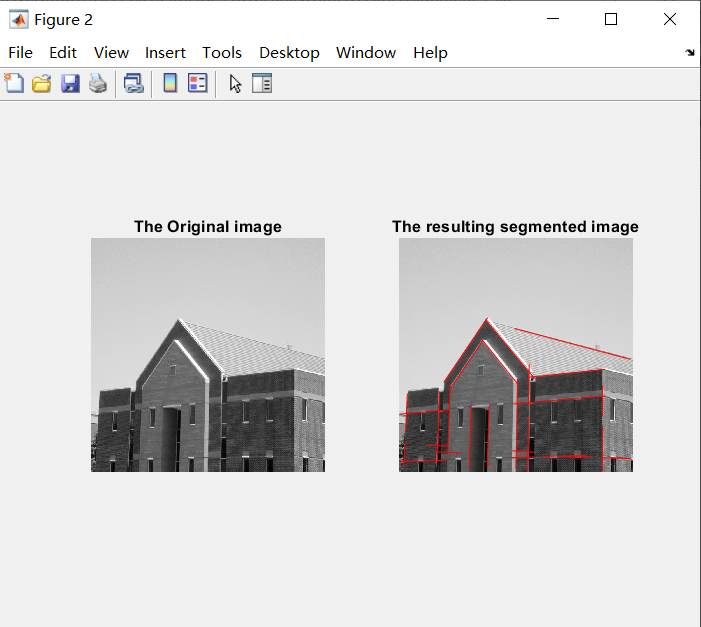
xy=[lines(k).point1;lines(k).point2];

plot(xy(:,1),xy(:,2),'Color','red');

end

title('The resulting segmented image');

Part II (results and comments):



**Figure 3:** The result of Q3

Figure 3 is to filter the image first to reduce the noise, and extract the image boundary. Then, the edge points of the image are obtained by Hough transform to extract line segments. Among them, the most important control variable is the threshold, which controls the accuracy of the image and the length of the selected line.