# CAP5400 – Digital Image Processing

Assign 3:
Edge detection and Open CV

Name: Zhenglong Wu
U#: U96342243
Date: Oct. 27th, 2019

# 1. Introduction

This assignment mainly focuses on the edge detection in the image to reduces the amount of data, and eliminates information that can be considered irrelevant, retaining important structural attributes of the image. And all image processing operations must be performed in the specified non-overlapping region of interest (ROI). The purpose of edge detection is to identify points in the digital image where the brightness changes significantly. Significant changes in image properties often reflect important events and changes in attributes. Edge detection is a research field in image processing and computer vision, especially feature extraction. There are many methods for edge detection, and most of them can be divided into two categories: one based on maximum and minimum values in the first derivative of the image and the other one based on finding the second derivative of the image zero crossing. In this experimental report, the first edge detection category will be mainly discussed, and some image processing functions, such as OpenCV package or Matlab built-in image processing functions, will also be called to process images for comparison. In the subsequent experiments, Section 2 describes the interpretation of some algorithms. Section 3 describes the operations and implementation details, such as parameters and all run options. In section 4 and 5, the results of experiment and conclusion are presented.

# 2. Description of algorithm

In this section, there are some explanations of the new algorithms encountered in the experiment. The description of ROI setting method will be omitted, which is to manually set the location and size of the ROI in the image. The Optimal threshold algorithm and Histogram stretching algorithm have already been discussed in the last assignment. The description will not be repeated here.

## 2.1. Sobel Edge Detection

The Sobel operator is a typical edge detection operator based on the first derivative. Since this operator introduces a similar local weighted average operation, it has a smoothing effect on noise and can well eliminate the influence of noise. The Sobel operator weights the effect of the position of the pixel. The common 3x3 Sobel operator consists of two sets of 3x3 matrices, which are horizontal and vertical templates, which are convolved with the image to obtain the horizontal and vertical gradient amplitude. The total gradient magnitude can be obtained by combining the gradient magnitudes of the horizontal and vertical directions by the Euclidean distance. And the direction of edge can be also obtained by the ratio of the gradient amplitudes of the horizontal and vertical directions.

## 2.2. Canny Edge Detection

Sobel efficiency is higher than canny edge detection, but Sobel edge is not as accurate and complete as Canny detection. The method of reducing the number of false edges in the Canny algorithm is to use a double threshold method. Select two thresholds and get an edge based on the high threshold, such an image contains few false edges. However, due to the higher threshold, the resulting image edges may not close and lose some real edges, so canny uses another low threshold to solve the problem. When the intensity of a pixel is between a low threshold and a high threshold, and it is adjacent to an edge pixel, canny also determines it as an edge.

## 2.3. Histogram Equalization

Histogram equalization is a method to adjust the gray distribution of the image, make the distribution of gray scale on 0~255 more balanced, improve the contrast of the image, and achieve the purpose of improving the visual effect of the image for human eyes. Images with lower contrast are suitable for using the histogram equalization method to enhance image detail. Histogram equalization maps one distribution (given a histogram) to another (a wider, more uniform distribution of intensity values), so the intensity distribution will be spread over the entire range. The mapping function should be a cumulative distribution function.

Assume that the histogram of the original image is f(x), then the cumulative histogram function should be F(x). And the equalized histogram is f(y), then cumulative equalized histogram function should be F(y). There is a certain mapping relationship between the original intensity value x and the new intensity value y: $y = T(x)$.

## 2.4. Otsu threshold segmentation

The algorithm assumes that an image consists of a foreground color and a background color, and a threshold is selected by statistical methods, which causes the threshold to separate the foreground color from the background color. The method utilizes the idea of the largest variance between the target area and the background area to achieve the purpose of segmenting the image. That is to say, when the optimal threshold T is selected, the variance value between the target and the background is the largest. The intensity lower than the threshold T is background, and the intensity higher than the threshold is foreground, so that the foreground and background can be segmented.

## 3. Description of implementation

The entire codes are developed in the MatLab. In parameter file, there are four common parameters for each image processing, which are “input filename”, “output filename”, “operation name” and “ROI location and size”. In addition, some operations needs more parameters, such as the user-defined operation in edge detection. The code reads the parameter file and run the specified functions to get the output image.

| parameter.txt – Common section | | | |
|---|---|---|---|
| Input filename | Output filename | Operation name | ROI location & size |
| lena.pgm | lena_histStretch.pgm | histStretch | [28,28…;…;…456] |
| … | … | … | … |

Each ROI is defined by 4 parameters: Rx, Ry, Sx, Sy. Rx and Ry is the location value in both x-axis and y-axis of the top left corner of the ROI, Sx and Sy is the size value in width and height of the ROI. The format of ROI parameters is [28,28,200,200; 28,284,200,200; 284,28,200,456]. It is a Nx4 matrix, which has N rows, and each row represents a ROI in the image. And four columns represent Rx, Ry, Sx, Sy separately.

When the program read the ROI information, it will also run an overlapping test right away to ensure there are no two ROIs overlaps in the same image.

| parameter.txt – Optional section | | | | |
|---|---|---|---|---|
| Operation | Parameter1 | Parameter2 | Parameter3 | … |
| edgeDetection | sobel3 | binary | [150,150,150] | … |
| … | … | … | … | … |

For example, the parameter1 is one of the parameters required by operation “edgeDetection”, which represents the operator is 3x3 Sobel or 5x5 Sobel. Parameter2 and parameter3 controls the display of image using function edgeDetection.

In the experimental part of calling the third-party method, the OpenCV tool cannot be imported into MatLab due to the machine configuration problems. Therefore, there are several third-party image processing functions built in MatLab are selected:

- edge(): Find edges in gray-scale intensity image.
- histeq(): Enhance contrast using histogram equalization.
- graythresh(): Global image threshold using Otsu's method.

# 4. Description of results

## 4.1. 3x3 Sobel detection of each ROI

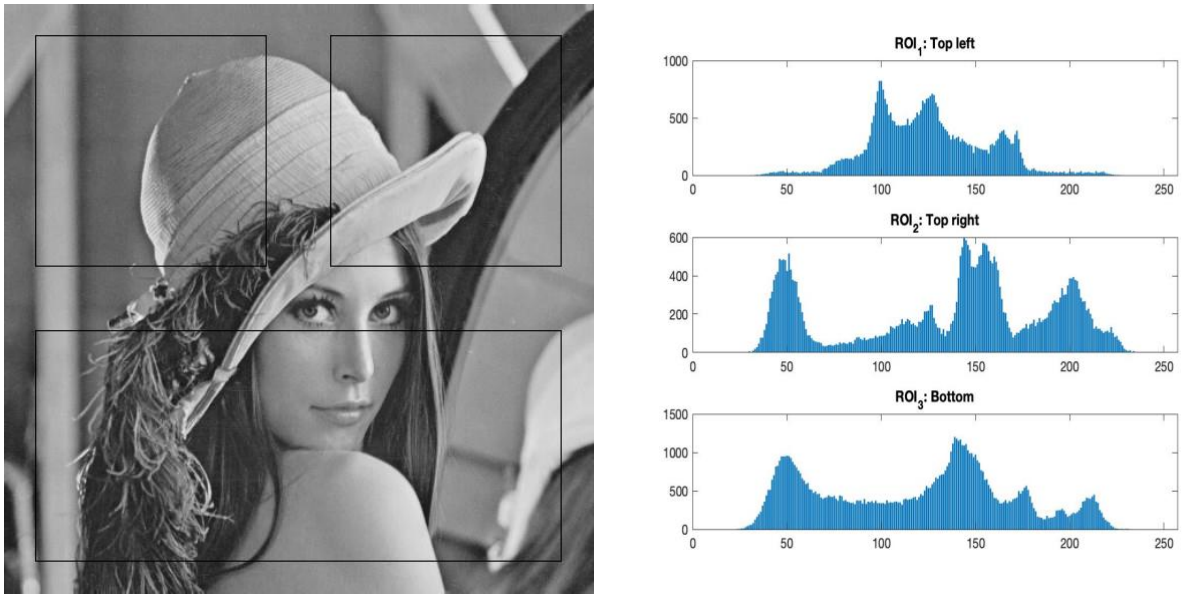


Figure 1.1 Original image and Histogram of each ROI

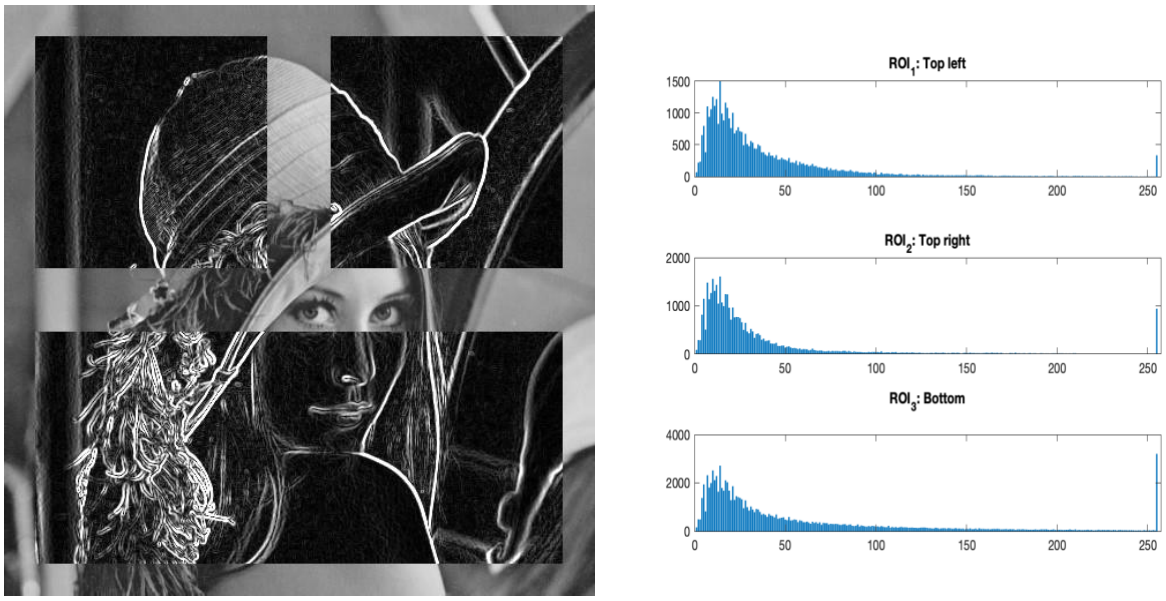


Figure 1.2 Gradient amplitute after 3x3 Sobel edge detection and Histogram of each ROI

Figure 1.3 Optimal Threshold for Gradient amplitute after 3x3 Sobel edge detection

RO1_1 (iterations, threshold): (12, 51)

RO1_2 (iterations, threshold): (22. 75)

RO1_3 (iterations, threshold): (17, 85)

Figure 1.4 Threshold 150 for Gradient amplitute after 3x3 Sobel edge detection

Figure 1.5 binary image based on specified direction after 3x3 Sobel edge detection

RO1_1 (start angle, end angle): (35, 55)

RO1_2 (start angle, end angle): (125. 145)

RO1_3 (start angle, end angle): (80, 100)

The figure 1.1 shows the three ROIs that we choose in the experiment, and the corresponding histogram of each ROI. The figure 1.2 shows the image after the Sobel edge detection, which based on the formula:

$$Sx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad Sy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad K = \begin{bmatrix} a_0 & a_1 & a_2 \\ a_3 & (i,j) & a_5 \\ a_6 & a_7 & a_8 \end{bmatrix}$$

$$dx_{(i,j)} = (a_0 + 2a_3 + a_6) - (a_2 + 2a_5 + a_8)$$
$$dy_{(i,j)} = (a_0 + 2a_1 + a_2) - (a_6 + 2a_7 + a_8)$$

$$G_{(i,j)} = \sqrt{dx_{(i,j)}^{\,2} + dy_{(i,j)}^{\,2}}$$

$$\theta_{(i,j)} = \arctan\left(dx_{(i,j)}/dy_{(i,j)}\right)$$

Sx and Sy are Sobel operator in 3x3 size. K is the window that we slide when we process the image. We use the Euclidean distance of the gradient amplitude of the pixel in the X direction and the gradient amplitude of the Y direction to represent the overall gradient amplitude G of the pixel. Figure 1.2 is an image where each pixel represents its own gradient amplitude.

In the histogram of Figure 1.2, we can notice that after Sobel edge detection, the histogram of each ROI presents a similar unimodal distribution with right skewness. Figure 1.3 presents a binarization result using the optimal threshold technique. Since the image is not a distinct bimodal distribution, it takes many iterations and contains too much detail. Based on the histogram in Figure 1.2, threshold 150 seems to be a good choice, and the binarization result is also better, which shows in Figure 1.4.

With $\theta_{(i,j)}$ value, the edges of the special angle are presented in Figure 1.5. If the horizontal is 0 degrees, ROI_1 shows all oblique edges of 35 degrees to 55 degrees, ROI_1 shows all oblique edges of 125 degrees to 145 degrees, and ROI_3 shows all vertical edges of 80 degrees to 100 degrees. It is worth mentioning that the gradient amplitude direction of the pixel is perpendicular to the edge direction. Therefore, when calculating the edge direction, the actually calculated gradient direction actually needs to be added or subtracted by 90 degrees.

## 4.2. 5x5 Sobel detection of each ROI

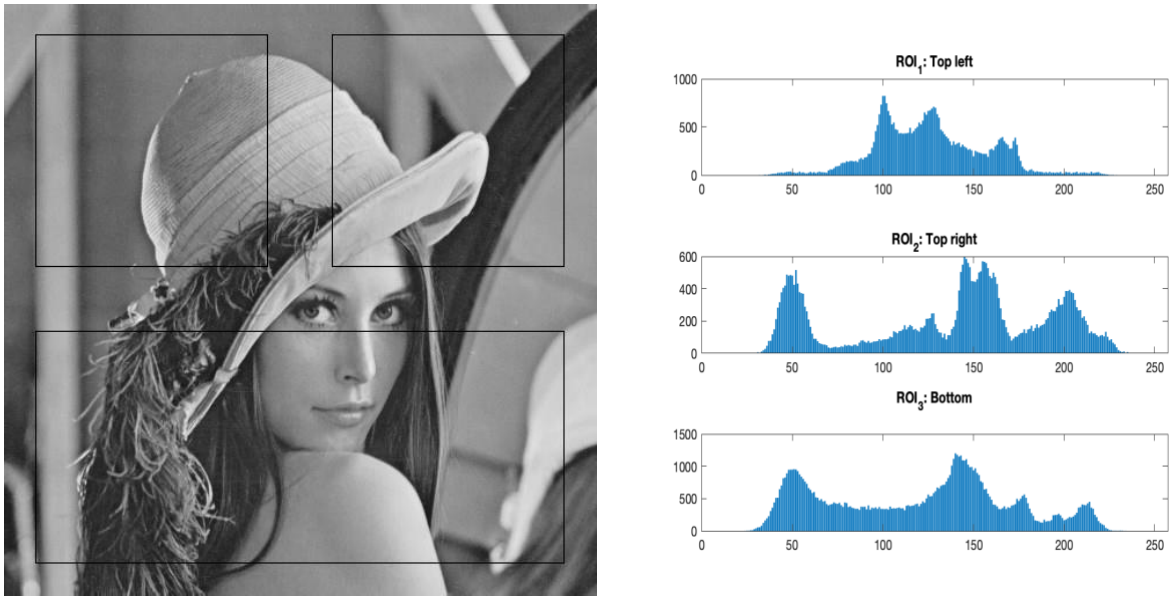


Figure 2.1 Original image and Histogram of each ROI

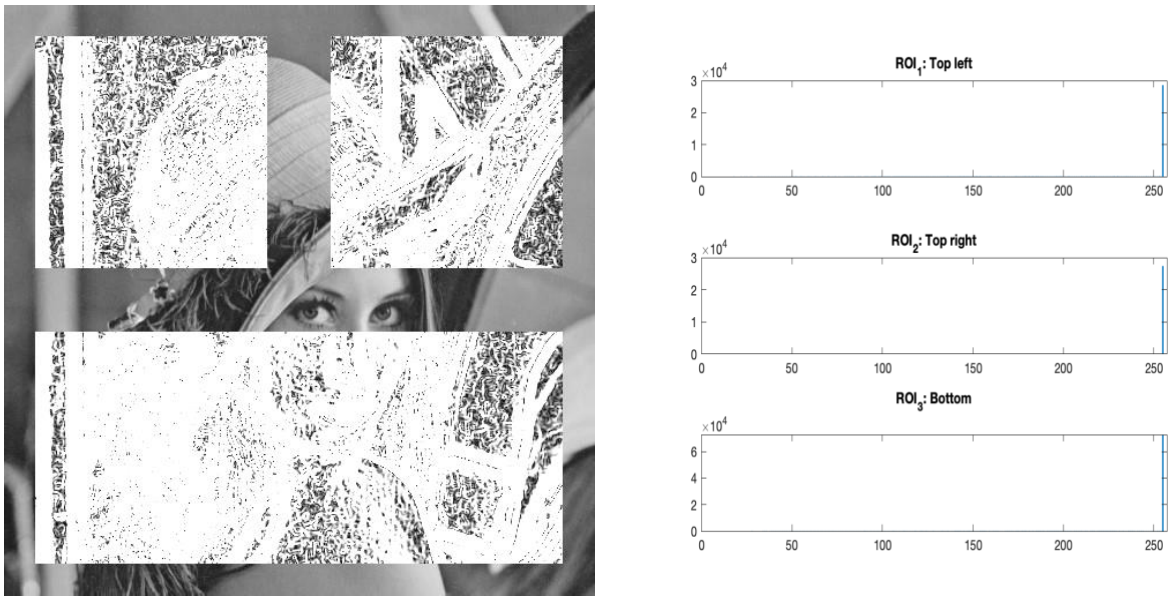


Figure 2.2 Gradient amplitute after 5x5 Sobel edge detection and Histogram of each ROI

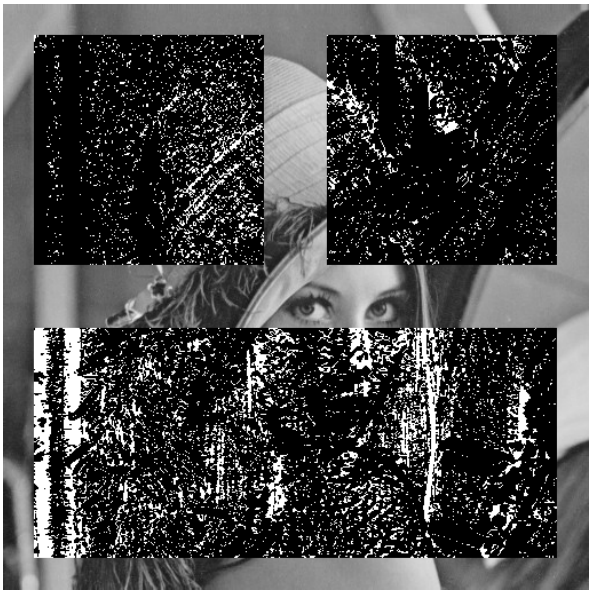

Figure 2.3 binary image based on specified direction after 5x5 Sobel edge detection

RO1_1 (start angle, end angle): (35, 55)

RO1_2 (start angle, end angle): (125. 145)

RO1_3 (start angle, end angle): (80, 100)

The figure 2.1 shows the three ROIs that we choose in the experiment, and the corresponding histogram of each ROI. The figure 2.2 shows the image after the 5x5 Sobel edge detection, which based on the formulas similar to the above 3x3 Sobel edge detection:

$$Sx = \begin{bmatrix} -5 & -4 & 0 & 4 & 5 \\ -8 & -10 & 0 & 10 & 8 \\ -10 & -20 & (0,0) & 20 & 10 \\ -8 & -10 & 0 & 10 & 8 \\ -5 & -4 & 0 & 4 & 5 \end{bmatrix} \qquad Sy = \begin{bmatrix} -5 & -8 & -10 & -8 & -5 \\ -4 & -10 & -20 & -10 & -4 \\ 0 & 0 & (0,0) & 0 & 0 \\ 4 & 10 & 20 & 10 & 4 \\ 5 & 8 & 10 & 8 & 5 \end{bmatrix}$$

$$dx_{(i,j)} = \sum_{m=-2}^{2} \sum_{n=-2}^{2} Sx(m,n) f(i-m, j-n)$$

$$dy_{(i,j)} = \sum_{m=-2}^{2} \sum_{n=-2}^{2} Sy(m,n) f(i-m, j-n)$$

$G_{(i,j)}$ *and* $\theta_{(i,j)}$ *formulas are the same as in the previous experiment*

The computation process is exactly the same as the previous 3x3 Sobel edge detection experiment. The only difference is that this experiment uses a different convolution mask (5x5 Sobel operator), and the sliding window for image processing increase to 5x5.

Compare Figure 1.2 and Figure 2.2, we can notice that 5x5 Sobel operator makes the overall brightness of the image increase, it is difficult to see what the original picture is after 5x5 Sobel Edge Detection.

Compare Figure 1.5 and Figure 2.3, we can notice that both have successfully detected the edge of a user-defined angle. However, 5x5 Sobel operator obviously detects more unwanted details than 3x3 Sobel operator, or it can be considered as noise.

## 4.3. Sobel Detection by the third-party function

Figure 3.1 Sobel Edge Detection by built-in function edge('sobel')
ROI_1 threshold: 0.0587
ROI_2 threshold: 0.0809
ROI_3 threshold: 0.0954

Figure 3.2 Sobel Edge Detection by built-in function edge('sobel') with threshold 0.05

Figure 3.3 Sobel Edge Detection by built-in function edge('sobel') with threshold 0.05 and direction 'Horizontal'

Figure 3.4 Sobel Edge Detection by built-in function edge('sobel') with threshold 0.05 and direction 'Vertical'

In this experiment, the Matlab built-in function "edge()" is used to find edges in gray-scale intensity image.

Figure 3.1 is the binarization result of Sobel edge detection by function edge(), and the threshold is automatically calculated when no threshold is defined by user. In the experiment, the threshold of ROI_1, ROI_2 and ROI_3 are 0.0587, 0.0809, 0.0954.

Figure 3.2 is the binarization result of Sobel edge detection by function edge() with defined threshold 0.05.

Figure 3.3 and Figure 3.4 are the binarization result of Sobel edge detection by function edge() with defined threshold 0.05 and defined direction 'Horizontal' and 'Vertical' separately.

In this built-in function, gradient amplitude of each pixel is standardized, and the threshold is used to represent the distance to the average gradient amplitude. Therefore the threshold can only be modified between [0,1]. Compare with the experiment 4.1, the binarization result of this built-in function is more detailed than the previous self-editing algorithm, and the boundary lines are narrower.

## 4.4. Compare Canny and Sobel Edge Detection

Figure 4.1 Canny Edge Detection by built-in function edge('canny')

Figure 4.2 Sobel Edge Detection by built-in function edge('sobel')

In this experiment, the Matlab built-in function "edge()" is used to find edges in gray-scale intensity image based on different defined methods.

Figure 4.1 is the binarization result of Canny edge detection by function edge('canny'). And Figure 4.2 is the binarization result of Sobel edge detection by function edge('sobel').

A very obvious and intuitive difference is that Canny algorithm detects more edges than only Sobel edge detection, and the edges in Canny algorithm are more complete. The reason is that Canny is a double threshold algorithm to detect and join edges. Canny first uses Gaussian filter to remove the noise of the picture and then uses the same method as Sobel edge detection to calculate the gradient magnitude of each pixel. And the final step to detect edges is the most essential part of the algorithm, which is based on the formula:

$$I_{(x,y)} = \begin{cases} 255 \text{ (a strong edge)} & if\ \ G_{(x,y)} \geq High\ Threshold \\ 255\ (an\ edge) & if\ G_{(x,y)} \geq Low\ Threshold\ and \\ & G_{(x,y)}\ connect\ to\ an\ edge\ pixel \\ 0\ (not\ an\ edge) & if\ G_{(x,y)} \leq Low\ Threshold \end{cases}$$

$G_{(x,y)}$ is the gradient amplitude of pixel located on (x, y) of the original image. And the gradient amplitude is calculated by the edge detection operator.

## 4.5. Compare Histogram Equalization and Histogram Stretching

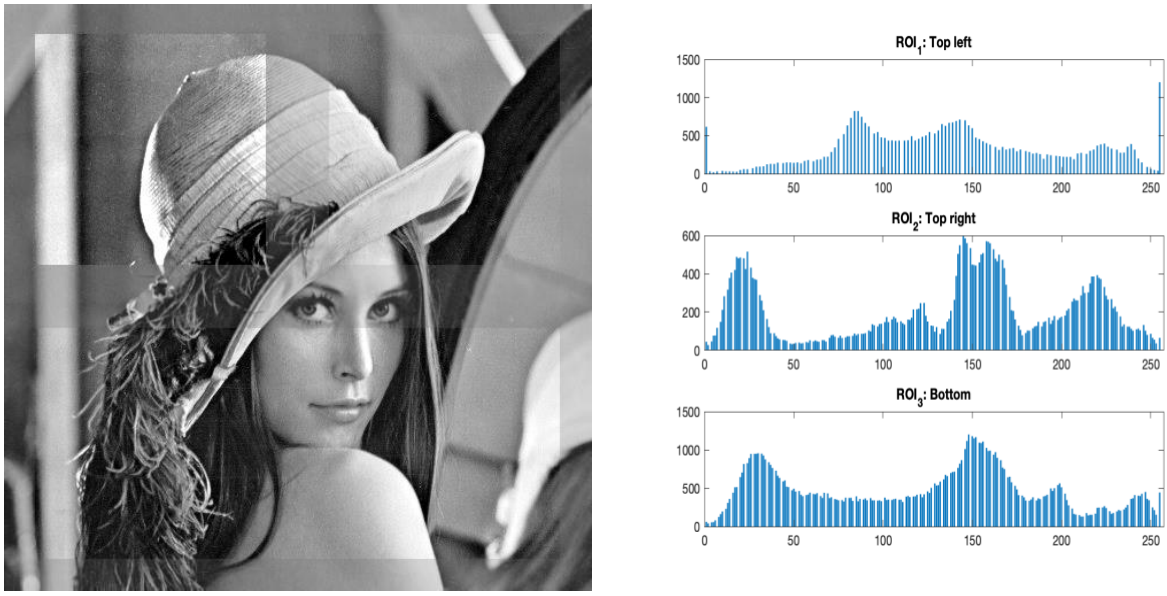


Figure 5.1 Image Histogram Stretching and Histogram of each ROI

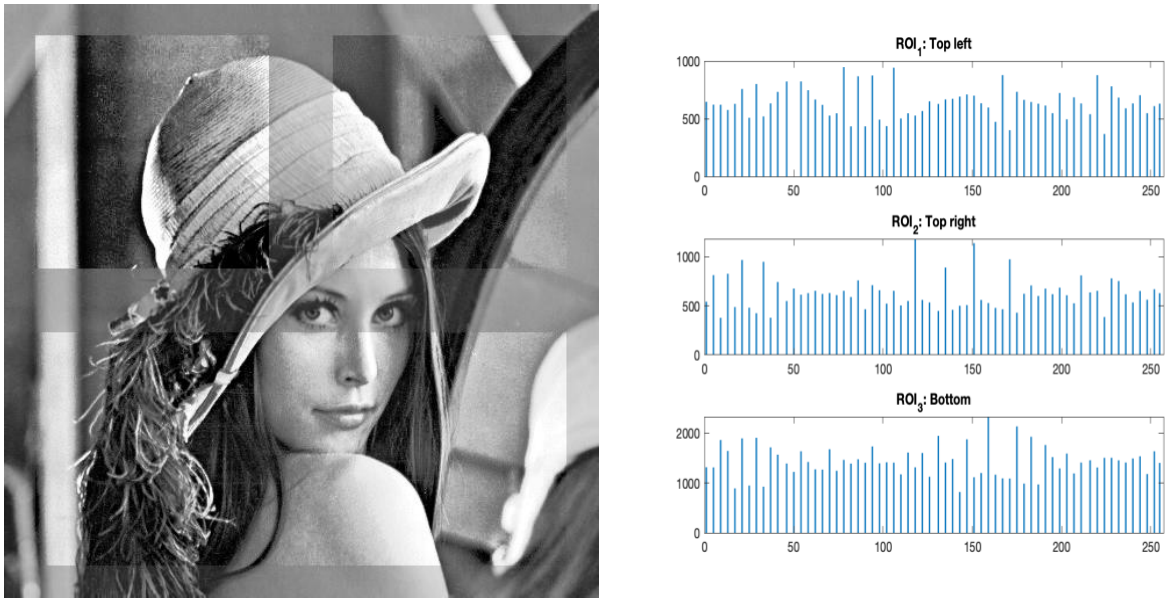


Figure 5.2 Image Histogram Equalization and Histogram of each ROI

In this experiment, the Matlab built-in function “histeq()” is used to enhance contrast using histogram equalization.

Figure 5.1 is the result image using histogram stretching algorithm. And the Figure 5.2 is the result image using histogram equalization function “histeq()”

Histogram equalization is to adjust the gray scale distribution of the image to make the distribution of gray scale on the 0~255 more balanced, which is based on the formula:

$$P(i) = H(i)/(M * N)$$

$$cdf(i) = \sum_{j=0}^{i} P(i)$$

$$f(v) = round(\frac{cdf(v) - cdf_{min}}{M * N} * 255)$$

H is the histogram of the image, H(i) is the counts of the intensity "i" in whole image. M and N are the length and height of the image in pixels. Cdf is the cumulative distribution function. "v" is an intensity level of a pixel in original image, and f(v) is the new intensity level of this pixel after histogram equalization.

Compare Figure 5.2 and Figure 5.1, the results ROI of histogram equalization is significantly more enhanced than histogram stretching.

## 4.6. Compare Otsu Thresholding and Optimal Thresholding

Figure 6.1 Binarization using Otsu thresholding

RO1_1 threshold: 0.5176
RO1_2 threshold: 0.4549
RO1_3 threshold: 0.4353

Figure 6.2 Binarization using Optimal thresholding

RO1_1 (iterations, threshold): (4, 126)
RO1_2 (iterations, threshold): (1, 147)
RO1_3 (iterations, threshold): (1, 128)

In this experiment, the Matlab built-in function “graythresh()” returns a global image threshold using Otsu's method.

Figure 6.1 is the result image using Otsu function “graythresh()” on each ROI. And the Figure 6.2 is the result image using Optimal thresholding algorithm on each ROI.

In Otsu algorithm, we need to get the global mean intensity “M” of the whole image at first. Then we start to traverse from the minimum gray level in the picture to the maximum gray level with “t”, and the gray level t divides the picture into two parts: the foreground and the background. Assume the ratio of the number of pixels in the foreground to the image is W0, and the average intensity is M0. Then the ratio of the number of pixels in the background to the image is W1, and the average intensity is M1. Finally, “t” is the optimal Otsu threshold when the following formula reaches the maximum value:

$$G = W_0 * (M_0 - M)^2 + W_1 * (M_1 - M)^2$$

Compare Figure 6.1 and Figure 6.2, the performance is similar, the human face of the Optimal thresholding has more details than Otsu thresholding.

## 4.7. Combine Otsu segmentation and foreground histogram equalization

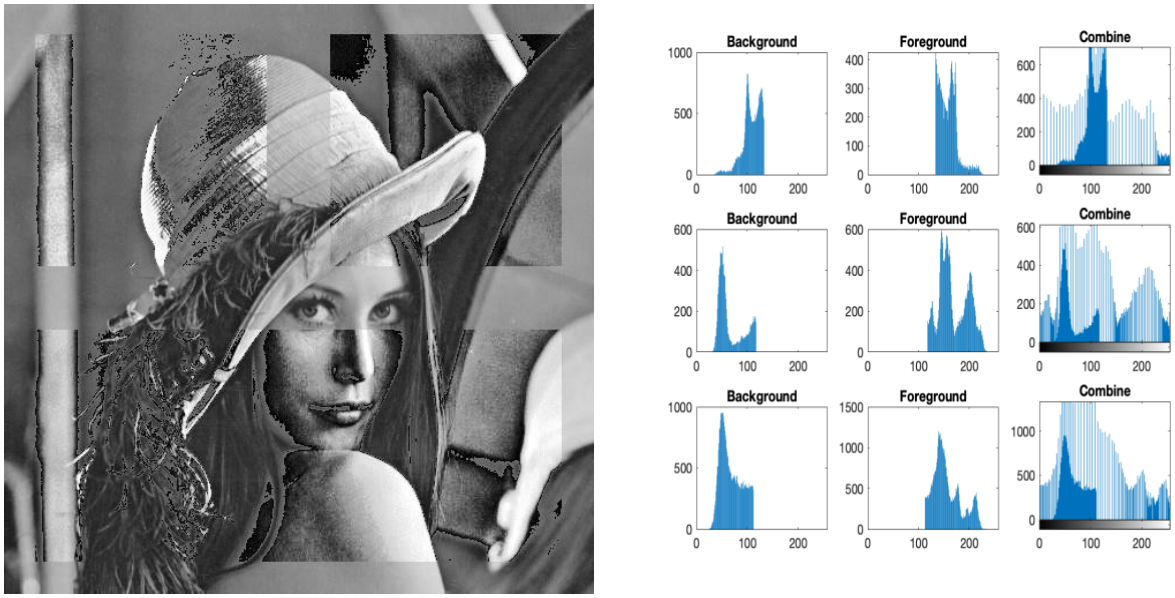


Figure 7.1 Combine Otsu segmentation & foreground histogram equalization, and histogram of each ROI

The Figure 7.1 shows the image processing result of using both Otsu segmentation and histogram equalization of foreground.

This experiment used built-in function “graythresh()” in experiment 4.6, and histogram equalization algorithm, respectively. And the implementation formula of the histogram equalization algorithm is described in the details of experiment 4.5.

In the histogram of Figure 7.1, the combine columns show that only the histogram of foreground is equalized in each ROI. And from the image of Figure 7.1, we can notice that the human part (foreground) is obviously contrast enhanced.

# 5. Conclusion

According to the experiment 4.1 and 4.2, we can conclude that 5x5 Sobel edge detection is able to detect more details than 3x3 Sobel operator, However, many details are not suitable for the human eye to read, and also highlight a lot of noise. Therefore, I think 3x3 Sobel edge detection is more suitable for image processing in this experiment image.

Compare the experiment 4.1 and 4.3, I think my own method performs better than MatLab built-in function edge() in this experiment image. In Figure 3.1, edge() function detects more edge details than the Figure 1.4. However, too many edges make the image look less intuitive. Figure 1.4 detected most of the important edges.

From experiment 4.4, we can clearly see that Canny performs better than Sobel, and Canny detects a more complete and accurate edges. However, Canny is less efficient than Sobel.

In experiment 4.5, histogram equalization makes the histogram distribution in the ROI more uniform than histogram stretching. And also, the results of histogram equalization are significantly more enhanced than histogram stretching.

In experiment 4.6, I think Otsu threshold and Optimal threshold performs similarly on segmentation in this experiment. Both Otsu threshold and Optimal threshold are good global image segmentation method. And they have similar shortcomings, when the image histogram is not bimodal distribution, they do not necessarily distinguish the background and foreground well.

In experiment 4.7, the image segmentation algorithm is well integrated with the image enhancement algorithm. Only the enhancement of the foreground part (human) makes the contrast of the foreground details stronger.