CSci 5561
Computer Vision

Programming assignment 1 report

Spring Semester 2016

Xiao Lu

ID：5079620

Overview Structure and Code Files

1. Algorithms and code files
   - 1.1 Edge operators (Sobel and Roberts) *SobelEdge.m* *RobertsEdge.m*
   - 1.2 Thresholding methods(OUST or Manual) *OTSUThreshold.m*
   - 1.3 Expansion *Expansion.m*
   - 1.4 Thinning method *Thining.m*
   - 1.5 Modified thinning method *ThinningModified.m* (Extra Credit)
2. Test images
   - 2.1 *Daisy.jpg*
   - 2.2 *Sunflower.jpg*
3. Main MATLAB implementation
   - 3.1 *SobelEdgeMain.m*
   - 3.2 *RobertsEdgeMain.m*

Part 1 Sobel and the Roberts edge operators

1. Sobel edge operator

1.1 Algorithm

The Sobel operator uses two 3×3 kernels which are convolved with the original image. Say, X represents the derivative of horizontal direction, Y represents the derivative of vertical direction. We set input image as A, thus we can get G_x and $G_y$by following steps. Here, G_x and G_y are represent horizontal and vertical derivative approximations respectively at each point.

$$G_x = X * A = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A, \; G_y = Y * A = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * A$$

Hence, the gradient magnitude at each point in the image is: $G = \sqrt{G_x^2 + G_y^2}$ [1].

And the gradient direction θ is: $\theta = \text{atan2}(G_y, G_x)$ [2].

2. Roberts edge operator

2.1 Algorithm

In contrast to Sobel edge operation, Roberts edge operator have different kernel and gradient formula which are shown as following:

Say, X represents the derivative of horizontal direction, Y represents the derivative of vertical direction.

$$X = \begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix}$$

Hence, the gradient magnitude at each point in the image is: $G = \sqrt{G_x^2 + G_y^2}$.

And the gradient direction θ is: $\theta = \arctan(G_y / G_x)$[3].

3. Results

Figure 1. Edge detection by Sobel operator for Daisy.jpg

Figure 2. Edge detection by Roberts operator for Daisy.jpg

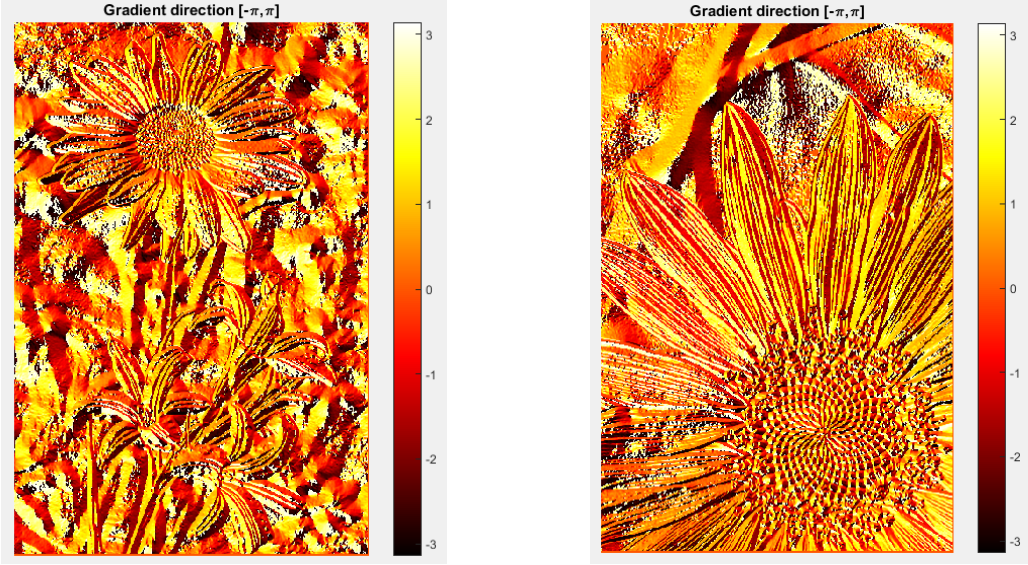

Figure 3. Gradient direction plot

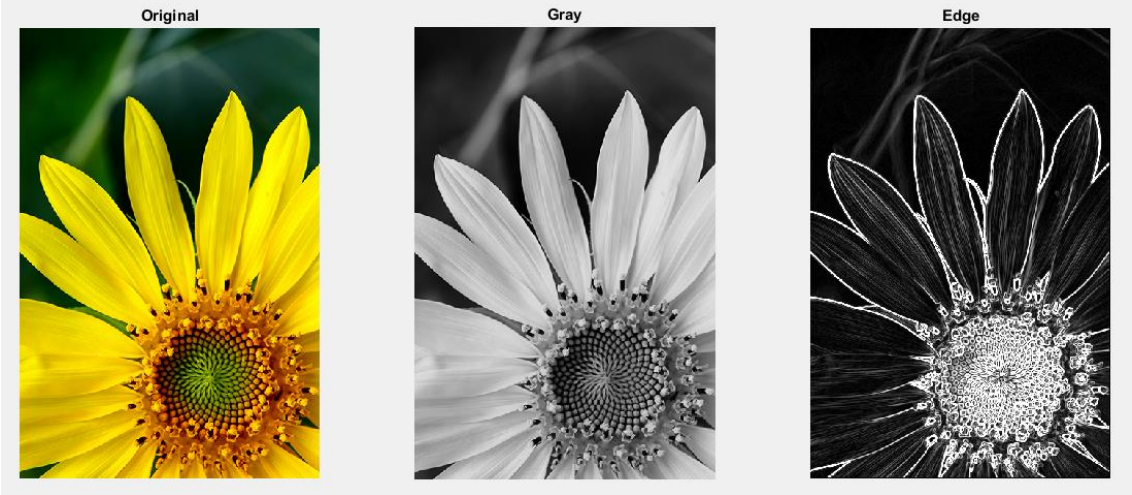

Figure 4. Edge detection by Sobel operator for Sunflower.jpg

Part 2 Thresholding Methods

1. Otsu’s method

   In Otsu's method we exhaustively search for the threshold that minimizes the intra-class variance, defined as a weighted sum of variances of the two classes:

   $$\sigma_{\omega}^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$

   Where weights ω_0, ω_1 are the probabilities of the two classes separated by a threshold t and σ_0, σ_1 are variances of these two classes.

   Consequently, Otsu's method is roughly a one-dimensional, discrete analog of Fisher's Discriminant Analysis[4].

2. Manual thresholding method

   In manual thresholding method, I just set the threshold manually. Through the histogram of gradient magnitude of each operator, I chose threshold equal to 20, 100 and 230 for Sobel edge operator, while 20, 80 and 120 for Roberts edge operator.

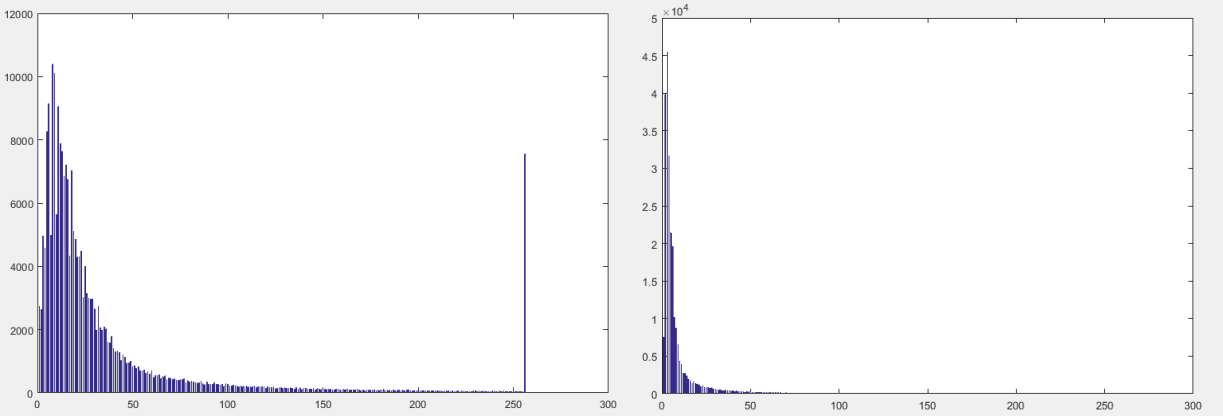

Figure 5(a).Histogram of *Daisy.jpg* for Sobel operator (b). Histogram of *Daisy.jpg* for Roberts operator

Figure 6. Otsu and manual threshold selection of Sobel operator

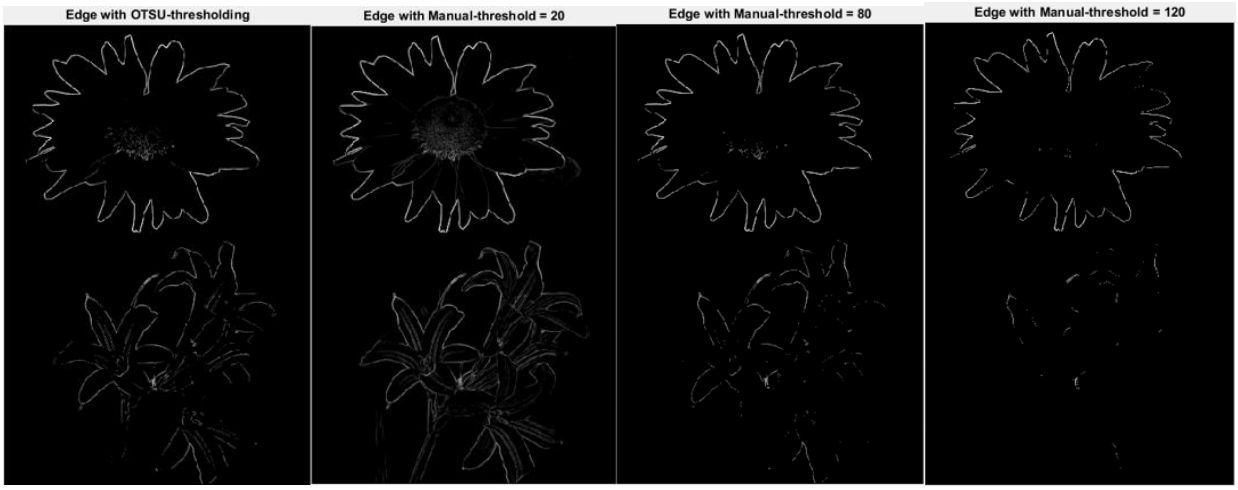

Figure 7. Otsu and manual threshold selection of Robers operator

Part 3 Expansion Method

1. Basic methodology
   In order to get a smoother thinning edge in next step, I should expand the edge of image. The basic idea is, I iteratively check the four directions (N, S, E and W) for a pixel, if all of them are 1, but the current pixel is 0, we manually set it to 1.
2. Result

	186	187	188	189	190	191
330	0	0	0	1	1	1
331	0	0	1	1	0	1
332	0	1	1	0	1	1
333	1	1	0	1	1	0
334	1	0	1	1	0	0
335	1	1	1	0	0	0
336	1	1	0	0	0	0
337	0	0	0	0	0	0

	186	187	188	189	190	191
330	0	0	0	1	1	1
331	0	0	1	1	1	1
332	0	1	1	1	1	1
333	1	1	1	1	1	0
334	1	1	1	1	0	0
335	1	1	1	0	0	0
336	1	1	0	0	0	0
337	0	0	0	0	0	0

Table 1. Expansion Result

Part 4 Thinning Methods

1. Basic thinning method
   The basic thinning method is to iteratively delete pixels inside the shape to shrink it without shortening it or breaking it apart. To decide whether an edge pixel P_1 should be deleted, consider its 8 neighbors in the 3 by 3 neighborhood P_2, P_3, P_4, P_5, P_6, P_7, P_8 and P_9, and define:
   $N(P_1)$: number of non-zero neighbors: $N(P_1) = P_2 + P_3 + \cdots + P_9$
   $S(P_1)$: number of 0 to 1 (or 1 to 0) transitions in the sequence : $(P_2, P_3, \cdots, P_9)$

We will delete the pixel if it meet all the following requirements: $(P_1 = 1)$ and $(2 \leq N(P_1) \leq 6)$ and $(S(P_1) = 1)$ and $(P_2 \cdot P_4 \cdot P_6 = 0)$ and $(P_4 \cdot P_6 \cdot P_8 = 0)$ and $(P_7 \neq 0)$. We repeat the previous procedure to delete pixels until it meet the maximum iteration or it cannot be deleted[5].

2. Modified thinning method (Extra Credit)

When I went through the detail of image, I noticed that the original thinning method make each '1' pixel totally connected, which makes line wider by comparing with MATLAB edge and thinning function. I show the detailed comparison between my original thinning method and MATLAB function as following:

Figure 8. Basic thinning method compare with MATLAB function

In order to get thinner line which is quite similar to MATLAB function, I suggested the algorithm to delete the interconnection part of each line. The boundary conditions are 1: $(P_1 = 1)$ and $(P_2 = 1)$ and $(P_4 = 1)$; 2: $(P_1 = 1)$ and $(P_2 = 1)$ and $(P_9 = 1)$ and $(P_3 + \cdots + P_8 = 0)$. We will delete the points which meet above conditions. Thus, I obtained the following results:

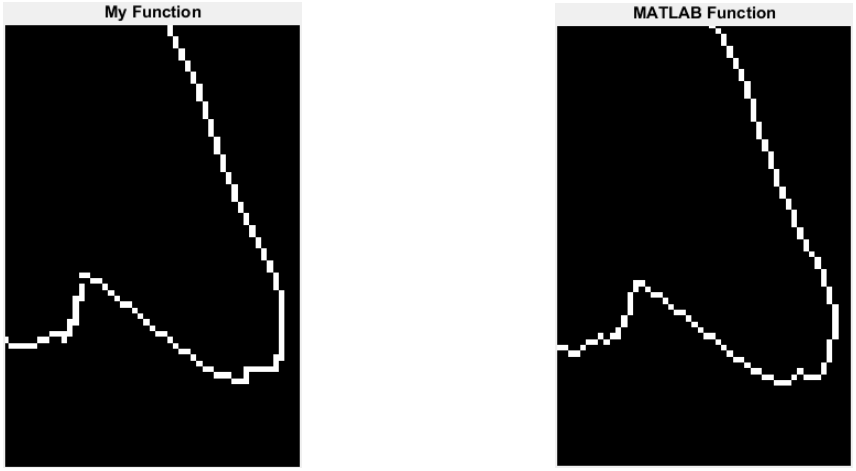

Figure 9. Modified thinning method compare with MATLAB function

Therefore, My results are shown as follows:

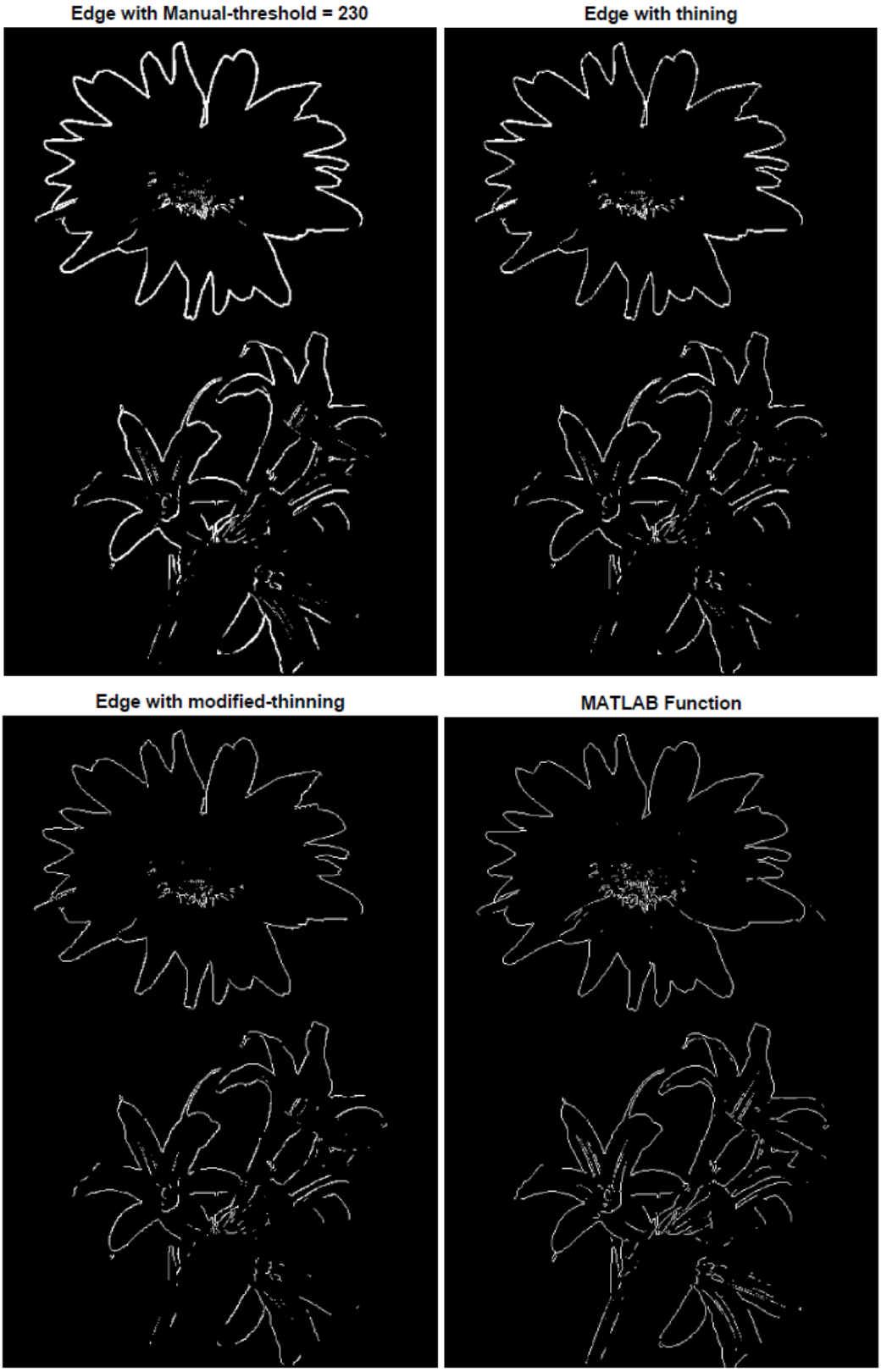

Figure 10. The overall results

Part 5 Discussion and Conclusion

1. Edge operator

   I implemented two edge operators: Sobel and Roberts.

   Through the comparison, I found Sobel edge operator obtained the much clearer and stronger edge in contrast to Roberts edge operator. On the other hand, Roberts edge operator is simpler, so that it is faster to implement.

   However, the basic principle of two algorithms are similar, we can get the quite similar target edge image by choosing proper threshold, which I will discuss in following part.

2. Thresholding methods

   In this project, I chose two methods to select threshold： Otsu and Manual selection.

   According to previous results, I could say, for Sobel edge operator, Otsu method did not perform very well, since the histogram is quite polarized and has clear boundary. Thus, the best threshold is close to 255. Here, to avoid the loss of important edge information, I manually set threshold as 230.

   For Roberts edge operator, Otsu method obtain the reasonable threshold, since the histogram is smooth and decrease gradually. Hence, I chose Otsu as its threshold.

3. Expansion Method

   I considered two method at the beginning of assignment. The first one is Gaussian blur to reduce the edge, so that it could be smoother. But it did not practically work very well. So I chose the second way to expand and smooth the image by just considering four directions of a pixel.

4. Thinning methods

   To determine if a thinning method is feasible and effective, the most important part is to set the boundary condition. The basic thinning mothed I implemented is literally good, but by comparing with MATLAB function, it practically has too many interconnected points, which make it wider and not enough to be considered as “ultimate thinning image”. Thus, I suggested two more new boundary conditions to make it looks better.

   Nevertheless, when I went over this modified thinning method, I noticed that it still has one shortage, which occurs some isolated points and noise. I consider this as my future work.

   Moreover, I just implemented two images in this project. If I take a large number of images into consideration, this implementation is not fast enough, I may think the matrix computation and GPU parallel programming in future.

References:

[1] Nikos Papanikolopou*los. "CSci 5561Computer Vision*" *Slide1-3.*

[2] Wikipedia. "Sobel operator" https://en.wikipedia.org/wiki/Sobel_operator

[3] Wikipedia. "Roberts cross" https://en.wikipedia.org/wiki/Roberts_cross

[4] Wikipedia. "Otsu's method" https://en.wikipedia.org/wiki/Otsu%27s_method

[5] Ruye Wang. "A thinning method" http://fourier.eng.hmc.edu/e161/lectures/morphology/node2.html