Project 3: Edge Detection Using Gradient with Prewitt Weights

Daniel Sawyer U3363-7705
CAP4401 Image Processing Spring 2017
University of South Florida, Tampa

***Abstract*— Edge detection using gradient and direction with Prewitt 3x3 and 5x5**

I. Introduction and overall description

This assignment focuses on edge detection by using gradient and direction using the Prewitt 3x3 and 5x5. This allows us to detect the edges in an image depending upon a threshold and direction. You can save the gradient image, binarized gradient using a threshold (edges), or binarization using threshold and direction. In Section 2, the basic algorithms used in the assignment are described. Section 3 describes the implementation details of the assignment. In Section 4 the results for the assignment is presented and finally we conclude in Section 5.

II. Description of algorithms

In this section the algorithms used in this assignment are described. The algorithm for calculating dx, dy, gradient and direction is the first algorithm. Then there is also a basic thresholding/binarization using a threshold T and direction D which then leaves only the edges or edges in a certain direction $\pm$ 10°.

A. Calculating Gradient and Direction

Calculating the gradient and direction requires you to first calculate dx and dy. This is done using the Prewitt operator and is similar to 2D smoothing in a sense. You take the weighted average of the pixels neighborhood in a m^2 where m is the size of the Prewitt window. We will be using 3 and 5. $\sum_{k=i-m}^{k=i+m}\sum_{l=j-m}^{l=j+m}(I[k][l]*Prewitt[k-i+m][l-i+m])$ to calculate dx. You use the same formula for calculating dy as well except the Prewitt is transposed. Here are the Prewitt 3x3 used for dx and transposed for dy:

$$dx = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$dy = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

After dx and dy have been calculated, you then calculate the gradient and direction. Gradient = $\sqrt{dx^2 + dy^2}$ and the Direction = $tan^{-1}(dy/dx)$. To save the gradient image, you just replace the pixel intensity with the gradient value for each pixel in the image. This works on all channels gray, R, G, B, H, S, and I.

B. Binarization of Gradient

This is a simple thresholding operation like explained in the first project. Basically just compare the gradient value G against the threshold value T. If the G $\geq$ T, set pixel to 255/White. Else, set pixel to 0/Black. This works the same for all channels. If doing RGB together, use set addition to decide: If G(Red$\vee$Green$\vee$Blue) $\geq$ T, set to 255/White. Else, set to 0/Black.

C. Binarization of Gradient with Direction

This is the same as above except you also check the direction value D as well as the threshold T. The direction D must be within $\pm$ 10° of the value the user enters.

III. Description of implementation

The entire code is developed in C++ language on Ubuntu 16.04.1 and USF's FSPrime linux server. The code for reading, writing the ppm and pgm image files, and conversion from HSI is provided as part of the Image Class file. The HSI Class contains conversion to HSI from the Image Class. The main is in project.cpp and reads the parameters file, which contains all the info on the input image, output image, and process to run on the image with their ROIs. It then saves the resulting image(s).For gradient the user just chooses the process and enters in the ROI and sets T = -1 and D = -99. For the edges the user sets T = [0,255] and D = -99. For the directional edges T = [0,255] and D = [-90,90]. All processes also have an ROI that uses the following format that is put in after the process parameter(s): startX startY SizeX SizeY. The README includes a more detailed explanation.

IV. Description and Analysis of Results

A. Description of Results

This section illustrates the results of the algorithms used. The first group of images are the before and after of the gray scale gradient, edges, and directional edges, Figures 1-6. The second group shows the color gradient and edges of individual channels R, G, B, all channels RGB, and I with Figures 7-14.

Fig. 1. Original Snow Boarder Gray Scale

Fig. 2. Gradient Prewitt 3x3

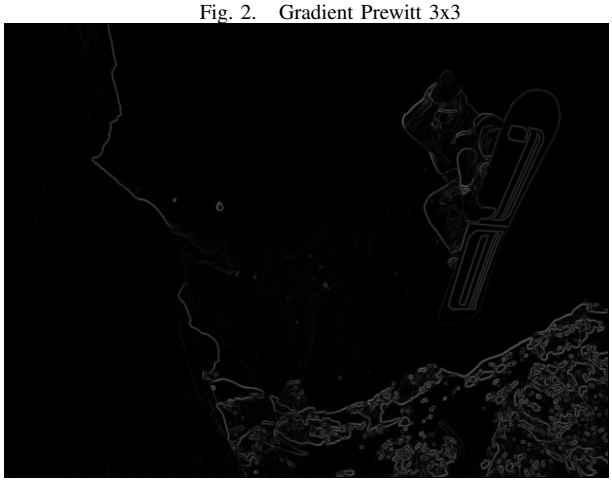


Fig. 3. Gradient Prewitt 5x5

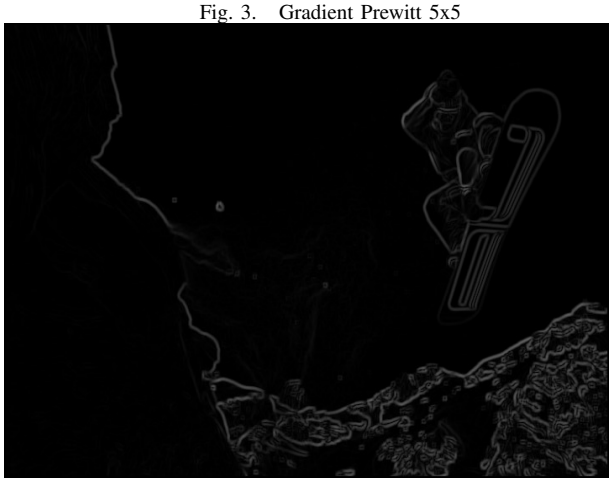


Fig. 4. Edges Prewitt 3x3

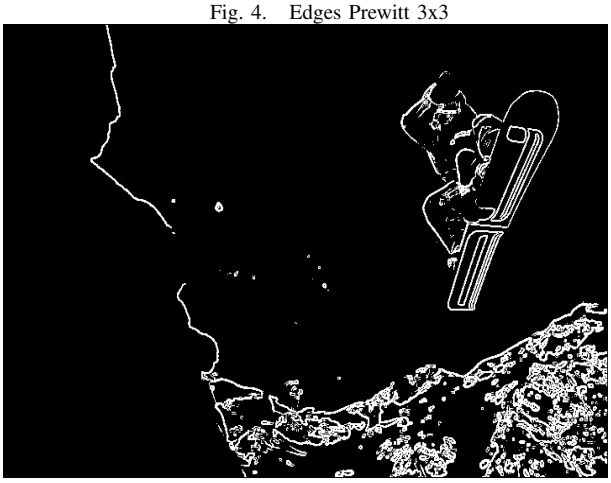


Fig. 5. Edges Prewitt 5x5

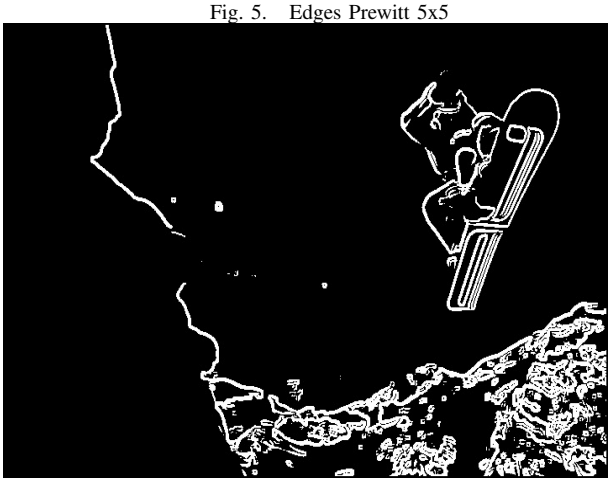


Fig. 6. Gradient & Direction 25° Prewitt 3x3

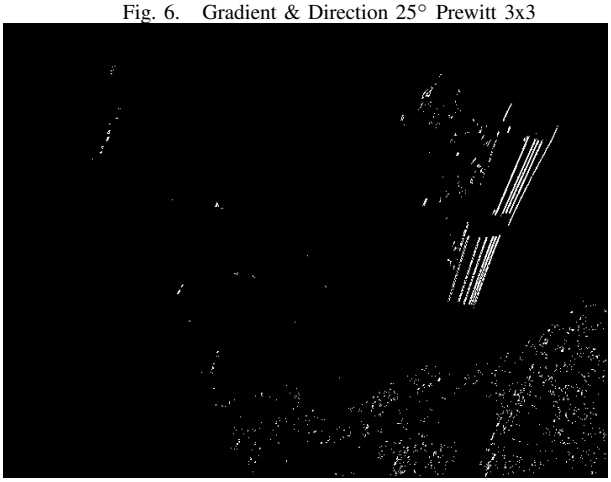


Fig. 7. Original Color Wheel

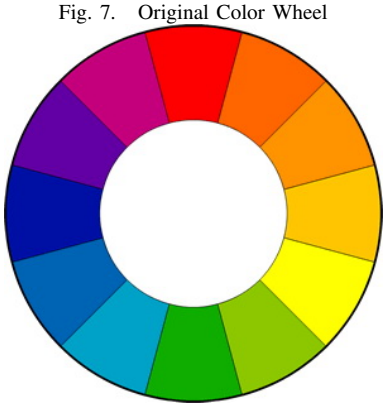


Fig. 8. Gradient RED Prewitt 3x3

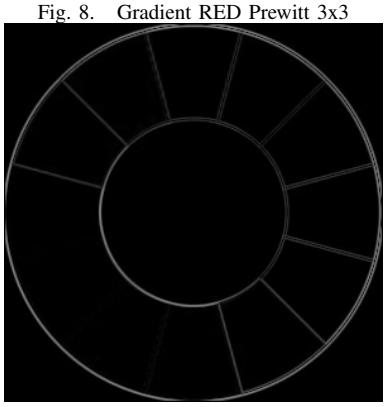


Fig. 9. Gradient GREEN Prewitt 3x3

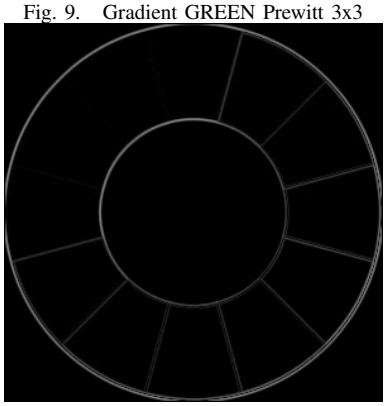


Fig. 10. Gradient BLUE Prewitt 3x3

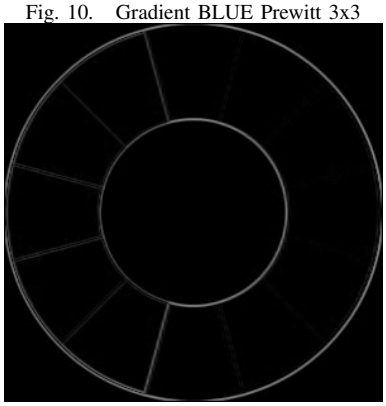


Fig. 11. Gradient RGB Prewitt 3x3

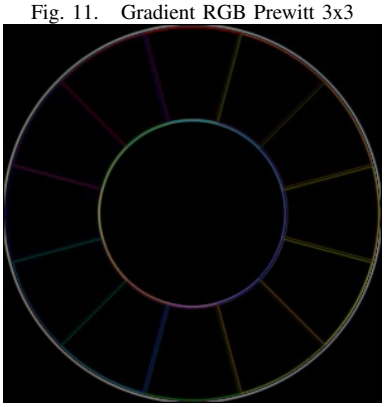


Fig. 12. Gradient I Prewitt 3x3

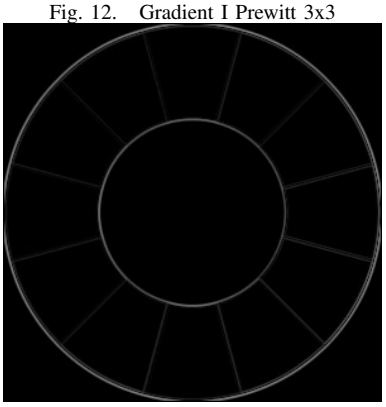


Fig. 13. Edges RGB Direction 45° Prewitt 3x3

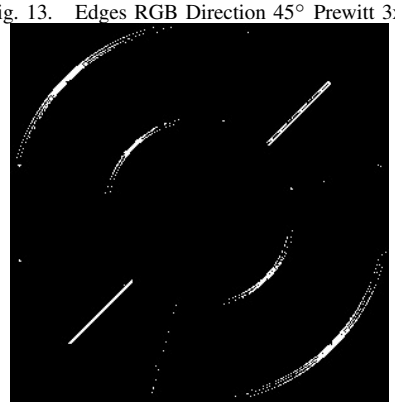
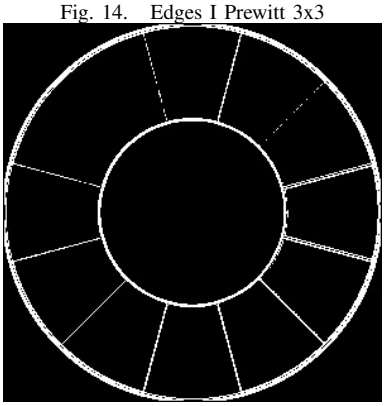


Fig. 14. Edges I Prewitt 3x3



B. Performance Evaluation and Analysis of Results

All algorithms run at O(m^2n^2), where n^2 is the size of the 2D pixel array or i x j and m is the size of the Prewitt window. This is because all the algorithms just go through every pixel and calculates dx, dy, gradient, and direction. The binarization by thresholding or thresholding and direction takes O(n^2) since it just checks all the pixels and sets them to black or white appropriately.The execution time was roughly the same on all processes except doing all RGB, which took a little longer. Besides that the performance is pretty good but will slow down significantly the larger the Prewitt window becomes.

V. Conclusion

This assignment introduces us to basic edge detection using the Prewitt 3x3 or 5x5 operator. There are more sophisticated edge detection processes out there like Canny edge detection but the Prewitt seems to find the edges pretty well. I also noticed that when comparing 3x3 to 5x5, the larger window seems to make the edges more predominate. It is very interesting to see how simply calculating the gradient and thresholding it can show the edges of an image. It will be neat to see how other edge detection methods work in the future.

References

[1] Example Project Code and Example Report from TA website.
[2] My Project 1.
[3] My Project 2.