**CAP 5400 - Digital Image Processing**

**Project 3: Sobel Edge Detection and OpenCV**

Yu Liang

Department of Computer Sc. and Engineering.

University of South Florida,

Tampa, Florida

**1. Introduction**

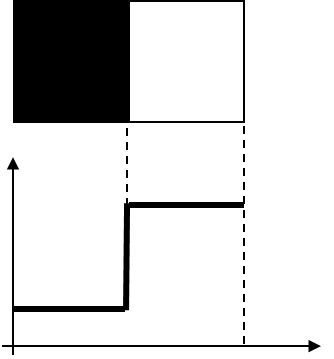
This assignment focuses on efficient implementation of basic edge detection and use of OpenCV using grey images wrote in C++. Edge detection is an image processing technique for finding the boundaries of objects within images. It works by detecting discontinuities in brightness. Edge detection is used for image segmentation and data extraction in areas such as image processing, computer vision, and machine vision. In this project, we will conduct Sobel edge detection and Canny edge detection.

OpenCV (Open Source Computer Vision Library: http://opencv.org) is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms. OpenCV has a modular structure, which means that the package includes several shared or static libraries. We will use OpenCV to duplicate the experiment we did in the first two labs and compare the performance.

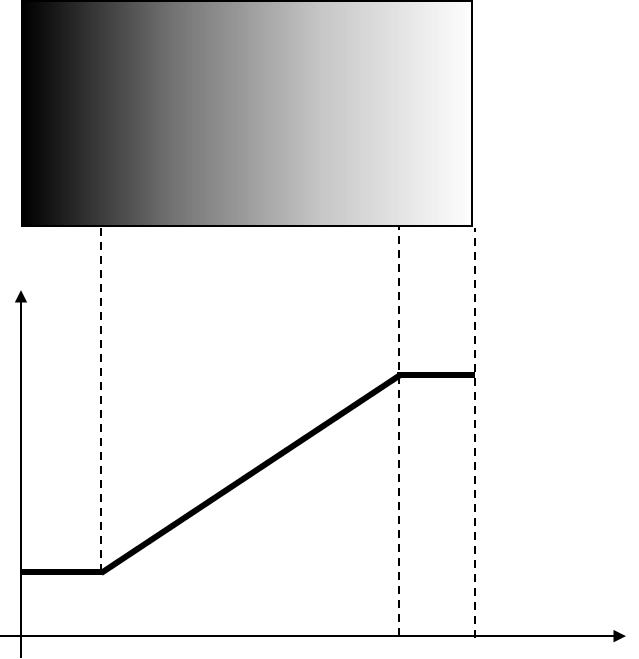
**2. Description of algorithms**

**2.1. Edge Detection**

Edge detection is the ability to measure gray-level transitions in a meaningful way. There are two types of gray-level transition:



Ideal



Ramp

There are many ways to perform edge detection. However, the most may be grouped into two categories, gradient and Laplacian. The gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image. The Laplacian method searches for zerocrossings in the second derivative of the image to find edges. This first figure shows the edges of an image detected using the gradient method (Roberts, Prewitt, Sobel) and the Laplacian method (Marrs-Hildreth).

The gradient of the image I(x,y) at location (x,y), is the vector:



The magnitude of the gradient:



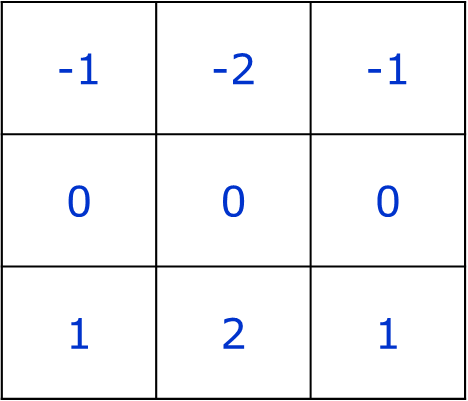
The direction of the gradient vector:



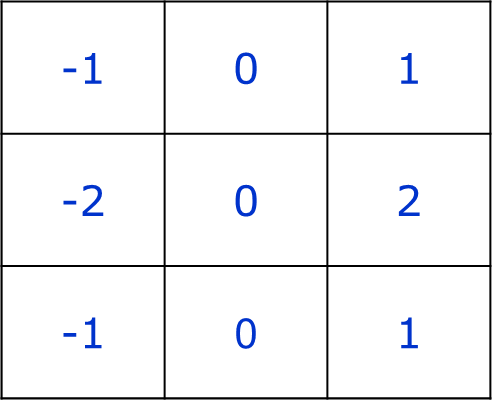
**2.2. Sobel Edge Detection**

The Sobel operator, sometimes called the Sobel–Feldman operator or Sobel filter, is used in image processing and computer vision, particularly within edge detection algorithms where it creates an image emphasising edges.

In theory at least, the operator consists of a pair of 3×3 convolution kernels as shown in Figure 1. One kernel is simply the other rotated by 90°.









**2.3. Canny Edge Detection**

The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986. Canny also produced a computational theory of edge detection explaining why the technique works.

The image is convolved with a Gaussian filter before gradient evaluation:

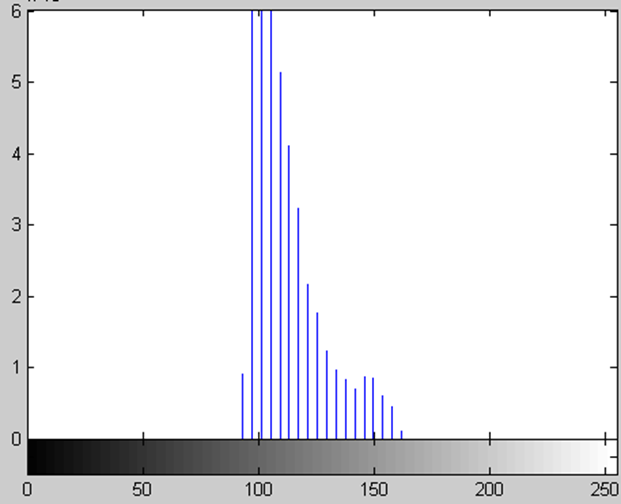


The gradient is calculated (using any of the four methods described in the previous slides), for each pixel in the picture. If the absolute value exceeds a threshold, the pixel belongs to an edge. The Canny method uses two thresholds, and enables the detection of two edge types: strong and weak edge. If a pixel's magnitude in the gradient image, exceeds the high threshold, then the pixel corresponds to a strong edge. Any pixel connected to a strong edge and having a magnitude greater than the low threshold corresponds to a weak edge.

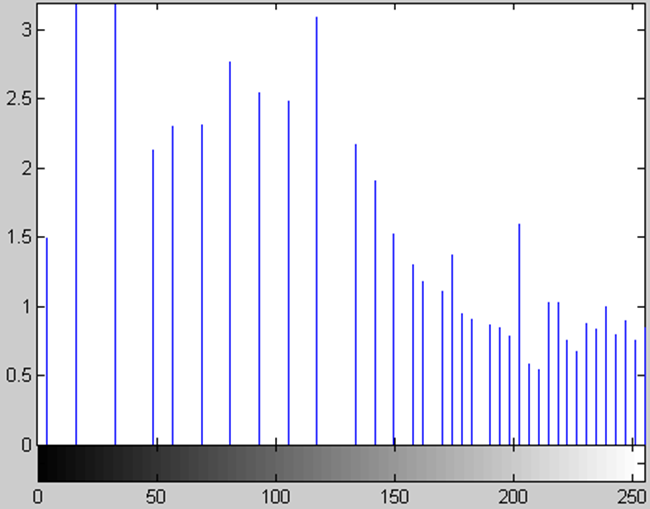
**2.4. Histogram Equalization**

Histogram is a graphical representation of the intensity distribution of an image. In simple terms, it represents the number of pixels for each intensity value considered.

Histogram Equalization is a computer image processing technique used to improve contrast in images. It accomplishes this by effectively spreading out the most frequent intensity values, i.e. stretching out the intensity range of the image. This method usually increases the global contrast of images when its usable data is represented by close contrast values. This allows for areas of lower local contrast to gain a higher contrast.



Histogram of low contrast image



Histogram of equalized image

**3. Results**

**3.1. Sobel Edge Detection**

First we applied Sobel 3\*3 operator with threshold value of 50 on the tree image as below. Figure b shows the gradient amplitude as intensity image. Figure c shows binary edge image of thresholded 50 gradient amplitude. We can see the boundaries of the branches of the tree.



1. Original image

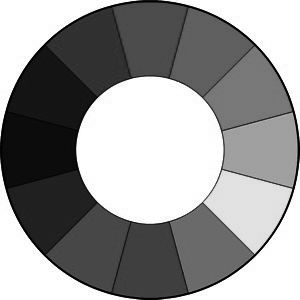


1. image of gradient amplitude as intensity image

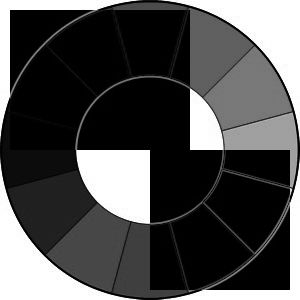


1. binary edge image of thresholded gradient amplitude

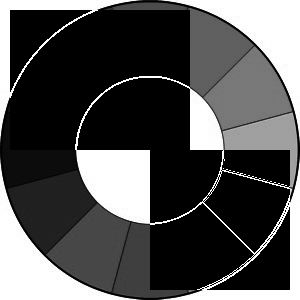
Second, we applied Sobel 3\*3 operator with threshold value of 48 on the wheel image as below. Figure b shows the gradient amplitude as intensity image. Figure c shows binary edge image of thresholded 48 gradient amplitude. In figure d, we set up the direction to be only 45 degree, so only 45 degree boundaries show up.



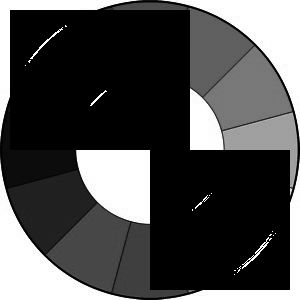
1. Original image



1. image of gradient amplitude as intensity image



1. binary edge image of thresholded amplitude



1. binary edge image of thresholded amplitude and direction 45 degree with +-10 degree range

**3.2. Sobel Edge Detection in OpenCV**

First we applied Sobel 3\*3 operator with threshold value of 48 in OpenCV on the lena image as below. Figure b shows the gradient amplitude as intensity image in X direction. Figure c shows the gradient amplitude as intensity image in Y direction. Figure d shows combined X and Y edge image.



1. Original image



1. image of gradient amplitude in X direction



1. image of gradient amplitude in Y direction



1. image of gradient amplitude of combine X and Y direction

**3.3. Canny Edge Detection in OpenCV**

First we applied Canny 3\*3 operator with threshold value of 50 in OpenCV on the lena image as below. Figure b shows the edge image.



1. Original image



1. Binary edge image of thresholded amplitude with lower threshold is 48 and higher threshold is 3\*48.

**3.4. Histogram Equalization in OpenCV**

First we applied Histogram Equalization function EqualizeHis in OpenCV on the slope image as below. Figure b shows histogram Equalization processed using OpenCV. Figure c shows histogram stretching processed in the last project



1. Original image

b. Histogram Equalization processed



1. Histogram stretching processed in the last project

**4. Discussion**

Discuss results and performance of Sobel Edge Detection: Sobel edge detection can successfully detect the edge in the tree image and after applied the appropriate threshold it can generate accurate binary image to show the boundaries. It can also show the boundaries in certain degree like 45 degree and 135 degree.

Discuss results and performance of Sobel Edge Detection in OpenCV: Sobel edge detection in OpenCV can successfully detect the edge in the lena image. It can generate amplitude image on X or Y direction and combine the two direction. But it can’t generate on any direction.

Discuss results and performance of Canny Edge Detection in OpenCV: Canny edge detection in OpenCV can successfully detect the edge in the lena image when set up the appropriate upper and lower thresholds. It can generate binary image. The boundaries are thin lines that outline the figure which is better than Sobel operator.

Discuss results of color processing: Histogram Equalization in OpenCV.

The EqualizeHis function in OpenCV can make the image contract look better. However when comparing to the histrogram

stretching used in last project. The histogram stretching generate a more natural result. But the histogram equalization generate a more clear image.

Yu Liang

Project 3

Sobel Edge Detection and OpenCV

\*\*\*\*\*

This software is architectured as follows. This software can work on grad server.

iptools -This folder hosts the files that are compiled into a static library.

image - This folder hosts the files that define an image.

utility- this folder hosts the files that students store their implemented algorithms.

lib- This folder hosts the static libraries associated with this software.

project- This folder hosts the files that will be compiled into executables.

bin- This folder hosts the binary executables created in the project directory.

OpenCV - OpenCV code, input image, output image

\*\*\* INSTALATION \*\*\*

On Linux

Sobel:

Enter the project directory in terminal and run make

As a result you should get iptool in project/bin directory.

OpenCV:

Enter the OpenCV directory in terminal and run:

g++ Sobel.cpp -o Sobel `pkg-config opencv --cflags --libs`

to compile Sobel

replace Sobel.cpp with Canny.cpp to compile Canny.

replace Sobel.cpp with EqualizeHis.cpp to compile Histogram Equalization.

\*\*\* FUNCTIONS \*\*\*

1. Sobel : Sobel33

Use the Sobel operator (3x3) to compute dx, dy, gradient amplitude and edge direction.

generate image of gradient amplitude as intensity image

generate binary edge image of thresholded gradient amplitude

generate binary edge image of thresholded gradient amplitude and direction

2. OpenCV : Sobel, Canny, EqualizeHis

Use the Sobel operator (3x3) in OpenCV

Use Canny module to do edge detection in OpenCV, lower threshold is 48

Implement histogram equalization using OpenCV (call the function)

ROI\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sobel:

50 150 100 200 50 45

200 200 100 100 50 45 # Parameters for ROI1: x, y, sx, sy, T, TD

#need a space after 45#

\*\*\* PARAMETERS FILE \*\*\*

There are for parameters:

1. the input file name;

2. the output file name;

3. the name of the filter. Use "Sobel33" for your filters;

\*\*\* Run the program: ./iptool parameters.txt

parameters.txt Example:

tree.pgm tree\_amplitude.pgm Sobel33 50

wheel.pgm wheel\_amplitude.pgm Sobel33 50