Sobel Operator and Canny Edge Detector
ECE 480 Fall 2013
Team 4
Daniel Kim

Executive Summary

In digital image processing (DIP), edge detection is an important subject matter. There are numerous edge detection methods such as Prewitt, Kirsch, and Robert cross. Two different types of edge detectors are explored and analyzed in this paper: Sobel operator and Canny edge detector. The performance of two edge detection methods are then compared with several input images.

Keywords:

Digital Image Processing, Edge Detection, Sobel Operator, Canny Edge Detector, Computer Vision, OpenCV

Table of Contents

1) Introduction

Edge detection is a crucial step in object recognition. It is a process of finding sharp discontinuities in an image. The discontinuities are abrupt changes in pixel intensity which characterize boundaries of objects in a scene. In short, the goal of edge detection is to produce a line drawing of the input image. The extracted features are then used by computer vision algorithms, e.g. recognition and tracking.

A classical method of edge detection involves the use of operators, a two dimensional filter. An edge in an image occurs when the gradient is greatest. The operator works by identifying these large gradients to find the edges. There are vast amount of operators designed to detect certain types of edges. The operators can be configured to search for vertical, horizontal, or diagonal edges. One major problem with edge detection is when noise is present in images. It is not enough to simply reduce the noise, because the image will be either distorted or blurred. Fortunately, the operator can average enough data to discount localized noisy pixels. However, the change in intensity is not always a step change. The intensity change can also be gradual where the operator then has to be modified for proper edge detection. Consequently, there are problems of missing true edges, false edge detection, and high computational time. In next two sections, Sobel and Canny detection methods are investigated.

2.1) Sobel Operator –Background

The Sobel operator is a discrete differential operator. The operator utilizes two 3x3 kernels: one estimates the gradient in the x-direction, while the other one estimates the gradient in the y-direction.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Figure 1: Sobel Operator uses 3x3 Kernel Masks

The image is convolved with both kernels to approximate the derivatives in horizontal and vertical change. At each given point, magnitude of the gradient can be approximated with:

$$G = \sqrt{G_x^2 + G_y^2}$$

However, it is faster to compute the gradient magnitude with:

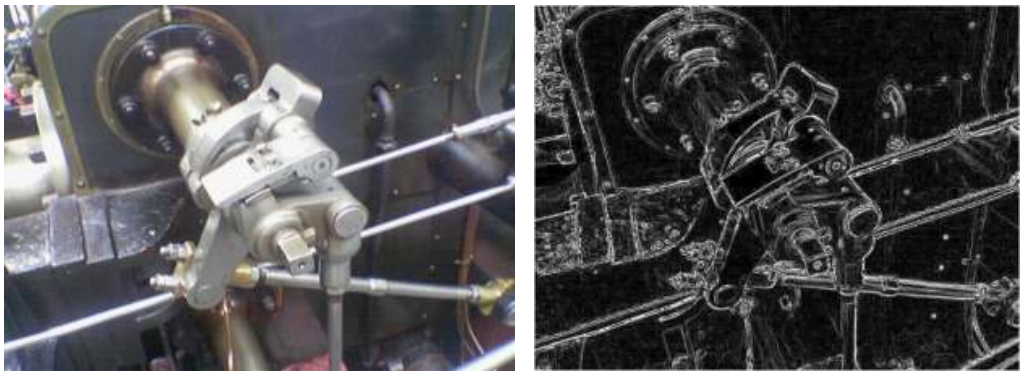
$$G = |G_x| + |G_y|$$

Due to Sobel operator's smoothing effect (Gaussian smoothing), it is less sensitive to noise present in images. On the other hand, smoothing affects the accuracy of edge detection. In other words, the Sobel method does not produce image with high accuracy for edge detection, but its quality is adequate enough to be used in numerous applications.

2.2) Sobel Operator - Examples

**Note: Refer to appendix for tutorial code with OpenCV (image processing and computer vision algorithms)*

Example 1:



(a) (b)

Figure 2: (a) Original image. The resulting image after applying Sobel method is shown in (b).

Example 2:

(a) (b)

Figure 3: (a) Original image. The resulting image after applying Sobel method is shown in (b).

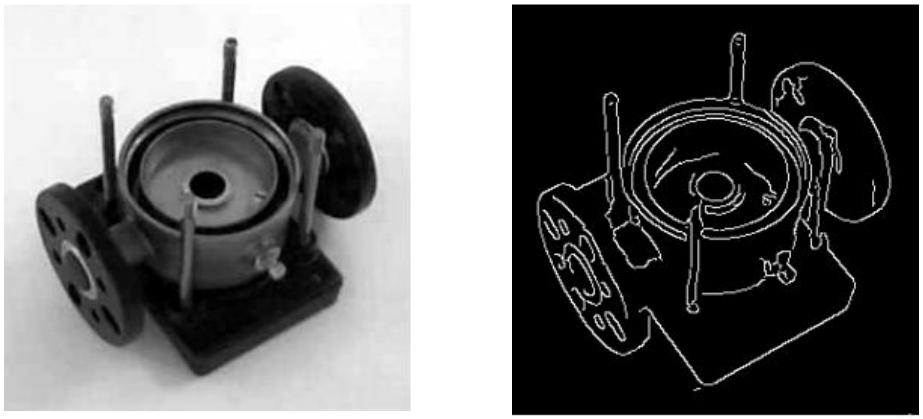
3.1) Canny Edge Detector – Background

The Canny operator is widely known as the optimal detector, developed by John F. Canny in 1986. There are multiple steps to implement the Canny operator. First, a Gaussian filters is used to smooth the image to remove noise in an image. Second, compute the gradient magnitude similar to Sobel operator. Third, non-maximum suppresion is applied in which the algorithm removes pixels that are not part of an edge. The final step involves the use of hysteresis thresholding along edges. Hystersis uses two thresholds, upper and lower. If a pixel gradient is higher than the upper threshold, then the pixel will be marked as an *edge*. If a pixel gradient is below the lower threshold, then the pixel will be discarded. Finally, if the pixel gradient is between the two thresholds, then only the pixel that is connected above the upper threshold is marked as an *edge*.

3.2 Canny Edge Detector – Example

**Note: Refer to appendix for tutorial code with OpenCV*

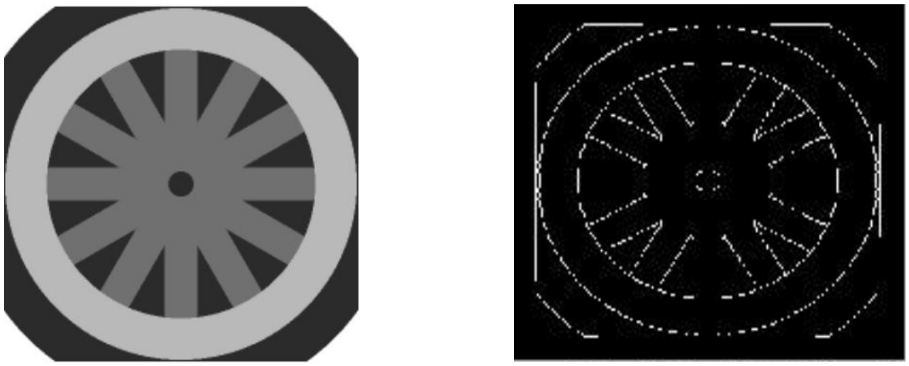
Example 1:



(a) (b)

Figure 4: (a) Original image. The resulting image after applying Canny operator (b).

Example 2:



(a) (b)

Figure 5: (a) Original image. The resulting image after applying Canny operator (b).

4.1) Comparison of Sobel operator and Canny edge detectors

I) Sobel Operator

The primary advantages of the Sobel operator lie in its simplicity. The Sobel method provides a approximation to the gradient magnitude. Another advantage of the Sobel operator is it can detect edges and their orientations. In this cross operator, the detection of edges and their orientations is said to be simple due to the approximation of the gradient magnitude.

On the other hand, there exist some disadvantages of the Sobel method. That is, it is sensitive to the noise. The magnitude of the edges will degrade as the level of noise present in image increases. As a result, Sobel operator accuracy suffers as the magnitude of the edges decreases. Overall, the Sobel method cannot produce accurate edge detection with thin and smooth edge.

II) Canny Edge Detector

With Gaussian filter, any noise present in an image can be removed. The next advantage is enhancing the signal with respect to the noise ratio. This is done by non-maxima suppression method as it results in one pixel wide ridges as the output. The third advantage is better detection of edges especially in noisy state by applying thresholding method. The effectiveness of Canny method is affected by adjustable parameters. Small filters are desirable for detection of small, sharp lines, since it causes fewer instances of blurring. Large filters are desirable for detecting larger, smoother edges. However, it causes higher instances of blurring.

The main disadvantage of Canny edge detector is that it is time consuming, due to its complex computation.

Operator	Strengths	Weaknesses
Sobel	Simple. Detects edges and their orientation	Inaccurate and sensitive to nosie
Canny	Smoothing effect to remove noise. Good localization and response. Enhances signal to noise ratio. Immune to noisy environment.	Difficult to implement to reach real time response. Time consuming.

Table 1: Compares strengths and weaknesses of two detectors.

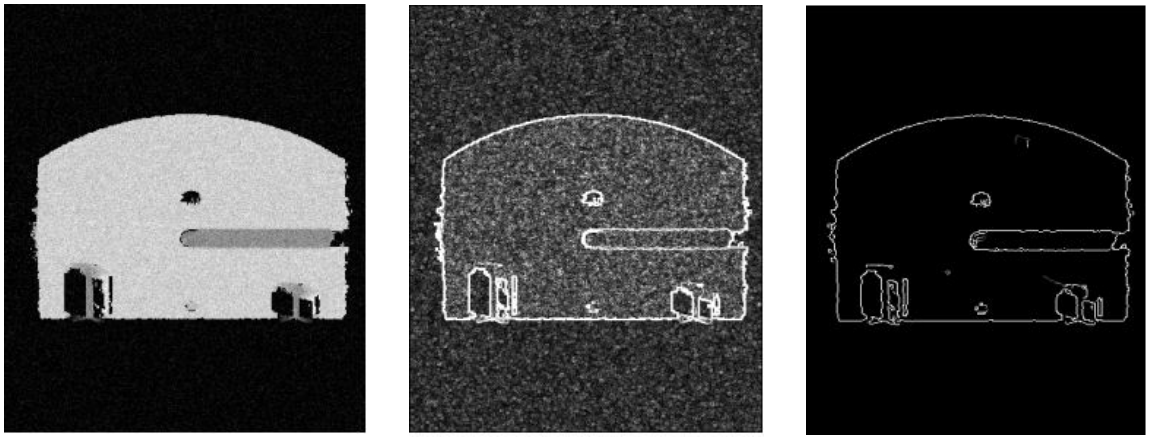
4.2) Comparison of Sobel operator and Canny edge detectors – Example

Example 1:

(a) (b) (c)

Figure 6: (a) Original image. Sobel (b). Canny (c)

Example 2:



(a) (b) (c)

Figure 7: (a) Original image. Sobel (b). Canny (c)

5) Conclusion

Edge detection is very useful in digital image processing. Out of numerous edge detection methods, this paper discussed and analyzed the strengths and weaknesses of Sobel and Canny edge detection methods. Sobel operator is simple, but its accuracy suffers in noisy conditions. On the contrary, Canny edge detector has many favorable features such as smoothing effect to remove noise, and improving signal to noise ratio through a process known as non-maximal suppression. Complex algorithms used in Canny method makes it time consuming and difficult to implement to reach real time response speeds. Despite its disadvantage, however, it is recommended to use Canny method over Sobel method.

6) Appendix

6.1) OpenCV function Sobel tutorial code.

```
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <stdlib.h>
#include <stdio.h>
using namespace cv;
/** @function main */
int main( int argc, char** argv )
{
  Mat src, src_gray;
  Mat grad;
  char* window_name = "Sobel Demo - Simple Edge Detector";
  int scale = 1;
  int delta = 0;
  int ddepth = CV_16S;

  int c;

  /// Load an image
  src = imread( argv[1] );

  if( !src.data )
  { return -1; }

  GaussianBlur( src, src, Size(3,3), 0, 0, BORDER_DEFAULT );

  /// Convert it to gray
  cvtColor( src, src_gray, CV_RGB2GRAY );

  /// Create window
  namedWindow( window_name, CV_WINDOW_AUTOSIZE );

  /// Generate grad_x and grad_y
  Mat grad_x, grad_y;
  Mat abs_grad_x, abs_grad_y;

  /// Gradient X
  //Scharr( src_gray, grad_x, ddepth, 1, 0, scale, delta, BORDER_DEFAULT );
  Sobel( src_gray, grad_x, ddepth, 1, 0, 3, scale, delta, BORDER_DEFAULT );
  convertScaleAbs( grad_x, abs_grad_x );

  /// Gradient Y
  //Scharr( src_gray, grad_y, ddepth, 0, 1, scale, delta, BORDER_DEFAULT );
  Sobel( src_gray, grad_y, ddepth, 0, 1, 3, scale, delta, BORDER_DEFAULT );
  convertScaleAbs( grad_y, abs_grad_y );

  /// Total Gradient (approximate)
  addWeighted( abs_grad_x, 0.5, abs_grad_y, 0.5, 0, grad );
  imshow( window_name, grad );
  waitKey(0);
  return 0;
  }
```

6.2) OpenCV function Canny tutorial code

```
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <stdlib.h>
#include <stdio.h>
using namespace cv;
/// Global variables
Mat src, src_gray;
Mat dst, detected_edges;
int edgeThresh = 1;
int lowThreshold;
int const max_lowThreshold = 100;
int ratio = 3;
int kernel_size = 3;
char* window_name = "Edge Map";
/**
 * @function CannyThreshold
 * @brief Trackbar callback - Canny thresholds input with a ratio 1:3
 */
void CannyThreshold(int, void*)
{
  /// Reduce noise with a kernel 3x3
  blur( src_gray, detected_edges, Size(3,3) );
  /// Canny detector
  Canny( detected_edges, detected_edges, lowThreshold, lowThreshold*ratio,
kernel_size );
  /// Using Canny's output as a mask, we display our result
  dst = Scalar::all(0);
  src.copyTo( dst, detected_edges);
  imshow( window_name, dst );
 }
/** @function main */
int main( int argc, char** argv )
{
  /// Load an image
  src = imread( argv[1] );
  if( !src.data )
  { return -1; }
  /// Create a matrix of the same type and size as src (for dst)
  dst.create( src.size(), src.type() );
  /// Convert the image to grayscale
  cvtColor( src, src_gray, CV_BGR2GRAY );
  /// Create a window
  namedWindow( window_name, CV_WINDOW_AUTOSIZE );
  /// Create a Trackbar for user to enter threshold
  createTrackbar( "Min Threshold:", window_name, &lowThreshold, max_lowThreshold,
CannyThreshold );
  /// Show the image
  CannyThreshold(0, 0);
  /// Wait until user exit program by pressing a key
  waitKey(0);
  return 0;
  }
```

References

[1] http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm

[2] http://homepages.inf.ed.ac.uk/rbf/HIPR2/canny.htm

[3] http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html

[4] http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/sobel_derivatives/sobel_derivatives.html

[5] http://comp.utm.my/pars/files/2013/04/Comparison-of-Canny-and-Sobel-Edge-Detection-in-MRI-Images.pdf

[6] R. M. Haralick. “Digital step edges from zero crossing of the second directional derivatives,” IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-6, no. 1, pp. 58-68, Jan. 1984.