Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 2, Issue. 6, June 2013, pg.427 – 434

RESEARCH ARTICLE

Various Edge Detection Methods for Foreground Detection

[1] Gurjeet kaur Seerha, [2] Rajneet Kaur

[1]Department of Computer Science, Sri Guru Granth Sahib World University, Fatehgarh Sahib, Punjab, India
[2]Assistant Professor, Department of Computer Science, Sri Guru Granth Sahib World University, Fatehgarh Sahib, Punjab, India

[1] nonuseerha42@gmail.com ; [2] rosy.rajneet@gmail.com

Abstract— In this paper, we study different edge detection techniques, edge detection is one of the most commonly used operations in image analysis, and there are probably more algorithms in the literature for enhancing and detecting edges than any other single subject. The goal of edge detection process in a digital image is to determine the frontiers of all represented objects based on automatic processing of the colour or gray level information in each present pixel. An edge is the boundary between an object and the background, and indicates the boundary between overlapping objects. This means that if the edges in an image can be identified accurately, all of the objects can be located and basic properties such as area, perimeter, and shape can be measured. Since computer vision involves the identification and classification of objects in an image, edge detections is an essential tool.

Key Terms: - Edges; gradient, canny; Sobel; prewitt

I. INTRODUCTION

Edge detection refers to the process of identifying and locating sharp discontinuities in an image [1]. Edge detection technique is usually applied on gray–scale image. The discontinuities are abrupt changes in pixel intensity which characterize boundaries of objects in a scene. Classical methods of edge detection involve convolving [2] the image with an operator (a 2-D filter [3]), which is constructed to be sensitive to large gradients in the image while returning values of zero in uniform regions.

A grey scale image can be represented by two-dimensional values of pixel in which each pixel represents the intensity. In image processing, the digitization process can be done by sampling and quantization of continuous data. The sampling process samples the intensity of the continuous-tone image, such as a monochrome, color or multi-spectrum image, at specific locations on a discrete grid. The grid defines the sampling resolution. The quantization process converts the continuous or analog values of intensity brightness into discrete data, which corresponds to the digital brightness value of each sample, ranging from black, through the grays, to white. A digitized sample is referred to as a picture element, or pixel. The digital image contains a fixed number of rows and columns of pixels. Pixels are like little tiles holding quantized values that represent the brightness at the points of the image. Pixels are parameterized by position, intensity and time. Typically, the pixels are stored in computer memory as a raster image or raster map, a two-dimensional array of small integers. Image is stored in numerical form which can be manipulated by a computer. A numerical image is divided into a matrix of pixels (picture elements). Digital image processing allows one to enhance image features of interest while attenuating detail irrelevant to a given application, and then extract useful information about the scene from the enhanced image. Images are produced by a variety of physical devices, including still and video cameras, x-ray devices, electron microscopes, radar, and ultrasound, and used for a variety of purposes, including entertainment,

medical, business (e.g. Documents), industrial, military, civil (e.g. traffic), security, and scientific. The goal in each case is for an observer, human or machine, to extract useful information about the scene being imaged.
Extremely large number of edge detection operators available, each designed to be sensitive to certain types of edges. Variables involved in the selection of an edge detection operator include:

- *Edge orientation:* The geometry of the operator determines a characteristic direction in which it is most sensitive to edges. Operators can be optimized to look for horizontal, vertical, or diagonal edges.
- *Noise environment:* Edge detection is difficult in noisy images, since both the noise and the edges contain high-frequency content. Attempts to reduce the noise result in blurred and distorted edges. This results in less accurate localization of the detected edges. Operators used on noisy images are typically larger in scope, so they can average enough data to discount localized noisy pixels.
- *Edge structure:* Not all edges involve a step change in intensity. Effects such as refraction or poor focus can result in objects with boundaries defined by a gradual change in intensity. The operator needs to be chosen to be responsive to such a gradual change in those cases.[4]

There are many ways to perform edge detection. However, the majority of different methods may be grouped into two categories:

- *Gradient:* The gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image. For a gradient image f(x, y), at location (x, y), where x and y are the row and column coordinates respectively, typically consider the two directional derivatives. The two functions that can be expressed in terms of the directional derivatives are the gradient magnitude and the gradient orientation. The gradient magnitude is defined by:

$$g(x,y) = (\Delta x^2 + \Delta y^2)^{1/2}$$

$$\Delta x = f(x+n,y) - f(x-n,y) \quad and \quad \Delta y = f(x,y+n) - f(x,y-n)$$

Where n is a small integer usually unity. This quantity gives the maximum rate of increase of f(x, y) per unit distance in the gradient orientation of g(x, y). The gradient orientation is also an important quantity. The gradient orientation is given by:

$$\theta(x,y) = \tan\left(\frac{\Delta y}{\Delta x}\right)$$

Here the angle is measured with respect to the x-axis. Then direction of the edge at (x, y) is perpendicular to the direction of the gradient vector at that point. The other method of calculating the gradient is given by estimating the finite difference. Suppose we have the following signal, with an edge shown by the jump in intensity below:

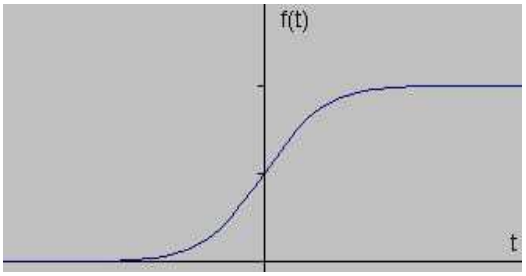


Figure: 1

If we take the gradient of this signal (which, in one dimension, is just the first derivative with respect to t) we get the following:

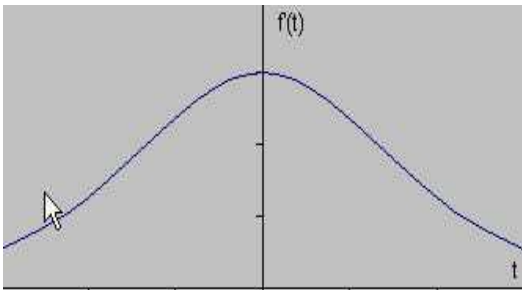


Figure: 2

- *Laplacian*: The Laplacian method searches for zero crossings in the second derivative of the image to find edges. An edge has the one-dimensional shape of a ramp and calculating the derivative of the image can highlight its location. Suppose we have the following signal, with an edge shown by the jump in intensity below:

Clearly, in above figure: 2 the derivative shows a maximum located at the center of the edge in the original signal. This method of locating an edge is characteristic of the "gradient filter" family of edge detection filters and includes the Sobel method. A pixel location is declared an edge location if the value of the gradient exceeds some threshold. As mentioned before, edges will have higher pixel intensity values than those surrounding it. So once a threshold is set, you can compare the gradient value to the threshold value and detect an edge whenever the threshold is exceeded [5]. Furthermore, when the first derivative is at a maximum, the second derivative is zero. As a result, another alternative to finding the location of an edge is to locate the zeros in the second derivative. This method is known as the Laplacian and the second derivative of the signal is shown below:

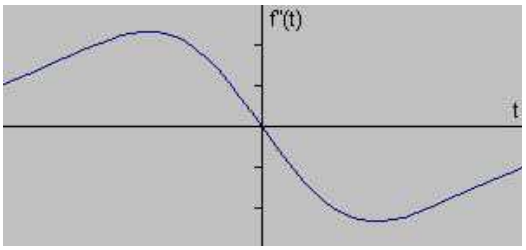


Figure: 3

II. EDGE DETECTION TECHNIQUES

A. Robert Edge Detector

The calculation of the gradient magnitude of an image is obtained by the partial derivatives G_x and G_y at every pixel location. The simplest way to implement the first order partial derivative is by using the Roberts cross gradient operator. Therefore

G_x = f (i, j) - f (i+1, j+1)
G_y = f (i+1, j) – f (i, j+1)

The above partial derivatives can be implemented by approximating them to two 2x2 masks. The Roberts operator masks are:

-1	0
0	1

G_x

0	-1
1	0

G_y

Figure: 4 Roberts operator

These filters have been the shortest support, thus the position of the edges is more accurate, but the problem with the short support of the filters is its vulnerability to noise. It also produces very weak response to genuine edges unless they are very sharp.

B. Prewitt Edge detector

The Prewitt edge detector is a much better operator than Roberts's operator. This operator having a 3 x 3 masks deals better with the effect of noise. An approach using the masks of size 3 x 3 are given below, the arrangement of pixels about the pixels
[i, j].

a_0	a_1	a_2
a_7	[i,j]	a_3
a_6	a_5	a_4

The partial derivatives of the Prewitt operator are calculated as:

$G_x = (a_6 + ca_5 + a_4) - (a_0 + ca_1 + a_2)$

$G_y = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6)$

The constant c implies the emphasis given to pixels closer to the center of the mask. G_x And G_y are the approximation at [i, j]. Setting c= 1, the Prewitt operator is obtained. Therefore the Prewitt masks are as follows:

-1	-1	-1
0	0	0
1	1	1

G_x

-1	0	1
-1	0	1
-1	0	1

G_y

These masks have longer support. They differentiate in one direction and average in the other direction, so the edge detector is less vulnerable to noise.

C. Sobel Edge Detector

The Sobel operator is the most known among the classical methods. The Sobel edge detector applies 2D spatial gradient convolution operation on an image. It uses the convolution masks shown in to compute the gradient in two directions (i.e. row and column orientations), and then works out the pixels 'gradient through $g=|gr+gc|$. Finally, the gradient magnitude is threshold. The Sobel edge detector is very much similar to the Prewitt edge detector. The difference between the both is that the weight of the center coefficient is 2 in the Sobel operator. The partial derivatives of the Sobel operator are calculated as:

$G_x = (a_6 + 2a_5 + a_4) - (a_0 + 2a_1 + a_2)$

$G_y = (a_2 + 2a_3 + a_4) - (a_0 + 2a_7 + a_6)$

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

G_x

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

G_y

Although the Prewitt masks are easier to implement than Sobel masks.

D. Laplacian of Gaussian (LoG)

The Laplacian is a 2-D isotropic measure of the 2nd spatial derivative of an image. The Laplacian of an image highlights regions of rapid intensity change and is therefore often used for edge detection. The Laplacian is often applied to an image that has first been smoothed with something approximating a Gaussian Smoothing filter in order to reduce its sensitivity to noise. The operator normally takes a single gray level image as input and produces another gray level image as output.

The Laplacian *L(x, y)* of an image with pixel intensity values *I(x, y)* is given by:

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \qquad (1.1)$$

Since the input image is represented as a set of discrete pixels, we have to find a discrete convolution kernel that can approximate the second derivatives in the definition of the Laplacian.

E. Canny's Edge Detection Algorithm

The Canny edge detection operator was developed by John F. Canny in 1986 and uses a multi-stage algorithm to detect a wide range of edges in images. Algorithm was developed to improve [6] the existing method of edge detection. The first and most obvious is low error rate: it is important that edges occurring in images should not be missed and that there be no response to non-edges. The second criterion is that the edge points be well localized. In other words, the distance between the edge pixels as found by the detector and the actual edge is to be at a minimum. A third criterion is to have only one response to a single edge. Canny edge detection algorithm runs mainly in five sequential steps:

1. Smoothing: By applying Gaussian filter, smoothing of and image is done to reduce noise.
2. Finding gradients: The edges have been marked where the gradients of the image is having large magnitudes.
3. Non-maximum suppression: Only local maxima have been marked as edges
4. Double thresholding: Potential and actual edges are determined by thresholding.
5. Edge tracking by hysteresis: Edges that are not connected with strong edges have been suppressed.

1) Smoothing: Images taken from a camera will contain some amount of noise. As noise can mislead the result in finding edges, we have to reduce the noise. Therefore the image is first smoothed [7] by applying a Gaussian filter. The kernel of a 5x5Gaussian filter with a standard deviation of σ= 1.4 is shown in Equation 1.2

$$B=\frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \qquad (1.2)$$

2) Finding Gradients: The principle of canny algorithm is based on finding edges where the intensity of the grey image changes to maximum value. These points are marked by determining the gradient of the image. Gradient for each pixel is calculated by applying Sobel operator. For each pixel, partial gradient towards x and y direction is determined respectively by applying the following equation:

$$|G| = \sqrt{G_x^{\,2} + G_y^{\,2}} \qquad (1.3)$$

By applying Pythagoras law to Equation 2.2, calculate the magnitude of gradient (strength of the edge) for each point. The directions of the edges have been calculated using the following Equation.

$$\theta = \tan^{-1}\frac{G_x}{G_y} \qquad (1.4)$$

The edges have been marked where the gradients of the image have large magnitudes. Sobel G_x and G_y masks are used to generate partial derivatives ∂x and ∂y (partial spatial derivatives) of the image. Spatial gradient value (d f (x; y)) is obtained from these partial derivatives.

3) Non-Maximum Suppression: This is necessary to convert the blurred edges in the image of the gradient magnitudes to sharp edges. Actually this is performed by considering only all local maxima in the gradient image and deleting everything rest. The algorithm is applied for each pixel in the gradient image:

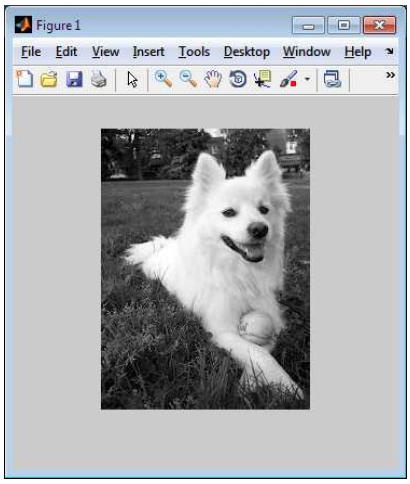
- Gradient direction is rounded to the nearest 45 degree. Thus, it can choose one of 8-connected neighbourhood.
- The edge strength of the current pixel is compared with the edge strength of the pixel in the positive and negative gradient direction. I.e. if the gradient direction is north ($\theta = 90^0$), compare with the pixels to the north and south.
- If the edge strength of the current pixel is largest; preserve the value of the edge strength. Otherwise, suppress (i.e. remove) the value.

4) Double Thresholding: After the non-maximum suppression step, the edge pixels are still marked with their strength pixel-by-pixel. Many of these may be true edges of the image, but some might be caused by noise or color variations for instance due to the rough surface. In order to get rid of such unpleasant situation, we can apply thresholding so that only edges stronger than a certain value would be preserved.

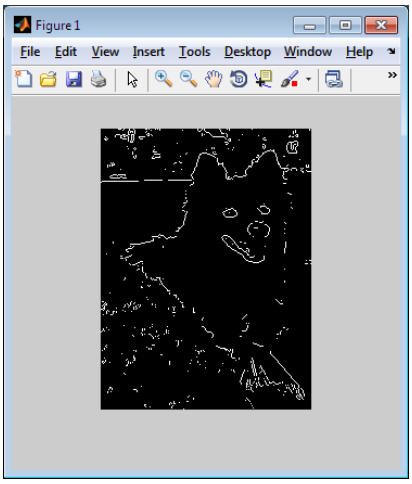
5) Edge Tracking by Hysteresis: Strong edges are referred as certain edges, and can immediately be included in the final edge image. Weak edges are considered if and only if they are connected to strong edges. The logic is of course that noise and other small variations are unlikely to result in a strong edge (with proper adjustment of the threshold levels). Thus strong edges will (almost) only be due to true edges in the original image. The weak edges can either be true edges or due to noise/color variations.

III. RESULTS AND DISCUSSIONS

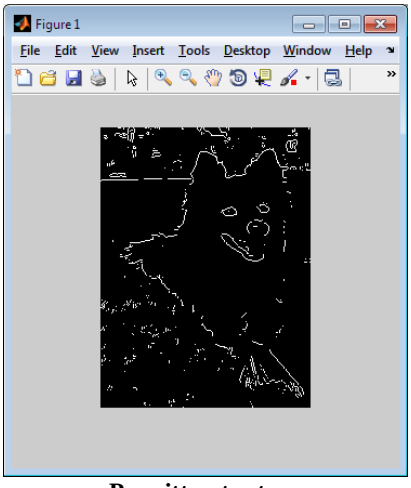
We experimented with a picture given below by passing it from many edge detection operators and the outputs are given bellow:



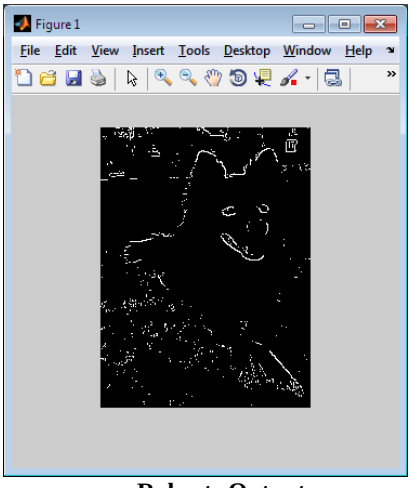
Original Image



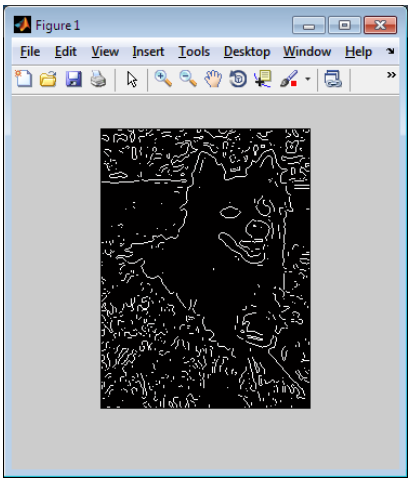
Sobel Output



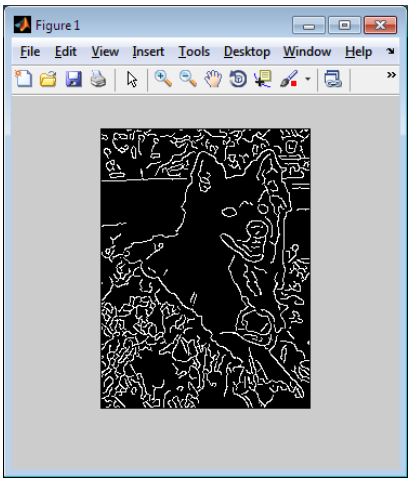
Prewitt output



Roberts Output



Log Output



Canny Output

Operator	Advantages	Disadvantages
Classical (Sobel, prewitt)	Simplicity, Detection of edges and their Orientations.	Sensitivity to noise, Inaccurate.
Zero Crossing (Laplacian, Second directional derivative)	Detection of edges and their orientations. Having fixed characteristics in all directions.	Responding to some of the existing edges, Sensitivity to Noise.
Laplacian of Gaussian(LoG)	Finding the correct places of edges, Testing wider area around the pixel.	Malfunctioning at the corners, curves and where the gray level intensity function varies. Not finding the orientation of edge because of using the Laplacian filter.
Gaussian(Canny)	Using probability for finding error rate, Localization and response. Improving signal to noise ratio, Better detection specially in noise conditions	Complex Computations, False zero crossing, Time consuming.

IV. CONCLUSION

We have reviewed and summarized the characteristics of different common operators. Each approach has its own advantages and drawbacks in different areas but experimental comparisons on different approaches show which approach is suitable for which image. Although, the all operators are roughly equivalent in case of noiseless image, but for practical applications, we conclude that the canny edge detection operator gives the best result in edge detection in a noiseless image.

REFERENCES

[1] O. Golan, S. Kiro, and I. Horovitz, "Method and System for Edge Detection," U.S. Patent 20 120 243 793, September, 2012.

[2] R. ONeil, "Convolution operators and L (p; q) spaces," Duke Mathematical Journal, vol. 30, no. 1, pp. 129–142, 1963.

[3] J. Weaver, Y. Xu, D. Healy, and L. Cromwell, "Filtering noise from images with wavelet transforms," Magnetic Resonance in Medicine, vol. 21, no. 2, pp. 288–295, 2005

[4] M. Barni, F. Bartolini, and A. Piva, "Improved wavelet-based watermarking through pixel-wise

masking," Image Processing, IEEE Transactions on, vol. 10, no. 5, pp. 783 –791, May 2001.

[5] F. Bergholm. "Edge focusing," in Proc. 8th Int. Conf. Pattern Recognition, Paris, France, pp. 597- 600, 1986

[6] L. Ding and A. Goshtasby, "On the Canny edge detector," Pattern Recognition, vol. 34, no. 3, pp. 721 – 725, 2001.

[7] X. Li, H. Zhao, X. Jin, and X. Qin, "Real-Time Image Smoothing Based on True Edges," in Computer-Aided Design and Computer Graphics (CAD/Graphics), 2011 12th International Conference on, sept. 2011,pp. 141 –145.

