Image Segmentation by Using Different Types of Edge Detection Algorithm with Different Thresholds

[1]Biswajit Basak, [2]Pritam Ghosh, [3]Tanushree Koley, [4]Sudipto Basak

[1]Assistant Professor, Dept of ECE, Hooghly Engineering &Technology College, Hooghly, India
[2,3]B.Tech Student, Dept of ECE, Abacus Institute of Engineering & Management, Hooghly, India
[4]M.Tech Student, Computer Science & Engineering., University of Kalyani, Kalyani, India

ABSTRACT:-Segmentation algorithms for images generally are based on one of two basic properties of image intensity values: discontinuity and similarity. In the first category, the approach is to position an image based on abrupt changes in intensity, such as edges. The principal approaches in the second category are based on partitioning an image into regions that are similar. In this paper we discuss a number of approaches into two categories just mentioned, as they apply to images. Edge detection has been staple of segmentation algorithms for many years. We compare different types of edge detection techniques like Sobel, Prewitt, Roberts, Laplacian of Gaussian(LoG),Zero crossing and Canny in this paper.

Keywords: Sobel, Prewitt, Roberts, LoG, Zero crossing, Canny

I. INTRODUCTION

Segmentation subdivides an image into its constituent regions or objects. The level to which the subdivision is carried depends on problem being solved. That is segmentation should stop when the objects of interest have been isolated. There are three basic techniques for detecting the basic types of discontinuities in digital image: points, lines and edges. Although point and line detection certainly are important in any discussion on image segmentation, edge detection is by far the most common approach f or detecting meaningful discontinuities in intensity values. Such discontinuities are detected by using first and second derivatives. The first order derivative of choice in image processing is the gradient. We repeat the pertinent equations here for convenience. The gradient of a 2-D function, f(x, y), is defined as the vector

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \qquad (\text{`}1)$$

The magnitude of this vector is

$$\nabla f = \text{mag}(\nabla f) = [g_x^2 + g_y^2]^{0.5} \tag{2}$$

$$= [(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial x})^2]^{0.5} \tag{3}$$

To simplicity computation, this quantity is approximated sometimes by omitting the square root operation,

$$\nabla f = |g_x| + |g_y| \tag{4}$$

These approximations still behaves as derivatives; that is, they are zero in areas of constant intensity and their values are related to the degree of intensity change in areas of variable intensity. A fundamental property of the gradient vector is that it points in direction of the maximum rate of change of f at coordinates (x,y). The angle at which this maximum rate of change occurs is

$$\alpha(x, y) = \tan^{-1}\left[\frac{g_x}{g_y}\right] \tag{5}$$

Methods for estimating g_x *and* g_y using function edge are discuss in this paper. For some of these estimators, it is possible to specify whether the edge detector is sensitive to horizontal or vertical edge or to both. The general syntax for this function is

$$[g, t] = \text{edge}(f, \text{'method'}, \text{parameters}) \tag{6}$$

Where f is input image, g is a logical array. Parameter t is optional; it gives the threshold used by edge to determine which gradient values are strong enough to be called edge point.

II. Pixel-based Approach

An image contains various regions corresponding to different objects or their parts in the scene. If we consider a single feature then the distribution of pixel values in the feature space is degenerated to a feature histogram and the boundary function to a threshold. One of the simplest kind of feature in gray level image is gray value at a pixel. Thus, image can be segmented by simple gray level thresholding method. Consider an image there exists a threshold t such that feature values of all pixel that actually belong to region of first type are less than or equal to t and gray value of all pixels that actually belong to regions of the second type are all greater than t. In this case the segmented image is obtained as

$$b(r, c) = \begin{cases} 1 \text{ if } p(r,c) \leq t \\ 0 \text{ if } p(r,c) > t \end{cases} \tag{7}$$

where p(r, c) is the feature value at pixel (r, c). If gray level is the feature then p(r, c) = g(r, c) for all (r, c). Note that, here 1 and 0 represent labels and not values. Thus, the use of thresholding techniques foe image segmentation needs to solve the following two problems. One problem is to choice of features/properties to achieve desired segmentation and second is selection of optimum threshold that would incur the least classification error.

III. Sobel Edge Detector

The Sobel edge detector computes the gradient by using the following discrete differences between row and columns of a 3X3 neighborhood where the center pixel in each row and column is weighted by 2 to provide smoothing. Function edge simply packages the preceding operations into one function call and adds other features, such as accepting a threshold value or determining a threshold automatically. In addition, edge contains edge detection techniques that are not implementable directly with imfilter.

The general calling syntax for the Sobel detector is

[g, t] = edge(f, ‘sobel’, T, dir) (8)

Where f is a input image, T is a specified threshold and dir specifies the preferred direction of the edge detected: ‘horizontally’, ‘vertically’, or ‘both’. Parameter t in the output is optional. It is the threshold value used by edge. If T is specified, then t=T. If T is not specified(or is empty,[ ]), edge sets t equal to a threshold it determines automatically and then uses for edge detection. One of the principal reason for including t in the output argument is to obtain an initial threshold value that can be modified and passed to the function in the subsequent calls. Function edge uses the Sobel detector as a default if the syntax g = edge(f), or [g, t] = edge(f), is used.

IV. Prewitt Edge Detector

The Prewitt edge detector uses the marks to approximate digitally the first derivatives g_x and $g_{y.}$ Its general calling syntax is

[g, t] = edge(f, ‘prewitt’, T, dir) (9)

The parameters of this function are identical to the Sobel parameter. The Prewitt detector is slightly simpler to implement computationally than the Sobel detector, but its trends to produce somewhat noisier results.

V. Roberts Edge Detector

The Roberts edge detector uses the marks to approximate digitally the first derivatives as differences between adjacent pixels. Its general calling syntax is

[g, t] = edge(f, ‘roberts’, T, dir) (10)

The parameters of this function are identical to the Sobel parameters. The Roberts detector is one of the oldest edge detectors in digital image processing and it also is the simplest. This detector is used considerably less than the others due in part to its limited functionality. However, it still is used frequently in hardware implementation where simplicity and speed are dominant factors.

VI. Laplacian of a Gaussian(LoG) Detector

Consider the Gaussian function

$$G(x, y) = e^{\frac{-x^2+y^2}{2\sigma^2}} \tag{11}$$

Where σ is the standard deviation. This is a smoothing function which, if convolved with an image, will blur it. The degree of blurring is determined by the value of σ. The Laplacian of this function is

$$\nabla^2 G(x, y) = \frac{\partial^2 G(x,x)}{\partial x^2} + \frac{\partial^2 G(x,x)}{\partial y^2}$$

$$= [\frac{x^2+y^2+2\sigma^2}{\sigma^4}]e^{\frac{-(x^2+y^2)}{2\sigma^4}} \tag{12}$$

For obvious reasons, this function is called LoG. Because the second derivative is a linear operation, convolution(filtering) as image with ∇^2 G(x, y) is the same as convolution the image with the smoothing function first and then computing the Laplacian of the result. This is the key concept underlying the LoG detector. We convolve the image with ∇^2 G(x, y) knowing that it has two effects: It smooth's the image (thus reducing noise), and its computes the Laplacian, which yields a double edge image. Locating edges then consists of finding the zero crossing between the double edges. The general calling syntax for LoG detector is

[g, t] = edge(f, 'log', T, sigma) (13)

Where sigma is the standard deviation and other parameter are explained previously. The default value of sigma is 2. As before, function edge ignores any edges that are not stronger than T. If T is not provided, or it is empty, [ ], edge chooses the value automatically. Setting T to 0 produces edges that are closed contours, a family characteristics of the LoG method.

VII. Zero –Crossing Detector

In detector is based on the same concept as the LoG method, but the convolution is carried out using a specified filter function, H. The calling syntax is

[g, t] = edge(f, 'zerocross', T, H) (14)

The other parameters are as explained for the LoG detector.

VIII. Canny Edge Detector

The Canny Detector (canny [1986]) is the most powerful edge detector in function edge. The method can be summarized as follows:

The image is smoothed using a Gaussian filter with a specified standard deviation ,σ,to reduce noise.

The local gradient ,$[g^2_x + g^2_y]^{1/2}$ and edge direction ,$\tan^{-1}(g_x / g_y)$, are computed at each point. Any of the first three techniques can be used to compute the derivatives. An edge point is defined to be a point whose strength is locally maximum in the direction of the gradient.

The edge points determined(2) give rise to ridges in the gradient magnitude image. The algorithm then tracks along the top of these ridges and sets to zero all pixels that are not actually on the ridge top so as to give a thin line in the output , a process known as nonmaximal suppression. The ridge pixels are then threshold by so-called hysteresis thresholding,which is based on using two thresholds, *T1* and *T2*, with *T1<T2* . Ridge pixels with values greater than *T2* are said to be "strong" edge pixels .Ridge pixels with values between *T1* and *T2* are said to be "weak" edge pixels.

Finally, the algorithm performs edge linking by incorporating the weak pixels that are 8-connected to the strong pixels.

The syntax for the Canny edge detector is

$$[g,\ t] = \text{edge}\ (f,\ \text{'canny'}\ ,\ T\ ,\ \text{sigma}) \qquad (15)$$

Where T is a vector, T = [T1, T2], containing the two thresholds explained in step 3 of the preceding procedure, and sigma is the standard deviation of the smoothing filter . If it is included in the output argument, it is a two-element vector containing the two threshold values used by the algorithm. The rest of the syntax is as explained for the other methods, including the automatic computation of thresholds if T is not supplied. The default value for sigma is 1.

IX. Experimental Result

In this section we take the famous "Lena" image as a input as shown in figure 1. Then we perform different edge detection operator with varying threshold values. The different output based on threshold values is shown in subsequent figures. The outputs of Sobel, Prewitt, Robert, LoG, Zerocrossing and Canny operators are shown in figure 2,3,4,5,6 and 7 respectively.

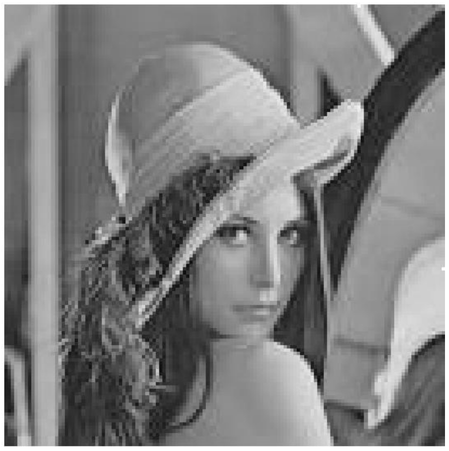

Figure 1: Original Lena Image

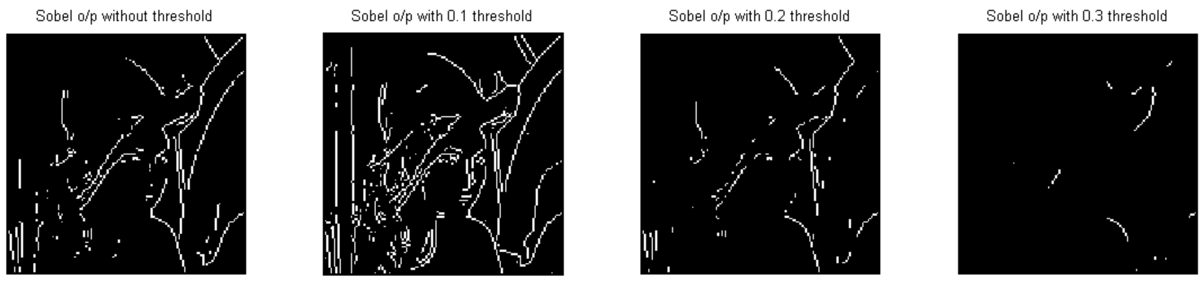
Sobel o/p without threshold
Sobel o/p with 0.1 threshold
Sobel o/p with 0.2 threshold
Sobel o/p with 0.3 threshold

Figure 2: Sobel Output

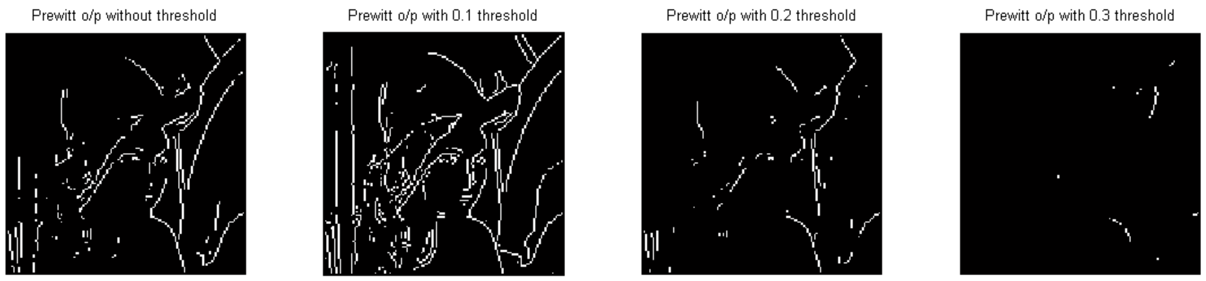
Prewitt o/p without threshold
Prewitt o/p with 0.1 threshold
Prewitt o/p with 0.2 threshold
Prewitt o/p with 0.3 threshold

Figure 3: Prewitt Output

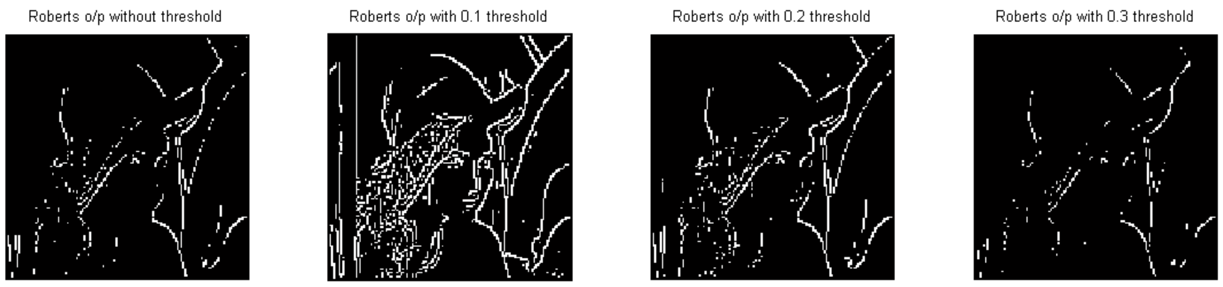
Roberts o/p without threshold
Roberts o/p with 0.1 threshold
Roberts o/p with 0.2 threshold
Roberts o/p with 0.3 threshold

Figure 4: Robert Output

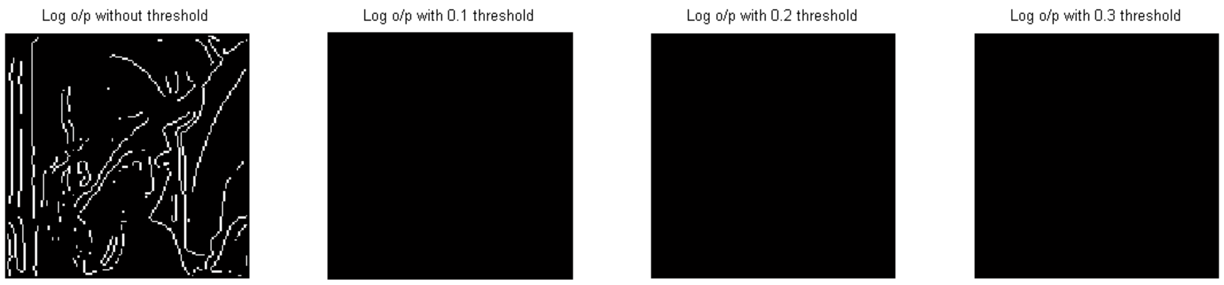
Log o/p without threshold
Log o/p with 0.1 threshold
Log o/p with 0.2 threshold
Log o/p with 0.3 threshold

Figure 5: LoG Output

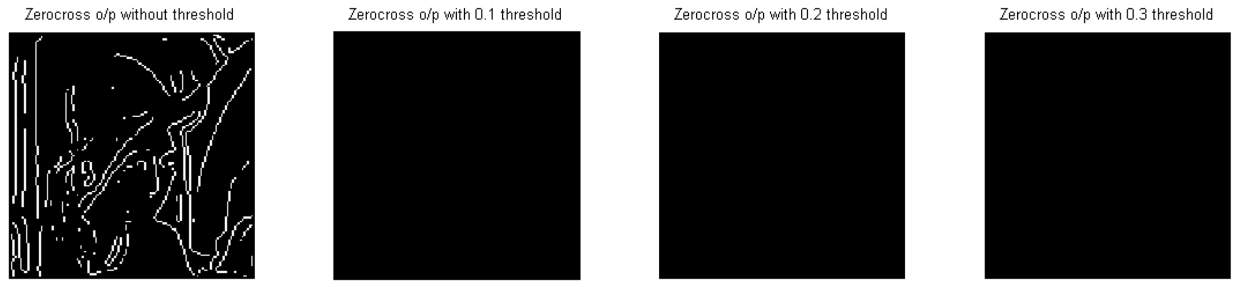

Figure 6: Zero-Crossing Output

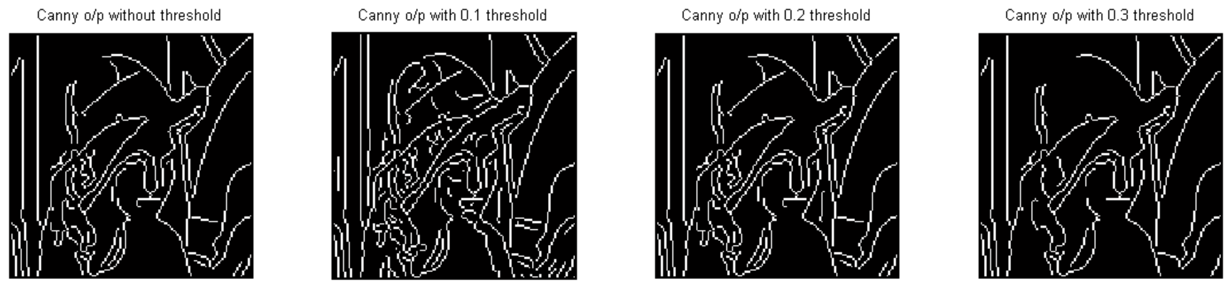

Figure 7: Canny Output

X. Conclusions

Linear features like edges and lines are important information for image analysis and computer vision. In our research, we have presented various techniques of edge detection. Major steps of a good edge detection algorithm are noise cleaning, computation of edge strength, edge linking or following and edge extraction. Properties of a good edge detector and method to evaluate them are described Prewitt operator can detect vertical edges better than that by Sobel operator; while Sobel is superior to Prewitt operator in detecting diagonal edges. Robert operator is sensitive to noise. Canny operator gives the better result also in higher threshold value. LoG and Zerocrossing operator give poor performances on higher threshold values.

References

[1]Ehsan Nadernejad, Sara Sharifza deh, Hamid Hassan pour Edge Detection Techniques: Evaluations and Comparisons Applied Mathematical Sciences, Vol. 2, 2008, no. 31, 1507 – 1520

[2]E. Argyle. “Techniques for edge detection,” Proc. IEEE, vol. 59, pp. 285-286, 1971

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

[4] J. Matthews. “An introduction to edge detection: The sobel edge detector,” Available at

http://www.generation5.org/content/2002/im01.asp, 2002.

[5] L. G. Roberts. "Machine perception of 3-D solids" ser. Optical and Electro-Optical Information Processing. MIT Press, 1965 .

[6] R. C. Gonzalez and R. E. Woods. "Digital Image Processing". 2nd ed. Prentice Hall, 2002.

[7] V. Torre and T. A. Poggio. "On edge detection". IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-8, no. 2, pp. 187-163, Mar. 1986.

[8] E. R. Davies. "Constraints on the design of template masks for edge detection". Partern Recognition Lett., vol. 4, pp. 11 1-120, Apr. 1986.

[9] M.B. Ahmad and T.S. Choi , Local Threshold and Boolean Function Based Edge Detection, IEEE Transactions on Consumer Electronics, Vol. 45, No 3. August 1999

[10]T. Lindeberg (1998) "Edge detection and ridge detection with automatic scale selection", International Journal of Computer Vision, 30, 2, pages 117--154.

[11] W. Zhang and F. Bergholm (1997) "Multi-scale blur estimation and edge type classification for scene analysis", International Journal of Computer Vision, vol 24, issue 3, Pages: 219 - 250.