Journal of Information and Operations Management
ISSN: 0976–7754 & E-ISSN: 0976–7762 , Volume 3, Issue 1, 2012, pp-103-106
Available online at http://www.bioinfo.in/contents.php?id=55

COMPARISON OF VARIOUS EDGE DETECTION TECHNIQUES

BINDU BANSAL[1]*, JASBIR SINGH SAINI[1], VIPAN BANSAL[2], AND GURJIT KAUR[1]

[1]GNDEC, Ludhiana, India
[2]LPU, Phagwara, India
*Corresponding Author: Email- er_bindu18@rediffmail.com

Received: December 12, 2011; Accepted: January 15, 2012

Abstract- 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. Edge detection refers to the process of identifying and locating sharp discontinuities in an image. The discontinuities are abrupt changes in pixel intensity which characterize boundaries of objects in a scene. The reason for this is that edges form the outline of an object. 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. In this paper, we have compared several techniques for edge detection in image processing.
Keywords- Edge detection, Prewitts, Roberts, LoG, Canny.

Citation: Bindu Bansal, et al. (2012) Comparison of Various Edge Detection Techniques. Journal of Information and Operations Management ISSN: 0976–7754 & E-ISSN: 0976–7762, Volume 3, Issue 1, pp– 103-106.

Introduction

Edge detection is a very important area in the field of Computer Vision. Edges define the boundaries between regions in an image, which helps with segmentation and object recognition. They can show where shadows fall in an image or any other distinct change in the intensity of an image. Edge detection is a fundamental of low -level image processing and good edges are necessary for higher level processing. [1]

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

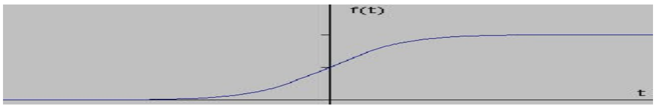
- Edge orientation: The geometry of the operator determines a characteristic direction in which it is most sensitive to edges. Operators can be optimized to look for horizontal, vertical, or diagonal edges.
- Noise environment: Edge detection is difficult in noisy images, since both the noise and the edges contain high-frequency content. Attempts to reduce the noise result in blurred and distorted edges. Operators used on noisy images are typically larger in scope, so they can average enough data to discount localized noisy pixels. This results in less accurate localization of the detected edges.

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. Newer wavelet-based techniques actually characterize the nature of the transition for each edge in order to distinguish, for example, edges associated with hair from edges associated with a face.

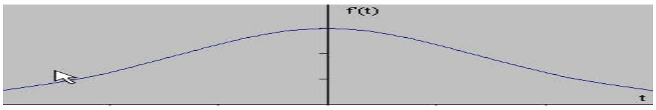
There are many ways to perform edge detection. However, the majority of different methods may be grouped into two categories:

- Gradient: The gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image.
- 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:

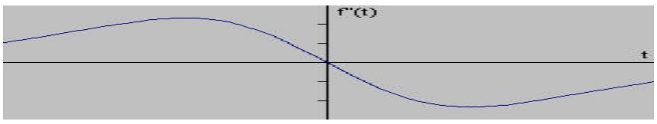
Suppose we have the following signal, with an edge shown by the jump in intensity below:



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:



Clearly, 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. 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:



Edge Detection Techniques

Robert’s cross Operator

The Roberts Cross operator performs a simple, quick to compute, 2-D spatial gradient measurement on an image. Pixel values at each point in the output represent the estimated absolute magnitude of the spatial gradient of the input image at that point.

The operator consists of a pair of 2×2 convolution kernels as shown in Figure. One kernel is simply the other rotated by 90°. This is very similar to the Sobel operator.

+1	0
0	-1

Gx

0	+1
-1	0

Gy

Fig.2- Roberts operator

These kernels are designed to respond maximally to edges running at 45° to the pixel grid, one kernel for each of the two perpendicular orientations. The kernels can be applied separately to the input image, to produce separate measurements of the gradient component in each orientation (call these *Gx* and *Gy*). These can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient. The gradient magnitude is given by:

$$|G| = \sqrt{Gx^2 + Gy^2}$$

although typically, an approximate magnitude is computed using:

$$|G| = |Gx| + |Gy|$$

which is much faster to compute.

The angle of orientation of the edge giving rise to the spatial gradient (relative to the pixel grid orientation) is given by:

$$\theta = \arctan(Gy/Gx) - 3\pi/4$$

Prewitt’s Operator

Prewitt operator is similar to the Sobel operator and is used for detecting vertical and horizontal edges in images.

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

$$h_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad h_3 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

filter in order to reduce its sensitivity to noise. The operator normally takes a single graylevel image as input and produces another graylevel image as output.

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

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

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.

Canny’s Edge Detection Algorithm

The Canny edge detection algorithm is known to many as the optimal edge detector. Canny's intentions were to enhance the many edge detectors already out at the time he started his work. He was very successful in achieving his goal and his ideas and methods can be found in his paper, "*A Computational Approach to Edge Detection*". In his paper, he followed a list of criteria to improve current methods 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 responses 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. This was implemented because the first 2 were not substantial enough to completely eliminate the possibility of multiple responses to an edge.

Based on these criteria, the canny edge detector first smoothes the image to eliminate and noise. It then finds the image gradient to highlight regions with high spatial derivatives. The algorithm then tracks along these regions and suppresses any pixel that is not at the maximum (non maximum suppression). The gradient array is now further reduced by hysteresis. Hysteresis is used to track along the remaining pixels that have not been suppressed. Hysteresis uses two thresholds and if the magnitude is below the first threshold, it is set to zero (made a non edge). If the magnitude is above the high threshold, it is made an edge. And if the magnitude is between the 2 thresholds, then it is set to zero unless there is a path from this pixel to a pixel with a gradient above T2.

Step 1

In order to implement the canny edge detector algorithm, a series of steps must be followed. The first step is to filter out any noise in the original image before trying to locate and detect any edges. And because the Gaussian filter can be computed using a simple mask, it is used exclusively in the Canny algorithm. Once a suitable mask has been calculated, the Gaussian smoothing can be performed using standard convolution methods. A convolution mask is usually much smaller than the actual image. As a result, the mask is slid over the image, manipulating a square of pixels at a time. The larger the width of the Gaussian mask, the lower is the detector's sensitivity to noise. The localization error in the detected edges also increases slightly as the Gaussian width is increased. The Gaussian mask used in my implementation is shown below.

$\frac{1}{115}$

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

Figure 3 Discrete approximation to Gaussian function with $\sigma=1.4$

Step 2

After smoothing the image and eliminating the noise, the next step is to find the edge strength by taking the gradient of the image. The Sobel operator performs a 2-D spatial gradient measurement on an image. Then, the approximate absolute gradient magnitude (edge strength) at each point can be found. The Sobel operator uses a pair of 3x3 convolution masks, one estimating the gradient in the x-direction (columns) and the other estimating the gradient in the y-direction (rows). They are shown below:

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

Gx

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

Gy

The magnitude, or edge strength, of the gradient is then approximated using the formula: $|G| = |Gx| + |Gy|$

Step 3

The direction of the edge is computed using the gradient in the x and y directions. However, an error will be generated when sumX is equal to zero. So in the code there has to be a restriction set whenever this takes place. Whenever the gradient in the x direction is equal to zero, the edge direction has to be equal to 90 degrees or 0 degrees, depending on what the value of the gradient in the y-direction is equal to. If GY has a value of zero, the edge direction will equal 0 degrees. Otherwise the edge direction will equal 90 degrees. The formula for finding the edge direction is just:

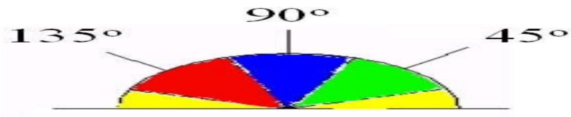
Theta = invtan (Gy / Gx)

Step 4

Once the edge direction is known, the next step is to relate the edge direction to a direction that can be traced in an image. So if the pixels of a 5x5 image are aligned as follows:

x	x	x	x	x
x	x	x	x	x
x	x	a	x	x
x	x	x	x	x
x	x	x	x	x

Then, it can be seen by looking at pixel "a", there are only four possible directions when describing the surrounding pixels - 0 degrees (in the horizontal direction), 45 degrees (along the positive diagonal), 90 degrees (in the vertical direction), or 135 degrees (along the negative diagonal). So now the edge orientation has to be resolved into one of these four directions depending on which direction it is closest to (e.g. if the orientation angle is found to be 3 degrees, make it zero degrees). Think of this as taking a semicircle and dividing it into 5 regions.



Therefore, any edge direction falling within the yellow range (0 to 22.5 & 157.5 to 180 degrees) is set to 0 degrees. Any edge direction falling in the green range (22.5 to 67.5 degrees) is set to 45 degrees. Any edge direction falling in the blue range (67.5 to 112.5 degrees) is set to 90 degrees. And finally, any edge direction falling within the red range (112.5 to 157.5 degrees) is set to 135 degrees.

Step 5

After the edge directions are known, non maximum suppression now has to be applied. Non maximum suppression is used to trace along the edge in the edge direction and suppress any pixel value (sets it equal to 0) that is not considered to be an edge. This will give a thin line in the output image.

Step 6

Finally, hysteresis is used as a means of eliminating streaking. Streaking is the breaking up of an edge contour caused by the operator output fluctuating above and below the threshold. If a single threshold, T1 is applied to an image, and an edge has an average strength equal to T1, then due to noise, there will be instances where the edge dips below the threshold. Equally it will also extend above the threshold making an edge look like a dashed line.

Tool Used

The methods have been developed in Matlab 7.0.1 GUI. The graphical interface allows a better representation of the image with its output image. Visual appearance of the image has been used as a deciding factor for the finding the best method of edge detection.

Results and Disscussions

Welcome screen for the project is

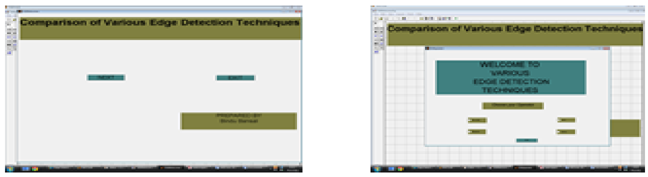
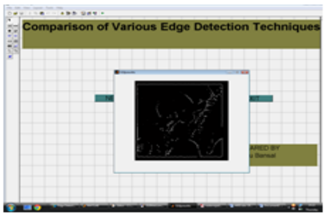
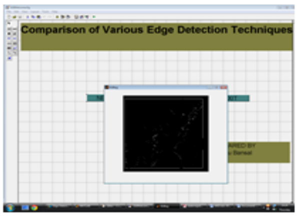


Fig.6- Welcome screen for "COMPARISON OF VARIOUS EDGE DETECTION TECHNIQUES"

Clicking on the *Next* we will be forwarded to next screen of the algorithms used.Choosing the technique we will go to the next screen for selecting input image.

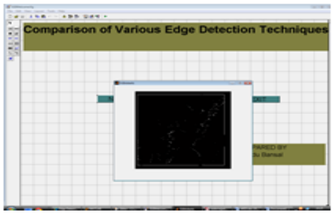
Putting the image on the coordinates looks like.Choosing the image we will apply a particular method on it. The results for the four techniques are as follows.
Result for Prewitts Operator.
Result for LoG Operator
Result for Roberts Operator
Result for Canny's Edge Detection

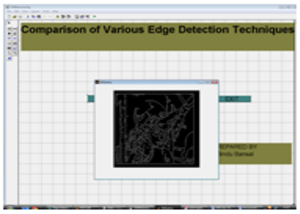




Concusion

Gradient-based algorithms such as the Prewitt filter have a major





drawback of being very sensitive to noise. The size of the kernel filter and coefficients are fixed and cannot be adapted to a given image. An adaptive edge-detection algorithm is necessary to provide a robust solution that is adaptable to the varying noise levels. Gradient-based algorithms such as the Prewitt filter have a major drawback of being very sensitive to noise. The size of the kernel filter and coefficients are fixed and cannot be adapted to a given image. An adaptive edge-detection algorithm is necessary to provide a robust solution that is adaptable to the varying noise levels of these images to help distinguish valid image contents from visual artifacts introduced by noise.

The performance of the Canny algorithm depends heavily on the adjustable parameters, σ, which is the standard deviation for the Gaussian filter, and the threshold values, 'T1' and 'T2'. σ also controls the size of the Gaussian filter. The bigger the value for σ, the larger the size of the Gaussian filter becomes. This implies more blurring, necessary for noisy images, as well as detecting larger edges. As expected, however, the larger the scale of the Gaussian, the less accurate is the localization of the edge. Smaller values of σ imply a smaller Gaussian filter which limits the amount of blurring, maintaining finer edges in the image. The user can tailor the algorithm by adjusting these parameters to adapt to different environments.Canny's edge detection algorithm is computationally more expensive compared to Sobel, Prewitt and Robert's operator. However, the Canny's edge detection algorithm performs better than all these operators under almost all scenarios.

References

[1] Argyle E (1971) *IEEE*, vol. 59, pp. 285-286.
[2] Bergholm F (1986) *8th Int. Conf. Pattern Recognition, Paris, France*, pp. 597- 600.
[3] Matthews J (2002) *An introduction to edge detection: The sobel edge detector*.
[4] Roberts L.G (1965) *ser. Optical and Electro-Optical Information Processing*. MIT Press.
[5] Gonzalez R.C. and Woods R.E (2002) *2nd ed. Prentice Hall.*
[6] Torre V and Poggio T.A (1986) *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, no. 2, pp. 187-163.
[7] Davies E.R (1986) *Partern Recognition Lett*.,vol. 4, pp. 11 1-120.
[8] Frei W and Chen C.C (1977) *IEEE Trans. Comput*., vol. C-26, no. 10, pp. 988-998.
[9] Grimson W.E and Hildreth E.C (1985) *IEEE Trans. Pattern Anal. Machine Intell.,* vol. PAMI-7, no. 1, pp. 121-129.
[10]Haralick R.M (1984) *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, no. 1, pp. 58-68.
[11]Canny J.F (1986) *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, no. 6, pp. 679-697.
[12]Canny J (1983) *Master's thesis,* MIT.
[13]Kirsch R.A (1971) *Comput. Eiorned*. Res., vol. 4, pp. 315-328.
[14]Hueckel M.H (1973) *J. ACM*, vol. 20, no. 4, pp. 634- 647.
[15]Yakimovsky Y (1976) *JACM,* vol. 23, no. 4, pp. 598-619.