Study and Comparison of Various Image Edge Detection Techniques

Raman Maini research_raman@yahoo.com
Reader
Punjabi University
Patiala-147002(Punjab), India

Dr. Himanshu Aggarwal himagrawal@rediffmail.com
Reader,
Punjabi University
Patiala-147002(Punjab), India

ABSTRACT

Edges characterize boundaries and are therefore a problem of fundamental importance in image processing. Image Edge detection significantly reduces the amount of data and filters out useless information, while preserving the important structural properties in an image. Since edge detection is in the forefront of image processing for object detection, it is crucial to have a good understanding of edge detection algorithms. In this paper the comparative analysis of various Image Edge Detection techniques is presented. The software is developed using MATLAB 7.0. It has been shown that the Canny's edge detection algorithm performs better than all these operators under almost all scenarios. Evaluation of the images showed that under noisy conditions Canny, LoG(Laplacian of Gaussian), Robert, Prewitt, Sobel exhibit better performance, respectively. 1. It has been observed that Canny's edge detection algorithm is computationally more expensive compared to LoG(Laplacian of Gaussian), Sobel, Prewitt and Robert's operator.

Keywords: Edge Detection, Noise, Digital Image Processing

1. INTRODUCTION

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, Noise environment and Edge structure. 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. 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

1 Raman Maini is corresponding author with e-mail address research_raman@yahoo.com , Mobile no +91-9779020951 and address as Reader, University College of Engineering, Punjabi University, Patiala-147002 (India), Fax No. +91-1753046324

results in less accurate localization of the detected edges. 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 [1]. The operator needs to be chosen to be responsive to such a gradual change in those cases. So, there are problems of false edge detection, missing true edges, edge localization, high computational time and problems due to noise etc. Therefore, the objective is to do the comparison of various edge detection techniques and analyze the performance of the various techniques in different conditions.

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

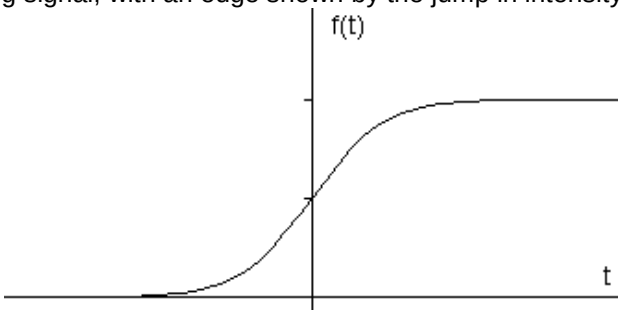
Gradient based Edge Detection:

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

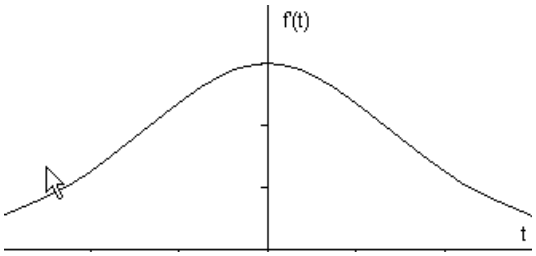
Laplacian based Edge Detection:

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:

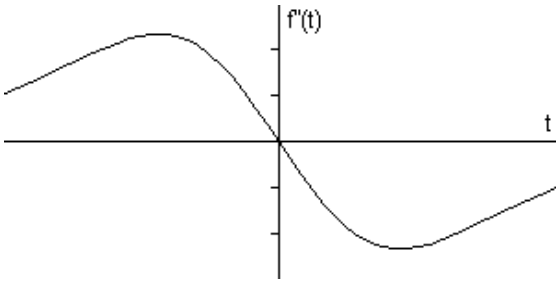
Suppose we have the following signal, with an edge shown by the jump in intensity below:



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 [2]. 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:



In this paper we analyzed and did the visual comparison of the most commonly used Gradient and Laplacian based Edge Detection techniques. In section 2 the problem definition is presented. In section 3 the various edge detection techniques have been studied and analyzed. In section 4 the visual comparisons of various edge detection techniques have been done by developing software in MATLAB 7.0. Section 5 discusses the advantages and disadvantages of various edge detection techniques. Section 6 discusses the conclusion reached by analysis and visual comparison of various edge detection techniques developed using MATLAB 7.0.

2. PROBLEM DEFINITION

There are problems of false edge detection, missing true edges, producing thin or thick lines and problems due to noise etc. In this paper we analyzed and did the visual comparison of the most commonly used Gradient and Laplacian based Edge Detection techniques for problems of inaccurate edge detection, missing true edges, producing thin or thick lines and problems due to noise etc. The software is developed using MATLAB 7.0

3. Edge Detection Techniques

3.1 Sobel Operator

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°.

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

Gx

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

Gy

FIGURE 1: Masks used by Sobel Operator

These kernels are designed to respond maximally to edges running vertically and horizontally relative 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 [3]. The gradient magnitude is given by:

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

Typically, an approximate magnitude is computed using:

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

which is much faster to compute.

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

$$\theta = \arctan(Gy / Gx)$$

3.2 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 2. One kernel is simply the other rotated by 90°[4]. This is very similar to the Sobel operator.

+1	0
0	-1

Gx

0	+1
-1	0

Gy

FIGURE 2: Masks used for Robert 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:

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

3.3 Prewitt's operator:

Prewitt operator [5] is similar to the Sobel operator and is used for detecting vertical and horizontal edges in images.

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

Gx

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

Gy

FIGURE 3: Masks for the Prewitt gradient edge detector

3.4 Laplacian of Gaussian:

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:

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

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[5]. Three commonly used small kernels are shown in Figure 4.

1	1	1
1	-8	1
1	1	1

-1	2	-1
2	-4	2
-1	2	-1

FIGURE 4. Three commonly used discrete approximations to the Laplacian filter.

Because these kernels are approximating a second derivative measurement on the image, they are very sensitive to noise. To counter this, the image is often Gaussian Smoothed before applying the Laplacian filter. This pre-processing step reduces the high frequency noise components prior to the differentiation step.

In fact, since the convolution operation is associative, we can convolve the Gaussian smoothing filter with the Laplacian filter first of all, and then convolve this hybrid filter with the image to achieve the required result. Doing things this way has two advantages: Since both the Gaussian and the Laplacian kernels are usually much smaller than the image, this method usually requires far fewer arithmetic operations.

The LoG (`Laplacian of Gaussian')[6] kernel can be pre-calculated in advance so only one convolution needs to be performed at run-time on the image.

The 2-D LoG function [7] centered on zero and with Gaussian standard deviation σ has the form:

$$\mathrm{LoG}(x,y) = -1/\pi\sigma^4 \left[\, 1 - \left(\frac{x^2+y^2}{2\sigma^2}\right)\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

and is shown

0	1	0
1	-4	1
0	1	0

in Figure 5.

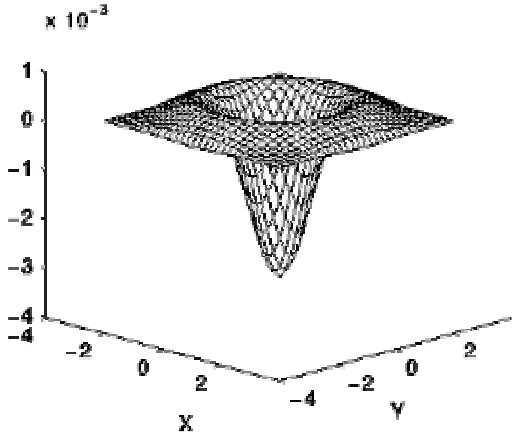


FIGURE 5. The 2-D Laplacian of Gaussian (LoG) function. The *x* and *y* axes are marked in standard deviations (σ).

A discrete kernel that approximates this function (for a Gaussian σ = 1.4) is shown in Figure 6.

0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

FIGURE 6. Discrete approximation to LoG function with Gaussian σ = 1.4.

Note that as the Gaussian is made increasingly narrow, the LoG kernel becomes the same as the simple Laplacian kernels shown in figure 4. This is because smoothing with a very narrow Gaussian (σ < 0.5 pixels) on a discrete grid has no effect. Hence on a discrete grid, the simple Laplacian can be seen as a limiting case of the LoG for narrow Gaussians [8]-[10].

3.5 Canny 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"[11]. 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 two 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 (nonmaximum 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.

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 [3] 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

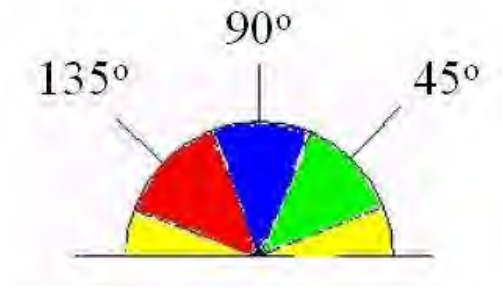
The magnitude, or edge strength, of the gradient is then approximated using the formula:
|G| = |Gx| + |Gy|

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[12] 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. To avoid this, hysteresis uses 2 thresholds, a high and a low. Any pixel in the image that has a value greater than T1 is presumed to be an edge pixel, and is marked as such immediately. Then, any pixels that are connected to this edge pixel and that have a value greater than T2 are also selected as edge pixels. If you think of following an edge, you need a gradient of T2 to start but you don't stop till you hit a gradient below T1.

4. Visual Comparison of various edge detection Algorithms

FIGURE 7: Image used for edge detection analysis (wheel.gif)

Edge detection of all four types was performed on Figure 7[13]. Canny yielded the best results. This was expected as Canny edge detection accounts for regions in an image. Canny yields thin lines for its edges by using non-maximal suppression. Canny also utilizes hysteresis with thresholding.

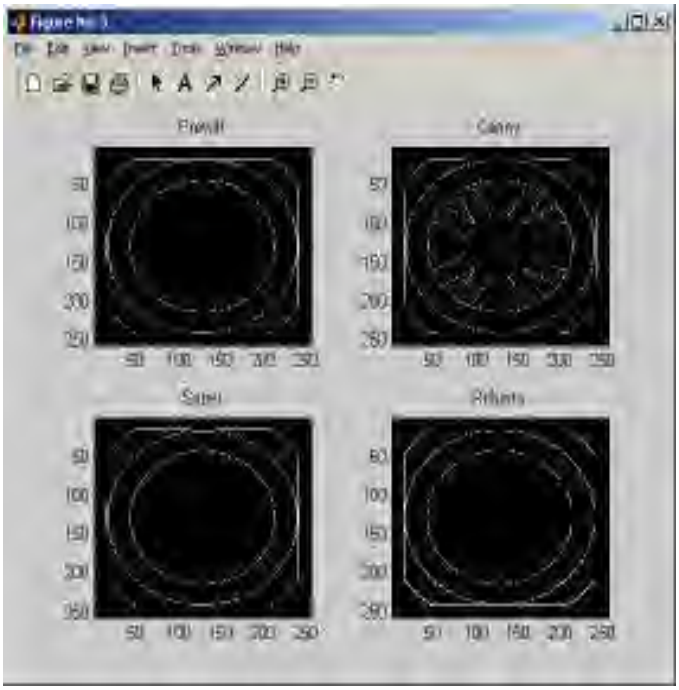
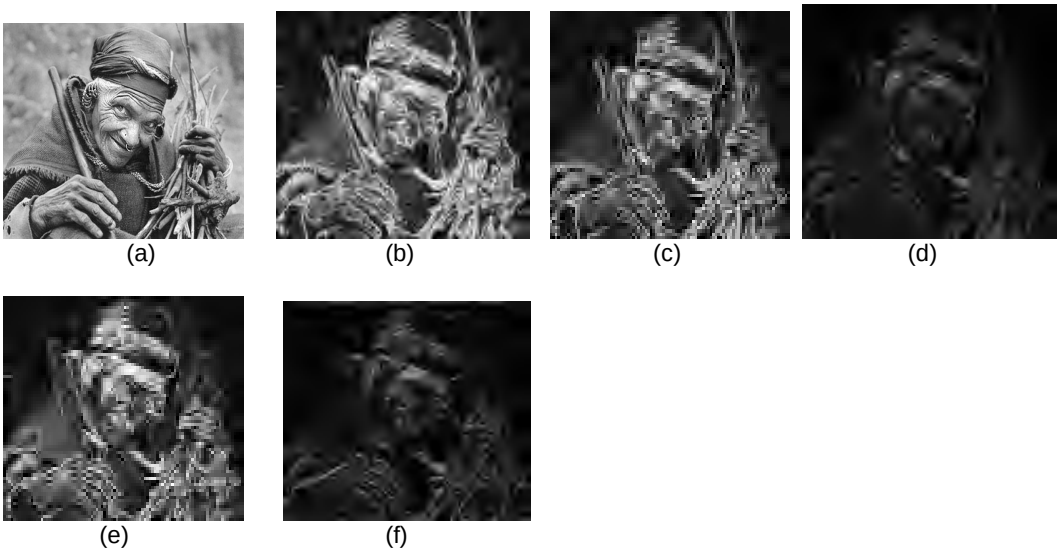


FIGURE 8: Results of edge detection on Figure 7. Canny had the best results



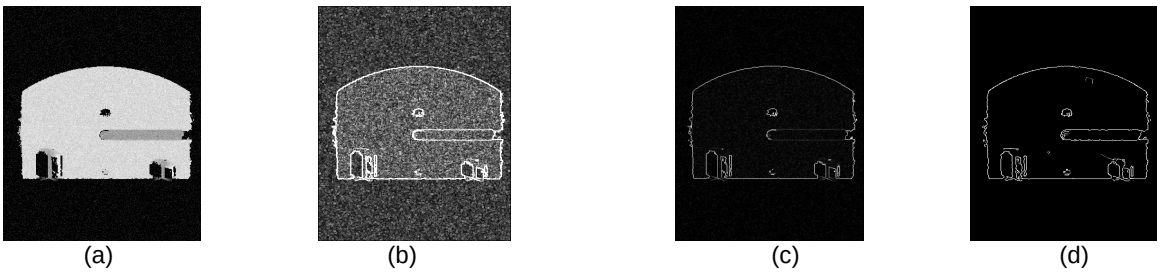
(a) (b) (c) (d)

(e) (f)

FIGURE 9: Comparison of Edge Detection Techniques Original Image (b) Sobel (c) Prewitt (d) Robert (e) Laplacian (f) Laplacian of Gaussian

(a) (b) (c) (d) (e)

FIGURE 10: Comparison of Edge Detection Techniques on Lena Image Original Image (b) Canny Method (c) Roberts Edges (d) LOG edges (e) Sobel



(a) (b) (c) (d)

FIGURE 11: Comparison of Edge Detection technique on Noisy Image (a) Original Image with Noise (b) Sobel (c) Robert (d) Canny

5. Advantages and Disadvantages of Edge Detector

As edge detection is a fundamental step in computer vision, it is necessary to point out the true edges to get the best results from the matching process. That is why it is important to choose edge detectors that fit best to the

application. In this respect, we first present some advantages and disadvantages of Edge Detection Techniques [13]-[21] with in the context of our classification in Table 1.

Operator	Advantages	Disadvantages
Classical (Sobel, prewitt, Kirsch,…)	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) (Marr-Hildreth)	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, Shen-Castan)	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

Table 1: Some Advantages and Disadvantages of Edge Detectors

6. CONCLUSIONS

Since edge detection is the initial step in object recognition, it is important to know the differences between edge detection techniques. In this paper we studied the most commonly used edge detection techniques of Gradient-based and Laplacian based Edge Detection. The software is developed using MATLAB 7.0.

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. Evaluation of the images showed that under noisy conditions, Canny, LoG, Sobel, Prewitt, Roberts's exhibit better performance, respectively.

• Acknowledgement

The authors are greatly indebted to the Department of Computer Engineering, University College of Engineering, Punjabi University, Patiala for providing excellent lab facilities that make this work possible.

• References

1. E. Argyle. "Techniques for edge detection," Proc. IEEE, vol. 59, pp. 285-286, 1971

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

3. J. Matthews. "An introduction to edge detection: The sobel edge detector," Available at http://www.generation5.org/content/2002/im01.asp, 2002.

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

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

6. 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.

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

8. W. Frei and C.-C. Chen. "Fast boundary detection: A generalization and a new algorithm ". lEEE Trans. Comput., vol. C-26, no. 10, pp. 988-998, 1977.

9. W. E. Grimson and E. C. Hildreth. "Comments on Digital step edges from zero crossings of second directional
derivatives''. IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-7, no. 1, pp. 121-129, 1985.

10. 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.

11. J. F. Canny. "A computational approach to edge detection". IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-8, no. 6, pp. 679-697, 1986

12 J. Canny. "Finding edges and lines in image". Master's thesis, MIT, 1983.

13. R. A. Kirsch. "Computer determination of the constituent structure of biomedical images". Comput. Eiorned. Res., vol. 4, pp. 315-328, 1971.

14. M. H. Hueckel. " A local visual operator which recognizes edges and line". J. ACM, vol. 20, no. 4, pp. 634-647, Oct. 1973.

15. Y. Yakimovsky, "Boundary and object detection in real world images". JACM, vol. 23, no. 4, pp. 598-619, Oct. 1976

16. A. Yuille and T. A. Poggio . “Scaling theorems for zero crossings”. IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-8, no. 1, pp. 187-163, Jan. 1986.

17. D. Marr and E.Hildreth. “Theory of Edge Detection”. Proceedings of the Royal Society of London. Series B, Biological Sciences,, Vol. 207, No. 1167. (29 February 1980), pp. 187-217

18. M. Heath, S. Sarkar, T. Sanocki, and K.W. Bowyer. “A Robust Visual Method for Assessing the Relative. Performance of Edge Detection Algorithms”. IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 19, no. 12, pp. 1338-1359, Dec. 1997

19. M. Heath, S. Sarkar, T. Sanocki, and K.W. Bowyer. “Comparison of Edge Detectors: A Methodology and Initial Study “. Computer Vision and Image Understanding, vol. 69, no. 1, pp. 38-54 Jan. 1998

20. M.C. Shin, D. Goldgof, and K.W. Bowyer .“Comparison of Edge Detector Performance through Use in an Object Recognition Task”. Computer Vision and Image Understanding, vol. 84, no. 1, pp. 160-178, Oct. 2001.

21. T. Peli and D. Malah. “A Study of Edge Detection Algorithms”. Computer Graphics and Image Processing, vol. 20, pp. 1-21, 1982.