International Journal of Computer Theory and Engineering, Vol. 1, No. 5, December, 2009
1793-8201

Performance Evaluation of Edge Detection Techniques for Images in Spatial Domain

Mamta Juneja , Parvinder Singh Sandhu

***Abstract*—Edges characterize boundaries and are therefore considered for prime importance in image processing. Edge detection filters out useless data, noise and frequencies 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 methods. In this paper the comparative analysis of various Image Edge Detection methods is presented. The evidence for the best detector type is judged by studying the edge maps relative to each other through statistical evaluation. Upon this evaluation, an edge detection method can be employed to characterize edges to represent the image for further analysis and implementation. It has been shown that the Canny's edge detection algorithm performs better than all these operators under almost all scenarios.**

***Index Terms*—About four key words or phrases in alphabetical order, separated by commas.**

I. Introduction

Edges are boundaries between different textures. Edge also can be defined as discontinuities in image intensity from one pixel to another. The edges for an image are always the important characteristics that offer an indication for a higher frequency. Detection of edges for an image may help for image segmentation, data compression, and also help for well matching, such as image reconstruction and so on[3]. Variables involved in the selection of an edge detection operator include Edge orientation, Noise environment and Edge structure[1]. 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[2]. 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. 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 [4].

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

A. Theoretical Concepts:

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

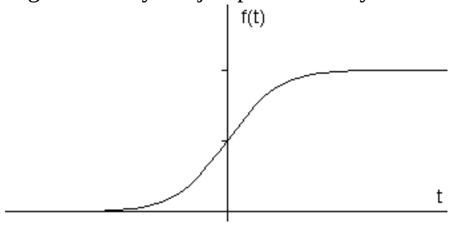
First Order Derivative Based Edge Detection (Gradient method): It detects the edges by looking for the maximum and minimum in the first derivative of the image. Sharpening an image results in the detection of fine details as well as enhancing blurred ones. The magnitude of the gradient is the most powerful technique that forms the basis for various approaches to sharpening. The gradient vector points in the direction of maximum rate of change. For a function fix, y), the magnitude of the gradient of f at coordinates (x, y) is defined as

$$|\nabla f(x,y)| = \sqrt{(\partial_x f(x,y))^2 + (\partial_y f(x,y))^2} \quad (1)$$

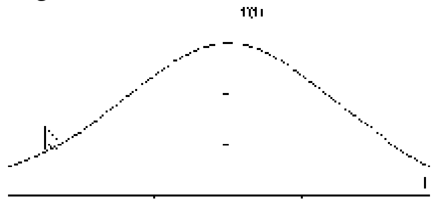
while the gradient orientation is given by

$$\angle \nabla f(x,y) = ArcTan\left(\partial_y f(x,y) / \partial_x f(x,y)\right) \quad (2)$$

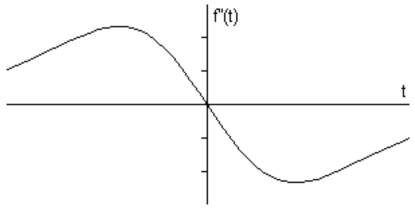
Second Order Derivative Based Edge Detection (Laplacian based Edge Detection): The Laplacian method searches for zero crossings in the second derivative of the image to find edges. An edge has the one-dimensional shape of a ramp and calculating the derivative of the image can highlight its location.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



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 [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:



This approach uses the zero-crossing operator which acts by locating zeros of the second derivatives of f(x, y). The differential operator is used in the so-called zero-crossing edge detectors.

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \qquad (3)$$

Thresholding allocates a range of pixel values to each object of interest. It works best with grayscale images that utilize the whole range of the grayscale. For the image f(x, y), the threshold image g(x, y) is defined as

$$g(x,y) = \begin{cases} 1 & if \quad f(x,y) > T \\ 0 & if \quad f(x,y) \le T \end{cases} \qquad (4)$$

Where T is the threshold value.

Convolution operates on images of different sizes but of the same dimensionality. For an image of M rows and JV columns, and a kernel of m rows and n columns, the convolved image will have M - m + 1 rows, and N - n + 1 columns, and the image is given by

$$O(i,j) = \sum_{k=1}^{m} \sum_{l=1}^{n} I(i+k-1, j+l-1) K(k,l) (5)$$

Where i runs from 1 to M - m + 1 and j runs from 1 to JV - 11 + 1. The methodology implements many different operators, particularly spatial filters and feature detectors.

1) Sobel Operator

Sobel filter is a simple approximation to the concept of gradient with smoothing. The 3x3 convolution mask is usually used to detect gradients in X and Y directions.

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	+2	+1
0	0	0
-1	-2	-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 [6]. 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:

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

2) Prewitt's operator:

The Prewitt filter[7] is very similar to Sobel filter. The 3x3 total convolution mask is used to detect gradient in the X, Y directions as shown in Figure 2. Prewitt filter is a fast method for edge detection. The difference with respect to Sobel filter is the spectral response. It is only suitable for well-contrasted noiseless images.

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

Gx

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

Gy

Figure 2: Masks for the Prewitt gradient edge detector

3) Robert's cross operator:

The Roberts Cross operator performs a simple, quick to compute, 2-D spatial gradient measurement on an image. It thus highlights regions of high spatial frequency which often correspond to edges. In its most common usage, the input to the operator is a grayscale image, as is the output. Pixel values at each point in the output represent the estimated absolute magnitude of the spatial gradient of the input image at that point.It uses 22 convolution masks. The operator consists of a pair of 2×2 convolution kernels as shown in Figure 3. 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

Figure 3: 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: $q = \arctan(Gy/Gx) - 3p/4$

4) Laplacian of Gaussian:

Laplacian of a Gaussian[8] function is referred to as LoG. The filtering process can be seen as the application of a smoothing Filter, followed by a derivative operation. The smoothing is performed by a convolution with a Gaussian function. Usually a truncated Gaussian function is used when the convolution is calculated directly. The derivatives applied to a smoothed function can be obtained by applying a convolution with the derivative of the convolution mask. One of the interesting characteristics of Gaussian is its circular symmetry which is coherent with the implicit anisotropy of physical data 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[7]. The commonly used small kernels are shown in Figure 4.

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

Gx

1	1	1
1	-8	1
1	1	1

Gy

Figure 4. 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') 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 centered on zero and with Gaussian standard deviation s has the form: $LoG(x,y) = -1/ps^4\left[1 - \left(\frac{x^2+y^2}{2s^2}\right)\right] e^{-\frac{x^2+y^2}{2s^2}}$

and is shown in Figure 5.

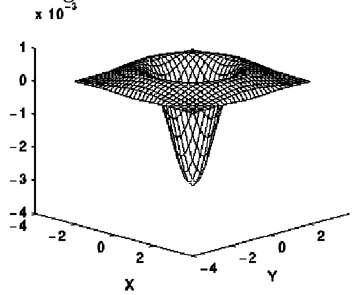


Figure 5. The 2-D Laplacian of Gaussian (LoG) function. The *x* and *y* axes are marked in standard deviations (s).

A discrete kernel that approximates this function (for a Gaussian s = 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 s = 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 (s < 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 [9]-[11].

5) *Canny Edge Detection Algorithm*

Canny operator is the result of solving an optimization problem with constraints. The criteria are sensibility, localization and local unicity. The method can be seen as a smoothing filtering performed with a linear combination of exponential functions, followed by derivative operations. 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*"[12]. 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.

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

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 (no 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.

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 [6] 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 Gy

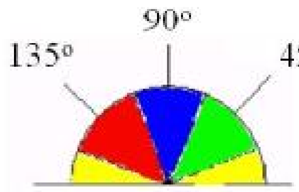
The magnitude, or edge strength, of the gradient is then approximated using the formula:
|G| = |Gx| + |Gy|

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[13] 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.

II. EXPERIMENTAL ANALYSIS

Edges are detected using the Sobel, Prewitt, and Roberts methods, by thresholding the gradient function. For the Laplacian of Gaussian method, thresholding is computed for the slope of the zero crossings after filtering the image with the LoG filter. For the Canny method, a threshold is applied to the gradient using the derivative of a Gaussian filter.

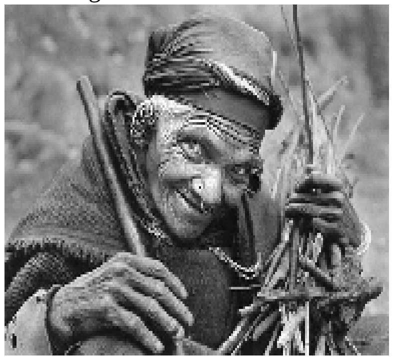


Figure 7

A. *Detection using Sobel filter*

As mentioned before, the Sobel method finds edges using the Sobel approximation to the derivative. It returns edges at those points where the gradient of the image is maximum. Figure 8 displays the results of applying the Sobel method to the image of Figure 7.

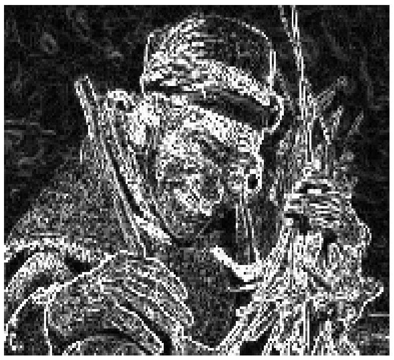


Figure 8: Sobel edge map of Figure 7

B. *Detection using Prewitt filter*

The Prewitt method finds edges using the Prewitt approximation to the derivative. It returns edges at those points where the gradient of the image is maximum. Results of applying this filter to Figure 7 are displayed in Figure 9.

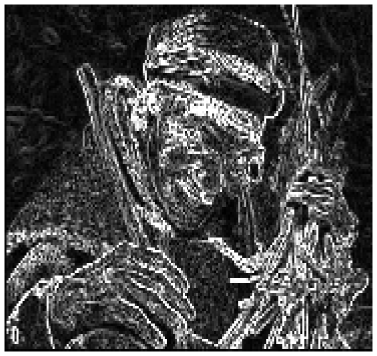


Figure 9: Prewitt edge map of Figure 7

C. *Detection using Roberts*

The Roberts method finds edges using the Roberts approximation to the derivative. It returns edges at those points where the gradient of the image is maximum. Results of applying this filter to Figure 7 are displayed in Figure 10.

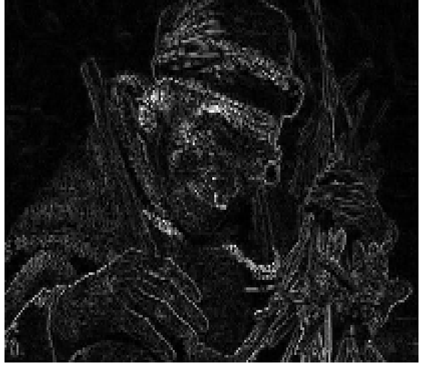


Figure 10: Roberts edge map of Figure 7

D. Detection using Laplacian of Gaussian

The Laplacian of Gaussian method finds edges by looking for zero crossings after filtering the image with the Laplacian of Gaussian filter. The edge map is shown in Figure 11.

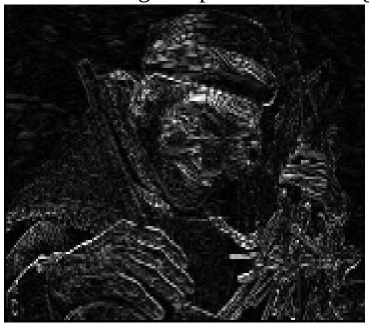


Figure 11: Laplacian of Gaussian edge map of Figure 7

E. Detection using Canny

The Canny method finds edges by looking for local maxima of the gradient of the image. The gradient is calculated using the derivative of the Gaussian filter. The method uses two thresholds to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. This method is therefore less likely than the others to be "fooled" by noise, and more likely to detect true weak edges. Figure 12 illustrates these points which are the result of applying this method to the image of Figure 7.

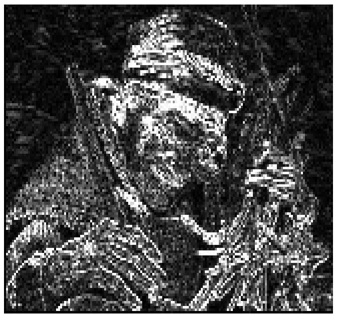
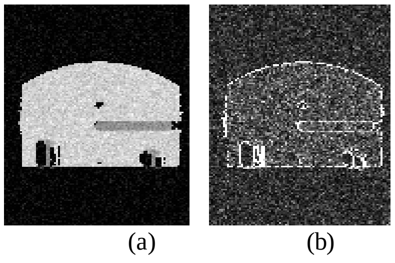
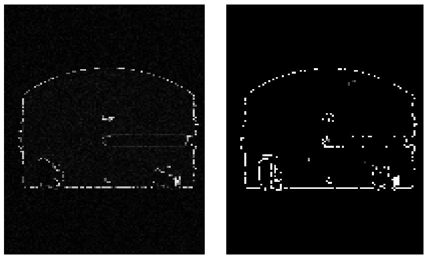


Figure 12: Canny edge map of Figure 7

Evaluation of the images showed that under noisy conditions Canny, Robert, Sobel exhibit better performance, respectively. Canny yielded the best results as shown in Figure 13. 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.



(a) (b)



(c) (d)

Figure 13: Comparison of Edge Detection technique on Noisy Image (a) Original Image with Noise (b) Sobel (c) Robert (d) Canny

III. PERFORMANCE EVALUATION

Edge detection methods investigated so far are further assessed by quality measures that give reliable statistical evidence to distinguish among the edge maps obtained[14]-[17]. The absence of the ground truth edge map reveals the search for an alternative approach to assess and compare the quality of the edge maps resulted from the detectors exploited so far. The evidence for the best detector type is judged by studying the edge maps relative to each other through statistical evaluation. Upon this evaluation, an edge detection method can be employed to characterize edges to represent the image for further analysis and implementation.

Table 1 Relative frequencies (R) of the detected edge pixels

Operator	**Canny**	**Lap of Gaussian**
Canny	1	0.62386301
Lap of Gaussian	1.602916	1
Prewitt	3.7234124 4	2.32289929
Sobel	3.6973248 4	2.3066242
Roberts	4.2846176 6	2.67301447

Operator	**Prewitt**	**Sobel**	**Robert**
Canny	0.2685708 4	0.2704658 2	0.2333930 5
Lap of Gaussian	0.4304964 9	0.4335339 9	0.3741094 6
Prewitt	1	1.0070558 1	0.8690186
Sobel	0.9929936 3	1	0.8629299 4
Roberts	1.1507233 5	1.1588426 3	1

Table 1 gives the relative frequencies of the occurrence of edge pixels in the previous filters. For each edge map, max (n_{df}) where n_{df} is the frequency *f* of occurrence for the filter f is reported, and the ratio with respect to each other gives comparative statistics for the occurrence of edges. The Canny filter reports the higher detected edge pixels.

Table 2: Significant edge differences at edge level 0.05

	Can/Lap	Can/Pre	Can/Sob	Can/Rob	Lap/Pre
H	1	1	1	1	1
P	0	0	0	0	0
CI	(0.0373, 0.0640)	(0.0772, 0.0847)	(0.0770, 0.0845)	(0.0811, 0.0886)	(0.0363, 0.0423)
STATS	18.7479	42.1842	42.1085	44.6352	25.7156

	Lap/Sob	Lap/Rob	Pre/Sob	Pre/Rob	Sob/Rob
H	1	1	0	1	1
P	0	0	0.8404	1.2382e-004	5.0089e-005
CI	(0.0361, 0.0421)	(0.0403, 0.0462)	(-0.0023 ,0.0018)	(0.0019, 0.0059)	(0.0021, 0.0061)
STATS	25.6107	28.6903	-0.2014	3.8533	4.0726

Table 2 summarizes the t-test for every pair combination of the detected edge maps on comparing the average of the pair wise edge maps, the following statistics gives evidence on the judge for the best method in such environment. The only non significant difference exists between the Prewitt and the Sobel at 0.05 level of significance, with P-value given in the second row of the table. The STATS gives the t-statistics for every pair.

The CI gives the confidence limit. In conclusion, Table 2 gives the evidence that the methods produce different edge maps, only for the Prewitt and Sobel as mentioned previously.

IV. DISCUSSION

Figures 2 through 6 give edges maps for the different operators highlighted above. The focus in this study is on the detection of edges that produces a map representing the original image. This provides a foundation for selecting an appropriate edge detector for further application. Investigation is aimed at aiding the choice of an appropriate operator that is capable of detecting boundaries based on intensity discontinuities[18]-[20]. From the results above, although the Sobel operator provides both differencing and smoothing, it detects part of the edges in the image. The problem with the Roberts detector is that it relies on finding high spatial frequencies which fail to detect fine edges. This is illustrated in Figure 10.

The Laplacian responds to transitions in intensity. As a second order derivative, the Laplacian is sensitive to noise. Moreover, the Laplacian produces double edges and is sometimes unable to detect edge direction. The canny edge detector is capable of reducing noise. The Canny operator works in a multistage process. These can be summarized in a smoothing with a Gaussian filter, followed by gradient computation and use of a double threshold. The analysis in Table 2 illustrates the differences in the methods pair wise, only Prewitt and Sobel have approximately the same edge map. The Canny produces the best edge map as evidenced by the relative frequency analysis in Table 1.

V. CONCLUSION

In this paper, we have analyzed the behavior of zero crossing operators and gradient operator on the capability of edge detection for images. The methods are applied to the whole image. No specific texture or shape is specified. The objective is to investigate the effect of the various methods applied in finding a representation for the image under study. On visual perception, it can be shown clearly that the Sobel, Prewitt, and Roberts provide low quality edge maps relative to the others. A representation of the image can be obtained through the Canny and Laplacian of Gaussian methods. Among the various methods investigated, the Canny method is able to detect both strong and weak edges, and seems to be more suitable than the Laplacian of Gaussian. A statistical analysis of the performance gives a robust conclusion for this complicated class of images.

REFERENCES

[1] A. Huertas and G. Medioni, "Detection of intensity changes with sub-pixel accuracy using Laplacian-Gaussian masks" IEEE Transactions on Pattern Analysis and Machine Intelligence. PAMI-8(5):651-664, 1986.

[2] S. Selvarajan and W. C. Tat," Extraction of man-made features from remote sensing imageries by data fusion techniques" 22nd Asian Conference on Remote Sensing, 5-9 Nov. 2001, Singapore.

[3] H. Voorhees and T. Poggio," Detecting textons and texture boundries in natural images" ICCV 87:250-25,1987.

[4] E. Argyle. "*Techniques for edge detection*," Proc. IEEE, vol. 59, pp. 285-286, 1971.

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

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

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

[13] J. Canny. "*Finding edges and lines in image*". Master's thesis, MIT, 1983.

[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] 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.

[17] [17] 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.

[18] [18] 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.

[19] [19] 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.

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