A Novel Real-Time Proficient Algorithm for Edge Detection

Ankur Gupta[1] **and Pradip K. Das**[1]
[1]Department of Computer Science & Engineering,
Indian Institute of Technology Guwahati, Assam, India

Abstract - *Image Processing has been a popular topic of research for many years. It refers to any form of processing for which the input is an image and the output can be either an image or a set of characteristics or parameters related to the images. We have studied and implemented the various feature detection algorithms for edge detection, such as Sobel edge detector, Robert edge detector, Prewitt edge detector, Laplacian of Gaussian(LoG) edge detector and Canny edge detector. The various characteristics of these algorithms are analyzed on the basis of image output, the number of operations used and the run-time needed. On the basis of this, we have proposed a novel algorithm for edge detection in real-time and implemented the same which gives comparable results for the image output with reduced number of operations run-time. We compare these results with the previous algorithms and discuss the relative performances.*

Keywords: Image Processing, Edge Detection, Sensitivity Check, Real-Time

1. Introduction

Image processing deals images which are two dimensional entities like scanned documents, X-ray films, satellite pictures etc., captured electronically through scanner or camera that digitizes spatially continuous coordinates into discrete sequences. A digital image is mapping of three dimensional world into a set of two dimensional detached points. These points store numerical values that represent color for the pixel, and this can be used as input for digital computers for image processing. Here, processing essentially means algorithmic enhancement, manipulation, or analysis of the digital image data. Every image processing technique or algorithm takes an input, an image or a sequence of images and produces an output, which may be a modified image or a description of the input image contents. Image processing has a wide application area in medicines, computer vision, remote sensing, space and military applications [2].

Edge detection and feature detection are two of the most common operations in image analysis. An edge in an image is a contour across which the brightness of the image changes abruptly. In image processing, an edge is often interpreted as one class of singularities. In a function, singularities can be characterized easily as discontinuities where the gradient approaches infinity [3]. However, image data is discrete, so edges in an image often are defined as the local maxima of the gradient. We have reviewed the standard approaches for the detection algorithm and also the recent advances in the area. Among these are Robert edge detector, Sobel edge detector, Prewitt edge detector, LoG edge detector and Canny edge detector.

The Roberts Cross operator performs a simple, quick-to-compute, two dimensional spatial gradient measurements 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 [4]. The operator consists of a pair of 2×2 convolution kernels as shown in equation (1). One kernel is simply the other rotated by 90°.

$$G_x = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad , \quad G_y = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \tag{1}$$

These kernels maximally respond to the edges running at 45° to the pixel grid, one kernel for each of the two perpendicular orientations. The kernels produce separate measurements of the gradient component in each orientation (G_x and G_y). The absolute gradient is given in equation (2).

$$|G| = \sqrt{{G_x}^2 + {G_y}^2} \quad \text{or} \quad |G| = |G_x| + |G_y| \tag{2}$$

The angle of orientation of the edge giving rise to the spatial gradient is given by equation (3).

$$\theta = \tan\left(\frac{G_y}{G_x}\right) - \frac{3\pi}{4} \tag{3}$$

Sobel edge detector operator consists of a pair of 3×3 convolution kernels as shown in equation (4). One kernel is simply the other rotated by 90° [4].

$$G_x = \begin{pmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{pmatrix}, \quad G_y = \begin{pmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \tag{4}$$

These can then be combined together as we done for Robert's operator. The angle of orientation of the edge giving rise to the spatial gradient is given by equation (5).

$$\theta = \tan(G_y / G_x) \tag{5}$$

Prewitt operator [5] is similar to the Sobel operator and is used for detecting vertical and horizontal edges in images. The Mask used in the Prewitt's Operator is shown in equation (6).

$$h_x = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}, \qquad h_y = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad (6)$$

The Laplacian is a two dimensional 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 applied to an image that has first been smoothed with Gaussian Smoothing filter in order to reduce its sensitivity to noise [3-5]. The Laplacian $L(x, y)$ of an image with pixel intensity values $I(x, y)$ is given by equation (7).

$$L(x, y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \quad (7)$$

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. These kernels are approximating a second derivative measurement on image, so they are very sensitive to noise.

The Canny edge detection algorithm mainly focuses on low error rate, well localized edge points, and single response to a single edge. Based on this, the Canny edge detector first passes the image through a low pass filter to eliminate noise. Then the edge strength is determined by taking the gradient of the image. The Sobel operator performs a 2-D spatial gradient measurement on the image[5-6]. Then edge direction is calculated based on the range as given in Figure 1.

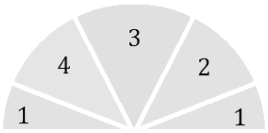


Fig. 1- Range of edge directions mapped on standard directions

Any edge direction falling within the range 1 (0 to 22.5 & 157.5 to 180 degrees) is set to 0 degrees, in the range 2 (22.5 to 67.5 degrees) is set to 45 degrees, in the range 3 (67.5 to 112.5 degrees) is set to 90 degrees and in the range 4 (112.5 to 157.5 degrees) is set to 135 degrees. Following it, non-maximum suppression is applied. Non-maximum suppression is used to trace along the edge in the edge direction and suppress any pixel value that is not considered to be an edge. This will give a thin line in the output image.

2. Novel Edge Detection Algorithm

As we have looked into several standard algorithms for the edge detection, we have now devised a novel enhancement algorithm. The novel algorithm aims at the decrease in the total number of operations, i.e. decrease in run-time, and the enhanced image should optimally give all the edge characteristics. Decrease in number of operation will help us to bring out real-time results at run-time. The algorithm consists of two parts, the first deals in calculating the sensitive part of the image and second part deals in edge detection in the sensitive parts only.

2.1 Sensitivity check in image

The image consists of different color set, but for all images, there is some background that differentiates it from rest of the image. The background enables the real object to come into focus. The algorithm here is to find out the background of the image. The background pixels can be eliminated for the process of edge detection. This drastically increases the performance of the algorithm.

The algorithm initially defines two threshold values A_T, the average threshold, and H_T, the histogram threshold. These thresholds are represented in equation (8).

$$A_T = \frac{1}{m \times n} \sum_{x=1}^{m} \sum_{y=1}^{n} f(x, y) \quad (8)$$

Here $(m \times n)$ is the image resolution and $f(x, y)$ represents the pixel value at point (x, y). Histogram threshold (H_T) can be calculated by processing the Image Histogram. The histogram of any image gives the pixel density of the image, say h_s, where s represents the grayscale range that is from $0\ to\ 255$. We find out the maximum dense range of the histogram as H_T, and it can be represented as in equation (9).

$$H_T = max_s \left(\sum_{t=-\alpha}^{\alpha} h_{s+t} \right) \quad (9)$$

Here, α represents the range for which H_T is to be calculated. We will be using these threshold values to define three new terms, background (β), low sensitive (γ), and high sensitive (η). A pixel $f(x, y)$ is considered to be in β if it is in range of $A_T\ and\ H_T$, γ if it is in A_T but not H_T and η if it is not in both, $A_T\ and\ H_T$. Here range represents a simple variation in $A_T\ and\ H_T$ such that pixels can be categorized in some broad manner. Let us define this range as r. Thus, β, γ, η are given in equation (10).

$$f(x, y) = \begin{cases} \beta & f(x, y) \in \{(A_T \pm r) \wedge (H_T \pm r)\} \\ \gamma & f(x, y) \in \{(A_T \pm r) \wedge \neg(H_T \pm r)\} \\ \eta & f(x, y) \notin \{(A_T \pm r) \wedge (H_T \pm r)\} \end{cases} \quad (10)$$

Now each pixel can be defined in a range of either of $\beta, \gamma\ or\ \eta$. Let us now focus into number of operations required to calculate all the thresholds and pixel's sensitivity. A_T requires $(m \times n)$ additions, 1 Multiplication and 1 division. H_T needs $s \times (2\alpha)$additions and α comparators. α is in range of $2 \leq \alpha \leq 4$ and s is 255 for the grayscale images. To classify a pixel in the sensitivity range we need 4 additions and 6 comparators. Same way can be represented for $m \times n$ pixels.

The total number of operations can be reduced, since now we need not to calculate the pixels in background, β

zone. We now further optimize this approach by disintegrating the image into small blocks. The block size may differ with respect to the user needs. We will represent block size by ω. By this the new image can be represented in $(m \times n)/\omega^2$ blocks. For each block we will be calculating a local average threshold value that will be used for the calculation of sensitivity. The average local threshold, A_{lt} value is shown in equation (11).

$$A_{lt}\left(\frac{m}{\omega},\frac{n}{\omega}\right) = \frac{1}{\omega^2}\sum_{x=m/\omega}^{\frac{m}{\omega}+\omega}\sum_{y=n/\omega}^{\frac{n}{\omega}+\omega} f(x,y) \tag{11}$$

Here $m/\omega \; and \; n/\omega$ gives the block number for which local average threshold is to be calculated. The number of operations required to calculate local threshold, A_{lt} is $(m \times n)$ additions, $(m \times n)/\omega^2$ multiplications and 1 division. The increased overhead can be decreased since now we just have to process these blocks for the sensitivity check rather than directly the pixel values. i.e. the number of operations will decrease by a factor of $1/\omega^2$on the basis of sensitivity.

The blocks will be processed by the edge detection algorithm with respect to sensitivity rule. The sensitivity rule defines a block in background, β should not pass through the edge detection algorithm. A block in low sensitive, γ needs a check, that if it is surrounded by any high sensitive, η block, then only it will be passed through the edge detection algorithm and block in high sensitive, η zone always needs to be passed through edge detection algorithm.

2.2 Edge Detection Technique

The edge algorithm should provide comparable results in real time. The various algorithms discussed above provide optimal edges. Here, we discuss a different approach, that detects edges and in less number of operations. As image can be represented as a two dimensional signal, the probability of a pixel to be an edge depends on the past pixel signal. We consider four two-dimensional signals, vertically down ($vecVer$), horizontally right ($vecHor$), South East ($vec45Dwn$) and North East ($vec45Up$). For all the signals, we calculate distortion in the signal. The distortion is controlled by upper threshold level σ, which is used to regulate the noise particles.

Each signal is represented in the form of vector of size τ. Each vector starts from the first pixel of the image and moves till end, covering the complete image. The vectors are implemented as queue structure, every time one pixel is added into the queue and the last element is removed from the queue, only if the queue size exceeds. The new pixel value, $g(x,y)$ after the edge detection is shown in equation (12).

$$g(x,y) = vecVer + vecHor + vec45Dwn + vec45Up \tag{12}$$

Distortions, δ can be represented as equation (13).

$$\delta = \left|\frac{1}{\tau}\sum_{\tau} vec_{\tau} - f(x,y)\right| \tag{13}$$

Here vec is the queue maintained and τ is the size of the queue. Vectors are updated as shown in equation (14) to (17).

$$vecVer(x,y) = \begin{cases}\delta & ; \quad \delta < \sigma \\ vecVer(x,y-1) & ; \quad \delta > \sigma\end{cases} \tag{14}$$

$$vecHor(x,y) = \begin{cases}\delta & ; \quad \delta < \sigma \\ vecVer(x-1,y) & ; \quad \delta > \sigma\end{cases} \tag{15}$$

$$vec45Dwn(x,y) = \begin{cases}\delta & ; \; \delta < \sigma \\ vecVer(x-1,y-1) & ; \; \delta > \sigma\end{cases} \tag{16}$$

$$vec45Up(x,y) = \begin{cases}\delta & ; \; \delta < \sigma \\ vecVer(x-1,y+1) & ; \; > \sigma\end{cases} \tag{17}$$

This Edge detection algorithm is capable of removing noise particle. The edges detected here are comparable to all other algorithms. Now, let us focus into the number of operations needed to perform this algorithm. $g(x,y)$ need $3(m \times n)$ addition operations, δ needs $4 \times 3(m \times n)$ additions and $4 \times (m \times n)$ multiplication, $vecVer$, $vecHor$, $vec45Dwn$, $vec45Up$ needs $(m \times n)$ comparators. Thus the total comes to $15(m \times n)$ additions, $4(m \times n)$ multiplications and $4(m \times n)$ comparators.

Comparing these with the Sobel, Prewitt's and LoG3x3 operator needs $17(m \times n)$ additions and $18(m \times n)$ multiplications. Canny Edge Algorithm needs much more than these. Thus our proposed edge detection algorithm needs less number of operations that decreases the run-time and increases the performance.

3. Experiments and Results

We have observed the standard operators with respect to the image output quality and also the number of operations used. The number of operations is directly proportional to the run-time. The less the number of the operations, the less will be the run-time. The novel algorithm proposed is giving good image outputs with lesser number of operations. Table 1 shows the number of operations for each of the algorithm.

Here, the number of comparisons is given for the worst case for the proposed algorithm. We can observe that the number of multiplications needed in Novel Edge Detection Algorithm is small, that needs largest cycles per instruction. This can compromise the extra additions and comparators for the same. Canny is not expanded since it contains a smoothing Gaussian filter, Sobel operator for edge detection, thinning of edges that would increase the number of operations.

Taking the worst case operations in the Table 1 for the novel algorithm, the algorithm performs better at run-time due to sensitivity check.

Table 1 – Number of Operations for each Operator

Name of the Operator	Number of			
	Addition	Multiplication	Division	Comparison
Sobel Operator	17(m×n)	18(m×n)	0	0
Robert Operator	8(m×n)	7(m×n)	0	0
Prewitt Operator	17(m×n)	18(m×n)	0	0
LoG 3×3 Operator	17(m×n)	18(m×n)	1	0
LoG 5×5 Operator	49(m×n)	50(m×n)	1	0
LoG 5×5 Operator	49(m×n)	50(m×n)	1	0

The sensitivity part detection algorithm detects the sensitive region and with respect to it only, the edges will be detected. We have shown a very simple example of it in Figure 2. The left side of the image contains the original grayscale image. The right hand side contains the image portion that will be only checked for the edges. This implies that only the sensitive part of the image may contain edges is only processed. This reduces the run-time complexity of the algorithm. Figure 3 shows the output of the images after implementing Robert operator, Sobel Operator, Prewitt Operator, LoG 3x3 Operator, LoG 5x5 Operator and our algorithm.

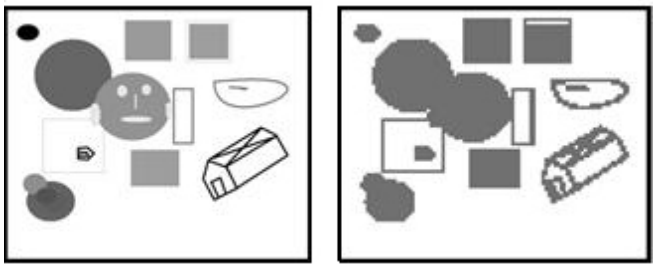
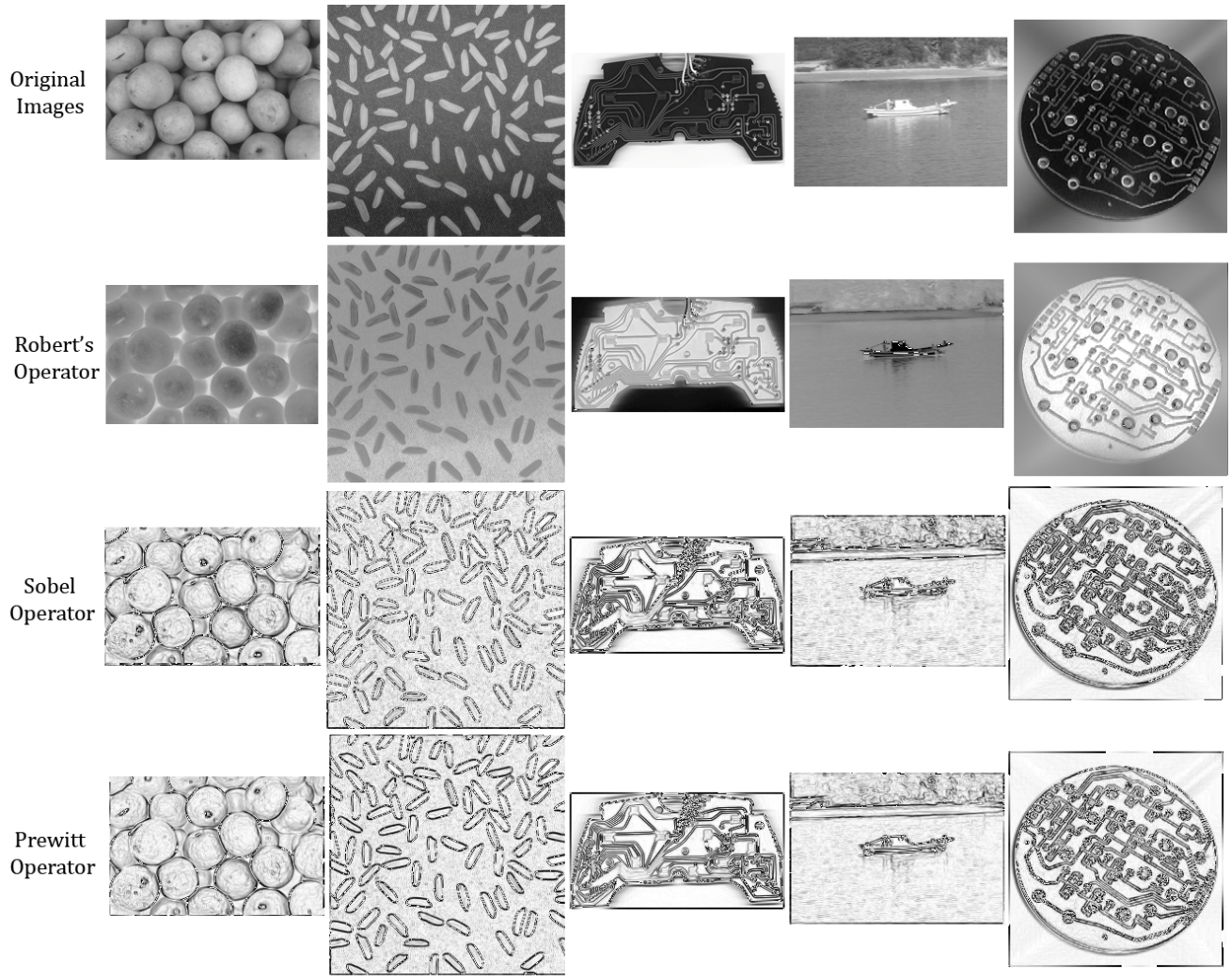


Figure 2 - Image Results for Sensitivity Check Algorithm

In Figure 3, we can see that Robert operator and LoG operator edges are not clear. Also we can observe various discontinuities in the edges in the Sobel and Prewitt operator. These discontinuities have been overcome in the results from novel algorithm. The edges in it are smoothly connected. Novel algorithm filters the noise particles and thus the output image will not be containing the noise particles as the part of edges.



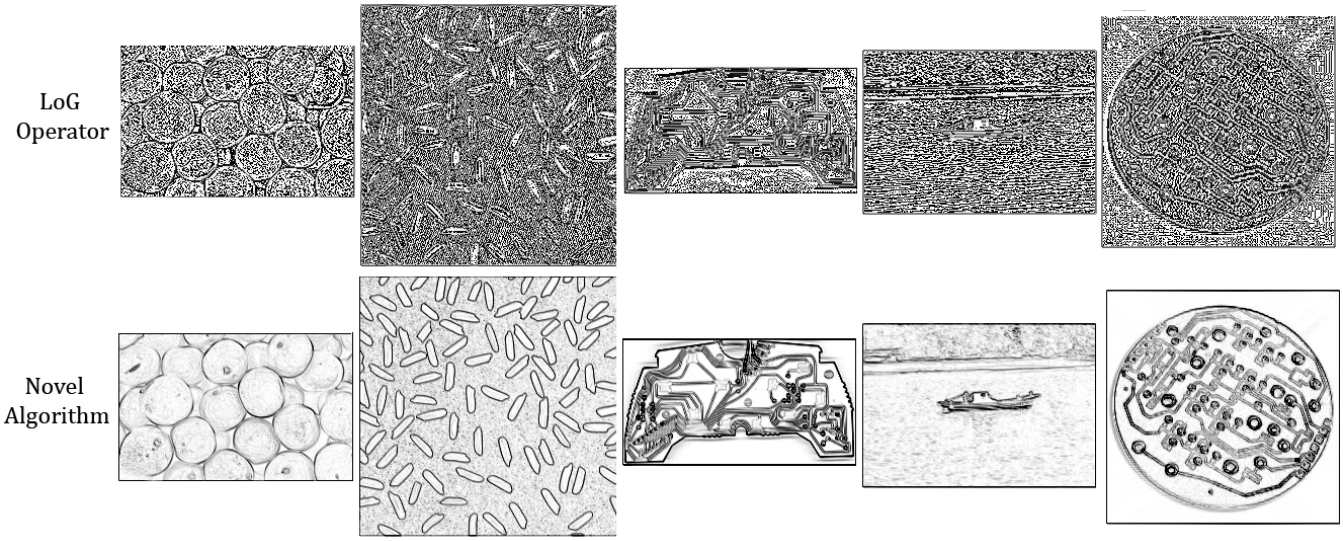


Fig. 3 - Image Results After Implementing Different Operators

Table 2 - Run-time for Each Operator given in milliseconds

Name of the Image	Resolution	Time of Execution (in Milliseconds)					
		Robert	Sobel	Prewitt	LoG 3x3	LoG 5x5	Novel Algorithm
(1).bmp	800×600	382	380	348	292	448	289
(2).bmp	800x600	343	363	347	290	452	336
(3).bmp	800×600	328	360	352	302	527	310
(4).bmp	800x600	318	379	353	304	429	323
(5).bmp	800×600	366	370	359	312	436	301
(6).bmp	800x600	388	346	357	300	455	325
(7).bmp	800×600	365	413	344	312	424	331
(8).bmp	800x600	392	417	344	302	405	329
(9).bmp	800×600	347	356	336	294	434	320
(10).bmp	800x600	372	344	337	288	412	276
(11).bmp	800×600	389	350	348	298	408	315
(12).bmp	800x600	303	359	354	293	417	321
(13).bmp	800×600	305	354	353	293	423	332
(14).bmp	800x600	292	343	348	296	531	309
(15).bmp	800×600	293	379	345	454	413	313
(16).bmp	800x600	295	390	342	293	445	278
(17).bmp	800×600	334	402	336	298	424	302
(18).bmp	800x600	294	359	344	297	409	297
(19).bmp	800×600	304	349	331	289	422	320
(20).bmp	800x600	365	344	339	295	405	328
(21).bmp	800×600	322	345	334	292	414	326
(22).bmp	800x600	328	348	335	305	452	250
(23).bmp	800×600	312	423	523	291	437	321
(24).bmp	800x600	316	343	340	294	444	315
(25).bmp	800×600	339	340	337	299	426	254
(26).bmp	800x600	365	394	356	297	423	287
(27).bmp	800×600	362	343	339	298	446	330
(28).bmp	800x600	308	338	359	450	407	298

(29).bmp	800×600	301	338	380	287	408	317
(30).bmp	450x600	181	190	203	170	272	167
(31).bmp	800×600	341	381	358	292	426	296
(32).bmp	450x600	178	192	206	166	227	163
(33).bmp	800×600	321	339	541	298	447	278
(34).bmp	800x600	336	323	533	299	422	26
(35).bmp	800×600	341	414	518	292	549	19
(36).bmp	800x600	303	330	355	307	422	110
(37).bmp	600x480	210	215	234	185	297	188
(38).bmp	800x600	310	334	438	298	435	323
(39).bmp	800×600	267	299	289	231	414	325
(40).bmp	800x600	304	267	384	240	367	114
(41).bmp	800×600	347	333	377	294	437	192

Table 2 gives run-time analysis of standard algorithms and novel algorithm using different images. This gives the run-time in milliseconds needed to run the algorithm on the respective image. From the observations in the table we can easily see that the Novel Algorithm is giving least run-time for most of the cases. For better understanding, the graphical representation of same is also given in Figure 4. We are showing Sobel Operator, Prewitt Operator, novel algorithm only, as the image results. Since, Robert operator and LoG operator are not giving good results, they are omitted from the graph. For the best approximation the average performance is shown in Figure 5.

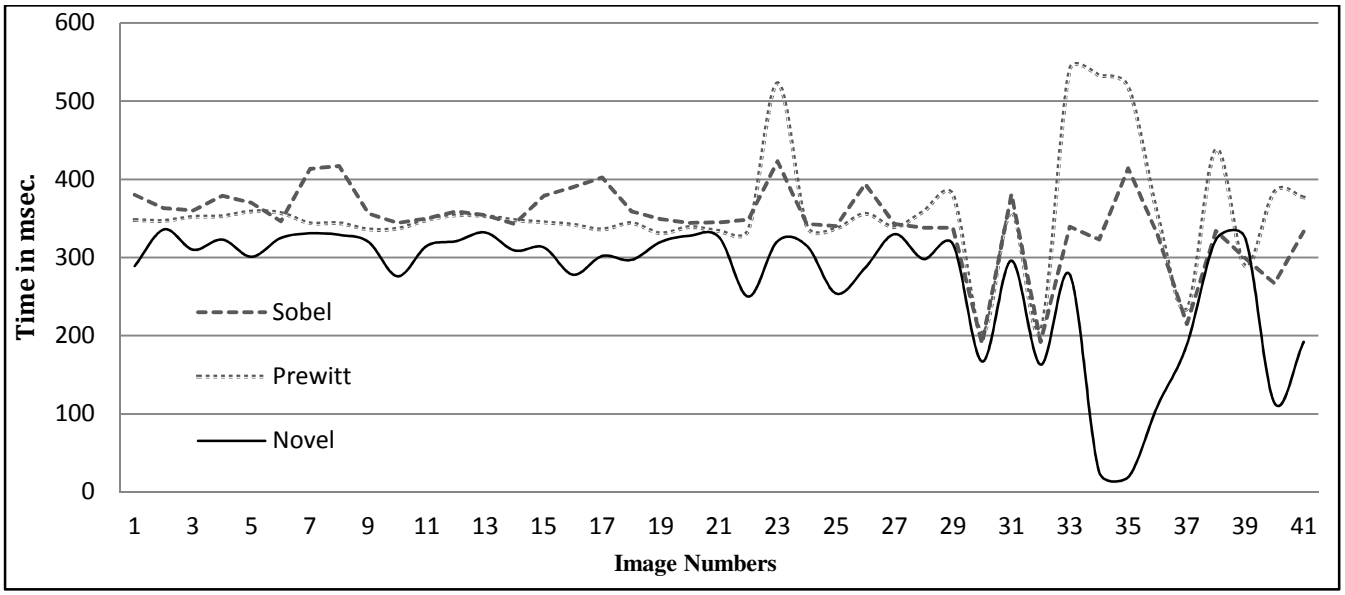


Figure 4 - Graphical Representation for the Run-time of different algorithms.

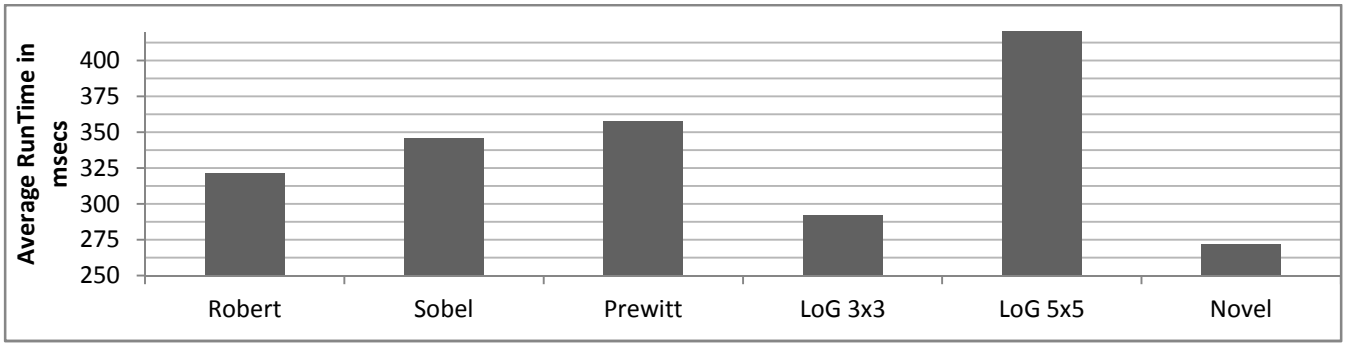


Figure 5 - Graphical Representation of the average run-time for various edge detection operators.

4. Conclusions and Future Work

The novel edge detection performs better from the various standard algorithms in respect to output image quality, number of operations and run-time. The algorithm can work in the real time environment to give the image features. The algorithm is able to give smooth and connected edges. Also, it filters the noise particles. Results have been verified by theoretical studies and on the basis of various experiments. We are now focusing into the properties of the sensitivity block definition with respect to different shapes and local threshold parameters. We will use this edge contours for the object recognition in real-time systems.

5. Bibliography

[1] Steve Mann. Intelligent image processing. IEEE, 2002.

[2] Manuel F M Costa. Application of image processing. In Advanced Study Center Co. Ltd., 2004.

[3] Leila Fallah Araghi Mohammed Reza Arvan. An implementation edge detection using neural networks. In Proceedings of the International Multi-Conference of Engineers and Computer Scientists, 2009.

[4] O R Vincent O Folorunso. A descriptive algorithm for Sobel edge detection. 2009

[5] J. Canny, "A computational approach to edge detection," IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-8, No. 6, pp. 679-698, 1986.

[6] Digital Image Processing Second Edition Rafael C. Gonzalez, Richard E. Woods[2002].

[7] Richard J Qian Thomas S Huang. Optimal Edge detection in two dimensional image. IEEE Transaction on Image Processing, Vol. *5,* No. 7, 1996

[8] Shalom Darmanjian, Michael Nechyba, A. Antonio Arroyo, Eric M. Schwartz. Darmanjian-Shaker Method (DSM), Novel Edge Detector with Blurring, 2004 Florida Conference on Recent Advances in Robotics (FCRAR 2004)

[9] Irsyadi Yani, M A Hannan, Hassan Basri, Edgar Scavino, Noor Ezlin bin Ahmad Basri. Detecting Object Using Combination of Sharpening and Edge Detection Method, European Journal of Scientific Research, ISSN 1450-216X Vol.32 No.1 (2009), pp.122-128, © EuroJournals Publishing, Inc. 2009, http://www.eurojournals.com/ejsr.htm

[10] LI Yong, WU Huayi, Adaptive Building Edge Detection by Combining Lidar Data & Aerial Images, ISPR

[11] Madhu S. Nair, *Member, IAENG*, R. Vrinthavani, and M. Wilscy, HBT Filter and Logarithmic Transform Based Edge Detection – A Modified Approach, Engineering Letters, 17:3, EL_17_3_02

[12] Febriliyan Samopa and Akira Asano. Hybrid Image Thresholding Method using Edge Detection, IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.4, April 2009

[13] Kim L. Boyer Sudeep Sarkar. On the Localization Performance Measure and Optimal Edge Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence Vol 16 , No 1m January 1994

[14] A. Nomura, M. Ichikawa, R. H. Sianipar, and H. Miike. Edge Detection with Reaction-Diffusion Equations Having a Local Average Threshold, ISSN 1054-6618, Pattern Recognition and Image Analysis, 2008, Vol. 18, No. 2, pp. 289–299. © Pleiades Publishing, Ltd., 2008.

[15] Qimei Hu, Xiangjian He and Jun Zhou. Multi-Scale Edge Detection with Bilateral Filtering in Spiral Architecture. Conferences in Research and Practice in Information Technology, Vol. 36

[16] Zhiqian Wang, K. Raghunath Rao, and Jezekiel Ben-Arie. Optimal Ramp Edge Detection Using Expansion Matching, IEEE Transactions On Pattern Analysis and Machine Intelligence, Vol 18, No 11 Novenmber 1996.