VOL. 6, NO. 1, JANUARY 2011 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences


www.arpnjournals.com

MULTI-LEVEL EDGE DETECTORS BASED ON CONVOLUTION MATRICES OF BASE-LENGTHS 2 AND 3

Yumnam Kirani Singh
Center for Development of Advanced Computing, Plot E2/1, Saltlake Sector-V, Kolkata, India
E-Mail: yksingh@cdackolkata.in

ABSTRACT

Proposed here is a new approach of edge detectors using edge-detecting masks generated from the basic edge detecting masks and convolution matrices of base lengths 2 and 3. The convolution matrices can be generated from a base unity matrix. These matrices behave like a low pass filter, which are effective in reducing noise in the edge detection process. The size of the convolution matrices increases with the increase in the levels. The higher the level, the generated edge detectors produce more prominent edges and have ability to suppress more noise. Experimental results show the superiority of the proposed scheme than most of the existing edge detection algorithms for different kinds of images. Algorithmically, it is simple and easy to implement.

Keywords: convolution matrices, edge detection, edge image, edge mask, unity matrix.

1. INTRODUCTION

Edge detection has been in various applications of image processing and pattern recognition. Several researchers have proposed several edge detectors using different approaches using simple differentiation operation [1, 7], Genetic Algorithms [2, 5] and Neural Networks [9]. Study on the techniques and performance of various edge detectors are available in the papers [5, 8, 10, 11]. Most commonly edge detectors are based on first and second order differentiation. Earlier edge detectors such as Robert's, Prewitt, Sobel, etc are based on first order differentiation. They are directional in the sense that they can detect edges in a certain direction such as horizontal, vertical or diagonal edges. The overall performance of directional edge detectors is poor and is highly dependent on the choice of the threshold values. Canny [3] proposed a new way of thresholding to enhance the overall performance of the directional edge detectors. But Canny's method is computationally complex and introduces unwanted edges if the threshold is not chosen appropriately. Later on multi-directional edge detectors were developed to extract the edges in all directions of the objects in an image. Edge detectors such as Laplacian and Laplacian of Gaussian [7] are multi-directional based on second order differentiation. They are simple but introduce false edges when used with zero crossing to detect edge positions [6]. To reduce the noise introduced with such an edge detector, generally a smoothing filter is applied before applying any edge detection filter. In Canny's or LoG, usually a Gaussian smoothing filter is applied before applying the edge detection filter.

In this paper, we propose a new multi-level edge detection scheme in which the generated edges preserve the shape of the objects in an image. The concept of multi-level edge detector is based on the convolution matrices, which are symmetric square matrices whose sizes increase with the increase in the level. The convolution matrices have similar shapes to Gaussian filters, which are appropriate to be used as low pass or smoothing filters. Once a convolution matrix at a particular level is generated, it can be used to generate an edge detecting filter by convolving the convolution matrix with a basic Laplacian mask or one of its derivatives. The generated edge mask is applied on the input image and then thresholded to give binary image which we call edge image. It is found that with the increase in the levels the prominence of the edges is increased as more edges are captured and more noises are suppressed.

The paper is divided into 6 sections. Section 2 deals with the generation of basic edge masks from the two Laplacian operators. The filtering operation and generation convolution matrices at various levels are described in Section 3. Section 4 gives the algorithm for the proposed multi-level edge detection. Experimental results are given in section 5 followed by a brief conclusion in section 6. In the paper, we use bold capitals to denote matrices, bold small letters denote vectors and others scalars.

2. GENERATION OF BASIC EDGE MASKS FROM LAPLACIANS

Edge detectors work by detecting locations where the sudden changes occur in the gray level of the image. The change in the gray level can be detected by comparing the difference in the successive pixel values in different directions. For a continuous image, I the change in the intensity can be computed by partial differentiation in that particular direction. To detect the change in the horizontal direction, partial differentiation is taken along the x-direction and to detect the change in vertical partial differentiation is done along y-direction. Mathematically,

$$I_x^{'} = \frac{\partial I(x,y)}{\partial x} = \lim_{\Delta x \to 0} \frac{I(x+\Delta x, y) - I(x,y)}{\Delta x} \text{ and}$$

$$I_y^{'} = \frac{\partial I(x,y)}{\partial x} = \lim_{\Delta y \to 0} \frac{I(x, y+\Delta y) - I(x,y)}{\Delta y}$$

Assuming $\Delta x, \Delta y \to 1$, we can write the above partial derivatives in terms of finite difference for a digital image $\mathbf{G}$ as

$\mathbf{G}_i^{'}(i,j) = \mathbf{G}(i+1,j) - \mathbf{G}(i,j)$ and $\mathbf{G}_j^{'}(i,j) = \mathbf{G}(i,j+1) - \mathbf{G}(i,j)$

Edge detectors based on first finite difference are known as crack edge detectors. These edge detectors can detect sharp changes in the gray level values. They are not suitable for detecting continual or slow variation in the gray level in an image. So, a second differentiation is usually performed to detect the slow changes in the intensity. This is usually applied using the Laplace operator. The Laplace operator for a continuous image I is given by

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

We can express the Laplacian operators in x and y directions in terms of the finite difference equations.
Let $\mathbf{G}_i^{''}$ and $\mathbf{G}_j^{''}$ denote the double finite difference in the horizontal and the vertical direction respectively. Then,

$$\mathbf{G}_i^{''}(i,j) = \mathbf{G}_i^{'}(i+1,j) - \mathbf{G}_i^{'}(i,j) = \mathbf{G}(i+2,j) - 2\mathbf{G}(i+1,j) + \mathbf{G}(i,j)$$

Which is nothing but convolution of the image $\mathbf{G}$ in the horizontal direction with a 1-dimensional filter $\mathbf{h}_i = [1,-2,1]$. If we shift one pixel to the left in the right hand side expression, we can write

$$\mathbf{G}_i^{''}(i,j) = \mathbf{G}(i+1,j) - 2\mathbf{G}(i,j) + \mathbf{G}(i-1,j)$$

Which similar to convolution of the image $\mathbf{G}$ with the filter shifted towards right by one position.
Similarly, we can write

$$\mathbf{G}_j^{''}(i,j) = \mathbf{G}(i,j+1) - 2\mathbf{G}(i,j) + \mathbf{G}(i,j-1)$$

which is the convolution of the image $\mathbf{G}$ in the vertical direction with a 1-dimensional filter $\mathbf{h}_j = \mathbf{h}_i^T = [1,-2,1]^T$, where $\mathbf{h}^T$ denotes the transpose of $\mathbf{h}$.

The filters $\mathbf{h}_i$ and $\mathbf{h}_j$ can also be written as 3x3 matrices and can be applied as 2-d filters.

$$\mathbf{H}_i = \begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix} \text{ and } \mathbf{H}_j = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Then, the filter corresponding to the finite Laplacian operators as

$$\mathbf{E}_1 = \mathbf{H}_i + \mathbf{H}_j = \begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

When we use E_1 as edge detector, the resulting output is an inverted image as central element is negative. To avoid this, we can write $\mathbf{E}_1$ as

$$\mathbf{E}_1 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

This filter $\mathbf{E}_1$ is the combination of edge detectors in the horizontal and the vertical directions.

We can modify H_i and H_j to detect diagonal edges which on adding give an edge filter.

$$\mathbf{E}_2 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & -1 \\ 0 & 2 & 0 \\ -1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} -1 & 0 & -1 \\ 0 & 4 & 0 \\ -1 & 0 & -1 \end{bmatrix}$$

We can generate more other edge detection masks of size 3 by 3 by linear combination of $\mathbf{E}_1$ and $\mathbf{E}_2$ using the relation $a\mathbf{E}_1 + b\mathbf{E}_2$ where a and b are integers both of which are not equal to zero. Some of the edge masks used for edge detections for simple values of a and b are given in Table-1.

Table-1. 8 effective edge masks of size 3x3.

$\mathbf{E}_1$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\mathbf{E}_4 = 2\mathbf{E}_2 - \mathbf{E}_1$	$\begin{bmatrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{bmatrix}$
$\mathbf{E}_2$	$\begin{bmatrix} -1 & 0 & -1 \\ 0 & 4 & 0 \\ -1 & 0 & -1 \end{bmatrix}$	$\mathbf{E}_5 = 3\mathbf{E}_2 - \mathbf{E}_1$	$\begin{bmatrix} -3 & 1 & -3 \\ 1 & 8 & 1 \\ -3 & 1 & -3 \end{bmatrix}$
$\mathbf{E}_3 = \mathbf{E}_1 + \mathbf{E}_2$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	$\mathbf{E}_6 = 3\mathbf{E}_2 - 2\mathbf{E}_1$	$\begin{bmatrix} -3 & 2 & -3 \\ 2 & 4 & 2 \\ -3 & 2 & -3 \end{bmatrix}$
$\mathbf{E}_4 = \mathbf{E}_2 - \mathbf{E}_1$	$\begin{bmatrix} -1 & 1 & -1 \\ 1 & 0 & 1 \\ -1 & 1 & -1 \end{bmatrix}$	$\mathbf{E}_8 = 4\mathbf{E}_2 - 3\mathbf{E}_1$	$\begin{bmatrix} -4 & 3 & -4 \\ 3 & 4 & 3 \\ -4 & 3 & -4 \end{bmatrix}$

The expression $\mathbf{E} = a\mathbf{E}_1 + b\mathbf{E}_2$ can be written as

$$\mathbf{E} = \begin{bmatrix} -b & -a & -b \\ -a & 4(a+b) & -a \\ -b & -a & -b \end{bmatrix}$$

Let us see the possible values of a and b to give a valid edge detector. As we know the central element of in the edge detector, the 3x3 matrix $\mathbf{E}$, need be non-negative in order to produce non-inverted binary edge image. We can consider the following three cases
Case-1: Both a and b are positive integers

In this case, the central element of $\mathbf{E}$ is always positive.

Case-2: a is positive and b is negative

In this case, a must be greater than b in order to make the central element of $\mathbf{E}$ positive. The resulting $\mathbf{E}$ becomes

$$\mathbf{E} = \begin{bmatrix} b & -a & b \\ -a & 4(a-b) & -a \\ b & -a & b \end{bmatrix}$$

But in addition, the value of a must be greater than twice the value of b so that the sum of elements of the first row or column does not equal to zero.

Case-3: a is negative and b is positive

In this case, the resulting $\mathbf{E}$ becomes

$$\mathbf{E} = \begin{bmatrix} -b & a & -b \\ a & 4(b-a) & a \\ -b & a & -b \end{bmatrix}$$

So, b must be greater than a in order to make the central element of $\mathbf{E}$ positive.

Thus, given any two integers a and b, we can easily generate a valid edge detection filter $\mathbf{E}$.

Finding suitable edge detector

It is found that the performance of an edge detecting filter depends on the type of the image. We can find the way for selecting a best edge detector for a given image by finding the values of a and b that gives the best output.

The criterion here for the best edge detector is the number of black pixels for a chosen threshold. The edge detecting filter corresponding to particular values of a and b that yields maximum number of black pixels in the binary image is assumed to be the best edge detector for the given image.

To find the edge detecting filter that gives the maximum number of black pixels, we put the edge mask to $\mathbf{E}_2$. We compute the number of black pixels in the resulting binary image for a chosen threshold. Then for each edge mask for different values of a and b for the three different cases, we compare the number of black pixels contained in the resulting binary images. The corresponding values of a and b which give the maximum number of black pixels within the chosen range of a and b for each case are then recorded. In this way, we get the different edge masks for different images, which give good edge images. The best masks for different images used in the experiment in this paper are given in the second row of the Table-2. The corresponding parameters of these masks such as a, b and the case types are given in the first row.

Table-2. Edges masks for different images.

Images	**Amrishpuri**	**Barbara**	**House**	**Lena**
Parameters	A=24, b=1, case-2	a=11, b=1, case-2	a=3, b=28, case-1	a=37, b=11, case-3
Basic masks **E**	1 -24 1 -24 92 -24 1 -24 1 (A)	1 -11 1 -11 40 -11 1 -11 1 (B)	-3 -28 -3 -28 124 -28 -3 -28 -3 (H)	-37 11 -37 11 104 11 -37 11 -37 (L)
Label-3 masks based on unity matrix of base-length-2. ($\mathbf{F}_2^3$)	1 -22 -46 -22 1 -22 0 44 0 -22 -46 44 180 44 -46 -22 0 44 0 -22 1 -22 -46 -22 1 (A2)	1 -9 -20 -9 1 -9 0 18 0 -9 -20 18 76 18 -20 -9 0 18 0 -9 1 -9 -20 -9 1 (B2)	-3 -34 -62 -34 -3 -34 0 68 0 -34 -62 68 260 68 -62 -34 0 68 0 -34 -3 -34 -62 -34 -3 (H2)	-37 -63 -52 -63 -37 -63 0 126 0 -63 -52 126 356 126 -52 -63 0 126 0 -63 -37 -63 -52 -63 -37 (L2)

One can also find different masks by inserting zeros between the successive elements.

$$\mathbf{E}_{04} = \begin{bmatrix} -2 & 0 & 1 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 4 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & 0 & 1 & 0 & -2 \end{bmatrix}$$

which is obtained by inserting zeros between successive elements of $\mathbf{E}_4$.

Note here, that with the increase in the size of the filter mask, the edges become thicker.

3. CONVOLUTION AND CONVOLUTION MATRICES

2D-Convolution

The filtering operation applied here is the 2-d convolution operation. When an image $\mathbf{X}$ of size $m \times n$ is convolved with a filter $\mathbf{H}$ of size $r \times c$, the size of the filter output $\mathbf{Y}$ becomes $(m+r-1)\times(n+c-1)$ which is larger than the size of the input image.

$$\mathbf{Y}(p,q) = \sum_{i=1}^{r}\sum_{j=1}^{c} \mathbf{X}(i+p-1, j+q-1) * \mathbf{H}(i,j)$$

Symbolically the convolution operation is denoted as $\mathbf{Y} = \mathbf{X} \otimes \mathbf{H}$

This filtering operation (2-d convolution) is used in the generation of multi-level filters. A level-n edge detection filter or mask is generated from a basic edge detection mask by convolving with a multi-level smoothing filter. The generation of an edge detection filter at level-n can be done in either of the following way.

$\mathbf{E}^n = \mathbf{E} \otimes \mathbf{F}^{n-1}$ where $\mathbf{F}^k = \mathbf{F}^{k-1} \otimes \mathbf{F}$

$\mathbf{E}^n = \mathbf{E}^{n-1} \otimes \mathbf{F}$ where $\mathbf{E}^k = \mathbf{E}^{k-1} \otimes \mathbf{E}$

In either ways, the generated edge mask will remain the same, since the convolution operation is commutative.

Convolution matrices

A convolution matrix is a square matrix which is generated by the 2-D convolution of a square matrix known as base matrix whose elements are all 1's. A convolution matrix which is generated from the base matrix of length k, after applying 2-D convolution n times will be known as base-k level-n convolution matrix. Such a convolution matrix will be denoted by $\mathbf{F}_k^n$.

Base matrices of length 2 and 3 at level 1 are $\mathbf{F}_2$ and $\mathbf{F}_3$ are given.

$$\mathbf{F}_2 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \text{ and } \mathbf{F}_3 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Similarly, we can define the base matrices of higher lengths. From a base matrix, a level-n convolution matrix can be generated by 2-D convolution.

$$\mathbf{F}_k^n = \mathbf{F}_k^{n-1} \otimes \mathbf{F}_k$$

Where the length of the $\mathbf{F}_k^n$ is $k + (n-1)(k-1)$.

Some convolution matrices of base length 2 are

$$\mathbf{F}_2^2 = \mathbf{F}_2 \otimes \mathbf{F}_2 = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$\mathbf{F}_2^3 = \mathbf{F}_2^2 \otimes \mathbf{F}_2 = \begin{bmatrix} 1 & 3 & 3 & 1 \\ 3 & 9 & 9 & 3 \\ 3 & 9 & 9 & 3 \\ 1 & 3 & 3 & 1 \end{bmatrix}$$

and so on. Since these matrices are generated by the 2-D, convolution operations, they are simply known as convolution matrices.

Note that any convolution matrix at level-n can also be generated by matrix multiplication of transpose of the 1-D convolution array of base length-k at that level and the convolution array itself.

The convolution array of base length-2 at level-n is obtained by doing 1-D convolution n-times on the corresponding base array. The base array of base length k is an array having k elements which are all 1's. Thus, the convolution array of base length-k at level-n is given by the following relation.

$$\mathbf{f}_k^n = \mathbf{f}_k^{n-1} \circ \mathbf{f}_k$$

Where $\mathbf{f}_k^1$ or simply $\mathbf{f}_k$ denotes base array of length k and $\circ$ denotes convolution operation of 1-d arrays.

The base arrays of lengths 2 and 3 are respectively the arrays $\mathbf{f}_2 = \mathbf{f}_2^1 = [1\ 1]$ and $\mathbf{f}_3 = \mathbf{f}_1^1 = [1\ 1\ 1]$.

The convolution arrays of length 2 at level 2, 3, and 4 are thus,

$\mathbf{f}_2^2 = [1\quad 2\quad 1]$, $\mathbf{f}_2^3 = [1\quad 3\quad 3\quad 1]$, and $\mathbf{f}_2^4 = [1\quad 4\quad 6\quad 4\quad 1]$

Similarly, we can find the convolution arrays of base length-3 at any level.

The relation between convolution matrix and convolution array is

$$\mathbf{F}_k^n = \left(\mathbf{f}_k^n\right)^T * \mathbf{f}_k^n,$$

where * denotes matrix multiplication and $(\ )^T$ denotes transpose operation. Thus

$$\mathbf{F}_2^2 = \left(\mathbf{f}_2^2\right)^T * \mathbf{f}_2^2 = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * [1\quad 2\quad 1] = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Using the above relation, we can generate a convolution matrix of any base length-k at any level-n from the corresponding convolution array of the base array at that level. Once such a convolution matrix has been generated, a new edge detection filter mask can be generated easily by 2-D convolution from a given mask with the convolution matrix.

2D- Convolution for filtering operation

The 2-dimensional convolution operation used in the generation of the multi-level edge mask cannot be directly used for filtering operation, if we want to preserve the image size. We know the convolution operation increases the size depending on the convolution matrix used. In order to preserve the original size of the image, first few columns from the left and last few columns from the right as well as first few rows from the top and last few rows from the bottom of the output image are to be rejected. In other words, we are taking the middle portion of the output image corresponding to the size of the input image. The number of rows and columns to be rejected from the output image in order to preserve the image size depends on the size $r \times c$ of the filter. Usually, the first $\lceil (r-1)/2 \rceil$ rows and the last $\lfloor (r-1)/2 \rfloor$ rows are rejected. Then, first $\lceil (c-1)/2 \rceil$ columns and last $\lfloor (c-1)/2 \rfloor$ columns are rejected from the resulting image. The resulted output becomes equal to the size of the input image. This is because, mathematically, $\lceil (r-1)/2 \rceil + \lfloor (r-1)/2 \rfloor = r-1$, which is the number of rows increased due the convolution operation. If we reject, $r-1$ rows from the $m+r-1$ rows, there will be only m rows, which is the number of rows in

the original image. Similar is the case for columns. We apply this size preservation technique when using the 2-dimensional convolution operation as a filtering operation.

4. EDGE DETECTION ALGORITHM

First, an edge mask at level-n is generated from a basic edge mask and a convolution matrix of base length 2 or 3. The generated edge mask is then used to filter the input image. The filter output is then rescaled to make its range between 0 and 255. It is then thresholded at an appropriate value. The threshold value is found to be 5 to 10 percent less than the pixel value corresponding to the highest peak in the histogram of the scaled filter output. The resulted image is the binary image formed mainly by the edges of the image.

The proposed Edge detection involves the following 5-steps.

Step-1: Generation of $\mathbf{E}^n$ from an edge mask and a convolution matrix of base length-2 or 3.
Step-2: Application of size preserving filtering operation on the input image with $\mathbf{E}^n$.
Step-3: Scaling of the filtered output in the range 0 to 255.
Step-4: Finding the approximate threshold value.
Step-5: Binarization of the filtered output around the approximate threshold value.

These steps are described in details as follows.
Let

$\mathbf{X}$ → Input Image
$\mathbf{F}_k$ →Unity matrix of base length 2 or 3 depending on the value of k=2 or 3.
$\mathbf{E}^n$ →Edge Filter at level-n
$\mathbf{Y}$ → Filter Output
Th → Threshold
$\mathbf{B}$ → Binary Output.

Step-1: Generation of $\mathbf{E}^n$ from the level-n Edge detection filters. $\mathbf{E}^n = \mathbf{E}^{n-1} \otimes \mathbf{F}_k$, where $\mathbf{E}$ is the basic edge mask.
Step-2: Do filtering operation and generate $\mathbf{Y}$

$$\mathbf{Y} = \mathbf{X} \otimes \mathbf{E}^n$$

Step-3: Scaling the filtered output range
Compute Ymin, Ymax the minimum and maximum values of $\mathbf{Y}$.
Modify $\mathbf{Y}$ as follows

$$\mathbf{Y} = \left\lfloor 255 * \frac{\mathbf{Y} - Y\min}{Y\max - Y\min} \right\rfloor$$

Step-4: Finding the approximate threshold value
Find the pixel value P whose number is the maximum in $\mathbf{Y}$.
Set the threshold to Th

$Th = P - p\%P$, where $5 \le p \le 10$.

Step-5: Binarization at the Threshold

$$\mathbf{B}(i,j) = \begin{cases} 255, \text{ if } \mathbf{Y}(i,j) \ge Th \\ 0, \text{ Otherwise} \end{cases}$$

for all (i, j) within the size of the image.

Note here that the convolution matrix has the similar shape as those of Gaussian masks of appropriate size. Hence, Gaussian filter of appropriate size can also be used in place of the unity matrix at the first level or in place of convolution matrix at any level. Note that, as the level increases the size of edge mask increases. Consequently, the resulted edge image becomes less noisy with thicker edges.

5. EXPERIMENTAL RESULTS

Some comparative studies have been made to see the superiority of the proposed edge detection algorithm with the some of the popular edge detection algorithms. For experiment, we use the four images (Amrishpuri, House, Barbara and Lena). The image of Amrishpuri was downloaded from the website: http://www.filmyfriday.com/actors/amrish-puri and the other three are the well known images in the image processing community. Of the existing edge detectors mainly, Sobel, Laplacian of Gaussian (LOG) and Canny edge detectors are considered for comparison purpose. MATLAB is used for implementing the proposed algorithm and the MATLAB function edge is used for the existing edge detectors. To find the edge mask suitable for the images considered, we search for the values of a and b in the range of 0 to 50 for each case of possible edge mask. The appropriate parameters and the corresponding basic edge masks for the different images are shown given in Table-2. The basic masks are referred to as mask A, mask B, Mask H and Mask L according to the first alphabet of the image names. Also, we use edge masks of size 5x5 obtained from the basic edge masks using convolution matrix at level-2 of basic unity matrix of base-length -2. The corresponding masks are referred to as A2, B2, H2 and L2 as given in the last row of Table-2. The results of edge detectors on the four images are given in the Figures 1, 2, 3 and 4. In each Figure, the original image is shown in the top left corner. Thresholds are selected manually using hit and trial methods to get optimum performance of the edge detectors. The corresponding thresholds are also given in the labels of each Figure. Figure-1 and 2 show the performance of edge detectors on Barabara and House. In these two Figures, the pixel noises are not removed from the edge images of the edge masks proposed. It is found that the proposed edge detector produce, in spite of being noisy, more prominent and undistorted edges than the existing edge detectors considered. Figure-3 and 4 shows the performance of the edge detectors on the images of Amrishpuri and Lena. The edge images of Figure-3(e), 3(f), 4(e) and 4(f) are obtained from by removing the noise speckles using bwareaopen function of the MATLAB. It can be seen that even after removing the isolated pixels, not much significant edge information is lost from the resulting edge images. On the

other hand it is found that the performance of the existing edge detectors depends much on the image. But among the Sobel, LOG and Canny edge detectors, Canny is found to be better in most images as can be seen from the (b), (c) and (d) parts of the Figures 1-4. Sobel edge detector introduces less distortion in the resulting edge images but misses several edge information. LOG and Canny capture most of the informative edges but introduce false and distorted edges in their edge images. The distortion in the case of LOG is so much that it makes the edge image appear ghostly. The edge images obtained by the proposed algorithms are less distorted and thus have more resemblance to the input images. From the Figures ((e) and (f) parts), it can be observed that the prominence of the edges increases with the increase in the level but some of the fine edges are beginning to miss. So, multi-level edge detectors may be used when we are interested only on the detection of significant edges. The proposed edge detector performs equally well in most of the images in the sense that an edge mask derived for a given image can also be used for other images giving slightly poor performance.

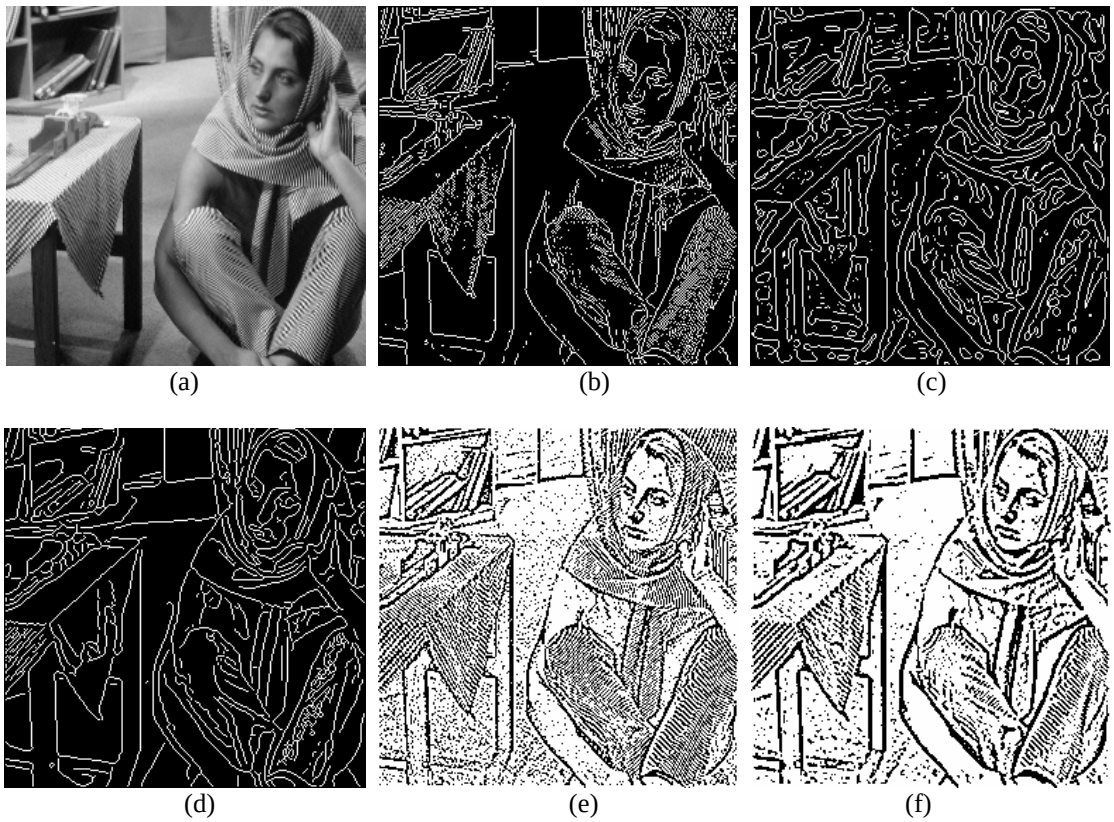
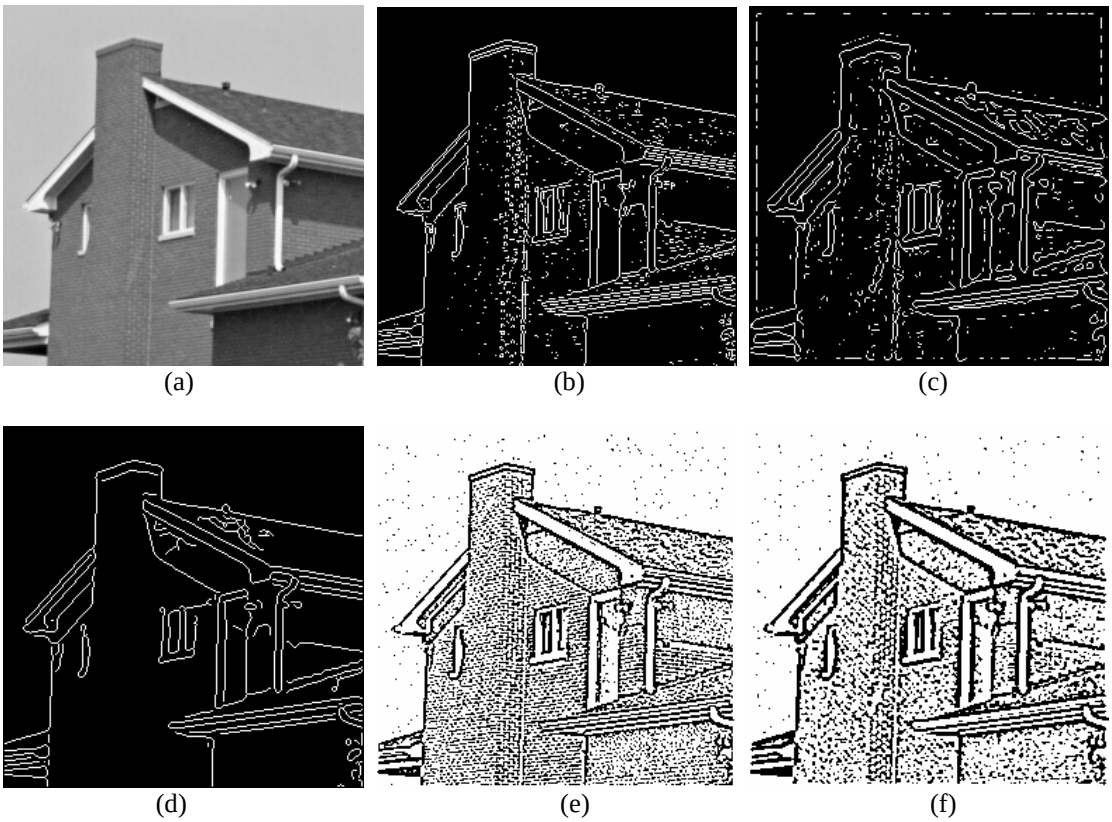


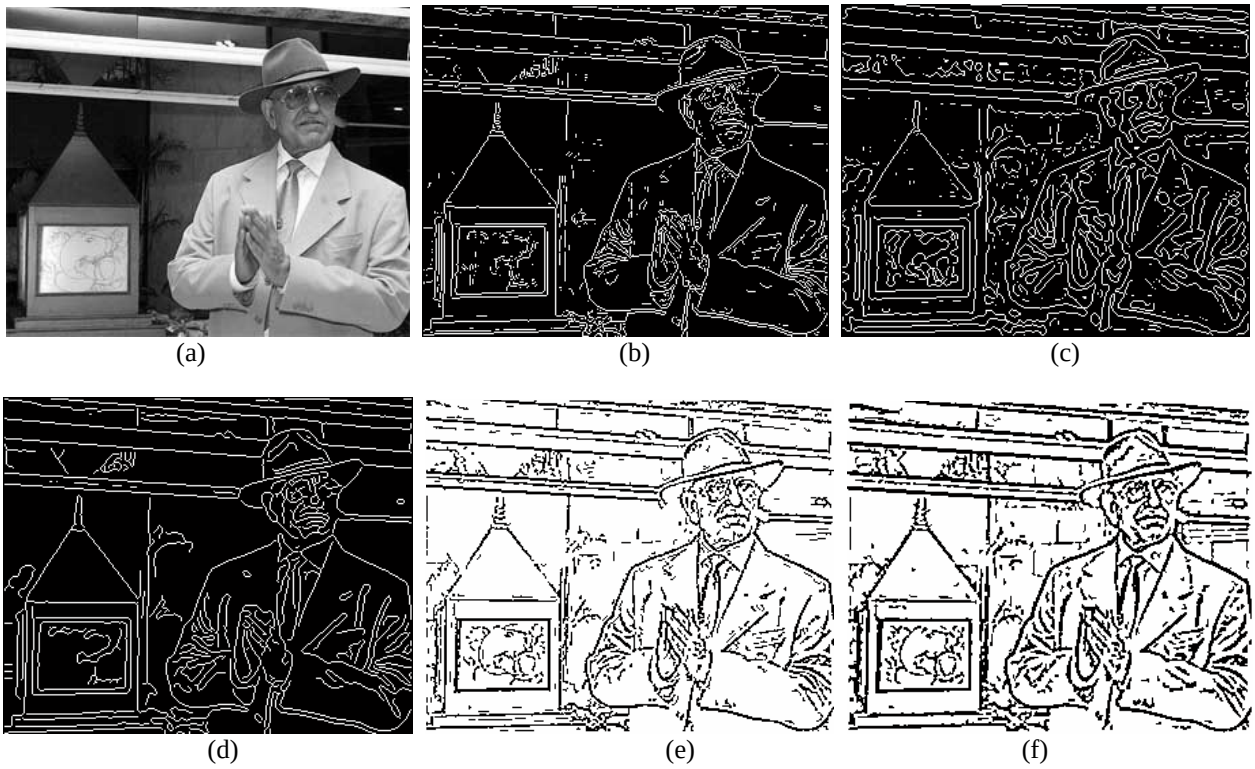
Figure-1. Performance of edge detectors on Barabara image. (a) Original Image, (b) Sobel, Thr=11, (c) LOG, Thr=0.25, (d) Canny, Thr= [0.0115, 0.1175] (e) Mask B, Thr=122 and (f) Mask B2, Thr=115.



(a) (b) (c)

(d) (e) (f)

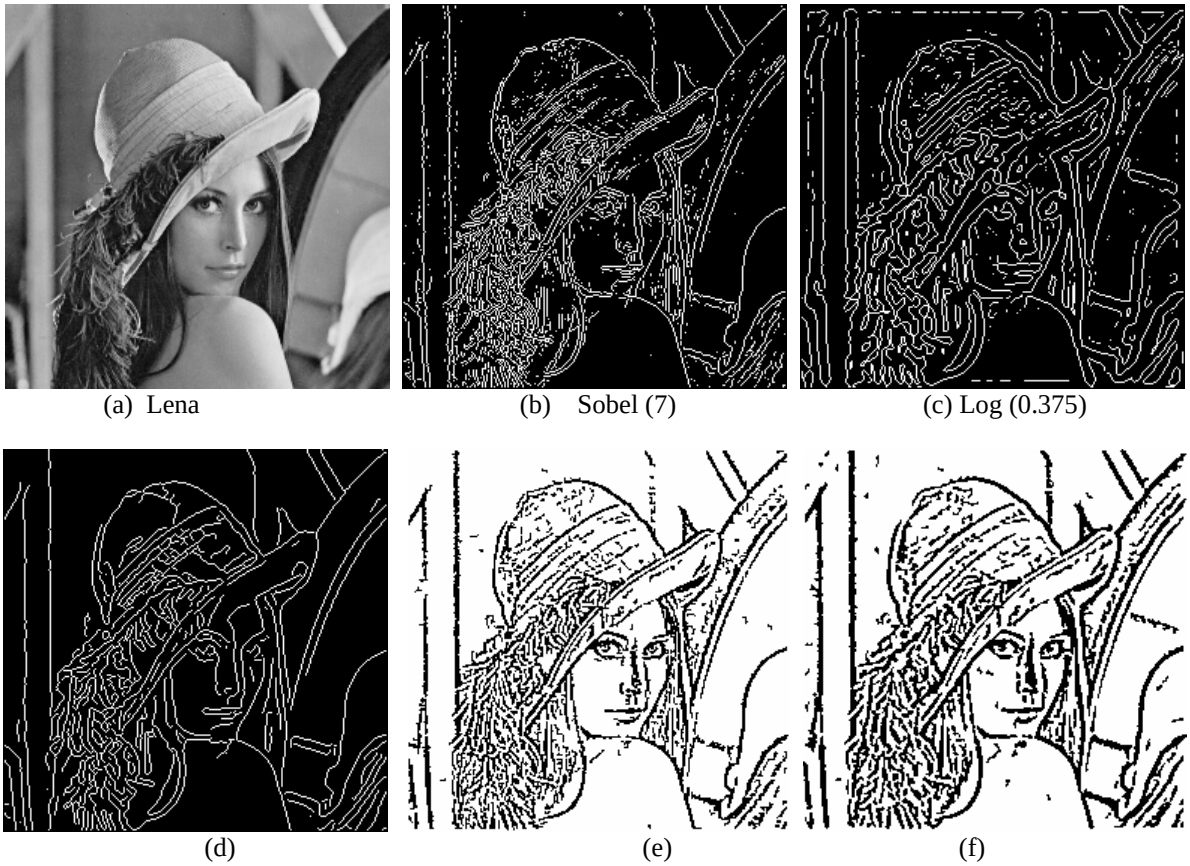
Figure-2. Performance of edge detectors on House image. (a) Original Image, (b) Sobel, Thr=7, (c) LOG, Thr=0.425, (d) Canny, Thr= [0.0115, 0.115] (e) Mask H, Thr=77 and (f) Mask H2, Thr=92.



(a) (b) (c)

(d) (e) (f)

Figure-3. Performance of edge detectors on Amrishpuri image. (a) Original Image, (b) Sobel, Thr=7, (c) LOG, Thr=0.425, (d) Canny, Thr= [0.0115, 0.125] (e) Mask A, Thr=98 and (f) Mask A2, Thr=96.



(a) Lena (b) Sobel (7) (c) Log (0.375)

(d) (e) (f)

Figure-4. Performance of edge detectors on Lena image. (a) Original Image, (b) Sobel, Thr=7, (c) LOG, Thr=0.375, (d) Canny, Thr= [0.015, 0.115] (e) Mask L, Thr=111 and (f) Mask L2, Thr=113.

6. CONCLUSIONS

A generalized way of multi-level edge detector is proposed. It provides easier way to find suitable edge detectors for a given image. The purposed edge detection algorithm can be used for generating thin as well as thick edges using convolution matrices at various levels. It has been observed that as the level increases the noise has been significantly reduced, the edges become prominent with the increase in the level but finer edges begin to disappear. If the purpose is to detect only the prominent edges, higher-level edge masks may be used. However, to get the finer edges, smaller masks at lower levels may be used. The proposed algorithm generates more natural and less distorted edge images as compared with the existing edge detectors. The proposed edge detection scheme can be used as edge detectors at lower levels where the thin edges are required. At higher level, the proposed edge detection scheme may also be used to generate natural sketches of the images.

REFERENCES

[1] E. Abdou and W. K. Pratt. 1979. Quantitative design and evaluation enhancement/thresholding edge detectors. Proc. IEEE. 67: 753-763.

[2] S. M. Bhandarkar, Y. Zhang and W. D. Potter. 1994. An edge detection technique using genetic algorithm based optimization. Pattern Recognit. 27(9): 1159-1180.

[3] J. Canny. 1986. A computational approach to edge detection. IEEE Trans. Pattern Anal. Machine Intell. PAMI-8: 679-697.

[4] J. J. Clark. 1989. Authenticating edges produced by zero-crossing algorithms. IEEE Trans. Pattern Anal. Machine Intell. 11: 43-57.

[5] L. Caponetti, N. Abbattists and G. Carapella. 1994. A genetic approach to edge detection. In: Intl. Conf. Image Processing. 94: 318-322.

[6] L. S. Davis. 1989. A survey of edge detection techniques. Comput. Graph. Image Process. 4(3): 248-270, 1976. M. Young, the Technical Writer's Handbook. Mill Valley, CA: University Science.

[7] D. Marr and E. C. Hildreth. 1980. Theory of edge detection. Proc. Roy. Soc. London B, No. 207, pp. 187-217.

[8] T. Peli and D. Malah. 1982. A study of edge detection algorithms. Comput. Graph. Image Process. 20(1): 1-21.

[9] V. Srinivasan, P. Bhatia and S. H. Ong. 1995. Edge detection using neural network. Pattern Recognit. 27(12): 1653-1662.

[10] V. Torre and T. Poggio. 1986. On edge detection. IEEE Trans. Pattern Anal. Machine Intell. PAMI-8: 147-163.

[11] D. Ziou and S. Tabbone. 1997. Edge Detection Techniques-An Overview. Dept. Math Informatique, Univ. Sherbrooke, Sherbrooke, QC, Canada, Tech. Rep. No. 195.