Diyala Journal of Engineering Sciences

ISSN 1999-8716

Printed in Iraq

Second Engineering Scientific Conference
College of Engineering –University of Diyala
16-17 December. 2015, pp. 430-440

EDGE DETECTION-APPLICATION OF (FIRST AND SECOND) ORDER DERIVATIVE IN IMAGE PROCESSING

Abdulbasit Alazzawi [1], Husam Alsaadi [2], Abidaoun Shallal [3], Saad Albwi [3]
[1]Science College, Diyala University, Iraq
[2] Basic Education Colleges, Al-Mustansiriyah University, Iraq.
[3] College of Engineering, Diyala University, Iraq.

ABSTRACT: - Edge detection is one of the most frequently used techniques in digital image processing. Edges typically occur on the boundary between two different regions in an image. In this paper the first method we will find the edge for image by using (1st *Order Derivative Filter)* method. In this method we take the 1st derivative of the intensity value across the image and find points where the derivative is maximum then the edge could be located. The gradient is a vector, whose components measure how rapid pixel value are changing with distance in the *x* and *y* direction. In second method we use the (2nd *Order Derivative Operators)* .

The 2nd derivative of an image where the image highlights regions of rapid intensity change and is therefore often used for edge detection zero crossing edge detectors. The zero crossing detector looks for places in the *Laplacian* of an image where the value of the *Laplacian* passes through zero i.e. points where the *Laplacian* changes sign. The criteria that used to comparison of results is (*Root mean square error*) for different threshold values that explain the result obtain by 2nd order are best of the result obtain by 1st order for all values of threshold.

1. INTRODUCTION

The goals of edge detection are: Produce a line drawing of a scene from an image of that scene, Important features can be extracted from the edges of an image (e.g. corners, lines and curve) and These features are used by higher-level computer vision algorithms (e.g., recognition).Various physical events cause intensity changes:(a. Geometric events such as Object boundary where discontinuity in depth and/or surface color and texture and Surface boundary where discontinuity in surface orientation and/or surface color and texture).

(b. Non-geometric events such as Specularity where direct reflection of light, i.e. a mirror, Shadows (from other objects or from the same object and Inter-reflections). The edge detection can be found by four steps:-

1- Smoothing: suppress as much noise as possible, without destroying the true edges.
2- Enhancement: apply a filter to enhance the quality of the edges in the image (sharpening).
3- Detection: determine which edge pixels should be discarded as noise and which should be retained (usually, threshold provides the criterion used for detection).
4- Localization: determine the exact location of an edge (sub-pixel resolution might be required for some applications, that is, estimate the location of an edge to better than the spacing between pixels). Edge thinning and linking are usually required here. The Criteria for good edge filters depend on some factor as following:-

i-No response to flat regions ⇒ Sum of mask values is zero: $\sum(r, c) = 0$.
ii- Isotropy: Response must be independent of edge orientation.
iii- Good detection: Minimize the probabilities of (detecting spurious edges caused by noise and missing real edges)

iv- Good localization: Detected edges must be as close as possible to true edges. v- Single response: Minimize number of false local maxima around true edge [1].

2. METHODS OF EDGE DETECTION

The majority of different methods may be grouped into two categories
Gradient method: The gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image.
Laplacian method: It 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 as shown in figure (1), with an edge by the jump in intensity. 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 signal as shown in figure (2).

Clearly, the derivative shows a maximum located at the center of the edge in the original signal. This method of locating an edge is characteristic of the “gradient filter” family of edge detection filters and includes the *Sobel method*. A pixel location is declared an edge location if the value of the gradient exceeds some threshold. As mentioned before, edges will have higher pixel intensity values than those surrounding it. So once a threshold is set, you can compare the gradient value to the threshold value and detect an edge whenever the threshold is exceeded. Furthermore, when the first derivative is at a maximum, the second derivative is zero. As a result, another alternative to finding the location of an edge is to locate the zeros in the second derivative. This method is known as the *Laplacian method* and the second derivative of the signal is shown in figure (3) [2].

2-1. 1st Order Derivative Filter (Sobel Filter)

Most edge detection methods work on the assumption that the edge occurs where there is a discontinuity in the intensity function or a very steep intensity gradient in the image. Using this assumption, if one take the derivative of the intensity value across the image and find points where the derivative is maximum then the edge could be located. The gradient is a vector, whose components measure how rapid pixel value are changing with distance in the *x* and *y* direction. Thus, the components of the gradient found by using the following equations (1) & (2).

$$\frac{\partial f(x,y)}{\partial x} = \Delta x = \frac{f(x+dx,y) - f(x,y)}{dx} \qquad \dots\dots(1)$$

$$\frac{\partial f(x,y)}{\partial y} = \Delta y = \frac{f(x,y+dy) - f(x,y)}{dy} \qquad \dots\dots(2)$$

Where ***dx & dy*** measure distance along the x and y directions respectively. In discrete images, one can consider ***dx & dy*** in terms of numbers of pixel between two points. ***dx = dy = 1*** (pixel spacing) is the point at which pixel coordinates are(i, j) thus, the value of (Δx *and* Δy) can calculated by equations (3) & (4).

$$\Delta x = f(i+1,j) - f(i,j) \qquad \dots\dots\dots(3)$$

$$\Delta y = f(i,j+1) - f(i,j) \qquad \dots\dots\dots(4)$$

In order to detect the presence of a gradient discontinuity, one could calculate the change in the gradient at (i, j) .This can be done by finding the following magnitude measure and the *gradient direction* θ is given by the equation (5).

$$\emptyset = \tan^{-1}\left[\frac{\Delta y}{\Delta}\right] \qquad \ldots\ldots..(5)$$

The *Sobel* operator is an example of the gradient method. It is a discrete differentiation operator, computing an approximation of the gradient of the image intensity function [3].

$$\Delta x = \begin{bmatrix} -1 & 1 \\ 0 & 0 \end{bmatrix}, \quad \Delta y = \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix}$$

An advantage of using a larger mask size is that the errors due to the effects of noise are reduced by local averaging within the neighborhood of the mask. An advantage of using a mask of odd size is that the operators are centered and can therefore provide an estimate that is based on a center pixel (i,j). One important edge operator of this type is the *Sobel* edge operator. The *Sobel* edge operator masks are given as:

$$\Delta x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad \Delta y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

The operator calculates the gradient of the image intensity at each point, giving the direction of the largest possible increase from light to dark and the rate of change in that direction. The result therefore shows how "abruptly" or "smoothly" the image changes at that point and therefore how likely it is that part of the image represents an edge, as well as how that the edge is likely to be oriented. In practice, the magnitude (likelihood of an edge) calculation is more reliable and easier to interpret than the direction calculation. Mathematically, the gradient of a two-variable function (the image intensity function) at each image point is a 2D vector with the components given by the derivatives in the horizontal and vertical directions. At each image point, the gradient vector points to the direction of largest possible intensity increase, and the length of the gradient vector corresponds to the rate of change in that direction. This implies that the result of the *Sobel* operator at any image point which is in a region of constant image intensity is a zero vector and at a point on an edge is a vector which points across the edge, from darker to brighter values [3].

2-2. 2nd Order Derivative Operators (Laplacian Filter)

The *Laplacian* is a 2-D measure of the ***2nd*** derivative of an image. The *Laplacian* of an image highlights regions of rapid intensity change and is therefore often used for edge detection zero crossing edge detectors). 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 binary image as output. The zero crossing detector looks for places in the *Laplacian* of an image where the value of the *Laplacian* passes through zero i.e. points where the Laplacian changes sign. Such points often occur at edges in images i.e. points where the intensity of the image changes rapidly, but they also occur at places that are not as easy to associate with edges. It is best to think of the zero crossing detector as some sort of feature detector rather than as a specific edge detector. Zero crossings always lie on closed contours, and so the output from the zero crossing detectors is usually a binary image with single pixel thickness lines showing the positions of the zero crossing points [4,5].The derivative operator *Laplacian* for an Image is defined as shown in equations (6), (7) and (8):

$$\Delta^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \qquad \ldots.\ldots..(6)$$

For X-direction, $\frac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y)$ ……..(7)

For Y-direction, $\frac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y)$ ……..(8)

By substituting, Equations (7) and (8) in (6), we obtain the equation (9)

$$\Delta^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y) \quad \dots(9)$$

If we use the value of Mask as explain blow we obtain on equation (9):

0	1	0
1	-4	1
0	1	0

3. THRESHOLD AND THINNING

i. Threshold

Once we have computed a measure of edge strength (typically the gradient magnitude), the next stage is to apply a threshold, to decide whether edges are present or not at an image point. The lower the threshold, the more edges will be detected, and the result will be increasingly susceptible to noise and detecting edges of irrelevant features in the image. Conversely a high threshold may miss subtle edges, or result in fragmented edges. If the edge thresholding is applied to just the gradient magnitude image, the resulting edges will in general be thick and some type of edge thinning post-processing is necessary. For edges detected with non-maximum suppression however, the edge curves are thin by definition and the edge pixels can be linked into edge polygon by an edge linking (edge tracking) procedure. On a discrete grid, the non-maximum suppression stage can be implemented by estimating the gradient direction using first-order derivatives, then rounding off the gradient direction to multiples of 45 degrees, and finally comparing the values of the gradient magnitude in the estimated gradient direction. We begin by using the upper threshold to find the start of an edge. Once we have a start point, we then trace the path of the edge through the image pixel by pixel, marking an edge whenever we are above the lower threshold. We stop marking our edge only when the value falls below our lower threshold. This approach makes the assumption that edges are likely to be in continuous curves, and allows us to follow a faint section of an edge we have previously seen, without meaning that every noisy pixel in the image is marked down as an edge. Still, however, we have the problem of choosing appropriate thresholding parameters, and suitable thresholding values may vary over the image. [8, 9]

ii. Thinning

Edge thinning is a technique used to remove the unwanted spurious points on the edges in an image. This technique is employed after the image has been filtered for noise (using median, Gaussian filter etc.), the edge operator has been applied to detect the edges and after the edges have been smoothed using an appropriate threshold value. This removes all the unwanted points and if applied carefully, results in one pixel thick edge elements. It can be used for several applications, but is particularly useful for skeletonization. In this mode it is commonly used to tidy up the output of edge detectors by reducing all lines to single pixel thickness. Thinning is normally only applied to binary images, and produces another binary image as output. The thinning operation is related to the hit-and-miss transform, and so it is helpful to have an understanding of that operator before reading on.

4. Result and Conclusion

The results that have been obtained by using (256 * 256) gray scale image as shown in Figure (4), after applied the first order derivative (*Sobel filter*) and second order derivative (*laplacian filter*) we gate on (gradient magnitude, gradient direction, horizontal gradient *X-axis* and vertical gradient *Y-axis*) as shown in figures (5) to (8) .But only the gradient magnitude use for determine edge, and compare with edge obtained by MATLAB function (canny filter) as shown in figure (9) to find root mean square error. The figures from (10) to (19) shows the result for different threshold (from 50 to 90), where the threshold here represent the value of gray level. For each threshold we find the root mean square error that represents the ***truncation error***. The root mean square is calculate between the result that obtain by applied the canny filter as true result and the results that obtain from (1st & 2nd) order as numerical result .The value of root mean square errors used as ***Objective*** fidelity criteria as shown in equation (10). Another fidelity criteria is ***subjective*** that performed by gathering a group of people that are representative the desired population, and then having all the test subjects evaluate the result according to a predefine scoring criterion. From result we note the quality of edges are increase with increase the value of threshold and the values of root mean square error are decrease as shown in table(1). This mean the information loses in first order more than the information loses in second order.

$$\textbf{\textit{rmse}}=\text{sqrut }(1/N^2\sum_{r=0}^{N-1}\sum_{c=0}^{N-1}[f(r,c)-h(r,c)]^2 \qquad \ldots\ldots\ldots (10)$$

Where the f (r,c) is edge obtain by applied canny filter, and h (r,c) is the edge obtain by applied 1st and 2nd order derivative.

REFERENCE

1) *Edge-Detection-Trucco*, Chapt-4-AND-Jain-et-al-Chapt-5-Definition of Edges, Edges are significant local changes of intensity.
2) *Edge Detection Tutorial.* Author: Bill Green, (2002).
3) *A Descriptive Algorithm for Sobel Image Edge Detection*, O. R. Vincent and O. Folorunso, Proceedings of Informing Science & IT Education Conference (InSITE), (2009).
4) *Filtering Corrupted Image and Edge Detection in Restored Gray scale Image Using Derivative Filters*, Chandra Sekhar Panda,Prof. (Dr.) Srikanta Patnaik, International Journal of Image Processing, (2009).
5) *Digital Image Processing (3rd Edition) Hardcover*, by Rafael C. Gonzalez & Richard, (2007).
6) *Numerical Methods for Engineers and Scientists, An Introduction with Applications using MATLAB*, Third Edition, Amos Gilat & Vish Subramaniam, (2014).
7) *Digital Image Processing Algorithms and Application*, by Ioannis Pitas, (2001).
8) Shapiro *Computer and Robot Vision*, Addison-Wesley Publishing Company, R. Haralick and L., 1992.
9) E. Davies Machine Vision: Theory, Algorithms and Practicalities, Academic Press, 1990.
10) E. Gose, R. Johnsonbaug, S. Jost, Pattern Recognition and Image Analysis,
a. Prentice-Hall, Englewood Cliffs, NJ, 1996
11) A. Cumani, “Edge detection in multispectral images,” CVGIP: Graphical
a. Models and Image Proc. 1991.
b. D. Marr and E. Hildreth, "Theory of Edge Detection," Proceedings of the Royal Society of London, B207, 1980.

Table (1): Show how the Value of Root Mean Square Error for (1st and 2nd) Derivative Change with Different Value of Threshold.

No	Threshold Value	Rmse for 1st derivtive	Rmse for 2nd derivtive
1	50	0.59743	0.51934
2	60	0.57274	0.49411
3	70	0.54882	0.54882
4	80	0.53078	0.4573
5	90	0.51542	0.44253

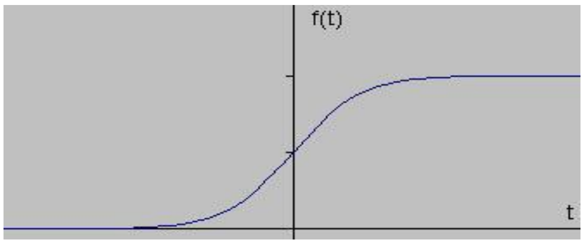

Figure (1): Signal Curve.

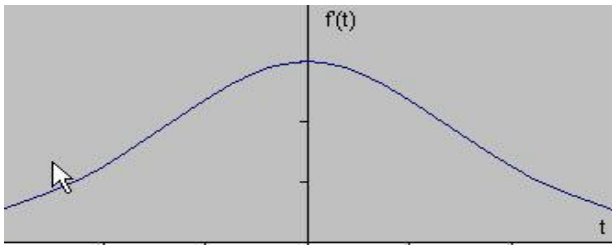

Figure (2): Max Value Derivative

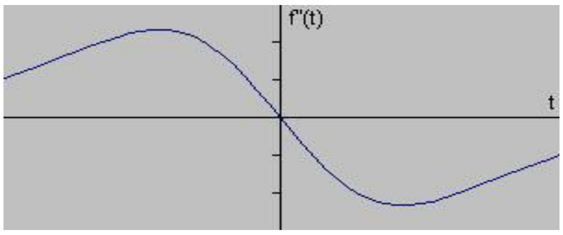

Figure (3): Zero Crossing Derivative

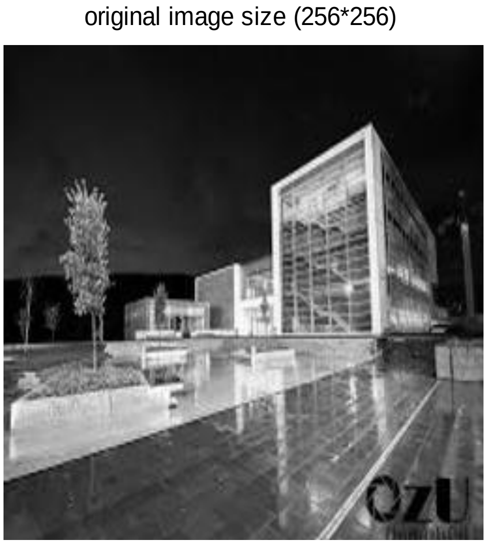

Figure (4): The Original Image with Size (256 * 256)

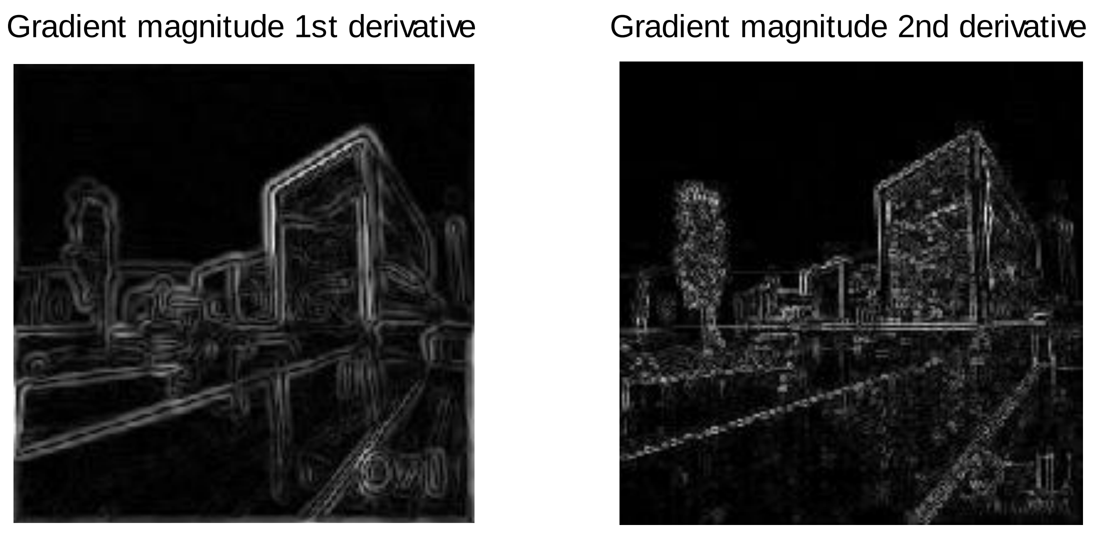
Gradient magnitude 1st derivative
Gradient magnitude 2nd derivative

Figure (5): Gradient Magnitude for (1st and 2nd) Derivative

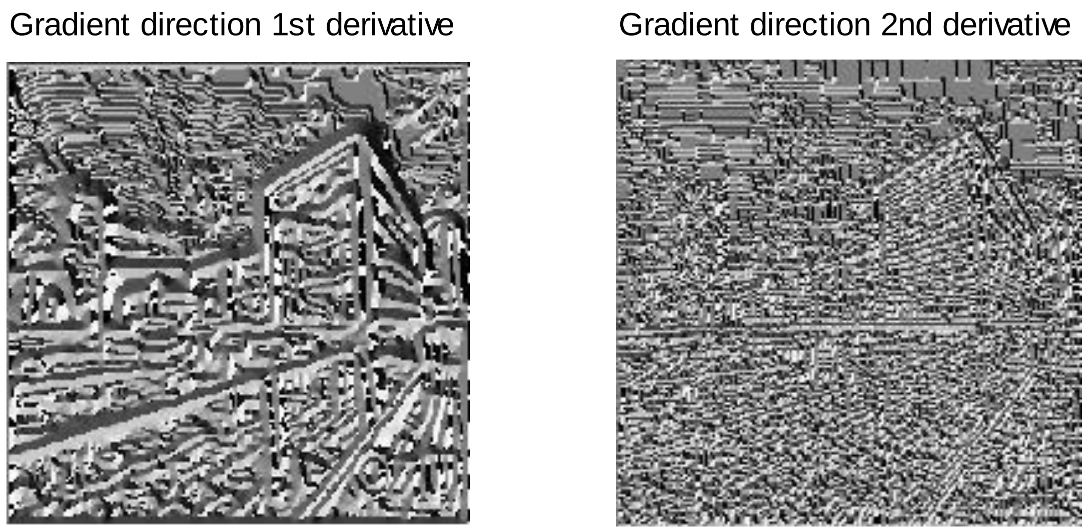
Gradient direction 1st derivative
Gradient direction 2nd derivative

Figure (6): Gradient Direction for (1st and 2nd) Derivative

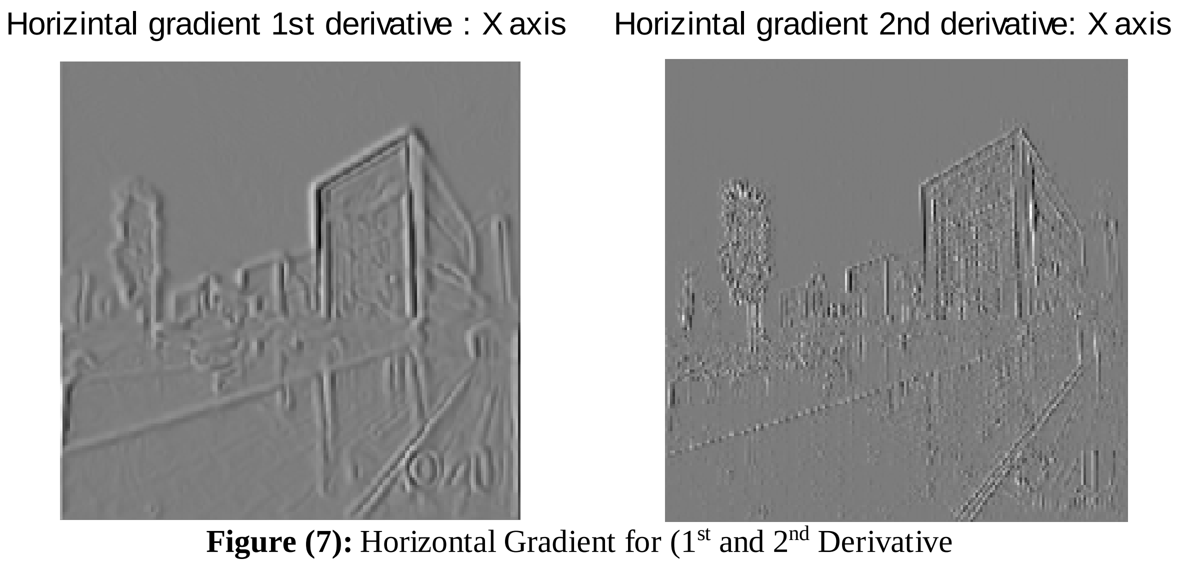
Horizintal gradient 1st derivative : X axis
Horizintal gradient 2nd derivative: X axis

Figure (7): Horizontal Gradient for (1st and 2nd Derivative

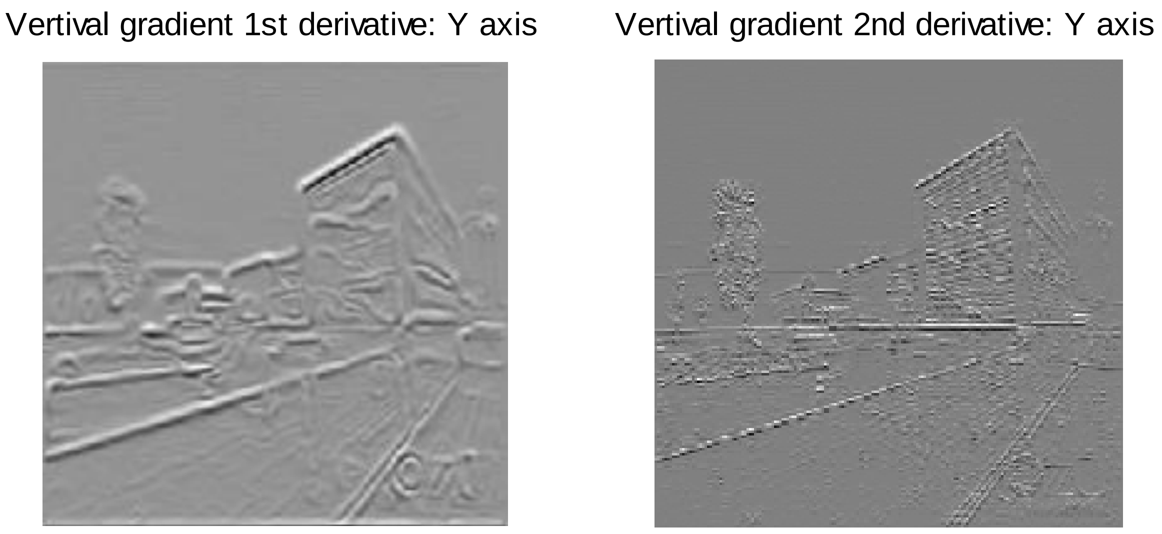
Vertival gradient 1st derivative: Y axis
Vertival gradient 2nd derivative: Y axis

Figure (8): Vertical Gradient for (1st and 2nd) Derivative

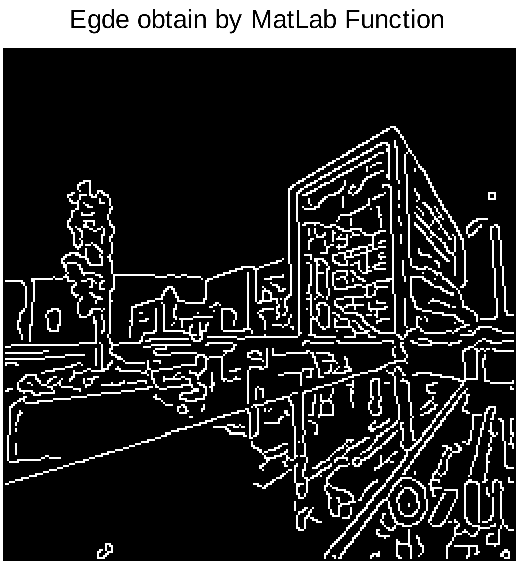

Figure (9): Edge obtain by matlab function

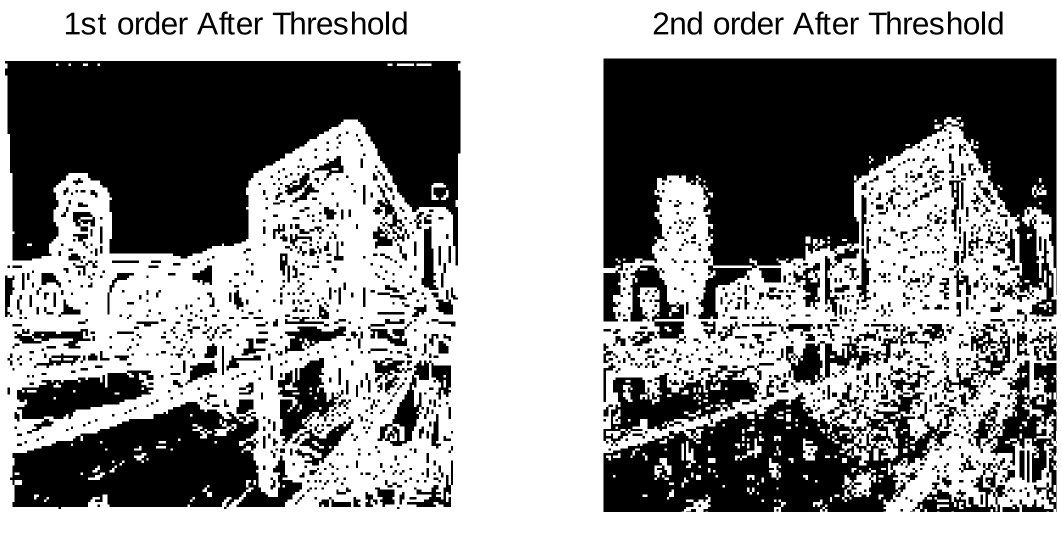

Figure (10): Edge of Image for (1st and 2nd) Derivative with Threshold=50

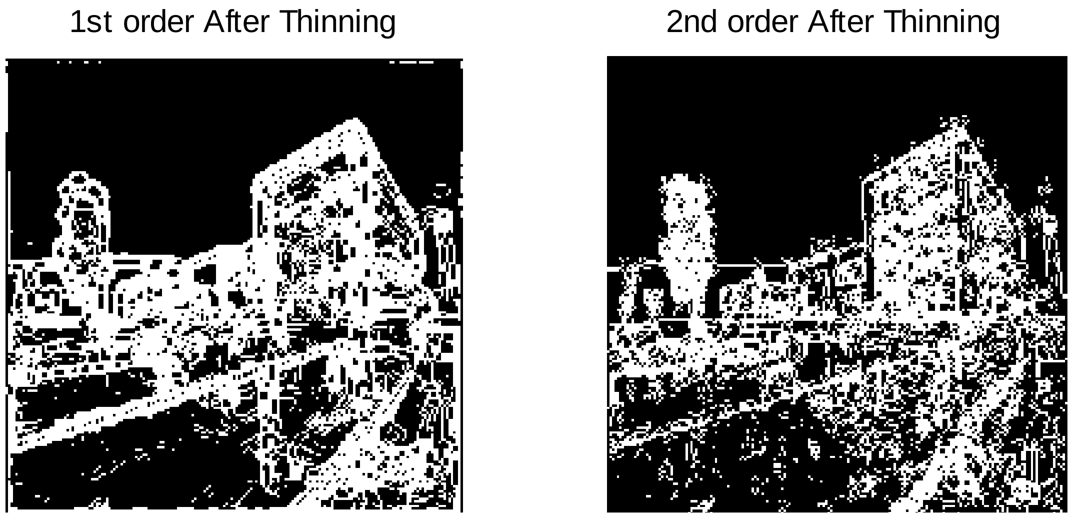

Figure (11): Edge after Thinning for (1st and 2nd) Derivative with Threshold=50

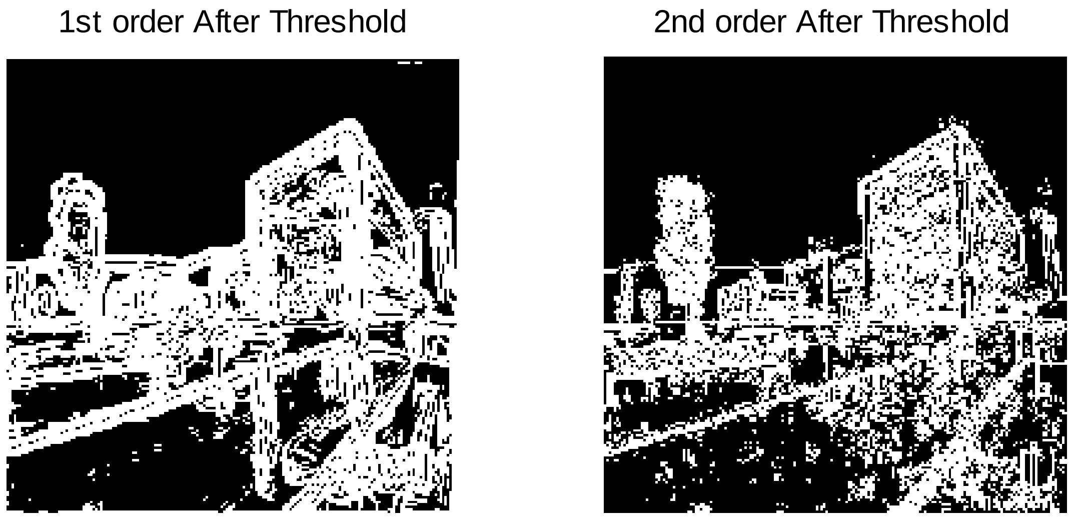

Figure (12): Edge after Threshold for (1st and 2nd) Derivative with Threshold=60

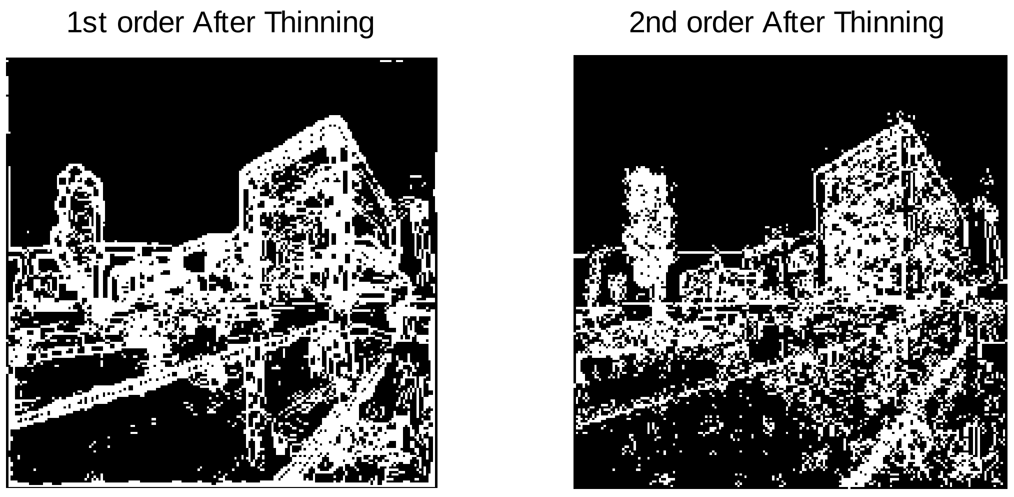

Figure (13): Edge after Thinning for (1st and 2nd) Derivative with Threshold=60

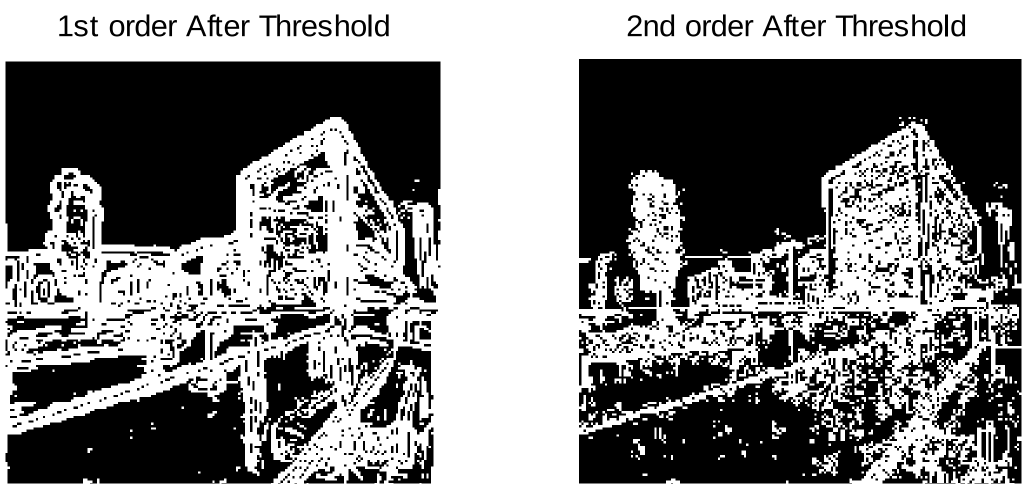

Figure (14): Edge after Threshold for (1st and 2nd) Derivative with Threshold=70

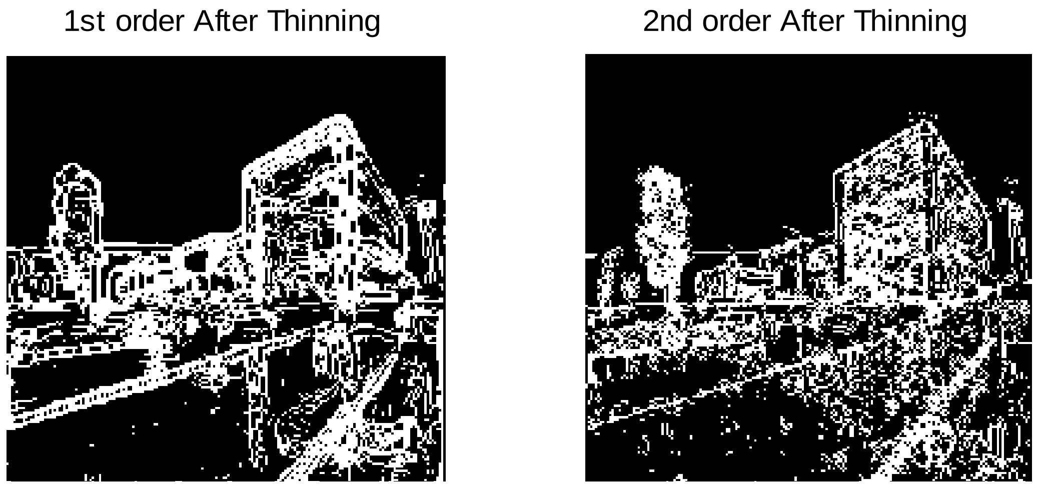

Figure (15): Edge after Thinning for (1st and 2nd) Derivative with Threshold=70

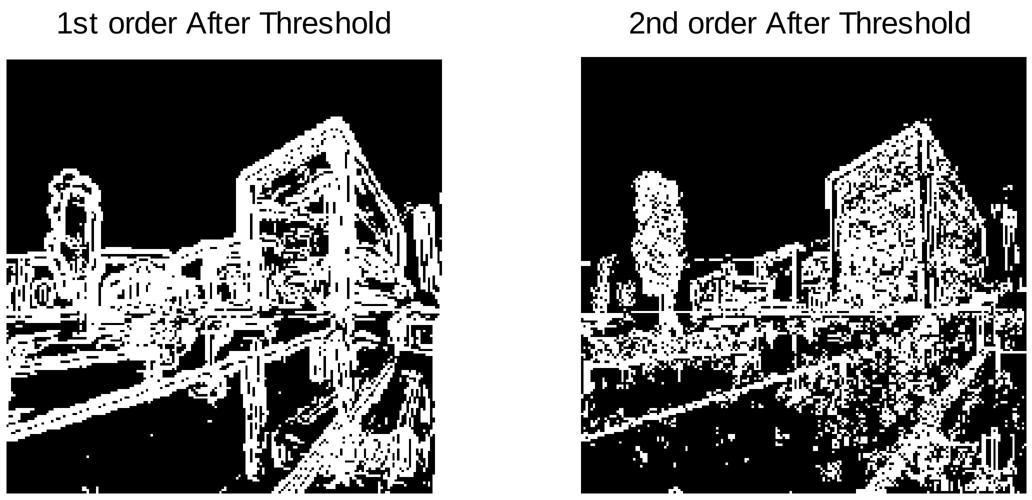

Figure (16): Edge after Threshold for (1st and 2nd) Derivative with Threshold=80

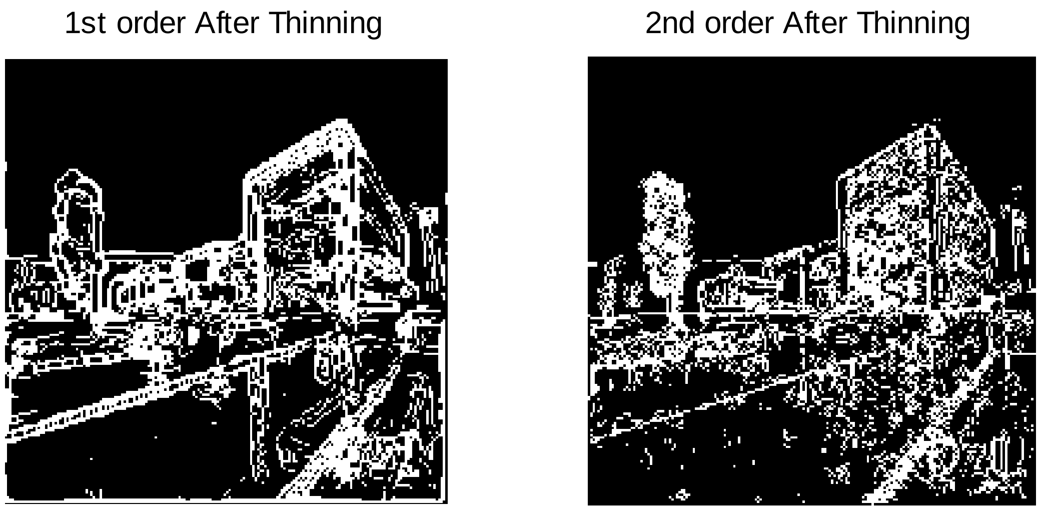

Figure (17): Edge after Thinning for (1st and 2nd) Derivative with Threshold=80

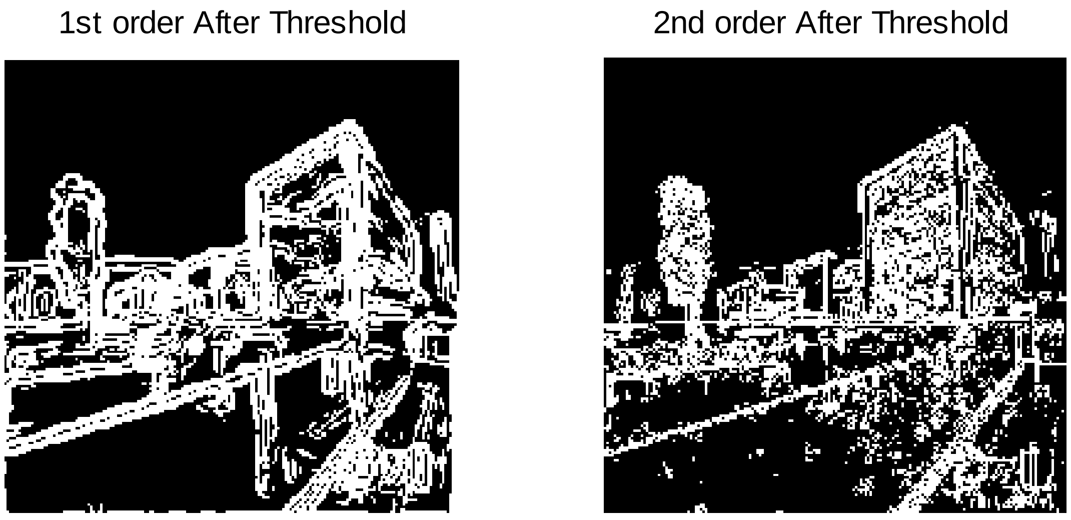

Figure (18): Edge after Threshold for (1st and 2nd) Derivative with Threshold=90

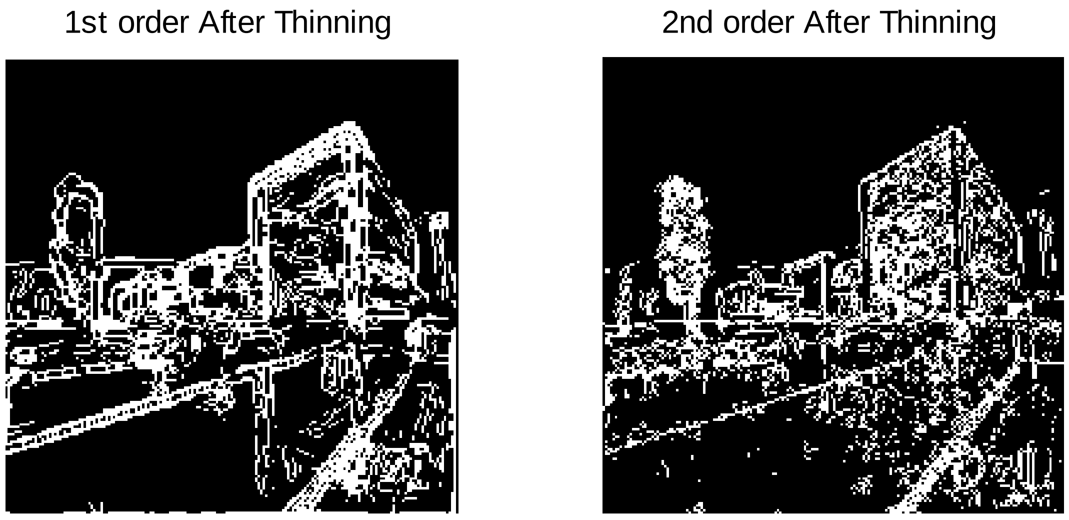

Figure (19): Edge after Thinning for (1st and 2nd) Derivative with Threshold=90