Comparison of Edge Detection Algorithms for Texture Analysis on Copy-Move Forgery Detection Images

Bashir Idris[1], Lili N. Abdullah[2], Alfian Abdul Halim[3], Mohd Taufik Abdullah Selimun[4]
Multimedia Department, Faculty of Computer Science and Information Technology[1, 2, 3]
Universiti Putra Malaysia (UPM), Selangor, Malaysia[1, 2, 3]
Computer Science Department, Faculty of Computer Science and Information Technology[4]
Universiti Putra Malaysia (UPM), Selangor, Malaysia[4]

***Abstract*—Feature extraction in Copy-Move Forgery Detection (CMFD) is crucial to facilitate image forgery analysis. Edge detection is one of the processes to extract specific information from Copy-Move Forgery (CMF) Images. It sensitizes the amount of information in the image and filters out useless ones while preserving the important structural properties in the image. This paper compares five edge detection methods: Robert, Sobel, Prewitt (first Derivative), Laplacian, and Canny edge detectors (second Derivatives). CMFD evaluation datasets images (MICC-F220) are tested with both methods to facilitate comparison. The edge detection operators were implemented with their respective convolution masks. Robert with a 2x2 mask, The Prewitt and Sobel with a 3x3 mask, while Laplacian and canny used adjustable masks. These masks determine the quality of the detected edges. Edges reflect a great-intensity contrast that is either darker or brighter.**

Keywords*—Edge detection; first derivative; second derivatives; robert; sobel; prewitt; laplacian; canny edge detector

I. Introduction

Image forensic analysis requires good-quality images in any orientation to accurately detect the image textural properties. A good quality image can give the best evaluation when investigating a crime scene on any query image. Therefore, the analysis of CMFD detection image texture plays a significant role in image forensics. Edge-based segmentation is one of the methods used to analyze image textures in CMFD.

Segmentation is an essential determinant for image information understanding and retrieval. It is also one of the frequent topics of discussion in the image processing and computer vision community [1]. Theoretically, image segmentation separates digital image data into a set of visually separate and identical regions based on parameters such as pixel intensity, similarity, discontinuity, cluster data, and so on [2]. The primary objective of segmentation is to clearly identify an image object from its background. Researchers have categorized segmentation into different techniques. The author [3] express segmentation as threshold techniques, edge detection techniques, region-based techniques, and connectivity-preserving relaxation methods. In [1] segmentation is seen as threshold-based, regional growth, edge, and segmentation based on clustering and weakly-supervised learning in CNN. In 2022, due to a significant improvement over the last four decades from the traditional segmentation-based method to advance deep learning algorithms, [2] categorized segmentation algorithms into rule-driven and data-driven methods. Methods for cleaning objects that rely on one or more rules are referred to as rule-driven. Classification methods that learn features from data, such as ANN and DL, are referred to as data-driven. According to [2], several rule-based segmentation methods, such as fuzzy-based image segmentation (FBS), have been abandoned because they do not give satisfactory performance or because their lengthy computation times do not sufficiently substantiate their use. However, the reviewed literature mentioned that thresholding and edge detection methods are the most popular and applicable traditional image segmentation methods. This paper primarily focused on the edge detection-based segmentation method.

Edge detection is a method for recognizing the points or pixels in an image where the luminance varies abruptly or significantly. These points are clustered along segments of lines known as edges. Edge detection seeks to identify and locate image discontinuities. Due to the high frequency of both noise and image, edge identification becomes problematic. This paper discusses the edge detection techniques to address the choice of an edge detection method based on distinctive edge identification, noisy removal, etc.

Filtering is a method of correcting or improving the process of selecting the best keypoints in an image [4]. Edge detectors utilize filters to highlight edges while removing noise in a given image. Smoothing, sharpening, and edge enhancement are some image-filtering activities invoked using filtering methods. These activities are used interchangeably to enhance images for proper image-processing applications [5]. The filters are utilized for locating the sharp, discontinuous edges. These discontinuities cause variations in pixel intensities, which define the object's boundaries [5]. By applying filters to images, salient key points can be efficiently located for object detection.

Edges are of fundamental importance during any image-processing task. The intensity values of pixels that share a neighborhood are compared in detecting edges. The significant changes in the dense regions are located as edges. These dense areas are often difficult to trace due to the presence of noise. The presence of noise in the image affects the detection of the Edges; therefore, noise needs to be cleaned before the detection of edges [6]. Noise reduction has been a challenge in the edge detection process, attracting the attention of many scholars in different areas of application that use images. Edge detection algorithms have been used for decades and have recently been employed in various domains because of their ability to detect and differentiate good boundaries in images.

Research has been conducted over the past decades to detect abrupt changes in image intensities (edges). Edge detection was employed in glass production for texture analysis [6], Canny Edge Detection Algorithm for Congregating Traffic Information [7], Sobel edge detector for synthetic aperture radar (SAR) detection [8], Laplacian edge detector for bladder cancer diagnosis [9], Kirsch edge detector for sharpening effect in images [10], Sobel edge detector for hardware implementation on the FPGA Nexys 4 DDR Board [11], copy-move forgery detection in digital images [12].

This paper presents an analysis of five types of edge detectors. It examines the edges by applying various edge operators and their respective filters. Five selected images in the MICC-F220 CMFD dataset are tested and compared to each other. This paper presented the related work in Section II; Section III presents the algorithms based on the first derivatives and explains the edge detector operators and their various masks. Section IV presented an algorithm based on the second Derivative, and Section V presented the result of the five operators. Finally, Section VI discussed the result, and the conclusion was depicted in Section VII.

II. Related Work

As shown in Fig. 1, an angle gradient vector indicates the intensity change at that edge pixel. The cycle spot in Fig. 1 is the corresponding gradient vector on one data point or pixel. The intensity changes from 0 to 255 in the gradient direction at that pixel. The gradient's magnitude determines the edge's strength by calculating the gradient in uniform regions; a zero vector is obtained, indicating that there are no edge pixels. In natural images, where there are rarely ideal discontinuities or uniform regions, as seen in Fig. 1, thus, the gradient magnitude is evaluated to determine whether to detect the edge pixels. Edge detection is one of the simplest and oldest image-processing procedures, it is frequently employed in recently advanced edge detection algorithms. Edge detection is a technique for detecting abrupt changes in image intensity. The use of first-order or second-order derivatives can be employed to detect those changes [13]. Edge detection methods were developed in the 1970s using small operators such as Sobel (3x3) to compute an approximation of the image's first Derivative. Edge detection involves the application of different size operators because the change in image intensity depends on the image scale. This variation in intensity can be demonstrated as a peak in Fig. 3, using gradient-based operators (first Derivative) or a sharp zero-crossing on gaussian-based operators (second Derivative), respectively.

The edge detection algorithms are divided into two distinct categories: Gradient-based (first Derivative) and Gaussian-based (second Derivative) [10]. An adaptive mask convolves the input image, resulting in a gradient image with edges recognized via the thresholding technique. To detect edges, the gradient operators (Sobel, Prewitt, and Roberts) examine the maximum and minimum intensity values. They determine the distribution of intensity values in each pixel's neighborhood to see if it should be categorized as an edge. Sobel, Prewitt, and Roberts operators are time-consuming and cannot be deployed for real-time applications (Bhardwaj & Mittal, 2012). On the other hand, the Laplacian of Gaussian (LoG) is categorized under the Gaussian-based method (second Derivative). Gradient-based operators are filters with various kernel sizes convolved with the original input image to produce the image gradient. Zero-crossing or second-order derivative methods are other names for Gaussian-based approaches. They significantly extract sharp zero-crossing points, and zero-crossing indicates the presence of maxima (an edge) [3]. The Canny operator is well-known for its exceptional performance, as it goes through noise reduction, gradient magnitude calculation as gradient operators, thresholding to maintain firm edges while deleting weak ones, and finally, non-maximum suppression for edge thinning through Hysteresis [8], [10].

Various survey has been conducted to investigate the impact of edge detection in digital images. This can be summarized in Table I.

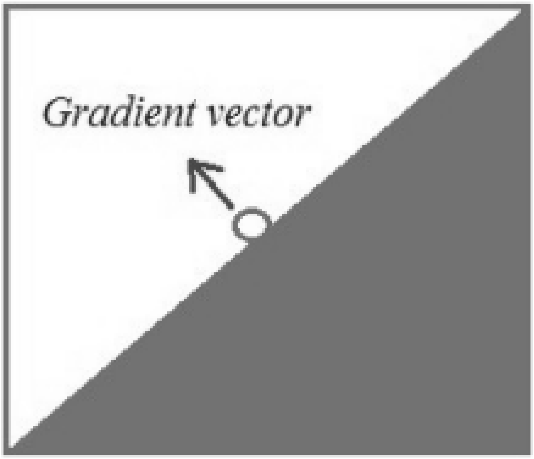


Fig. 1. Gradient Vector with Red Spot Indicating the Location of a Pixel.

TABLE I. Related Edge Detection Survey

S/N	Year	Author	Description of Survey
1	2022	Kheradmand, & Mehranfar [2]	Edge Segmentation-based techniques
2	2022	Jin et al.[14]	Recent advances in image edge detection
3	2017	Song, & Yan, [1]	Image Segmentation Techniques
4	2015	Öztürk & Akdemir [6]	Edge Detection Algorithms for Glass Production analysis
5	2015	Vikram Mutneja [15]	Methods of Image Edge Detection
6	2015	Li et al. [16]	Visual Feature Detection
7	2013	Lopez-Molina et al. [17]	Quantitative error measures for edge detection
8	2012	Shrivakshan et al. [17]	Color filter technology
9	2011	Papari, & Petkov [18]	Edge and line-oriented contour detection

III. Algorithms Based on First Derivative (Gradient-Based)

The main difference between the algorithms based on the first derivatives is the nature of the mask or filters (known as low pass filters) they applied during the computation of the derivatives. A general flow diagram for computing the First Derivative of an image is presented in Fig. 2.

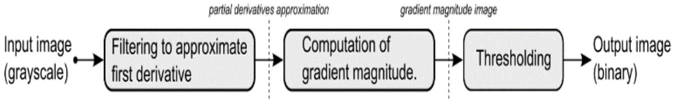

partial derivatives approximation
gradient magnitude image
Input image (grayscale)
Filtering to approximate first derivative
Computation of gradient magnitude.
Thresholding
Output image (binary)

Fig. 2. Block Diagram of the First Derivative Edge Detection Algorithms [13].

The intensity variations in the magnitude and direction of an image *f* can be calculated using the gradient operator (∇). It is a well-known tool in image processing for calculating image orientations. Let's consider a 1 D image signal to detect an edge.

A. Edge Detection using Gradient Operator based on 1D Image

Consider a 1D image signal with edges in Fig. 3.

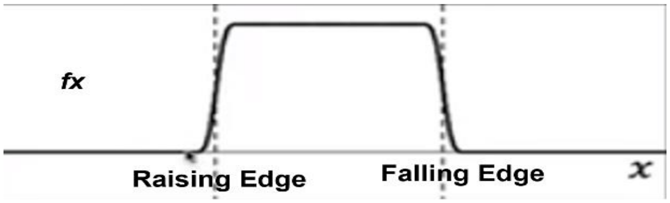


Fig. 3. Image Signal (f(x)).

The first Derivative of the image is computed to determine the intensity changes in the above image, *f(x)*. Ideally, the Derivative of a continuous function represents the amount of change in the function [19]. When dealing with edges, those changes are obtained from the function. When the first Derivative of *f* for *x* is applied to the image in Fig. 3, it is transformed into the image in Fig. 4. The local extrema indicate the edges in the image. Finding the absolute value of the first Derivative obtained the two peaks (local extrema Fig. 4). The location of the peaks indicates where the edges occur, and the height of the peaks expresses the strength of the edges (local maxima Fig. 4).

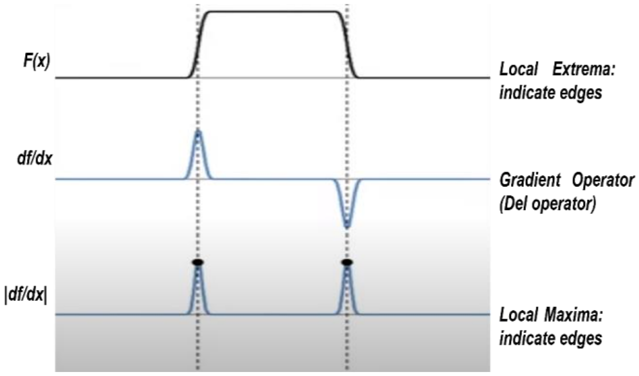


Fig. 4. First Derivative of f with Respect to x.

B. Edge Detection using Gradient Operator based on 2D image

When the above idea is applied to 2 D images, it is referred pure derivatives (partial Derivatives). According to calculus, a partial derivative of a 2D continuous function represents the number of changes along each dimension [19]. The gradient operator (partial Derivative) represents the most rapid change in intensity (see Fig. 5).

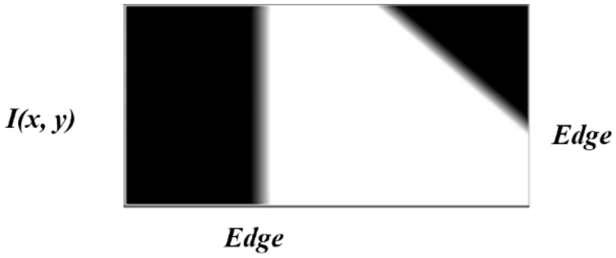
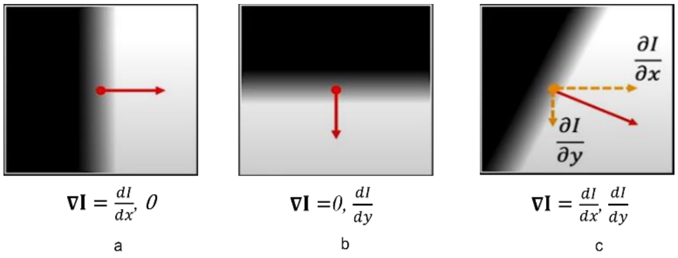


Fig. 5. Edge Detection using 2D Image.

The gradient operator is computed by:

$$\Delta I = \left[\frac{\partial I}{\partial x} + \frac{\partial I}{\partial y}\right] \quad (1)$$

When equation (1) is applied to an image, it produces two numbers; the Derivative of the image *(∂I)* with respect to *x* and *y*. these two numbers (vectors) comprise all information needed to know about the edges. For example:


$\frac{\partial I}{\partial x}$
$\frac{\partial I}{\partial y}$

$\mathbf{\nabla I} = \frac{dI}{dx}, 0$
a

$\mathbf{\nabla I} = 0, \frac{dI}{dy}$
b

$\mathbf{\nabla I} = \frac{dI}{dx}, \frac{dI}{dy}$
c

Fig. 6. Gradient Operator when Applied to an Image.

After applying the del operator, *(ΔI)*, Fig. 6 produces two number vectors along each edge. Image (a) is a vertical edge with a non-zero value for the *x* direction and a zero value for the *y* component. Image (b) is a horizontal edge, with zero for the *x* direction and a non-zero value for the *y* component. Image (c) is a tilted edge and therefore has a non-zero value for both the *x* and *y* direction. From these two numbers at each pixel, we can obtain both the edge strength and the edge orientation. The magnitude equals the sum of squares of the two partial derivatives, and their square root equals 1. While the orientation of the edge is measured with respect to the horizontal axis, Eq. 3.

$$\text{Gradient Magnitude S} = \left|\left|\boldsymbol{\nabla I}\right|\right| = \sqrt{\left(\frac{\partial I}{\partial x} + \frac{\partial I}{\partial y}\right)2} \quad (2)$$

$$\text{Gradient Orientation } \boldsymbol{\theta} = \tan\text{-}1 \left(\frac{\partial I}{\partial x} / \frac{\partial I}{\partial y}\right) \quad (3)$$

When Eq. 2 and Eq. 3 are applied to discrete images, we will have a finite difference approximation for the partial Derivative *(f)* for *x* and *y*. To find the difference in the *x* and *y* direction, we need at least two pixels in each direction, making four pixels in Fig. 7. Assuming the physical distance between the pixels is epsilon. We obtain a finite difference in the image as in Fig. 7.

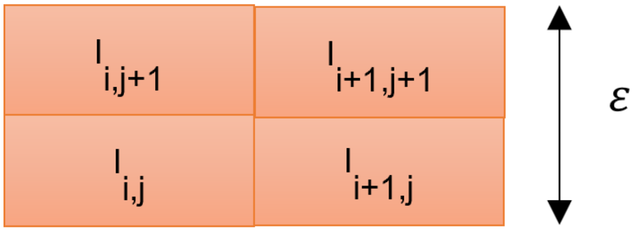


Fig. 7. Four Pixels Images in 1D Image.

$$\frac{dI}{dx} \approx \frac{1}{2\varepsilon}((\text{Ii+1,j+1 - Ii,j+1}) +(\text{ Ii+1,j - Ii,j})) \quad (4)$$

$$\frac{dI}{dy} \approx \frac{1}{2\varepsilon}((\text{Ii+1,j+1 - Ii+1,j}) + (\text{Ii+1,j - Ii,j})) \quad (5)$$

Eq. 4 and Eq. 5 can be applied as a convolution using the filter in Eq. 6 and Eq. 7 to convolve the image for *x* and *y*. when this is done, the values obtained are enough to determine the edge magnitude and orientation.

$$\frac{dI}{dx} \sim \frac{1}{2\varepsilon}\begin{pmatrix} -1 & 1 \\ -1 & 1 \end{pmatrix} \quad (6)$$

$$\frac{dI}{dy} \sim \frac{1}{2\varepsilon}\begin{pmatrix} 1 & 1 \\ -1 & -1 \end{pmatrix} \quad (7)$$

With the gradient approach, a variety of gradient operators have been proposed over the last few decades [6], [20]–[23], which include Robert, Prewitt, Sobel (3x3), and larger Sobel (5x5). The Sobel operator (3x3) has been broadly used across various applications in the literature. The Derivative is found in the diagonal direction to the orthogonal direction of the convolution. The main obstacle of these operators lies mainly between the Robert operator, a small operator, and the larger Sobel. The Robert (2x2) operator with four-pixel values has a perfect localization ability for the edges but is very sensitive to noise because of the fewer pixels to revolve around. Any minor modification with any of those pixels can affect the detection accuracy. Therefore, noise sensitivity does not produce a good edge in a noisy image. On the other hand, the larger Sobel operator (5x5) has very poor localization accuracy. This is because when the operator becomes larger, determining an edge at a particular pixel can be affected by the activities happening in other larger pixels around it, affecting the edge localization. From a practical point of view, getting the orientation and magnitude does not declare an edge to be an edge. It still needs to be localized by thresholding. In this case, two standard thresholds are applied such that for a threshold T, the edge is obtained when Eq. 9 is satisfied:

$$\left|\left|\nabla I\ (x, y)\right|\right| < T \ \text{(not an edge)} \quad (8)$$

$$\left|\left|\nabla I\ (x, y)\right|\right| > T \ \text{(an edge)} \quad (9)$$

By introducing hysteresis-based thresholding, two thresholds are used (T0, T1) to obtain an edge, Eq. 11 and Eq. 12.

$$\left|\left|\nabla I\ (x, y)\right|\right| < T0 \ \text{(not an edge)} \quad (10)$$

$$\left|\left|\nabla I\ (x, y)\right|\right| < \text{T1} \qquad \text{(an edge)} \quad (11)$$

$\text{T0} \leq \left|\left|\nabla I\ (x, y)\right|\right| < \text{T1}$ (an edge if the neighboring pixel is an edge) (12)

An edge pixel is defined by two fundamental characteristics: its edge strength, which is equal to the gradient magnitude, and its edge direction, which is equal to the gradient angle. However, for a discrete function, a gradient is not defined; instead, the gradient may be defined as an ideal continuous image, which is inferred using some specific operators. These operators use a pre-determined convolution mask to detect edges.

C. The Roberts Cross Operators

Roberts (1963) introduced the Roberts Cross operator. He clearly and concisely measures a 2-D spatial gradient on an image. As a result, high spatial frequency zones are emphasized, which typically correspond to edges. The operator's input and output are grayscale images in the most common situation. The estimated absolute magnitude of the spatial gradient of the input image at that point is represented by pixel values at each position in the output. Eq. 13 and Eq. 14 are the 2x2 convolution mask that convolves images in *x* and *y* directions *(Gx, Gy)*. These masks are made such that *Gy* is a 90^0 rotation of *Gx* [11].

Roberts Operators Masks

$$\text{Gx} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad (13)$$

$$\text{Gy} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad (14)$$

Eq. 13 and Eq. 14 can be applied independently to the query image to obtain separate gradient components at each orientation. The absolute value of the combination of the gradient components (Gx, Gy) gives the magnitude at each point in Eq. 16 and the orientation of the gradient in Eq. 17.

$$|G| = \sqrt{\text{G}x^2 + \text{G}y^2} \quad (15)$$

Which can also be computed approximately as:

$$|G| = |Gx| + |Gy| \quad (16)$$

The angle θ of orientation is given by:

$$\theta = \ \textbf{arctan}\left(\frac{G_x}{G_y}\right) - \frac{3\pi}{4} \quad (17)$$

D. Sobel Operator

The Sobel operator introduced by [21] indicates a high-frequency spatial region. The Sobel operator is achieved through a 2D image spatial frequency measurement by converting the image into grayscale and computing the absolute approximate gradient magnitude value at each point.

The Sobel operator is convolved using a (3x3) convolution kernel. With one kernel rotated 90 degrees over the other. Eq. 18 and Eq. 19 show the Robert cross operators.

Sobel Operators Masks

$$\text{Gx} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad (18)$$

$$\text{Gy} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad (19)$$

Sobel kernels can be used to compute separate measurements of the gradient component at each orientation, which are later combined to form the magnitude of the gradient for x and y orientations. The gradient magnitude can be computed using Eq. 20 and Eq. 21, while the orientation is computed using Eq. 22.

$$|\boldsymbol{G}| = \sqrt{\mathbf{G}\boldsymbol{x}^{\mathbf{2}} + \mathbf{G}\boldsymbol{y}^{\mathbf{2}}} \quad (20)$$

It can further be approximated by:

$$|\boldsymbol{G}| = |\boldsymbol{Gx}| + |\boldsymbol{Gy}| \quad (21)$$

The angle of orientation is given by:

$$\boldsymbol{\theta} = \textbf{arctan}\,(\tfrac{G_x}{G_y}) \quad (22)$$

One of the problems of Sobel is using a gaussian smooth to reduce noise, which in turn affects the detection of a good edge. Despite this limitation, Sobel exhibits the quality of edge detection applied to solving various computer vision problems [24].

E. Prewitt Operator

The Prewitt operator introduced by [25] has similar properties to that of Sobel, such as convolution kernels. The kernel for the Prewitt operators are shown in Eq. 23 and Eq. 24. When applied to a noiseless and well-contrasted image, it is a computationally less expensive and faster edge detection method (Pujare et al., 2020). It is a gradient-based edge detection operator, and it has gradient features. Compared to the success of edge detection in complex images, the success of the Prewitt operator is greater than Roberts's operator [6]

$$\text{Gx} = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad (23)$$

$$\text{Gx} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & -1 & -1 \end{bmatrix} \quad (24)$$

The gradient-based operators can be determined using some pre-determined steps known as algorithms. Algorithm 1 is one of the general algorithms used for gradient-based edge detection.

Algorithms 1: Gradient-based edge detection algorithm

```
Define x and y operators(ox and oy)
gx ← convolve(img, ox)
gy ← convolution(im,oy)
maxM ← 0
for pixels |i| in image do
    Mag|i| ← Eqn. 17
    if Mag|i|>maxM then
        maxMag ←Mag|i|
    end if
end for
for pixels i in image do
    if Mag|i| >=thresh x maxMag then
        detected_edges|i| ←1
    else
        detected_edges|i|←0
    end if
end for
return detected_edges
```

IV. Algorithms based on Second Derivatives (Non-gradient-based)

The methods outlined in the previous discussions merely involve filtering the image with various masks without considering the edges' properties or the image's noise. The second Derivative first dealt with the noise problem before going into edge detection. It is based on the concept introduced by Haralick and Marr and Hildreth Algorithms [26], [27]. This idea brings us to the concept of the Laplacian operator.

A. Laplacian Operator

Pierre-Simon de Laplace first applied the Laplace operator to the study of celestial mechanics or the motion of objects in space, and it is named after him. Since then, the Laplace operator has been used to represent many phenomena, including electric potentials, heat and fluid flow diffusion equations, and quantum physics. It's also been requested in discrete space, where it's been employed in image processing and spectral clustering applications [28].

The intensity variations in the sharp zero-crossing of an image f can be calculated using the Laplacian operator ($\boldsymbol{\nabla^2}$). Consider a 1D signal *f(x)* in Fig. 8. The edge is located at the local extrema, which is the first Derivative of the signal $(\partial f/\partial x)$. But in the case of the second Derivative of the image (i.e., the Derivative of an image $(\partial^2\mathrm{f}/\partial^2\mathrm{x})$. The zero-crossing through each peak of the signal indicates an edge. Peaks are not obtained at the edges, but a strong zeros-crossing indicates an edge (Fig. 8).

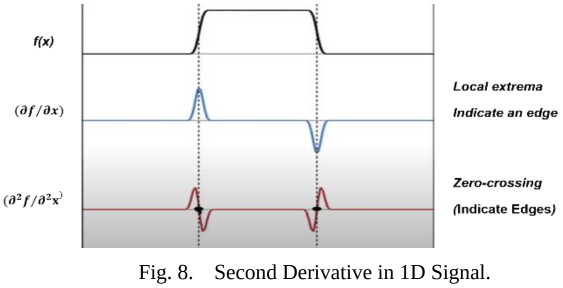
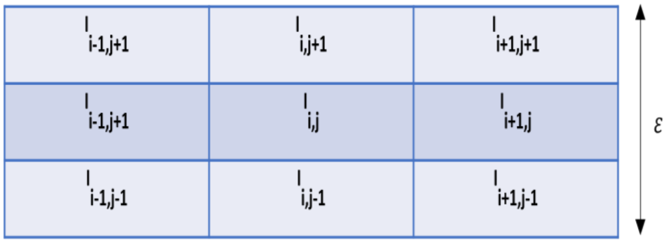


Fig. 8. Second Derivative in 1D Signal.

The zero-crossing is obtained through the Laplacian operator in Eq. 25, known as dell square operator II.

$$\boldsymbol{\nabla^2 I} = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \quad \text{(Dl square II)} \tag{25}$$

When Eq. 25 is applied to an image, the edges obtained are zero-crossing in the Laplacian of the image. The Laplacian operator does not provide the direction of the edges but a sharp zero-crossing which indicates the location of the edge. In discrete images, the second Derivative is in terms of finite difference (difference of the difference). This difference requires at least three pixels (3x3), known as the 3x3 Laplacian operator. Assuming the physical distance between the pixels is epsilon. We obtain a finite difference using the 3x3 image pixels in Fig. 9.



$I_{i-1,j+1}$	$I_{i,j+1}$	$I_{i+1,j+1}$
$I_{i-1,j+1}$	$I_{i,j}$	$I_{i+1,j}$
$I_{i-1,j-1}$	$I_{i,j-1}$	$I_{i+1,j-1}$

Fig. 9. 3x3 Laplacian Operator.

Fig. 9 is a 3x3 Laplacian operator called the "Del square operator II," where the epsilon (ε) denotes the distance between the pixels. It is referred to as the sum of the second Derivative of the image for *x* and the second Derivative of the image for *y*. For example, obtain an output of the Laplacian operator for the center pixel *(Ii+1, j)* from the above *3x3* operator. Eq. 26 and Eq. 27 are derived.

$$\frac{\partial^2 I}{\partial x^2} \approx \frac{1}{\varepsilon^2}\,(\text{Ii-1,j} - 2\text{Ii,j} + \text{Ii+1,j}) \tag{26}$$

$$\frac{\partial^2 I}{\partial y^2} \approx \frac{1}{\varepsilon^2}\,(\text{Ii,j-1} - 2\text{Ii,j} + \text{Ii,j+1}) \tag{27}$$

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \tag{28}$$

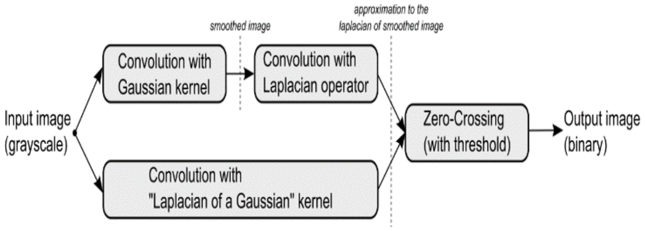
Adding Eq. 26 and Eq. 27 and the Derivative of I with respect to y formulate the convolution mask in Eq. 29:

$$\nabla^2 \approx \frac{1}{\varepsilon^2}\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{29}$$

The convolution mask in Eq. 29 obtained the second Derivative for x and the second Derivative for y. However, the edge can appear in any orientation. Assuming the edge appears in 450^0, the epsilon in that direction *(Ii,j, Ii+1,j+1)* is not yet accounted for. This problem is peculiar to a continuous grid. However, the convolution can be modified on a discrete grid to fit all possible orientations in the image. The modified convolution, which is the most applicable Laplacian operator, is obtained in Eq. 30:

$$\nabla^2 \approx \frac{1}{6\varepsilon^2}\begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix} \tag{30}$$

One of the common algorithms used to implement the Laplacian operator for finding zero-crossing is the Marr-Hildreth algorithm [13]. This method obtains a sharp zero-crossing point by applying a convolution with a gaussian kernel and approximating the Laplacian operator with a 3x3 filter. Presented in Fig. 10 is a general flow diagram for finding zero-crossing, while Algorithm 2 is used for the implementation [13].



smoothed image
approximation to the laplacian of smoothed image
Input image (grayscale)
Convolution with Gaussian kernel
Convolution with Laplacian operator
Convolution with "Laplacian of a Gaussian" kernel
Zero-Crossing (with threshold)
Output image (binary)

Fig. 10. Marr-Hildreth Algorithm;s Block Diagram for Finding Zero-crossing [13].

Algorithms 2: Marr-Hildreth edge detection algorithm

```
Imput required: input img, sigma value σ, size of kernel      k, and
threshold for zero-crossing Tzc
          If kernel size then
              Kernel ⟵ compute-kernel k (n,σ)
              imggm ⟵ convolve (img, k)
              #Define Laplacian operator Laplacian L
              imgl ⟵ convolve(imggs, L)
          else
             log_k ⟵ compute_log_k(n,σ)
             imgl ⟵ convolve(imggs, log_k)
          end if
          maxL ⟵ 0

          for pixels i in images imgl do
             if imgl [i] > maxl then
                  maxL ⟵ imgl [i]
             end if
          end for
          for pixels i in images imgl, except borders do
             for pair (pts1, pts2) of opposite neighbors
     of p in imlap do
                     if          (sign(imgl[pts1])           ≠
     sign(imgl[pts2])) and (|imgl[pts1]|>TZC then
                         detected_edges[i] ⟵ 1
                     else
                         detected_edges[i] ⟵ 0
                      end if
                 end for
          end for
          return edges
```

B. *Canny Edge Detection*

The Canny Edge Detection Algorithm consists of a specific sequence of steps. Smooth the image with a Gaussian filter first. Then, compute the gradient magnitude and orientation by approximating the partial derivatives with finite-difference approximations. The gradient magnitude is then subjected to a non-maximum suppression. Then, the double threshold technique to find and connect edges [20] using Eq. 31.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{31}$$

Where x and y distance from the origin to the horizontal and vertical direction, respectively, σ is the Gaussian factor that determines the level of smoothing. The gradient of the smoothed array G(x,y) is employed to generate the x and y Partial Derivatives. After that, the results of x and y partial are added to obtain the normal gradient. Then non-maximum suppression is applied after determining the edge direction through the use of two thresholding in the Hysteresis [15][20].

Non-maximum suppression: Facilitate the detection of thin layered edges that produces smarter edges. The Canny edge detector employs non-maximum suppression to emphasize the local maxima as edges while suppressing all other ones along with the gradient magnitudes.

Other operators employed the use of a single threshold value to remove edges that do not meet the required edge limit. Canny uses Hysteresis with upper and lower threshold values to suppress the edge that falls below the required threshold limit. If the high threshold is T1, the low threshold is T2, and the gradient magnitude is GM, then GM< T1 are dropped, and GM>T1 are maintained as the edge. However, an edge that falls between T1 and T2, is only maintained if it has a pixel value higher than T2 [11] and is expressed it as:

GM<T1 = drop

GM>T1 = maintain

If T2≤GM<T1

GM = edge if Pixel $GM > T2 \; \forall \; i \in$ I(x, y)

The canny edge detector has the advantage of intensive noise filtering than other detectors. It has three distinct attributes that differentiate it from the other detectors. Table II shows the advantages and disadvantages of various edge detection methods.

Low error rate: The detected edges are refined not to include non-edges. It also ensures that all occurring edges are not missed. This is achieved by Eq. 32.

$$\boldsymbol{SNR} = \frac{\int_{-w}^{w} G(-x) f(x) dx}{n_0 \sqrt{\int_{-w}^{w} f^2(x) dx}} \tag{32}$$

Where f is the filter, G is the edge signal; the denominator is the root-mean-square (RMS) response to noise n(x) only.

Good localization ensures a minimal distance between the actual edge and the detected edge which is expressed in Eq. 33.

$$\boldsymbol{Localization} = \frac{1}{\sqrt{E[X_0^2]}} = \frac{\left|\int_{-w}^{w} G'(-x) f'(x) dx\right|}{n_0 \sqrt{\int_{-w}^{w} f'^2(x) dx}} \tag{33}$$

Single response rate: Each detected edge should maintain a single response in the total edges detected. This is implicit in the first criterion but made explicit about eliminating multiple responses. The first two criteria can be minimized by setting the parameter in Eq. 34 [29].

$$f(x) = G(-x) \tag{34}$$

TABLE II. ADVANTAGES AND DISADVANTAGES OF VARIOUS EDGE DETECTION METHODS

Edge detection methods	Advantages	Disadvantages
An algorithm based on the first Derivative (Robert, Prewitt, and Sobel)	Fast and more accessible in computation Edges are detected along with their orientation.	Sensitive to noise. The inaccurate and unreliable edge detection output
An algorithm based on the second Derivative (LoG)	Due to ease in an approximation of gradient magnitude, the cross-operation detection of edges and their orientation is also simple The characteristics of all directions of the image are fixed. A wide testing area around the pixel is possible	The detection of edges and their orientation degrades the magnitude of the edges and increases the noise Malfunctioning at the corners, curves, and where the grey level intensity function varies
Canny Edge Detector	Better detection in noise conditions.	Do not require zero crossing. Difficult to specify a generic threshold value that works well across all images

V. Experimental Result and Discussion

The edge detection methods discussed in the paper are applied to five sample images taken from the MICC-F220 CMFD evaluation dataset. The result of each algorithm is shown in Fig. 11.

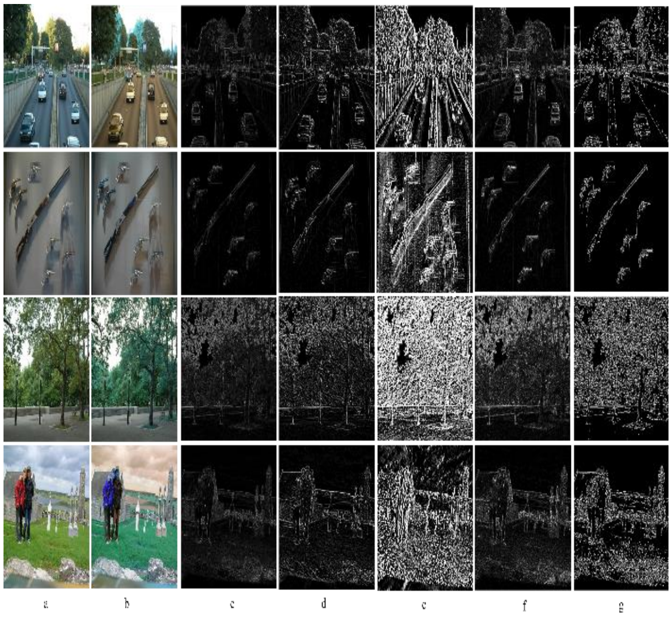


Fig. 11. Comparison of Edge Detection Algorithms using Five Sample Images from MICC-F220 Benchmark Dataset, (a) Original Image (b) Grayscale (c) Robert (d) Prewitt (e) Sobel (f) Laplacian and (g) Canny Edge Detector.

VI. Discussion

Edge detection methods play an important role in object detection. It is used primarily in the pre-processing stage before feature extraction, an essential step in CMFD. Images are converted to grayscale to reduce noise. Edge detection based on first derivatives is generally sensitive to noise and, in most cases, produces unrealistic features.

We have implemented five selected images from MICC-F220 datasets. Roberts, Sobel, Prewitt, Laplacian, and canny edge detectors have been implemented on those images. The result in Fig. 11 shows that the edges in (c, d and e) detected using Robert, Prewitt, and Sobel, are too cluttered and almost lost the important image structures. The image appeared noisy with thick layered unsmoothed edges, mainly the result in (e), the Sobel, the edges appeared two to three times the pixel thickness. Since most of those images appeared to have lost their essential edge information, the results did not show the appropriate edge structure. However, the result can still indicate the geometric parts of the images but does not reveal the thin layer edges, essential for feature extraction in CMFD.

On the other hand, edge detectors based on second derivatives, such as Laplacian and canny edge detectors, use adjustable parameters such as the size of the Gaussian filter and threshold [20]. The edges are smoothly detected on the image, and almost no noise pixels were detected while protecting the important structural properties in the images, as shown in Fig. 11 (f and g). However, the canny method produces smoother and thin edges than the Laplacian. In Fig. 11 (g), we can see a good view of the various image structures. Fewer noise pixels were detected compared to gradient-based edge detectors.

VII. Conclusion

This study demonstrates that the Canny approach can produce equally acceptable edges with smooth continuous pixels and thin edges. Unlike the canny method, the Laplacian method also has better edge features compared to Robert, Prewitt, and Sobel method. These methods cannot generate a smooth and thin edge. However, the Laplacian and Canny algorithms are often susceptible to noise pixels. Sometimes it is impossible to filter a noisy image perfectly. Noisy pixels that are not eliminated will affect the outcome of edge detection. Based on our investigation, we have determined that between the Canny and other edge detection algorithms, Canny edge detection provided a superior response for CMFD images in MICC-F220 datasets.

References

[1] Y. Song and H. Yan, “Image Segmentation Techniques Overview,” AMS 2017 - Asia Model. Symp. 2017 11th Int. Conf. Math. Model. Comput. Simul., pp. 103–107, 2018, doi: 10.1109/AMS.2017.24.

[2] N. Kheradmandi and V. Mehranfar, “A critical review and comparative study on image segmentation-based techniques for pavement crack detection,” Constr. Build. Mater., vol. 321, no. January, p. 126162, 2022, doi: 10.1016/j.conbuildmat.2021.126162.

[3] P. Dhankhar and N. Sahu, “A Review and Research of Edge Detection Techniques for Image Segmentation Related papers A Review and Research of Edge Detection Techniques for Image Segmentation,” IJCSMC J., vol. 2, no. 7, pp. 86–92, 2013.

[4] S. Jeyalaksshmi and S. Prasanna, “A Review of Edge Detection Techniques for Image Segmentation,” Int. J. Data Min. Tech. Appl., vol. 5, no. 2, pp. 140–142, 2016, doi: 10.20894/ijdmta.102.005.002.008.

[5] G. T. Shrivakshan, “A Comparison of various Edge Detection Techniques used in Image Processing,” Int. J. Comput. Sci. Issues, vol. 9, no. 5, pp. 269–276, 2012.

[6] Ş. Öztürk and B. Akdemir, “Comparison of Edge Detection Algorithms for Texture Analysis on Glass Production,” Procedia - Soc. Behav. Sci., vol. 195, pp. 2675–2682, 2015, doi: 10.1016/j.sbspro.2015.06.477.

[7] T. Tahmid and E. Hossain, “Density based smart traffic control system using canny edge detection algorithm for congregating traffic information,” 3rd Int. Conf. Electr. Inf. Commun. Technol. EICT 2017, vol. 2018-Janua, no. December, pp. 1–5, 2018, doi: 10.1109/EICT.2017.8275131.

[8] R. F. Abbas, “A proposed approach to determine the edges in SAR images,” Iraqi J. Sci., vol. 61, no. 1, pp. 185–192, 2020, doi: 10.24996/ijs.2020.61.1.21.

[9] I. Lorencin, N. Anđelić, J. Španjol, and Z. Car, “Using multi-layer perceptron with Laplacian edge detector for bladder cancer diagnosis,” Artif. Intell. Med., vol. 102, no. May 2019, 2020, doi: 10.1016/j.artmed.2019.101746.

[10] J. S. S. Yin, T. T. Swee, A. Bin Yahya, M. T. F. Thye, and J. S. Y. Xian, “Mini Kirsch edge detection and its sharpening effect,” Indones. J. Electr. Eng. Informatics, vol. 9, no. 1, pp. 228–244, 2021, doi: 10.11591/ijeei.v9i1.2597.

[11] A. Pujare, P. Sawant, H. Sharma, and K. Pichhode, “Hardware Implementation of Sobel Edge Detection Algorithm,” ITM Web Conf., vol. 32, p. 03051, 2020, doi: 10.1051/itmconf/20203203051.

[12] P. Niyishaka and C. Bhagvati, “Copy-move forgery detection using image blobs and BRISK feature,” Multimed. Tools Appl., 2020, doi: 10.1007/s11042-020-09225-6.

[13] H. Spontón and J. Cardelino, “A Review of Classic Edge Detectors,” Image Process. Line, vol. 5, pp. 90–123, 2015, doi: 10.5201/ipol.2015.35.

[14] J. Jing, S. Liu, G. Wang, W. Zhang, and C. Sun, “Recent advances on image edge detection: A comprehensive review,” Neurocomputing, vol. 503, pp. 259–271, 2022, doi: 10.1016/j.neucom.2022.06.083.

[15] D. Vikram Mutneja, “Methods of Image Edge Detection: A Review,” J. Electr. Electron. Syst., vol. 04, no. 02, 2015, doi: 10.4172/2332-0796.1000150.

[16] Y. Li, S. Wang, Q. Tian, and X. Ding, “A survey of recent advances in visual feature detection,” Neurocomputing, vol. 149, no. PB, pp. 736–751, 2015, doi: 10.1016/j.neucom.2014.08.003.

[17] C. Lopez-Molina, B. De Baets, and H. Bustince, “Quantitative error measures for edge detection,” Pattern Recognit., vol. 46, no. 4, pp. 1125–1139, 2013, doi: 10.1016/j.patcog.2012.10.027.

[18] G. Papari and N. Petkov, “Edge and line oriented contour detection: State of the art,” Image Vis. Comput., vol. 29, no. 2–3, pp. 79–103, 2011, doi: 10.1016/j.imavis.2010.08.009.

[19] G. X. Ritter and J. N. Wilson, Handbook of Computer Vision Algorithms in Image Algebra. CRC Press, CRC Press LLC, 1996.

[20] Z. Othman, M. Rafiq, and A. Kadir, “Comparison of Canny and Sobel Edge Detection in MRI Images,” pp. 133–136, 2009.

[21] I. E. Sobel, “Camera Models and Machine Perception,” University Microfilms, Inc., Ann Arbor, Michigan THIS, 1970.

[22] N. Nausheen, A. Seal, P. Khanna, and S. Halder, “A FPGA based implementation of Sobel edge detection,” Microprocess. Microsyst., vol. 56, no. September 2016, pp. 84–91, 2018, doi: 10.1016/j.micpro.2017.10.011.

[23] C.-C. Zhang and J.-D. Fang, “Edge Detection Based on Improved Sobel Operator,” Adv. Comput. Sci. Res., vol. 52, no. 16, pp. 129–132, 2016, doi: 10.2991/ceis-16.2016.25.

[24] D. Kim, “Sobel Operator and Canny Edge Detector,” pp. 1–10, 2013, [Online]. Available: http://www.egr.msu.edu/classes/ece480/capstone/fall13/group04/docs/danapp.pdf.

[25] J. M. S. Prewitt, Object enhancement and extraction. Picture processing and Psychopictorics, 1970.

[26] R. M. Haralick, Digital Step Edges from Zero Crossing of Second Directional Derivatives. Morgan Kaufmann Publishers, Inc., 1987.

[27] D. Marr and E. Hildreth, “Theory of edge detection,” Proc. R. Soc. London - Biol. Sci., vol. 207, no. 1167, pp. 187–217, 1980, doi: 10.1098/rspb.1980.0020.

[28] P.-S. Laplace, A Philosophical Essay on Probabilities, vol. 2, no. 38. 1903.

[29] I. E. Igbinosa, “Comparison of Edge Detection Technique in Image Processing Techniques,” Int. J. In. f. Technol. Electr. Eng., vol. 2, no. 1, pp. 25–29, 2013.