Journal of Chemical Health Risks

www.jchr.org

JCHR (2025) 15(5), 2851-2867 | ISSN:2251-6727

Bisection Method-Based Adaptive Thresholding for Robust Edge Detection in Image Processing

Laxmi Kumari Jha[1], Suresh Kumar Sahani[*2]

[1]Faculty of Science, Technology and Engineering

[*2]Faculty of Science, Technology and Engineering

(Received: 25 August 2025 Revised: 27 September 2025 Accepted: 14 October 2025)

KEYWORDS

Bisection, Sobel, Prewitt, Kirsch, Canny, Image processing, Threshold

ABSTRACT:

Thresholding is the process of dividing an image into a collection of tiny numbers known as pixels, which correspond to a physical volume in digital image processing. Image thresholding can enhance image quality by reducing noise and perfecting overall visual clarity. In this environment, different ways are used for image processing, including, Sobel, Prewitt, Kirsch, and canny drivers. But all of them have several limitations which we will try to overcome using bisection system of numerical styles. According to the numerical style theory known as the "bisection system," if a function f(x) is nonstop between a and b, and f(a) and f(b) are of contrary signs, also there exists at least one root bittern a and b. In this paper we will bandy the former styles of thresholding for edge discovery and bisection system for edge discovery in image processing.

Introduction:

1. Definition:

One of the important phases in image processing is edge detection. A solid grasp of the edge detection algorithm is essential for object detection. In image analysis, it is among the most often utilized operations. An edge is the line that separates a thing from its surroundings. In this section, work done in the area of edge detection is reviewed and focus has been made on detecting the problems of the previous work done and also how can we apply bisection method in edge detection in image processing. The most commonly used methods for edge detection include Sobel, Prewitt, and canny operators. Two masks, Sx and Sy, are used in the X and Y directions, respectively, in the Sobel method. These masks are used to perform convolution on the gray image and then obtain edge intensities Ex and Ey in the vertical and horizontal directions. Regretfully, the edge that the Sobel technique detects is typically thicker than the real edge.

Canny method uses the same edge intensity as that of Sobel method to define edge angle as tan-1 Ey / Ex. The Canny method can detect much thinner edges than the Sobel method. However, both Canny and Sobel methods will obtain zero intensity if a line of one-pixel-width passes through the mask and then the incorrect edge called “double edges” on the both sides of this line will be generated. Additionally, the Sobel and Canny methods may yield edge intensity values that are extremely big and outside of a known range. So the threshold value should be based on many heuristic attempts in an unknown range.

The Prewitt approach has a different convolution mask but functions similarly to the Sobel method. The Prewitt technique detects edges both vertically and horizontally using 3x3 convolution masks. But same as Sobel method, it can be sensitive to noise and may not beeffective for detecting edges at angles other than 00,450, 900 and 1350.

Similarly, the bisection method is a straightforward and accurate numerical technique for determining a function's roots inside a certain interval. It involves repeatedly diving the interval in half and selecting the half that contains the root based on a change in sign of the function. Here, we will use this method in image processing to reduce noise and enhance the quality of the image.

2. Literature Review

Edge Detection in Image Processing: An Introduction [1], describes that edge detection is a crucial step in computer vision used to identify boundaries in images by

Journal of Chemical Health Risks

www.jchr.org

JCHR (2025) 15(5), 2851-2867 | ISSN:2251-6727

detecting changes in intensity, color, or texture. Techniques like Canny, Sobel, and LoG are applied for tasks such as image segmentation, feature extraction, object detection, recognition, and motion analysis. Tools like Roboflow help streamline experimentation by allowing easy image uploads, annotation, and dataset generation for faster results.

A Comprehensive Analysis of Image Edge Detection Techniques (Mohd. Aquib Ansari, Diksha Kurchaniya, Manish Dixit,2017)[2] . This paper compared various edge detection techniques and found that second-order derivatives like Canny and LoG perform better than first-order methods such as Sobel, Prewitt, and Roberts. Canny is more dependable at detecting both inner and outer edges with strong noise resistance than LoG, notwithstanding LoG's effectiveness. Sobel performs well mainly for continuous outer boundaries, and future work should focus on designing new filters to reduce noise and enhance image quality.

Recent advances on image edge detection: A comprehensive review (Junfeng Jing, Shenjuan Liu, Gang Wang, Weichuan Zhang,2022)[3], highlighted existing edge detection methods in detail. It begins by introducing the definition and properties of edges, then classifies and explains both hand-crafted and machine learning-based methods. Finally, it summarizes commonly used datasets and evaluation criteria for image edge detection.

Research on Adaptive Edge Detection Method of Part Images Using Selective Processing (Yaohe Li, Long Jin, Min Lu, Youtang Mo, Weiguang Zheng, Dongyuan Ge, Yindi Bai, 2024)[4], evaluates the limitations of existing edge detection methods by proposing a selectively processed adaptive edge detection approach for part images. The method uses a selective mixed filtering algorithm to remove noise while preserving detail, then applies a four-parameter adaptive model to improve accuracy, adaptability, and detection of both strong and faint edges. Experimental results show that the method outperforms classical approaches under various noise conditions, offering stable, efficient, and flexible edge detection, with future work focusing on multi-channel images and FPGA-based real-time applications for industrial part inspection.

A comprehensive study of edge detection for image processing applications (P Ganesan, G. Sajiv, 2017)[5], explains a comprehensive study of edge detection methods, highlighting their role in identifying discontinuities in pixel intensity to enhance image analysis. Edge detection reduces data while preserving essential structural details, but distinguishing edges from noise remains challenging due to their similar high-frequency characteristics. The study compares various edge detection techniques, emphasizing that their effectiveness is application-specific and no single algorithm is universally suitable for all image types.

Deep learning-based edge detection for random natural images (Kanjia Muntarina, Rafid Mostafiz, Sumaita Binte Shorif, Mohammad Shorif Uddin, 2025)[6-10], highlights the difficulties with gradient-based approaches in terms of accuracy, connectivity, and uniformity while introducing both conventional and deep learning-based edge detection models. While models like HED, FCN, UNet, and DeepEdge show weaknesses in noise handling and subtle edge detection, the MultiResEdge model demonstrates superior robustness and effectiveness on natural images. The findings provide insights for optimizing edge detection in real-time applications, with future work planned on expanding datasets, incorporating GAN-generated synthetic data, and exploring transformer-based models for improved performance. This research is motivated by the works of Sahani, et al; 2022, Sahani and Sah, 2022, Munjal, et al 2024, and so on (see [11-66]).

3. Objective:

This study aims to use bisection method for edge detection in image processing for enhancing the quality of image and reducing noise. For that the previous methods such as Sobel method, Canny method and Prewitt method were also effective. This paper explores and explains the limitations of Sobel's, Canny's, and Prewitt's method in edge detection as well as we will discuss bisection method of numerical methods for edge detection and image processing. This research mainly focuses on how we can implement bisection method in image processing. A key focus will be on ensuring the image quality and reducing the noise.

4. Methodology:

Regarding future technology, image processing is one of the most researched fields in computer graphics. Different methods are used to enhance the quality of

Journal of Chemical Health Risks
www.jchr.org
JCHR (2025) 15(5), 2851-2867 | ISSN:2251-6727

image and reduce the noise. In this context, Sobel, Canny, and Prewitt's method are highly used. This paper includes all of these three methods along with their examples and limitations. This paper also explores utilizing bisection method in edge detection in image processing. There is also an example showing images by using all these methods. This paper also compares all these methods in order to make image processing more effective.

5. Result

Problem:

Traditional methods of edge processing are effective for edge processing but they are more resistant to noise, they do have directional limitations. Also they produce thicker edges, making it difficult to identify true boundaries accurately.

We shall start by talking about Sobel's approach in this context. One of the most widely used edge detectors is the Sobel. Its computations are quite cheap because it is based on convolving the image with a tiny, separable, integer-valued filter in both horizontal and vertical directions. It makes use of two 3x3 convolution kernels to estimate the gradient of the image intensity. This gradient's magnitude draws attention to the existence of edges.

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Here is a python implementation of Sobel method.

```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt

# Open the image
img = np.array(Image.open('girl.jpg')).astype(np.uint8)

# Apply gray scale
gray_img = np.round(0.299 * img[:, :, 0] +
                    0.587 * img[:, :, 1] +
                    0.114 * img[:, :, 2]).astype(np.uint8)

# Sobel Operator
h, w = gray_img.shape
# define filters
horizontal = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])  # s2
vertical = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]])  # s1
```

Journal of Chemical Health Risks
www.jchr.org
JCHR (2025) 15(5), 2851-2867 | ISSN:2251-6727

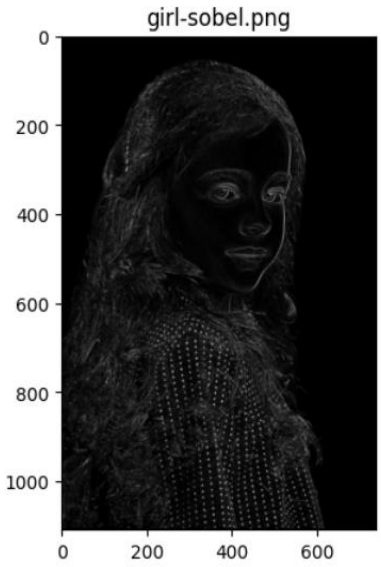
```
# define images with 0s
newhorizontalImage = np.zeros((h, w))
newverticalImage = np.zeros((h, w))
newgradientImage = np.zeros((h, w))

# offset by 1
for i in range(1, h - 1):
    for j in range(1, w - 1):
        horizontalGrad = (horizontal[0, 0] * gray_img[i - 1, j - 1]) + \
                         (horizontal[0, 1] * gray_img[i - 1, j]) + \
                         (horizontal[0, 2] * gray_img[i - 1, j + 1]) + \
                         (horizontal[1, 0] * gray_img[i, j - 1]) + \
                         (horizontal[1, 1] * gray_img[i, j]) + \
                         (horizontal[1, 2] * gray_img[i, j + 1]) + \
                         (horizontal[2, 0] * gray_img[i + 1, j - 1]) + \
                         (horizontal[2, 1] * gray_img[i + 1, j]) + \
                         (horizontal[2, 2] * gray_img[i + 1, j + 1])

        newhorizontalImage[i - 1, j - 1] = abs(horizontalGrad)

        verticalGrad = (vertical[0, 0] * gray_img[i - 1, j - 1]) + \
                       (vertical[0, 1] * gray_img[i - 1, j]) + \
                       (vertical[0, 2] * gray_img[i - 1, j + 1]) + \
                       (vertical[1, 0] * gray_img[i, j - 1]) + \
                       (vertical[1, 1] * gray_img[i, j]) + \
                       (vertical[1, 2] * gray_img[i, j + 1]) + \
                       (vertical[2, 0] * gray_img[i + 1, j - 1]) + \
                       (vertical[2, 1] * gray_img[i + 1, j]) + \
                       (vertical[2, 2] * gray_img[i + 1, j + 1])

        newverticalImage[i - 1, j - 1] = abs(verticalGrad)
```

Journal of Chemical Health Risks
www.jchr.org
JCHR (2025) 15(5), 2851-2867 | ISSN:2251-6727

```
        # Edge Magnitude
        mag = np.sqrt(pow(horizontalGrad, 2.0) + pow(verticalGrad, 2.0))
        newgradientImage[i - 1, j - 1] = mag

plt.figure()
plt.title(girl-sobel.png')
plt.imsave('girl-sobel.png', newgradientImage, cmap='gray', format='png')
plt.imshow(newgradientImage, cmap='gray')
```

In the code above we first apply gray scale to the image, then we define our masks for horizontal and vertical pass then we loop over the image and calculate the gradients.

The result after processing this program is given below:



Although the Sobel's method is effective, it have certain limitations. It is gradient-based and sensitive to noise, which can lead to the detection of spurious edges. They tend to produce thick or double edges, making accurate edge localization challenging. While less likely to be confused by noise than some other methods, they can still result in fragmented edge detection, particularly with complex images.

The Canny method is the next most used edge detection technique. A multi-stage technique called Canny Edge Detection seeks to locate edges as accurately and noise-resistantly as possible. It is a cutting-edge edge detector due to its intricate design. ItIn order to decrease noise, it first applies a Gaussian filter on the image. After that, the gradient is determined using the Sobel or Prewitt operators. The Prewitt or Sobel operators are then used to calculate the gradient. After that it extract edge points i.e. Non-maximum suppression. Then after linking and thresholding occurs. The first two stages are simple, but take note that the second step also involves calculating the gradients' orientation: "theta = arctan (Gy/Gx)". Gx and Gy stand for the gradient's x and y directions, respectively. Let's now discuss non-maximum suppression and its effects. Using the gradient strengths

Journal of Chemical Health Risks
www.jchr.org
JCHR (2025) 15(5), 2851-2867 | ISSN:2251-6727

and edge directions that were previously determined, we are attempting to tie the edge direction to a path that can be followed along the edges in this phase. Each pixel point can be in one of four directions. To determine whether the gradient reaches its maximum here, we search in all directions. The values of pixels that are perpendicular to the edge direction are contrasted with each other. If their value is lower than the edge pixel's, they are suppressed. Since this step will result in thin, fractured edges that need to be fixed, let's go on to the next one.

The broken lines created in the previous step can be connected using hysteresis. Iterating through the pixels and determining if the current pixel is an edge is how this is accomplished. If it is an edge, look for edges in the surrounding region. We designate them as edge pixels if they point in the same direction. We also use both a high and low threshold. An edge is indicated if the pixel count exceeds the lower threshold. Strong edge pixels are then chosen from among those pixels that are both higher than the high threshold and higher than the lower threshold. When there are no more changes made to the image, we are done.

Now let’s convert our paragraph in coded form to see the actual results:

```
import cv2
import matplotlib.pyplot as plt
from google.colab import files
import numpy as np

# Upload image
uploaded = files.upload()

# Get the uploaded file name
file_name = list(uploaded.keys())[0]

# Read image in grayscale
img = cv2.imdecode(np.frombuffer(uploaded[file_name], np.uint8), cv2.IMREAD_GRAYSCALE)

# Apply Canny edge detection
edges = cv2.Canny(img, 100, 200)

# Save the edge-detected image
output_file = file_name.rsplit(".", 1)[0] + "_edges.png"
cv2.imwrite(output_file, edges)

# Display result
plt.imshow(edges, cmap='gray')
```

Journal of Chemical Health Risks
www.jchr.org
JCHR (2025) 15(5), 2851-2867 | ISSN:2251-6727

```
plt.title("Edge Detection - Canny")
plt.axis('off')
plt.show()

print(f"Edge image saved as: {output_file}")
```

Now, let's see the actual results:


```
Edge image saved as: girl_edges.png
```

As we can observe the resulting binary image obtained by Canny method is thin, well-defined and pixels are classified as strong edges, weak edges or non-edges using two thresholds and hysteresis. But due to its multistep process, it is computationally intensive than other methods, making it less suitable for real-time applications. Also Canny can sometimes produce incomplete or fragmen edges, leading to disjoint contours. Although it includes noise reduction steps, Canny can still be sensitive to noise, especially in challenging images.

Another popular method of edge detection is Prewitt. Similar to the Sobel operator, the Prewitt operator is used to identify both horizontal and vertical edges in pictures. This operator, in contrast to the Sobel, does not give any weight to the pixels that are nearer the mask's center.

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

The mask detects an edge anywhere there is an abrupt change in pixel brightness. Since differentiation is defined as the change in pixel intensities, it can be utilized to compute the edge. In the graph form of Prewitt-mask's result, the edge is denoted by the local maxima or minima.

We will illustrate this by using a python program:

Journal of Chemical Health Risks
www.jchr.org
JCHR (2025) 15(5), 2851-2867 | ISSN:2251-6727

```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
from google.colab import files

# Upload image
uploaded = files.upload()

# Get the filename of the uploaded image
filename = next(iter(uploaded))

# Open the image
img = np.array(Image.open(filename)).astype(np.uint8)

# Convert to grayscale
gray_img = np.round(0.299 * img[:, :, 0] +
                    0.587 * img[:, :, 1] +
                    0.114 * img[:, :, 2]).astype(np.uint8)

# Prewitt Operator
h, w = gray_img.shape

# Define filters
horizontal = np.array([[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]])
vertical = np.array([[-1, -1, -1], [0, 0, 0], [1, 1, 1]])

# Initialize gradient image
newgradientImage = np.zeros((h, w))

# Compute gradients
for i in range(1, h - 1):
    for j in range(1, w - 1):
        horizontalGrad = np.sum(horizontal * gray_img[i-1:i+2, j-1:j+2])
```

Journal of Chemical Health Risks
www.jchr.org
JCHR (2025) 15(5), 2851-2867 | ISSN:2251-6727

```
        verticalGrad = np.sum(vertical * gray_img[i-1:i+2, j-1:j+2])
        mag = np.sqrt(horizontalGrad**2 + verticalGrad**2)
        newgradientImage[i, j] = mag

# Save and display the result
plt.figure(figsize=(8, 8))
plt.title('Prewitt Edge Detection')
plt.imsave('prewitt_edge.png', newgradientImage, cmap='gray', format='png')
plt.imshow(newgradientImage, cmap='gray')
plt.axis('off')
plt.show()

print("Edge-detected image saved as 'prewitt_edge.png'")
```

Now, let's the outcome of this code:

As we can see the Prewitt method detects edges by calculating the gradient of the image's intensity, producing a gradient magnitude image that shows edge strength. This method is relatively inexpensive computationally, making it suitable for real-time applications. However, the Perwitt operator is sensitive to noise, which can lead to the detection of many false edges. Especially for high-frequency image fluctuations, it generates a quite imprecise approximation of the gradient of the image. This method is prone to creating false edges due to its sensitivity to noise and its realtively basic nature compared to other advances operators.

Solve:

As we have studied three methods of edge detection in image processing. We can conclude that these methods are very efficient and produces well-defined images. But they do have certain limitations and they are difficult to obtain. We will try to do edge detection using bisection

Journal of Chemical Health Risks

www.jchr.org

JCHR (2025) 15(5), 2851-2867 | ISSN:2251-6727

method that will be more easier and will overcome the problems of other methods.

The Bisection method is a simplest and most reliable of iterative methods mainly for solution of non linear equations. It asserts that there is at least one root between a and b if a function f(x) is continuous between a and b and f(a) and f(b) have opposite signs. Let f(a) be negative and f(b) be positive for definiteness. The root thus falls between a and b, and x_o = (a+b)/2 can be used to approximate its value. If $f(x_o) = 0$, we conclude that x_o is a root of the equation f(x) = 0. Otherwise, the root lies either between xo and b, or between x_o and a depending on whether $f(x_o)$ is negative or positive. This new interval, whose length is |b-a| / 2, is designated as [a1,b1]. This is once again divided at x1, and the new interval will be precisely half as long as the old one. Until the most recent interval, which contains the root, is as small as required, let's say E, we continue this process. At the end of the n^{th} step, the new interval will be $[a_n,b_n]$ of length |b-a| /2^n, as it is evident that the interval width is cut in half at each step.

We then have,

$\frac{|b-a|}{2^n} \leq E$,

Which gives on simplification

$n \geq \frac{\log_e(\frac{|b-a|}{E})}{\log_e 2}$

This equation gives the number of iterations required to achieve accuracy E.

Hence, this method will find the most accurate edges and our image will be more enhanced and noise free. Since, there are large number of iterations that will help to find out the closest point of images, there will be very less chance of error and also each and every edge can be shown in binary image. It is also easier to understand and implement. Since it is less sensitive to function's behavior, and it does not require the derivate of function. It also doesn't require any extra computational tools.

We will check this out using a program that is given below:

```
# Edge Detection using Bisection Method for Adaptive Thresholding
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
from google.colab import files

# Upload image
uploaded = files.upload()
image_path = list(uploaded.keys())[0]

# Open the image
img = np.array(Image.open(image_path)).astype(np.uint8)

# Convert to grayscale
gray = np.round(0.299 * img[:, :, 0] + 0.587 * img[:, :, 1] + 0.114 * img[:, :,
2]).astype(np.uint8)

# Define gradient operator (Sobel)
```

Journal of Chemical Health Risks
www.jchr.org
JCHR (2025) 15(5), 2851-2867 | ISSN:2251-6727

```
def compute_gradient(image):
    Kx = np.array([[1, 0, -1], [2, 0, -2], [1, 0, -1]])
    Ky = np.array([[1, 2, 1], [0, 0, 0], [-1, -2, -1]])
    Gx = np.zeros_like(image, dtype=float)
    Gy = np.zeros_like(image, dtype=float)

    h, w = image.shape
    for i in range(1, h-1):
        for j in range(1, w-1):
            region = image[i-1:i+2, j-1:j+2]
            Gx[i, j] = np.sum(Kx * region)
            Gy[i, j] = np.sum(Ky * region)
    G = np.sqrt(Gx**2 + Gy**2)
    G = (G / G.max()) * 255  # Normalize
    return G.astype(np.uint8)

gradient = compute_gradient(gray)

# Threshold function for bisection
def threshold_function(T, grad):
    # Fraction of pixels above threshold minus 0.5 (we aim for 50%)
    fraction = np.sum(grad >= T) / grad.size
    return fraction - 0.5

# Bisection method
def bisection_method(f, a, b, tol=1e-3, max_iter=100):
    fa = f(a)
    fb = f(b)
    if fa * fb > 0:
        raise ValueError("Bisection method fails. f(a) and f(b) must have opposite
signs.")
    for _ in range(max_iter):
        c = (a + b) / 2
        fc = f(c)
```

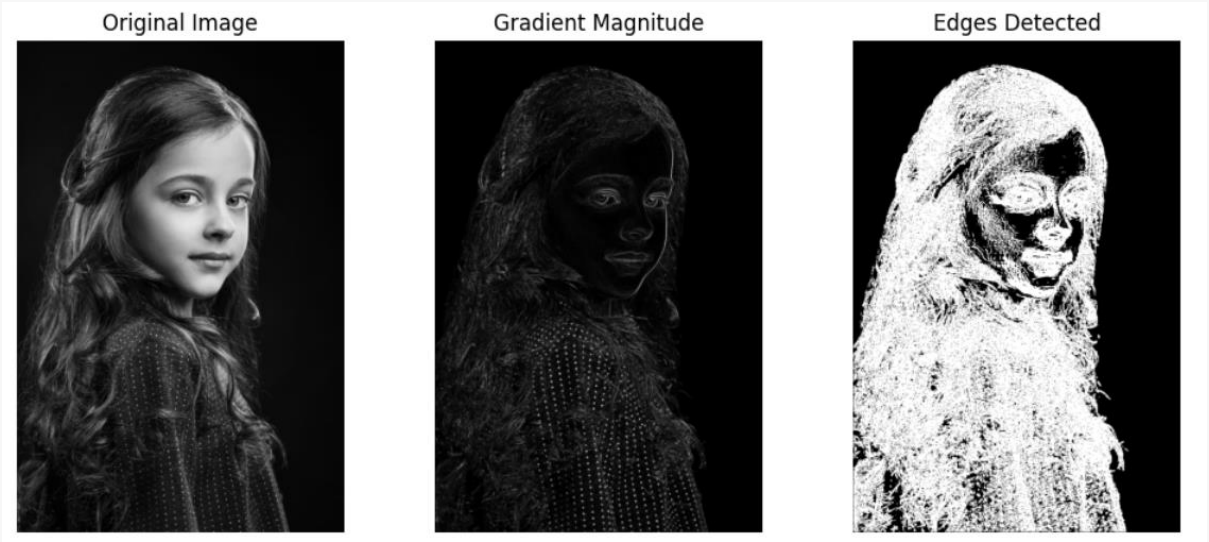
Journal of Chemical Health Risks
www.jchr.org
JCHR (2025) 15(5), 2851-2867 | ISSN:2251-6727

```
        if abs(fc) < tol:
            return c
        if fa * fc < 0:
            b = c
            fb = fc
        else:
            a = c
            fa = fc
    return (a + b) / 2

# Find adaptive threshold
threshold = bisection_method(lambda T: threshold_function(T, gradient), 0, 255)
print(f"Adaptive threshold found: {threshold:.2f}")

# Apply threshold to get edges
edges = np.zeros_like(gradient)
edges[gradient >= threshold] = 255

# Display results
plt.figure(figsize=(12,5))
plt.subplot(1,3,1)
plt.title("Original Image")
plt.imshow(img)
plt.axis('off')

plt.subplot(1,3,2)
plt.title("Gradient Magnitude")
plt.imshow(gradient, cmap='gray')
plt.axis('off')

plt.subplot(1,3,3)
plt.title("Edges Detected")
plt.imshow(edges, cmap='gray')
```

Journal of Chemical Health Risks
www.jchr.org
JCHR (2025) 15(5), 2851-2867 | ISSN:2251-6727

```
plt.axis('off')
plt.show()
```

Here the code will first convert the image into grayscale and then compute the gradient using Sobel operator. It then uses bisection method to find a threshold such that about 50% of gradient pixels are considered edges. Hence more edges can be shown in binary image. The given code shows the original, gradient, and edge-detected images that is given below:



As we can observe the binary image obtained using bisection method has strong edges (like hair outlines, eyes, face counter) and are mostly white. Similarly, the weak edges (like smooth skin regions) are mostly black. The edge detection adapts to image contrast instead of using a fixed threshold. The thickness of edge is usually 1-2 pixels wide, depending on gradient intensity. Unlike Sobel and Prewitt that need a manual threshold to convert gradient magnitude into a binary edge image and Canny that uses double thresholds, often chosen manually or heuristically, in bisection method the threshold is calculated automatically from the image data. Bisection method searches for the root of a function that measures how many pixels exceed a candidate threshold. This makes the robust to different lighting conditions or different images. Sobel / Prewitt with fixed thresholds might fail on very bright or dark images. Hence, using bisection method in edge detection in image processing provides several benefits over older methods.

6. Conclusion

Hence, it can be concluded that the Sobel, Canny and Prewitt operators are produces very high quality and enhanced images but they do have certain limitations that can be overcome by using bisection method. Also bisection method is easy to understand and implement. It is an iterative process and finds the closest points to the edges so it is also more accurate than other methods. While Sobel, Canny, and Prewitt requires a manual threshold, bisection calculated the threshold value automatically from the image data. Bisection method combines high-quality edge computation with automatic thresholding, overcoming the major limitations of classical edge detection. We can still use Sobel, Prewitt, or even Canny's gradient magnitude, but use bisection for threshold selection. If the function is continuous and changes sign, the bisection method is easy to use and converges rapidly.

References:

1. Kaushal, M., Singh, A., Singh, B.,(2010). Adaptive Thresholding for Edge Detection in Gray Scale Images International Journal of Engineering Science and Technology Vol. 2(6), 2010, 2077-2082
2. https://www.geeksforgeeks.org/computer-vision/image-thresholding-techniques-in-computer-vision/

Journal of Chemical Health Risks

www.jchr.org

JCHR (2025) 15(5), 2851-2867 | ISSN:2251-6727

3. https://en.wikipedia.org/wiki/Bisection_method
4. Li, Y., Jin, L., Liu, M., Mo, Y., Zheng, W., Ge, D., Bai, Y., Research on Adaptive Edge Detection Method of Part Images Using Selective Processing **https://doi.org/10.3390/pr12102271**
5. Gaur, S., B.c., Vajpai, J., Mehta, S., Adaptive Local Thresholding for Edge Detection
6. Li, H., Xu, K., Innovative adaptive edge detection for noisy images using wavelet and Gaussian method, 10.1038/s41598-025-86860-9
7. Tsankashvili, N., Comparing Edge Detection Methods,
8. Ansari, M., A., Kurchaniya, D., Dixit, M., (2017).A Comprehensive Analysis fo Image Edge Detection Techniques International Journal of Multimedia and Ubiquitous Engineering 12(11):1-12.10.14257/ijmue.2017.12.11.01
9. Jing, J., Liu, S., Wang, G., Zhang, W., (2022).Recent advances on image edge detection : A comprehensive review September 2022Neurocomputing 503:259-271. 10.1016/j.neucom.2022.06.083
10. https://encord.com/blog/image-thresholding-image-processing/
11. Sahani, S. K., and K. Sahani,(2018). Study and Analysis of Dynamical Models of Plant Growth Analysis in Calculus, Int. J. of Engineering, Pure and Applied Sciences, vol. 3, No.1, March 2018, 30-35.
12. Sahani, S. K., and D. Jha, (2022). Study and Investigation on Real Life Application of First Ordered Differential Equation, Published: The Mathematics Education ISSN 0047-6269,Volume: LVI,No.1,March2022,pp.18-27.
13. Sahani, S.K., and K.S. Prasad, (2022). Some New Applications of Calculus to Non-linear Sciences, Published in: Applied Science Periodical ISSN 0972-5504,Volume: XXIV, No. 4, November 2022, pp. 1-9.Online Link: https://internationaljournalsiwan.com/applied, science-volume-XXIV-4-nov-2022.php DOI No.: https://doi.org/10.5281/zenodo.7865875, ORCID Link: https://orcid.org/0009-0008-5249-8441,Google Scholar Link:https://scholar.google.com/citations?user=BRweiDcAAAAJ&hl=en, Refereed and Peer-Reviewed Quarterly Periodical.
14. Sahani, S. K., and K.S. Prasad,(2022).Some New Applications of Calculus to Non-linear Sciences, Published in: Applied Science Periodical ISSN 0972-5504,Volume: XXIV, No. 4, November 2022, pp. 1-9.Online Link: https://internationaljournalsiwan.com/applied, science-volume-XXIV-4-nov-2022.php DOI No.: https://doi.org/10.5281/zenodo.7865875, ORCID Link: https://orcid.org/0009-0008-5249-8441,Google Scholar Link:https://scholar.google.com/citations?user=BRweiDcAAAAJ&hl=en, Refereed and Peer-Reviewed Quarterly Periodical.
15. Sahani, S.K., et al., (2023). Evaluation of International Complexity of Mathematical Programming Problems, Korea Review of International Studies, Vol.16, Issue 46, 51-55, May 2023
16. Sahani, S. K., et al., (2023). Some new approach on ellipsoid algorithm for non-linear programming, Manager-The British journal of Administrative Management, Vol.59, Issue163 30-37, June 2023.
17. Jha, A., Sahani, S. K., Jha, A., and Sahani, K. (2023). From Equations to Insights: Navigating the Canvas of Tumor Growth Dynamics, MASALIQ Jurnal Pendidikan dan Sains, Vol.3, No.6,1246-1264, November 2023 https://doi.org/10.58578/masaliq.v3i6.1983.
18. Sharma, P., Sahani, S.K., Sahani, K.S., and Sharma, K. (2023).Modeling Planetary and Stellar Motion Using Differential Equations, ARZUSIN Journal Manajeman Pendidikan Dasar, Vol.3, Issue 6, 769-782, December 2023, https://ejournal.yasin-alsys.org/index.php/arzusin
19. Sahani, S. K., et al. (2024). Analytical Framework, Differential Equations in Aerospace Engineering, ALSYSTECH Journal of Education Technology, Vol.2, Issue 1, 13-30, January 2024.
20. Sahani, S.K., et al. (2024). Deteriorates with A Certain Studies on Mathematical Modeling; An Optimization Oriented the Replacement of Items Whose Efficiency Time, MIKAILALSYS Journal of Multidisciplinary, Vol.2, Issue 1, 1-12, April 2024.
21. Jha, A., Sahani, S. K., Jha, A., and Sahani, K. (2023). Journey to the Cosmos; Navigating Stellor Evolution with Differnetial Equations, Alifmatika;

Journal of Chemical Health Risks

www.jchr.org

JCHR (2025) 15(5), 2851-2867 | ISSN:2251-6727

Jurnal Pendidikan Dan Pembelajaran Matematika, Vol.5, Issue 2, 282-297, December 2023.

22. Sahani, S. K., et al., Assessment of the Manufacturing Quality of Metalic Surface Finishes in Artificial Intelligence, IGI Global Publishing Tomorrow's Research Today, DOI:10.4018/979-8-3693-1347-3.ch016.
23. Sah, S., Sahani, S.K., Sahani, K., & Sharma, D. K. (2024). Some real Life Application of Derivatives in Economics. Mikailalsys Journal of Advanced Engineering International, 1(2), 120-131. https://doi.org/10.58578/mjaei.v1i2.2954.
24. Sah, S., Sahani, S. K., Sahani, K., & Sharma, D. K. (2024). Some real Life Application of Derivatives in Economics. Mikailalsys Journal of Advanced Engineering International, 1(2), 120-131. https://doi.org/10.58578/mjaei.v1i2.2954.
25. Sahani, S. K., Prasad, N. K., & Shachi, K. (2024). Biochemical Adaptations to Prolonged Starvation in Freshwater Catfish: Gonadal Glycogen Dynamics. Formosa Journal of Multidisciplinary Research, 3(7), 2479–2488. https://doi.org/10.55927/fjmr.v3i7.9714
26. Sahani, S. K., Prasad, N. K., & Shachi, K. (2024). Biochemical Adaptations to Prolonged Starvation in Freshwater Catfish: Gonadal Glycogen Dynamics. Formosa Journal of Multidisciplinary Research, 3(7), 2479–2488. https://doi.org/10.55927/fjmr.v3i7.9714
27. Sahani, S. K., Prasad, N. K., & Shachi, K. (2024). Biochemical Adaptations to Prolonged Starvation in Freshwater Catfish: Gonadal Glycogen Dynamics. Formosa Journal of Multidisciplinary Research, 3(7), 2479–2488. https://doi.org/10.55927/fjmr.v3i7.9714
28. Das, R., Mahato, A.K., Sahani, S. K., Mahto, S.K.(2024). STUDY ON DIOPHANTINE EQUATION AND ITS REAL-LIFE APPLICATIONS. International Journal of Innovative Studies, 9(1), 58-66. https://ijois.com/index.php/ijoisjournal/article/view/155
29. Sahani, S. K., et al, (2025) An Investigation on the Stationary Shop Queuing Model; an Overview of Mathematical Modeling, Panamerican Mathematical Journal, Vol.35. No.1(S), 335-343.
30. Munjal, A., Kaur, R., Rathour, L.,Sahani, S.K.,Mishra, L.N.,(2022). PYTHON PROGRAMMING CODES FOR SOLUTION OF SYSTEM OF LINEAR EQUATIONS. Engineering (ICMASE 2022) 4-7 July 2022, Technical University of Civil Engineering Bucharest, Romania, 76. AIP Conf. Proc. 3005, 020035 (2024) https://doi.org/10.1063/5.0210813
31. Munjal, A., Kaur, R., Rathour, L.,Sahani, S.K.,Mishra, L.N.,(2022). PYTHON PROGRAMMING CODES FOR SOLUTION OF SYSTEM OF LINEAR EQUATIONS. Engineering (ICMASE 2022) 4-7 July 2022, Technical University of Civil Engineering Bucharest, Romania, 76. AIP Conf. Proc. 3005, 020035 (2024) https://doi.org/10.1063/5.0210813
32. Munjal, A., Kaur, R., Rathour, L.,Sahani, S.K.,Mishra, L.N.,(2022). PYTHON PROGRAMMING CODES FOR SOLUTION OF SYSTEM OF LINEAR EQUATIONS. Engineering (ICMASE 2022) 4-7 July 2022, Technical University of Civil Engineering Bucharest, Romania, 76. AIP Conf. Proc. 3005, 020035 (2024) https://doi.org/10.1063/5.0210813
33. Sahani, S. K., Sah, B. K., Sahani, K., Das, R., & Mahato, A. K. (2024). Mathematical input-output analysis for economic impact assessment: A case study on local government's bridge construction project. Alifmatika: Jurnal Pendidikan Dan Pembelajaran Matematika, 6(2), 279–292.
34. Sahani, S. K., Sah, B.K., Sahani, K. (2023), Reliability-Centered Maintenance (RCM) in Cement Manufacturing Plants, Advances in Nonlinear Variational Inequalities, Vol. 26, No.1, 2023, 16-25. DOI: https://doi.org/10.52783/anvi.v26.4224
35. Sahani, S. K., Sah, B.K., Sahani, K. (2023), Reliability-Centered Maintenance (RCM) in Cement Manufacturing Plants, Advances in Nonlinear Variational Inequalities, Vol. 26, No.1, 2023, 16-25. DOI: https://doi.org/10.52783/anvi.v26.4224
36. Sahani, S. K., Oruganti, S.K., Kumar, K.S., and Karna, S.K.. (2025). "Reliability Assessment of a Plywood Production Facility Utilizing Laplace Transform and Runge- Kutta Fourth-Order Differential Equations: Overview of Industrial Plant". Metallurgical and Materials Engineering 31 (4):417-423. https://doi.org/10.63278/1453.

Journal of Chemical Health Risks

www.jchr.org

JCHR (2025) 15(5), 2851-2867 | ISSN:2251-6727

37. Sahani, S. K. Oruganti, S.K., & K. Sathish Kumar. (2025). Research Paper Optimization of Maintenance Strategies for Cement Manufacturing Plants: Optimization of Maintenance Strategies for Cement Manufacturing Plants. SGS - Engineering & Sciences, 1(1). Retrieved from https://spast.org/techrep/article/view/5234.
38. Mahato, A. K., Das, R., & Sahani, S. K. (2025). Simulation of Realistic Motion in Computer Graphics Using Runge-Kutta Methods. African Multidisciplinary Journal of Sciences and Artificial Intelligence, 2(2), 325-342. https://doi.org/10.58578/amjsai.v2i2.5681.
39. Sahani, S. K., (2022). Evaluation of Plastic Pipe Performance in Industrial Engineering Utilizing Laplace Transform and the Fourth-Order Runge-Kutta Method, Fuel Cells Bulletin, 2022, 12,1-7. https://doi.org/10.52710/fcb.210
40. Sahani, S. K. (2023). Evaluation and Analysis of a Rice Production Facility Utilizing Laplace Transform and Runge-Kutta Fourth-Order Method in Reliability Theory, Fuel Cells Bulletin, 2023. 1, 15-21.
https://doi.org/10.52710/fcb.212
41. Sahani, S. K. Sah, B.K. (2023). A Comprehensive Analysis of Cement Manufacturing Plants Using the Chapman-Kolmogorov Differential Equation. The International Journal of Multiphysics, 17(4), 538 - 544. https://doi.org/10.52783/ijm.v17.1856
42. Sahani, S. K., Mandal, R.H. (2022). Modeling Population Growth in a Rural Area Using the Fourth-Order Runge-Kutta Method. The International Journal of Multiphysics, 16(4), 453 - 461. https://doi.org/10.52783/ijm.v16.1857
43. Sahani, S. K. and Sah, D.K., (2023). Hybrid Analytical-Numerical Techniques for Machine Learning Optimization: Integrating Laplace Transform and Runge-Kutta Fourth-Order Methods, Review of Contemporary Philosophy, 22, 1, 6854-6860.
DOI: https://doi.org/10.52783/rcp.1165.
44. Sahani, S. K., Sah, B.K., (2023). Performance Assessment of Industrial Plants Using Laplace Transform and Chapman-Kolmogorov Differential Equations. Journal of Computational Analysis and Applications (JoCAAA), 31(1), 1308–1317. Retrieved from https://www.eudoxuspress.com/index.php/pub/article/view/2913
45. Sahani, S. K., Sah, D.K. (2022). Euler's Method: A Numerical Model for the Decomposition of Organic Waste .Journal of Computational Analysis and Applications (JoCAAA), 30(1), 857–868. Retrieved from https://eudoxuspress.com/index.php/pub/article/view/2924
46. Sahani, S. K., Sah, B.K. (2021). A Dynamical Systems Perspective on Echo Chamber Polarization: Integrating Laplace and Runge-Kutta Methods, Communications on Applied Nonlinear Analysis, Vol. 28, 4, 105-113. https://doi.org/10.52783/cana.v28.5574
47. Sahani, S. K. et al., (2022).A Comprehensive Study on Predicting Numerical Integration Errors using Machine Learning Approaches, Letters in High Energy Physics, 96-103. DOI: https://doi.org/10.52783/lhep.2022.1465
48. Sahani, S. K., Sah, B.K. (2024).An In-Depth Stability and Convergence Analysis of the Runge-Kutta 4th Order Method for Nonlinear Ordinary Differential Equations, Panamerican Mathematical Journal,34,2, 300-308. https://doi.org/10.52783/pmj.v34.i2.5583
49. Sahani, S. K., Sah, D.K., Mandal, R.H., (2022). A Runge-Kutta numerical approach to modeling population dynamics with age-structured differential equations, Advances in Nonlinear Variational Inequalities, Vol 25 No. 1, 109-116. https://doi.org/10.52783/anvi.v25.5625.
50. Sahani, S. K., Sah, D.K., Mandal, R.H., (2022).Analyzing the Impact of Birth and Death Rates on Population Growth Using Modified Euler Method: A Case Study of Dhanusha District Panamerican Mathematical Journal, Vol 32 No. 3, 9-21. https://doi.org/10.52783/pmj.v32.i3.5624
51. Sah, D.K., Sahani, S.K., (2021). Numerical Analysis of Greenhouse Gas Impact on Heat Retention in Atmosphere, Journal of Chemical Health Risks, 11, 4, 503-509. https://doi.org/10.52783/jchr.v11.i4.8641
52. Sahani, S. K., Mandal, R.H., (2023). Application of Logistic Regression and Numerical Methods for Automated Bot Detection in Social Media Networks, Linguistic and Philosophical Investigations, Vol22, 2,192-206.

Journal of Chemical Health Risks

www.jchr.org

JCHR (2025) 15(5), 2851-2867 | ISSN:2251-6727

53. Sahani, S. K., Mandal, R.H., (2024). Predictive Analysis of Urban Population Growth Using Least Squares Regression. The International Journal of Multiphysics, 18(4), 1215 - 1226. https://doi.org/10.52783/ijm.v18.1868
54. Sahani, S. K., Mandal, R.H., (2025). A Numerical Method for Analyzing the Effects of Migration on Regional Population Change. The International Journal of Multiphysics, 19(1), 980 - 992. https://doi.org/10.52783/ijm.v19.1867
55. Sahani, S. K., Shah, B. (2024). A Statistical Evaluation of Customer Feedback on Social Media and Its Role in Product Development, Linguistic and Philosophical Investigations, Vol23, 1, 3038-3057
56. Sahani, S. K., & Sah, B. K. (2025). Optimization of Cultural Education Program Using Numerical Techniques Journal of Posthumanism, 5(6), 4265–4280. https://doi.org/10.63332/joph.v5i6.2600
57. Sahani, S. K. (2024). Big Data Learning Analytics & Optimization: Algorithms, Challenges, and Applications, Analysis and Metaphysics, Vol23 (1), 2024pp. 912–924.
58. Sahani, S. K., & Sah, B. K. (2025). Optimization of Cultural Education Program Using Numerical Techniques Journal of Posthumanism, 5(6), 4265–4280. https://doi.org/10.63332/joph.v5i6.2600
59. Sahani, S.K. (2024). Big Data Learning Analytics & Optimization: Algorithms, Challenges, and Applications, Analysis and Metaphysics, Vol23 (1), 2024pp. 912–924.
60. Sahani, S.K. (2023). Application of Numerical Methods in Structural Health Monitoring Using IoT Sensors, J. Electrical Systems 19-1(2023): 194-207. https://doi.org/10.52783/jes.8941
61. Sahani, S.K. (2021). Differential Equation on Astrophysics: A Fundamental Approach to Understanding Cosmic Structures and Their Dynamic Evolution, Letters in High Energy Physics, Vol. 2021, 1-13. DOI: https://doi.org/10.52783/lhep.2021.1468
62. Sahani, S.K. (2024). AI-Enhanced Finite Element Method (FEM) for Structural Analysis, J. Electrical Systems 20, 1, 661-676. https://doi.org/10.52783/jes.8946
63. Sahani, S. K., & Karna, S. K. (2025). Simulation of the Clash between Cultural Values in Heterogeneous Society using Numerical Methods. Journal of Posthumanism, 4(3), 543–559. https://doi.org/10.63332/joph.v4i3.2686
64. Sahani, S. K., & Karna, S. K. (2025). Program for Cultural Education Optimization by the Application of Numerical Methods. Journal of Posthumanism, 5(1), 1611–1627. https://doi.org/10.63332/joph.v5i1.2687
65. Sahani, S. K., Sah, M. K., Agarwal, S. K., Pareek, V., Pareek, V., & Singh, V. V. (2025). Numerical Simulation of Cultural Value Conflict in Multicultural Societies. Journal of Posthumanism, 5(3), 1815–1831 . https://doi.org/10.63332/joph.v5i3.2651
66. Sahani, S.K., and Sah, D.K. (2020). Numerical Optimization for Efficient Design and Scheduling of Public Transport Network. *International Journal of Multiphysics,*14(4), 426-435. https://doi.org/10.52783/ijm.v14.1860