IMAGE PROCESSING: EDGE DETECTION (USING SOBEL FILTER)

J COMPONENT PROJECT REPORT FOR THE COURSE

BECE301L – DIGITAL SIGNAL PROCESSING

BY

Diya Singh (21BEC1086)

Georgie Thomas (21BEC1093)

Ayush Kumar (21BEC1536)

SUBMITTED TO

Dr. RAMESH R

Associate Professor, VIT Chennai

SCHOOL OF ELECTRONICS ENGINEERING

VELLORE INSTITUE OF TECHNOLOGY

CHENNAI – 600127

JULY 2023

CERTIFICATE

This is to certify that the Project work titled "**Project title**" is being submitted by **Student Name (Register No)** for the course B**ECE301L – DIGITAL SIGNAL PROCESSING**, is a record of bonafide work done under my guidance. The contents of this project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University

Prof. RAMESH R

Associate Professor

School of Electronics Engineering (SENSE)

VIT University, Chennai

Chennai – 600 127

ABSTRACT

Edge detection in image processing is a process of identifying and highlighting the boundaries or edges of objects within an image. In image processing and computer vision, edge detection is an essential step for various tasks, such as object recognition, image segmentation, and future extraction.

The edges in an image correspond to significant changes in intensity or color. Edge detection algorithms analyze the variations in pixel values and locate areas where these variations are large. These areas often represent transitions between different regions or objects in the image. There are several techniques for edge processing such as:

- Gradient-based methods
- Laplacian-based methods
- Canny edge detection
- Edge detection with machine learning

In this project, we focus on the gradient based operator-Sobel operator, which applies convolution masks to estimate the gradient in the horizontal and vertical directions.

Sobel method of detection:

It creates image emphasizing edges, for the edges to be noticed clearly. Out of the other different types of operators/ filters Sobel is a better operator. It looks for strong changes on first derivative of the image. It is based on the shadows or brightness around each vertices to find changes/disruptions.

It performs a 2-D spatial gradient measurement on an image and so emphasizes regions of high spatial frequency.

It has a wide variety of uses in an array of different industries across the globe.

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, Dr**. Ramesh R,** Associate Professor, School of Electronics Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to **Dr. Susan Elias,** Dean of the School of Electronics Engineering, VIT Chennai, for extending the facilities of the School towards our project and for his unstinting support.

We express our thanks to our Head of the Department **Dr. Mohanaprasad. K for** his support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

TABLE OF CONTENTS

CHAPTER NO.	TITLE		PAGE NO.
	ABSTRACT		**III**
	ACKNOWLEDGMENT		**IV**
	LIST OF FIGURES		**VI**
	LIST OF TABLES		**VII**
1	**INTRODUCTION**		**1**
	1.1	OBJECTIVES	**1**
	1.2	SCOPE	**1-2**
2	**DESIGN/ IMPLEMENTATION**		**2-9**
	2.1	INTRODUCTION	**3-7**
	2.2	DESIGN APPROACH	**7**
	2.3	OVERVIEW OF SOFTWARE	**7-9**
	2.4	SUMMARY	**9**
3	**RESULT AND ANALYSIS/TESTING**		**10-11**
	3.1	WORK DONE	**10-11**
	3.2	ANALYSIS OF RESULTS	**11**
4	**CONCLUSION AND FUTURE ENHANCEMENT**		**12**
	4.1	CONCLUSION	**12**
	4.1	FUTURE ENHANCEMENT	**12**
5	**APPENDIX**		**13-14**
6	**REFERENCES**		**15**

LIST OF FIGURES

Figure No.	Title	Page No.
2.1	INPUT IMAGE	3
2.2	SOBEL MATRIX IN HORIZOTAL AXIS	4
2.3	SOBEL MATRIX IN VERTICAL AXIS	5
2.4	CONVOLUTION OPERATION	6
2.4	CONVOLUTION OPERATION WITH ANOTHER EXAMPLE	7
2.5	OUTPUT IMAGE	7
2.6	BLOCK DIAGRAM	9
3.1	GRAY-SCALE INPUT IMAGE	10
3.2	RGB INPUT IMAGE	10
3.3	OUTPUT IMAGE-1	10
3.4	OUTPUT IMAGE-2	10
3.5	COMPARED FILTER OUTPUT IMAGE	11

LIST OF TABLES

Table No..	Title	Page No.
2.1	SOBEL MATRIX	3
2.2	EXAMPLE IMAGE MATRIX	4
2.3	MATRIX TO BE CALCULATED	5
2.4	MATRIX TO BE CALCULATED	5

CHAPTER 1
INTRODUCTION

1.1 PURPOSE

The aim of the project is to emphasize on the Sobel method of edge detection and comparing it with other methods by providing an input image of either a RGB image or a grayscale image and obtaining an edged image highlighting the slope of change of gradient.

1.2 SCOPE

The scope of edge detection is quite broad and encompasses various fields and applications. Here are some areas where edge detection plays a crucial role:

1. Computer Vision: Edge detection is a fundamental step in computer vision tasks such as object recognition, image segmentation, shape analysis, and tracking. By identifying edges, it becomes easier to distinguish objects and extract meaningful features for further analysis.

2. Image Processing: Edge detection is extensively used in image enhancement, where it helps to enhance or emphasize important features in an image. It can be used for edge sharpening, contrast enhancement, noise reduction, and image restoration.

3. Medical Imaging: Edge detection is vital in medical image analysis and diagnostics. It assists in the segmentation of anatomical structures, tumor detection, boundary delineation, and feature extraction for automated diagnosis and treatment planning.

4. Robotics and Autonomous Systems: Edge detection is essential in robotics and autonomous systems for perception and environment understanding. It enables robots to detect objects, navigate obstacles, and map their surroundings. Edge detection also plays a role in robot vision systems for object grasping and manipulation.

5. Industrial Inspection: In manufacturing and quality control, edge detection helps identify defects, measure dimensions, and inspect the integrity of products. It is commonly used in tasks such as surface inspection, PCB inspection, defect detection in materials, and part recognition.

6. Video Processing: Edge detection is applicable in video analysis and surveillance systems. It aids in moving object detection, activity recognition, video summarization, and tracking of objects over time.

7. Artistic Rendering: Edge detection techniques have been used in computer graphics and image synthesis to create artistic effects. They can be used for non-photorealistic rendering (NPR), edge-based stylization, and image abstraction.
8. Augmented Reality (AR) and Virtual Reality (VR): Edge detection can assist in object tracking, pose estimation, and depth estimation, which are important for realistic and accurate rendering of virtual objects in AR and VR applications.
The scope of edge detection extends to many other fields, including geospatial analysis, autonomous vehicles, fingerprint recognition, document analysis, and more. Its wide range of applications showcases its significance in extracting valuable information from images and improving various aspects of visual data analysis and understanding.

CHAPTER 2

DESIGN AND IMPLEMENTATION

2.1 INTRODUCTION

Detection/classification of objects in an image is no easy task for a processor to perform. An image may have many complexities involved in it. For example, take case of slight variation in brightness or gradient of the image, detecting that edge is a difficult task. But then, we must remind ourselves that the processor sees only a 3-dimensional matrix containing integers between 0-255. For it to discover patterns from a matrix of numbers is not an easy task. Yet the visualization of it is made easier using padding the matrix of the image.

Let's take an example image and discuss about it.

FIGURE 2.1: Grayscale image (Image credit: Google)

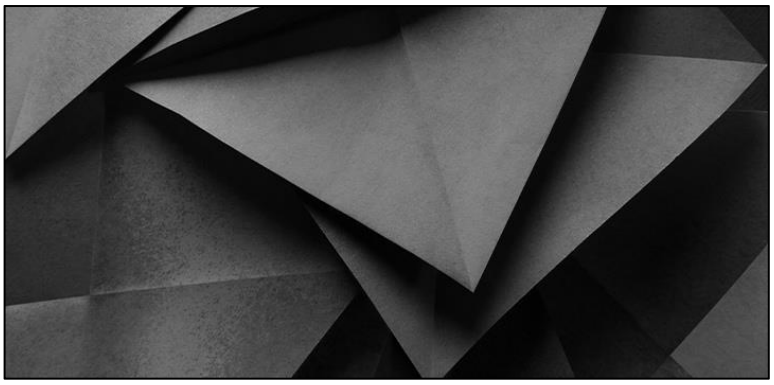


TABLE 2.1: SOBEL TABLE

X-Directional Kernel

-1	0	1
-2	0	2
-1	0	1

Y-Directional Kernel

-1	0	1
-2	0	2
-1	0	1

Convolution:

The operator consists of 3×3 convolution kernels (One in X-direction and one in Y-direction). The kernel or mask in edge detection is convoluted with the input image matrix. The kernel slides over the image, performing a dot product between its values and the corresponding pixel values in the image. This process generates a new output image, which highlights edges or other features of interest.

In the Sobel method, two separate convolution masks are used: one for estimating the gradient in the horizontal (Gx) direction and another for the vertical (Gy) direction. These masks have the values as shown in Table 2.1:

To perform convolution, the Sobel masks are placed over the image, aligned with a target pixel. The center of the kernel is positioned on the target pixel, and element-wise multiplication is performed between the kernel and the corresponding pixel values in the image region covered by the kernel. The resulting values are then summed to obtain a single value for that target pixel in the output image.

Let's take an example to demonstrate the convolution process using the Sobel masks. Consider a grayscale image and its pixel values:

Image:

Table 2.2: Example image matrix

20	50	30	10
40	80	60	20
50	100	90	30
30	60	40	10

To calculate the gradient in the horizontal (Gx) direction for the pixel at (2,2), we place the Gx mask centered on that pixel and perform element-wise multiplication:

Fig 2.2: Gx Table

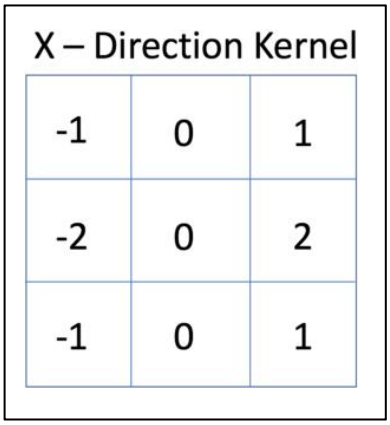


Table 2.3: Matrix to be calculated

100	90	30
60	40	10
60	40	10

Performing element-wise multiplication and summing the results, we get:

(-1 * 100) + (0 * 90) + (1 * 30) +

(-2 * 60) + (0 * 40) + (2 * 10) +

(-1 * 60) + (0 * 40) + (1 * 10) = -100

This value (-100) represents the gradient in the horizontal direction for the pixel at (2,2).

Similarly, we can calculate the gradient in the vertical (Gy) direction for the same pixel by placing the Gy mask centered on that pixel and performing element-wise multiplication:

Fig 2.3: Gy Table

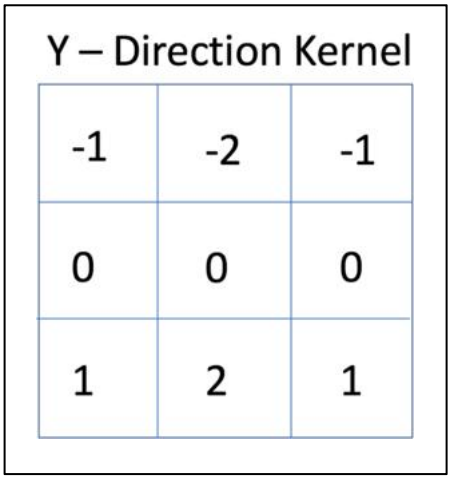


Table 2.4: Matrix to be calculated

100	90	30
60	40	10
60	40	10

Summing the results, we get:

(-1 * 100) + (-2 * 90) + (-1 * 30) +

(0 * 60) + (0 * 40) + (0 * 10) +

(1 * 60) + (2 * 40) + (1 * 10) = -210

This value (-210) represents the gradient in the vertical direction for the pixel at (2,2).
The same process is repeated for every pixel in the image to obtain the complete gradient maps in both the horizontal and vertical directions. These gradient maps can then be used to compute the gradient magnitude and direction, as described earlier in the Sobel method explanation.
By performing convolution with the Sobel masks, the Sobel method estimates the gradient of image intensities in the x and y directions, enabling the detection of edges based on rapid intensity changes.

FIGURE 2.4: CONVOLUTION OPERATION WITH ANOTHER EXAMPLE

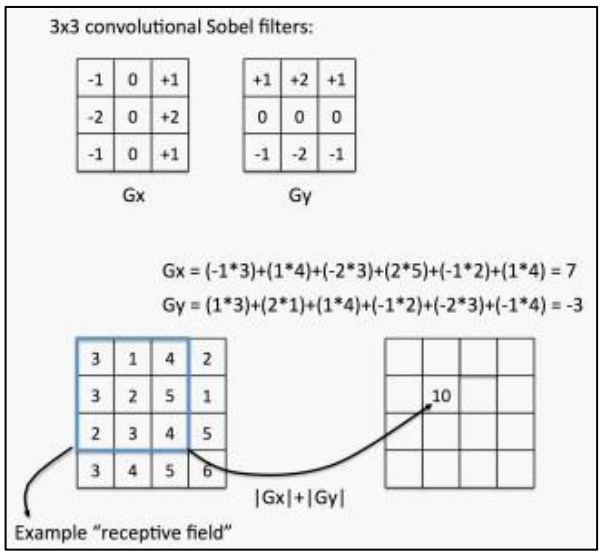
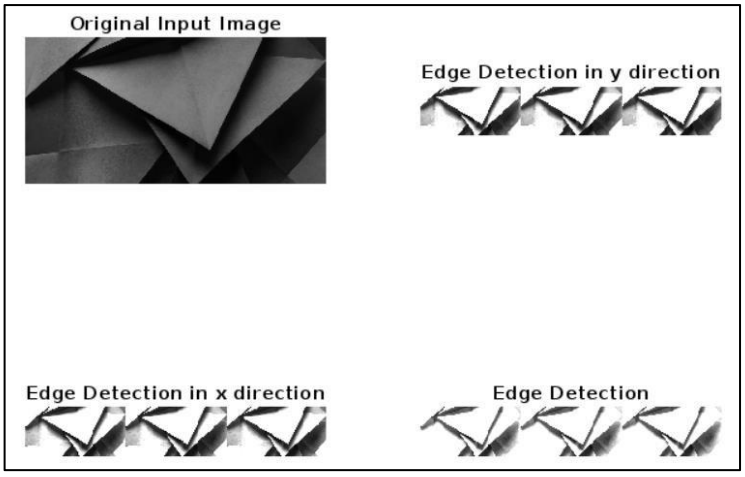


FIGURE 2.5: OUTPUT IMAGE



2.2 PROPOSED SYSTEM

In this project we aim to emphasize the significance of edge detection (specifically through Sobel method). We'll detect the edges through Sobel method code (not using in-built functions) and verify it with other filters (using in-built functions).

2.3 OVERVEIW OF SOFTWARE

MATLAB is a high-level programming language and software environment that is widely used for numerical computation, data analysis, visualization, and algorithm development. Originally developed by MathWorks, MATLAB stands for "MATrix LABoratory" and was primarily designed for matrix manipulations. However, it has evolved into a comprehensive tool for various applications in engineering, science, finance, and many other fields.

Here is an overview of the key features and capabilities of MATLAB:

1. Programming Language: MATLAB provides a powerful and expressive programming language that supports procedural, object-oriented, and functional programming paradigms. It has built-in data structures and operators for working with arrays, matrices, strings, and other data types.

2. Numeric Computation: MATLAB excels at performing mathematical and numerical computations. It offers a vast collection of built-in mathematical functions and libraries for linear algebra, optimization, statistics, signal processing, image processing, and more. MATLAB also supports vectorized operations, allowing for efficient computations on large datasets.

3. Data Analysis and Visualization: MATLAB provides tools for data analysis, manipulation, and visualization. It allows users to import, preprocess, and analyze data from various sources, such as files, databases, and external devices. MATLAB's plotting and visualization capabilities enable the creation of 2D and 3D graphs, charts, images, and animations.

4. Algorithm Development: MATLAB is widely used for developing and prototyping algorithms. It provides a platform for designing, testing, and refining algorithms before deployment. MATLAB supports the creation of user-defined functions, scripts, and custom toolboxes, allowing for modular and reusable code.

5. Application Development: MATLAB allows the creation of standalone applications and user interfaces using its graphical development environment, called MATLAB App Designer. This feature enables the development of interactive applications with buttons, sliders, plots, and other components, without the need for low-level programming.

6. Integration and Deployment: MATLAB can be integrated with other programming languages, tools, and hardware devices. It supports interoperability with languages like C/C++, Java, and Python, allowing for code integration and external function calls. MATLAB also provides options for generating standalone executables, deploying applications to the web, and integrating with hardware for data acquisition and control.

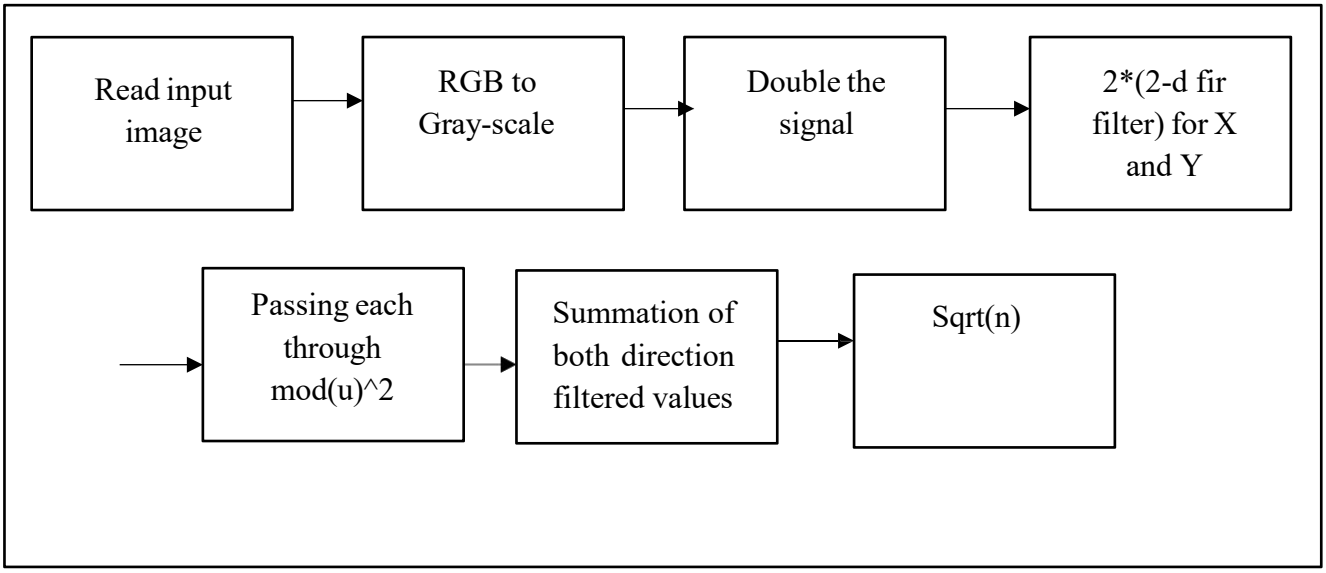
7. Community and Ecosystem: MATLAB benefits from a large and active community of users and developers. It offers extensive documentation, tutorials, and examples to help users get started and learn new features. Additionally, there is a wide range of third-party toolboxes and add-ons available, expanding MATLAB's capabilities for specialized domains and applications.

MATLAB is widely adopted in academia, industry, and research institutions for a variety of tasks, including scientific computing, data analysis, control system design, image and signal processing, machine learning, and more. It provides a versatile environment for exploring, analyzing, and solving complex problems efficiently.

2.4 SUMMARY

With a basic introduction to the concept, the design approach was built on and the project was proposed. The major curveballs and barriers in terms of compiling the various edge detection systems are done in MATLAB Software. We understood and compared the various different types pf MATLAB filters.

FIG 2.6: BLOCK DIAGRAM OF EDGE DETECTION ALGORITHM



CHAPTER 3

RESULTS AND ANALYSIS TESTING

In edge detection, matrix multiplication is an essential step, it is the key building block for gradient solving. First, RGB-grayscale is checked for the input image and it is then proceeded to further steps.

3.1 WORK DONE

Understanding the primary objective is the most crucial stage when solving any task. Our aim is to identify the edges in an input image (using Sobel detector). For better precision and analysis the output is compared with Canny, Prewitt, Log and Roberts for the same input using in-built functions.

FIG 3.1: GRAY-SCALE INPUT IMAGE

FIGURE 3.2: RGB INPUT IMAGE

FIG 3.3: OUTPUT IMAGE

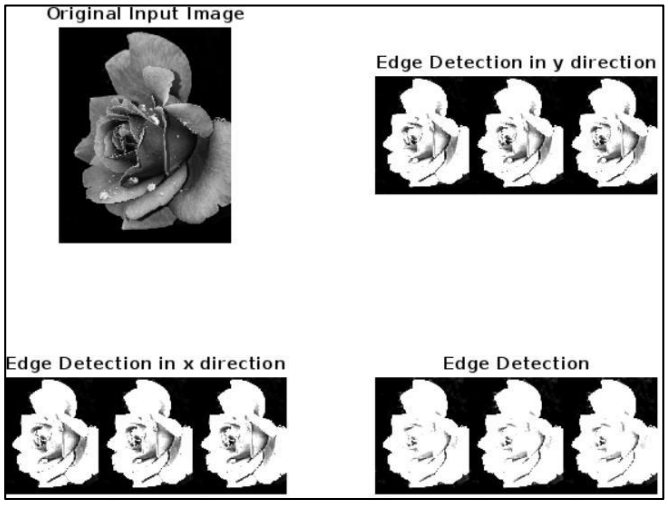


FIG 3.4: OUTPUT IMAGE

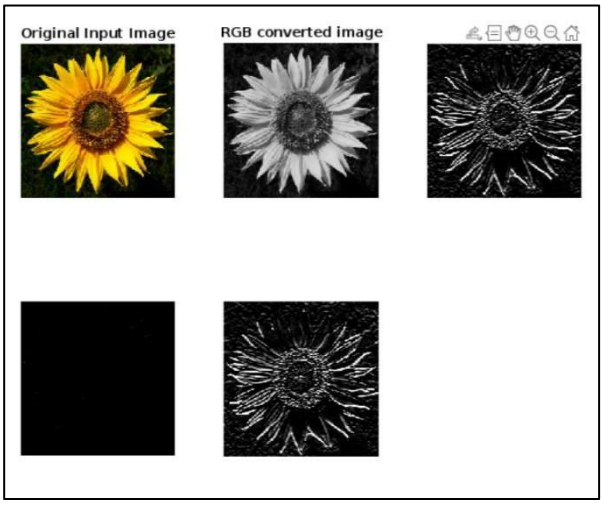
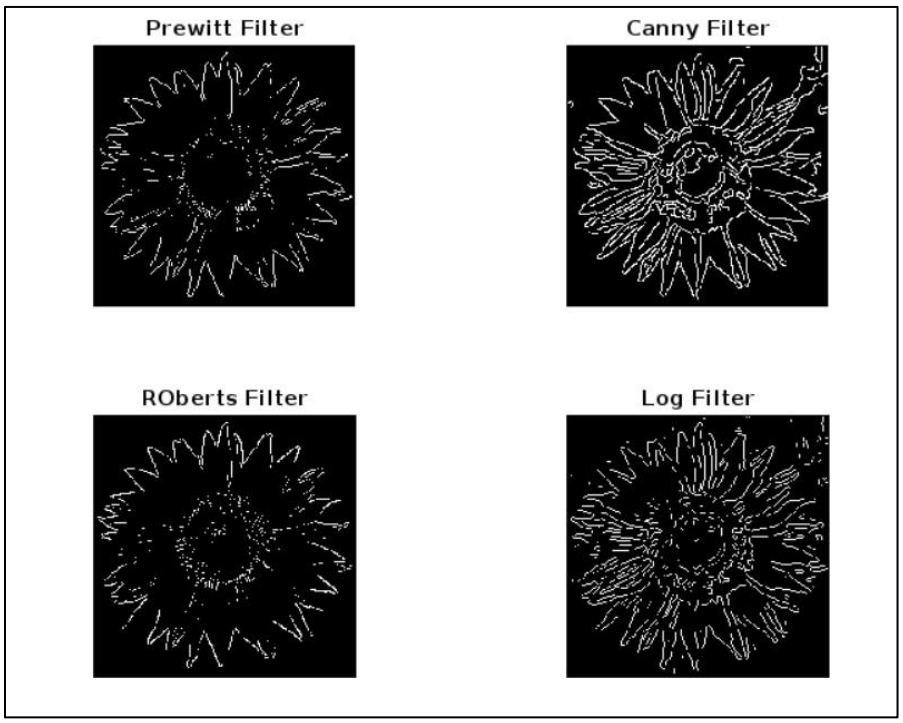


FIG 3.5: COMPARED FILTER OUTPUT IMAGE



3.2 ANALYSIS OF RESULT

On comparison with other filters it is seen that Canny is more precise with respect to other filters. It is also noted that the base of Canny filter is Sobel filter. Canny is sobel detected image but 1 pixel wider.

CHAPTER 4
CONCLUSION AND FUTURE ENHANCEMENT

4.1 CONCLUSION

Through this project, we have obtained the detected the enhancement of edges through the Sobel edge detection method and also compared the obtained results with the results obtained through the other filters and found out why Sobel is a preferred method for edge detection due to its efficiency and smooth edge line detection. However, when we compared with other filters, we saw Canny is more precise. It is to be noted that Sobel is base for canny wherein the only difference canny gives is that it is 1 pixel wider than Sobel, thus the précised output

4.2 FUTURE ENHANCEMENT

Today edge detection methods like Sobel have immense applications in many industries like in the medical industry like for bone fracture detection and also in used in crime scene detection like fingerprinting and also in the monitoring of self driving cars through motion edge detection.

Edge detection is an important field in image processing, machine vision and computer vision, particularly in the areas of feature detection and feature extraction, which aims at identifying points in a digital image at which the image brightness changes sharply or, more formally, has discontinuities. The future scope will be to study the reasons for this in detail and improve this hybrid method so that it combines the advantages of all of these methods without affecting the highlighting of true edge.

APPENDIX

CODE:

Code for edge detection (using Sobel detector):

```
clc;
clear all;
close all;
I=imread('download.jpg');
subplot(2,3,1);
imshow(I);
title('Original Input Image');
a=rgb2gray(I);
subplot(2,3,2);
imshow(a);
title('RGB converted image');
k=1;
I1=padarray(a,[k k],'replicate');
[m n]=size(a);
w1=[1,2,1;0,0,0;-1,-2,-1];
w2=[-1,0,-1;-2,0,-2;-2,0,1];
for i=2:(m-1)
   for j=2:(n-1)
      v=double(I1(i-1:i+1,j-1:j+1)).*w1;
      r=sum(v(:));
      c(i-1,j-1)=uint8(ceil(r));
   end
end
for i=2:(m-1)
   for j=2:(n-1)
      v1=double(I1(i-1:i+1,j-1:j+1)).*w2;
      r1=sum(v1(:));
      d(i-1,j-1)=uint8(ceil(r1));
   end
end
e=c+d;
subplot(2,3,3);
imshow(c);
subplot(2,3,4);
imshow(d);
subplot(2,3,5);
imshow(e);
```

Code for comparison with other filters (using in-built functions):

```
clc;
clear all;
close all;
I = imread('download.jpg');
imshow(I)
a=rgb2gray(I)
BW1 = edge(a,'prewitt');
BW2 = edge(a,'canny');
BW3 = edge(a,'roberts');
BW4= edge(a,'log');
tiledlayout(2,2)
nexttile
imshow(BW1)
title('Prewitt Filter')
nexttile
imshow(BW2)
title('Canny Filter')
nexttile
imshow(BW3)
title('ROberts Filter')
nexttile
imshow(BW4)
title('Log Filter')
```

REFERENCES

1) Matlab In-built functions
2) Shrivakshan, G. T., & Chandrasekar, C. (2012). A comparison of various edge detection techniques used in image processing. International Journal of Computer Science Issues (IJCSI), 9(5), 269.
3) Aybar, E. (2006). Sobel edge detection method for matlab. *Anadolu University, Porsuk Vocational School*, *26410*.
4) Maini, R., & Aggarwal, H. (2009). Study and comparison of various image edge detection techniques. International journal of image processing (IJIP), 3(1), 1-11.