Spring 2017

Path-Finding Methodology for Visually-Impaired Patients Based on Image-Processing

Abhilash Goyal
San Jose State University

Path-Finding Methodology for Visually-Impaired Patients Based on Image-Processing

A

Project Report

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Computer Science

By Abhilash Goyal

May 2017

The Designated Project Committee Approves the Project Titled

Path-Finding Methodology for Visually Impaired Patients Based on Image-Processing

By

Abhilash Goyal

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

_____

Professor T Y Lin, Department of Computer Science Date

_____

Professor Robert Chun, Department of Computer Science Date

_____

Dr. HO Howard, IBM Date

ACKNOWLEDGEMENTS

I would like to deeply thank my project advisor, Dr. T Y Lin, for his continuous support, encouragement and guidance. I would also like to thank my committee members Prof. Robert Chun and Dr. Ho Howard for devoting their valuable time and encouragements.

I am extremely thankful to my parents, bother, sister-in-law and my wife for their true friendship and support. Also, I would like to thank all my friends.

Finally, I would like to thank the San Jose State University for giving me a unique opportunity to do Masters in Computer Science. In addition, personal thanks to DeAnna at Computer Science Department for her efforts.

ABSTRACT

The objective of this project is to propose and develop the path-finding methodology for the visually impaired patients. The proposed novel methodology is based on image-processing and it is targeted for the patients who are not completely blind. The major problem faced by visually impaired patients is to walk independently. It is mainly because these patients can not see obstacles in front of them due to the degradation in their eye sight. Degradation in the eye-sight is mainly because either the light doesn't focus on the retina properly or due to the malfunction of the photoreceptor cells on the retina, such as in Retinitis Pigmentosa.

Recently, Second Sight Medical introduced Argus retinal prosthesis, an electronic retinal implant which costs more than US$150,000. This solution is good, but it is very costly. Thus, there is a need of low-cost alternate path-finding methodology for visually impaired patients whose photoreceptor cells on the retina do not work properly.

In this project, the proposed approach is very different from the state-of-the-art. The proposed methodology is for patient's whose vision is impaired such that the patient can not see obstacles, but they are not completely blind. In the proposed methodology, image in front of the patient is captured and processed in the real-time such that the final processed image facilitate patient to find the obstacles in front of them. In this novel proposed methodology, *firstly*, the edges in the captured images are detected, then, *secondly*, these edges are super-imposed on the original image and, *finally*, these edges are widened such that the processed image facilitates the visually impaired patients to recognize obstacles and to figure out the path for walking independently.

TABLE OF CONTENTS

List of Figures

List of Figures (Continue)

List of Acronyms and Terms

GPS – Global Positioning System

RFID - Radio-frequency identification

1. INRODUCTION

As we all know, the major problem faced by visually impaired patients is to walk independently. It is mainly because these patients can not see obstacles in front of them due to the degradation in their eye sight. Degradation in the eye-sight is mainly because either the light does not focus on the retina properly [1] or due to the malfunction of the photoreceptor cells such as in Retinitis Pigmentosa [2]. Recently, Second Sight Medical introduced Argus retinal prosthesis [3], an electronic retinal implant which costs more than US$150,000. This solution is good, but it is very costly, Thus, there is a need of alternate path-finding methodology for visually impaired patients whose photoreceptor cells on the retina do not work properly. Recently, various path-finding methodologies have been proposed, such as GPS based [4], RFID based [5], indoor wheelchair navigation system [6], obstacle detection system [7] and ultrasound positioning based navigation system for blinds [8]. These methodologies are good, but require dedicated network infrastructure as they are mainly targeted for completely blind patients. Thus, that increases the cost of implementation and therefore limits the large adaptation of these methodologies by the patients who are not completely blind.

In this project, the proposed methodology is very different from the state-of-the-art. The proposed methodology is for patients whose vision is impaired such that the patient can not see obstacles, but they are not completely blind. In the proposed methodology, image in front of the patient is captured and processed in the real-time such that the final processed image facilitates patient to find the obstacles in front of them. For illustration purposes, capturing the raw image and displaying the final image can be done as shown in Figures 1.01 and 1.02. Raw images in the projects are taken from www.google.com search [9].

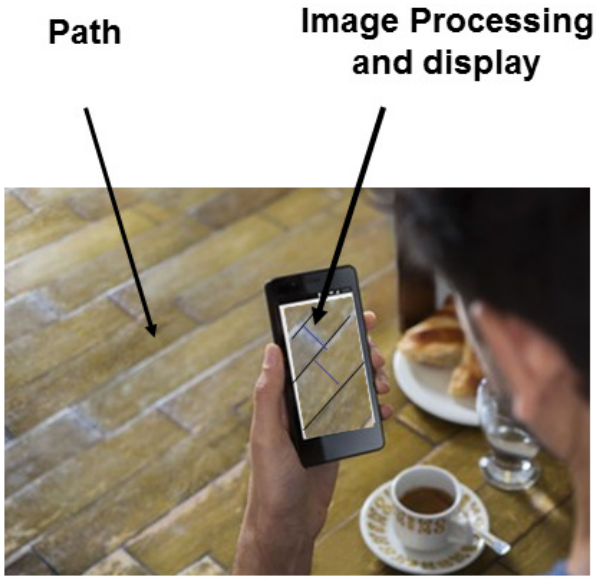


Figure 1.01: Model: capturing and displaying Image using cell phone or tablet [9]

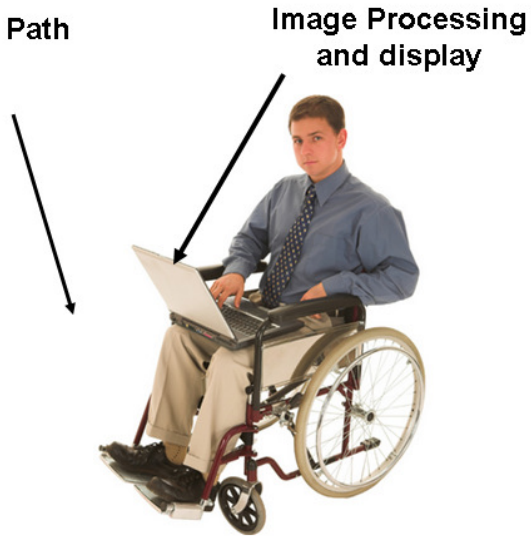


Figure 1.02: Model: capturing and displaying image using laptop [9]

In this novel proposed methodology, firstly, the edges in the captured images are detected, then, secondly, these edges are super-imposed on the original image and, finally, these edges are widened such that the final processed image facilitates the visually impaired patients to recognize obstacles and to figure out the path for walking independently.

Challenging aspects of this project:

Following are the innovative and challenging aspects of the project

1. To develop the *novel methodology* which can be used by the considerable number of patients as compared to the current bionic

eye, which is an electronic retinal implant and costs about US$150,000 (excluding cost of training) [3].

2. To develop and propose the *novel methodology* which can be used in the real-time by visually impaired patients.
3. To demonstrate that the proposed novel methodology is implementable.

The rest of the project report is organized as follows: In Chapter 2, the path-finding methodology based on image processing is proposed, In Chapter 3, the proposed methodology is modeling and image processing is shown in Matlab. Chapter 4 shows demonstration of the proposed methodology in the Python. Chapter 5, proposes the new edge detection technique. Chapter 6 marks the future scope of the project, which is followed by references.

2. NOVEL PROPOSED METHODOLOGY

Novel path-finding methodology for visually-impaired patients based on image-processing involves three steps as shown below the Figure 2.01.

Step 1). The edges in the captured images are detected

Step 2). These edges are super-imposed on the original image

Step 3). These edges are widened

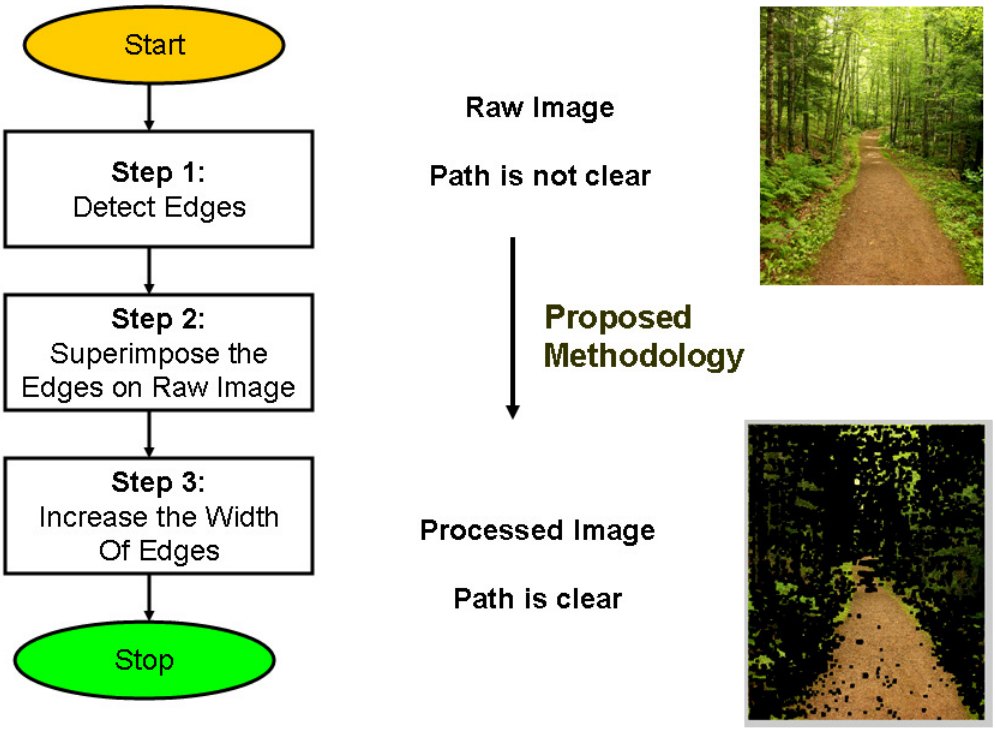


Figure 2.01: The proposed methodology: flow chat and steps

These three steps on the raw image produce the final processed image, which facilitates the visually impaired patients to recognize obstacles and to figure out the path for walking independently. For the illustration, the raw image and processed image are also shown in the Figure 2.01.

In the next chapters, the proposed methodology is demonstrated by modeling in Matlab [10] and with actual implementation in the Python.

3. MATLAB MODELING OF PROPOSED METHODOLOGY

In this chapter, the proposed methodology is demonstrated by image processing in the Matlab

3.1 Image Processing and Modeling

In this sub-section, for edge detection, three edge detection methods are used such as Prewitt [11, 12], Sobel [13] and Canny [14] edge detection. For illustration, the edge detected on the raw image of Figure 3.01 is shown in Figure 3.02.

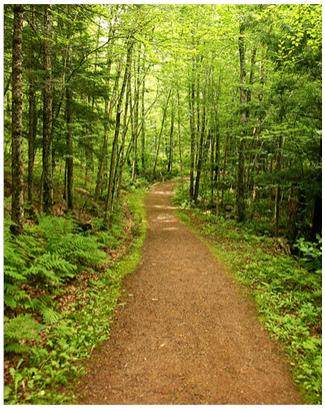


Figure 3.01: Raw image, without any image processing. Image courtesy [9]

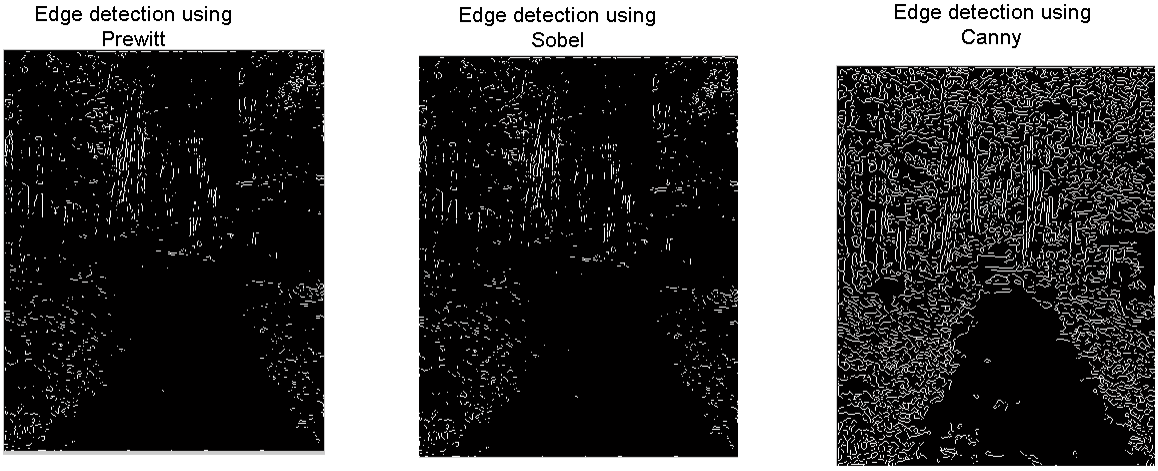


Figure 3.02: Edge detection in the raw image of Figure 3.01 using different methods

As it can be observed from the above results that just detecting the edges in the image is not sufficient to enhance the image because the image with only the edges are more difficult to defer. Therefore, in the proposed methodology, after the edges are detected they are superimposed on the raw image and to increase the visibility, the edges in the image are widened. The final processed image is shown in Figures 3.03 and 3.04.

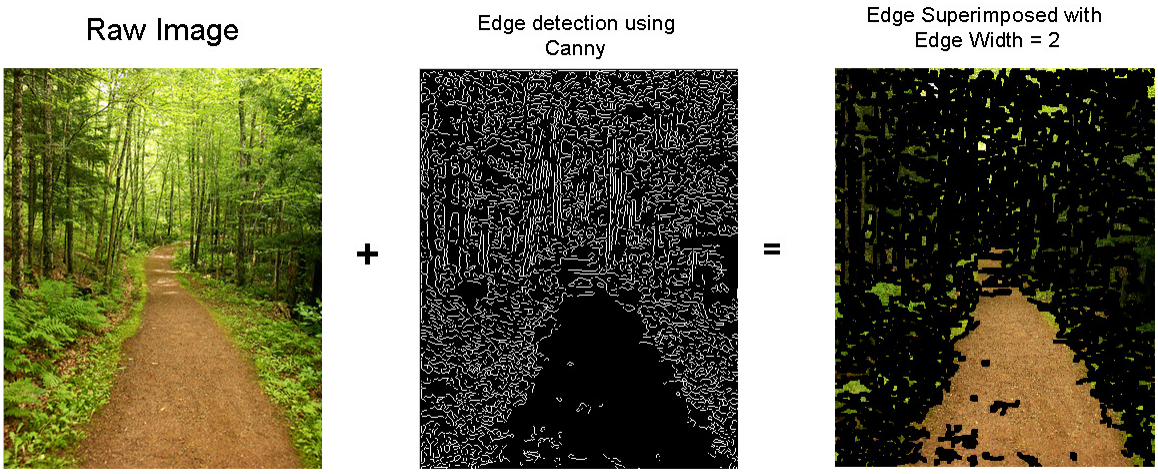


Figure 3.03: Superposing the edges on the raw image and increasing the edge width

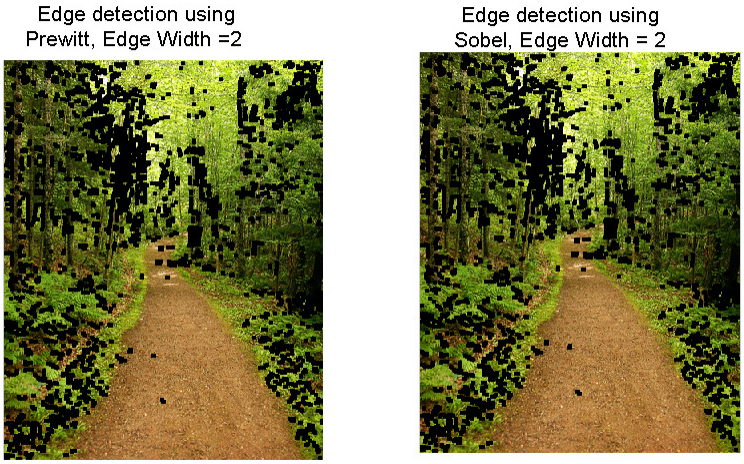


Figure 3.04: Final processed images by the proposed methodology using Prewitt and Sobel edge detection

This can be easily inferred from the above Figures 3.03 and 3.04 that after following the proposed methodology, the path in the image is easy to defer and therefore it enables the visually impaired patients to recognize obstacles and to figure out the path for walking independently.

3.2 Test Cases

To further bolster the proposed methodology, different images are randomly chosen from the internet using www.google.com [9] and they are enhanced as shown in the below Figures 3.05 and 3.06.

Test Image 2:

Figure 3.05 (A)

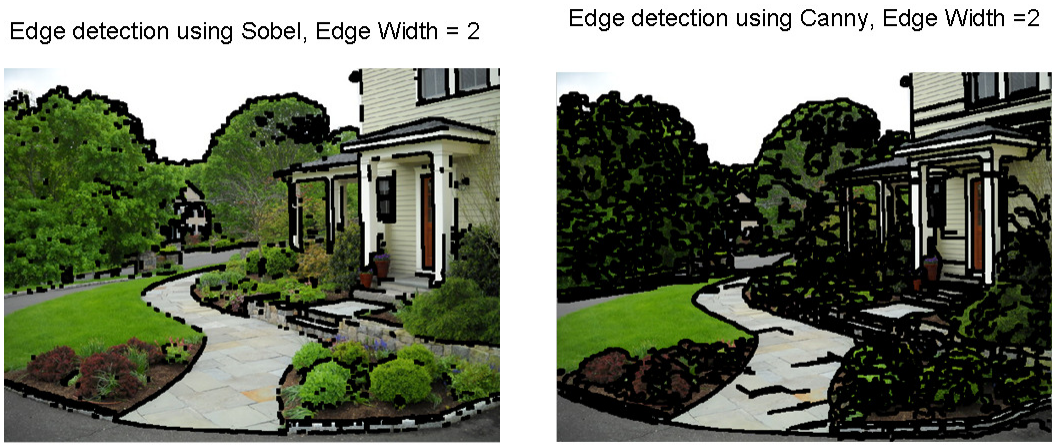


Figure 3.05 (B)

Figure 3.05: Test Image 2 and its enhancement using the proposed methodology

Test Image 3:

Figure 3.06 (A)

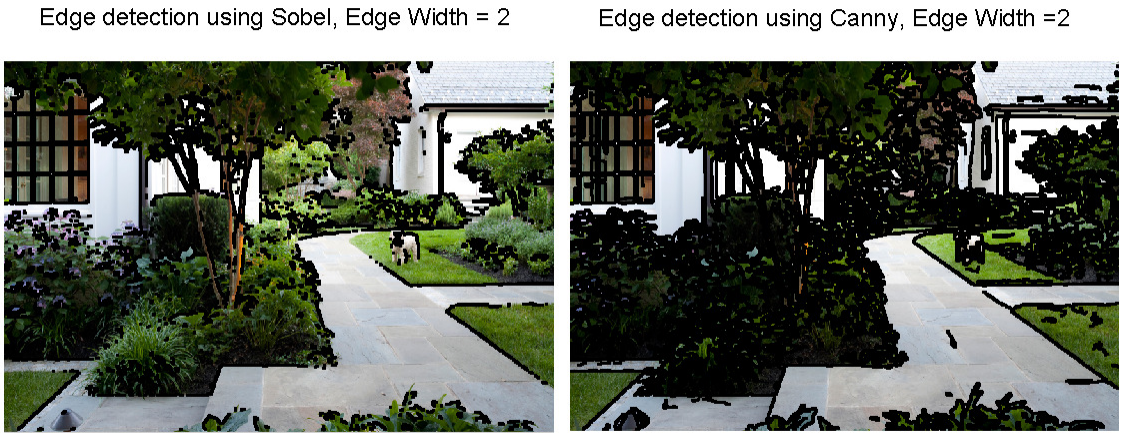


Figure 3.06 (B)

Figure 3.06: Test Image 3 and its enhancement using the proposed methodology

3.3 Matlab Code

The Matlab code of the above results is given below.

```
clear all
close all


IMG_COLOR = imread('Path.jpg');
%IMG_COLOR = imread('Path2.jpg');
%IMG_COLOR = imread('Path3.jpg');

Processed_IMG_COLOR = IMG_COLOR;

figure(1)
imshow(IMG_COLOR);
title('input image');

IMG_ORG = rgb2gray(IMG_COLOR);
figure(11)
imshow(IMG_ORG);
title('input image');
```

```
%%%%%%%%% Gaussian Filter %%%%%%%%%%
HAG = fspecial('gaussian',5,1);
IMG_ORG = filter2(HAG,IMG_ORG);
[row_image column_image] = size(IMG_ORG)

Marker_size = 2;

%%%%%%%%%%%%%% PREWITT %%%%%%%%%%%%%%

figure(1116)
hold on
edge_Prewitt = edge(IMG_ORG,'Prewitt');
imshow(edge_Prewitt);title('Prewitt processing');

TM = IMG_COLOR;
Edge_IMG = edge_Prewitt;
High_level = 0;
Marker = 0;

for kk = Marker_size + 1 : row_image - Marker_size -1
    for jj = Marker_size + 1 : column_image -Marker_size -1

    if (Edge_IMG(kk,jj) > High_level)
    TM(kk-Marker_size:kk+Marker_size, jj-Marker_size:jj+Marker_size,:)
= 0;
    end

    jj = jj + 1;

    end
    kk= kk + 1;
end

figure(1126)
hold on
imshow(TM);
title(' Prewitt : Marker Size = 2');


%%%%%%%%%%%%%% SOBEL %%%%%%%%%%%%%%%%

figure(1117)
hold on
edge_Sobel = edge(IMG_ORG,'Sobel');
imshow(edge_Sobel);title('Sobel processing');

TM = IMG_COLOR;
Edge_IMG = edge_Sobel;
High_level = 0;
Marker = 0;
```

```
for kk = Marker_size + 1 : row_image - Marker_size -1
    for jj = Marker_size + 1 : column_image -Marker_size -1

    if (Edge_IMG(kk,jj) > High_level)
    TM(kk-Marker_size:kk+Marker_size, jj-Marker_size:jj+Marker_size,:)
= 0;
    end

    jj = jj + 1;

    end
    kk= kk + 1;
end

figure(1127)
hold on
imshow(TM);
title('Sobel : Marker Size = 2');

%%%%%%%%%%%%%% CANNY %%%%%%%%%%%%%%%

figure(1118)
hold on
edge_canny = edge(IMG_ORG,'canny');
imshow(edge_canny);title('canny processing');


TM = IMG_COLOR;
Edge_IMG = edge_canny;
High_level = 0;
Marker = 0;

for kk = Marker_size + 1 : row_image - Marker_size -1
    for jj = Marker_size + 1 : column_image -Marker_size -1

    if (Edge_IMG(kk,jj) > High_level)
    TM(kk-Marker_size:kk+Marker_size, jj-Marker_size:jj+Marker_size,:)
= 0;
    end

    jj = jj + 1;

    end
    kk= kk + 1;
end

figure(1128)
hold on
imshow(TM);
title('CANNY : Marker Size = 2');
```

3.4 Effect of Edge Detection and Edge size

It can be inferred from the previous sub-sections that the quality of edge detection in the image is very much depended on the kind of edge detection technique used. Thus, for the real-world application, the patient should be given an option to choose kind of the edge detection they want to use.

In addition to the edge detection, the width of the edge is also important. To illustrate the effect of edge width size, enhanced image with different edge width size is shown below in Figure 3.07. Thus, again for the real-world application, visually impaired patient should be given an option to choose different edge-width size.

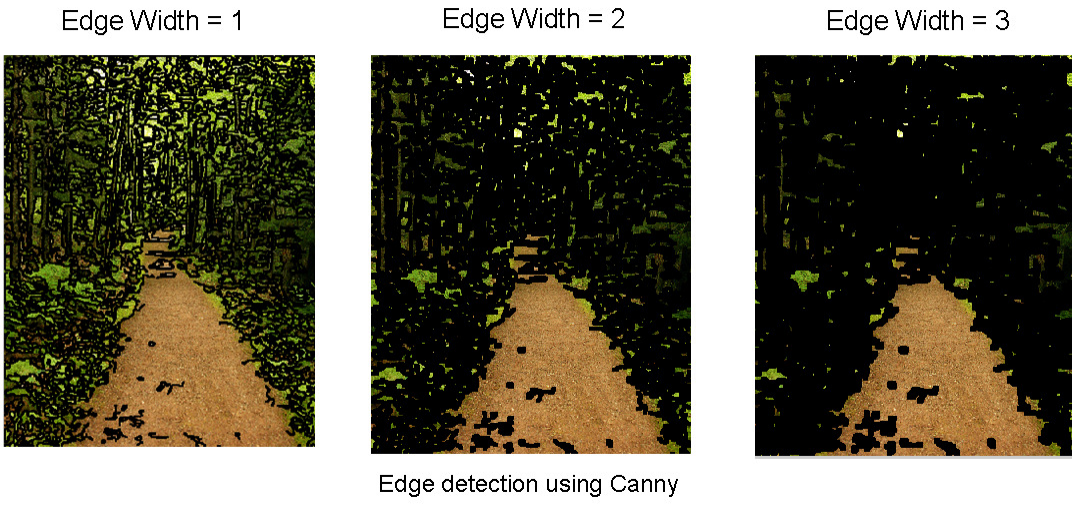
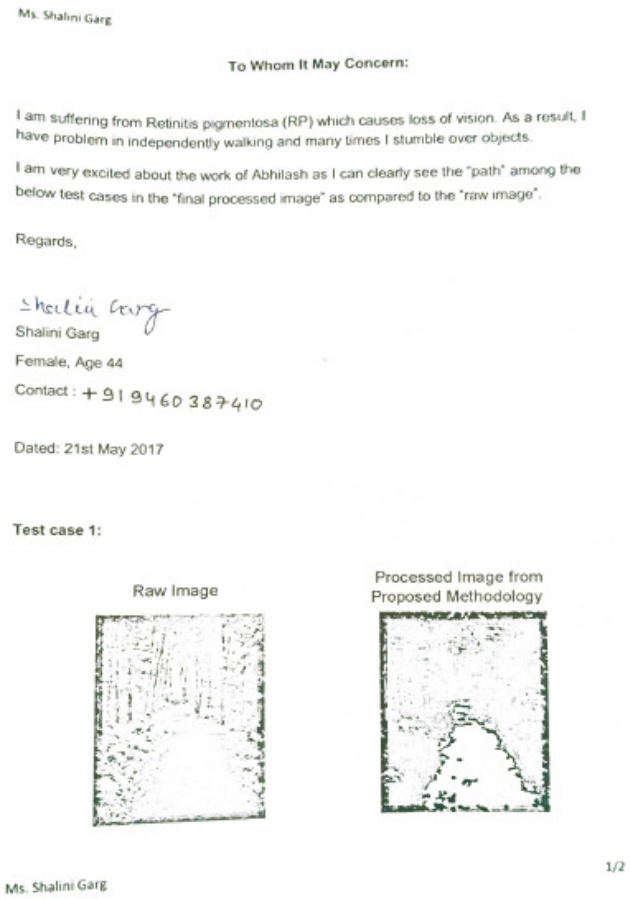


Figure 3.07: Enhancement of raw image of Figure 3.01 using the proposed methodology with different Edge widths

3.5 Visually Impaired Patient: Testing and Feedback

To get real feedback, the above test images where shown to the visually-impaired patient Ms. Shalini Garg, who is suffering from "Retinitis pigmentosa (RP)". Testing was successful and feedback is very positive. Ms. Shalini Garg, mentioned that from "processed images", she can clearly see the path. Feedback report from Ms. Shalini is shown in Figure 3.08.



Ms. Shalini Garg

To Whom It May Concern:

I am suffering from Retinitis pigmentosa (RP) which causes loss of vision. As a result, I have problem in independently walking and many times I stumble over objects.

I am very excited about the work of Abhilash as I can clearly see the "path" among the below test cases in the "final processed image" as compared to the "raw image".

Regards,

Shalini Garg

Shalini Garg

Female, Age 44

Contact : +91 9460 387410

Dated: 21st May 2017

Test case 1:

Raw Image

Processed Image from Proposed Methodology

1/2

Ms. Shalini Garg

Figure 3.08 (A)

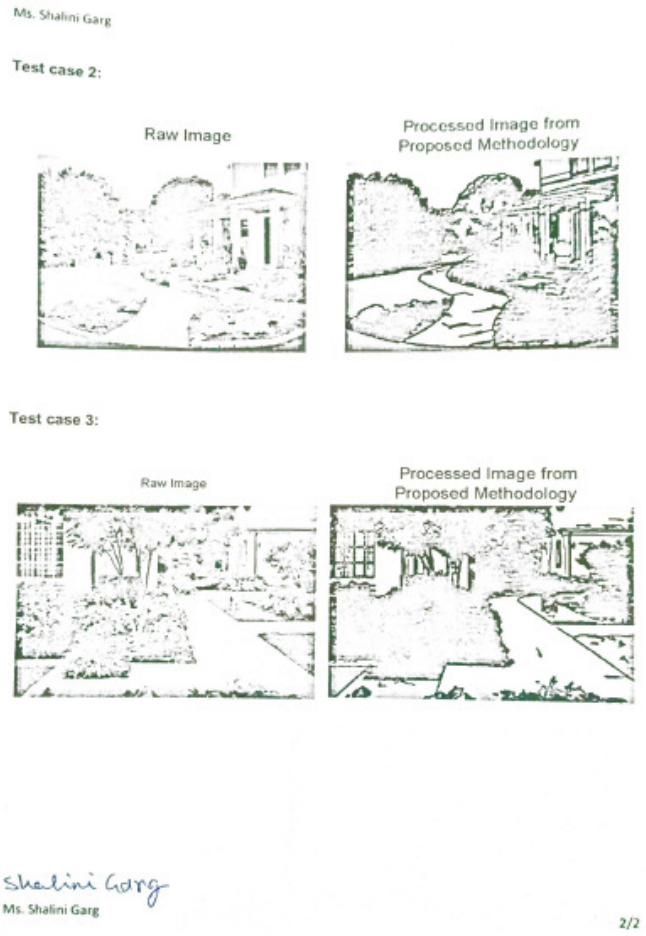


Figure 3.08 (B)

Figure 3.08: Feedback report form Ms. Shalini (Visually Impaired Patient)

3.6 Conclusion

Thus, based on the results obtained from this and previous sub-sections, it can be concluded that the proposed methodology is promising and for the real-world application it can be implemented as shown below in Figure 3.09.

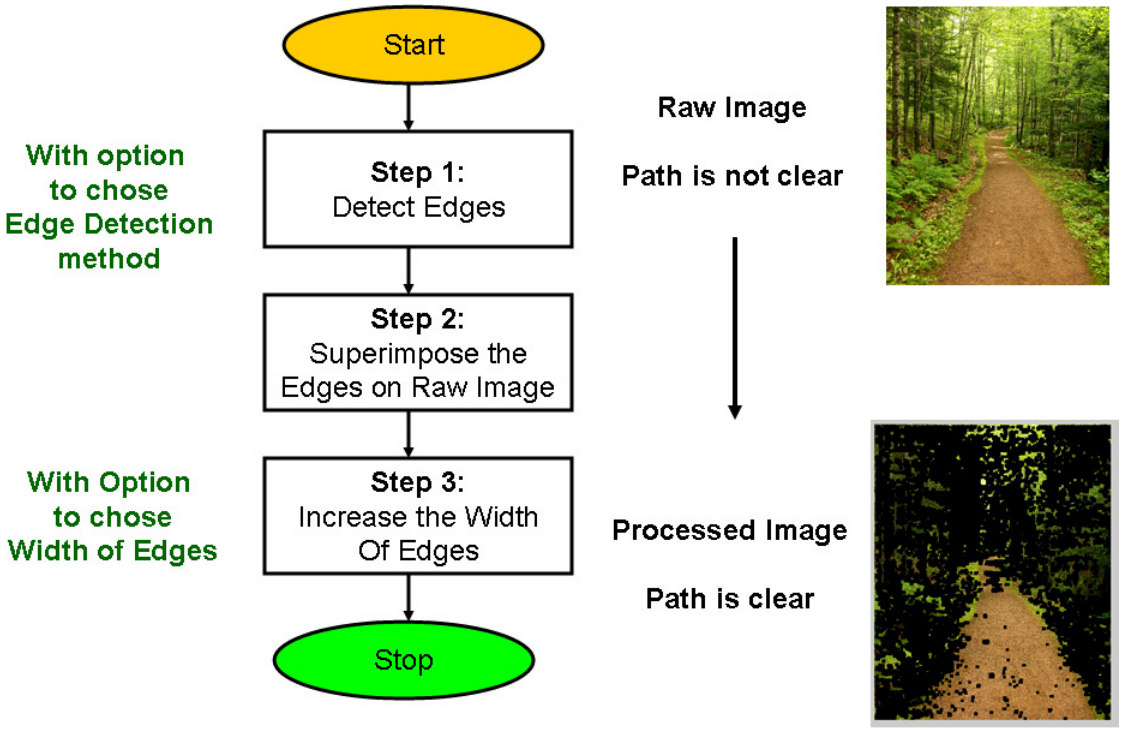


Figure 3.09: The proposed methodology framework

4. PYTHON DEMONSTRATION OF PROPOSED METHODOLOGY

In this chapter, the proposed methodology is demonstrated by implementing in the Python. For this implementation, Python version 2.7 [15] and OpenCV [16] image-processing library is used.

The methodology is demonstrated for still images and real time application.

4.1 Python Implementation for Still Image

The Python code for still images is given below. For demonstration, this code is using the OpenCV in-built library which has “canny edge” detection. As proposed in the methodology, after edge detection, the detected edges are widened and superimposed on the raw image.

4.1.1 Python Code for Still Images

```
#!/usr/bin/env python
import cv2
def main():
      org_image = cv2.imread('Path.jpg')
```

```
        copy_image = org_image
        # save image
        cv2.imwrite("Input_image.jpg",org_image)
        cv2.imshow("Input Image",org_image)
        cv2.waitKey(10)

        ####################################
        ####################################
        gray_image = cv2.cvtColor(org_image,cv2.COLOR_BGR2GRAY)

        #Guassian Filtering
        temp_gray = cv2.GaussianBlur(gray_image,(5,5),0)

        #Image Processing using Canny Edge detection

        thres1 = 220
        thres2  = 251
        Marker  = 0
        edge_image = cv2.Canny(gray_image,thres1,thres2)
        cv2.imwrite("Canny_Edge_Image.jpg",edge_image)
        cv2.imshow("Canny Edge Image",edge_image)
        cv2.waitKey(10)

        row, col = edge_image.shape
        for kk in range(1,row):
              for jj in range(1,col):
                    if(edge_image[kk,jj] > thres2):
                          copy_image[kk,jj,0] = Marker
                          copy_image[kk,jj,1] = Marker
                          copy_image[kk,jj,2] = Marker

                          copy_image[kk-1,jj,0] = Marker
                          copy_image[kk-1,jj,1] = Marker
                          copy_image[kk-1,jj,2] = Marker

                          copy_image[kk-1,jj-1,0] = Marker
                          copy_image[kk-1,jj-1,1] = Marker
                          copy_image[kk-1,jj-1,2] = Marker

                          copy_image[kk,jj-1,0] = Marker
                          copy_image[kk,jj-1,1] = Marker
                          copy_image[kk,jj-1,2] = Marker


        #save image
        cv2.imwrite("Proposed_Image.jpg",copy_image)
        cv2.imshow("Processed Image with Marker",copy_image)
        cv2.waitKey(100)



main().
```

4.1.1 Test Result

Figure 4.01: Test result of Python demonstration of the proposed methodology on image

This can be inferred from the above result that after the image is enhanced by the proposed methodology, the path in the final processed image is very much prominent. Also, obtained results are similar to the results as obtained in the Chapter 3.

4.2 Python Implementation for Dynamic Application

To further bolster the proposed path-finding methodology for the visually impaired patients, the real-time processing code is implemented in the Python. In this real-time code, the image is captured continuously and the

image is enhanced in the real time. The implemented Python code is shown below. In this code, the image is captured using the camera on the laptop and final processed image is shown on the screen in the real time. Two example of the real-time capture is shown in Figures 4.02 and 4.03. The average processing time in the real time is around 10s.

4.2.1 Python Code for Dynamic Application

```
#!/usr/bin/env python

import cv2

def main():



    # initialize the camera
    # index of camera
    Marker = 0
    cam = cv2.VideoCapture(1)

    flag = 1
    while flag:
        s, org_image = cam.read()
        copy_image = org_image

        if s:     # for image captured without any errors
            cv2.namedWindow("Captured Image",cv2.CV_WINDOW_AUTOSIZE)
            cv2.imwrite("Input_image.jpg",org_image)
            cv2.imshow("Original Image",org_image)
            cv2.waitKey(10)

            gray_image = cv2.cvtColor(org_image,cv2.COLOR_BGR2GRAY)
            #cv2.namedWindow("Gray Image",cv2.CV_WINDOW_AUTOSIZE)

            temp_gray = cv2.GaussianBlur(gray_image,(5,5),0)

            ########################
            thres1 = 80
            thres2  =150
            analyze_image = cv2.Canny(temp_gray,thres1,thres2)
```

```
            final_image = analyze_image
            cv2.imwrite("Edge_Image.jpg",final_image)
            cv2.imshow(" Edge Image",final_image)
            cv2.waitKey(10)

            row, col = final_image.shape
            for kk in range(3,row-3):
                for jj in range(3,col-3):
                    if(final_image[kk,jj] > thres2):
                        ## JJ
                        copy_image[kk,jj,0] = Marker
                        copy_image[kk,jj,1] = Marker
                        copy_image[kk,jj,2] = Marker

                        copy_image[kk-1,jj,0] = Marker
                        copy_image[kk-1,jj,1] = Marker
                        copy_image[kk-1,jj,2] = Marker

                        copy_image[kk+1,jj,0] = Marker
                        copy_image[kk+1,jj,1] = Marker
                        copy_image[kk+1,jj,2] = Marker

                        #JJ -1
                        copy_image[kk,jj-1,0] = Marker
                        copy_image[kk,jj-1,1] = Marker
                        copy_image[kk,jj-1,2] = Marker

                        copy_image[kk-1,jj-1,0] = Marker
                        copy_image[kk-1,jj-1,1] = Marker
                        copy_image[kk-1,jj-1,2] = Marker

                        copy_image[kk+1,jj-1,0] = Marker
                        copy_image[kk+1,jj-1,1] = Marker
                        copy_image[kk+1,jj-1,2] = Marker

                        #JJ + 1
                        copy_image[kk,jj+1,0] = Marker
                        copy_image[kk,jj+1,1] = Marker
                        copy_image[kk,jj+1,2] = Marker

                        copy_image[kk-1,jj+1,0] = Marker
                        copy_image[kk-1,jj+1,1] = Marker
                        copy_image[kk-1,jj+1,2] = Marker

                        copy_image[kk+1,jj+1,0] = Marker
                        copy_image[kk+1,jj+1,1] = Marker
                        copy_image[kk+1,jj+1,2] = Marker

            cv2.imwrite("Processed_Image.jpg",copy_image)
            cv2.imshow("Processed : Image with Marker",copy_image)
            cv2.waitKey(10)

main()
```

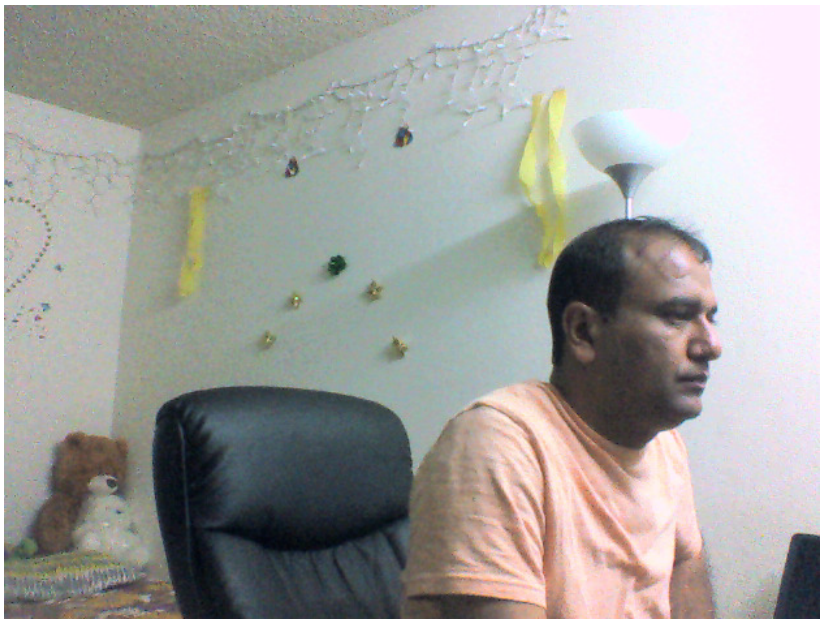


Figure 4.02 (A): Raw captured Image

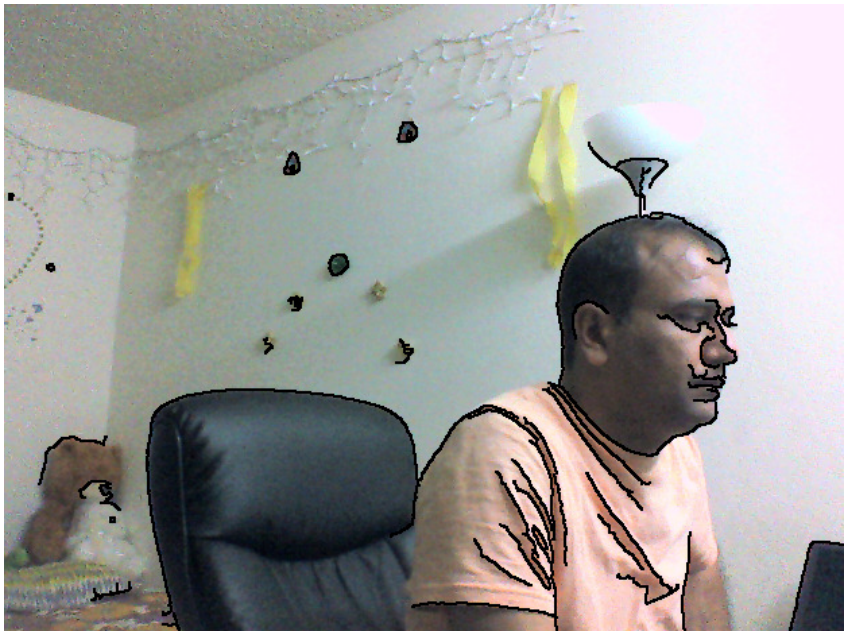


Figure 4.02 (B): Real Time Processed Image

Figure 4.02: Real time processing Example 1.

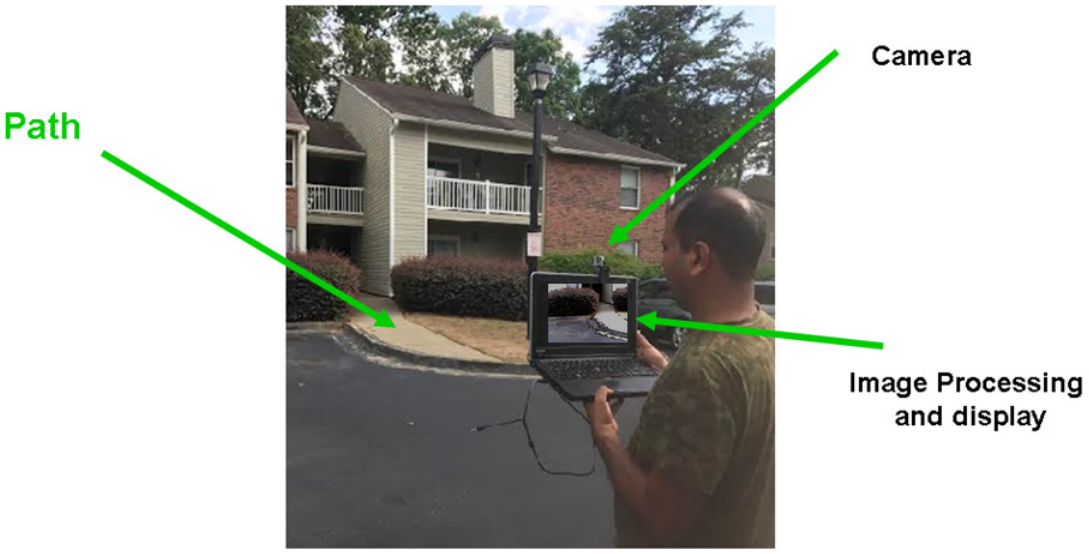


Figure 4.03 (A): Experimental setup

Figure 4.03 (B): Real time images

Figure 4.03: Experimental setup for real-time application.

4.3 Conclusion

In this chapter, the novel “Path-Finding Methodology for Visually-Impaired Patients Based on Image-Processing" is demonstrated in Python. Based on the obtained results, it can very easily be concluded that the proposed methodology is very much application for still and real-time image processing. Thus, enabling the visually impaired patients for recognizing obstacles and figuring out the path to walk independently.

5. NEW TECHNIQUE FOR EDGE DETECTON

The conventional methods of edge detection are based on derivate of the image as shown in Figures 5.01 and 5.02.

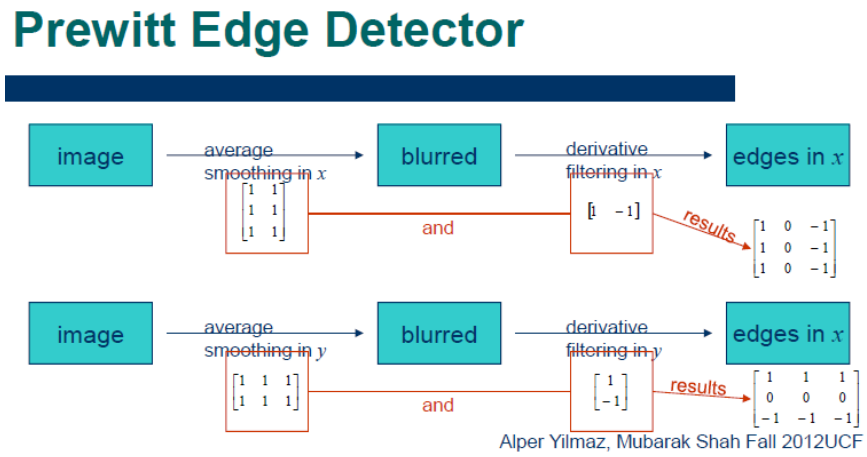


Figure 5.01: Prewitt edge detection [17]

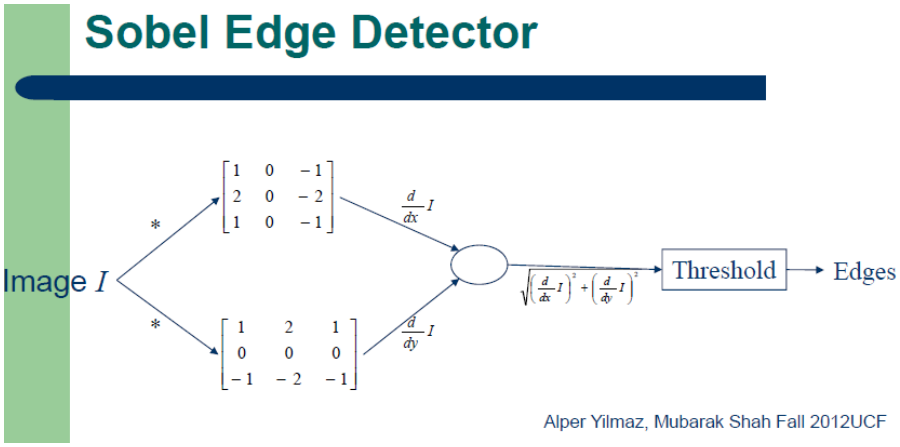


Figure 5.02: Sobel edge detection [17]

In the proposed new edge detection technique, for each pixel, number of the nearby pixel with same intensity (or within the intensity range) is calculated. If the pixel has less than 8 adjacent pixel count, then it is considered as edge pixel, otherwise not. This simple technique leads to proper edge detection as shown in the below "model image" and do not require taking any derivates.

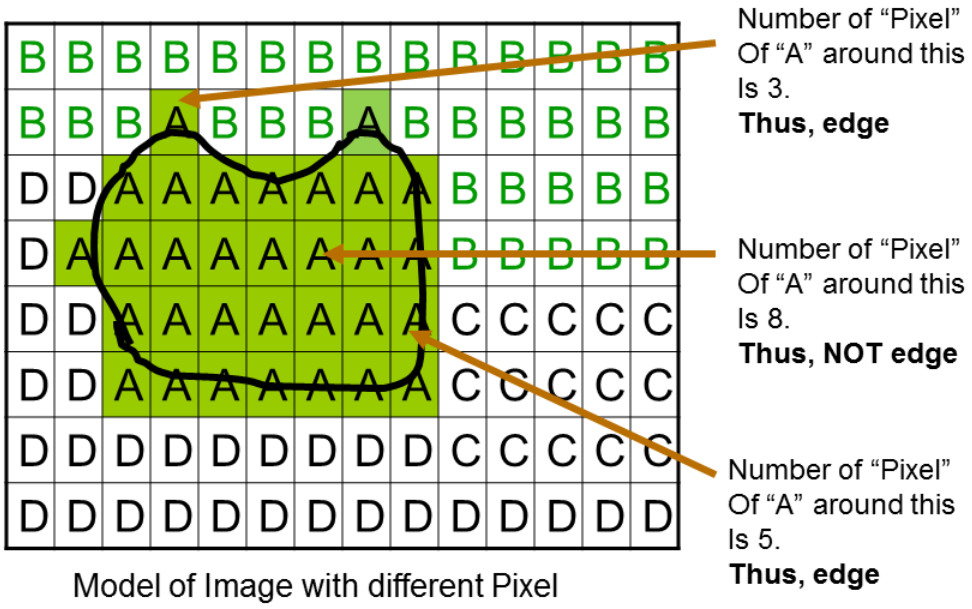


Figure 5.03: Underline principle of the proposed novel edge detection technique

5.1 Comparison with Conventional Edge Detection Techniques

In this sub-section, the edge detection using the proposed technique is compared with conventional edge detection techniques, such as “Sobel”, “Prewitt” and “Canny. Results are shown below in Figures 5.04 to 5.06. For this comparison, well known “Lena” image [18] is used as shown in Figure 5.04 and Matlab is used for the modeling.

Image of “Lena”

Figure 5.04 : Image of “Lena”, standard image used in image processing community

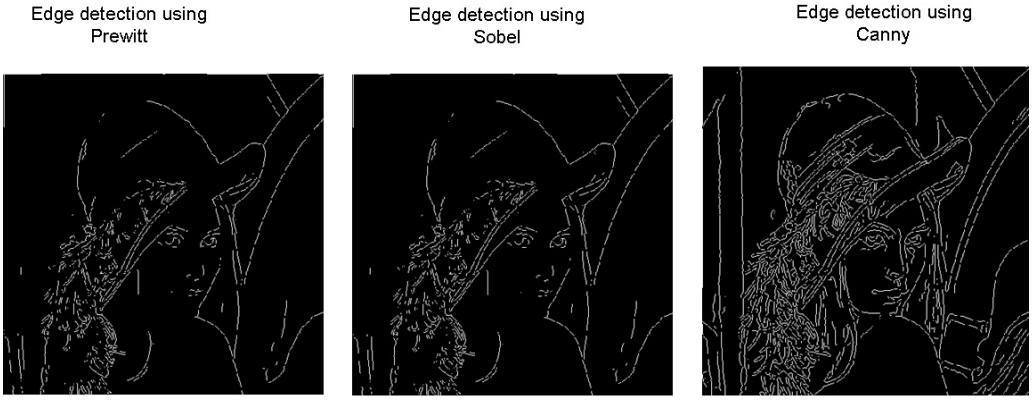


Figure 5.05: Detected edges in the “Lena” image using state-of-the-art techniques

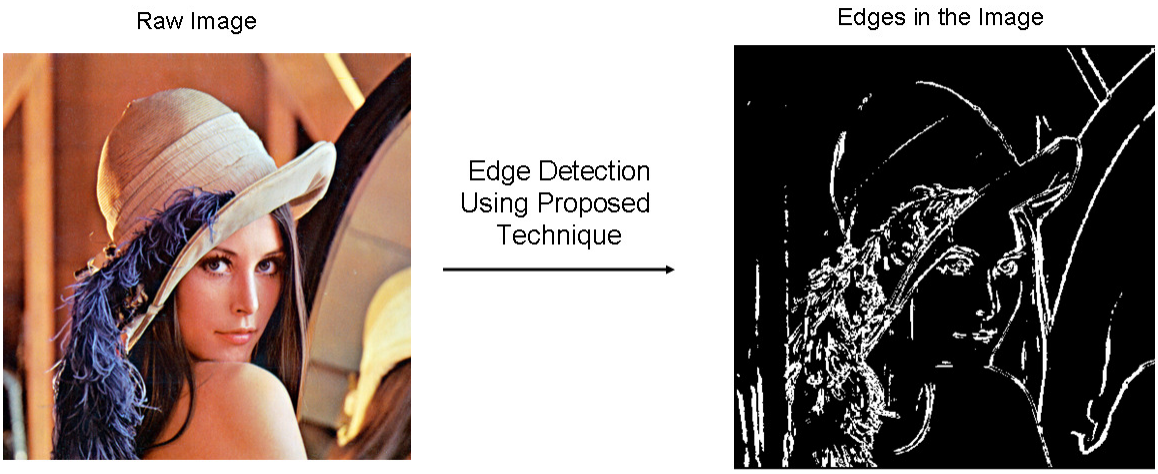


Figure 5.06: Detection of the edges in the image of "Lena" using the proposed edge detection technique.

Based on the above results of Figures 5.04, 5.05 and Figure 5.06, it can be concluded that the proposed edge detection technique can be used to detect edges in the images. Also, the performance of the proposed edge-detection technique is comparable to the conventional edge-detection techniques.

5.2 Integration of the proposed edge detection technique in the proposed "path-finding" methodology

The proposed edge detection technique is further integrated into the proposed "path-finding" methodology of Chapter 3. To demonstrate the integration, the Matlab is used and Matlab code is shown below.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%% Abhilash's Edge Detection Technique %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%Image

TM = IMG_ORG;
%Proximity tolerance
High_level = 15;
Marker_value = 0;
edge_value = 255
Marker_size = 1;
%noise limit and core count
noise_limit = 2;
core_limit = 6;
%Edge Width Size
bs = 2;

TM_Edge = IMG_COLOR;
[row_image column_image] = size(TM)
edge_image(row_image,column_image) = 0;


for kk = Marker_size + 1 + bs : row_image - Marker_size - bs
    for jj = Marker_size + 1 + bs : column_image -Marker_size - bs
      count_adj = 0;
        for mm_rr = kk - Marker_size : kk + Marker_size
            for mm_cc = jj - Marker_size : jj + Marker_size
            if(TM(kk,jj) - High_level <= TM(mm_rr,mm_cc) &&
TM(mm_rr,mm_cc) <= TM(kk,jj) + High_level)
                count_adj = count_adj + 1;
            end
            mm_cc = mm_cc + 1;
            end
        mm_rr = mm_rr + 1;
        end

    count_matrix(kk,jj) = count_adj;
    %%%%% Pointer boundary
    if( (noise_limit < count_adj) && (count_adj < core_limit + 1))
    edge_image(kk,jj) = edge_value;
    TM_Edge(kk - bs : kk + bs ,jj - bs : jj + bs, :) = Marker_value;
    end
    jj = jj + 1;
        end
    kk= kk + 1;
end

count_matrix;
```

```
figure(1129)
hold on
imshow(edge_image);
title('AG: Edge Image');

figure(1130)
hold on
imshow(TM_Edge);
title('AG: Processed Image');
```

5.3 Test Cases

In this sub-section, the results obtained in the Chapter 3 are compared with integrated proposed edge-detection technique. Summary of the results is shown in Figures 5.07 to 5.09 and obtained results are promising.

5.3.1 Test Image 1

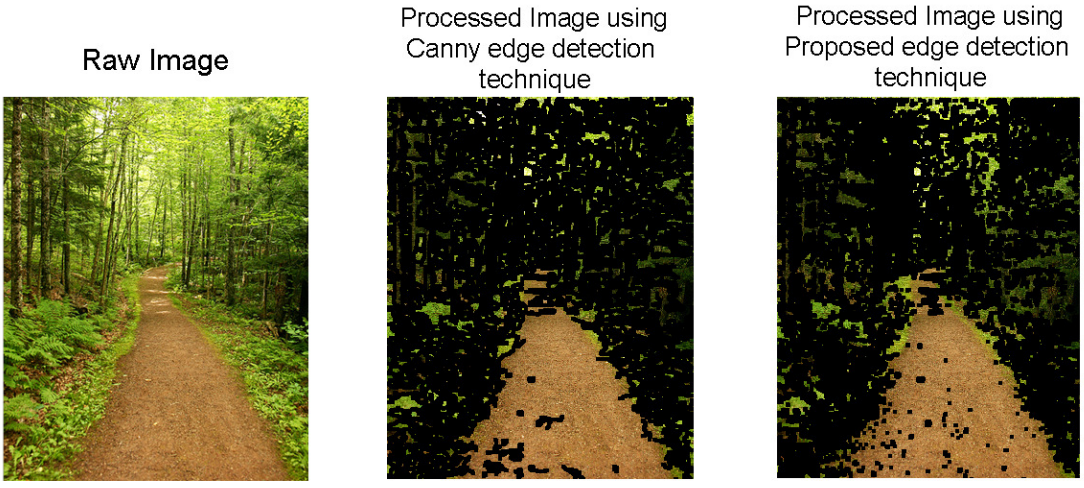


Figure 5.07: Test Image 1: Processed Image using Proposed edge detection technique

5.3.1 Test Image 2

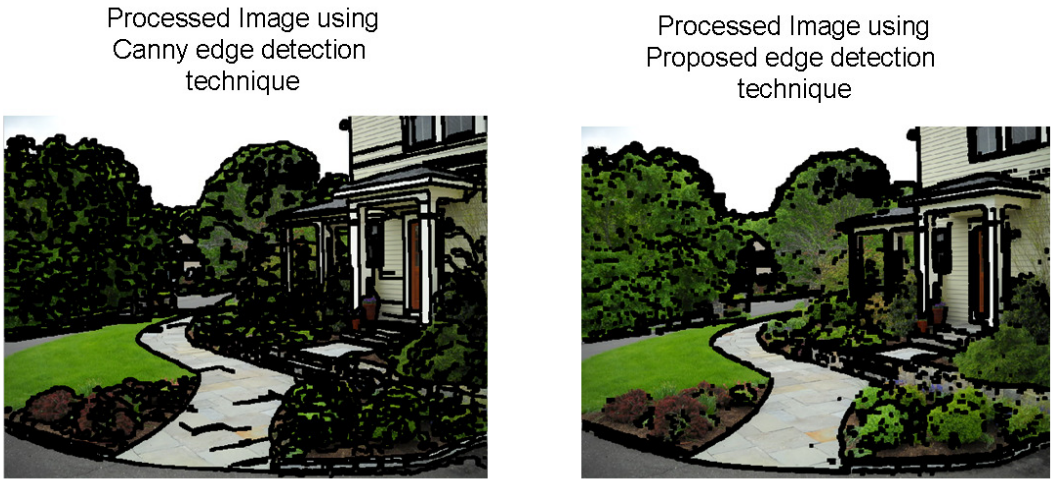


Figure 5.08: Test Image 2: processed image using proposed edge detection technique

5.3.1 Test Image 3

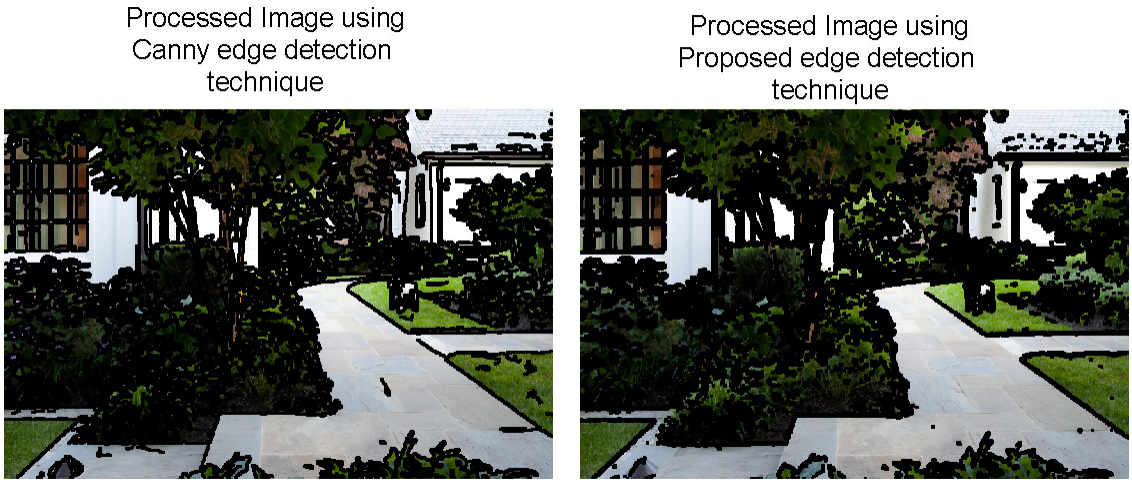


Figure 5.09: Test Image 3: processed image using proposed edge detection technique

5.4 Conclusion

In this chapter, the new edge-detection technique is proposed and it is modeled in the Matlab. Unlike conventional edge detection technique, the proposed technique does not depend on the derivate of the image pixel. In the proposed technique, each pixel intensity is directly compared with its adjacent pixel intensity and decision of the edge pixel is made. The technique is also integrated in the proposed path-finding methodology for visually impaired patients.

6. FUTURE SCOPE OF THE PROJECT

Since visually impaired patient's eyesight becomes weak, they cannot recognize person around them. To help them further, the methodology based on machine-learning to recognize person around them can be explored. The framework of the methodology is shown in below Figure 6.01.

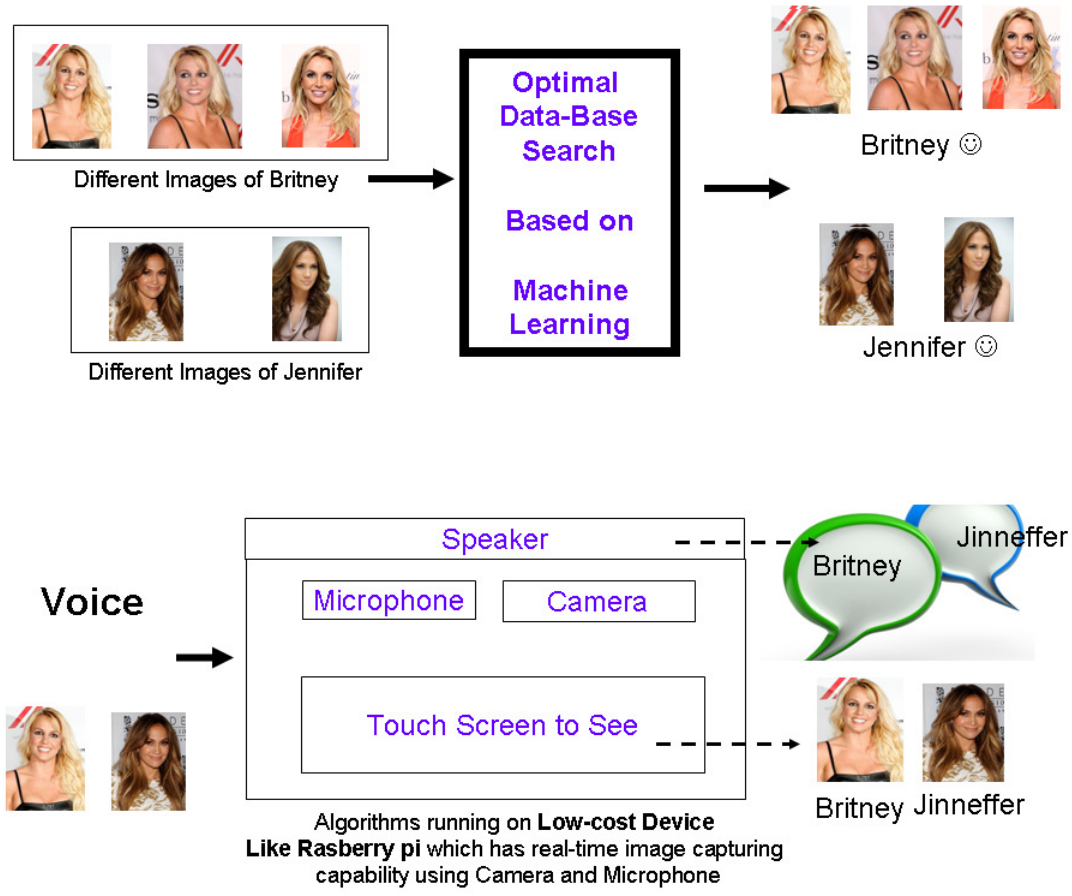


Figure 6.01: Future Scope of the project

7. REFERENCE

Please note: Raw images in this project are taken from search at www. google.com

1. https://www.visionexpress.com/eye-health/eye-conditions/myopia-hypermetropia/ Accessed on : May 20, 2017
2. http://www.blindness.org/retinitis-pigmentosa Accessed on : January 31, 2017
3. https://en.wikipedia.org/wiki/Argus_retinal_prosthesis Accessed on : January 31, 2017
4. Al-Faiz, M.Z. and Ghufran, E.M., 2015. GPS-based Navigated Autonomous Robot. International Journal of Emerging Trends in Engineering Research, 3(4).
5. Dinçer, Z., Öktem, R. and Aydın, E., 2008. An algorithms for RFID based location estimation. Atilim University Graduate School of Natural and Applied Sciences.
6. Hosny, M., Alsarrani, R. and Najjar, A., 2015. Indoor Wheelchair Navigation for the Visually Impaired. In HCI International 2015-

Posters' Extended Abstracts (pp. 411-417). Springer International Publishing.

7. Mohajeri N, Raste R, Daneshvar S, 2011, An Obstacle Detection System for Blind People, Proceedings of the World Congress on Engineering 2011 Vol II WCE 2011, July 6- 8
8. Ran, L., Helal, S. and Moore, S., 2004, March. Drishti: an integrated indoor/outdoor blind navigation system and service. In Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on (pp. 23-30).
9. www.google.com, Accessed on : May 20, 2017
10. https://www.mathworks.com/, Accessed on : May 20, 2017
11. https://en.wikipedia.org/wiki/Prewitt_operator, Accessed on : April 20, 2017
12. https://www.tutorialspoint.com/dip/prewitt_operator.htm, Accessed on : April 20, 2017
13. http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm, Accessed on : April 15, 2017

14. http://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html, Accessed on : January 1, 2017
15. https://www.python.org/download/releases/2.7/, Accessed on : January 1, 2017
16. http://opencv.org/, Accessed on : January 1, 2017
17. https://pcvlab.engineering.osu.edu/people/yilmaz.15, Accessed on : May 20, 2017
18. http://www.cs.cmu.edu/~chuck/lennapg/lenna.shtml, Accessed on : May 20, 2017