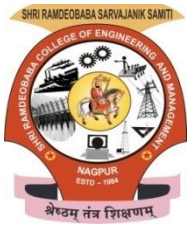


SHRI RAMDEOBABA COLLEGE OF ENGINEERING AND MANAGEMENT

Department of Computer Science & Engineering

Session: 2019-2020

SOFTWARE-TECH LAB-II

PROJECT

VI Semester, B.E. Shift I

Group Members

Stavan Khisty(77)

Latish Magnani(78)

Nakul Sambare(79)

Yogesh Kothari(80)

Course Co-ordinator

Prof. Jignyasa B Sanghavi

Abstract

Roads are considered to be the main mode of transportation. But due to this heavy use of roads and environmental factors, these roads need a scheduled maintenance. Often this maintenance is not performed since it is not possible to monitor each and every place or simply because of ignorance. This leads to the formation of potholes which causes unwanted traffics and the majority of accidents. Our software aims to detect these potholes from the images and then these images which will contain different detected potholes can be used to fill those potholes

CONTENTS:

LIST (FIGURES AND TABLES)

FIGURES:

TABLES:

INTRODUCTION

India is one of the most populous country, roads are the main mode of transportation in this developing country. But due to the heavy use of roads, there is a high amount of wear and tear carried out. Since these roads cannot sustain itself for a long time, a timely maintenance is expected to be carried out in order to prevent the formation of potholes. The manner in which a pothole is formed depends on the type of bituminous pavement surfacing. The heavy traffic on the road is the primary reasons for the fatiguing of the road surface, resulting in the formation of the crack. These depressions collect water and allow the water to mix with the asphalt. When vehicles drive through such holes the water is expelled along with some of the asphalt, and this slowly creates a cavity underneath the crack. If a regular road maintenance is neglected, the road surface will eventually collapse into the cavity, resulting in a visibly huge pothole over the surface. In order to repair these roads in a timely manner, it is necessary that the entity knows which area is affected by the pothole or decaying road section is located and an automated process could assist with this.

All these reasons demand that it is important to collect information of the road conditions and through a series of processing and analysing the obtained information, appropriate conclusions are derived which in turn, warn the officials of the respective area.

Our approach uses the image processing techniques to detect the potholes on the road the images can then be used for further analysis. Our application allows the user to save the image with detected potholes.

The user can also view the images in different filters like grayscale, blue, canny edge with the help of these filters the user can see how the application is actually detecting the potholes from the images.

We have us many image processing techniques for the detection of the potholes in the image like grey scaling, blurring, canny edge detection etc.

Objectives

- To create an application the will enable the user to detect the potholes in the image
- To show the user the steps which are used in the detection of the potholes

IMPLEMENTATION

The following project is a Image Processing based project which is partitioned in the following four components 1. Gray Scaling 2.Canny Edge Formation 3.Bounding Box Formation 4. Displaying Detected Potholes. The front end provides the user with a option to select the image int the form of .jpg or .png file from the computer and do the processing on that selected image.

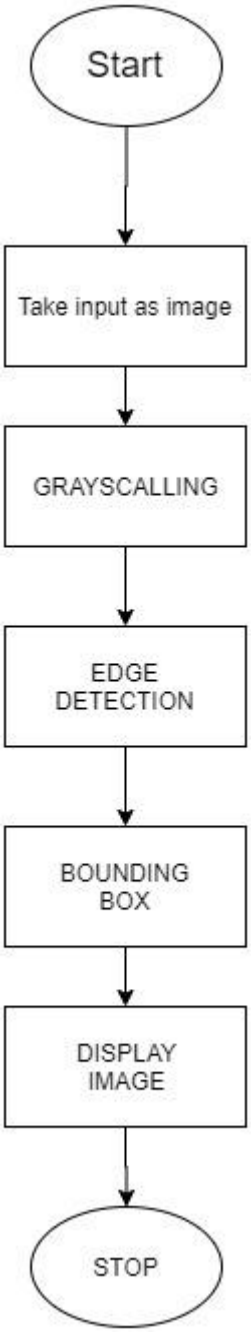


Figure.1. flowchart of image processing

FRONTEND

We have created the GUI using Matlab App Designer. The user is provided with several buttons to select the image and do the processing. First button helps user to select the image in the form of .png file or .jpg file. The rest of the buttons helps user to convert the image to gray scale and have a blurred effect on the image as it is required by our module to detect potholes. Other two buttons help form the edges using canny edge detection and finally to display the bounding box in the images.

Backend

The backend is also created using the matlab. The backend does the processing of the image once the image is uploaded in the application. The backend applies different filters on the image to detect potholes

Steps performed by model for detection of potholes.

1.GrayScalling

Gray Scale Image : Grayscale is a range of monochromatic shades from black to white. Therefore, a grayscale image contains only shades of gray and no color.

A grayscale (or greylevel) image is simply one in which the only colors are shades of grey. The reason for differentiating such images from any other sort of color image is that less information needs to be provided for each pixel.

Need for gray scaling

1. Signal to noise. For many applications of image processing, color information doesn't help us identify important edges or other features. There are exceptions. If there is an edge (a step change in pixel value) in hue that is hard to detect in a grayscale image, or if we need to identify objects of known hue (orange fruit in front of green leaves), then color information could be useful. If we don't need color, then we can consider it noise. At first it's a bit counterintuitive to "think" in grayscale, but you get used to it.
2. Complexity of the code. If you want to find edges based on luminance AND chrominance, you've got more work ahead of you. That additional work (and additional debugging, additional pain in supporting the software, etc.) is hard to justify if the additional color information isn't helpful for applications of interest.

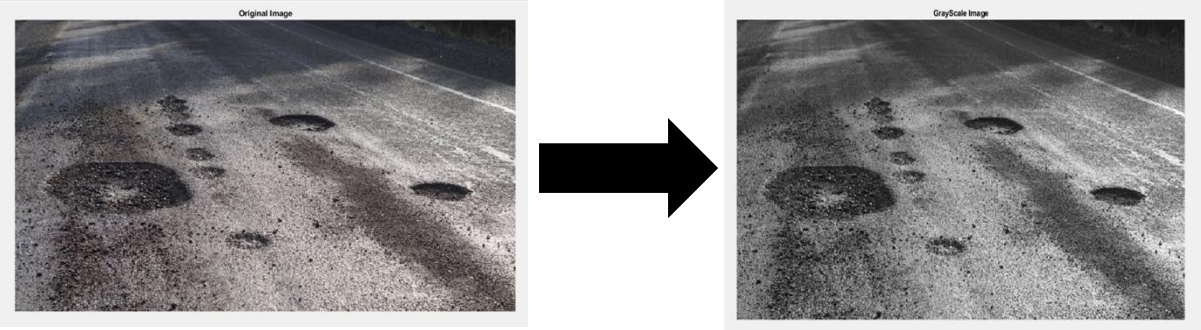


Figure 2:Gray Scale

3. CANNY EDGE FORMATION

Canny Edge Detector: The **Canny edge detector** is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986. Canny also produced a *computational theory of edge detection* explaining why the technique works.

The Canny edge detection algorithm is composed of 5 steps:

a. Noise reduction;
b. Gradient calculation;
c. Non-maximum suppression;
d. Double threshold;
e. Edge Tracking by Hysteresis.

After applying these steps, you will be able to get the following result:

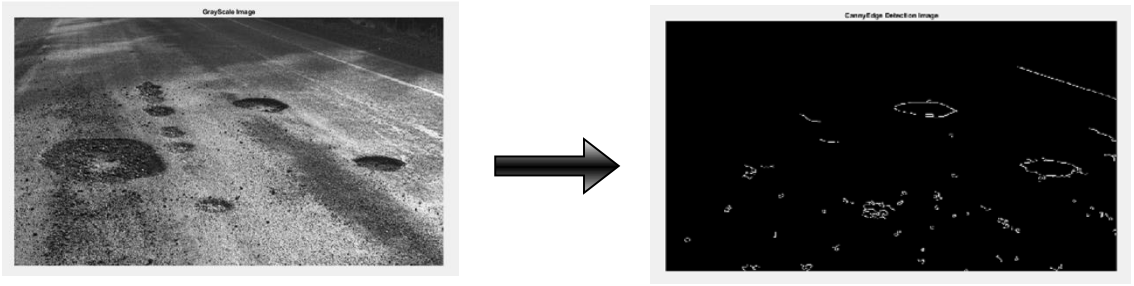


Figure 3:CannyEdge Detection

a. Noise Reduction :

Since the mathematics involved behind the scene are mainly based on derivatives (cf. Step 2: Gradient calculation), edge detection results are highly sensitive to image noise.

One way to get rid of the noise on the image, is by applying Gaussian blur to smooth it. To do so, image convolution technique is applied with a Gaussian Kernel (3x3, 5x5, 7x7 etc…). The kernel size depends on the expected blurring effect. Basically, the smallest the kernel, the less visible is the blur.

The equation for a Gaussian filter kernel of size $(2k+1)\times(2k+1)$ is given by:

$$H_{ij} = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(i-(k+1))^2+(j-(k+1))^2}{2\sigma^2}\right); 1 \le i,j \le (2k+1)$$

Figure 4. Gaussian filter kernel equation

b. Gradient Calculation

The Gradient calculation step detects the edge intensity and direction by calculating the gradient of the image using edge detection operators.

Edges correspond to a change of pixels' intensity. To detect it, the easiest way is to apply filters that highlight this intensity change in both directions: horizontal (x) and vertical (y)

When the image is smoothed, the derivatives ***Ix*** and ***Iy*** w.r.t. ***x*** and ***y*** are calculated. It can be implemented by convolving ***I*** with Sobel kernels ***Kx*** and ***Ky***, respectively:

$$K_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, K_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}.$$

Figure 5. Sobel filters for both direction (horizontal and vertical)

Then, the magnitude ***G*** and the slope $\boldsymbol{\theta}$ of the gradient are calculated as follow:

$$|G| = \sqrt{I_x^2 + I_y^2},$$
$$\theta(x, y) = arctan\left(\frac{I_y}{I_x}\right)$$

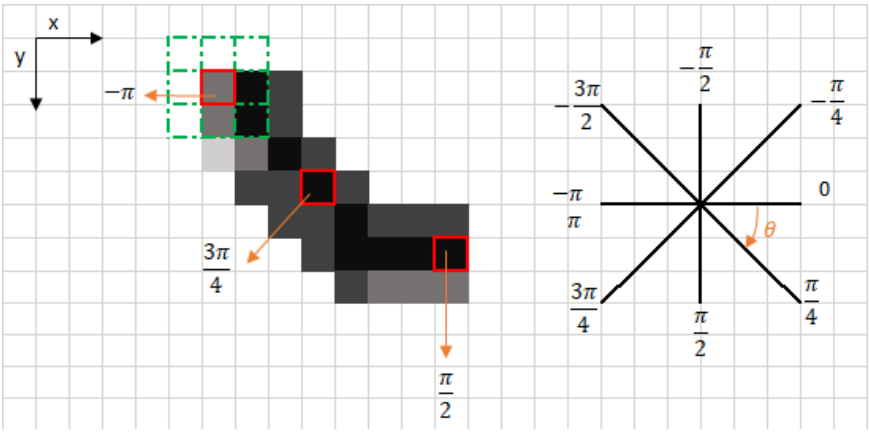
Figure 6. Gradient intensity and Edge direction

c. Non-Maximum Suppression

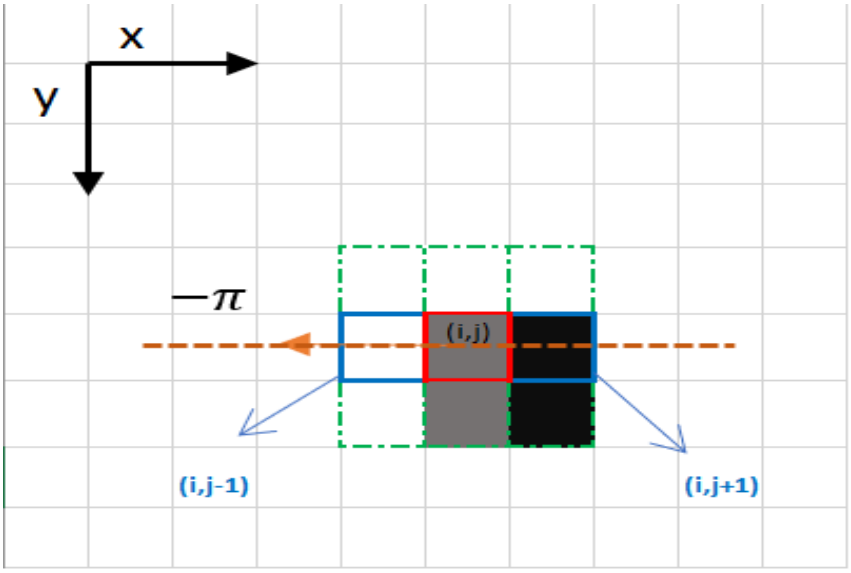
Ideally, the final image should have thin edges. Thus, we must perform non-maximum suppression to thin out the edges.

The principle is simple: the algorithm goes through all the points on the gradient intensity matrix and finds the pixels with the maximum value in the edge directions.

Let's take an easy example:



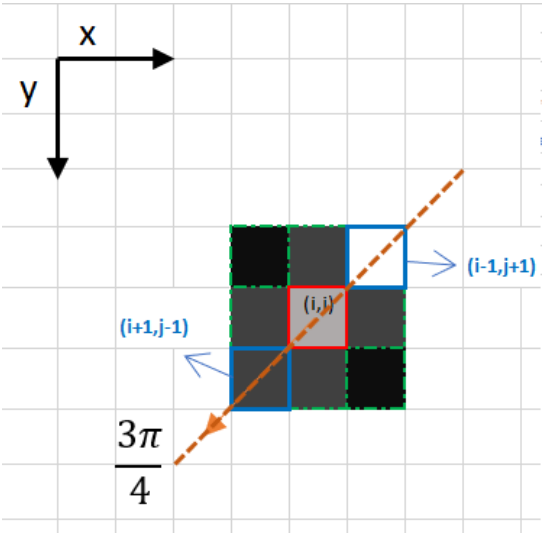
The upper left corner red box present on the above image, represents an intensity pixel of the Gradient Intensity matrix being processed. The corresponding edge direction is represented by the orange arrow with an angle of –pi radians (+/-180 degrees).



Focus on the upper left corner red box pixel

The edge direction is the orange dotted line (horizontal from left to right). The purpose of the algorithm is to check if the pixels on the same direction are more or less intense than the ones being processed. In the example above, the pixel ***(i, j)*** is being processed, and the pixels on the same direction are highlighted in blue ***(i, j-1)*** and ***(i, j+1).*** If one those two pixels are more intense than the one being processed, then only the more intense one is kept. Pixel ***(i, j-1)*** seems to be more intense, because it is white (value of 255). Hence, the intensity value of the current pixel ***(i, j)*** is set to 0. If there are no pixels in the edge direction having more intense values, then the value of the current pixel is kept.

Let's now focus on another example:



In this case the direction is the orange dotted diagonal line. Therefore, the most intense pixel in this direction is the pixel ***(i-1, j+1).***

Let's sum this up. Each pixel has 2 main criteria (edge direction in radians, and pixel intensity (between 0–255)). Based on these inputs the non-max-suppression steps are:

- Create a matrix initialized to 0 of the same size of the original gradient intensity matrix;
- Identify the edge direction based on the angle value from the angle matrix;
- Check if the pixel in the same direction has a higher intensity than the pixel that is currently processed;
- Return the image processed with the non-max suppression algorithm.

d. Double threshold

The double threshold step aims at identifying 3 kinds of pixels: strong, weak, and non-relevant:

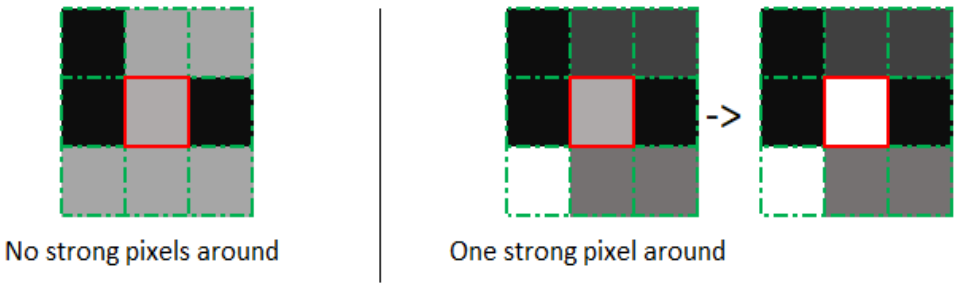
- Strong pixels are pixels that have an intensity so high that we are sure they contribute to the final edge.
- Weak pixels are pixels that have an intensity value that is not enough to be considered as strong ones, but yet not small enough to be considered as non-relevant for the edge detection.
- Other pixels are considered as non-relevant for the edge.

Now you can see what the double thresholds holds for:

- High threshold is used to identify the strong pixels (intensity higher than the high threshold)
- Low threshold is used to identify the non-relevant pixels (intensity lower than the low threshold)
- All pixels having intensity between both thresholds are flagged as weak and the Hysteresis mechanism (next step) will help us identify the ones that could be considered as strong and the ones that are considered as non-relevant.

e. Edge Tracking by Hysteresis

Based on the threshold results, the hysteresis consists of transforming weak pixels into strong ones, if and only if at least one of the pixels around the one being processed is a strong one, as described below:



4. Bounding Box Formation

Bounding boxes are imaginary boxes that are around objects that are being checked for collision, like pedestrians on or close to the road, other vehicles and signs. There is a 2D coordinate system and a 3D coordinate system that are both being used.

In digital image processing, the bounding box is merely the coordinates of the rectangular border that fully encloses a digital image when it is placed over a page, a canvas, a screen or other similar bi-dimensional background.

Figure 7:Bounding Box

We have used insertShape() function for drawing the rectangle on each edge detected after the canny edge detection .Whenever the edge is detected on the image the program automatically draws the image around it.

TECHNOLOGY STACK

Front End	App Designer Library Compiler Simulink Application Compiler
Back End	Matlab Matlab Coder

Table 1: Technology Stack

Sample Screenshots

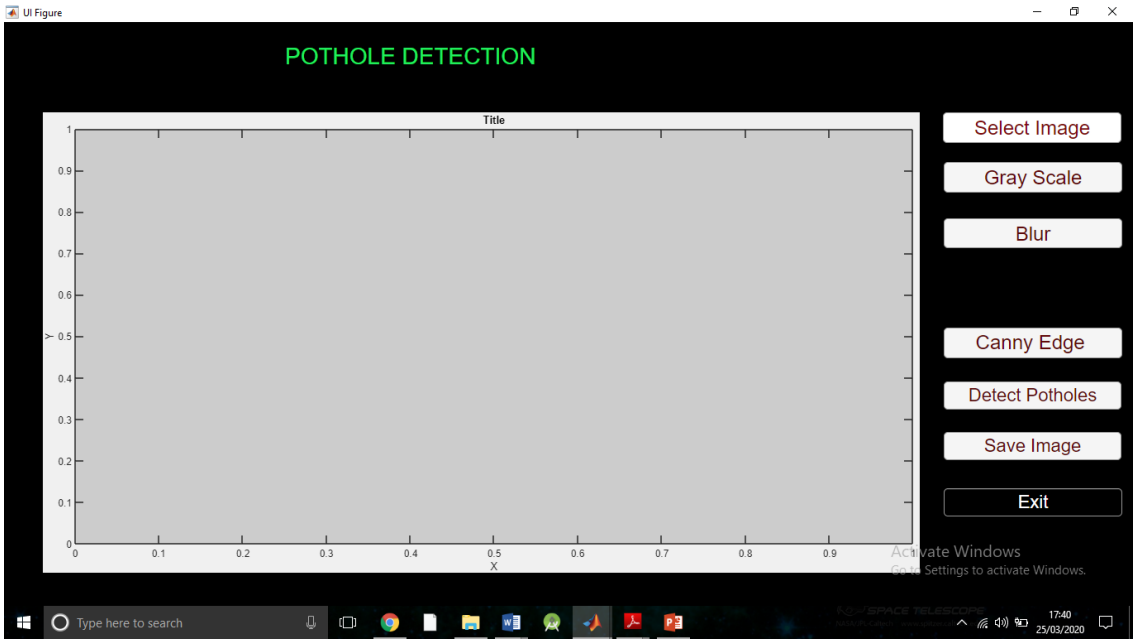


Figure 8: GUI

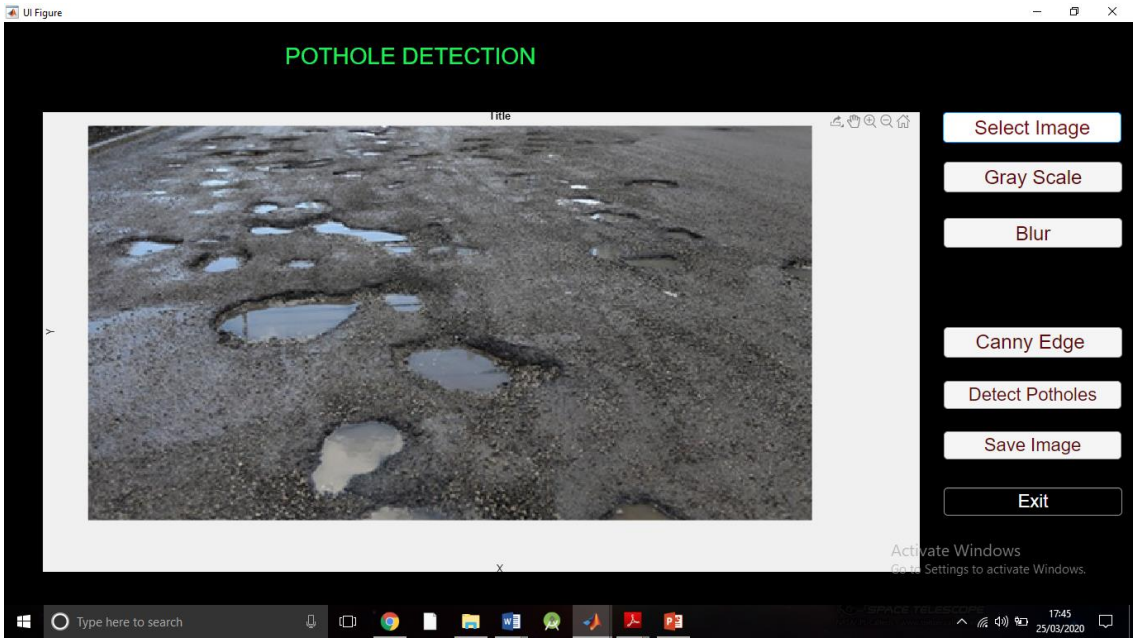


Figure 9: Uploaded Image

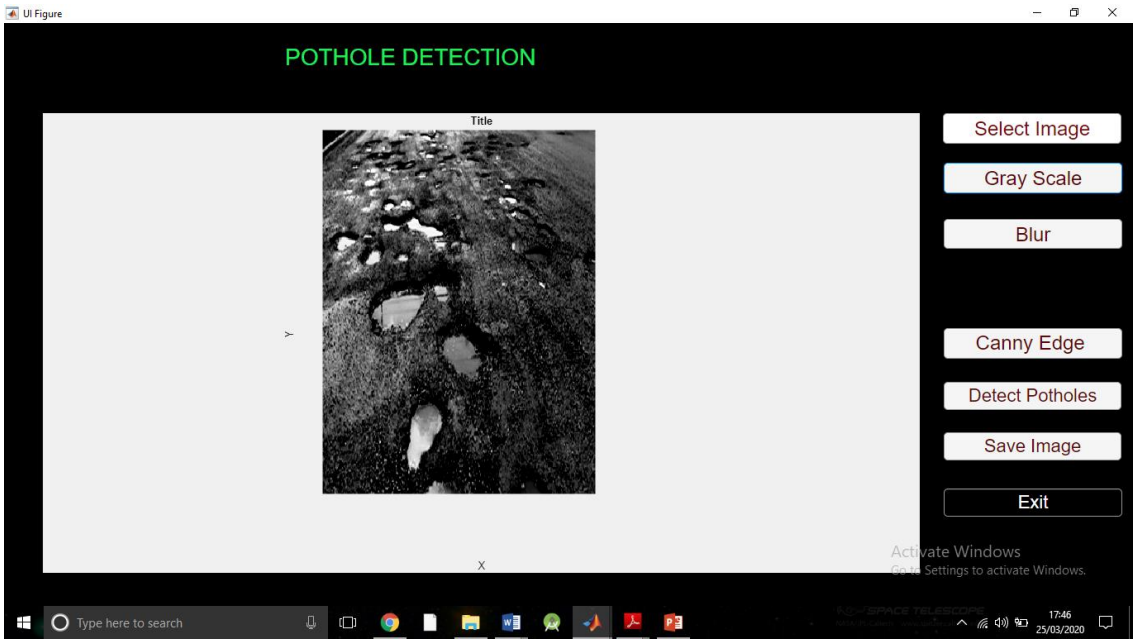


Figure 10: GrayScale Image

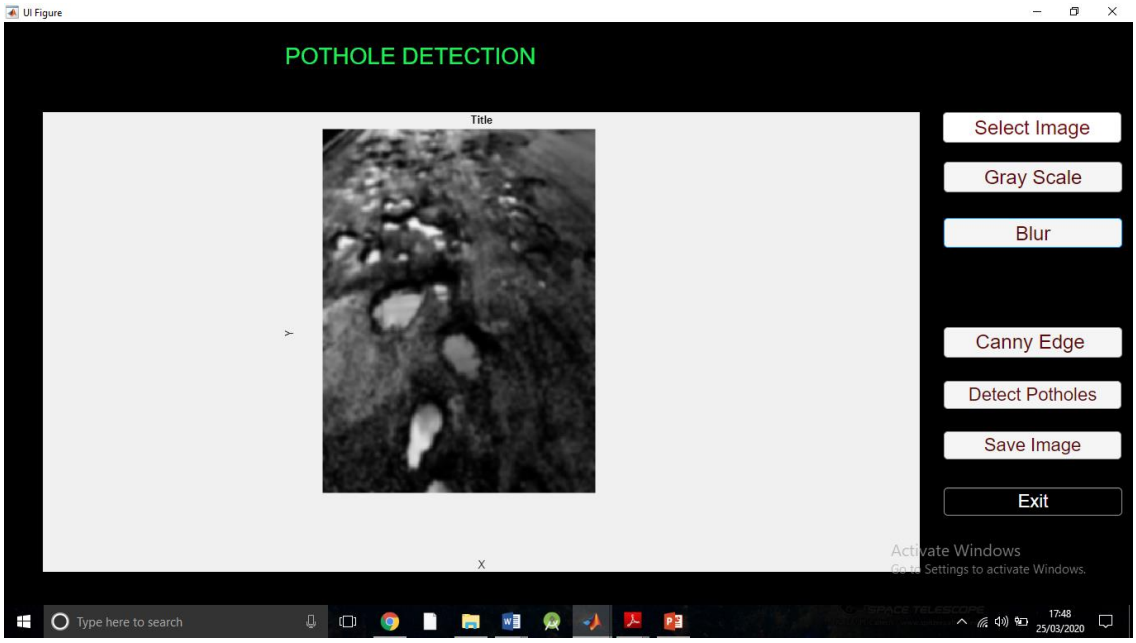


Figure 11: Blur Image

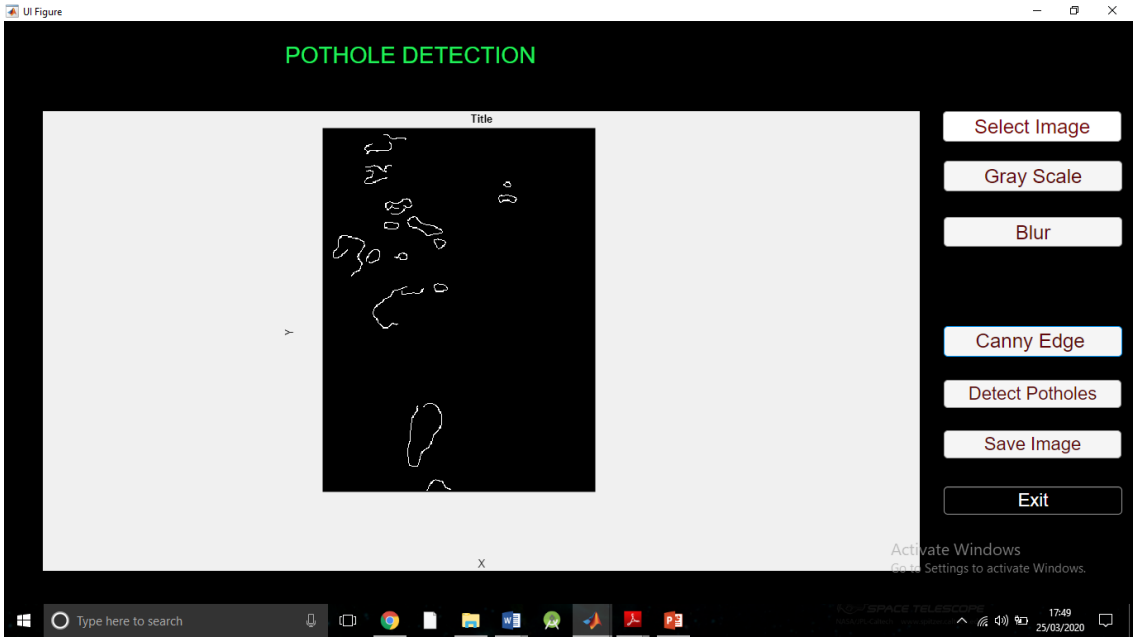


Figure 12: Canny Image

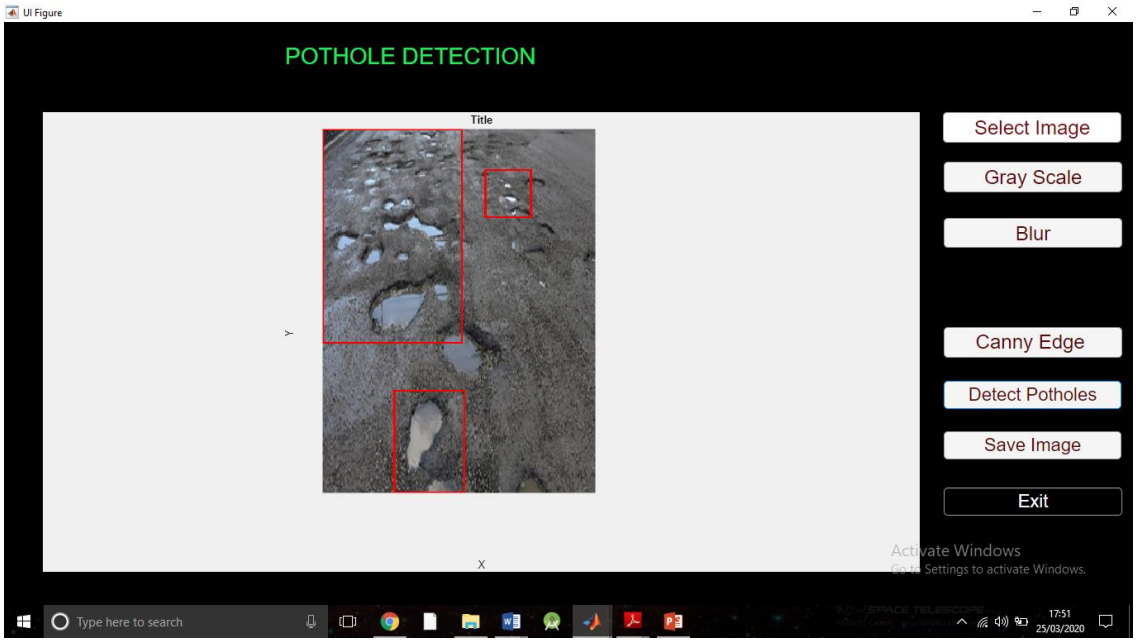


Figure 13: Detected Potholes

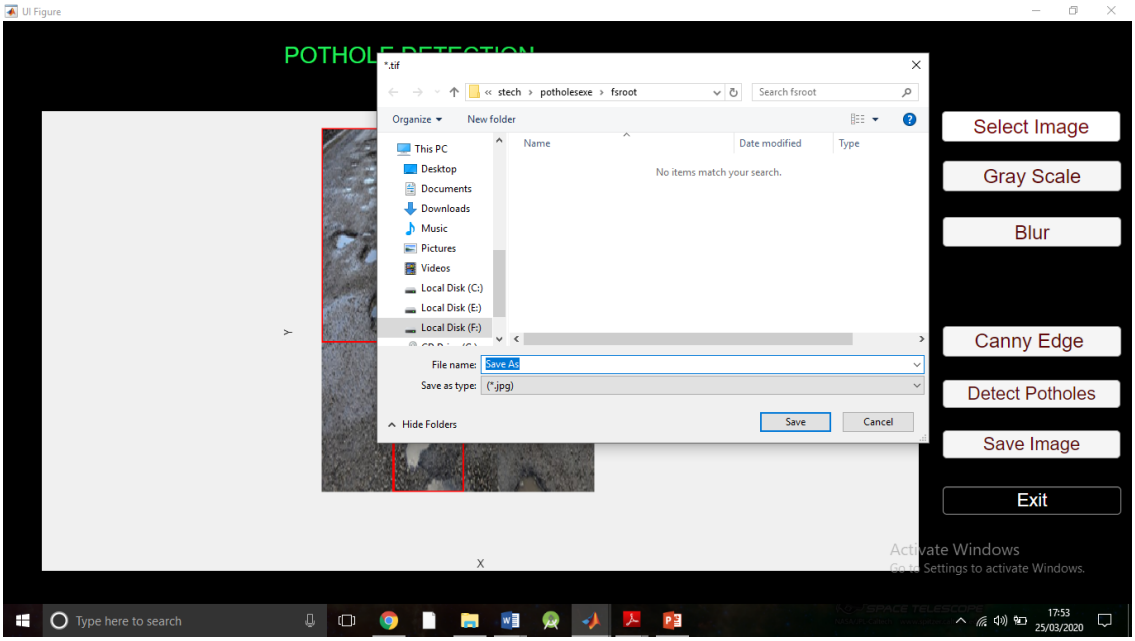


Figure 14: Save Image

Result

Our project has resulted into application which requires user to have an image as an input the user can upload the image in the application and then run the application to detect the potholes in the image. Our application also provides the user with the functionality to save the image with the detected potholes.

FUTURE SCOPE

In the future we have decided to add some other functionality to our application like detecting potholes from the videos as input. We have also decided to add a function which will enable the user to send the image to the authorities which are responsible for removal of potholes in city. We are also going to add a location feature in our application so that the user can send the location of the image to the authorities.

CONCLUSION

The project is completed under the given time the application is now able to recognize the potholes from the image which is given to it as an input. Our application also provide the user with the functionality to save the image. We have also provided different functionality to the user so that he/she can see how actually the potholes are detected from the image by our application

REFERENCES

[1]	Dewiani Djamaluddin, Andani Achmad, Rivanto Parung, Prototype of vehicles potholes detection based blob detection method, *Journal of Theoretical and Applied Information Technology, 15th June 2017. Vol.95. No 11*.
[2]	Sudarshan, R. (2007) "A Pothole Detection System".
[3]	J. Eriksson, L. Girod and B. Hull, "The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring,"
[4]	Samarth, B, Grayscaling in image processing, https://samarthbhargav.wordpress.com/2014/05/05/image-processing-with-pythonrgb-to-grayscale-conversion/
[5]	Syed Mohammad Abid, Hasan Kwanghee Ko, Depth edge detection by image-based smoothing and morphological operations, Journal of Computational Design and Engineering, July 2016 Volume 3, Issue 3.
[6]	Suganya, Menaka, Various Segmentation Techniques in Image Processing, International Journal of Innovative Research in Computer and Communication Engineering