CHAPTER-1

1.1 INTRODUCTION

Plant diseases have turned into a nightmare as it can cause significant reduction in both quality and quantity of agricultural products, thus negatively influence the countries that primarily depend on agriculture in its economy. Consequently, detection of plant diseases is an essential research topic as it may prove useful in monitoring large fields of crops and thus automatically detects the symptoms of diseases as soon as they appear on plant leaves.Therefore, looking for a fast,less expensive and accurate method to detect plant disease cases is of great realistic significance. Studies shows that image processing can successfully be used as a disease detection mechanism. The proposed approach is image-processing based and is composed of three main phases. In the first phase we create a color transformation structure for the RGB leaf image and then, in the second phase we are traversing the pixels that are affected on the leaf, in the third phase, we are detecting and plotting them separately.

An image that we see with our eyes is analogous in nature to the image captured via digital camera.To identify the affected spots or region of a leaf and displaying it using python libraries. We are using the pixel-by-pixel approach to detect the affected spots of a leaf.

The technology and functionality that we are used in this project is a python interpreter and modules such as matplotlib, OpenCV and NumPy. The process of identifying the affected area of a leaf is done by using the color requirements such as HSV values. Farmer’s economic growth depends on the quality of products that they produce which relies on plant’s growth they get. The problem that we are dealing with the affected leaf identification is displaying the affected parts of the leaf efficiently. This problem can be solved by using this project efficiently to report the farmers that the region of leaf is affected in their agriculture efficiently to report the farmers that the region of leaf is affected in their agricultural fields.

CHAPTER-2

2.1 LITERATURE REVIEW

Leaves are the natural creation and may vary in terms of their physical appearances. This project presents detection techniques that can be used for plant leaf affected area identification. The images that is already possessed with RGB mode, the desired colors are collected using our program approach and functionality of the code. The program will extract the image of affected area of the leaf. Then the image is converted into binary image.

The project depicts the technique of infected regions using image processing. The area, that is affected is captured from the leaf and plotted it to the separate graph. This process can be done by using conversion of image from RGB into HSV format.

The detection of affected areas or infected regions involves some steps are RGB image acquisition. Converting the input image from RGB to HSV format. Masking and removing the green pixels. This method is unsupervised, non-deterministic and iterative, then separating the infected parts from the leaf.

CHAPTER-3

3.SYSTEM ENGINEERING

3.1 EXISTING SYSTEM

- Plant recognition and infected area analysis may be stream lined by an image of a complete, isolated leaf as an initial input. Background and leaf markers are created using color.
- Two approaches investigated: watershed and graph-cut and results compared.
- There are the applications is to extract important features from image data which a description, interpretation or understanding of the scene can be provided by the machine.
- But there is no proper processing can be defined as processing or alternating an existing image in a desired manner.

3.2DISADVANTAGES OF EXISTING SYSTEM

- The output will be predicted slower.

- The compiled time taken by the existing system is more than the proposed system.
- It is not a pixel-by-pixel approach mechanism.

3.3PROPOSED SYSTEM

- The powerful image processing system is the human brain together with the eye.

- The user will send the images according to the specifications they will be modified.
- To process these things we needed to proceed with the python programming language, that has a large amount of modules, that are allows us to make use of them.
- Then image processing techniques are applied to these images to extract useful features

which will be required for further analysis. The modules we used in this project

3.4 ADVANTAGES OF THE PROPOSED SYSTEM

- The detection of the affected parts become easier.
- This process used the simple modules like OpenCV, matplotlib which are available to everyone on the internet.
- This project will identify the infected leaf and shows the infected region.

3.5 DOMAIN DESCRIPTION

The technology and functionality that we are used in this project is a python interpreter and modules such as matplotlib, OpenCV and NumPy. The process of identifying the affected area of a leaf is done by using the color requirements such as HSV values.

3.6 PROBLEM STATEMENT

Farmer's economic growth depends on the quality of products that they produce which relies on plant's growth and the yield they get. Manual detection of plant affected using leaf images is tedious job. Hence it is required to develop the computational methods which will make a process affected area detection. The problem that we are dealing with the affected leaf identification is displaying the affected parts of the leaf efficiently. This problem can be solved by using this project efficiently to report the farmers that the region of leaf is affected in their agricultural fields.

3.7 PROPOSED SOULTION

The powerful image processing system is the human brain together with the eye. The user will sends the images according to the specifications they will be modified. To

process these things, we needed to proceed with the python programming language that has a large number of modules, that are allows . Then image processing techniques are applied to these images to extract useful features which will be required for further analysis. The modules we used in this project is python-OpenCV, matplotlib, NumPy.

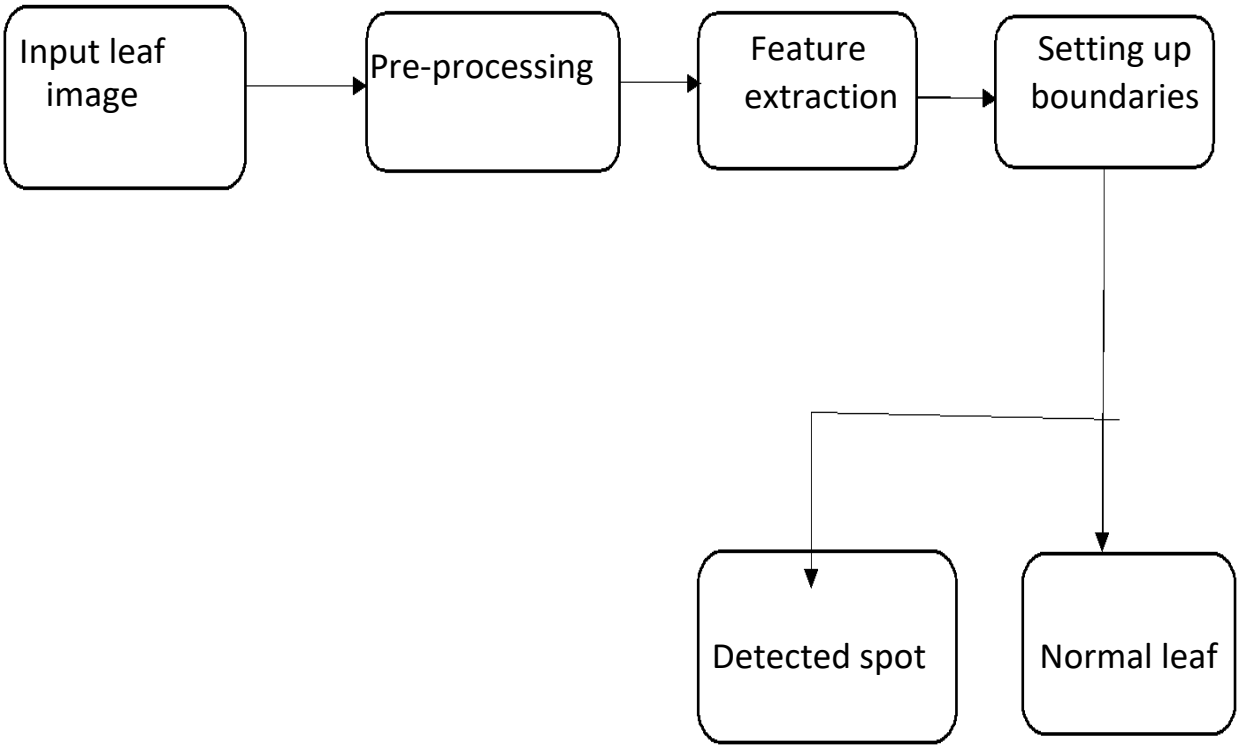
3.8 MOTIVATION

Leaves are the natural creation and may vary in terms of their physical appearances. This project presents detection techniques that can be used for plant leaf affected area identification. The images that is already possessed with RGB mode, the desired colors are collected using our program approach and functionality of the code. The program will extract the image of affected area of the leaf. Then the image is converted into binary image. The project depicts the technique of infected regions using image processing. The area, that is affected is captured from the leaf and plotted it to the separate graph. This process can be done by using conversion of image from RGB into HSV format.

CHAPTER-4

4.SYSTEM DESIGN

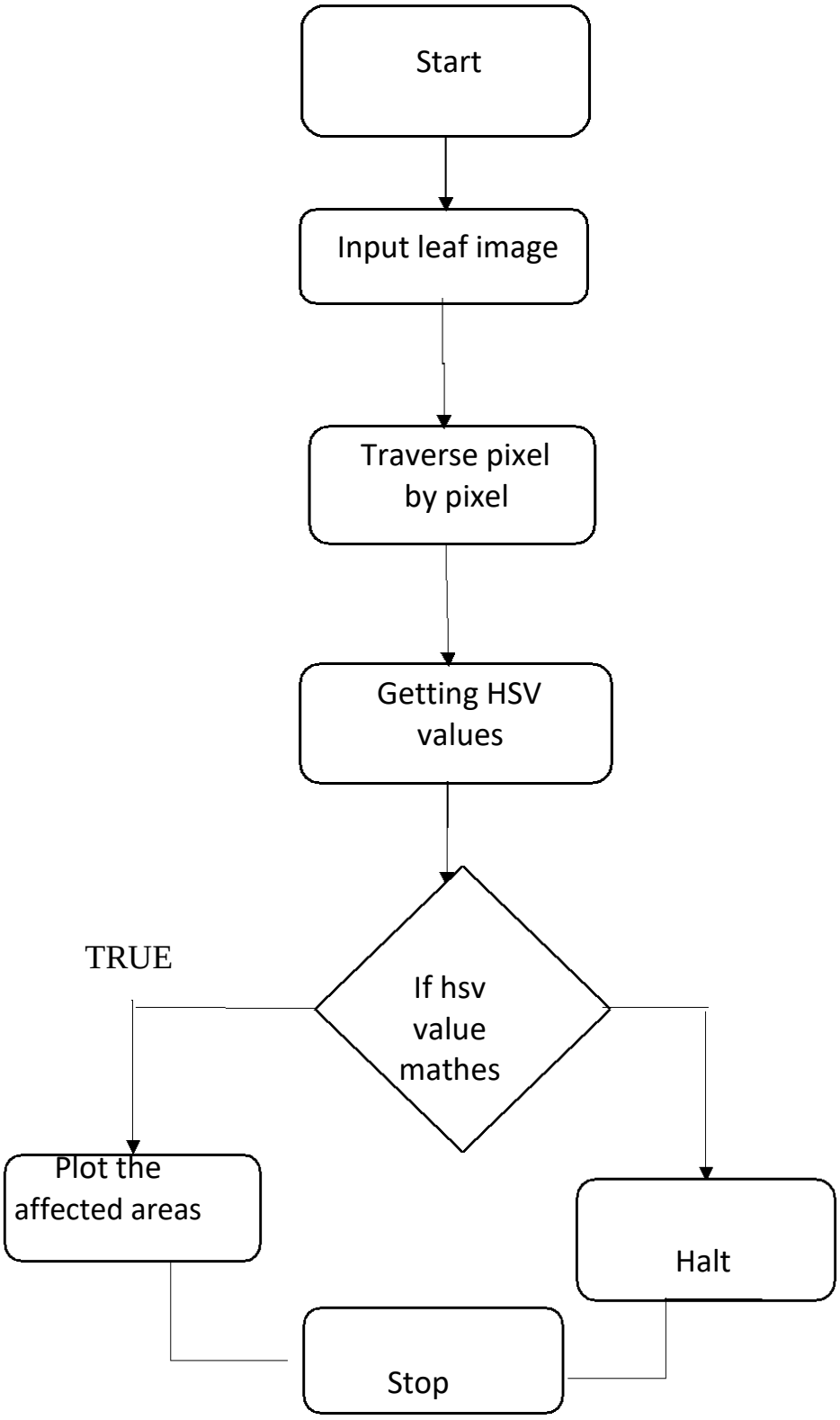
4.1 SYSTEM ARCHITECTURE



4.2 DATA FLOW DIAGRAM

4.2.1 ACTIVITY DIAGRAM

In this activity diagram first we have collect a large dataset of both infected and non infected leaf. We give leaf image as input convert the input leaf image that possessed with RGB mode into gray scale image.By converting into gray scale measuring the pixels of the infected region,it traverse pixel by pixel it convert gray scale value to get the HSV values if condition is used to true or false if HSV value maths value is trtue plot affected areas if the conditions is false it gives halt and then it stops.


Start
Input leaf image
Traverse pixel by pixel
Getting HSV values
If hsv value mathes
TRUE
Plot the affected areas
Halt
Stop

4.3 SYSTEM REQUIREMENTS

4.3.1 SOFTWARE REQUIREMENTS

1. python interpreter
2. operating system-windows/ubuntu/linux
3. technology used: python
4. libraries (matplotlib, numpy, opencv)

4.3.2 HARDWARE REQUIREMENTS

- 12MB RAM
- processor i3
- hard disk- 10GB

4.4 MODULES:

- Color transformation
- Traversing pixels
- Detecting and plotting

Color transformation

 In the first module, we take the input of the leaf which have collected large dataset of both infected and not-infected leaf. The input leaf which we had taken will in green color that is in the RBG color.

2.Traversing pixels

 In the second module, after converting the leaf from RBG to grayscale. By converting into grayscale, the grayscale measures the leaf by pixels by pixels

From the grayscale conversion measuring pixels of the infected region will be identified by using HSV convertion then output is compared. By the HSV values the infected region in the leaf will be identified.

3. Detecting and plotting

In the third module, by using HSV values the affected region will be indentified and plotted. Traversing by pixel by pixel using HSV values the affected area will be detected and plotted by the scale

4.5 MODULE DESCRIPTION:

4.5.1 OPENCV

- Opencv(Open source computer vision) is a tool, that helps in image processing.
- (Open source compute vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source BSD license.
- Advance vision research by providing not only open but also optimized code for basic vision infrastructure. No more reinventing the wheel.
- Disseminate vision knowledge by providing a common infrastructure .
- Advance vision-based commercial applications by making portable, performance-optimized code available for free – with a license that did not require code to be open or free itself.

The first alpha version of OpenCV was released to the public at the IEEE Conference on Computer Vision and Pattern Recognition in 2000, and five betas were released between 2001 and 2005. The first 1.0 version was released in 2006. A version 1.1 "pre-release" was released in October 2008.The second major release of the OpenCV was in October 2009. OpenCV 2 includes major changes to the C++ interface, aiming at easier, more type-safe patterns, new functions, and better implementations for existing ones in terms of performance (especially on multi-core systems).

- In August 2012, support for OpenCV was taken over by a non-profit foundation OpenCV.org, which maintains a developer and user site.
- On May 2016, Intel signed an agreement to acquire Itseez, a leading developer of OpenCV.
- OpenCV is written in c++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. There are bindings in python, and java. Wrappers in other languages such as c#,perl, ch, haskell, and ruby have been developed to encourage adoption by a wider audience.
- Since version 3.4, **OpenCV.js** is a java script binding for selected subset of OpenCV functions for the web platform.
- All of the new developments and algorithms in OpenCV are now developed in the C++ interface.

The library has than 2500 optimized algorithms, which includes a comprehensive set of both computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, more classify human actions in videos, track camera movements, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications.A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and STL containers.

4.5.2 MATPLOTLIB

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

Matplotlib consists of several plots like line, bar, scatter, histogram etc.

- **Matplotlib** is a plotting library for the python programming language and its numerical mathematics extension numpy. It provides an object oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxpython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like openGL), designed to closely resemble that of MATLAB, though its use is discouraged. Scipy makes use of Matplotlib.
- Matplotlib was originally written by Jhon.D.Hunter, has an active development community, and is distributed under a BSD style licence Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012, and further joined by Thomas Caswell.
- As of 23 June 2017, matplotlib 2.0.x supports Python versions 2.7 through 3.6. Matplotlib 1.2 is the first version of matplotlib to support Python 3.x. Matplotlib 1.4 is the last version of Matplotlib to support Python 2.6
- Several toolkits are available which extend Matplotlib functionality. Some are separate downloads, others ship with the Matplotlib source code but have external dependencies.
- Basemap: map plotting with various map projections, coastlines, and political boundaries
- Cartopy: a mapping library featuring object-oriented map projection definitions, and arbitrary point, line, polygon and image transformation capabilities. (Matplotlib v1.2 and above)
- Excel tools: utilities for exchanging data with Microsoft exel.
- GTK tools: interface to the GTK+ library
- QT interface
- Mplot3d: 3-D plots

a) **Image (RGB) load:** The images of the plant leaf are captured through the camera, this image

is in RGB (Red, Green and Blue) form, color transformation structure for the leaf image is created, and then an independent color space transformation for the color transformation structure is applied.

b) Pre-processing: To remove noise in image or other object removal, pre-processing techniques is considered. Image clipping i.e. cropping of the leaf image to get the interested image region. Image smoothing is done using the smoothing filter. Image enhancement is carried out for increasing the contrast. The RGB images into the grey images using color conversion using equation(x) =0.2989*R + 0.5870*G + 0.114*B Then the
histogram equalization which distributes the intensities of the images is applied on the image to enhance the plant disease images. The cumulative distribution function is used to distribute intensity values

c) Segmentation: Segmentation of leaf image is important while processing image from that Segmentation
means partitioning of image into various part of same features or having some similarity. The segmentation can be done using various methods like Otsu' method, k-means clustering.

d) Feature extraction: Feature extraction plays an important role for classification of an image. In many application feature extraction of image is used. Color, texture, morphology, edges etc. are the features which can be used in plant disease classification, texture means how the color is distributed in the image, the roughness, hardness of the image. In this paper considers color, texture and morphology as a feature for disease detection. They have found that morphological result gives better result than the other features. It can use for identify the infected plant leaf of classification plant image.

CHAPTER-5

CODING AND IMPLEMENTATION

5.1 CODING

5.1.1 CODE:

```
import cv2
from matplotlib import pyplot as plt
import numpy as np
#import numpy as np
count,count1=0,0
arr=[]
brr=[]
img=cv2.imread("C:\\Users\\ASUS\\Desktop\\leaf_dataset\\test.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.startWindowThread()
#cv2.namedWindow("leaf detection")
cv2.waitKey(0)
cv2.destroyAllWindows()
gray=cv2.resize(gray,(400,400))
#cv2.imshow("sathish",z)
rows,cols=gray.shape

for i in range(rows-1):
for j in range(cols-1):
   if gray[i,j]!=0:
```

```
        count+=1
    else:
    count1+=1
    if count!=0 and count1!=0:
        print("it is not a healthy leaf")
    else:
        print("it is a healthy leaf")
    plt.imshow(gray,cmap="gray",interpolation="bicubic")
    plt.xlabel("X-axis")
    plt.ylabel("Y-axis")
    #x,y=np.load("sathish.npy"),np.load("haritha.npy")
    plt.show()
```

5.2 IMPLEMENTATION & TESTING

5.2.1 INPUT AND OUTPUT

5.2.1 INPUT

The input is, we have collected large dataset of both infected and non-infected leaf. Some of the input we taken has shown below.

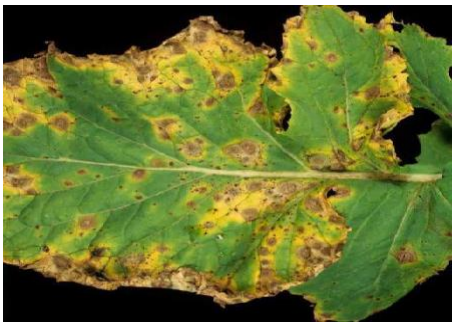


Fig 4.4.1 Set of leaf input images.

5.2.2 PROCESS

1. Input the leaf image

Fig 4.4.2 Leaf input.

2. Convert the input leaf image that possessed with RGB mode into gray scale image.

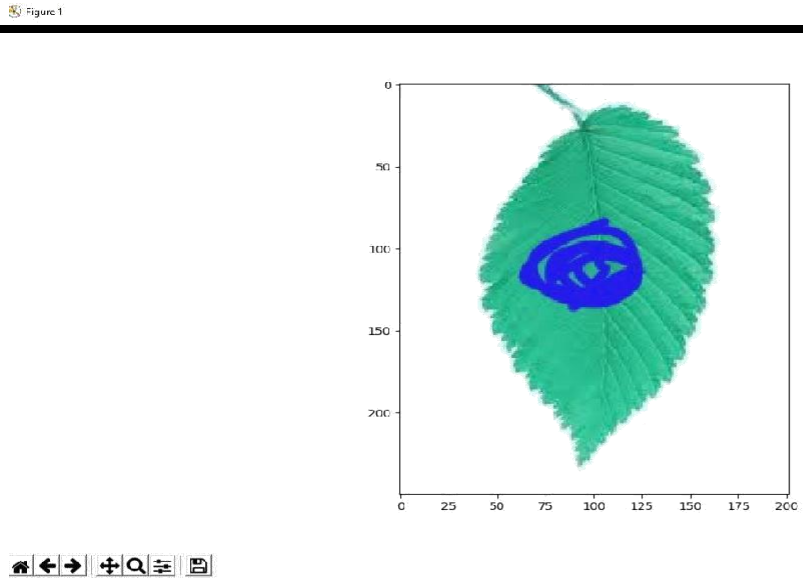


Fig 4.4.3 Gray scale convertion

3. By converting into grayscale, measuring the pixels of the infected region and by HSV convertion the output is compared.

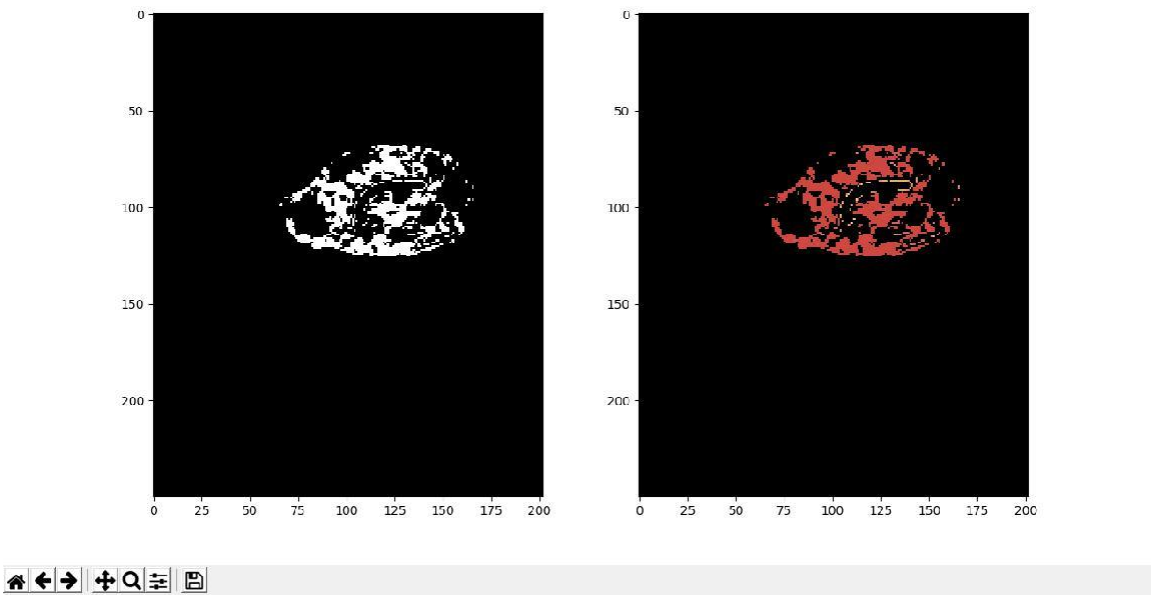


Fig 4.4.4 Infected region -output

CHAPTER 6

RESULT

6.1Comparison of proposed system and existing system

Table 6.1 : Comparision

Existing system	Proposed system
The output will be predicted slower	The output will be predicted faster then existing system
The compiled time taken by the existing system is more than the proposed system	The compiled time taken by the proposed system is less than the existing system
It is not a pixel by pixel approach mechanism	It is a pixel by pixel approach mechanism

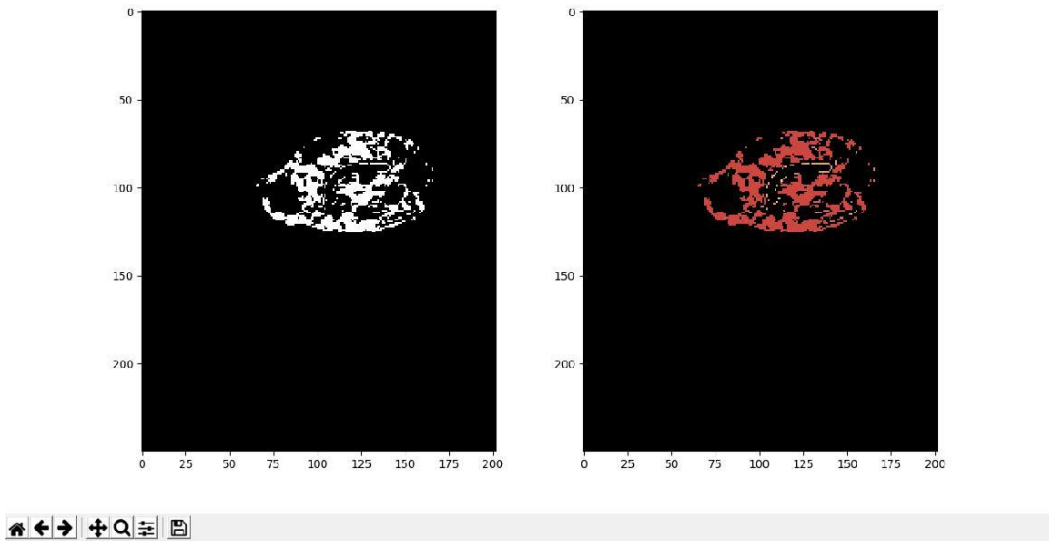


Fig 6.1 Output image -result.

CHAPTER-7

CONCLUSION

Plant disease detection attracts significant attention in the field of agriculture where image based disease detection plays an important role. To improve the yield of plants, it is necessary to detect the onset of diseases in plants and advice the farmers to act based on the suggestions. It is very difficult to monitor the plant diseases manually. It requires tremendous amount of work, expertize in the plant diseases, and also require the excessive processing time. Hence, image processing is used for the detection of plant diseases. Smart farming and precision agriculture involve the integration of advanced technologies into existing farming practices in order to increase production efficiency and the quality of agricultural products. As an added benefit, they also improve the quality of life for farm workers by reducing heavy labor and tedious tasks.

7.1 FUTURE ENHANCEMENTS

We are yet to develop the graphical user interface by using tensorflow.the number of leaves can be displayed simultaneously and the selection of the leaf can be done by using tensorflow.the graphical user RRinterface will makes the user interactive very easily.indeed, future work involves implementation of other different techniques, methods and algorithms and investigation their performance using various databases with different covariate factors.

REFERENCES

[1] Khirade, Sachin D., and A. B. Patil. "Plant disease detection using image processing." *2015 International conference on computing communication control and automation*. IEEE, 2015.

[2] Khater, Mohga, Alfredo de la Escosura-Muñiz, and ArbenMerkoçi. "Biosensors for plant pathogen detection." *Biosensors and Bioelectronics* 93 (2017): 72-86.

[3] Ashourloo, Davoud, et al. "An investigation into machine learning regression techniques for the leaf rust disease detection using hyperspectral measurement." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 9.9 (2016): 4344-4351.