Leaf Disease Detection Using Traditional Image Processing Techniques

A PROJECT REPORT SUBMITTED BY

B.W.D.U. BANDARA
(S / 20 / 314)

in partial fulfillment of the requirement for the course of
CSC 3141 Image Processing Laboratory

to the
Department of Statistics and Computer Science

of the

FACULTY OF SCIENCE
UNIVERSITY OF PERADENIYA
SRI LANKA
2024

DECLARATION

I do hereby declare that the work reported in this project report was exclusively carried out by me under the supervision of **Prof. Amalka Pinidiyaarachchi** . It describes the results of my own independent work except where due reference has been made in the text. No part of this project report has been submitted earlier or concurrently for the same or any other degree.

Date: ……. ……

Signature of the Candidate

Certified by:

1. Instructor: **Mr. Kansaji Kotuwage**
   Date: ……. Signature:…….

2. Supervisor: **Prof. Amalka Pinidiyaarachchi**
   Date: ……. Signature:…….

3. Head of the Department: **Dr. Hakim Usoof**

   Date: ……. Signature:…….

ABSTRACT

Leaf disease detection is critical in agriculture for monitoring crop health and improving productivity. This project presents a conventional image processing method for identifying and measuring diseased areas in leaf photos using OpenCV. The system begins with Gaussian-based low-pass filtering to reduce noise, followed by segmentation in the HSV color space to isolate the leaf from the background. The predominant color (usually green or yellow) is used to identify healthy areas, and diseased areas are subtracted from these healthy zones. Contour analysis is applied to determine the extent of affected areas, and the final output presents the disease coverage as a percentage of the total leaf area. For early disease detection in agricultural settings, this portable, non-learning-based approach offers a productive and economical substitute.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to **Prof. Amalka Pinidiyaarachchi**, whose lectures and theoretical instruction in image processing provided the foundational knowledge that guided the direction of this project.

I am also deeply thankful to **Mr. Kansaji Kotuwage** for his practical guidance, continuous support, and encouragement throughout the course of this work. His feedback and mentorship were crucial to the implementation of this project.

Finally, I extend my appreciation to the **Department of Statistics and Computer Science**, Faculty of Science, University of Peradeniya, for providing the resources and academic environment essential for the successful completion of this project

TABLE OF CONTENTS

TABLE OF FIGURES

CHAPTER 01

INTRODUCTION

1.1. INTRODUCTION

In agriculture, early detection of plant diseases plays an important and crucial role in preventing yield loss and to ensure crop health. Traditional methods, where farmers check plants manually, are slow, hard work, and can often be inaccurate. Therefore, to significantly enhance the agricultural productivity, automating the process using image processing techniques can be noted.

This project aims to build a system that focuses on developing a disease detection system using classical image processing techniques. The system aims to identify the infected regions of a leaf by analyzing digital images captured under natural conditions. This technique is totally dependent on color space transformations, image filtering, and contour-based segmentation, in contrast to machine learning or deep learning techniques. The algorithm isolates the leaf and healthy areas using preprocessing methods like HSV color space segmentation and Gaussian filtering for noise reduction. By evaluating the remaining infected area, the system provides a visual and quantitative understanding of disease spread, which can support farmers in timely decision-making.

1.2. PROBLEM STATEMENT

In order to minimize crop losses and guarantee food security, timely and precise identification of plant leaf diseases is crucial in agriculture. Traditionally, farmers and agricultural workers detect leaf diseases through manual inspection, which is labor-intensive, time-consuming, and often inconsistent due to human subjectivity. Furthermore, manual diagnosis is not scalable for monitoring numerous crops over wide geographic areas or for large plantations.

While advanced deep learning techniques have shown promise in automated disease detection, such methods often require large labeled datasets, significant training time, and high-performance computing resources—making them less suitable for deployment in resource-constrained agricultural environments.

Consequently, there is a growing need for a lightweight, low-cost, and effective image-based system that can detect leaf diseases using classical computer vision techniques. Accurately separating the leaf from the image, identifying healthy and diseased areas based on color traits, and providing quantitative data regarding the degree of infection are the main obstacles. An efficient and reproducible method must be developed that achieves these goals without relying on data-driven models.

1.3. SOLUTION

This project suggests a conventional image processing pipeline using OpenCV to identify unhealthy spots in plant leaves in order to overcome the difficulties of manual inspection and the constraints of data-driven techniques. The approach is appropriate for low-resource setups and quick deployment because it doesn't require any pre-training or tagged datasets.

The system begins by applying **Gaussian blur**, a low-pass filtering technique, to reduce high-frequency noise and smoothen the image. It then converts the image into the **HSV color space**, which is more effective than RGB for isolating color-based features such as leaf areas and disease patterns. The leaf is segmented from the background based on HSV thresholds.

Next, the algorithm determines the **dominant healthy color** of the leaf—usually green or yellow—and uses this to isolate healthy regions. The remaining areas are considered potential disease zones. **Contour detection** is applied to extract these diseased regions, and their total area is computed and compared against the whole leaf area to calculate the **percentage of infection**.

The entire procedure is lightweight, automated, and transparently and simply designed. It provides a useful tool for identifying plant leaf diseases in photos, especially in situations without access to sophisticated hardware or datasets.

CHAPTER 02

RELATED WORK

2.1. RELATED WORK

A wide range of techniques have been explored over the years to automate plant disease detection using digital image processing. Most of these approaches fall into two prominent and broad categories: **traditional computer vision techniques** and **machine learning or deep learning-based models**.

Traditional methods commonly use color space transformations (such as HSV and LAB), filtering, morphological operations, and thresholding to segment diseased areas. For example, researchers have employed **HSV segmentation** to distinguish between healthy green parts and diseased brown or yellow patches. In some studies, **contour detection** has been used to measure the spread of the infection, while **morphological filters** help refine segmentation results.

However, because of their high accuracy, deep learning techniques—particularly Convolutional Neural Networks, or CNNs—have become popular in recent years. However, these approaches are less appropriate for offline or small-scale deployments because they necessitate substantial computational resources and annotated datasets.

The approach in this project builds upon classical techniques such as **Gaussian filtering**, **HSV color segmentation**, and **contour-based analysis**. These methods offer the advantage of being lightweight, interpretable, and easy to implement without requiring data annotation or training.

Previous work such as [1] used thresholding in the LAB color space to isolate diseased regions, while [2] compared multiple color spaces for effectiveness in disease detection. This project adopts a similar motivation but implements a cleaner pipeline with Gaussian noise reduction, healthy color masking, and precise contour-based disease area calculation.

CHAPTER 03

METHODOLOGY

3.1. IMAGE PROCESSING PIPELINE

The suggested system follows a structured pipeline of traditional image processing techniques implemented using OpenCV and Python. In order to determine the percentage of diseased areas in a leaf image, this pipeline processes the image. The stages are as follows:

1. **Image Acquisition and Resizing**
   The input leaf image is loaded and resized to a consistent resolution (e.g., 512×512) to ensure uniformity in analysis.

2. **Noise Reduction using Gaussian Blur**
   Gaussian blur is applied to the image to reduce high-frequency noise and improve the accuracy of color segmentation and contour detection.

3. **Color Space Conversion (BGR to HSV)**
   The image is converted from the BGR color space to HSV (Hue, Saturation, Value), as HSV allows easier separation of color information (hue) from intensity.

4. **Background Removal**
   The background, and a binary mask is created using HSV thresholds to isolate the leaf from the background. Morphological operations are then used to clean the mask.

5. **Dominant Hue Detection (Mode Hue)**
   To dynamically detect healthy regions, the hue values of pixels within the leaf mask are analyzed. The **most frequent hue (mode)** is calculated and used to define a narrow hue range (±10) for healthy color masking. This approach ensures adaptability to different types of leaves.

6. **Healthy Region Masking**
   Based on the dominant hue, a healthy region mask is created using HSV thresholding. This mask highlights the healthy parts of the leaf and is subtracted from the overall leaf mask.

7. **Disease Region Extraction**
   By subtracting the healthy region mask from the full leaf mask, the system isolates potentially diseased areas. Morphological operations are used to clean the result.

8. **Contour Detection and Area Measurement**
   Contours are detected on the disease mask. The area of each diseased contour is calculated, and the sum is used to compute the **percentage of infection** relative to the total leaf area.
   a. $Disease\ \% = (Total\ Diseased\ Area\ /\ Total\ Leaf\ Area) \times 100$

9. **Visualization**
   Several outputs are produced, such as a printed disease percentage, distinct visualizations of the extracted leaf and diseased areas, and the overlaying of detected contours on the original image.

3.2. USED FUNCTION AND THEIR APPLICATIONS

This section explains the key functions and operations used in your project and their role in the processing pipeline.

Function / Operation	Library	Purpose / Description
`cv2.imread()`	OpenCV	Reads the input image from file.
`cv2.resize()`	OpenCV	Resizes the image to a standard size (512×512) for consistent processing.
`cv2.cvtColor()`	OpenCV	Converts images between color spaces (e.g., BGR to HSV).
`cv2.GaussianBlur()`	OpenCV	Applies a Gaussian low-pass filter to reduce high-frequency noise.
`cv2.inRange()`	OpenCV	Creates binary masks by thresholding HSV or grayscale values within a specified range.
`cv2.bitwise_not()`	OpenCV	Inverts a binary mask (used to get leaf from background mask).
`cv2.bitwise_and()`	OpenCV	Performs bitwise AND between image and mask to isolate regions (e.g., healthy or diseased).
`cv2.morphologyEx()`	OpenCV	Applies morphological operations (e.g., OPEN, CLOSE) to clean binary masks.
`np.bincount()` + `argmax()`	NumPy	Finds the most frequent hue value (mode) from hue pixels inside the leaf region.
`cv2.subtract()`	OpenCV	Subtracts the healthy region mask from the leaf mask to isolate diseased areas.
`cv2.findContours()`	OpenCV	Detects contours in the binary disease mask to outline infected regions.
`cv2.drawContours()`	OpenCV	Draws the detected disease contours on the original image for visualization.
`cv2.countNonZero()`	OpenCV	Computes the area (in pixels) of the leaf and disease regions.
`matplotlib.pyplot.imshow()`	Matplotlib	Displays RGB or grayscale images during the visualization phase.
`matplotlib.pyplot.subplot()`	Matplotlib	Creates a grid of subplots to compare multiple image stages.

3.3. USED TOOLS, TECHNOLOGIES AND LIBRARIES

This section unfolds the tools, programming libraries, and development environments used to build and test your leaf disease detection system

Tool / Library	Purpose / Use
Python 3.x	Main programming language used for developing the image processing pipeline.
OpenCV (cv2)	Image processing library used for filtering, masking, segmentation, and contour detection.
NumPy	Used for array and pixel-level operations, including mode hue calculation.
Matplotlib	Visualization library for displaying and comparing image stages in the pipeline.
PyCharm IDE	Integrated development environment used to write, run, and debug Python code.
Kaggle Plant Leaf Dataset	Real-world dataset of plant leaf images used to evaluate and test the system.

3.4. PROCEDURE

1. **Image Acquisition**
   Leaf images are selected from a publicly available Kaggle dataset and loaded into the system.

2. **Preprocessing with Gaussian Blur**
   A Gaussian blur is applied to the image to remove high-frequency noise and smooth out variations for more accurate segmentation.

3. **Color Space Conversion to HSV**
   The image is converted from the BGR to HSV color space to allow easier color-based thresholding.

4. **Background Removal**
   A mask is created to separate the background from the leaf using HSV range filtering. Morphological operations refine the binary mask.

5. **Dominant Hue Detection**
   The most frequently occurring hue value (mode) within the leaf region is calculated. A dynamic HSV range is created around this value to identify healthy regions.

6. **Healthy Region Masking**
   A mask is created for healthy leaf areas based on the dominant hue range. This mask is refined using morphological filters.

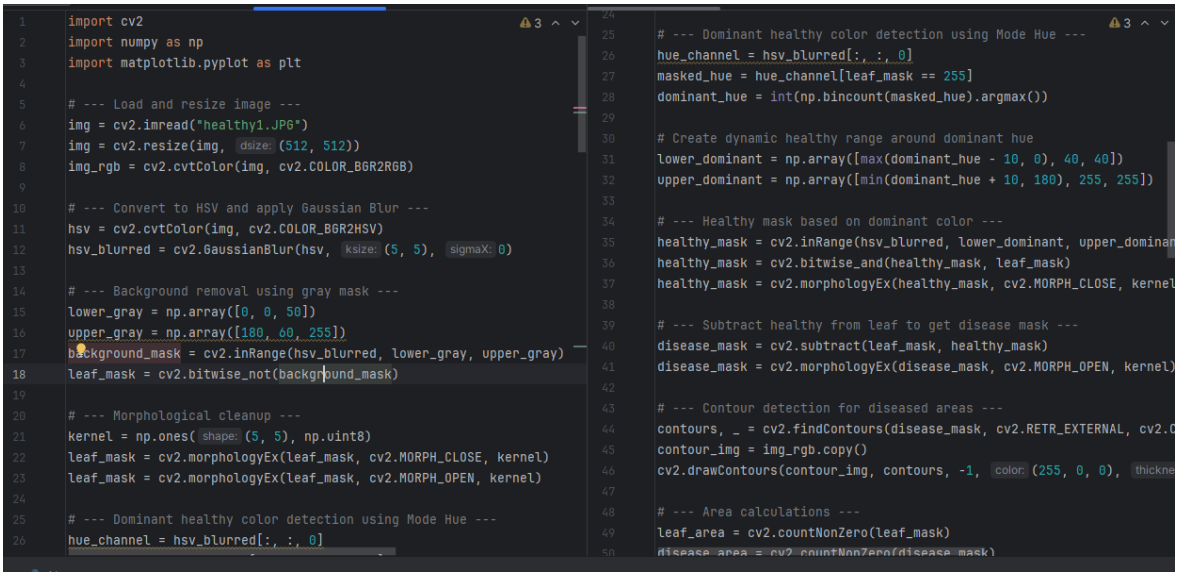
7. **Disease Region Isolation**
   By subtracting the healthy mask from the full leaf mask, potentially diseased regions are isolated.

8. **Contour Extraction**
   Contours are detected around the diseased regions. These contours visually indicate affected areas on the original image.

9. **Area and Percentage Calculation**
   The disease percentage is calculated by dividing the total leaf area by the number of non-zero pixels in the disease mask.

10. **Visualization and Output**
    The system output a set of images at each major step, including overlays of disease regions and the calculated infection percentage.

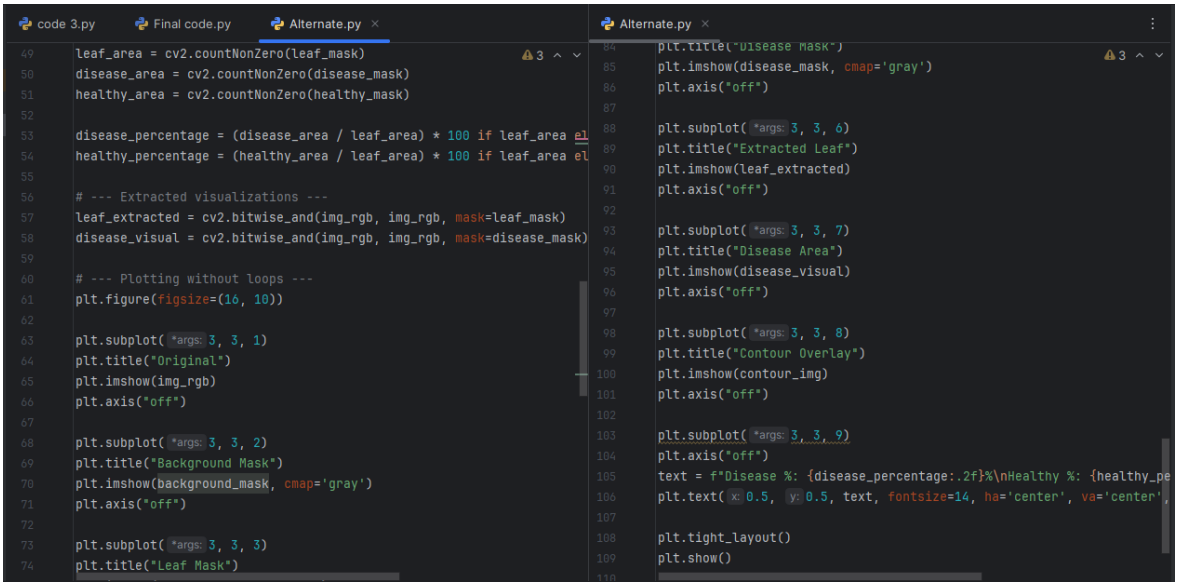
3.5. PROCEDURE EXPLAINED



```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

# --- Load and resize image ---
img = cv2.imread("healthy1.JPG")
img = cv2.resize(img, dsize: (512, 512))
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# --- Convert to HSV and apply Gaussian Blur ---
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
hsv_blurred = cv2.GaussianBlur(hsv, ksize: (5, 5), sigmaX: 0)

# --- Background removal using gray mask ---
lower_gray = np.array([0, 0, 50])
upper_gray = np.array([180, 60, 255])
background_mask = cv2.inRange(hsv_blurred, lower_gray, upper_gray)
leaf_mask = cv2.bitwise_not(background_mask)

# --- Morphological cleanup ---
kernel = np.ones( shape: (5, 5), np.uint8)
leaf_mask = cv2.morphologyEx(leaf_mask, cv2.MORPH_CLOSE, kernel)
leaf_mask = cv2.morphologyEx(leaf_mask, cv2.MORPH_OPEN, kernel)

# --- Dominant healthy color detection using Mode Hue ---
hue_channel = hsv_blurred[:, :, 0]
```

```python
# --- Dominant healthy color detection using Mode Hue ---
hue_channel = hsv_blurred[:, :, 0]
masked_hue = hue_channel[leaf_mask == 255]
dominant_hue = int(np.bincount(masked_hue).argmax())

# Create dynamic healthy range around dominant hue
lower_dominant = np.array([max(dominant_hue - 10, 0), 40, 40])
upper_dominant = np.array([min(dominant_hue + 10, 180), 255, 255])

# --- Healthy mask based on dominant color ---
healthy_mask = cv2.inRange(hsv_blurred, lower_dominant, upper_dominan
healthy_mask = cv2.bitwise_and(healthy_mask, leaf_mask)
healthy_mask = cv2.morphologyEx(healthy_mask, cv2.MORPH_CLOSE, kernel

# --- Subtract healthy from leaf to get disease mask ---
disease_mask = cv2.subtract(leaf_mask, healthy_mask)
disease_mask = cv2.morphologyEx(disease_mask, cv2.MORPH_OPEN, kernel)

# --- Contour detection for diseased areas ---
contours, _ = cv2.findContours(disease_mask, cv2.RETR_EXTERNAL, cv2.C
contour_img = img_rgb.copy()
cv2.drawContours(contour_img, contours, -1, color: (255, 0, 0), thickne

# --- Area calculations ---
leaf_area = cv2.countNonZero(leaf_mask)
disease_area = cv2.countNonZero(disease_mask)
```

Figure 1-Code Snippet 1



```python
leaf_area = cv2.countNonZero(leaf_mask)
disease_area = cv2.countNonZero(disease_mask)
healthy_area = cv2.countNonZero(healthy_mask)

disease_percentage = (disease_area / leaf_area) * 100 if leaf_area el
healthy_percentage = (healthy_area / leaf_area) * 100 if leaf_area el

# --- Extracted visualizations ---
leaf_extracted = cv2.bitwise_and(img_rgb, img_rgb, mask=leaf_mask)
disease_visual = cv2.bitwise_and(img_rgb, img_rgb, mask=disease_mask)

# --- Plotting without loops ---
plt.figure(figsize=(16, 10))

plt.subplot( *args: 3, 3, 1)
plt.title("Original")
plt.imshow(img_rgb)
plt.axis("off")

plt.subplot( *args: 3, 3, 2)
plt.title("Background Mask")
plt.imshow(background_mask, cmap='gray')
plt.axis("off")

plt.subplot( *args: 3, 3, 3)
plt.title("Leaf Mask")
```

```python
plt.title("Disease Mask")
plt.imshow(disease_mask, cmap='gray')
plt.axis("off")

plt.subplot( *args: 3, 3, 6)
plt.title("Extracted Leaf")
plt.imshow(leaf_extracted)
plt.axis("off")

plt.subplot( *args: 3, 3, 7)
plt.title("Disease Area")
plt.imshow(disease_visual)
plt.axis("off")

plt.subplot( *args: 3, 3, 8)
plt.title("Contour Overlay")
plt.imshow(contour_img)
plt.axis("off")

plt.subplot( *args: 3, 3, 9)
plt.axis("off")
text = f"Disease %: {disease_percentage:.2f}%\nHealthy %: {healthy_pe
plt.text( x: 0.5, y: 0.5, text, fontsize=14, ha='center', va='center',

plt.tight_layout()
plt.show()
```

Figure 2-Code Snippet 2

CHAPTER 04

RESULTS AND DISCUSSION

4.1. MAJOR RESULTING STEPS

The proposed system successfully generated the following intermediate and final results for each tested image:

1. **Original RGB Image**
   Displays the input leaf image after resizing and color conversion to RGB for visualization.
2. **Background Mask**
   Shows the binary mask created to remove the gray background using HSV color range filtering.
3. **Cleaned Leaf Mask**
   Outputs a refined binary mask representing the leaf area only, after morphological operations.
4. **Healthy Region Mask (Dominant Hue)**
   Highlights healthy regions on the leaf using a dynamic threshold centered around the most frequent hue value. This allows adaptation to green or yellow shades.
5. **Disease Region Mask**
   Displays areas of the leaf identified as potentially diseased, obtained by subtracting the healthy mask from the full leaf mask.
6. **Extracted Leaf Image**
   Shows the isolated leaf after removing the background, retaining only the useful region for disease analysis.
7. **Disease Area Visualization**
   Outputs the diseased regions as a separate image, showing only the infected zones extracted from the RGB image.
8. **Contour Overlay on Original Image**
   Final visualization step where all disease contours are drawn on the original leaf image, making infected areas clearly visible.
9. **Disease Percentage Display**
   Shows the calculated disease percentage for each image, providing a quantitative measure of infection.

The results held steady across a range of test samples, demonstrating the pipeline's adaptability to handling a variety of leaf shapes and colors using mode-based hue segmentation.

4.2. SOME TESTED IMAGE RESULTS

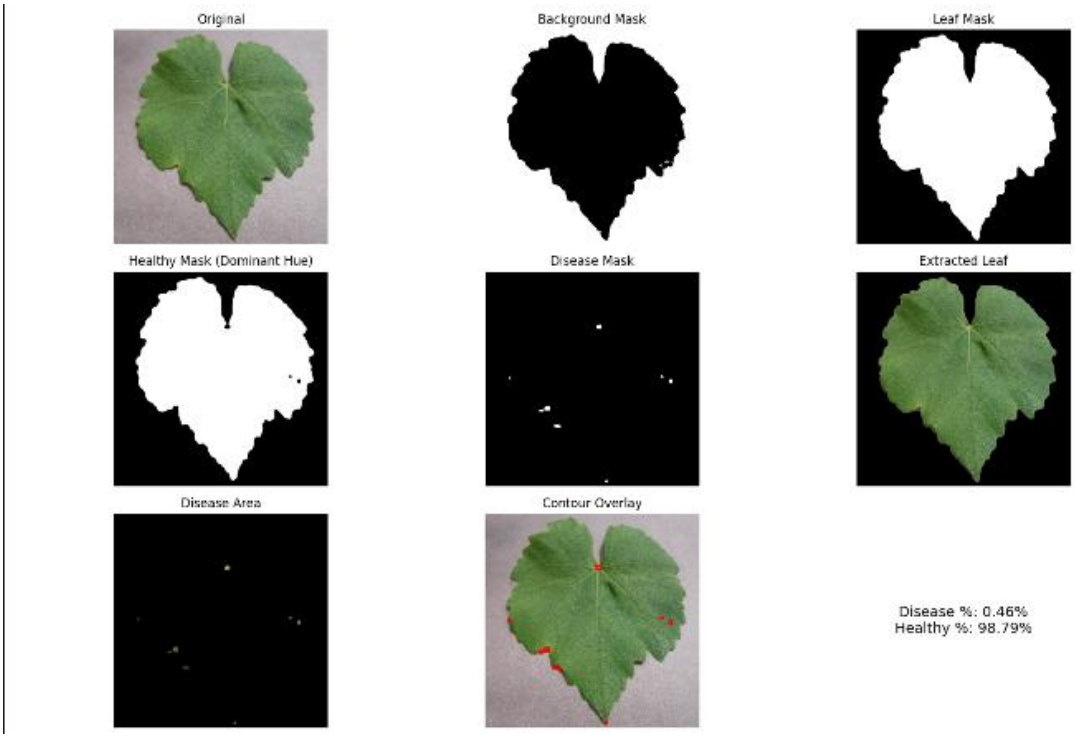


Figure 3-Healthy Leaf 1

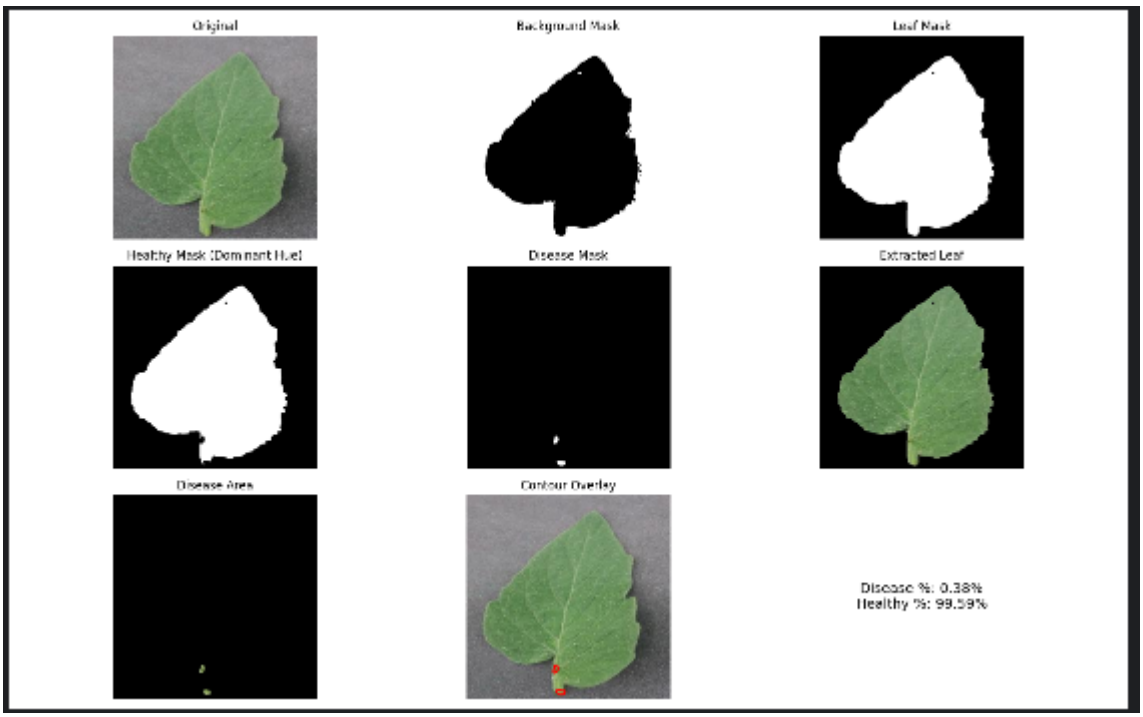


Figure 4-Healthy Leaf 2

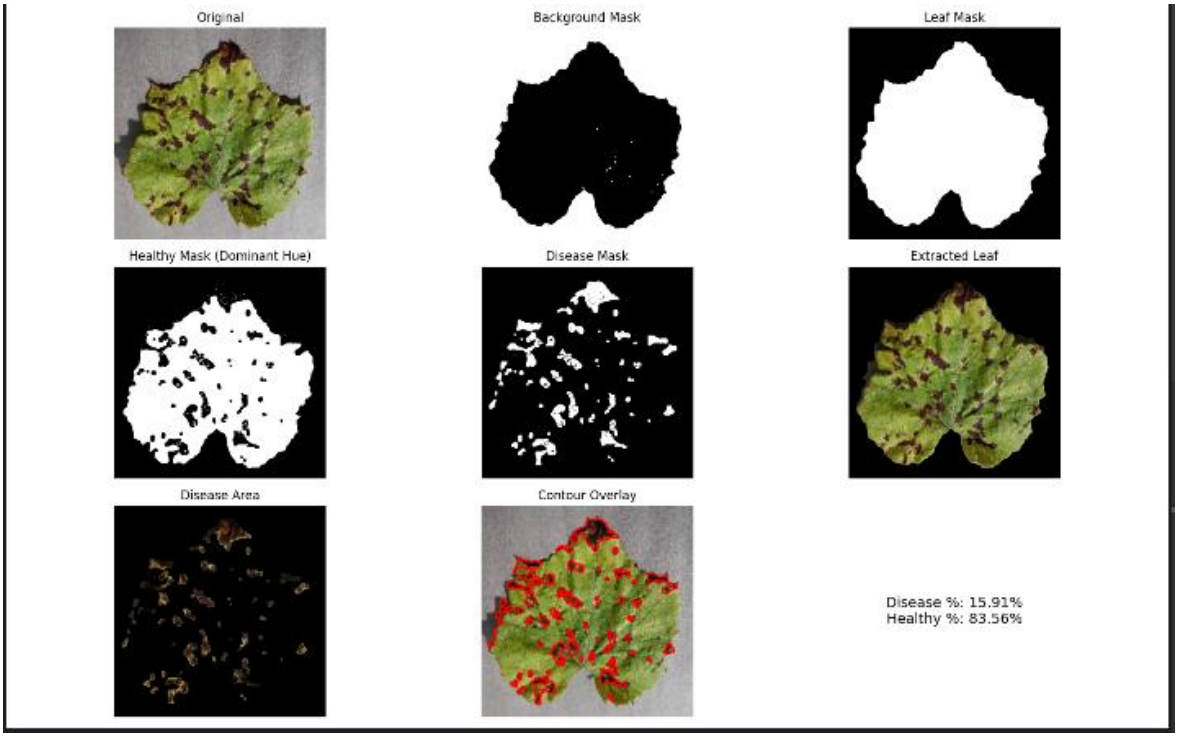


Figure 5- Disease Leaf 1

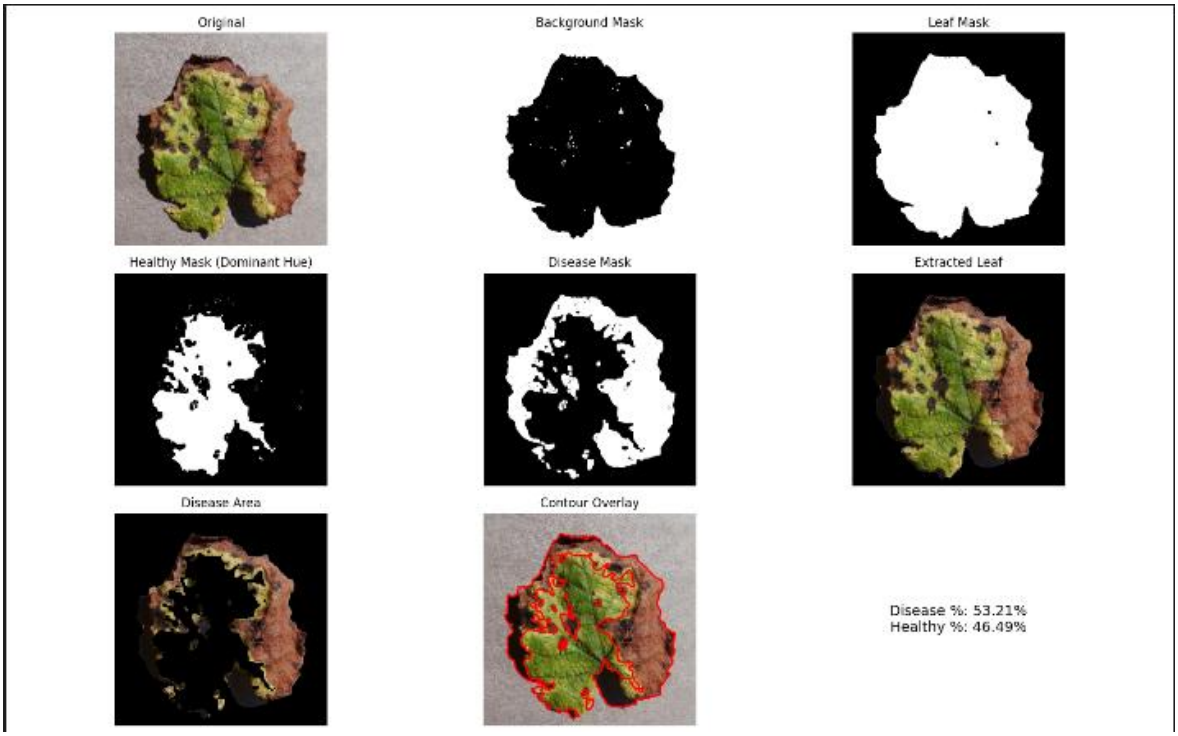


Figure 6-Disease Leaf 2

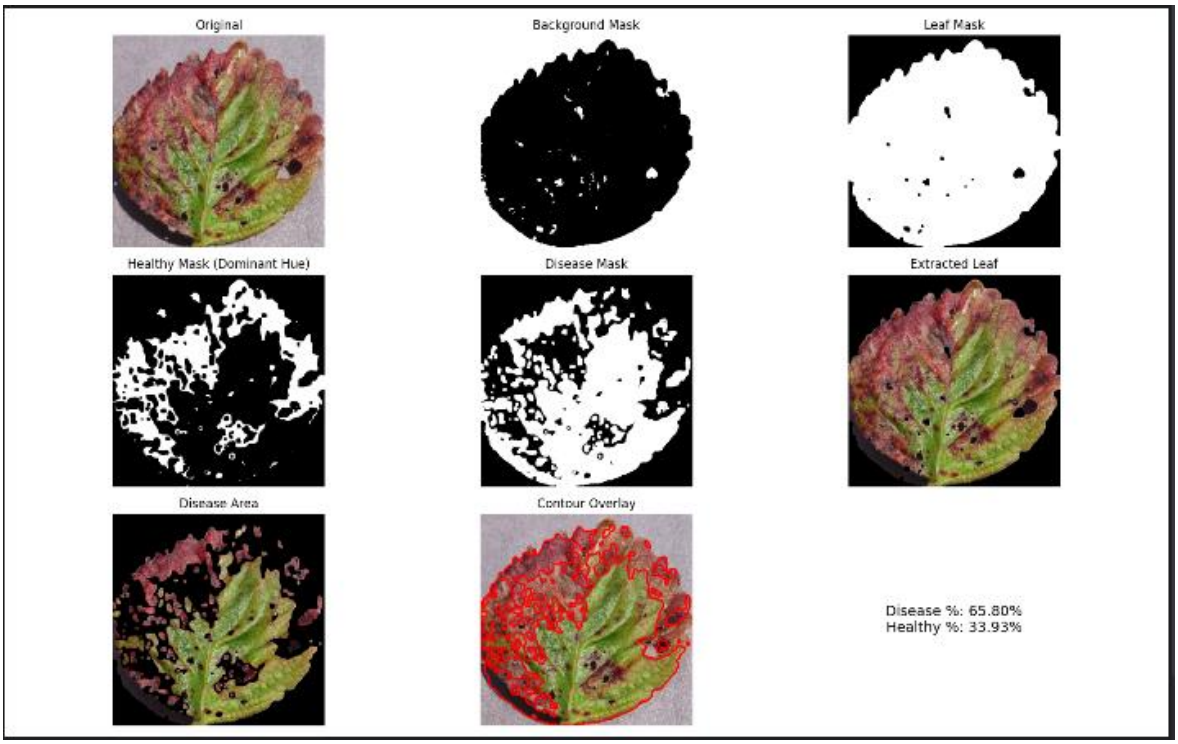


Figure 7-Disease Leaf 3

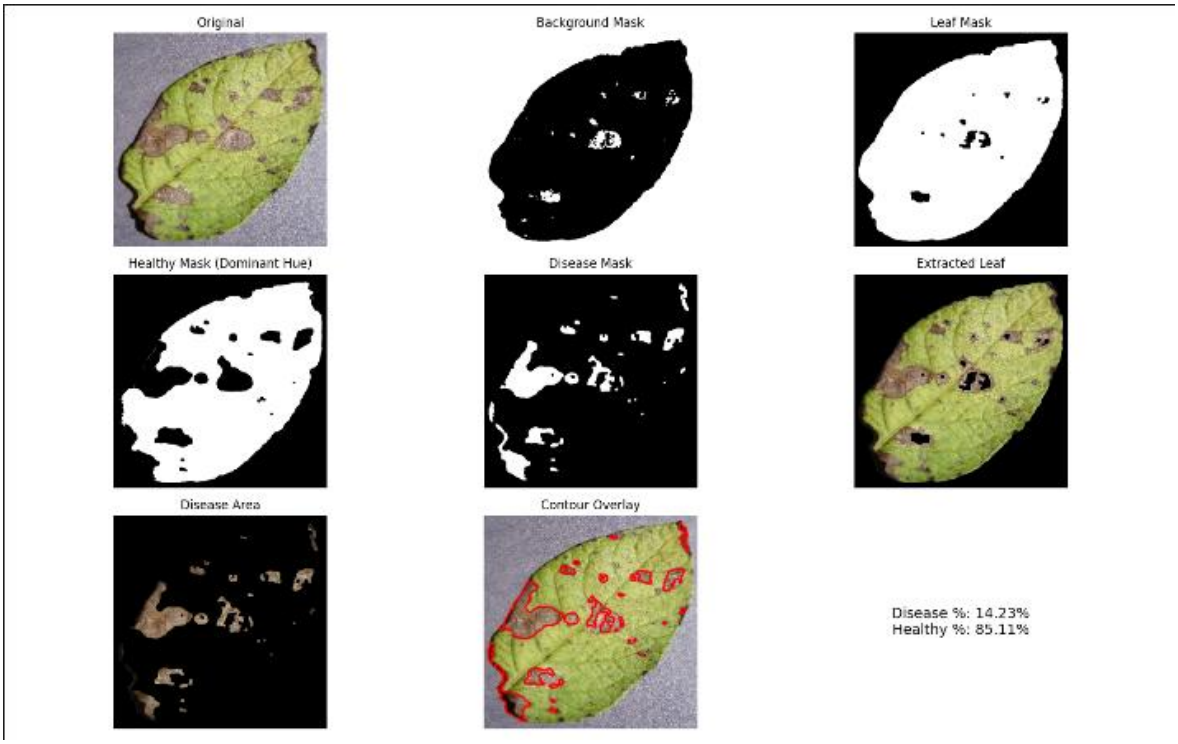


Figure 8-Disease Leaf 4

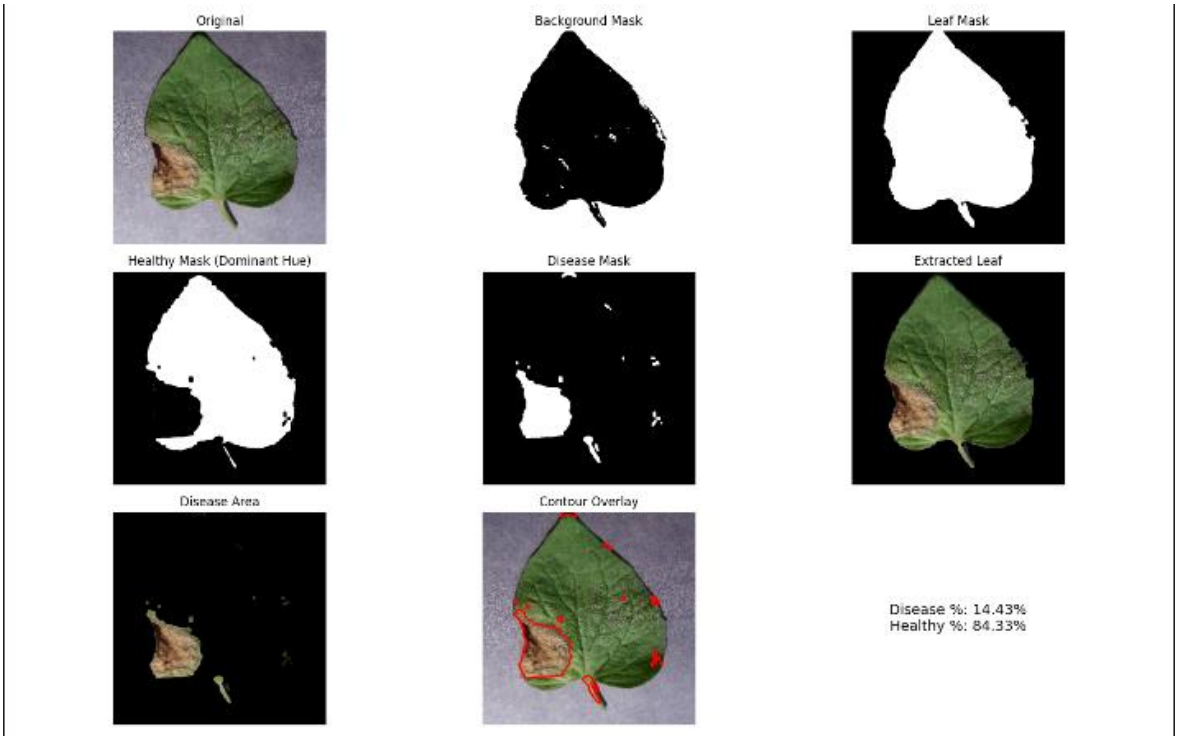


Figure 9-Disease Leaf 5

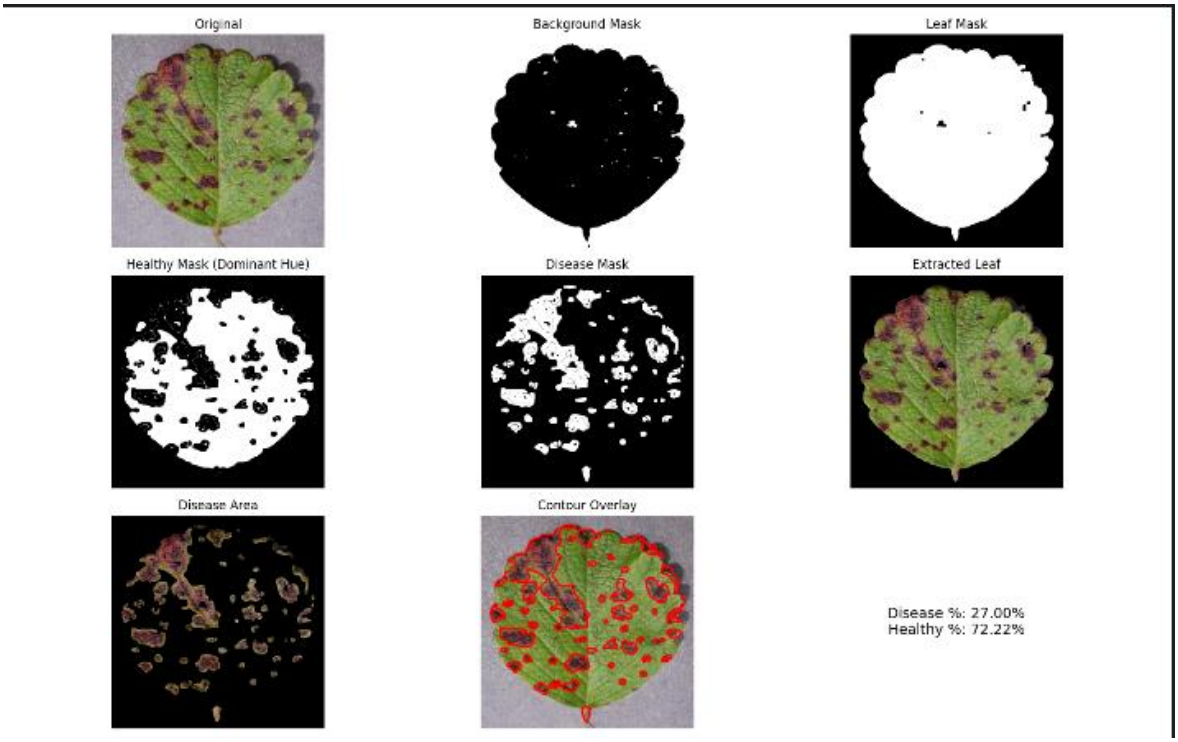


Figure 10-Disease Leaf 6

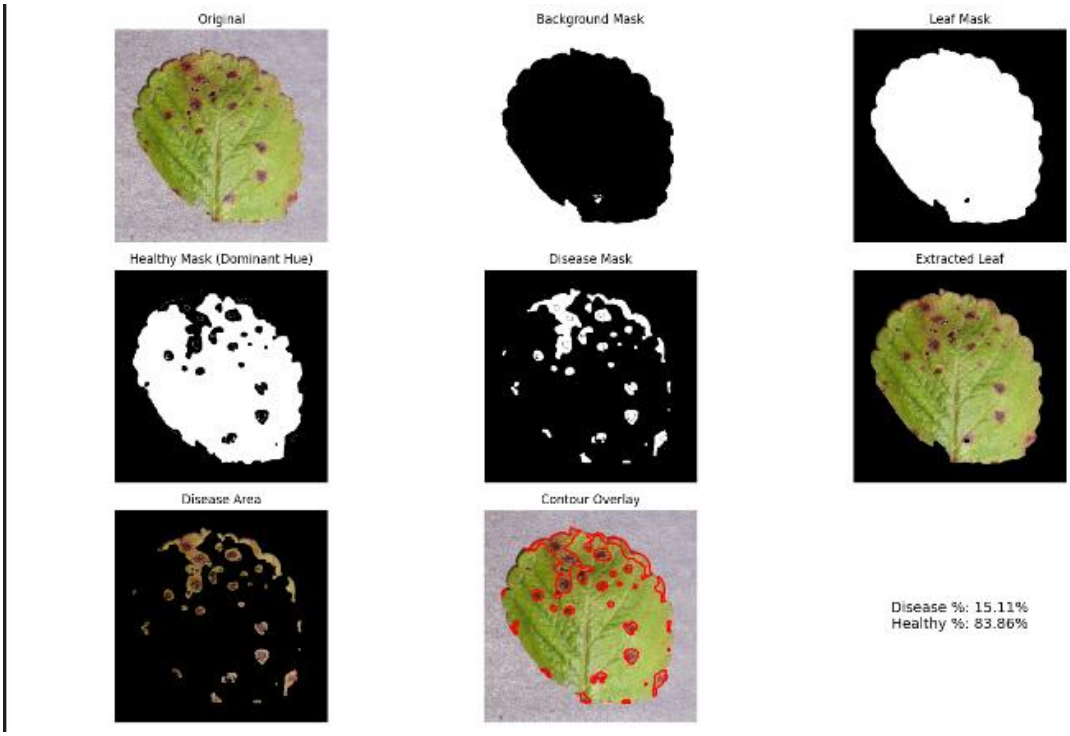


Figure 11-Disease Leaf 7

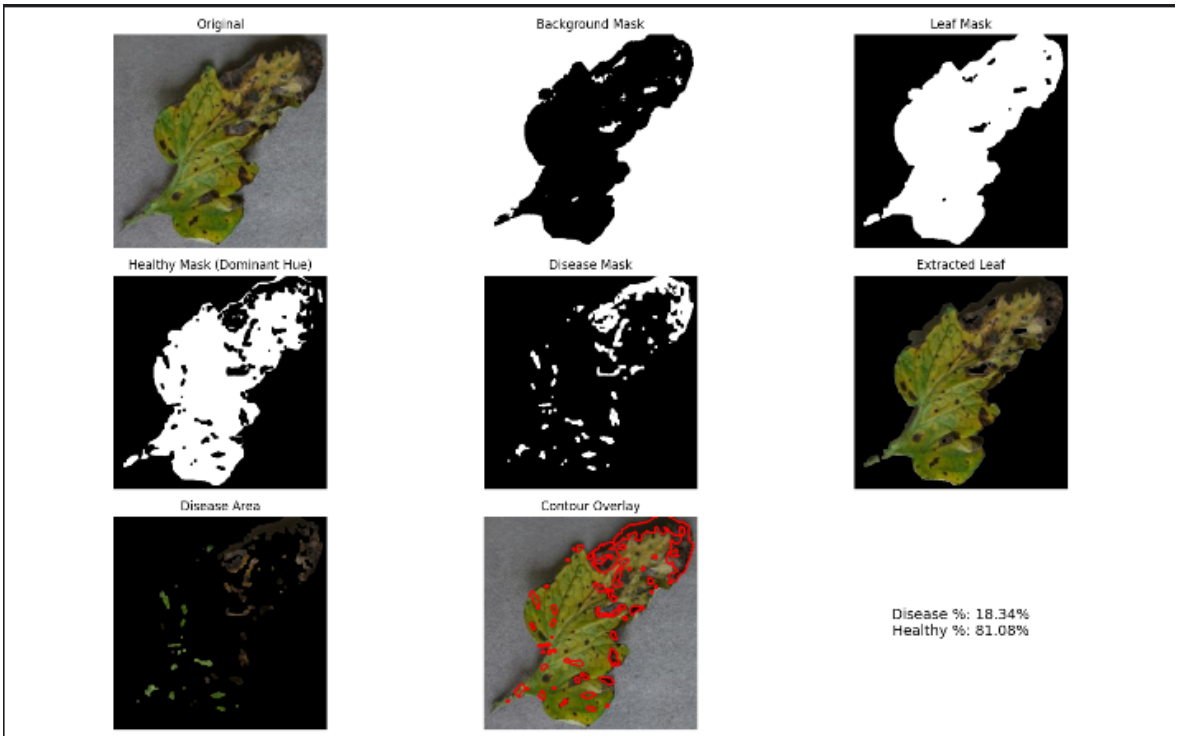


Figure 12-Disease Leaf 8

4.3. DISCUSSION

The proposed system successfully applies classical/traditional image processing methods to identify and measure diseased areas in plant leaf images. The approach is lightweight, interpretable, and performs reliably across multiple test samples without requiring any machine learning or training data.

The use of **Gaussian blur** helped reduce high-frequency noise, which improved mask accuracy and stability. **HSV color space segmentation** allowed clear separation of hue-based features, making it more suitable than RGB for plant image analysis. Crucially, **mode hue detection** enabled better and accuracy across different leaf types and diseases.

The **contour-based area calculation** accurately identified infected regions and enabled straightforward visualization and percentage computation. The disease percentage values matched visual intuition, confirming the method's consistency.

However, the system has a few limitations:

- It may misclassify **natural color variations** (e.g., yellowing due to aging) as disease.
- It may **or** and may not work optimally on complex or noisy backgrounds.
- **Vein and boundary interference** can occasionally affect the precision of the disease region mask.

Still, the method remains practical and suitable for offline use in low-resource environments where deep learning is not viable.

CHAPTER 05

CONCLUSIONS

5.1. ADVANTAGES OF IMPLEMENTED SYSTEM

- **Lightweight and Fast**: The entire system runs on traditional image processing techniques without requiring training or GPU acceleration.
- **No Dataset Required**: Unlike deep learning methods, it doesn't need labeled datasets or model training.
- **Adaptable**: The use of **mode hue detection** allows the system to adapt to different types of leaves and healthy color tones.
- **Interpretable Results**: The contour overlays and percentage calculation provide clear visual and numerical outputs.
- **Useful in Low-Resource Settings**: Suitable for farmers or institutions with limited computing resources.

5.2. ISSUES

- **False Positives on Natural Yellowing**: The system might misclassify natural aging or leaf discoloration as disease.
- **Complex Background**: Performance may degrade on complex or colored backgrounds.
- **Vein and Texture Artifacts**: Leaf veins or strong textures may occasionally be marked as disease unless filtered out.
- **Limited to Color-Based Disease**: Infections that don’t result in visible color change might not be detectable.

5.3. CONCLUTION

This project effectively illustrates a practical image processing pipeline for detecting diseased areas in plant leaves using conventional computer vision techniques. By employing Gaussian filtering, HSV segmentation, and dominant hue-based healthy region masking, the system identifies and quantifies disease regions without requiring machine learning In settings with limited resources, its ease of use, effectiveness, and versatility make it a good choice for early disease diagnosis. Despite certain drawbacks, the approach serves as a solid basis for further improvements and integration into real-world (actual) agricultural equipment.

REFERENCES

- **Gonzalez, R.C. and Woods, R.E., 2018. *Digital Image Processing*. 4th ed. Pearson.**
- **OpenCV, 2024. *Open Source Computer Vision Library*. [online] Available at: https://opencv.org/ [Accessed 9 Jul. 2025].**
- **Kaggle, 2024. *Plant Leaf Disease Dataset*. [online] Available at: https://www.kaggle.com/datasets [Accessed 9 Jul. 2025].**