Leaf Disease Detection Using Traditional Image Processing Techniques

B.W.D.U. BANDARA

CSC 3141 – Image Processing Laboratory

S/20/314

INTRODUCTION

- Agriculture is a vital sector where plant health directly impacts productivity and food supply.
- Leaf diseases are common, and early detection is crucial to prevent widespread damage.
- Manual inspection of plant diseases is time-consuming and prone to human error.
- This project aims to build an automated system to detect diseased regions in plant leaves using traditional image processing techniques.

Problem Statement

Manual disease identification lacks consistency and scalability.

Machine learning models need large labeled datasets and computational power.

Many agricultural settings lack access to such resources.

Need for a **lightweight, fast, and reliable** system using only classic image processing.

Should adapt to **different leaf colors and disease types** without training.

Objective

- To detect leaf diseases using only traditional image processing techniques.
- To accurately isolate the leaf from the background.
- To identify healthy regions using the dominant hue (mode) of the leaf.
- To segment and highlight diseased areas using contour analysis.
- To compute and display the percentage of leaf area affected.
- To visualize each processing step clearly for interpretability.

Methodology Overview

- Load & Resize Image
- Gaussian Blur (Noise Removal)
- HSV Conversion
- Background Removal
- Leaf Mask (Morphological Cleaning)
- Dominant Hue Detection (Mode)
- Healthy Region Mask
- Disease Mask = Leaf - Healthy
- Contour Detection
- Area Calculation & Visualization

Tools & Technologies

Tool / Library	Purpose
Python 3.x	Programming language
OpenCV (cv2)	Image processing operations
NumPy	Numerical operations, dominant hue
Matplotlib	Result visualization
PyCharm	IDE used for coding and testing
Kaggle Dataset	Source of real-world leaf images

Image Processing Steps

- **Image Preprocessing Steps**
  - Image Acquisition
    Load and resize the input image (e.g., 512×512).
  - Gaussian Blur
    Reduce noise using low-pass filtering.
  - HSV Conversion
    Convert from BGR to HSV color space for easier color-based segmentation.
  - Background Removal
    Identify and remove background using HSV thresholding.
  - Leaf Mask Creation
    Morphological operations clean up the leaf region for further processing.

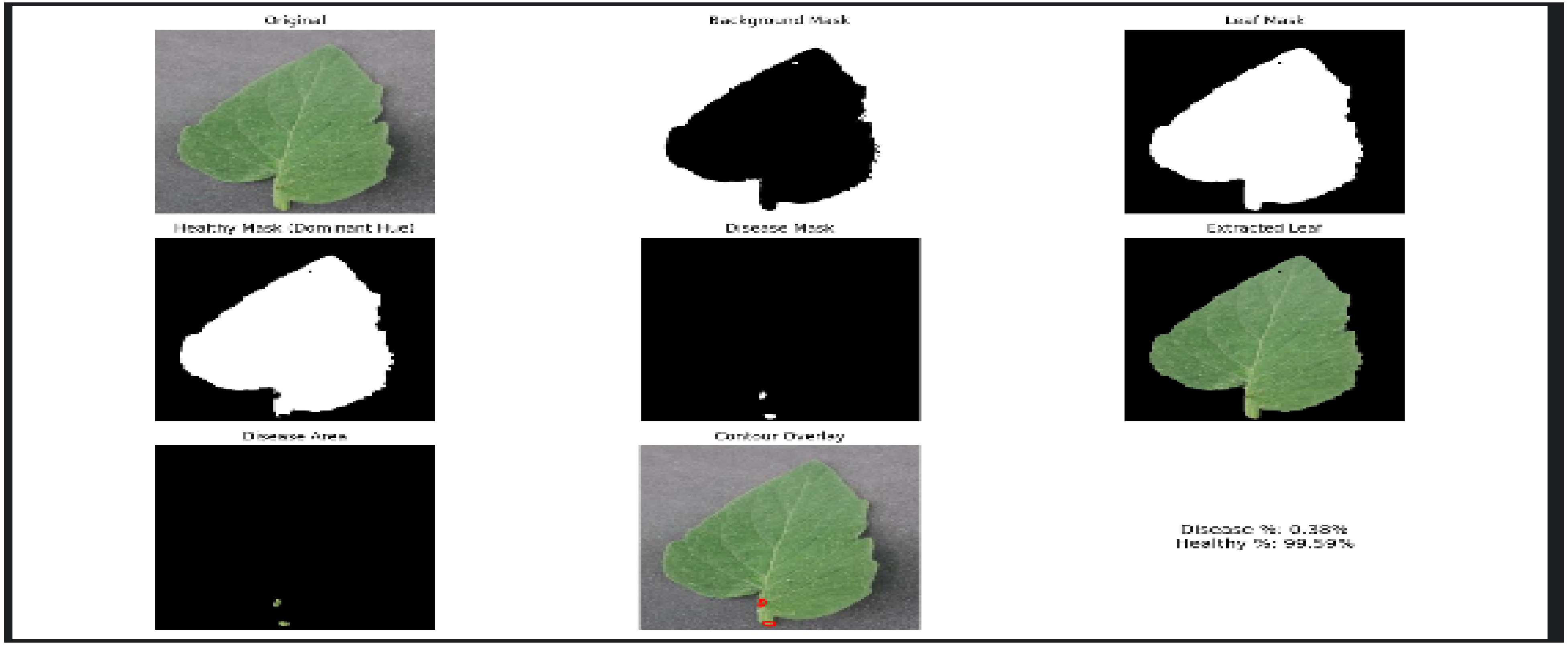
Image Processing Steps

- **Dominant Hue Detection**
  - After isolating the leaf, the hue values are analyzed to find the most common color. This "**mode hue**" represents the healthy part of the leaf.
  - A dynamic HSV threshold is created around this mode value (±10).
  - This makes the system adaptable to both **green** and **yellow** healthy leaves.
  - **Why Mode Hue?**
    → Fixed green/yellow ranges may fail for different crops.
    → Mode hue allows the system to generalize across leaf types.

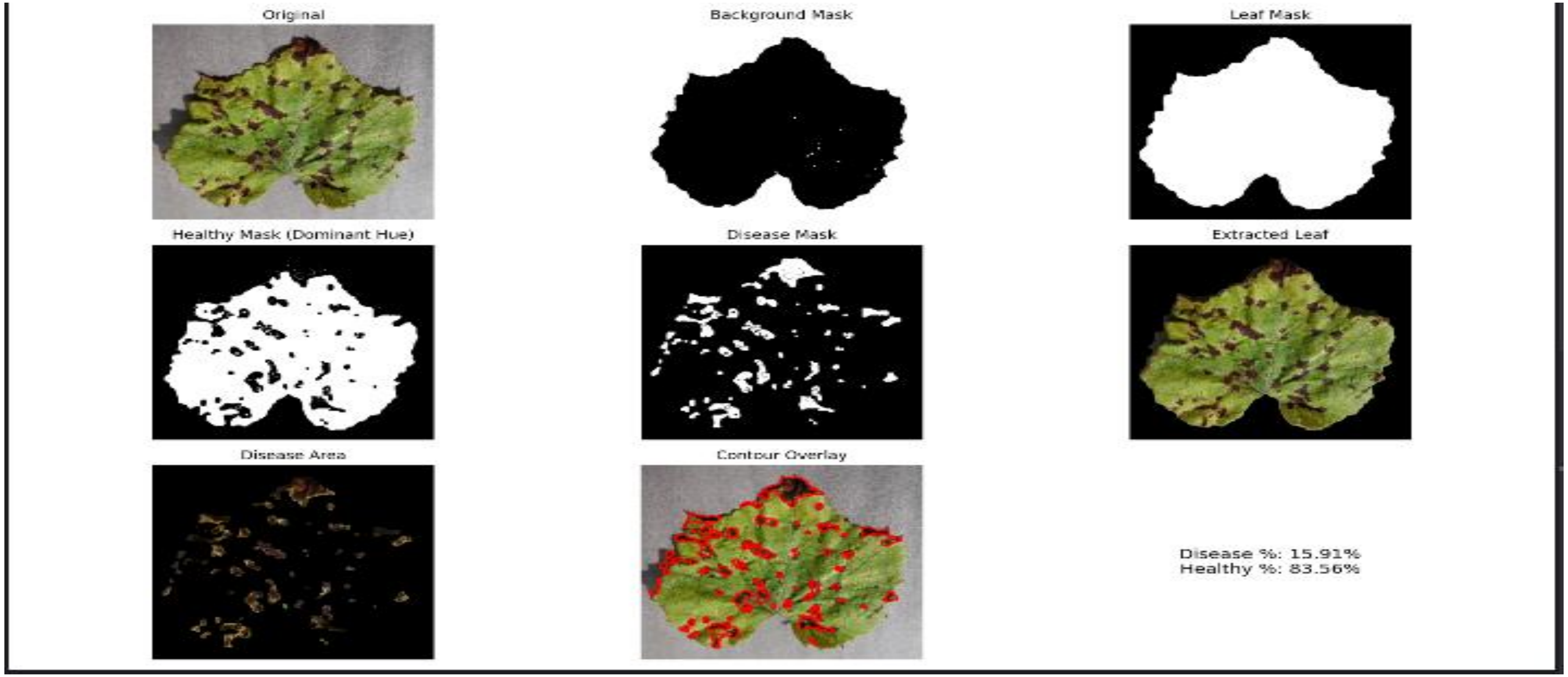
Image Processing Steps

- **Disease Masking & Contour Analysis**
  - After creating the healthy mask:
  - Subtract it from the leaf to get the disease mask
  - Use contour detection to outline infected regions
  - Calculate:
  - Disease % = (Area of Diseased Region / Total Leaf Area) × 100
  - Display this visually and numerically

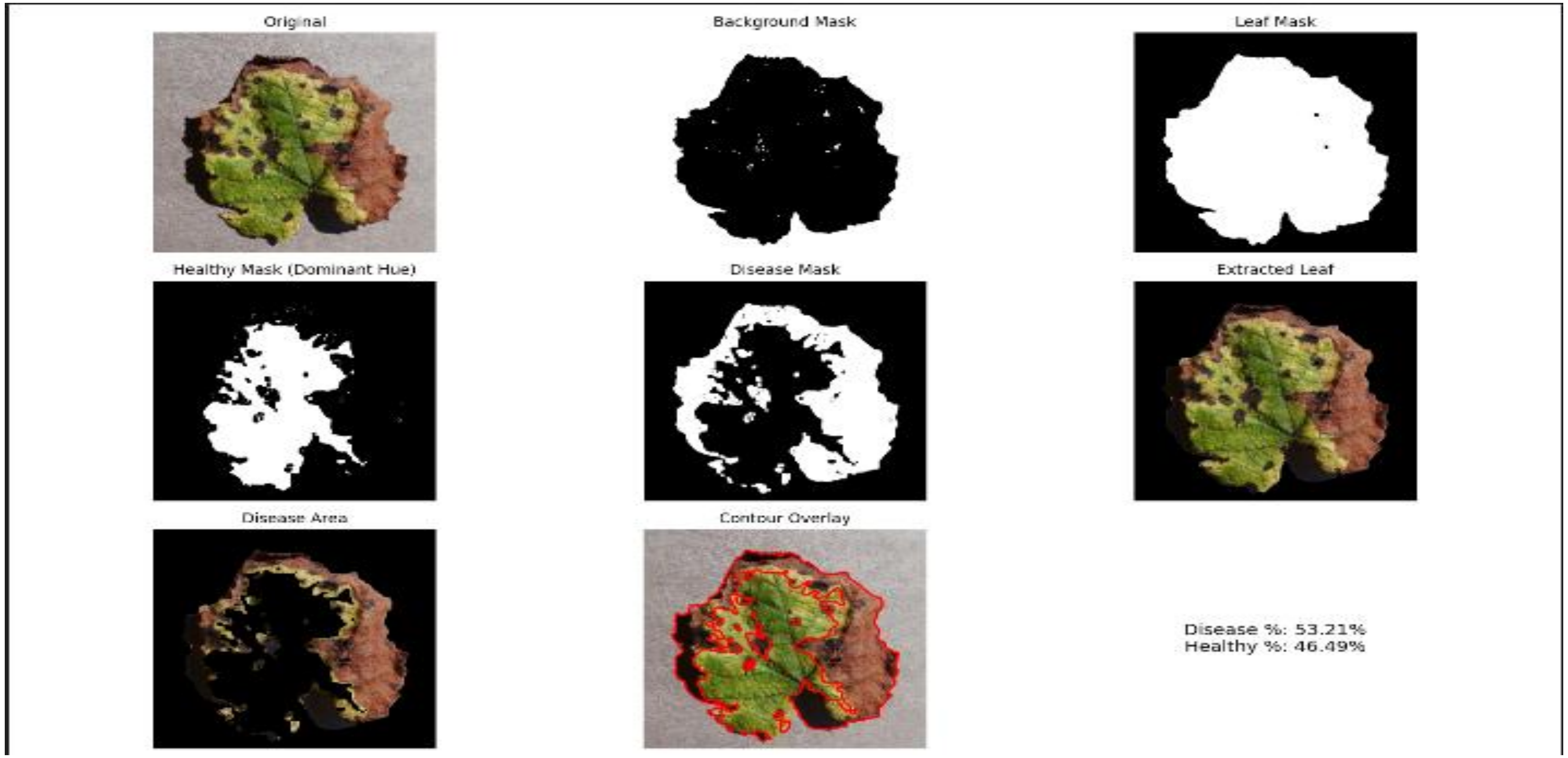
Sample Result



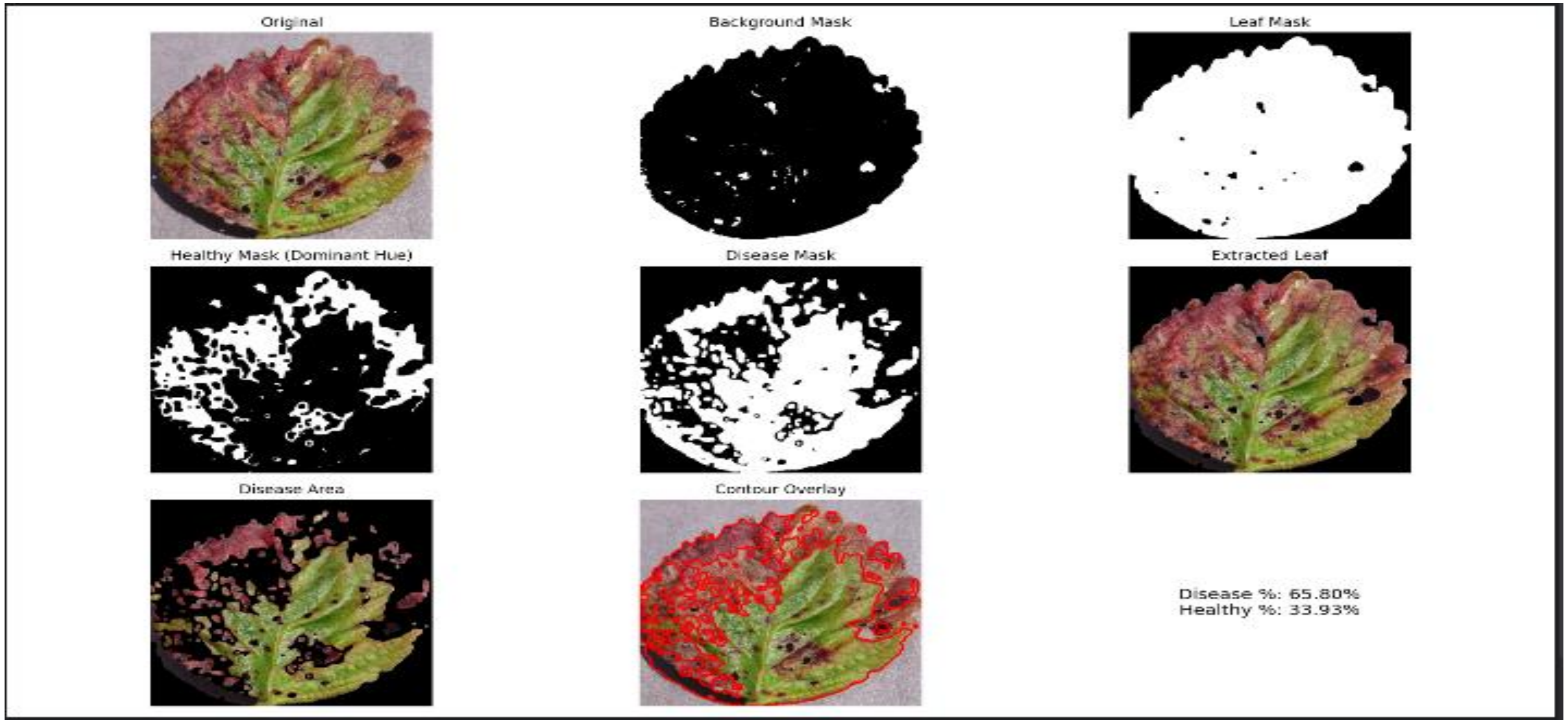
Sample Result



Sample Result



Sample Result



Result Summary Table

Image Name	Condition	Disease %	Healthy %
Leaf_01.jpg	Healthy	0.38%	99.56%
Leaf_02.jpg	Mild Infection	15.912%	83.56%
Leaf_03.jpg	Moderate Infection	53.21%	46.49%
Leaf_04.jpg	Severe Infection	65.80%	33.93%

Code Snippet

```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]
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)
```

Code Snippet

code 3.py   Final code.py   Alternate.py

```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(3, 3, 1)
plt.title("Original")
plt.imshow(img_rgb)
plt.axis("off")

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

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

Alternate.py

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

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

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

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

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

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

Advantages

- ✅ **Lightweight** – No ML model or training needed
- ✅ **Fast** – Runs in real time on basic hardware
- ✅ **Interpretable** – Outputs visual and numerical results
- ✅ **Adaptable** – Works with green/yellow leaves via dominant hue
- ✅ **OpenCV-based** – Easy to integrate into larger systems

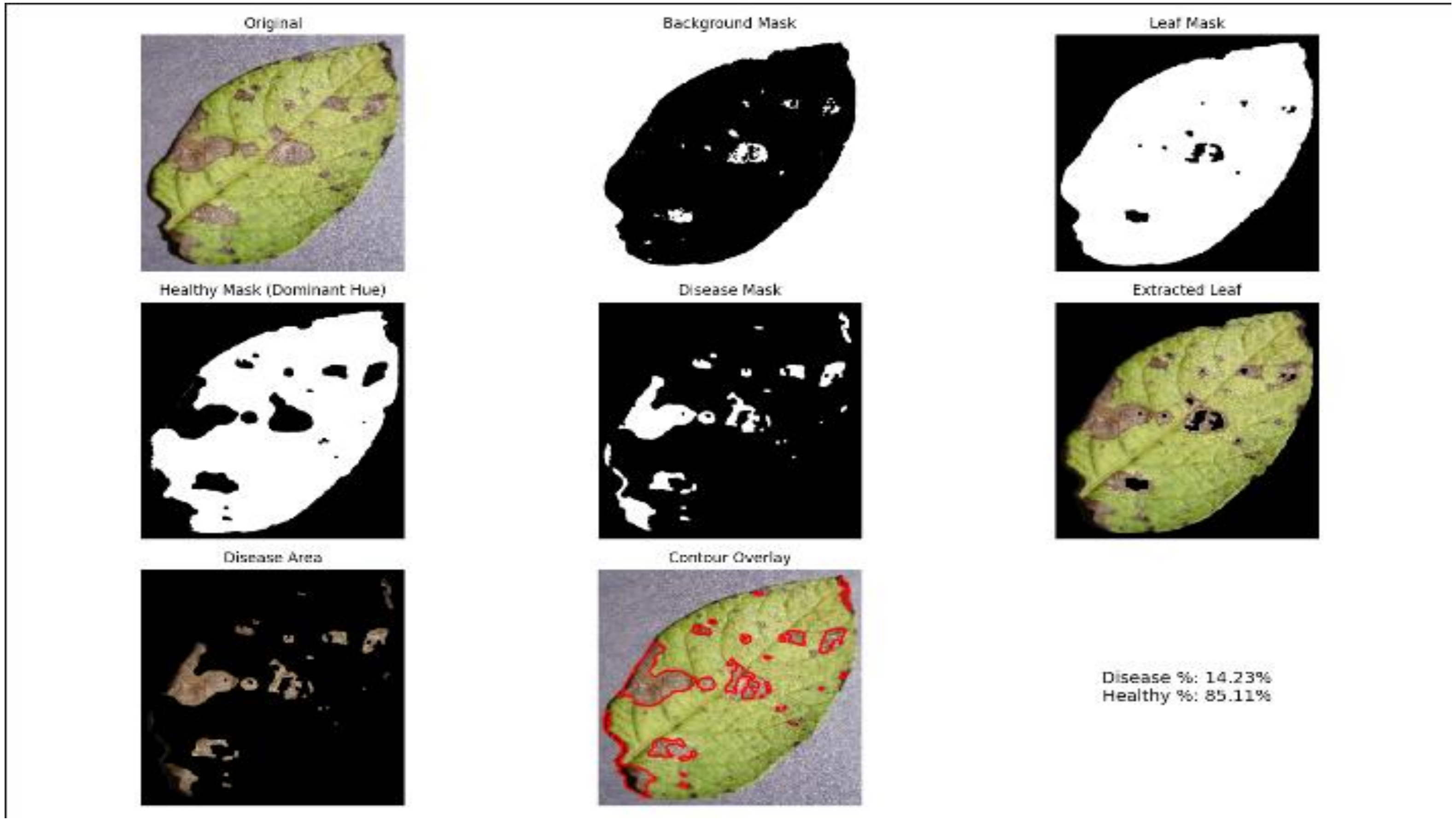
Limitations

- ⚠️ **False Positives**: May confuse aging/yellowing with disease
- ⚠️ **Color-Based Only**: Cannot detect non-visible infections
- ⚠️ **Vein/Edge Confusion**: May mislabel sharp leaf veins as infected

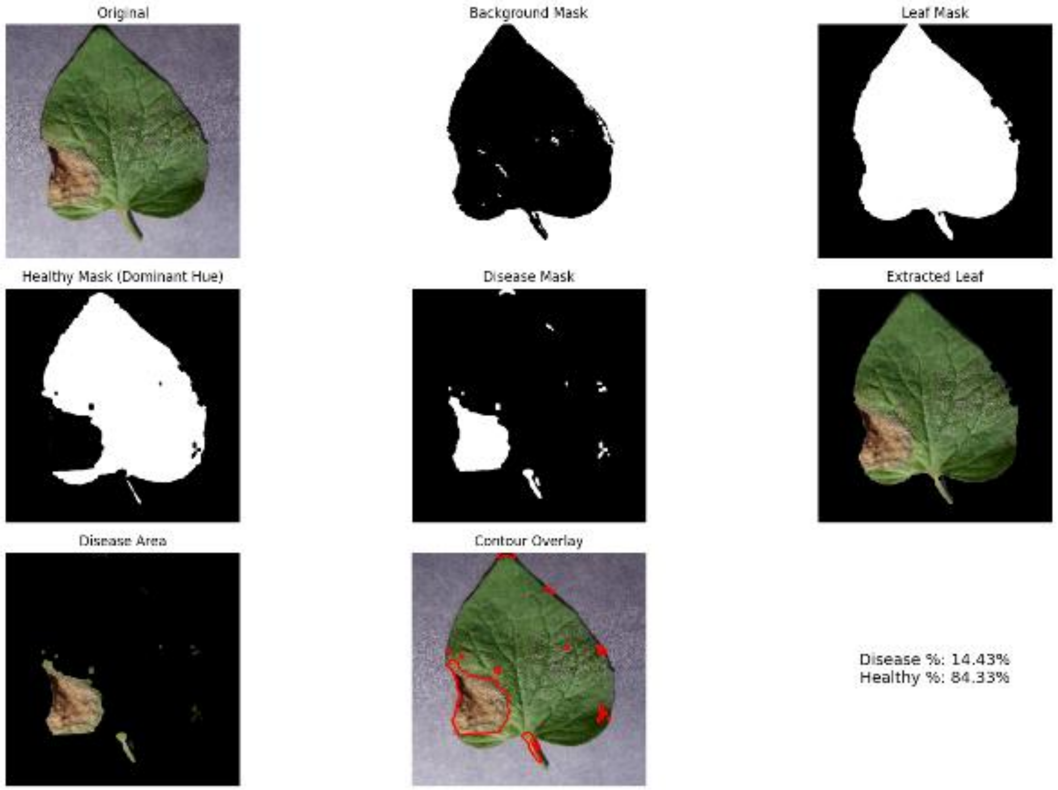
Conclusion

- Successfully built a disease detection system using only **traditional image processing**.
- Pipeline includes: Gaussian filtering, HSV segmentation, dynamic healthy color masking, and contour-based disease analysis.
- Produces clear **visual and numerical outputs** with **0–64%+ disease detection accuracy** across varied leaf samples.
- Requires **no machine learning**, making it suitable for **offline, low-resource environments**.

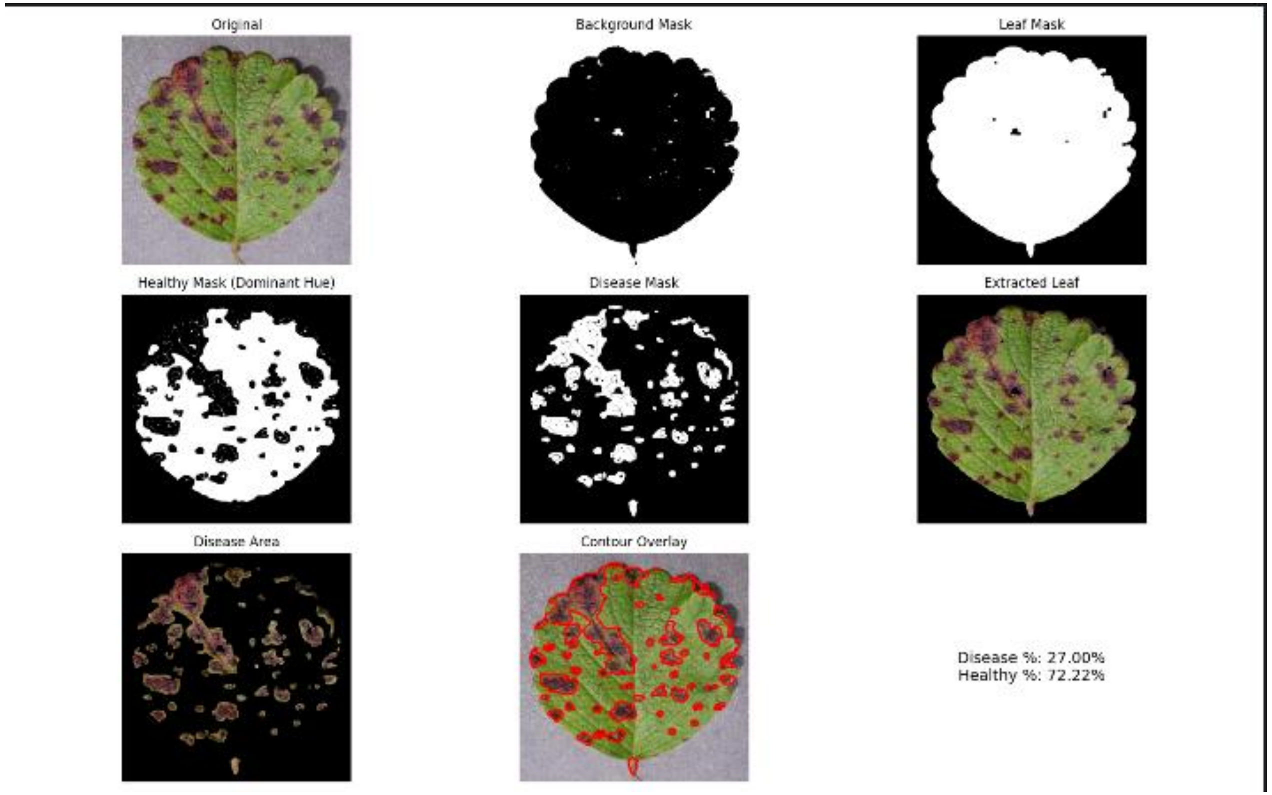
Sample Result



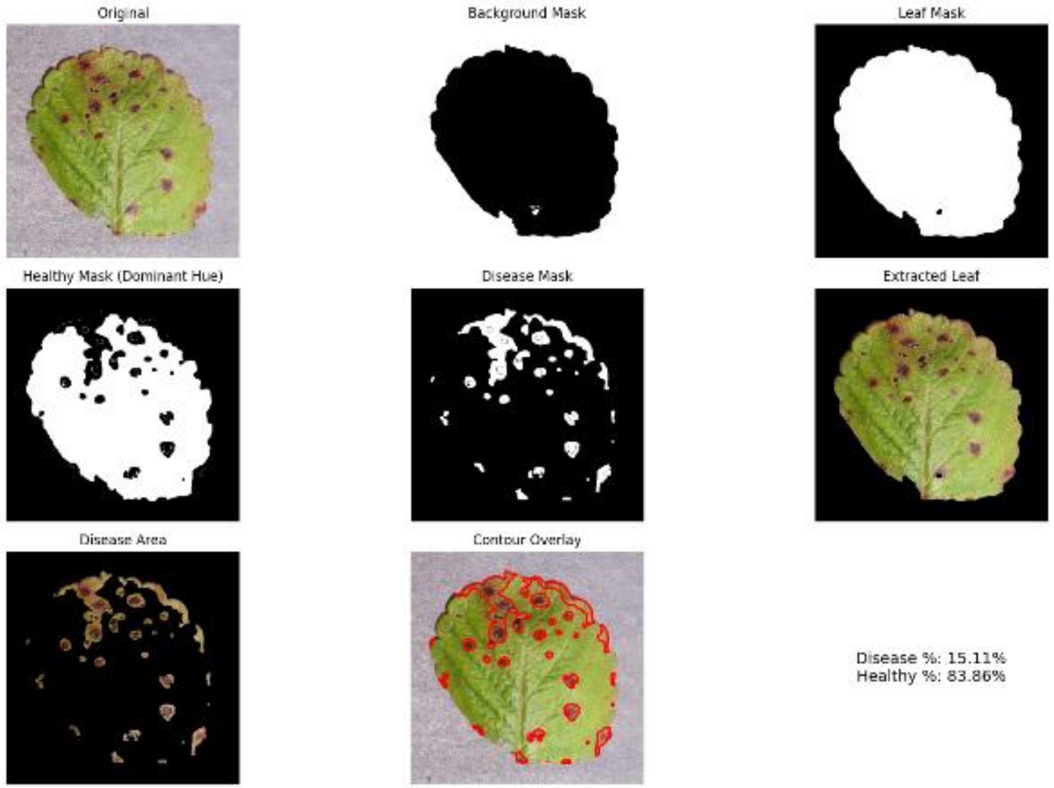
Sample Result



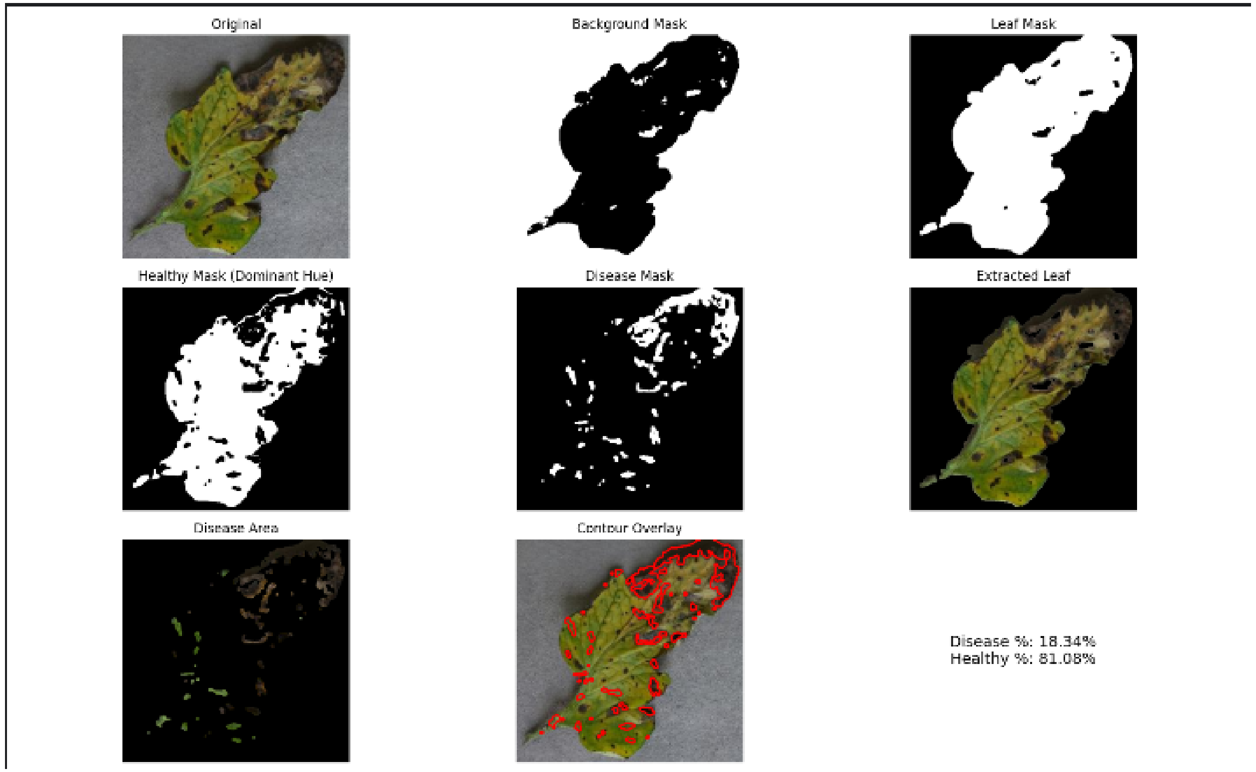
Sample Result



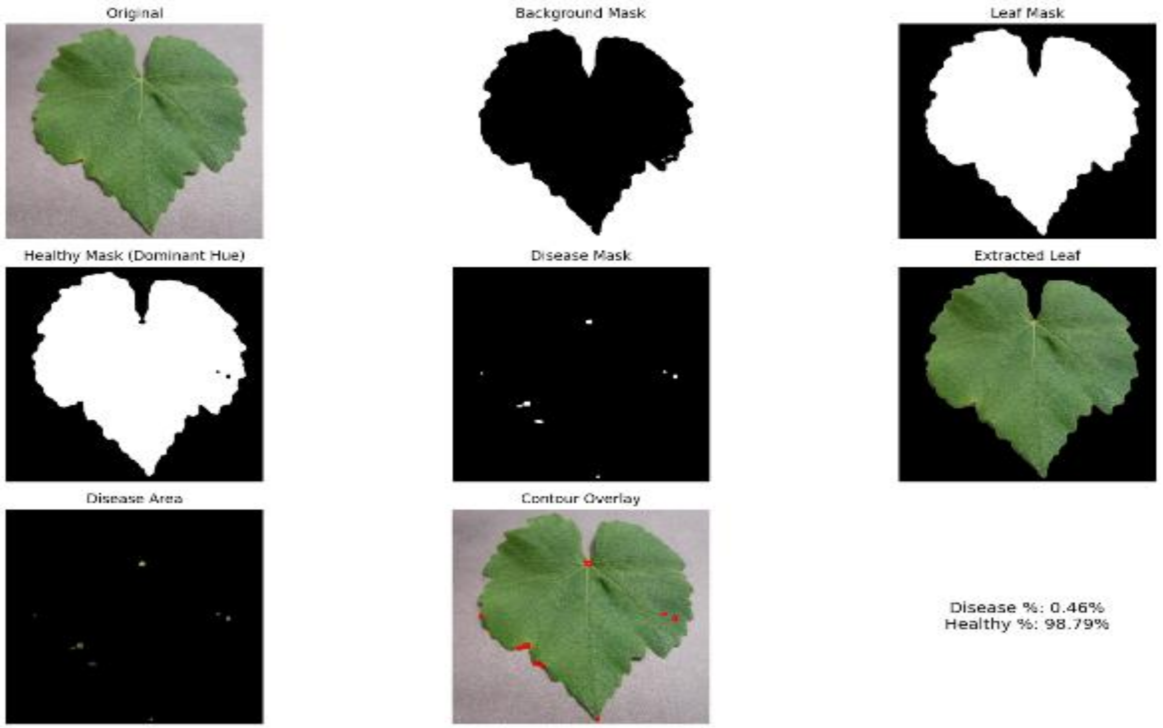
Sample Result



Sample Result



Sample Result



Q & A

THANK YOU!