*Original Research Article*

Automated plant leaf disease severity estimation using classical segmentation techniques

.

ABSTRACT

|  |
| --- |
| Aim: This study aimed to develop a cost-effective method for automated plant leaf disease severity estimation using classical segmentation techniques that can be readily deployed in resource-poor settings. Study Design: We evaluated the performance of four segmentation algorithms—global thresholding, adaptive thresholding, Otsu thresholding, and edge detection—across nine curated images of disease-affected leaves from tomato, bell pepper, and potato plants. Each image was segmented into healthy and diseased regions, and quantitative metrics (diseased pixel counts, percentage diseased, healthy-to-diseased ratios, and computational time) were computed to assess algorithm performance. Results: The segmentation methods operated nearly instantaneously (0–0.001 seconds per image). Adaptive and Otsu thresholding consistently demonstrated high sensitivity, capturing between 50% and 78% of diseased pixels with balanced healthy-to-diseased ratios. Global thresholding produced variable results, and while edge detection provided precise lesion boundaries, it considerably underestimated overall disease severity by detecting only 8–12% diseased areas. Conclusion: Based on the comparative analysis, Otsu thresholding emerged as the optimal approach for leaf disease severity estimation, offering robust performance with minimal computational overhead. Adaptive thresholding may serve as a complementary technique where enhanced sensitivity is required. These findings suggest that classical computer vision methods can effectively support plant disease diagnostics in resource-constrained environments. |

*Keywords: plant disease severity estimation, leaf image segmentation, Otsu thresholding, adaptive thresholding, classical computer vision, resource-poor settings, automated diagnostics*

1. INTRODUCTION

Plant diseases are a major threat to global agriculture, causing significant yield losses and economic damage every year (1–6). Early detection and accurate quantification of disease severity are essential for timely intervention and effective management. Traditionally, plant disease assessment has been performed manually by experts (7), a process that is both subjective and time-consuming. Recently, advances in computer vision have provided promising alternatives by automating the detection and segmentation of disease-affected regions on leaves. Many studies have leveraged machine learning and deep learning approaches for this purpose (8); however, significant barriers such as high computational costs, the requirement of large training datasets, and the need for specialized expertise limit their practical application in resource-poor regions (9–11). This challenge necessitates the exploration of robust yet simple classical computer vision algorithms that can operate with minimal infrastructure.

The problem addressed in this study is the automation of leaf disease severity estimation without recourse to machine learning techniques. We propose a solution based on classical segmentation algorithms—specifically, global thresholding, adaptive thresholding, Otsu thresholding, and edge detection—to isolate diseased portions from healthy areas of leaves. To evaluate the performance of these algorithms, we analyzed segmentation results across nine different images representing various diseases, covering species such as tomato, bell pepper, and potato. The scope of the work includes assessing each algorithm’s ability to accurately segment diseased regions using key metrics such as the percentage of diseased pixels, healthy-to-diseased ratios, and computational efficiency. The justification for this approach lies in its potential to offer a cost-effective, rapid, and resource-conservative solution for plant disease diagnostics in regions where access to extensive machine learning resources is limited. This study, therefore, aims to bridge the gap in technology by demonstrating that classical vision-based methods can reliably estimate leaf disease severity and support the implementation of automated systems in low-resource environments.

2. methodology

**2.1. Workflow**

A single directory held the dataset. The images were loaded from it, and results directories were created during the analysis for storing the results. The preprocessing phase involved resizing images to a target size, converting them into specific color spaces (grayscale or HSV), and applying Gaussian blur for noise reduction. Subsequently, segmentation techniques, including global thresholding, adaptive thresholding, Otsu thresholding, and edge detection were implemented to identify distinct image features. Each segmentation technique’s processing time was recorded for efficiency analysis. Boundaries within images were highlighted using contour detection, and metrics related to pixel classification (diseased vs. healthy) were calculated. Metrics included total pixels, diseased pixel count, percentage diseased, and the healthy-to-diseased ratio. These metrics were compiled into tabular formats and exported to Excel files for documentation. Visualization results, such as segmentation comparisons and original image overlays were generated and saved in the results directories.

**2.2. Dataset**

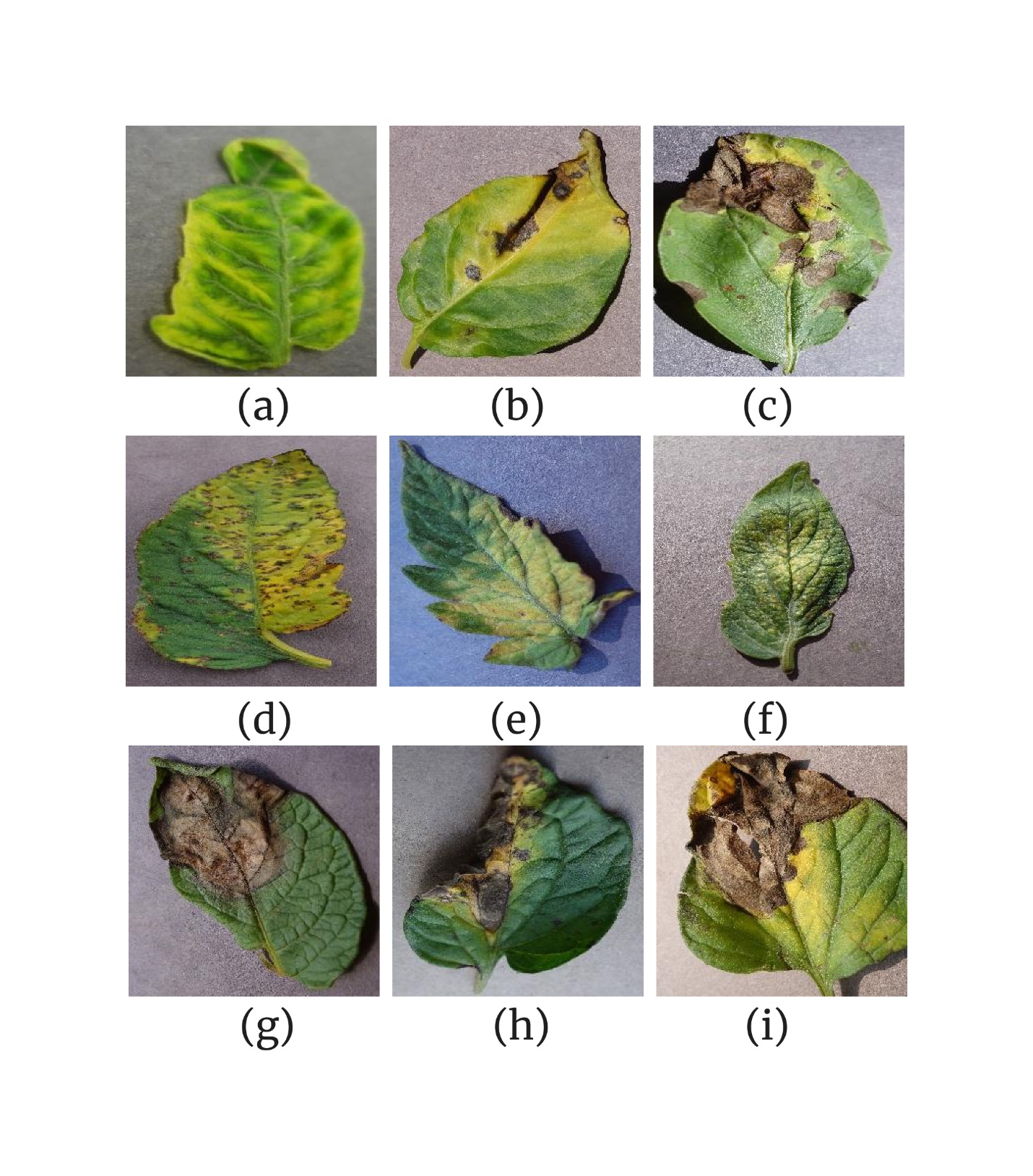
The dataset comprised nine curated images of disease-affected plant leaves (Figure 1), derived from the PlantVillage dataset (12). Each image, labeled sequentially from (a) to (i), was chosen not only for its clear representation of the disease class but also for the diversity in symptom presentation and lesion placement on the leaf.

Image (a) depicted a tomato leaf exhibiting the characteristic yellowing and curling associated with yellow leaf curl disease. In Image (b), a bell pepper leaf showed distinct dark necrotic spots with yellow halos, indicative of bacterial spot disease, while Image (c) presented a potato leaf marred by brown, concentric lesions surrounded by a yellow border, symptomatic of early blight disease.

The dataset further included Image (d), which featured another instance of bacterial spot disease on a tomato leaf, reinforcing the visual diagnosis with its repetitive, diagnostic patterns. Image (e) illustrated a tomato leaf affected by leaf mold disease, where widespread moldy patches and yellow discoloration suggested significant fungal colonization. In Image (f), the subtle yet telling stippling, yellowing, and discoloration on a tomato leaf pointed to a two-spotted spider mite infestation.

Image (g) captured the impact of late blight disease on a potato leaf, evidenced by extensive water-soaked lesions and subsequent tissue necrosis, while Image (h) returned to a tomato leaf showing early blight disease, echoing the concentric lesion patterns previously observed. Finally, Image (i) documented a tomato leaf with severe late blight disease, marked by extensive dark lesions and a water-soaked appearance, underscoring the aggressive nature of the fungal infection.

This carefully constructed collection was designed to represent a range of disease manifestations across economically important crops such as tomatoes, potatoes, and bell peppers. By ensuring variability in both the type of disease and the specific localization of symptoms, the dataset was suited for enhancing diagnostic differentiation in plant pathology and served as a robust foundation for developing computer vision algorithms.

****

**Fig. 1.** Disease-affected plant leaves used for this study. *(a) tomato leaf affected by yellow leaf curl disease; (b) bell pepper leaf affected by bacterial spot disease; (c) potato leaf affected by early blight disease; (d) tomato leaf affected by bacterial spot disease; (e) tomato leaf affected by leaf mold disease; (f) tomato leaf affected by a two-spotted spider mite; (g) potato leaf affected by late blight disease; (h) tomato leaf affected by early blight disease; (i) tomato leaf affected by late blight disease*.

**2.3. Within-image boundary highlighting**

The image highlight algorithm, often utilized in OpenCV for image segmentation, involves a series of mathematical operations to enhance and isolate specific features within an image. This process typically includes converting images to grayscale, applying thresholding, and using morphological operations to refine the segmentation.

**2.3.1. Grayscale conversion**

When an image is loaded into OpenCV, it is typically represented in BGR (Blue, Green, Red) format. To simplify the processing, images are often converted to grayscale (13–15). This conversion reduces the dimensionality of the image data from three color channels to a single channel, which represents the intensity of each pixel. Mathematically, the conversion from BGR to grayscale can be represented as:



where *α*, *β* and *γ* are coefficients that determine the contribution of each color channel to the grayscale value. In OpenCV, these coefficients are typically set to *α*=0.114, *β*=0.587, and *γ*=0.299, for BGR images.

**2.3.2. Thresholding**

Thresholding is a technique used to separate the foreground from the background by setting all pixels above or below a certain threshold to a specific value (16–18). This step is crucial for creating a binary mask that highlights the desired features. Let *I(x, y)* be the grayscale image, and *T* be the threshold value. The binary thresholding operation can be represented as:

**2.3.3. Morphological Operations**

These are used to refine the thresholded image by removing noise or filling gaps. The two primary operations are erosion and dilation. Erosion shrinks the foreground objects by removing pixels from the object boundaries (19). It can be mathematically represented using the Minkowski subtraction:

where *S* is the structuring element. Dilation expands the foreground objects by adding pixels to the object boundaries (20). It can be represented as:



**2.3.4. Mathematical representation of highlighting**

To mathematically represent the highlighting process, let *C* be the color used for highlighting (light green, in this study) and *I(x, y)* be the original image. The highlighted image *H(x, y)* can be represented as:



This formulation ensures that pixels corresponding to differences are highlighted with the chosen color, while other pixels retain their original values.

**2.3.5. Role of neighborhood size and kernel in within-image boundary highlighting**

The neighborhood size refers to the number of pixels surrounding a central pixel that are considered when applying an operation (21). This size determines how much of the image's local information is used in calculations. A larger neighborhood size means more pixels are considered, which can lead to more robust feature extraction but may also increase computational complexity. In within-image highlighting, the neighborhood size affects how morphological operations (like erosion and dilation) are applied. For example, a larger neighborhood size can help remove noise more effectively but might also alter the shape of objects more significantly.

A kernel, often referred to as a filter or mask, is a small matrix that slides over the image, performing a specific operation on each pixel and its neighbors (22). The kernel's values determine how each pixel is weighted relative to its neighbors. In within-image highlighting, kernels are used for operations like blurring, sharpening, or edge detection. For instance, a sharpening kernel can enhance the contrast of edges, making them more visible and thus "highlighting" them.

The interaction between the neighborhood size and the kernel is critical: The neighborhood size dictates how many pixels are considered when applying a kernel. A larger neighborhood size means the kernel will operate on more pixels, potentially leading to more pronounced effects. The kernel itself defines how each pixel in the neighborhood contributes to the output. Different kernels can perform different operations (e.g., smoothing vs. sharpening, as mentioned heretofore), depending on their values. Mathematically, the application of a kernel to a neighborhood can be represented as a convolution operation. Let *I(x, y)* be the input image, *K* be the kernel, and *N* be the neighborhood size. The output, *O(x, y)*, at position *(x, y)*, can be calculated as:

This formula shows how the kernel, *K*, is applied to each pixel in the neighborhood of size *N*, to produce the output image.

**2.4. Segmentation techniques**

**2.4.1. Global thresholding**

Here, an image is divided into two distinct regions based on a single threshold value (23,24). This method is particularly effective for images with bimodal intensity distributions, where the foreground and back ground have distinct intensity peaks. Mathematically, global thresholding can be described as follows: Let *f(x, y)* represent the intensity of a pixel at coordinates *(x, y)* in a grayscale image. The thresholded image, denoted as *g(x, y)*, is defined by applying a threshold, *T*, to *f(x, y)*:



This binary classification assigns pixels with intensities greater than *T* to the foreground (object) and those with intensities less than *T* to the background. More precisely, consider the following steps:

(1) Histogram analysis: The histogram of an image represents the distribution of pixel intensities. For global thresholding, the histogram is analyzed to determine the optimal threshold, *T*. The histogram can be represented as a discrete probability distribution *P(f)*, where *f* is the intensity level and *P(f)* is the probability of occurrence of that intensity.

(2) Threshold selection: The choice of *T* is critical. One common approach is to use Otsu's method, which maximizes the between-class variance to separate the foreground and background effectively. Mathematically, Otsu's method can be formulated as follows: Let *μ1* and *μ2* be the mean intensities of the foreground and background, respectively, and let *ω1* and *ω2* be the probabilities of the fore ground and background pixels. The between-class variance *σb2* is given by:



The optimal threshold, *T*, is the one that maximizes σb2.

(3) Optimization problem: The optimization problem for finding the optimal threshold can be formulated as:



Here, *ω1(T)*, *ω2(T)*, *μ1(T)*, and *μ2(T)* are functions of the threshold *T*, representing the probabilities and mean intensities of the foreground and background, respectively, given the threshold.

(4) Iterative thresholding: Another approach to finding an optimal threshold is through iterative methods. One such method involves starting with an initial estimate of *T*, segmenting the image, calculating the mean intensities of the foreground and background, and then updating *T* as the average of these means. This process is repeated until convergence:



where *Tn* is the threshold at iteration *n*.

**2.4.2. Adaptive thresholding**

Adaptive thresholding dynamically adjusts the threshold value for each pixel based on its local neighborhood (25,26). This approach is particularly useful in scenarios where images exhibit varying lighting conditions or complex backgrounds, making global thresholding methods less effective.

(1) Mathematical formulation: Let *I(x, y)* denote the intensity of a pixel at coordinates *(x, y)* in a grayscale image, *I*. The goal of adaptive thresholding is to segment the image into foreground and background regions by applying a threshold *T(x, y)* that varies across the image.

(2) Neighborhood definition: For each pixel *(x, y)*, a neighborhood *N(x, y)* is defined as a set of pixels within a certain distance *r* from *(x, y)*.

(3) Threshold calculation: The adaptive threshold *T(x, y)* for a pixel *(x, y)* can be calculated using the mean or median intensity of its neighborhood. Here, the focus is placed on the mean-based approach. The mean intensity *μ(x, y)* of the neighborhood *N(x, y)* is given by:

where |*N(x, y)*| is the number of pixels in the neighborhood. To refine the threshold, a constant *C* is often added or subtracted from the mean intensity. This adjustment helps in fine-tuning the segmentation based on the specific application:



The value of *C* is determined empirically based on the image characteristics.

(4) Binary segmentation: Once the adaptive threshold *T(x, y)* is determined for each pixel, the image can be segmented into foreground and background regions using the following rule:

If *I(x, y)* > *T(x, y)* , the pixel is classified as foreground. Otherwise, it is classified as background. Mathematically, this can be represented as:

where *S(x, y)* is the segmented image.

(5) Optimization of adaptive thresholding: To optimize the adaptive thresholding process, one might consider minimizing an objective function that measures the quality of segmentation. For instance, if *F* represents the set of foreground pixels and *B* the set of background pixels, an objective function could be defined as:

Minimizing this objective function involves adjusting the threshold *T(x, y)* and the constant *C* to achieve the best separation between foreground and background.

**2.4.3. Otsu thresholding**

This method is based on minimizing the intra-class variance or equivalently maximizing the inter-class variance (27,28). The algorithm operates on the histogram of the image, treating it as a probability distribution of pixel intensities.

(1) Mathematical formulation: The histogram of an image can be denoted as



where *hi* represents the number of pixels with intensity *i*, and *L* is the total number of possible intensity levels (e.g., *L*=256 for an 8-bit grayscale image). The total number of pixels in the image is given by

(2) Class probabilities and means: For a given threshold, *t*, the image is divided into two classes: foreground (*F*) and background (*B*). The probability of each class can be calculated as follows:

The probability of pixels belonging to the background class is given by the cumulative sum of histogram values up to the threshold, *t*:

The probability of pixels belonging to the foreground class is the complement of the background probability:



The mean intensity of the background class can be computed as:

The mean intensity of the foreground class can be computed as:

(3) Intra-class variance: The intra-class variance is a weighted sum of the variances of the two classes:

where *σB2* and *σF2* are the variances of the background and foreground classes, respectively. These variances are calculated as:



and

respectively.

(4) Inter-class variance: This can be calculated using the formula



(5) Otsu threshold optimization: The goal of Otsu's method is to find the threshold, *t*, that maximizes the inter-class variance, or equivalently minimizes the intra-class variance. This is achieved by iterating through all possible threshold values and computing the inter-class variance for each. The threshold that yields the maximum inter-class variance is chosen as the optimal threshold.

**2.4.4. Edge detection**

Edge detection involves the identification of boundaries or edges within an image. This can be achieved through various algorithms, each with its unique mathematical formulation. The mathematical details of the Canny edge detection algorithm (29,30), one of the most widely used and robust methods for edge detection, shall be explained first.

(1) Mathematical formulation of Canny Edge detection: The first step in the Canny edge detection algorithm is to smooth the input image using a Gaussian filter. This is done to reduce noise and enhance the quality of edge detection. The Gaussian filter is defined by the following kernel:

where *σ* is the standard deviation of the Gaussian distribution. The convolution of the image *I(x, y)* with the Gaussian kernel *G(x, y)* can be expressed as:

After smoothing, the next step is to calculate the gradient of the image intensity function. This is typically done using the Sobel operator, which approximates the gradient in the *x* and *y* directions using the following kernels:



The gradient components, *Gx* and *Gy*, are calculated by convolving these kernels with the smoothed image:



The magnitude of the gradient, *M(x, y)*, and its direction, *θ(x, y)*, are then computed as:

To refine the edges, non-maximum suppression is applied. This involves checking each pixel to see if it has the maximum gradient magnitude along the direction of the gradient. If not, the pixel is suppressed (set to zero). Mathematically, this can be represented as:

where

refers to the gradient magnitudes of the neighboring pixels along the direction of the gradient.

Finally, double thresholding is applied to determine strong and weak edges. Two thresholds, *Thigh* and *Tlow*, are used:

Strong edges are retained as part of the final edge map. Weak edges are only included if they are connected to strong edges.

(2) Mathematical representation of other Edge detection algorithms:

(Sobel Edge detection) The Sobel operator (31–33) calculates the gradient directly without smoothing:

The gradient magnitude is then computed as for Canny. Laplacian Edge detection (34) uses the second derivative to find edges:



This can be approximated using a Laplacian kernel:

Edges are detected where the Laplacian is zero or changes sign.

**2.5. Computed image metrics**

**2.5.1. Total pixels**

As already seen, the total number of pixels (𝑁) in an image can be expressed mathematically as:

where 𝑥 and 𝑦 are the dimensions of the image in pixels (width and height).

**2.5.2. Disease pixels**

Let 𝐷 represent the number of diseased pixels. This can be determined by applying a threshold to the image data, where pixels above or below the threshold are classified as diseased. Using a binary classification function *f(i, j)*, which outputs 1 for diseased pixels and 0 otherwise, the total number of diseased pixels is:



**2.5.3. Percentage of diseased pixels**

The percentage of diseased pixels, *Pdiseased*, is calculated by dividing the number of diseased pixels by the total number of pixels and multiplying by 100:

**2.5.4. Healthy-to-diseased ratio**

Let *H* represent the number of healthy pixels, which can be determined as:

The healthy-to-diseased ratio, *Rhealthy:diseased*, is calculated using the formula

**2.5.5. Mathematical representation**

Using probability theory, the distribution of diseased and healthy pixels can be modeled as a binomial distribution. Let *p* represent the probability of a pixel being diseased. Let *q = 1 – p* represent the probability of a pixel being healthy. The expected number of diseased pixels is:

and the expected number of healthy pixels is:

The probability of observing 𝐷 diseased pixels in an image can be modeled using the binomial probability mass function:

where

represents the binomial coefficient.

3. results and discussion

**3.1. Segmentation and quantitative metrics**

The segmentation results were analyzed across the nine images representing leaf diseases affecting several plant types, using four classical computer vision algorithms. Each figure (2–10) displayed the segmentation results visually, divided into six panels: the original image (a), contour-detection highlighting boundaries (b), global thresholding segmentation (c), adaptive thresholding segmentation (d), Otsu thresholding segmentation (e), and edge detection segmentation (f). Corresponding tables (1–9) provided quantitative metrics, including diseased pixel counts, percentage of diseased areas, healthy-to-diseased ratios, and computational time.

Figure 2 and Table 1 highlighted the segmentation of a tomato leaf affected by yellow leaf curl disease. Global thresholding detected 47.92% of pixels as diseased, with a healthy-to-diseased ratio of 1.09. Adaptive thresholding produced the highest diseased percentage (74.19%), with a ratio of 0.35. Otsu thresholding identified 58.58% diseased pixels, with a ratio of 0.71. Edge detection was the most conservative, detecting only 2.14% as diseased, with a ratio of 45.68.

In Figure 3 and Table 2, the segmentation of a bell pepper leaf affected by bacterial spot disease demonstrated similar trends. Global thresholding identified 73.72% of pixels as diseased, while adaptive thresholding showed a slightly lower percentage of 67.81%. Otsu thresholding captured the highest diseased percentage at 76.72%. Edge detection, as expected, conservatively identified only 4.40% of pixels as diseased, yielding a high healthy-to-diseased ratio of 21.74.

The segmentation of a potato leaf affected by early blight disease, shown in Figure 4 and Table 3, revealed variations in algorithm sensitivity. Adaptive thresholding detected 64.62% of pixels as diseased, compared to 72.20% by global thresholding and 78.59% by Otsu thresholding. Edge detection again performed conservatively, with only 10.67% of pixels identified as diseased.

For the tomato leaf affected by bacterial spot disease (Figure 5 and Table 4), adaptive thresholding detected 69.83% of diseased pixels, compared to 48.63% by Otsu thresholding and 28.23% by global thresholding. Edge detection identified a mere 7.11%, reflecting its precision-focused approach.

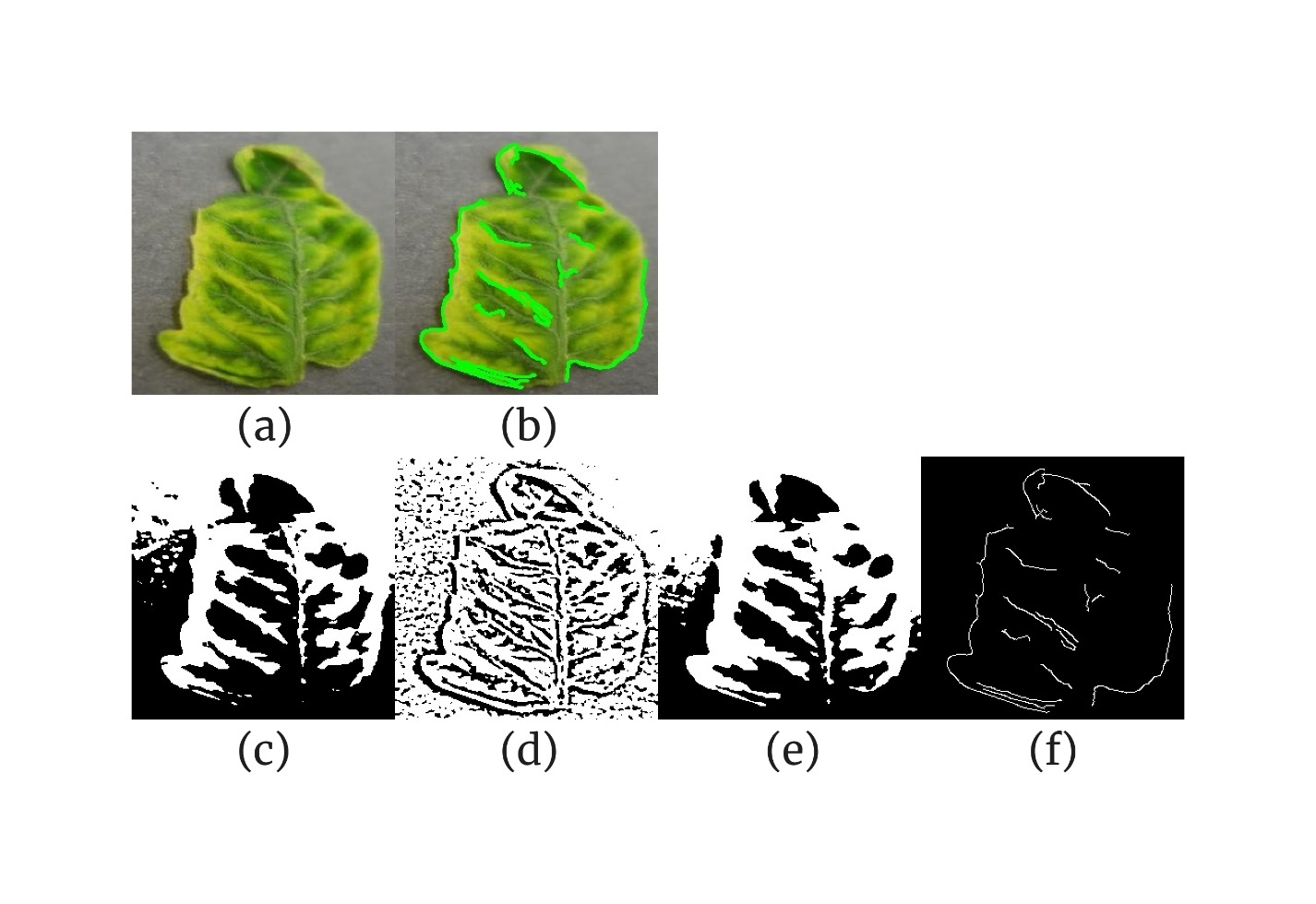
Figure 6 and Table 5 examined segmentation of tomato leaves with leaf mold disease. Adaptive thresholding identified 67.25% diseased pixels, followed closely by Otsu thresholding with 74.10%. Global thresholding estimated 42.46%, while edge detection captured only 6.35%.

Figure 7 and Table 6 evaluated the tomato leaf affected by two-spotted spider mite. Adaptive thresholding flagged 63.62% of pixels as diseased, with Otsu thresholding at 52.79%. Global thresholding estimated 58.04% diseased pixels, while edge detection marked 8.93%.

In Figure 8 and Table 7, segmentation of the potato leaf with late blight disease showed adaptive thresholding as the most sensitive, estimating 67.63% diseased pixels. Global thresholding and Otsu thresholding detected 25.96% and 55.95%, respectively. Edge detection identified 8.95%.

Early blight disease on tomato leaves (Figure 9 and Table 8) revealed adaptive thresholding detected the highest diseased percentage of 70.83%, followed by Otsu thresholding at 50.32%. Global thresholding captured 34.01% diseased pixels. Edge detection showed consistent conservatism with 6.64%.

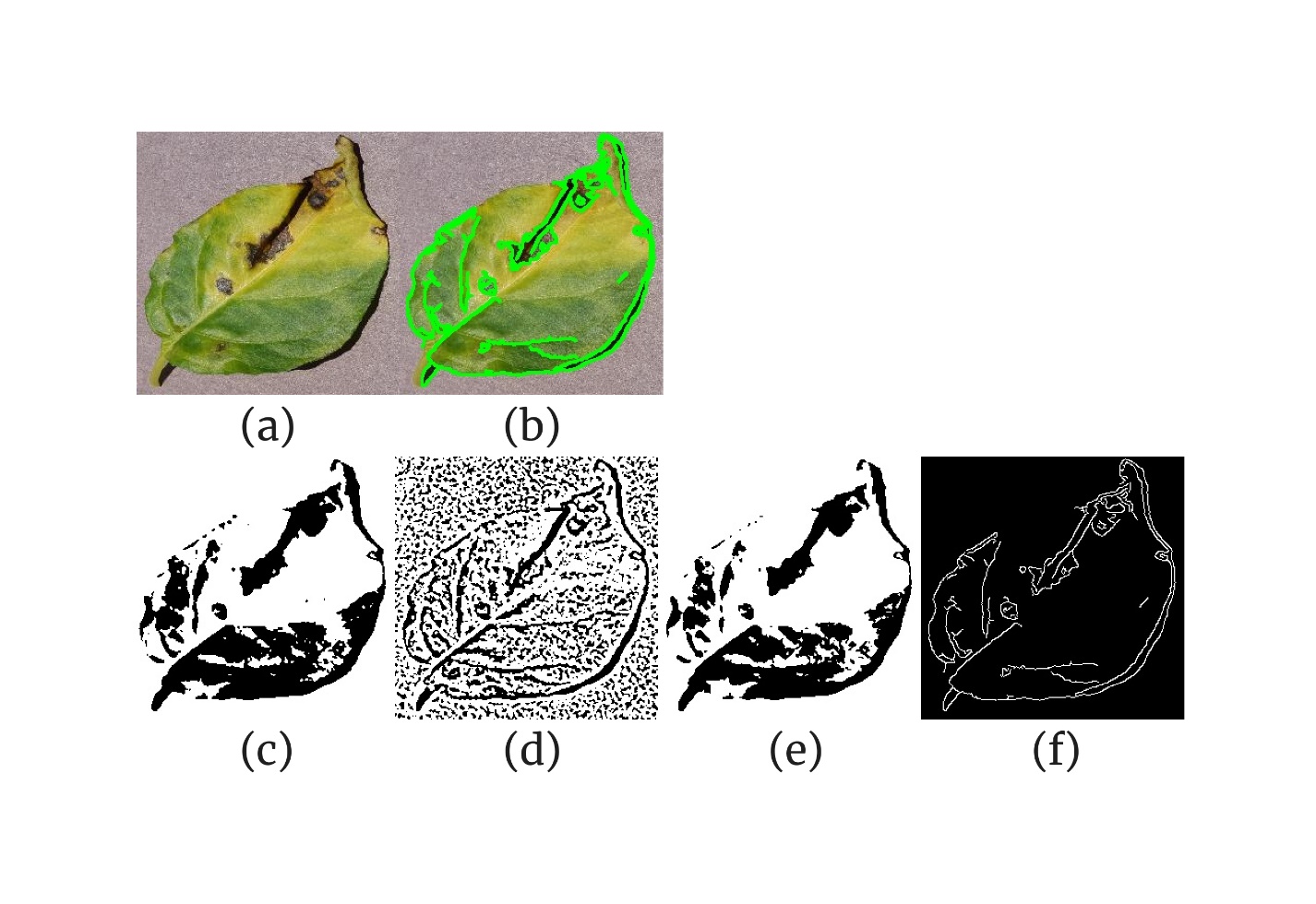
Finally, Figure 10 and Table 9 showcased segmentation of tomato leaves affected by late blight disease. Adaptive thresholding estimated 64.71% diseased pixels, followed closely by global thresholding at 54.73% and Otsu thresholding at 51.39%. Edge detection marked 11.10%.



**Figure 2**. Segmentation of the healthy and diseased portions of the tomato leaf affected by yellow leaf curl disease. *(a) original image; (b) within-image boundaries highlighted using contour detection; (c) global thresholding-based segmentation; (d) adaptive thresholding-based segmentation; (e) Otsu thresholding-based segmentation; (f) edge detection-based segmentation*.

**Table 1**. Per segmentation technique metrics for a tomato leaf affected by yellow leaf curl disease

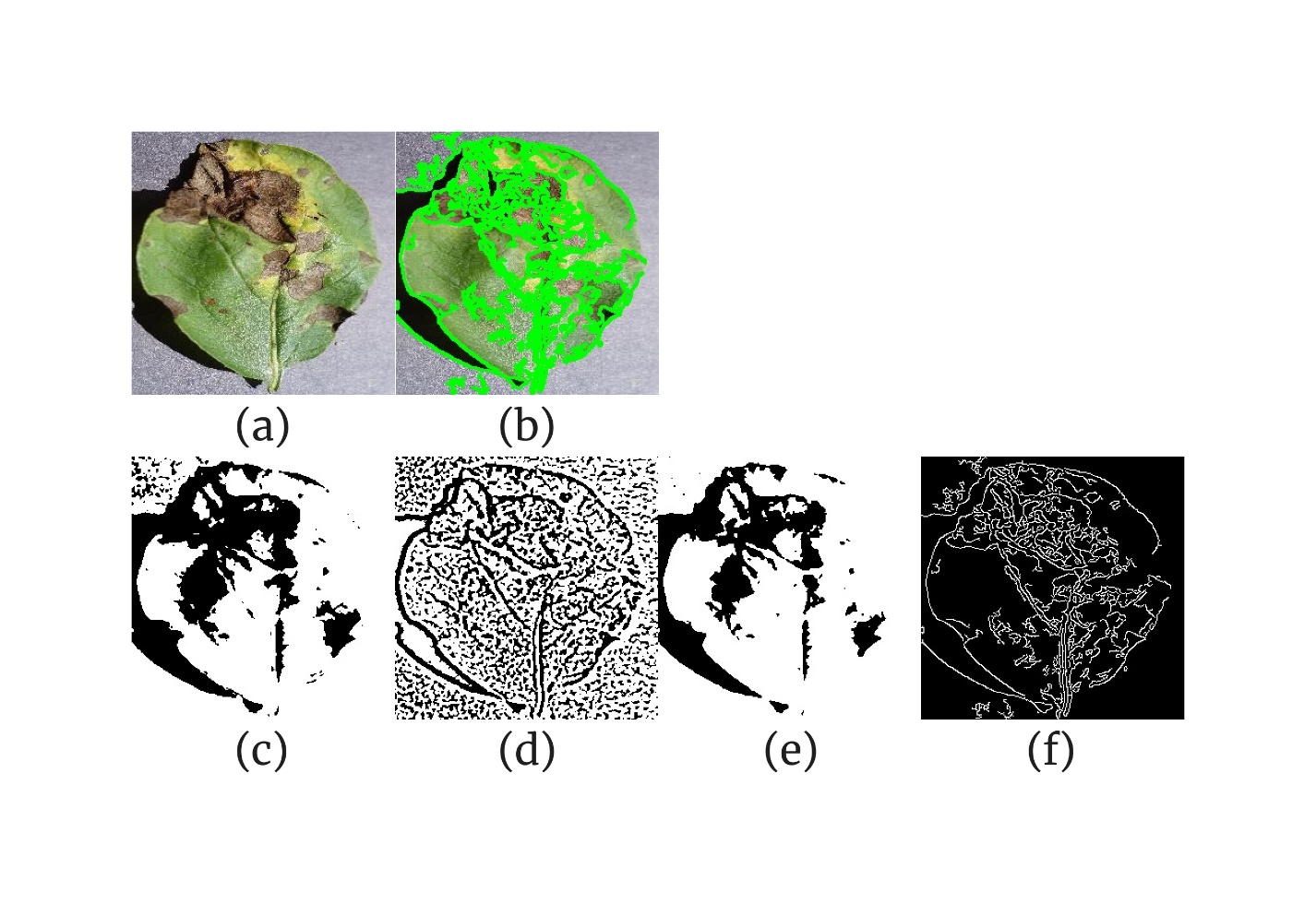
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technique** | **Total Pixels** | **Diseased Pixels** | **Percentage Diseased** | **Healthy-to-Diseased Ratio** | **Time Taken (s)** |
| Global Thresholding | 65536 | 31405 | 47.92022705 | 1.086801465 | 0 |
| Adaptive Thresholding | 65536 | 48624 | 74.19433594 | 0.34781178 | 0.00299859 |
| Otsu Thresholding | 65536 | 38390 | 58.57849121 | 0.707111227 | 0.000999928 |
| Edge Detection | 65536 | 1404 | 2.142333984 | 45.67806268 | 0.001998663 |



**Figure 3**. Segmentation of the healthy and diseased portions of the bell pepper leaf affected by bacterial spot disease. *(a) original image; (b) within-image boundaries highlighted using contour detection; (c) global thresholding-based segmentation; (d) adaptive thresholding-based segmentation; (e) Otsu thresholding-based segmentation; (f) edge detection-based segmentation*.

**Table 2**. Per segmentation technique metrics for a bell pepper leaf affected by bacterial spot disease

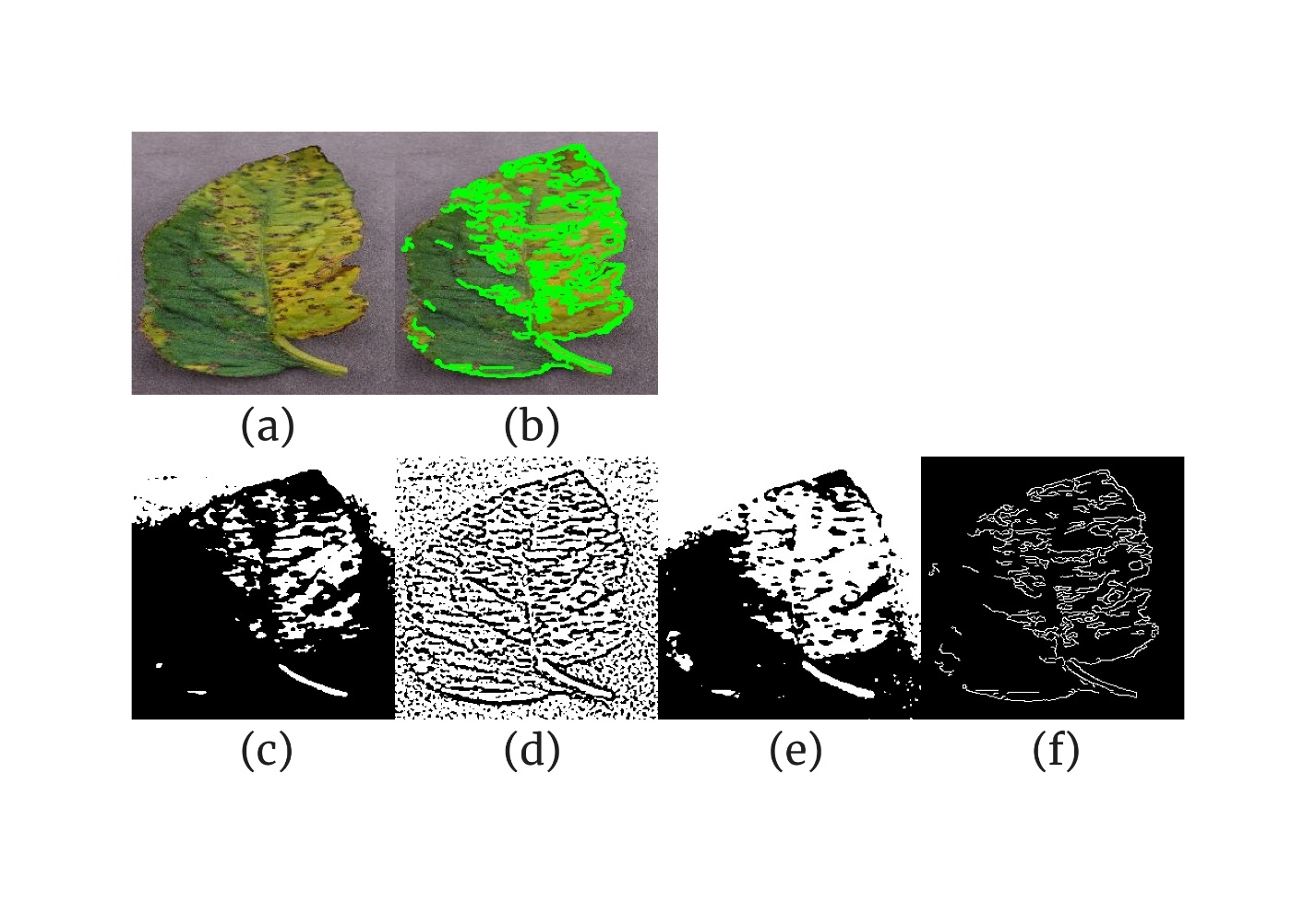
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technique** | **Total Pixels** | **Diseased Pixels** | **Percentage Diseased** | **Healthy-to-Diseased Ratio** | **Time Taken (s)** |
| Global Thresholding | 65536 | 48316 | 73.72436523 | 0.356403676 | 0 |
| Adaptive Thresholding | 65536 | 44443 | 67.81463623 | 0.474607925 | 0 |
| Otsu Thresholding | 65536 | 50276 | 76.71508789 | 0.303524545 | 0 |
| Edge Detection | 65536 | 2882 | 4.397583008 | 21.73976405 | 0.000998974 |



**Figure 4**. Segmentation of the healthy and diseased portions of the potato leaf affected by early blight disease. *(a) original image; (b) within-image boundaries highlighted using contour detection; (c) global thresholding-based segmentation; (d) adaptive thresholding-based segmentation; (e) Otsu thresholding-based segmentation; (f) edge detection-based segmentation*.

**Table 3**. Per segmentation technique metrics for a potato leaf affected by early blight disease

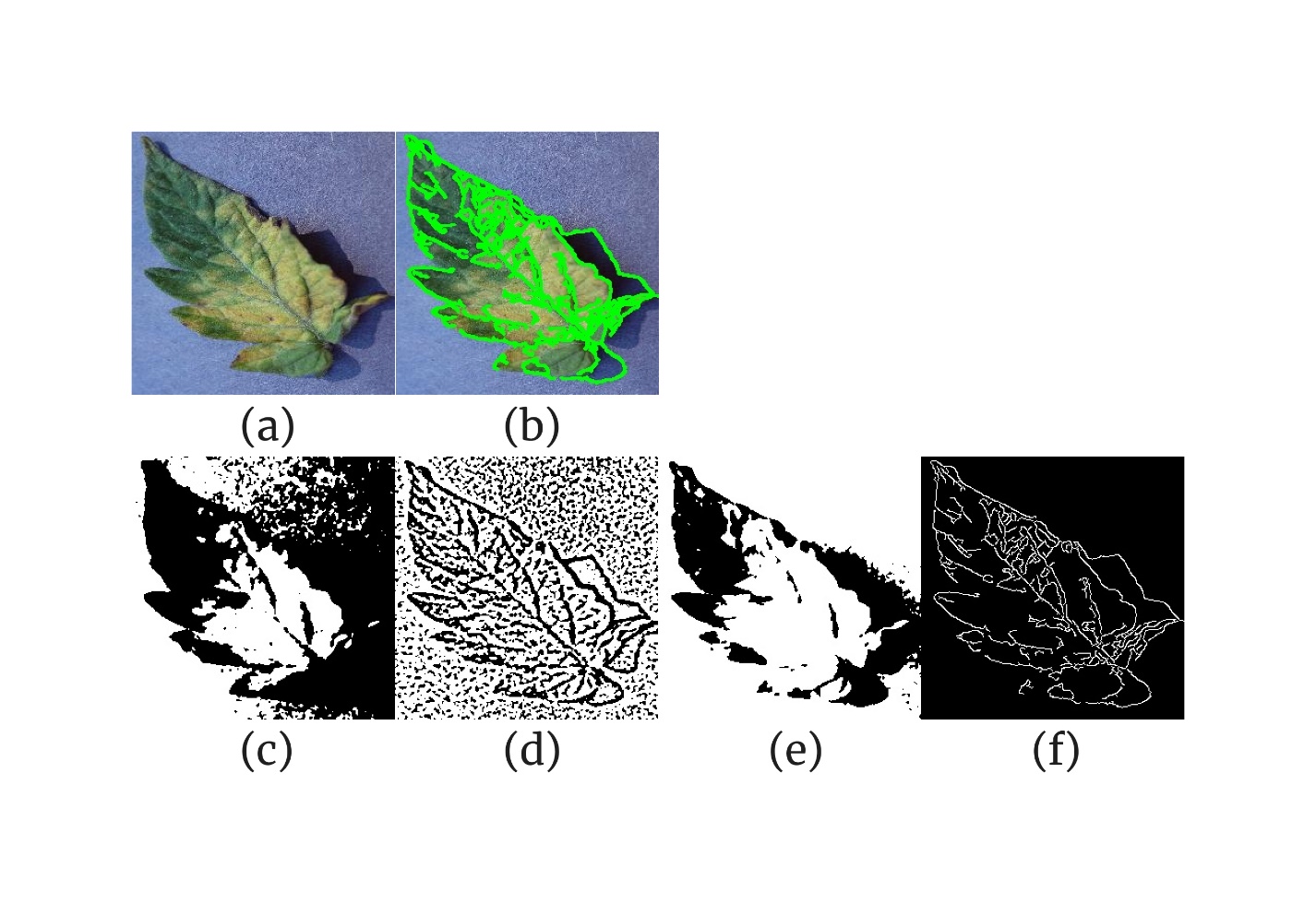
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technique** | **Total Pixels** | **Diseased Pixels** | **Percentage Diseased** | **Healthy-to-Diseased Ratio** | **Time Taken (s)** |
| Global Thresholding | 65536 | 47320 | 72.20458984 | 0.384953508 | 0 |
| Adaptive Thresholding | 65536 | 42347 | 64.61639404 | 0.547594871 | 0 |
| Otsu Thresholding | 65536 | 51502 | 78.58581543 | 0.272494272 | 0 |
| Edge Detection | 65536 | 6994 | 10.67199707 | 8.370317415 | 0.000998259 |



**Figure 5**. Segmentation of the healthy and diseased portions of the tomato leaf affected by bacterial spot disease. *(a) original image; (b) within-image boundaries highlighted using contour detection; (c) global thresholding-based segmentation; (d) adaptive thresholding-based segmentation; (e) Otsu thresholding-based segmentation; (f) edge detection-based segmentation*.

**Table 4**. Per segmentation technique metrics for a tomato leaf affected by bacterial spot disease

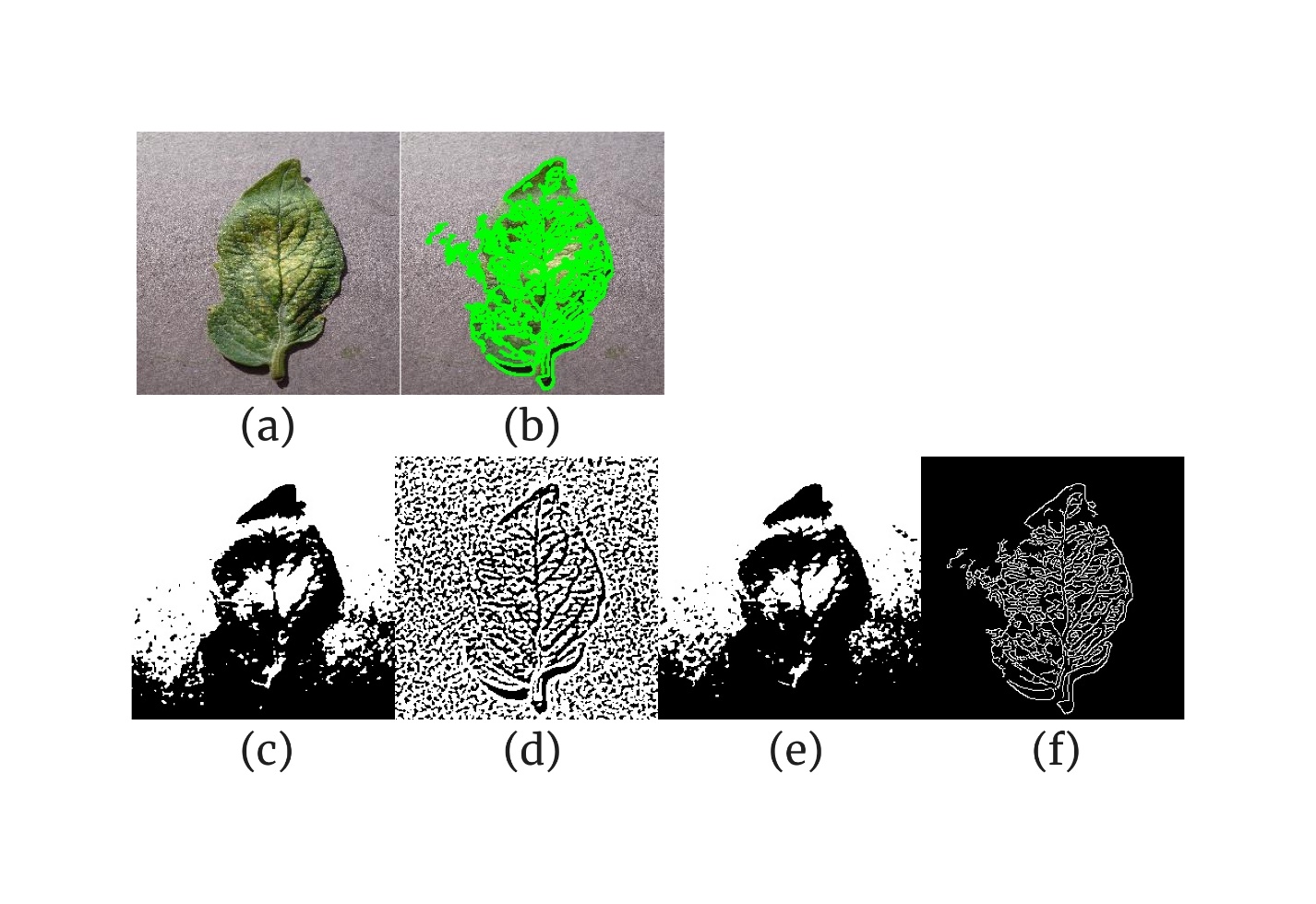
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technique** | **Total Pixels** | **Diseased Pixels** | **Percentage Diseased** | **Healthy-to-Diseased Ratio** | **Time Taken (s)** |
| Global Thresholding | 65536 | 18502 | 28.23181152 | 2.542103556 | 0 |
| Adaptive Thresholding | 65536 | 45761 | 69.82574463 | 0.432136535 | 0 |
| Otsu Thresholding | 65536 | 31870 | 48.62976074 | 1.056353938 | 0 |
| Edge Detection | 65536 | 4659 | 7.109069824 | 13.06653788 | 0.000998497 |



**Figure 6**. Segmentation of the healthy and diseased portions of the tomato leaf affected by leaf mold disease. *(a) original image; (b) within-image boundaries highlighted using contour detection; (c) global thresholding-based segmentation; (d) adaptive thresholding-based segmentation; (e) Otsu thresholding-based segmentation; (f) edge detection-based segmentation*.

**Table 5**. Per segmentation technique metrics for a tomato leaf affected by leaf mold disease

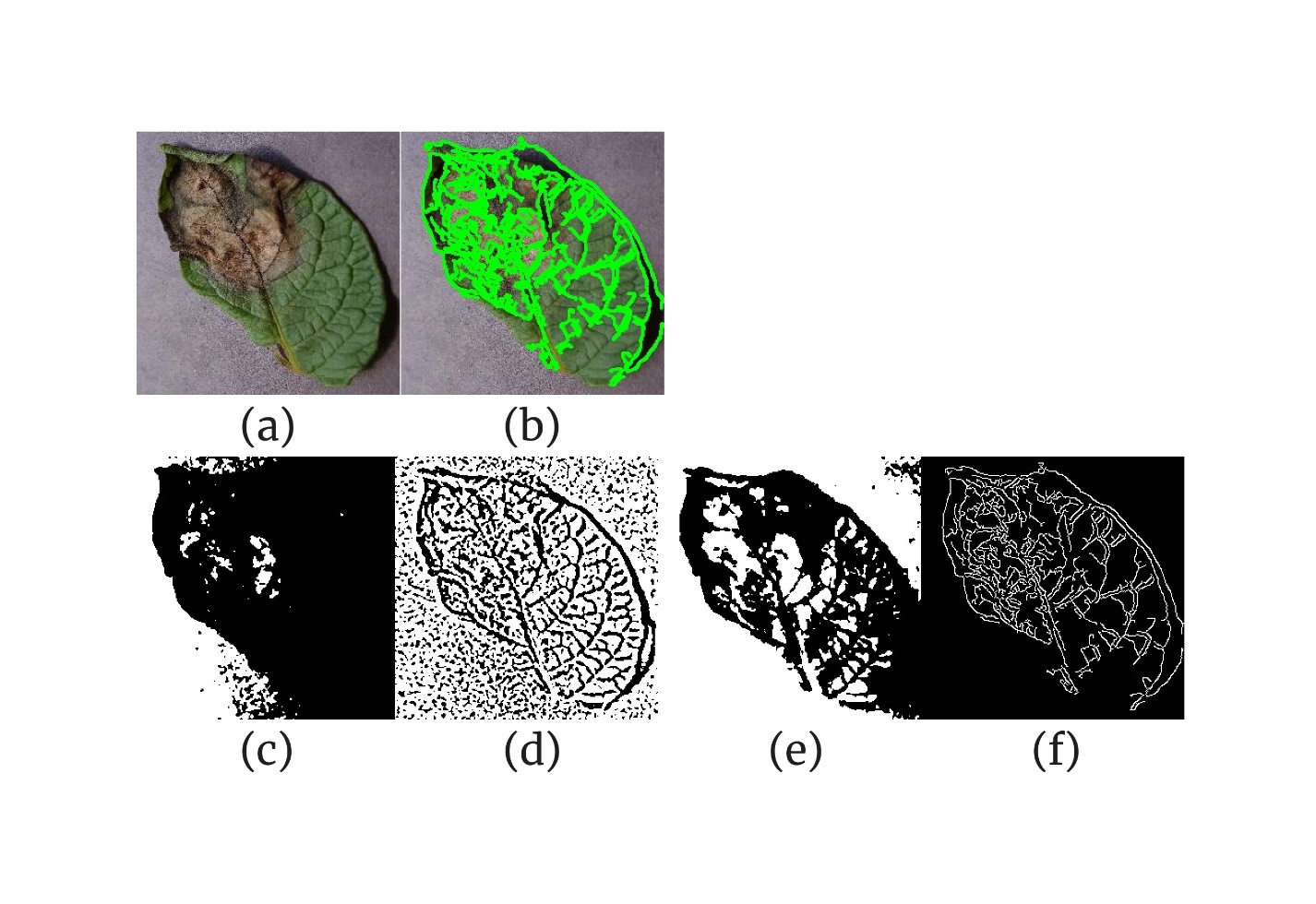
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technique** | **Total Pixels** | **Diseased Pixels** | **Percentage Diseased** | **Healthy-to-Diseased Ratio** | **Time Taken (s)** |
| Global Thresholding | 65536 | 27824 | 42.45605469 | 1.355376653 | 0 |
| Adaptive Thresholding | 65536 | 44072 | 67.24853516 | 0.487021238 | 0.000998259 |
| Otsu Thresholding | 65536 | 48561 | 74.09820557 | 0.349560347 | 0 |
| Edge Detection | 65536 | 4160 | 6.34765625 | 14.75384615 | 0.001000643 |



**Figure 7**. Segmentation of the healthy and diseased portions of the tomato leaf affected by a two-spotted spider mite. *(a) original image; (b) within-image boundaries highlighted using contour detection; (c) global thresholding-based segmentation; (d) adaptive thresholding-based segmentation; (e) Otsu thresholding-based segmentation; (f) edge detection-based segmentation*.

**Table 6**. Per segmentation technique metrics for a tomato leaf affected by a two-spotted spider mite

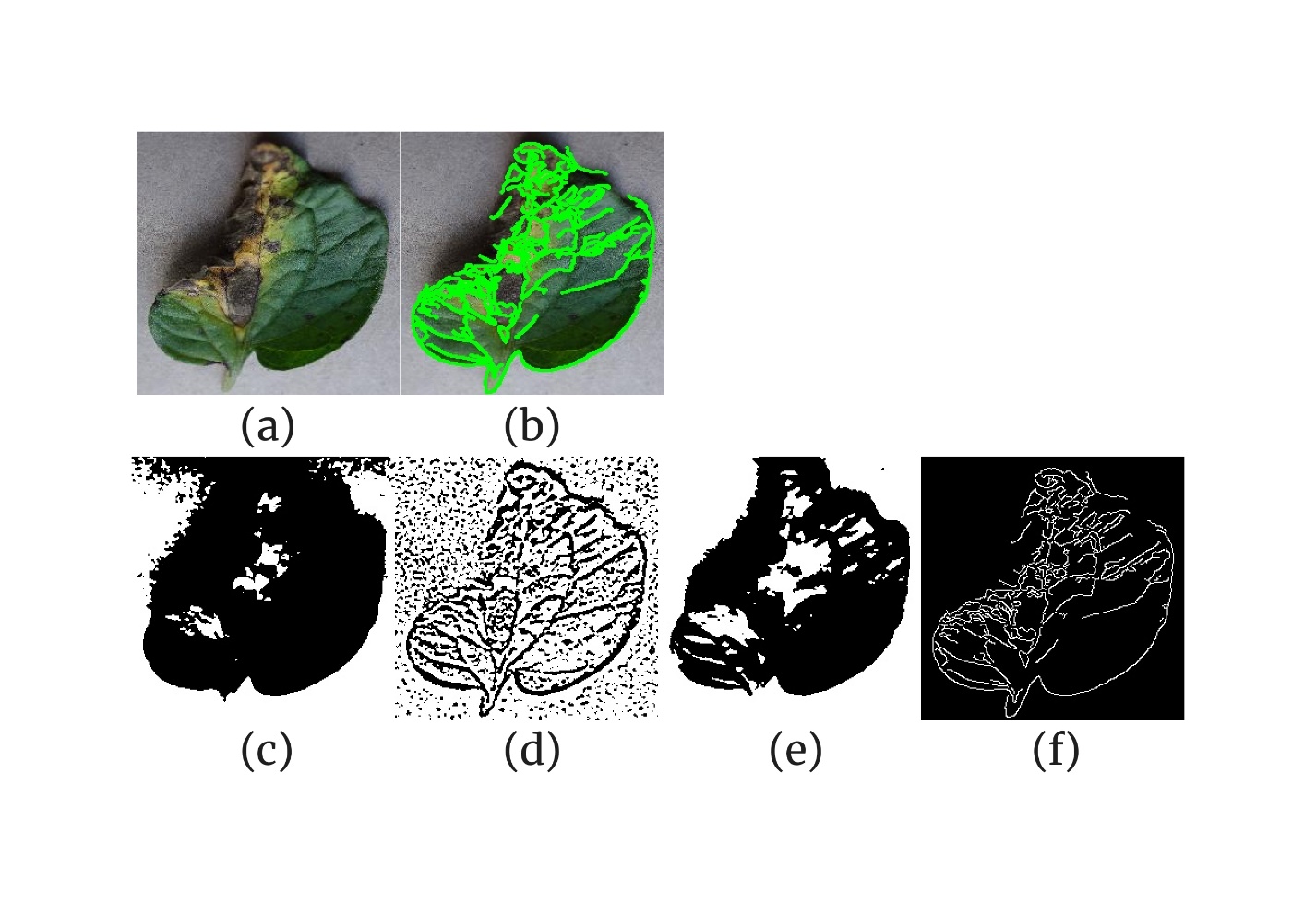
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technique** | **Total Pixels** | **Diseased Pixels** | **Percentage Diseased** | **Healthy-to-Diseased Ratio** | **Time Taken (s)** |
| Global Thresholding | 65536 | 38038 | 58.04138184 | 0.72290867 | 0 |
| Adaptive Thresholding | 65536 | 41691 | 63.61541748 | 0.571945984 | 0.000998974 |
| Otsu Thresholding | 65536 | 34597 | 52.79083252 | 0.894268289 | 0 |
| Edge Detection | 65536 | 5850 | 8.926391602 | 10.20273504 | 0 |



**Figure 8**. Segmentation of the healthy and diseased portions of the potato leaf affected by late blight disease. *(a) original image; (b) within-image boundaries highlighted using contour detection; (c) global thresholding-based segmentation; (d) adaptive thresholding-based segmentation; (e) Otsu thresholding-based segmentation; (f) edge detection-based segmentation*.

**Table 7**. Per segmentation technique metrics for a potato leaf affected by late blight disease

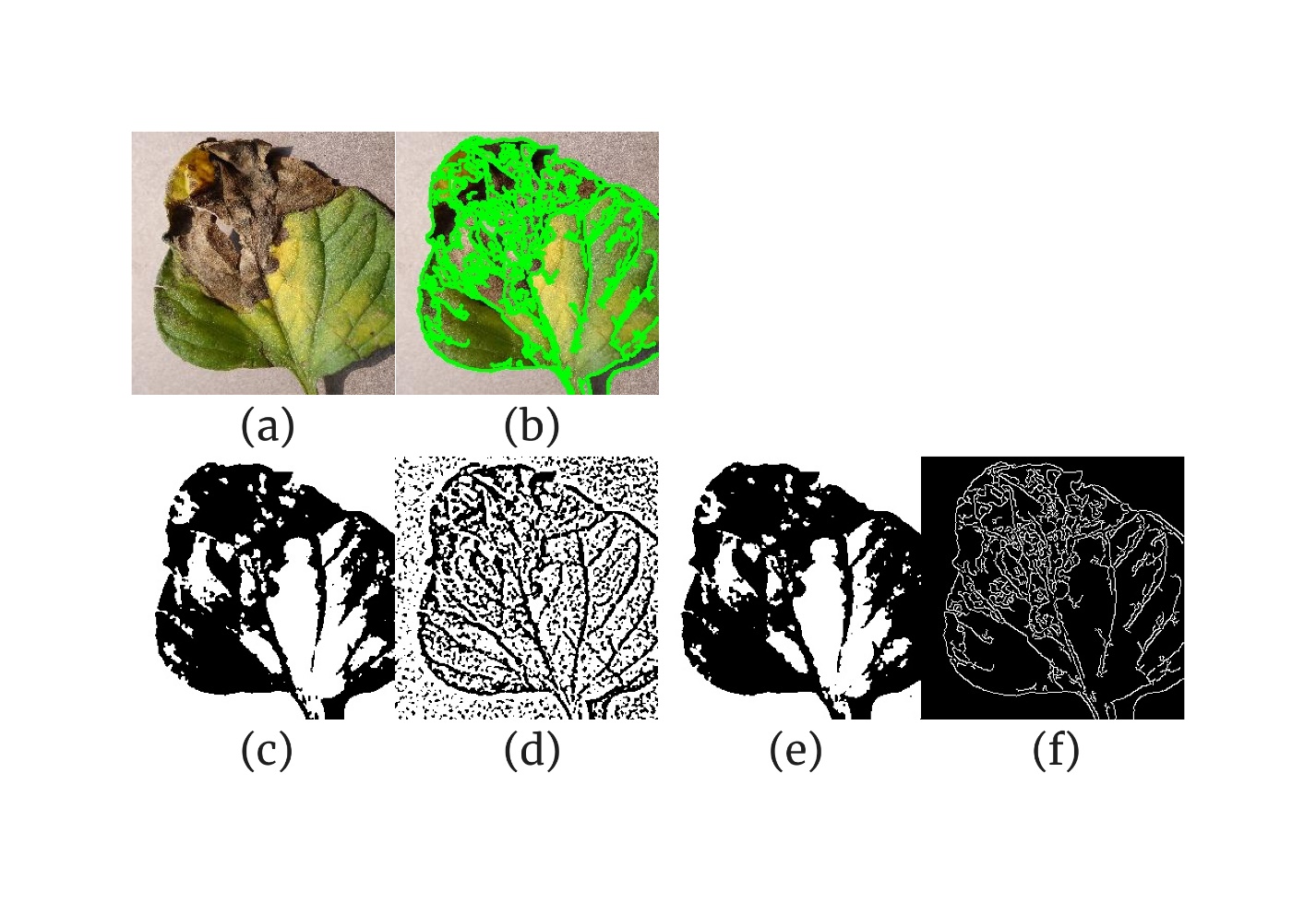
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technique** | **Total Pixels** | **Diseased Pixels** | **Percentage Diseased** | **Healthy-to-Diseased Ratio** | **Time Taken (s)** |
| Global Thresholding | 65536 | 17016 | 25.96435547 | 2.851433945 | 0 |
| Adaptive Thresholding | 65536 | 44319 | 67.62542725 | 0.478733726 | 0.000994205 |
| Otsu Thresholding | 65536 | 36665 | 55.9463501 | 0.787426701 | 0 |
| Edge Detection | 65536 | 5864 | 8.947753906 | 10.17598909 | 0.00099802 |



**Figure 9**. Segmentation of the healthy and diseased portions of the tomato leaf affected by early blight disease. *(a) original image; (b) within-image boundaries highlighted using contour detection; (c) global thresholding-based segmentation; (d) adaptive thresholding-based segmentation; (e) Otsu thresholding-based segmentation; (f) edge detection-based segmentation*.

**Table 8**. Per segmentation technique metrics for a tomato leaf affected by early blight disease

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technique** | **Total Pixels** | **Diseased Pixels** | **Percentage Diseased** | **Healthy-to-Diseased Ratio** | **Time Taken (s)** |
| Global Thresholding | 65536 | 22290 | 34.01184082 | 1.940152535 | 0 |
| Adaptive Thresholding | 65536 | 46421 | 70.83282471 | 0.411774843 | 0.000998974 |
| Otsu Thresholding | 65536 | 32977 | 50.31890869 | 0.987324499 | 0 |
| Edge Detection | 65536 | 4349 | 6.636047363 | 14.06921131 | 0.000999689 |



**Figure 10**. Segmentation of the healthy and diseased portions of the tomato leaf affected by late blight disease. *(a) original image; (b) within-image boundaries highlighted using contour detection; (c) global thresholding-based segmentation; (d) adaptive thresholding-based segmentation; (e) Otsu thresholding-based segmentation; (f) edge detection-based segmentation*.

**Table 9**. Per segmentation technique metrics for a tomato leaf affected by late blight disease

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Technique** | **Total Pixels** | **Diseased Pixels** | **Percentage Diseased** | **Healthy-to-Diseased Ratio** | **Time Taken (s)** |
| Global Thresholding | 65536 | 35869 | 54.73175049 | 0.827093033 | 0 |
| Adaptive Thresholding | 65536 | 42411 | 64.71405029 | 0.545259485 | 0 |
| Otsu Thresholding | 65536 | 33677 | 51.38702393 | 0.946016569 | 0 |
| Edge Detection | 65536 | 7276 | 11.10229492 | 8.007146784 | 0.000998974 |

**3.2. Comparative analysis of segmentation methods**

**3.2.1. Performance metrics comparison**

Across all nine figures and tables, adaptive and Otsu thresholding consistently demonstrated high sensitivity in detecting diseased regions, capturing between 50–78% diseased pixels. Global thresholding produced variable results, while edge detection was conservative across all datasets, detecting an average of 8–12% diseased pixels and maintaining high healthy-to-diseased ratios (8–45).

**3.2.2. Consistency and reliability**

Adaptive thresholding repeatedly captured the largest diseased area, suggesting its robustness in identifying subtle intensity changes. Otsu thresholding balanced sensitivity with specificity, making it a versatile choice. Edge detection excelled at boundary precision but consistently underestimated total diseased areas.

**3.2.3. Computational efficiency**

All methods operated with near-instantaneous processing times (approximately 0–0.001 seconds). This efficiency ensures feasibility for deployment in real-time agricultural diagnostics in resource-poor areas.

**3.3. Algorithm ranking**

Based on sensitivity, specificity, consistency, and practical application, the algorithms were ranked as follows: (1) Otsu Thresholding: Balanced performance across all figures, producing consistent results with moderate diseased area percentages and near-unity healthy-to-diseased ratios. (2) Adaptive Thresholding: Most sensitive method, consistently capturing the largest diseased area; however, it may overestimate diseased regions. (3) Global Thresholding: Effective in some cases but exhibited variability, making it less reliable for general application. (4) Edge Detection: Highly precise in boundary delineation but overly conservative, limiting its use for comprehensive disease severity estimation.

This ranking provides clarity on the most effective segmentation methods, with Otsu Thresholding emerging as the optimal choice for resource-poor areas due to its balance of sensitivity, specificity, and reliability. Adaptive thresholding could serve as a complementary method where sensitivity is paramount. For applications requiring precise lesion boundaries, edge detection remains valuable.

**3.4. Study limitations**

Despite the promising outcomes of this study, several limitations are evident. First, the dataset comprised only nine images, which may not sufficiently capture the variability in disease manifestations and image conditions present in real-world agricultural settings. Moreover, the fixed thresholding approaches can lead to either overestimation or underestimation of disease severity, and the absence of extensive ground-truth data restricts the ability to rigorously validate the segmentation results against expert assessments. Additionally, while the computational efficiency of these methods is beneficial for low-resource settings, their performance under diverse environmental conditions remains uncertain. Finally, the study did not explore hybrid approaches that might combine the strengths of different segmentation techniques, which could potentially enhance robustness and accuracy in more complex scenarios.

4. Conclusion

This study demonstrated that classical segmentation techniques can be effectively utilized for automated plant leaf disease severity estimation in resource-poor settings. Through the evaluation of four segmentation methods—global, adaptive, Otsu thresholding, and edge detection—across nine disease-affected leaf images, it was found that adaptive and Otsu thresholding consistently detected a high percentage of diseased regions. Notably, Otsu thresholding exhibited balanced performance with moderate diseased area estimates and near-unity healthy-to-diseased ratios, proving to be the most reliable and precise method among those tested. While adaptive thresholding achieved high sensitivity by often capturing more extensive diseased areas, it occasionally overestimated disease severity. Global thresholding displayed variable performance, and although edge detection excelled in accurately delineating lesion boundaries, it largely underestimated the overall disease extent. Consequently, Otsu thresholding is recommended as the optimal approach for low-resource environments, potentially complemented by adaptive thresholding when heightened sensitivity is required. These findings confirm that classical computer vision methods can offer a robust, efficient, and cost-effective solution for plant disease diagnostics in regions with limited technological resources.

References

1. Records A, Lapitan N, Bertram R. CHAPTER 1: Emerging Plant Diseases Threaten Global Food Security. In: Emerging Plant Diseases and Global Food Security [Internet]. The American Phytopathological Society; 2020 [cited 2025 Mar 29]. p. 3–11. (Epidemiology). Available from: https://apsjournals.apsnet.org/doi/10.1094/9780890546383.001

2. Hossain MdM, Sultana F, Mostafa M, Ferdus H, Rahman M, Rana JA, et al. Plant disease dynamics in a changing climate: impacts, molecular mechanisms, and climate-informed strategies for sustainable management. Discov Agric [Internet]. 2024 Dec 27 [cited 2025 Mar 29];2(1):132. Available from: https://doi.org/10.1007/s44279-024-00144-w

3. Sileshi GW, Gebeyehu S. Emerging infectious diseases threatening food security and economies in Africa. Glob Food Secur [Internet]. 2021 Mar 1 [cited 2025 Mar 29];28:100479. Available from: https://www.sciencedirect.com/science/article/pii/S2211912420301322

4. Ristaino JB, Anderson PK, Bebber DP, Brauman KA, Cunniffe NJ, Fedoroff NV, et al. The persistent threat of emerging plant disease pandemics to global food security. Proc Natl Acad Sci [Internet]. 2021 Jun 8 [cited 2025 Mar 29];118(23):e2022239118. Available from: https://www.pnas.org/doi/10.1073/pnas.2022239118

5. Strange RN, Scott PR. Plant disease: a threat to global food security. Annu Rev Phytopathol. 2005;43:83–116.

6. Mwangi RW, Mustafa M, Charles K, Wagara IW, Kappel N. Selected emerging and reemerging plant pathogens affecting the food basket: A threat to food security. J Agric Food Res [Internet]. 2023 Dec 1 [cited 2025 Mar 29];14:100827. Available from: https://www.sciencedirect.com/science/article/pii/S2666154323003344

7. Bock CH, Chiang KS, Del Ponte EM. Plant disease severity estimated visually: a century of research, best practices, and opportunities for improving methods and practices to maximize accuracy. Trop Plant Pathol [Internet]. 2022 Feb 1 [cited 2025 Mar 29];47(1):25–42. Available from: https://doi.org/10.1007/s40858-021-00439-z

8. Shi T, Liu Y, Zheng X, Hu K, Huang H, Liu H, et al. Recent advances in plant disease severity assessment using convolutional neural networks. Sci Rep [Internet]. 2023 Feb 9 [cited 2025 Mar 29];13(1):2336. Available from: https://www.nature.com/articles/s41598-023-29230-7

9. Faye D, Diop I, Mbaye N, Dione D, Diedhiou MM. Plant Disease Severity Assessment Based on Machine Learning and Deep Learning: A Survey. J Comput Commun [Internet]. 2023 Sep 19 [cited 2025 Mar 29];11(9):57–75. Available from: https://www.scirp.org/journal/paperinformation?paperid=127946

10. Ngugi HN, Ezugwu AE, Akinyelu AA, Abualigah L. Revolutionizing crop disease detection with computational deep learning: a comprehensive review. Environ Monit Assess [Internet]. 2024 Feb 24 [cited 2025 Mar 29];196(3):302. Available from: https://doi.org/10.1007/s10661-024-12454-z

11. Omara J, Talavera E, Otim D, Turcza D, Ofumbi E, Owomugisha G. A field-based recommender system for crop disease detection using machine learning. Front Artif Intell. 2023;6:1010804.

12. Hughes DP, Salathé M. An open access repository of images on plant health to enable the development of mobile disease diagnostics through machine learning and crowdsourcing. ArXiv [Internet]. 2015 Nov 25 [cited 2025 Mar 29]; Available from: https://www.semanticscholar.org/paper/An-open-access-repository-of-images-on-plant-health-Hughes-Salath%C3%A9/3e2da7c1c7dfc7960d1515b61f32fdc55359eea7

13. Liu S. Two Decades of Colorization and Decolorization for Images and Videos [Internet]. arXiv; 2022 [cited 2025 Mar 29]. Available from: http://arxiv.org/abs/2204.13322

14. You S, Barnes N, Walker J. Perceptually Consistent Color-to-Gray Image Conversion [Internet]. arXiv; 2016 [cited 2025 Mar 29]. Available from: http://arxiv.org/abs/1605.01843

15. Saravanan C. Color Image to Grayscale Image Conversion. 2010 [cited 2025 Mar 29];196–9. Available from: http://ieeexplore.ieee.org/document/5445596/

16. Al-amri SS, Kalyankar NV, D KS. Image Segmentation by Using Threshold Techniques [Internet]. arXiv; 2010 [cited 2025 Mar 29]. Available from: http://arxiv.org/abs/1005.4020

17. Munoz-Minjares J, Vite-Chavez O, Flores-Troncoso J, Cruz-Duarte JM. Alternative Thresholding Technique for Image Segmentation Based on Cuckoo Search and Generalized Gaussians. Mathematics [Internet]. 2021 Jan [cited 2025 Mar 29];9(18):2287. Available from: https://www.mdpi.com/2227-7390/9/18/2287

18. Xu M, Chen S, Gao X, Ye Q, Ke Y, Huo C, et al. Research on Fast Multi-Threshold Image Segmentation Technique Using Histogram Analysis. Electronics [Internet]. 2023 Jan [cited 2025 Mar 29];12(21):4446. Available from: https://www.mdpi.com/2079-9292/12/21/4446

19. Said KAM, Jambek AB. Analysis of Image Processing Using Morphological Erosion and Dilation. J Phys Conf Ser [Internet]. 2021 Oct [cited 2025 Mar 29];2071(1):012033. Available from: https://dx.doi.org/10.1088/1742-6596/2071/1/012033

20. Sun S, Huang Y, Inoue K, Hara K. Order Space-Based Morphology for Color Image Processing. J Imaging [Internet]. 2023 Jul 7 [cited 2025 Mar 29];9(7):139. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10381322/

21. Bazgir O, Zhang R, Dhruba SR, Rahman R, Ghosh S, Pal R. Representation of features as images with neighborhood dependencies for compatibility with convolutional neural networks. Nat Commun [Internet]. 2020 Sep 1 [cited 2025 Mar 29];11(1):4391. Available from: https://www.nature.com/articles/s41467-020-18197-y

22. Al Shoura T, Leung H, Balaji B. An Adaptive Kernels Layer for Deep Neural Networks Based on Spectral Analysis for Image Applications. Sensors [Internet]. 2023 Jan 30 [cited 2025 Mar 29];23(3):1527. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9921880/

23. Mandyartha EP, Anggraeny FT, Muttaqin F, Akbar FA. Global and Adaptive Thresholding Technique for White Blood Cell Image Segmentation. J Phys Conf Ser [Internet]. 2020 Jul [cited 2025 Mar 29];1569(2):022054. Available from: https://dx.doi.org/10.1088/1742-6596/1569/2/022054

24. Abera KA, Manahiloh KN, Motalleb Nejad M. The effectiveness of global thresholding techniques in segmenting two-phase porous media. Constr Build Mater [Internet]. 2017 Jul 1 [cited 2025 Mar 29];142:256–67. Available from: https://www.sciencedirect.com/science/article/pii/S0950061817304191

25. Gorgolewski K, Storkey AJ, Bastin ME, Pernet CR. Adaptive thresholding for reliable topological inference in single subject fMRI analysis. Front Hum Neurosci [Internet]. 2012 Aug 25 [cited 2025 Mar 29];6. Available from: https://www.frontiersin.org/journals/human-neuroscience/articles/10.3389/fnhum.2012.00245/full

26. Arques P, Aznar F, Pujol M, Rizo R. Real Time Image Segmentation Using an Adaptive Thresholding Approach. In: Marín R, Onaindía E, Bugarín A, Santos J, editors. Current Topics in Artificial Intelligence. Berlin, Heidelberg: Springer; 2006. p. 389–98.

27. Houssein EH, Mohamed GM, Ibrahim IA, Wazery YM. An efficient multilevel image thresholding method based on improved heap-based optimizer. Sci Rep [Internet]. 2023 Jun 5 [cited 2025 Mar 29];13(1):9094. Available from: https://www.nature.com/articles/s41598-023-36066-8

28. Singh S, Mittal N, Singh H, Oliva D. Improving the segmentation of digital images by using a modified Otsu’s between-class variance. Multimed Tools Appl [Internet]. 2023 Nov 1 [cited 2025 Mar 29];82(26):40701–43. Available from: https://doi.org/10.1007/s11042-023-15129-y

29. Ding J, Chang J, Han R, Yang L. CDSE-UNet: Enhancing COVID-19 CT Image Segmentation with Canny Edge Detection and Dual-Path SENet Feature Fusion [Internet]. arXiv; 2024 [cited 2025 Mar 29]. Available from: http://arxiv.org/abs/2403.01513

30. Saini S, Kasliwal B, Bhatia S. Comparative Study Of Image Edge Detection Algorithms [Internet]. arXiv; 2013 [cited 2025 Mar 29]. Available from: http://arxiv.org/abs/1311.4963

31. Bogdan V, Bonchiş C, Orhei C. Custom Extended Sobel Filters [Internet]. arXiv; 2019 [cited 2025 Mar 29]. Available from: http://arxiv.org/abs/1910.00138

32. Chang Q, Li X, Li Y, Miyazaki J. Multi-directional Sobel operator kernel on GPUs [Internet]. arXiv; 2023 [cited 2025 Mar 29]. Available from: http://arxiv.org/abs/2305.00515

33. Laske O, Ziegelmeier L. Image Triangulation Using the Sobel Operator for Vertex Selection. LIPIcs Vol 293 SoCG 2024 [Internet]. 2024 [cited 2025 Mar 29];293:91:1-91:7. Available from: http://arxiv.org/abs/2408.16112

34. Wang X. Laplacian operator-based edge detectors. IEEE Trans Pattern Anal Mach Intell. 2007 May;29(5):886–90.