International Journal of Scientific Research and Engineering Development-– Volume 7 Issue 4, July-Aug 2024
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

A New Perspective on Measuring and Analysing Damaged Leaf Areas Using Digital Image Processing Techniques

Pijush Kanti Kumar
(Asst. Prof. (IT), Government College of Engineering &
Textile Technology, Serampore, West Bengal-712201, INDIA
Email: pijush752000@yahoo.com)

----------**********----------

Abstract:

The detection of plant leaf diseases typically involves visual observation of patterns on the leaf surface. However, many diseases exhibit very subtle changes in these observable patterns. This study aims to utilize image processing techniques to identify diseases on plant leaves, with a particular focus on detecting Citrus canker. Citrus canker is a bacterial infection characterized by brown spots on leaves, often appearing water-soaked or oily, and surrounded by a yellow halo, visible on both leaf surfaces. The paper explores various methods for detecting Citrus canker, including Histogram Comparison, Colour Co-occurrence Matrix, and K-means Clustering. These methods successfully identified the presence of Citrus canker through histogram analysis, colour co-occurrence patterns, and leaf pattern recognition. The findings can be used to determine appropriate treatments in consultation with agricultural experts.

Keywords —Citrus canker, Digital image processing, K-means clustering, Leaf area calculation, Disease detection, Feature extraction, Colour co-occurrence matrix

----------**********----------

I. INTRODUCTION

The detection of diseases on plant leaves plays a crucial role in the overall growth environment of plants. These diseases are often easily identifiable by examining the infected areas of the crop. Typically, the leaf will visibly show the infected parts, making it easy to recognize the disease. Therefore, changes in the color of the crop are significant indicators of infection. A healthy crop exhibits a different color compared to one that is being affected by harmful pathogens, which cause the color to change automatically.

Crop diseases typically affect specific parts of the plant, leading to a reduction in productivity. Traditionally, plant diseases have been detected through the visual observation of experts. In this study, we propose the use of feature extraction techniques to analyze the affected parts of leaves. Specifically, we employ skew divergence of edge, color, and texture variance features, utilizing methods such as the Color Co-occurrence Matrix and K-means clustering. These techniques have been used to predict six types of diseases, and we have obtained performance evaluation results.

This research aims to reduce computational complexity by enhancing the automatic detection of crop diseases, which can cause significant damage in agricultural fields. Early detection allows for potential prevention and treatment of infected plants. Therefore, we need robust, cost-effective, and accurate methods to detect and classify crop diseases.

Initially, test images of various cotton leaves were captured. Image processing techniques were then applied to these images to extract useful features for further analysis. Several statistical techniques were used to classify the images based on the specific problems at the affected leaf spot area. Feature selection was employed to find the best match for the affected leaf feature results, leading to an optimal solution. In the classification phase, edge, colour, and texture feature variance values were stored in the image domain, and damaged spots were identified based on the affected regions of the leaf disease.

II. MATERIALS AND METHODS

A. Materials

Nikon Make 12.5 Megapixels Digital camera, PC, White Paper Sheet for background, MATLAB 7.4 version or above, Photographs of the leaves were arranged in number.

B. Methods

1. Graphical Method

Leaf, whose area to be measured was placed on the graph paper, having smallest grid size of 1 mm. Leaf is outlined with a pencil accurately and carefully on the graph paper. The total number of grids covered by outline edge of the leaf was calculated; consideration was, if edge outline occupied more than one half grids treated as one otherwise zero. The number of grid count corresponds to the actual area of leaf.

2. Image Processing Method

The technique based on MATLAB, is used for the image processing, is a semi automatic method to calculate leaf area and for more users this will be easy way to calculate leaf area. The code is written in MATLAB version 7.4 or above

This code will work in any higher version of the program.

Algorithm

1. Read the image.
2. Convert RGB to grayscale image.
3. Convert grayscale image to binary image.
4. Calculate the leaf area.

K Means

The Basic Idea

Say you are given a data set where each observed example has a set of features, but has **no**labels. Labels are an essential ingredient to a supervised algorithm like Support Vector Machines, which learns a hypothesis function to predict labels given features. So we can't run supervised learning. What can we do?

One of the most straightforward tasks we can perform on a data set without labels is to find groups of data in our dataset which are similar to one another -- what we call clusters.

K-Means finds the best centroids by alternating between (1) assigning data points to clusters based on the current centroids (2) choosing centroids (points which are the centre of a cluster) based on the current assignment of data points to clusters.

Visual Cortex:

K-Means is the first algorithm you must implement for the Visual Cortex assignment.

The Algorithm

In the clustering problem, we are given a training set ${x^{(1)}, ... , x^{(m)}}$, and want to group the data into a few cohesive "clusters." Here, we are given feature vectors for each data point $x^{(i)} \in \mathbb{R}^n$ as usual; but no labels $y^{(i)}$ (making this an unsupervised learning problem). Our goal is to predict k centroids **and** a label $c^{(i)}$ for each datapoint. The k-means

1. Initialize **cluster centroids** $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$ randomly.
2. Repeat until convergence: {

   For every i, set
   $$c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2.$$

   For each j, set
   $$\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}}.$$

   }

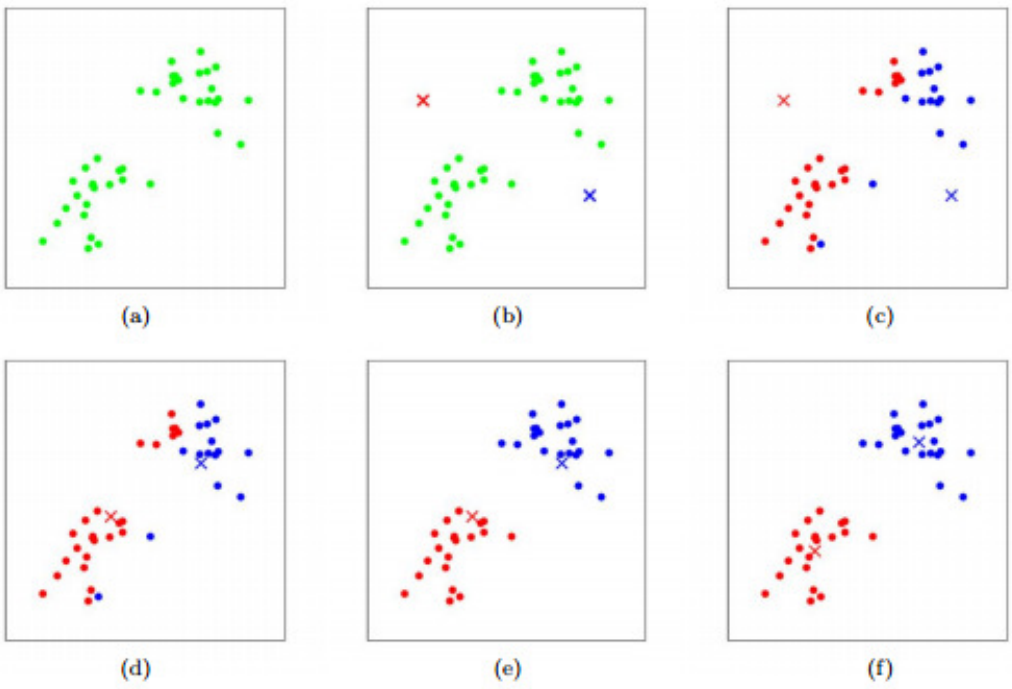


Figure 1: K-means algorithm. Training examples are shown as dots, and cluster centroids are shown as crosses. (a) Original dataset. (b) Random initial cluster centroids. (c-f) Illustration of running two iterations of k-means. In each iteration, we assign each training example to the closest cluster centroid (shown by "painting" the training examples the same color as the cluster centroid to which is assigned); then we move each cluster centroid to the mean of the points assigned to it. Images courtesy of Michael Jordan.

Euclidean Distance

The notation $\lVert x - y \lVert$ means euclidean distance between vectors x and y.

Implementation

Here is pseudo-python code which runs k-means on a dataset. It is a short algorithm made longer by verbose commenting.

```
# Function: K Means

# K-Means is an algorithm that takes in a dataset and a constant

# k and returns k centroids (which define clusters of data in the

# dataset which are similar to one another).

def kmeans(dataSet, k):

    # Initialize centroids randomly
```



```
numFeatures = dataSet.getNumFeatures()
    centroids = getRandomCentroids(numFeatures, k)
        # Initialize book keeping vars.
    iterations = 0
oldCentroids = None
        # Run the main k-means algorithm
    while not shouldStop(oldCentroids, centroids, iterations):
        # Save old centroids for convergence test. Book keeping.
oldCentroids = centroids
        iterations += 1
        # Assign labels to each datapoint based on centroids
        labels = getLabels(dataSet, centroids)
        # Assign centroids based on datapoint labels
        centroids = getCentroids(dataSet, labels, k)
    # We can get the labels too by calling getLabels(dataSet, centroids)
    return centroids
# Function: Should Stop
# Returns True or False if k-means is done. K-means terminates either
# because it has run a maximum number of iterations OR the centroids
# stop changing.
def shouldStop(oldCentroids, centroids, iterations):
    if iterations > MAX_ITERATIONS: return True
    return oldCentroids == centroids
# Function: Get Labels
# Returns a label for each piece of data in the dataset.
def getLabels(dataSet, centroids):
    # For each element in the dataset, chose the closest centroid.
    # Make that centroid the element's label.
# Function: Get Centroids
```



```
# Returns k random centroids, each of dimension n.

def getCentroids(dataSet, labels, k):

    # Each centroid is the geometric mean of the points that

    # have that centroid's label. Important: If a centroid is empty (no points have

    # that centroid's label) you should randomly re-initialize it.
```

Important note: You might be tempted to calculate the distance between two points manually, by looping over values. This will work, but it will lead to a slow k-means! And a slow k-means will mean that you have to wait longer to test and debug your solution.

Let's define three vectors:

```
x = np.array([1, 2, 3, 4, 5]

y = np.array([8, 8, 8, 8, 8])

z = np.ones((5, 9))
```

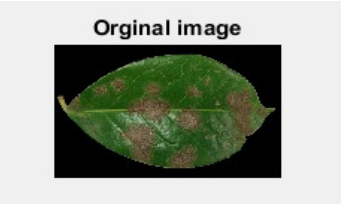
To calculate the distance between x and y we can use:

```
np.sqrt(sum((x - y) ** 2))
```

To calculate the distance between all the length 5 vectors in z and x we can use:

```
np.sqrt(((z-x)**2).sum(axis=0))
```

1) Image acquisition
The leaf whose infected area is to be measured is placed on the black background, without any light reflection. The camera is held horizontally to the plane of the leaf. The photograph distance is neither too close nor too far; it is adjusted such a way that the photograph is covering only background.



2) Reading the image
The image is saved in the computer as ‘b3.jpeg’. This image is read for further processing by MATLAB.

3) Segment the Image
In this step, original image is segmented into 2 types of images according to variation of colour. From these images images have been selected that identified affected regions of the leaf and unaffected regions of the leaf. It is done by K-means clustering method. The algorithm of this method is given below:

Step 1: Read Image

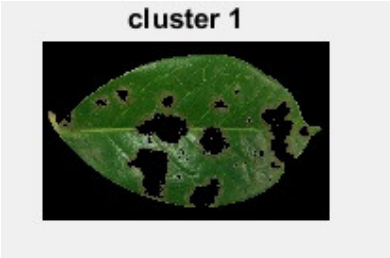
Step 2: Convert image from RGB colour space to LAB colourspace.

Step 3: Classify the colours in 'a*b*' space using K-means clustering

Step 4: Label every pixel in the image using the results from K-means

Step 5: Create images that segment the original image by colour.

After applying the above algorithm following three segmented images were obtained.



4) Convert the image (cluster-1) into Binary image.

After analysing the image before everything this picture must be transformed from RGB to Gray scale picture then the gray scale picture can be transformed into binary photo. on this photograph white coloured areas indicates unaffected a part of the leaf as shown in the following determine. It allows to calculate total range of pixels of unaffected areas of the leaf.

5) Convert the image (cluster-2) into Binary image.

After converting the image into binary image, following image is obtained where white coloured regions indicate affected regions of the leaf. It helps to calculate total number of pixels of affected regions of the leaf

.

6)Pixel Calculation

i) From cluster-1
WP (pixels of unaffected regions) = 195612
Where, WP denotes white pixels of cluster -1
ii) From Figure 12
WP1 (pixels of affected regions) =41246
Where, *WP1* means white pixels. Of cluster-2
Now, total pixels of the leaf area that is obtained is shown below:

TP=WP+WP1

TP=195612+41246

=236858

Percentage of

affected pixels can

be obtained by the

following way

ERROR=

$$\frac{WP1(pixelsofaffectedregions)}{TP(Totalpixelsoftheleafarea)}$$

×100

ERROR=

$\frac{41246}{236858}\times 100$

ERROR (%) = 17.4138

III. RESULTS AND DISCUSSION

To evaluate the performance of the new measurement system, leaves from various plots and different varieties of Citrus were selected. The leaf area was calculated using the square grid method, which served as the standard reference. By sampling data from various affected leaves, we can gain a clear understanding of the total damaged area in relation to a specific region. This proposed method is faster and more accurate than traditional methods.

IV. CONCLUSIONS

This work outlines the use of digital image processing techniques to calculate the area of Citrus leaves. The implemented algorithm has proven to be adequate for computing damaged leaf areas. While grid counting techniques offer high accuracy, they are time-consuming. In contrast, image processing methods are both highly precise and faster. Stored images of leaves can be processed at any time, providing a theoretical foundation for developing handheld leaf area meters to meet the demands of precision farming. The percentage error component is valuable for accurate area calculation and assessing disease severity, which is crucial for the appropriate application of pesticides and fertilizers.

V. REFERENCES

[1] [1] Mrunalini R. Badnakhe and Prashant R. Deshmukh..”Infected Leaf Analysis and Comparison by Otsu Threshold and k-Means Clustering”.2(3): 449-452. 2012.
[2] [2] Morlon Marcon , Kleber Mariano (et.al),”Estimation of total leaf area in perennial plants using image analysis”,
[3] R.Bras.Eng.Ambiental,2011,v.15,96-101.
[4] [3] ChaohulLu,Hui Ren (et.al) “Leaf area measurement based on image processing,”IEEE,2010,580-582.
[5] [4] Tian You-wen,Wang Xiao-juan,”Analysis of leaf parameters measurement of cucumber based on image processing, “World congress on software engineering,2009,34-37.
[6] [5] Yan Cheng Zhang, Han Ping Mao, Bo Hu, Ming Xili,“Features selection of Cotton disease leaves image based on fuzzy feature selection techniques” IEEE Proceedings of the 2007 International Conference on Wavelet Analysis and Pattern Recognition, Beijing,
[7] China, 2-4.2007.
[8] [6] Meunkaewjinda. A, P.Kumsawat, K.Attakitmongcol and A.Sirikaew.”Grape leaf disease Detection n from color imaginary using Hybrid intelligent system”, Proceedings of ECTI-CON. 2008
[9] [7] Gulhane.V. A & A. A. Gurjar, “Detection of Diseases on Cotton Leaves and Its Possible Diagnosis” (IJIP), 5 (5): 591-598. 2011
[10] [8] Ajay A. Gurjar and Viraj A. Gulhane. Disease Detection On Cotton Leaves by Eigenfeature Regularization and Extraction
[11] Technique”. IJECSCSE.1 (1): 1-4.2012.
[12] [9] Qinghai He, Benxue Ma, Duanyang Qu, Qiang Zhang,Xinmin Hou, and Jing Zhao.” Cotton Pests and Diseases Detection based on
Image Processing”. Telkomnika.11 (6): 3445- 3450.June 2013.

