Mehran University Research Journal of Engineering and Technology

https://doi.org/10.22581/muet1982.2302.03

2023, 42(2) 19-29

Real time vision-based implementation of plant disease identification system on FPGA

Junaid Ahmed *, Syed Azhar Ali Zaidi, Sumair Aziz, Aamir Rashid, Shafiq Haider

Department of Electronics Engineering, University of Engineering and Technology, Taxila Pakistan

* Corresponding author: Junaid Ahmed, Email: ahmedjunaid339@gmail.com

Received: 26 January 2021, Accepted: 24 March 2023, Published: 01 April 2023

K E Y W O R D S

Artificial Intelligence

K-Nearest Neighbors

Field Programmable Gate Array

A B S T R A C T

Plant diseases have turned into a dilemma as it can cause significant reduction in both quality and quantity of agricultural products. To overcome that loss, we implemented a computer vision based real time system that can identify the type of plant diseases. Computer vision-based applications are computationally intensive and time consuming, so FPGA-based implementation is proposed to have a real time identification of plant diseases. In this paper an image processing algorithm is proposed for identifying two types of disease in Potato leaves. The proposed algorithm works well on images taken under different luminance conditions. The hardware/software-based implementation of the proposed algorithm is done on Xilinx ZYNQ SoC FPGA. Results show that our proposed algorithm achieves an accuracy of up to 90%, whereas the hardware implementation takes 0.095 seconds achieving a performance gain of 76.8 times as compared to the software implementation.

1. Introduction

Agriculture sector plays a big role in this world, and it has a huge impact on the economy of any country. Over the past few decades, a lot of research has been done on the automatic detection of plant diseases to enhance the quality of production, save fields from disaster and reduce manpower with the help of robotics, artificial intelligence, image processing and many other techniques. Plant diseases vary from plant to plant and every disease has its own symptoms. It may affect stems, and leaves or damage roots. So, every disease should be treated differently. Image processing techniques can be applied for an early diagnosis of plant disease through the physical appearances of leaves, stems etc. Early diagnosis of disease may protect farms from disaster. The naked eye approach for disease identification is not suitable for huge farms or land areas. So, there is a great need to deploy automatic surveillance and real time identification and classification techniques in agriculture.

In general, computer vision based methods proposed for identifying visual symptoms from plant leaves use a combination of various preprocessing, segmentation, feature extraction and classification techniques [1] and [2]. These techniques tend to be computationally intensive and deteriorate the overall system performance in real time environments.

Most of the previous research has only focused towards designing efficient software algorithms without considering the efficient hardware implementation of these algorithms. High computational requirements of these algorithms force designers to implement highly complicated hardware to achieve performance. These approaches have various shortcomings, such as high deployment costs and long design time. Due to the availability of large number of computational and memory resources, field-programmable gate arrays (FPGAs) are an

excellent choice for implementing highly parallel hardware and hardware/software programmable system-on-chip (SoC) design.

In this work, a real time vision-based system image processing algorithm is developed and implemented in hardware that can identify and classify plant diseases using visual symptoms. Our proposed algorithm first performs image preprocessing and then image enhancement, followed by clustering-based segmentation. After that, feature extraction is performed using gray level co-occurrence matrix (GLCM), and the K-nearest neighbors (KNN) algorithm is deployed for classifying the plant leaf. The main contributions of this work are as follows:

- A computationally efficient and accurate implementation of an image processing algorithm for the identification and classification of plant diseases.
- Usage of a more realistic dataset containing images under various luminance conditions and testing the proposed algorithm with this dataset achieving accuracy of 90%.
- Implementation of the complete algorithm on a programmable FPGA-based SoC with various optimizations at the algorithmic level resulting in an area-efficient hardware implementation. The proposed implementation achieves a gain in speed of 76.8 and 23.26 times as compared to the software implementation on general-purpose computers and ARM cortex-A9 microprocessors, respectively.

The rest of the paper is organized as follows. In section 2, a comprehensive literature review is provided. Section 3 describes the proposed algorithm. In section 4, detailed hardware implementation is given. Section 5 presents experimental results and discussions, and section 6 concludes our work with insights into future directions.

2. Literature Review

Many researchers have focused on the detection and classification of different types of plant diseases. Because it is hard to detect infected portions in real scenario images. Sunlight and environmental effects result in many problems in image processing algorithms. Jobin Francis et al. worked on the pepper plant and analyzed two diseases, namely, berry spot and quick wilt [1]. First, the RGB image was transformed to HSV color space and masks were generated to extract the diseased area. Gray level co-occurrence matrix (GLCM) was applied for feature extraction, and Artificial Neural Network (ANN) with back propagation was used to classify images into the diseased and healthy categories. John William Orillo et al. worked on three diseases of rice plants: bacterial leaf blight, brown spot and rice blast. Features utilized for classification were mean values of RGB and HSV, the standard deviation of RGB and the fraction of leaf covered by disease. ANN back propagation algorithm was applied for classifying images for three different diseases [3].

In [4], authors proposed a method for Brinjal disease detection using K-means, SGDM, GLCM and texture features. Mean intensity, area, perimeter, centroid, and diameter were calculated for every leaf and ANN was employed for classification. Sanjeev S Sannakki et al. proposed a method for identifying grape leaf diseases through feed-forward and Back Propagation Neural Network (BPNN) as a classifier, whereas K-means clustering and GLCM is used for feature extraction. Two types of grape leaf diseases, downy mildew and powdery mildew, were considered in the experiments [5].

In [6], a mobile vision system for plant leaf disease detection is proposed. They used a mobile phone to segment the images and extract the infected part of the leaf in each image. For image segmentation, authors used K-means clustering and texture features for disease identification. Some part of the algorithm was executed by mobile phone and the rest was transmitted to a central server for further processing. The central server then sends results back to the mobile phone to inform the user what type of disease is detected. The main advantage of this method was a reduction in power consumption at the user end. In [7], the authors performed research on detecting two diseases of rice plants. The greenness of plant images was measured by vegetation indices and segmentation was done to extract the diseased area. GLCM, Otsu's method [8] and texture features were used for feature extraction and 15 different classifiers available in the WEKA tool are used for disease classification. A comparison was made between vegetation indices and gray images. A new approach was developed, and the algorithm results were 84% accurate.

Sachin D. Khirade and A. B. Patil used Spatial Gray-level Dependence Matrices (SGDM), GLCM, Otsu and k-means for feature extraction and BPNN for classification in [9]. Image transformation into grayscale and histogram equalization techniques was applied to enhance the images. The authors showed that by using ANN and Support Vector Machine (SVM), diseases can be detected efficiently. In [10], plant disease area extraction from leaves was performed through fuzzy entropy and fuzzy membership functions. The probabilistic neural network was applied for classification. The accuracy

of paddy disease identification is 91.46%. They developed mobile phone applications to identify diseases. Pooja Pawar et al. proposed a classification framework for cucumber disease. First-order statistical moments, mean, standard deviation, kurtosis, skewness and second-order statistical moments with GLCM were used for feature extraction, whereas, an ANN-based algorithm was utilized for classification. The classification accuracy achieved was 80.45% [11]. Jitesh P. Shah et al. performed a comprehensive survey on the detection and classification of rice plants. Their work discussed OTSU's method, texture, shape, and color for feature extraction and SVM, ANN and NN for classification [12].

Jagadeesh D. Pujari et al. used GLCM and Gray Level Run Length Matrix (GLRLM) for feature extraction and Probabilistic Neural Network (PNN) for classification [13]. Sandika Biswas et al. used diagonal variance fuzzy c-means contrast, uniformity, maximum probability, homogeneity, inverse difference and difference variance for feature extraction and minimum distance classifier (MDC) and KNN for classification [14]. In [15], different deep learning models are explored for the automated recognition of late and early blight diseases based on the optical images of potato leaves. They achieved an accuracy of 97.89% with VGG 16 model. Trong Yen Lee et al. [16], proposed a new CNN model for classification, where in preprocessing, the Gaussian filter is applied to reduce noise and then normalization is performed, and the green part is extracted. Their proposed model has an accuracy of 99.53%. Benjamín Luna-Benoso et al. [17], performed a negative operator and median filter on a grayscale computed image, and then thresholding is applied using the Otsu method. After that, nine features were computed along with four color moments. Three classifiers multi-layer perceptron (MLP), KNN and SVM are used for analysis and classification. They achieved an accuracy of 97.39% in classifying late blight, tomato mosaic virus and Septoria leaf spot diseases. They also achieved an accuracy of 86.45% between healthy and sick tomato diseases.

In [18], the authors proposed an algorithm that transforms the images into three color spaces, which are processed simultaneously. The algorithm executes in a series of intermediate steps, including histogram equalization, morphological operations, contrast stretching, feature vector construction and identification of salient regions. Hardware implementation is also done using Network-on-Chip (NoC) technique. The proposed technique achieves an accuracy between 63% and 87%, whereas, the FPGA implementation takes 0.169 seconds for computation. The architecture has been synthesized for 90 nm process. Chaojun Hou et al. [19], performed recognition of early blight and late blight diseases on potato leaves based on graph cut segmentation. First, filtering and color conversion is performed in the preprocessing stage. After that, super pixels are generated by the adaptive simple linear iterative clustering (SLIC) algorithm and then seeds of foreground and background are collected using thresholding methods. Also, graph cut based algorithm is used in the segmentation process. KNN, SVM, ANN and random forest (RF) are used for classification. They achieved maximum accuracy of 97.4% using SVM.

Most of the work in literature has used an ideal dataset i.e., images are taken under specific conditions of luminance and with simple background while some of the authors also used already preprocessed dataset. In the proposed algorithm, a more realistic dataset is used containing images taken under different luminance conditions and not already preprocessed. In our implementation a novel algorithm is proposed which is intelligent enough to control noise factor up to some extent. Most of the work has also done only the software implementation of their algorithm and therefore, the computational performance and accuracy on real time hardware was not evaluated. In this work, the hardware implementation of the complete algorithm is done, and the computational performance and accuracy is compared with the state of the artwork.

3. Proposed Work

We targeted the potato leaves for disease identification and classification. Images of Potato leaves are collected from internet resource. Some images are taken from [20]. Around 450 images have been used in this project. Each image is scaled to 200x200 pixels size in RGB format. A few samples of collected images are shown in Fig. 2. Two types of potato leaf diseases are considered, namely, late blight and early blight. Research work is carried out in two major stages, i.e. algorithm development and hardware implementation. In the proposed algorithm, first RGB to CIE Lab conversion is performed in the preprocessing stage, then K-means clustering and thresholding is used to segment out the diseased part. After that, GLCM based features are calculated in order to perform classification using KNN algorithm. Overall flow of algorithm design is given in (Fig. 1). In the second step FPGA based hardware implementation is done on Zynq-7000 SoC which will be discussed in section 4.

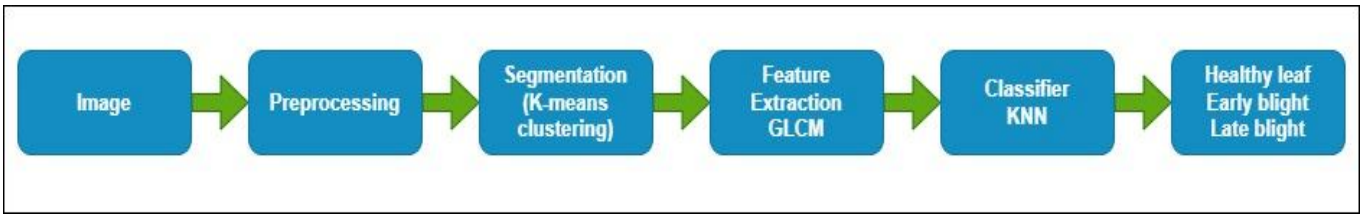

Fig. 1. Block diagram of the proposed method

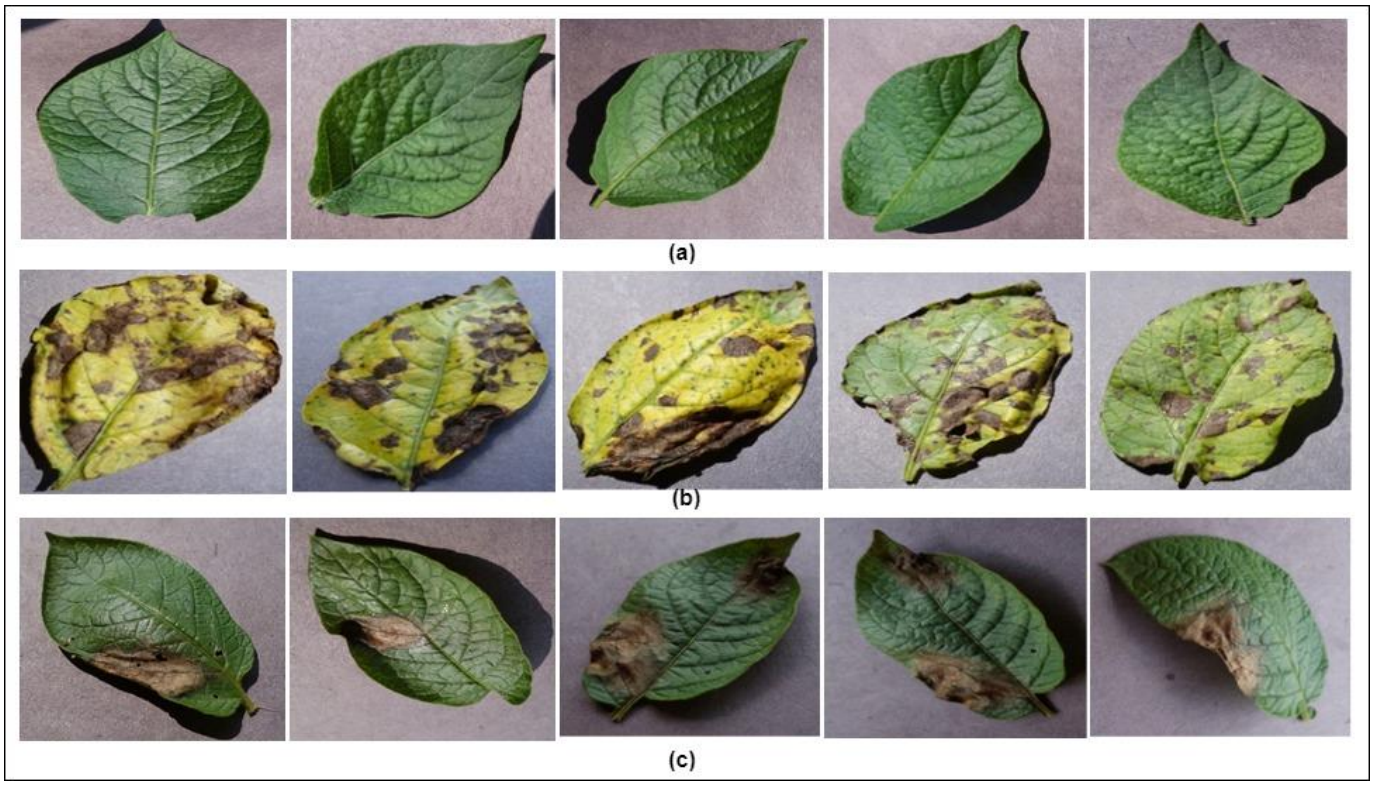

Fig. 2. Sample Images of Potato leaves. (a) Healthy leaves samples (b) Early blight samples (c) Late blight samples

3.1 Image Preprocessing

In the first step, RGB images are converted into CIE Lab color space. International Commission on Illumination defined CIE Lab color space for use as a reference color model because it is very close to the color model of human eye. This conversion is used to eliminate the effect of the luminance. For the RBG to Lab conversion, first RGB pixels are converted into XYZ then in the second step XYZ representation is converted into CIE Lab.

3.1.1 RGB to XYZ conversion

Original RGB pixels are first gamma expanded by using equation (1) and then converted into the tri-stimulus values *(XYZ)* by using equation (2) defined in [21]. *XYZ* coordinate system defines real-world light, where, *'X'* is a mixture of three CIE RGB positive values, *'Y'* is representing luminance and *'Z'* is quasi-equal to blue. 'α' is constant with value 0.055.

$$\mathrm{C}(linear) = \begin{cases} \frac{C(RGB)}{12.92}, & C(RGB) < 0.04045 \\ \left(\frac{C(RGB)+\alpha}{1+\alpha}\right)^{2.4}, & x \geq 0 \end{cases} \quad (1)$$

RGB to *XYZ* transformation matrix based on color space and standard illuminant *D65* white point referenced is given below:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3575 & 0.1804 \\ 0.2126 & 0.7151 & 0.0721 \\ 0.0193 & 0.1191 & 0.9503 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (2)$$

Algorithm 1 RGB → XYZ

1. for i ← 1, 40000
2. if red_i> 0.04045
3. $R_i = ((red_i, 0.055) / (1 + 0.055))^{2.4}$
4. else
5. $R_i = red_i / 12.92$
6. end if
7. if $green_i$> 0.04045
8. $G_i = ((green_i, 0.055) / (1 + 0.055))^{2.4}$
9. else
10. $G_i = green_i / 12.92$
11. end if
12. if $blue_i$> 0.04045
13. $B_i = ((blue_i, 0.055) / (1 + 0.055))^{2.4}$
14. else
15. $B_i = blue_i / 12.92$
16. end if
17. $X_i = (R_i * 0.4124) + (G_i * 0.3576) + (B_i * 0.1805)$
18. $Y_i = (R_i * 0.2126) + (G_i * 0.7152) + (B_i * 0.0722)$
19. $Z_i = (R_i * 0.0193) + (G_i * 0.1192) + (B_i * 0.9505)$
20. end for

3.1.2 XYZ to RGB conversion

In second step, the *XYZ* variables are converted into *CIE-Lab* by using equations 3, 4, 5 and 6 [21]:

$$L = 116 * f\left(\frac{Y}{Yn}\right) - 16 \quad (3)$$

$$a = 500 * \left(f\left(\frac{X}{Xn}\right) - f\left(\frac{Y}{Yn}\right)\right) \quad (4)$$

$$b = 200 * \left(f\left(\frac{Y}{Yn}\right) - f\left(\frac{Z}{Zn}\right)\right) \quad (5)$$

Algorithm 2 XYZ → CIE-Lab

1. for i ← 1, 40000
2. $X\_norm_i = X_i / 95.047$
3. $Y\_norm_i = Y_i / 100.000$
4. $Z\_norm_i = Z_i / 108.883$
5. if X_i_div_Xn > 0.008856
6. X_temp_i =3X_normi
7. else
8. $X\_temp_i = (((7.787 * X\_norm_i) + 16) / 116)$
9. end if
10. if Y_i_div_Yn > 0.008856
11. Y_temp_i =3Y_normi
12. else
13. $Y\_temp_i = (((7.787 * Y\_norm_i) + 16) / 116)$
14. end if
15. if Z_i_div_Zn > 0.008856
16. Z_temp_i =3Z_normi
17. else
18. $Z\_temp_i = (((7.787 * Z\_norm_i) + 16) / 116)$
19. end if
20.
21. $CIE\_L_i = (116 * Y\_temp_i) - 16$
22. $CIE\_a_i = 500 * (X\_temp_i - Y\_temp_i)$
23. $CIE\_b_i = 200 * (Y\_temp_i - Z\_temp_i)$
24. end for

The function *f(t)* is given below:

$$f(t) = \begin{cases} t^{1/3} & , t > 0.008856 \\ 7.787 * t + \frac{16}{116} & , t \leq 0.008856 \end{cases} \quad (6)$$

Where, '*L*' is correlated to Lightness, '*a*' and '*b*' are the opponent color axis and '*Xn*', '*Yn*' and '*Zn*' are the *X, Y and Z* tri-stimulus values of the specified white object, respectively. Values under *D65* are given as follows: *Xn*=95.047, *Yn*=100.000 and *Zn*=108.883.

Algorithm 3 K-means Clustering

1. M1 = 0
2. M2 = 0
3. M3 = 5
4. M4 = 5
5. for j ← 1, 5
6. sum1 = 0
7. sum2 = 0
8. sum3 = 0
9. sum4 = 0
10. for i← 1, 40000
11. $D1 = (CIE\_a_i - M1)^2 + (CIE\_b_i - M2)^2$
12. $D2 = (CIE\_a_i - M3)^2 + (CIE\_b_i - M4)^2$
13. if D1 < D2
14. $M1\_new = CIE\_a_i$
15. $M2\_new = CIE\_b_i$
16. M3_new = 0
17. M4_new = 0
18. $cluster\_id_i$ = 0
19. total = total + 1
20. else
21. M1_new = 0
22. M2_new = 0
23. $M3\_new = CIE\_a_i$
24. $M4\_new = CIE\_b_i$
25. $cluster\_id_i$ = 1
26. end if
27. sum1 = M1_new + sum1
28. sum2 = M2_new + sum2
29. sum3 = M3_new + sum3
30. sum4 = M4_new + sum4
31. end for
32. M1 = sum1 / total
33. M2 = sum2 / total
34. M3 = sum3 / (40000 - total)
35. M4 = sum4 / (40000 - total)
36. total = 0
37. sum1 = 0
38. sum2 = 0
39. sum3 = 0
40. sum4 = 0
41. end for

3.2 Image Segmentation

Three dimensions named 'L', 'a' and 'b' were created in preprocessing step, out of which two are used for further processing named 'a' and 'b'. Dimension 'L' was not used to reduce the problem of luminance in image processing. K-means algorithm is used for the clustering of the data into two segments. K-means is applied on two dimensions, m1 and m2, simultaneously. So, four cluster means are used, M1 and M2 for first dimension and M3 and M4 for second dimension. The steps in K-means clustering algorithm are as follows.

1. First assign random values to cluster means *M1*, *M2*, *M3* and *M4*.

2. Compute the Euclidean distance *D1* between *M1* and each pixel of *m1*, *M2* with each pixel of *m2*. Similarly, distance *D2* is calculated between *M3* with each pixel of *m1* and *M4* with each pixel of *m2*.

$$D1 = \sqrt{((m1 - M1)^2 + (m2 - M2)^2)} \quad (7)$$

$$D2 = \sqrt{((m1 - M3)^2 + (m2 - M4)^2)} \quad (8)$$

3. Now based on minimum distance pixel is assigned to either mean *M1* and *M2* or mean *M3* and *M4*.

4. New means are then calculated from assigned pixels.

$$Mi = \frac{\sum AssignedPixels}{TotalPixels} \quad (9)$$

After that the steps 2, 3 and 4 are repeated over and over until each sample is near to the mean of each portion. K-means clustering divides the image in two segments, one containing segment of green pixels and other containing background with disease information in case of diseased leaf.

Segment containing disease is automatically selected by counting the number of green pixels in each segment. Segment containing a smaller number of green pixels is the diseased segment and it is selected for image background separation. Background is segmented out by comparing 'b' component in each pixel with a constant value of '6'. Pixels whose 'b' component pixel value is greater than '6' are selected to generate a final binary mask. The binary image is then multiplied with original RGB image to extract out the original RGB diseased part. (Fig. 3 (c, d)) shows two segments that are generated after the application of k-means clustering algorithm and thresholding. Clearly it shows that one segment containing green pixels has been separated from region of interest. Segment in (Fig. 3 (e)) shows the region of interest containing diseased part in case of diseased image or nothing in case of healthy image. After extracting the region of interest, KNN classifier is used for the classification of healthy and diseased images.

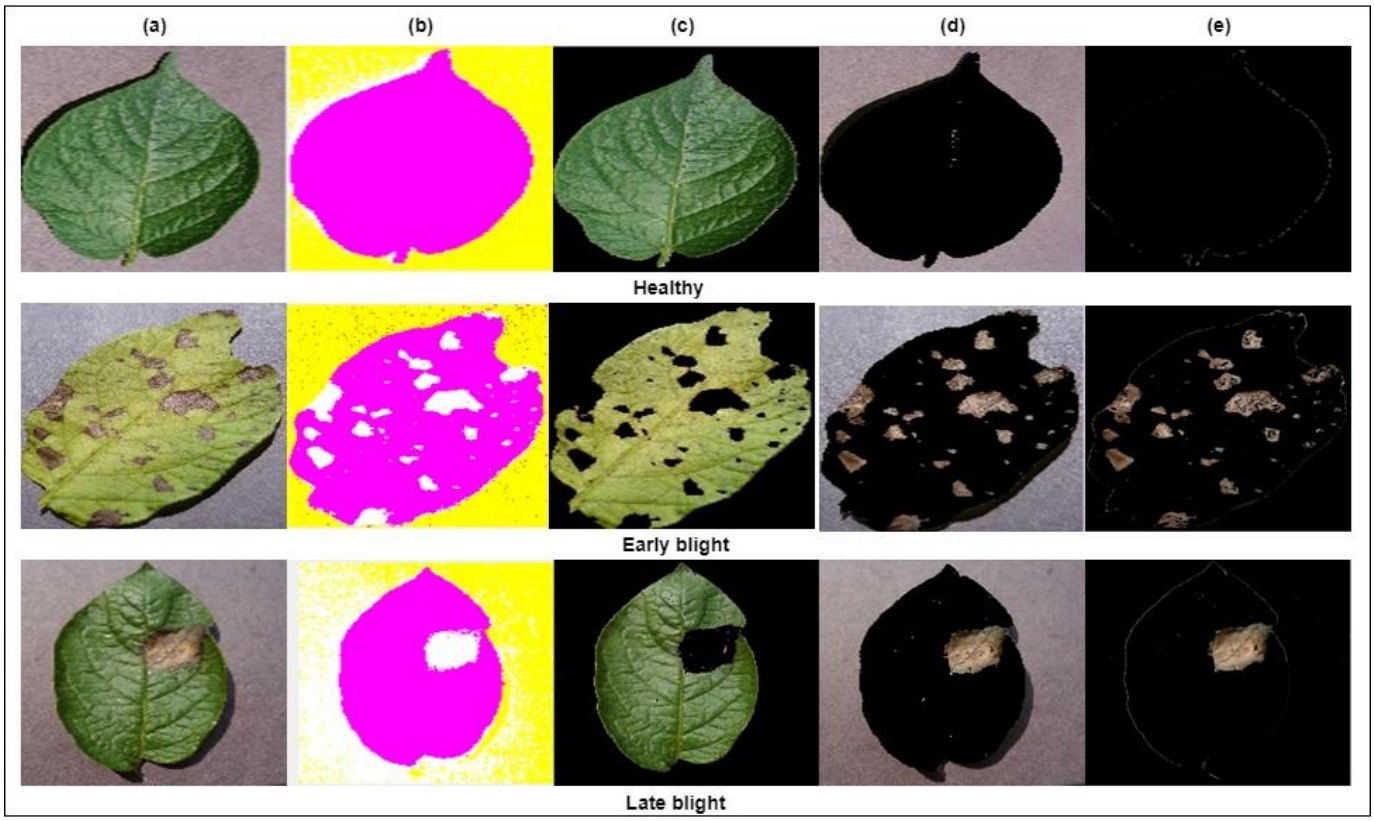

Fig. 3. (a) Original RBG Image of healthy, Early blight and Late blight (b) RGB to CIE-Lab Conversion (c) k-means first Segment (d) K-means Second Segment (e) Background separation

3.3 Feature Extraction

In general, word texture refers to the surface of an object. It refers to characteristics and appearance of an object given by the shape, density, arrangement, size, proportion of its elementary parts. A basic stage to collect such features through texture analysis process is called as texture feature extraction. Texture feature extraction is the major step in extracting information from the images. There are many methods that are used in measuring the texture base feature extraction. In this work, a statistical approach is used. GLCM is used for measuring texture information and features

calculations. The segmented diseased part is converted from RGB to gray scale and then an 8x8 GLCM is calculated.

3.3.1 Energy

This feature measures the non-uniformity or disorder in the image. It measures the rate of change in the color/ brightness/ magnitude in pixel-pair repetition. It is also known as the angular second moment. The Energy feature is calculated as follows [5]. Where *'P'* represents pixel value and *'i'* and *'j'* are the row and column indexes of the pixel *'P'* in equation 10, 11 and 12.

$$\text{Energy} = \sum_i \sum_j P_{ij}^2 \quad (10)$$

3.3.2 Contrast

It is difference of the highest and the lowest set of the contiguous set of pixels. It measures the variations in the images. Contrast measures the spatial frequency of an image. Formula for calculating the contrast is given below [5]:

$$\text{Contrast} = \sum_i \sum_j (i-j)^2 P_{ij} \quad (11)$$

3.3.3 Autocorrelation

It measures the repeating pattern in the image. It measures the uniqueness between the different samples of the dataset. Formula is given below [5]:

$$\text{Autocorrelation} = \sum_{i,j} (ij) * P_{ij} \quad (12)$$

3.4 Classification

After feature extraction the next main step is to classify the data. The classifier used in this step is K-nearest neighbors (KNN), which is a supervised learning algorithm. KNN is based on the feature similarity of the samples. It classifies the data on the bases of the K-nearest neighbor i.e., the samples having the minimum distance from the cluster group.

Table 1

Profiling results on ARM Cortex-A9 processor

Algorithm tasks	Time (s)
RGB to CIE-Lab conversion	0.23
K-means	1.68
Feature extraction	0.2
Classification	0.1
Total time	2.21

4. Hardware Implementation

We have targeted the Xilinx ZYNQ SoC platform for the implementation of the proposed algorithm for real time plant disease identification. The targeted device is Zed board evaluation kit containing Xilinx ZYNQ XZ7020 programmable SoC device. Xilinx XZ7020 contains a dual core ARM Cortex-A9 hard core processor (processing system (PS)) and the FPGA fabric (programmable logic (PL)) in the same chip. The steps performed for the overall algorithm implementation on ZYNQ are shown in Fig. 4.

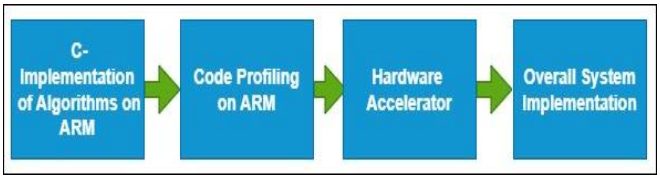

Fig. 4. Hardware implementation flow

Algorithm first implemented in MATLAB is then ported on ARM processor using C++ language. After "C++" implementation, the algorithm is executed on the processor and the code profiling is done. The execution time of different parts of algorithm is given in Table 1. For this purpose, the ZYNQ internal timer interrupts are used.

To improve performance, all modules are implemented on FPGA fabric. The hardware implementation is distributed in the following blocks: CIE Lab Converter, K-means Clustering, Feature Extraction, KNN, VGA Data Logic, VGA Display Controller, and several BRAM blocks. Overall architecture is given in Fig. 5.

In this implementation the *ZYNQ PS* is used to receive images over the UART protocol from the host computer and sends it to *ZYNQ PL* and controls the flow of *ZYNQ PL* using *PS to PL Controller IP*. It also receives calculated features and classification results from *PL* and sends these results and internal SCU timer information to the host system over the UART protocol. *ZYNQ PS* is connected to *PS to PL Controller IP* via *M_AXI_GP0 AXI* port.

The *PS to PL Controller IP* controls the flow of execution of different blocks implemented in *PL*. It has a memory mapped register file through which it communicates data between *PS* and *PL*. It gets commands from *PS* and communicates to *IP* blocks in the *PL*. It sends RGB data using a 1-bit *wr_en* signal and three 8-bit *r, g, b* signals to *CIE Lab Converter*. It also sends a *start_kmeans* signal to *k-means Clustering IP* to start the operation. During *K-means Clustering* and *Feature Execution, ZYNQ PS* waits for *done_feature* flag. When this flag is asserted, the *PS to PL Controller IP* updates the corresponding memory mapped registers with energy, contrast and autocorrelation features.

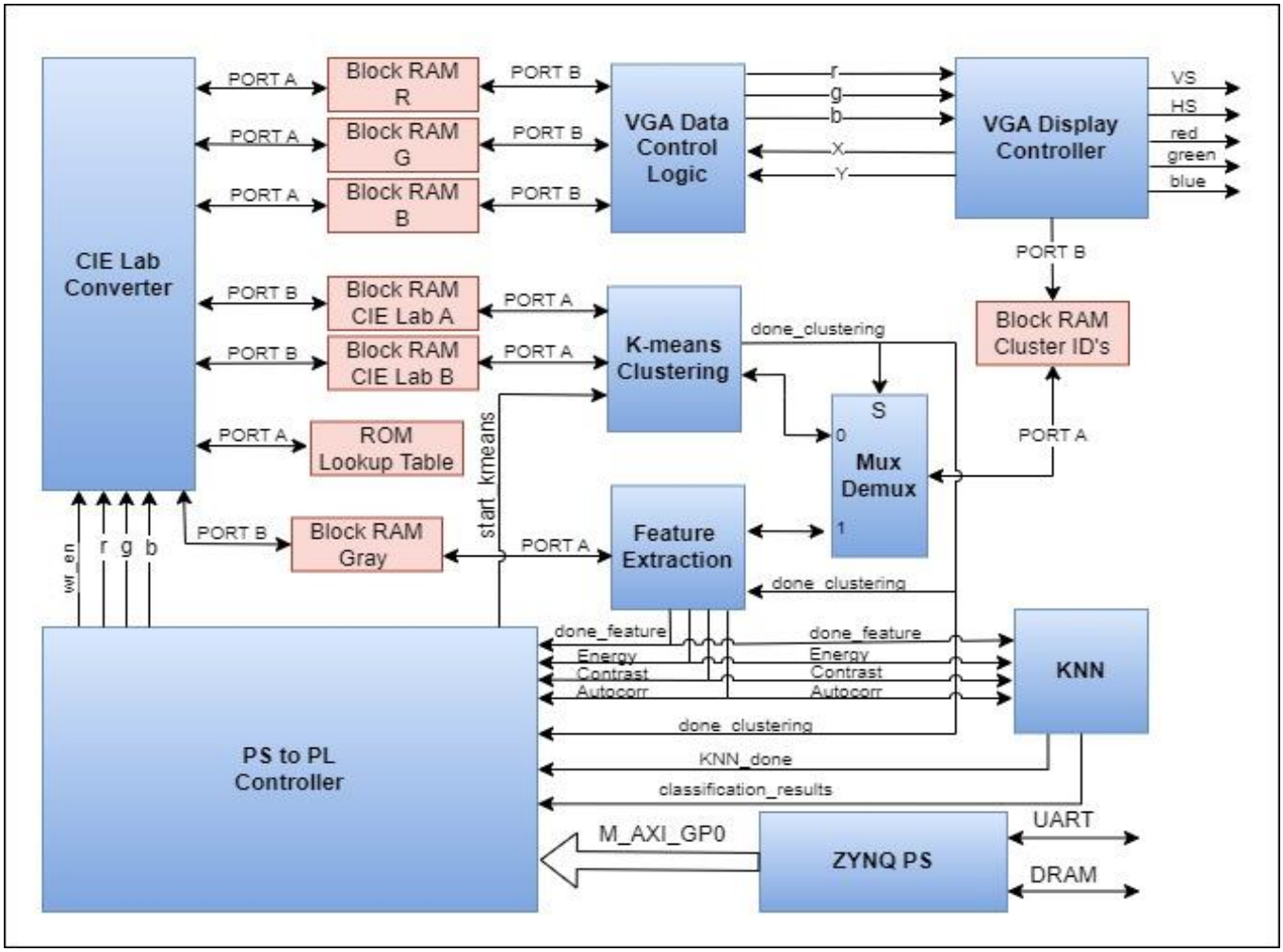

Fig. 5. Hardware architecture of proposed algorithm on FPGA

CIE Lab Converter IP block is used to convert RGB data to CIE Lab A and B components. It has a *wr_en* and three 8-bit *r, g, b* signals, used to receive image from *PS to PL Controller IP*. Also, it has seven BRAM ports. Three ports are connected to *Block RAM R, G and B* to store R, G and B data respectively. Two ports are connected *to Block RAM CIE Lab A and CIE Lab B* to store calculated CIE Lab A and B data, respectively. It also converts RGB data into Gray and stores it in *Block RAM Gray IP* Block. The last port is connected to *ROM Lookup Table* module which has pre-calculated results of Eq. 1, used to accelerate the conversion process of RGB to CIE Lab using lookup table. After that, equations 2, 3, 4, 5 and 6 are implemented using Xilinx Floating-Point IP v7.1. In Eq. 6, if the input *'t'* is greater than *0.008856*, then the cubic root is calculated by Taylor series expansion. Taylor series approximation is performed at two points: a=0.2 and a=1. If *'t'* is greater than 0.3 then output is calculated using a=1, otherwise, output will be calculated using a= 0.2. Experimental results show that Taylor series approximation at these two points resemble very closely with the MATLAB cubic root function. This block is also synchronized with UART protocol, so that when a pixel is received, its RGB to CIE Lab conversion is performed before the arrival of next pixel and hence this results in performance gain.

When all the pixels are received, *ZYNQ PS* asserts the *start_kmeans* signal for activating the *K-means Clustering IP* block. This block takes *CIE a* and *b* data from *Block RAM CIE a* and *Block RAM CIE b* and writes calculated cluster ID's in *Block RAM Cluster IP*. Equation 7, 8 and 9 are implemented using same Xilinx Floating-Point IP v7.1. When the clustering is done, *K-means Clustering Block* asserts *done_clustering* signal to start the execution of Feature Extraction block. *Feature Extraction IP* block reads cluster ID's information from *BLOCK RAM Cluster ID's* and gray scale image data from *Block RAM Gray* to compute GLCM for feature extraction. After that, Eq. 10, 11 and 12 are implemented to extract Energy, Contrast and Autocorrelation features. Once computation is done, a *done_feature* flag is set along with calculated values of energy, contrast and autocorrelation. In the end, *KNN IP* block gets

executed and compares trained data with the input features and classify them by selecting minimum Euclidean distance. It also sends classification results to *PS to PL Controller IP* block. After that, *ZYNQ PS* read back features and classification results using AXI Lite protocol and sends results to the host via UART. It also sends SCU timing information to the host, which is used to measure hardware execution performance.

5. Experimental Results

Around 450 images have been used in this work and the training-to-testing ratio is set to 80/20%. The accuracy results of the proposed algorithm are discussed with the help of a confusion matrix, as given in Fig. 6. It shows that in the case of healthy class, accuracy is 86.66%, and in the case of Early Blight and Late Blight classes, accuracies are 100% and 83.33%, respectively. The accuracy of each class is calculated as the number of true predictions divided by the total number of predictions. The overall accuracy of algorithm is 90% calculated using Eq. 13, where EB_{count}, LB_{count} and H_{count} are the true predictions for early blight, late blight and healthy class, respectively.

$$Accuracy_{Tot} = \frac{EB_{count}+LB_{count}+H_{count}}{90} * 100 \quad (13)$$

The main reason behind higher Early Blight accuracy is the distribution of diseased patterns over the surface and the best segmentation result.

Software profiling results are given in Table 1. It is found that almost all tasks have some bottleneck for real-time performance. Therefore, it is decided to implement all the software components into hardware blocks. The execution time of the algorithm after hardware implementation is 0.095 sec. So, hardware implementation has reduced execution time up to 2.11 seconds, and this result in a performance gain of 76.8 times. Performance results are also compared with Talha Akram et al. [18]. Their hardware implementation takes up to 0.169 seconds, while the proposed hardware implementation takes 0.095 seconds, as shown in Table 2. It shows a performance gain of 43.787% as compared to similar work.

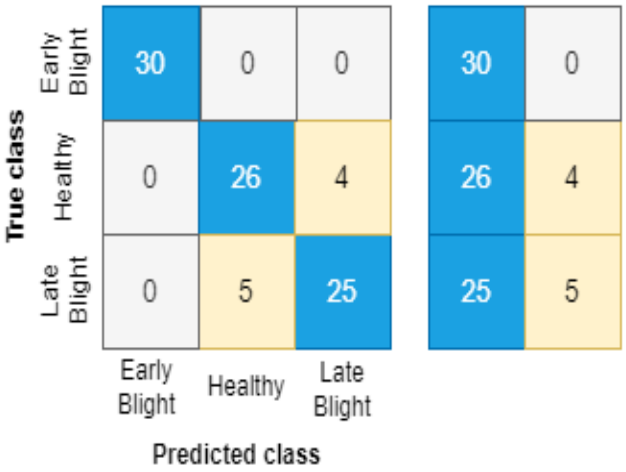

Fig. 6. Confusion matrix

Resources of implemented design are given in Table 3. All DSP resources and some LUT resources are utilized to perform parallel integer and floating-point arithmetic operations to achieve real-time performance. BRAMs modules have been used extensively to store images and intermediate processed data of different blocks.

6. Conclusion

Computer-vision based applications are computationally intensive and time consuming, so FPGA-based implementation is necessary to have a real-time identification of plant diseases. In this work, a real-time system on Xilinx ZYNQ platform is proposed for the detection and identification of potato leaves diseases. A computer-vision based algorithm is proposed for accurately detecting and identifying the potato leaf diseases under various luminance conditions. The proposed algorithm is then ported to ARM cortex A9 processor on ZYNQ, where the execution performance of the algorithm is analyzed.

Table 2

Execution Performance comparison with existing work

Author	Hardware	Accuracy achieved	Performance
Proposed algorithm software (C/C++) implementation	ZYNQ PS ARM-A9 Processor running at 667 MHz	Between 90% and 95%	2.21
Proposed algorithm hardware implementation on Zynq ZED Board	ZYNQ PS ARM-A9 Processor running at 667 MHz and PL running at 100 MHz and 166 MHz	90%	0.095
Talha Akram et al. **[18]**	Spartan-6 FPGA Embedded Kit, DK-S6-EMBD-G	Between 63% and 87%	0.169

The hardware accelerator is then designed for the proposed algorithm on FPGA fabric of ZYNQ, where different optimizations are applied at the algorithmic and architectural level to increase the execution performance on FPGA. The proposed implementation on ZYNQ FPGA achieves a performance gain of 76.8 times as compared to software implementation on general-purpose computer and 43.79% than previous work.

Table 3

Resource usage of the proposed hardware on Xilinx ZYNQ XZ7020

Recourses	Utilization	Available	Utilization (%)
LUTs	29582	53200	55.61
LUTRAM	866	17400	4.98
Flipflops	41496	106400	39.00
Block Ram	109	140	77.86
DSP Blocks	220	220	100.00
IO Blocks	22	200	11.00

7. References

[1] J. Francis, S. A. Dhas D and A. B. K, "Identification of leaf diseases in pepper plants using soft computing techniques", Conference on Emerging Devices and Smart Systems, Namakkal, INDIA, 2016.

[2] A. A. Joshi and B. Jadhav, "Monitoring and controlling rice diseases using image processing techniques", in International Conference on Computing, Analytics and Security Trends, College of Engineering Pune, India., 2016.

[3] J. W. Orillo, J. D. Cruz, L. Agapito, P. J. Satimbre and I. Valenzuela, "Identification of diseases in rice plant (Oryza Sativa) using back propagation artificial neural network", Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management, Palawan, Philippines, 2014.

[4] R. Anand, S. Veni and J. Aravinth, "An application of image processing techniques for detection of diseases on brinjal leaves using k-means clustering method", International Conference on Recent Trends In Information Technology, Chennai, India, 2016.

[5] S. S. Sannakki, V. S. Rajpurohit, V. B. Nargund and P. Kulkarni, "Diagnosis and classification of grape leaf diseases using neural networks", International Conference on Computing, Communications and Networking Technologies, Tiruchengode, India, 2013.

[6] S. Prasad, S. K. Peddoju and D. Ghosh, "Energy efficient mobile vision system for plant leaf disease identification," IEEE Wireless Communications and Networking Conference, Istanbul, Turkey, 2014.

[7] S. Phadikar and J. Goswami, "Vegetation indices based segmentation for automatic classification of brown spot and blast diseases of rice", 3rd Int'l Conf. on Recent Advances in Information Technology, Salt Lake, Kolkata, India, 2016.

[8] N. Otsu, "A threshold selection method from gray-level histograms", IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, no. 1, pp. 62-66, Jan 1979.

[9] S. D. Khirade and A. B. Patil, "Plant disease detection using image processing", International Conference on Computing Communication Control and Automation, Pune, India, 2015.

[10] K. Majid, Y. Herdiyeni and A. Rauf, "I-PEDIA: Mobile application for paddy disease identification using fuzzy entropy and probabilistic neural network", International Conference on Advanced Computer Science and Information Systems, Bali, Indonesia, 2013.

[11] P. Pawar, V. Turkar and P. Patil, "Cucumber disease detection using artificial neural network", International Conference on Inventive Computation Technologies, Coimbatore, India, 2016.

[12] J. P. Shah, H. B. Prajapati and V. K. Dabhi, "A survey on detection and classification of rice plant diseases", IEEE International Conference on Current Trends in Advanced Computing, Bangalore, India, 2016.

[13] J. D. Pujari, R. Yakkundimath and A. S. Byadgi, "Identification and classification of fungal disease affected on agriculture/horticulture crops using image processing techniques", IEEE International Conference on Computational Intelligence and Computing Research, Coimbatore, India, 2014.

[14] S. Biswas, B. Jagyasi, B. P. Singh and M. Lal, "Severity Identification of potato late blight

disease from crop images captured under uncontrolled environment", IEEE Canada International Humanitarian Technology Conference, Montreal, QC, Canada, 2014.

[15] K. K. Chakraborty, R. Mukherjee, C. Chakroborty and K. Bora, "Automated recognition of optical image based potato leaf blight diseases", Physiological and Molecular Plant Pathology, vol. 117, p. 10, 2022.

[16] T. Y. Lee, . I. A. Lin, J. Y. Yu, J. m. Yang and Y. C. Chang, "High efficiency disease detection for potato leaf with convolutional neural network", SN Computer Science, p. 11, 2021.

[17] B. L. Benoso, J. C. M. Perales, J. C. Galicia, R. F. Carapia and V. M. Silva-García, "Detection of diseases in tomato leaves by color analysis", Electronics, p. 16, 2021.

[18] T. Akram, S. R. Naqvi, S. A. Haider and M. Kamran, "Towards real-time crops surveillance for disease classification: exploiting parallelism in computer vision", Computers and Electrical Engineering, 2017.

[19] C. Hou, J. Zhuang, Y. Tang, Y. He, A. Miao, H. Huang and S. Luo, "Recognition of early blight and late blight diseases on potato leaves based", Journal of Agriculture and Food Research, vol. 5, no. 100154, p. 12, 2021.

[20] D. Commons, "Mendeley data", Digital Commons Data, 28 4 2019. [Online]. Available: https://data.mendeley.com/datasets/tywbtsjrjv/1.

[21] M. Hassan and C. Bhagvati, "Optimal color palette for error diffusion techniques", Proceedings of the Fourth International Conference on Signal and Image Processing, 2012.

