A Real-Time (or) Field-based Research Project Report

On

Early Pest Detection From Crop Using Image Processing And Computational Intelligence

submitted in partial fulfilment of the requirements for the award of the degree

of

Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING

by

G. HARSHITHA [227R1A05E5]

S. AMIT [227R1A05J5]

M. RAHUL [227R1A05G1]

Under the guidance of

Ms Tabeen Fatima

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

Accredited by NBA & NAAC with 'A' Grade

Approved by AICTE, New Delhi and JNTUH Hyderabad

Kandlakoya (V), Medchal Road, Hyderabad-501401

JUNE 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Real-Time (or) Field-based Research Project Report entitled "**Early Pest Detection From Crop Using Image Processing And Computational Intelligence**" being submitted by **G.HARSHITHA(227R1A05E5), S.AMIT (227R1A05J5), M.RAHUL (227R1A05G1)** in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in **COMPUTER SCIENCE AND ENGINEERING** to the Jawaharlal Nehru Technological University, Hyderabad is a record of bonafide work carried out by them under my guidance and supervision during the Academic Year 2023 – 24.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any other degree or diploma.

Ms. Tabeen Fatima
Assistant Professor

Dr. K. Srujan Raju
Head of the Department

Dr. A. Raji Reddy
Director

ABSTRACT

Early pest detection is a major challenge in agriculture field. The easiest way, to control the pest infection is the use of pesticides. But the excessive use of pesticides are harmful to plants, animals as well as human beings. Integrated pest management combines biological and physical methods to prevent pest infection. The techniques of machine vision and digital image Processing are extensively applied to agricultural science and it have great perspective especially in the plant protection field, which ultimately leads to crops management.

This paper deals with a new type of early detection of pests system. Images of the leaves affected by pests are acquired by using a digital camera. The leaves with pest images are processed for getting a gray colored image and then using feature extraction, image classification techniques to detect pests on leaves. The images are acquired by using a digital camera. The images are then transferred to a PC and represented in python software. The RGB image is then converted into gray scale image and the feature extraction techniques are applied on that image. The Support Vector Machine classifier is used to classify the pest types.

TABLE OF CONTENTS:

1.INTRODUCTION

1.1 What is Python :-

Python is currently the most widely used multi-purpose, high-level programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc. The biggest strength of Python is huge collection of standard library which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

1.2 History of Python : -

Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners[1], Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it."Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

Python Development Steps : -

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt. Sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and has modules. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been

on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

- Print is now a function
- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g. a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e. int. long is int as well.
- The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
- Text Vs. Data Instead Of Unicode Vs. 8-bit

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge

standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

1.3 Modules Used in Project :-

Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

1.4 Advantages of Python :-

Extensible

As we have seen earlier, Python can be **extended to other languages**. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

Improved Productivity

The language's simplicity and extensive libraries render programmers **more productive** than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

Simple and Easy

When working with Java, you may have to create a class to print **'Hello World'**. But in Python, just a print statement will do. It is also quite **easy to learn, understand,** and **code.** This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and **indentation is mandatory.** This further aids the readability of the code.

Object-Oriented

This language supports both the **procedural and object-oriented** programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the **encapsulation of data** and functions into one.

Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to **code only once**, and you can run it anywhere. This is called **Write Once Run Anywhere (WORA)**. However, you need to be careful enough not to include any system-dependent features.

Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in **slow execution**. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the **client-side**. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called **Carbonnelle**.
The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions

As you know, Python is **dynamically-typed**. This means that you don't need to declare the type of variable while writing the code. It uses **duck-typing**. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can **raise run-time errors**.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like **JDBC (Java DataBase Connectivity)** and **ODBC (Open DataBase Connectivity)**, Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

1.5 Early Pest Detection From Crop Using Image Processing And Computational Intelligence

India is an agricultural country. 70 percent of the people mainly depends upon agriculture. So increasing the productivity of crops is an important matter now. Most of the scientists are doing their researches on this field. By using their new techniques and practical implementations this is very easy. But one of the most important problem now exists is „pest infection" on plants. This paper mainly focuses on greenhouse crops.

There are different crops cultivated under greenhouse. for example, vegetables like cucumber, potato, tomato etc and flower plants like rose, jasmine etc. The most common pests which will affect on this green house crops are whiteflies a, aphids and thrips. One way to control the pest infection is by using the pesticides. Pesticides will suppress particular species of pests. Pesticides are detrimental for the environment and produce considerable damage to eco systems.

The excessive use of pesticides will pollute air, water, and soil. Carried by the wind pesticides suspensions contaminate other areas. In this paper, we focus on early pest detection. This implies to regular observation the plants. Images are acquired using cameras. Then the acquired image has to be processed to interpret the image contents by image processing methods. The focus of this paper is on the interpretation of image for pest detection.

2.LITERATURE SURVEY

In this section we will discuss some methods which are presently used for the early detection of pests in greenhouse crops along with their advantages and disadvantages. The methods are explained below with their features and drawbacks.

Detection of Pests Using Video Analysis

This work combines image processing techniques as well as knowledge based technique. It will detect only whiteflies. The resulst of this system are more reliable and accurate than that of the manual methods. This is actually a multidisciplinary cognitive vision system that combines different types of techniques like computer vision , artificial intelligence, image processing etc. In this work, they chose rose plant as the testing crop and white fly as the pest for testing. The early stage of detection was quite difficult. So they chose adult flies. But some problems were there in detection of adult also. The adult may fly away during the image capturing time. So they chose to scan the leaves of rose when flies were not active. The future scope of the work is to detect whiteflies in its early stage.

Method which use Sticky Traps

The goal of Detection of insects by a video camera network is to detect the pest infection on leaves by using a video analysis. The traditional methods will take more time to detect and count the pests. Because of this reason they have developed an automatic system based on video analysis. They used 5 wireless cameras in greenhouse. They chose rose as a crop for testing . sticky traps are used in this work. Sticky traps are nothing but a sticky material which is having some colours to attract the pests. For the detection of insects, they used video segmentation algorithms with learning and adaptation techniques. The adaptive system can be used in any weather conditions. The future scope of this system is to detect new types of pests in early stage.

3.ANALYSIS AND DESIGN

3.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the

users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.2 SYSTEM DESIGN

UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

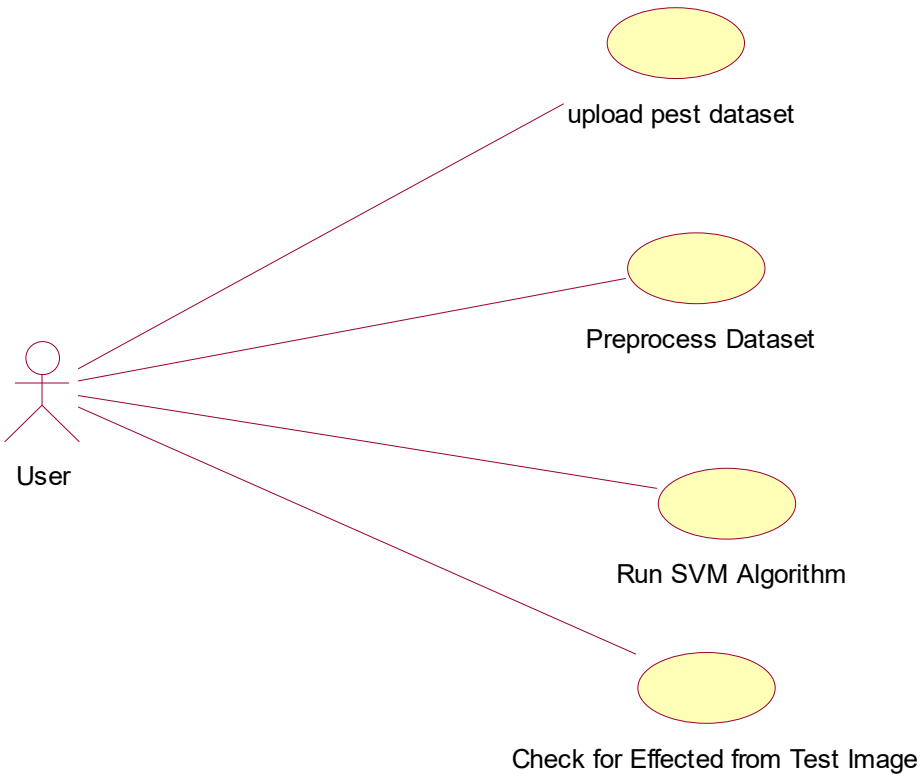
The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.

5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

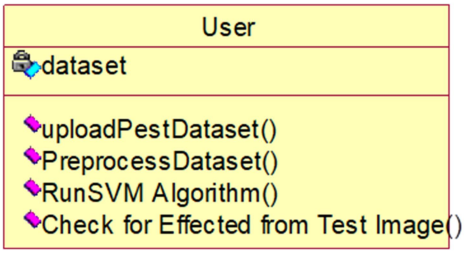
USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

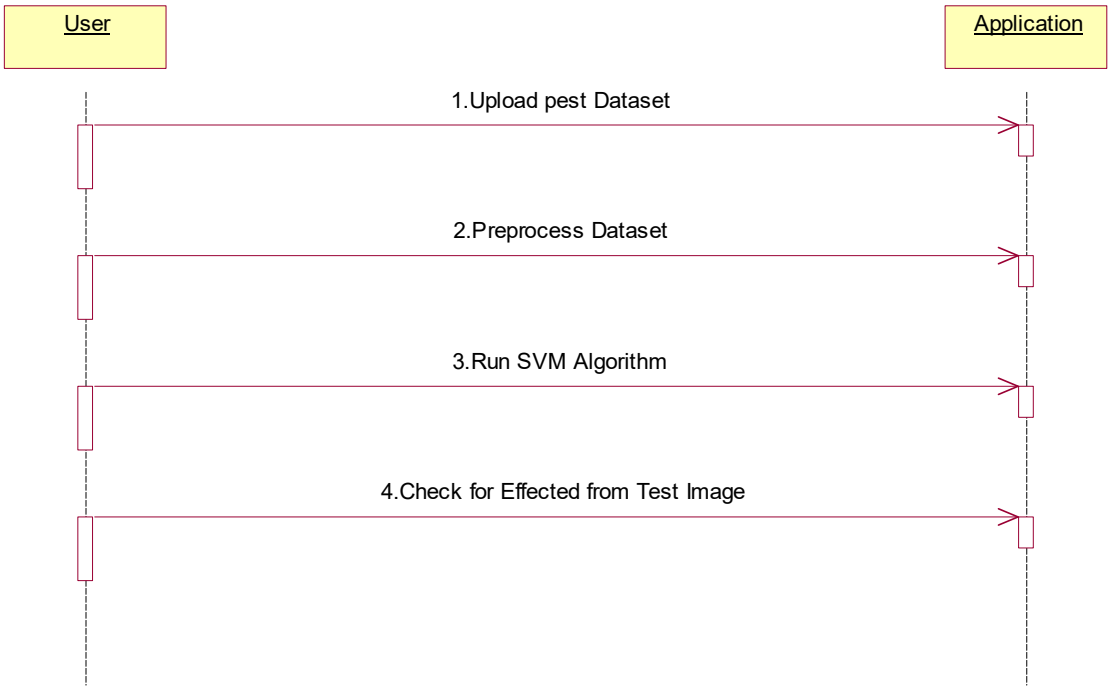


CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes.

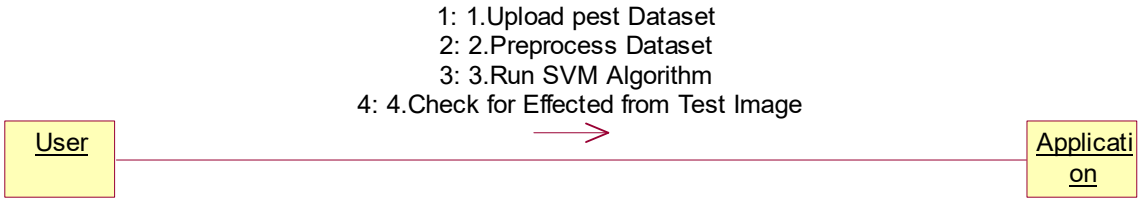


SEQUENCE DIAGRAM:



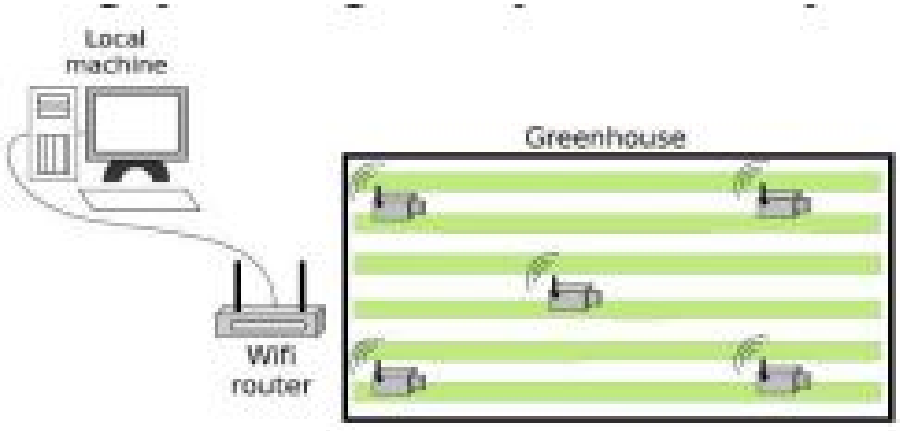
A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenario and timing diagrams.

COLLABORATION DIAGRAM:



PROPOSED METHOD

For this study, whiteflies and aphids are chosen because this pest requires early detection and treatment to prevent durable infection. Samples are collected by using the pan tilt camera with zoom in greenhouse as shown in Fig.1.The acquired Images are given to the local machine and the image processing techniques will takes place.



System Architecture

3.3 METHODOLOGY

Image capturing

The first step of every image processing application is image acquisition or image capturing. The images of leaves are captured by using the camera and it will store it in some formats like .PNG, .JPG, .JPEG etc.

Image pre-processing

Image preprocessing is used to create an enhanced and please full version of the captured image. The image preprocessing steps used in the system are:

1) Conversion of RGB image to gray image

2) Resizing of the image

3) Filtering of the image.

a) Conversion of RGB to Gray Image

In RGB color model, each color appears in its primary spectral components of red, green, and blue. The color of a pixel is made up of three components; red, green, and blue (RGB). The disadvantages of RGB models are, it requires large space to store and it will take more time to process. So there is a need for converting the RGB model to Gray model .

b) Resizing of the Image

Resizing is an important step in image preprocessing. The acquired image is resized according to the requirement of the system. Resizing of the image: Resizing is nothing but, changing the dimensions of an image. The captured image is resized using some resizing methods according to the requirement of the system. There are different methods for the

resizing of images. Blinear, Bicubic and Nearest neighborhood interpolation are the common resizing methods. Here in our system, we are using bicubic method

c) Filtering of the image

Filtering is nothing but, eliminating the unwanted portion of the image. Different types of filters are available. Low pass filters are smoothening filters, it will pass only low frequency signals and eliminate all the high frequency signals. High pass filters are sharpening filters, and it will eliminate all the low frequency signals and pass only high frequency signals. Band pass filters will pass the signals which is having a specific range of frequencies. In our system we are using smoothening filter. The purpose of smoothing is to reduce noise and improve the visual quality of the image. Spatial filters are applied to both static and dynamic images, whereas temporal images are applied only to dynamic images. The simplest smoothening filter is average filter. It consists of a 3X3 matrix of 1 and it is divided by 9

3) Feature Extraction

Feature extraction is the most important part of this project. Some properties of the images are considered here. The different types of properties includes region properties, gray covariance matrix properties etc. The properties standard deviation, entropy, contrast etc are extracted from the image and are used to train the dataset for the SVM classification. Support Vector Machines (SVM's) are a relatively new learning method used for binary classification. The basic idea is to find a hyper plane which separates the d-dimensional data perfectly into its two classes. The different types of properties of an image is listed in the table below fig1.

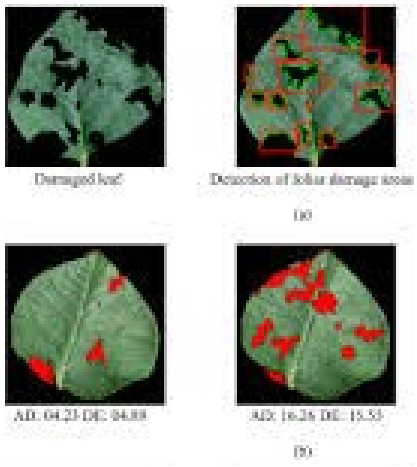
TABLE I: PROPERTIES OF AN IMAGE

Mean	Returns the mean value of the elements along different parameters of an array
Standard Deviation	Computes the standard deviation of the values in matrix.
Contrast	Returns a measure of intensity contrast between pixels.
Energy	Returns the sum of squired elements in the glcm.
Filled Area	Scalar specifying the number of pixels in filled area

Fig1: IMAGE PROPERTIES

Detection and Classification

In this module the affected and unaffected images are compared by using the dataset provide in the SVM. If it is an affected image again it is compared by using the second dataset provided in the SVM. From this comparison the type of pest can be detected



Affected leafs

FLOWCHART

Flowchart for the proposed system is given in figure 3. The images are acquired by using camera and it is filtered by using bicubic filters to avoid unwanted noise portions. This is actually the image preprocessing step. The next step is svm classification to detect the pest infection. If the image is affected, then again it is applied to the svm to detect the type of pest

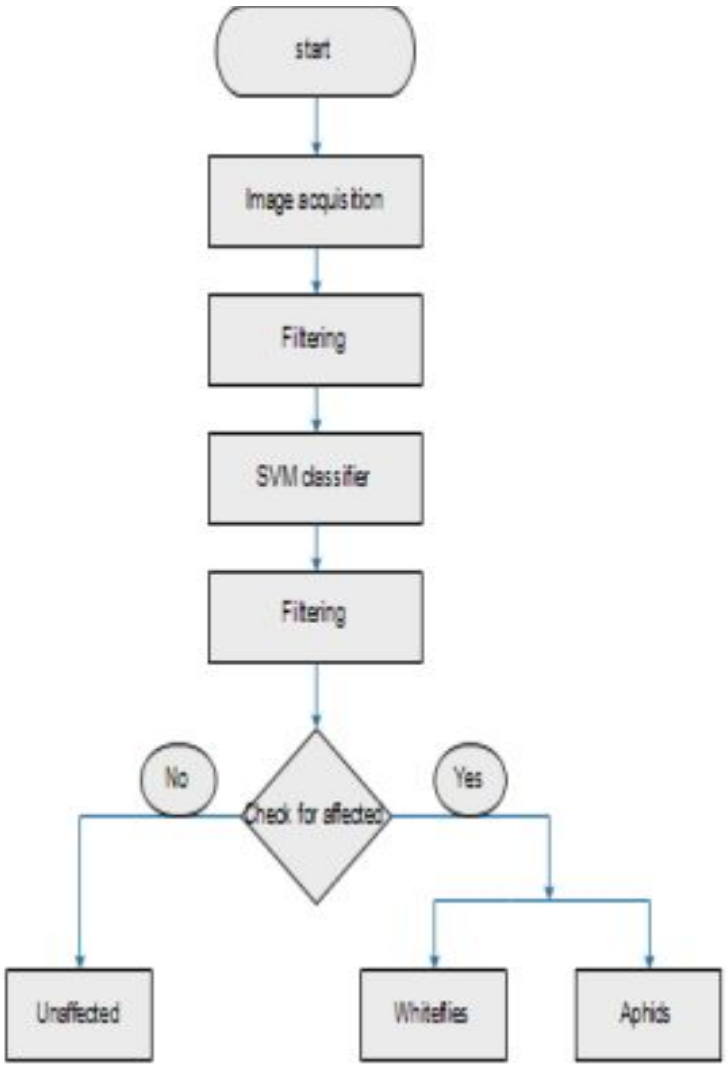


fig2 steps followed

4.IMPLEMENTATION:

4.1 MODULES:

1) Upload Pest Dataset:

using this module we will upload dataset to application

2) Preprocess Dataset:

using this module we will acquire images from dataset and then filter images to grey colour and then normalize images and then split dataset into train and test part where application use 80% images for training and 20% for testing

3) Run SVM Algorithm:

process images will be input to SVM algorithm for training and then calculate its prediction accuracy.

4) Check for Effected from Test Image:

using this module we will upload test image and then SVM will predict type of pest as Aphid, White fly or Uneffected.

The Pest Detection System is structured to automate and enhance the process of identifying pests in greenhouse crops through a series of well-defined modules. Starting with the **Upload Pest Dataset** module, users can seamlessly input their data into the system, setting the stage for a comprehensive analysis. This initial step is crucial as it ensures that the system has a robust set of images to learn from, covering various scenarios and types of pests.

4.2 CODE:

```
from tkinter import *

import tkinter

from tkinter import filedialog

import numpy as np

from tkinter.filedialog import askdirectory

from tkinter import simpledialog

from sklearn.model_selection import train_test_split

import cv2

from sklearn import svm

main = tkinter.Tk()

main.title("Early Pest Detection From Crop Using Image Processing And Computational Intelligence") #designing main screen

main.geometry("1000x650")

global filename

global X,Y

global X_train, X_test, Y_train, Y_test

global predicted_data

labels = ['Aphids','Uneffected','Whitefly']

global svm_classifier
```

```
def getID(name):
  index = 0
  for i in range(len(labels)):
    if labels[i] == name:
      index = i
      break
  return index
def upload():
  global filename
  filename = filedialog.askdirectory(initialdir = ".")
  text.delete('1.0', END)
  text.insert(END,filename+' Loaded')
def preprocess():
  global X, Y
  global X_train, X_test, Y_train, Y_test
  X = []
  Y = []
  text.delete('1.0', END)
  if os.path.exists('model/X.txt.npy'):
    X = np.load('model/X.txt.npy')
```

```
  Y = np.load('model/Y.txt.npy')
  X = X.astype('float32')
  X = X/255
  indices = np.arange(X.shape[0])
  np.random.shuffle(indices)
  X = X[indices]
  Y = Y[indices]
else:
  for root, dirs, directory in os.walk(filename):
    for j in range(len(directory)):
      name = os.path.basename(root)
      print(name+" "+root+"/"+directory[j])
      if 'Thumbs.db' not in directory[j]:
        img = cv2.imread(root+"/"+directory[j],0)
        img = cv2.resize(img, (28,28))
        im2arr = np.array(img)
        im2arr = im2arr.reshape(28,28)
        X.append(im2arr.ravel())
        Y.append(getID(name))
  X = np.asarray(X)
```

```
  Y = np.asarray(Y)
  np.save('model/X.txt',X)
  np.save('model/Y.txt',Y)
  X = np.load('model/X.txt.npy')
  Y = np.load('model/Y.txt.npy')
  X = X.astype('float32')
  X = X/255
  indices = np.arange(X.shape[0])
  np.random.shuffle(indices)
  X = X[indices]
  Y = Y[indices]
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
text.insert(END,"Total images found in dataset: "+str(X.shape[0])+"\n\n")
text.insert(END,"Total classes found in dataset: "+str(labels)+"\n\n")
text.insert(END,"Dataset train and test split 80 and 20%\n\n")
text.insert(END,"Training 80% images: "+str(X_train.shape[0])+"\n")
text.insert(END,"Testing 20% images: "+str(X_test.shape[0])+"\n")
test = X[3]
test = test.reshape(28,28)
test = cv2.resize(test,(200,200))
```

```
  cv2.imshow("Process Image",test)
  cv2.waitKey(0)


def runSVM():
  text.delete('1.0', END)
  global svm_classifier
  global X, Y
  global X_train, X_test, Y_train, Y_test
  svm_classifier = svm.SVC()
  svm_classifier.fit(X, Y)
  predict = svm_classifier.predict(X_test)
  svm_acc = accuracy_score(Y_test,predict)*100
  text.insert(END,"SVM Prediction Test Accuracy: "+str(svm_acc))
def checkEffected():
  global svm_classifier
  filename = filedialog.askopenfilename(initialdir="testImages")
  image = cv2.imread(filename,0)
  img = cv2.resize(image, (28,28))
  im2arr = np.array(img)
  im2arr = im2arr.reshape(28,28)
```

```
  img = np.asarray(im2arr)
  img = img.astype('float32')
  img = img/255
  temp = []
  temp.append(img.ravel())
  predict = svm_classifier.predict(np.asarray(temp))
  predict = predict[0]
  print(predict)
  img = cv2.imread(filename)
  img = cv2.resize(img, (400,400))
  cv2.putText(img, 'Pest Detected as : '+labels[predict], (10, 25), cv2.FONT_HERSHEY_SIMPLEX,0.7, (0, 0, 255), 2)
  cv2.imshow('Pest Detected as : '+labels[predict], img)
  cv2.waitKey(0)
def close():
    main.destroy()
font = ('times', 16, 'bold')
title = Label(main, text='Early Pest Detection From Crop Using Image Processing And Computational Intelligence', justify=LEFT)
title.pack()
font1 = ('times', 13, 'bold')
```

```
uploadButton = Button(main, text="Upload Pest Dataset", command=upload)

processButton = Button(main, text="Preprocess Dataset", command=preprocess)

processButton.config(font=font1)

svmButton = Button(main, text="Run SVM Algorithm", command=runSVM)

svmButton.place(x=650,y=100)

svmButton.config(font=font1)

testButton = Button(main, text="Check for Effected from Test Image", command=checkEffected)

testButton.place(x=10,y=150)

text=Text(main,height=20,width=120)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=10,y=300)

text.config(font=font1)

main.config(bg='light coral')

main.mainloop()
```

5.TESTING AND DEBUGGING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

5.1 TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of

components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

5.2 Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

5.3 Acceptance Testing and Debugging

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Debugging in the context of early pest detection using image processing and computational intelligence involves a systematic approach to identify and resolve issues in the software system. This system processes images and utilizes algorithms to detect pests like whiteflies and aphids. The debugging process begins with reproducing the error, which requires collecting and analyzing the images and conditions under which these errors occur. By recreating these conditions, developers can consistently observe the issue and begin to understand its root causes.

Diagnosing the issue is the next crucial step. This involves checking various components of the system, such as data preprocessing, image processing algorithms, computational intelligence models, and system integration. For instance, if there are inconsistencies in preprocessing steps like filtering or resizing images, these need to be corrected. Similarly, if the feature extraction or classification algorithms are flawed, these components must be validated and corrected. Reviewing the training data for the models is also essential to ensure it is sufficient and diverse enough to handle various pest scenarios.

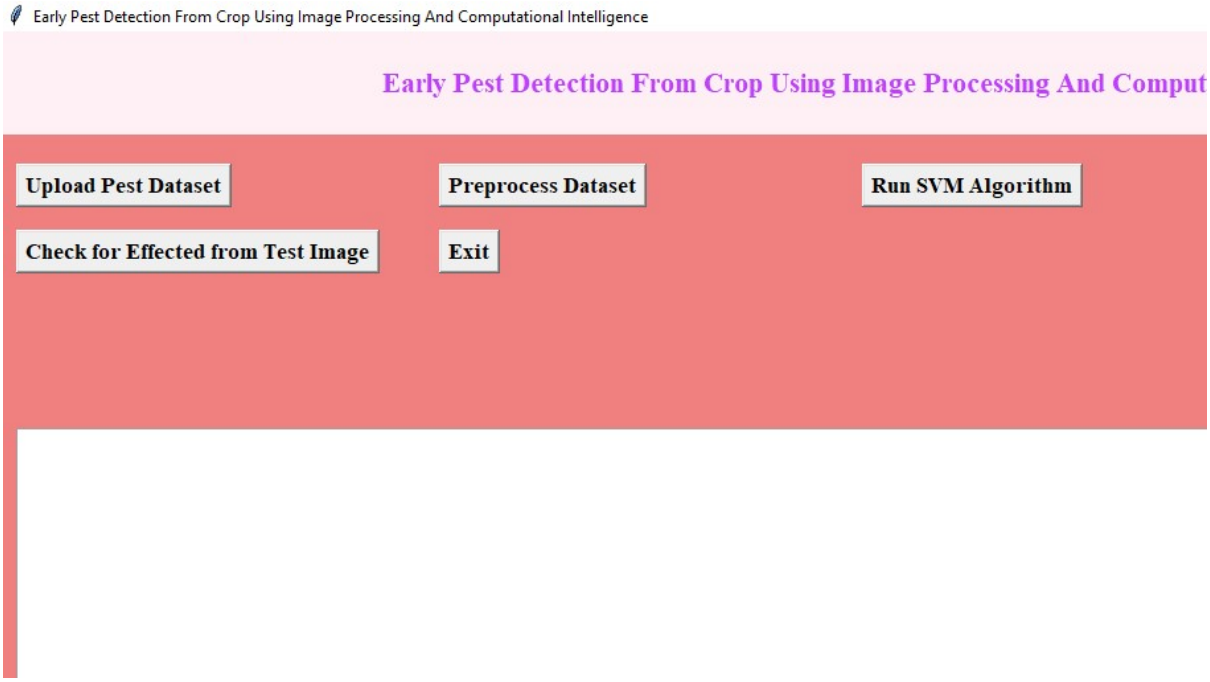
Integration issues, where different components of the system do not communicate correctly, should also be identified and resolved through comprehensive testing.

Once the issues are diagnosed, fixing them involves updating algorithms, re-training models, and optimizing the processing pipeline. Adjustments to image processing algorithms might include refining feature extraction methods to improve accuracy. Re-training computational intelligence models with augmented datasets can enhance their robustness. Additionally, optimizing the preprocessing and processing pipelines ensures consistency and efficiency, balancing image quality and processing speed.

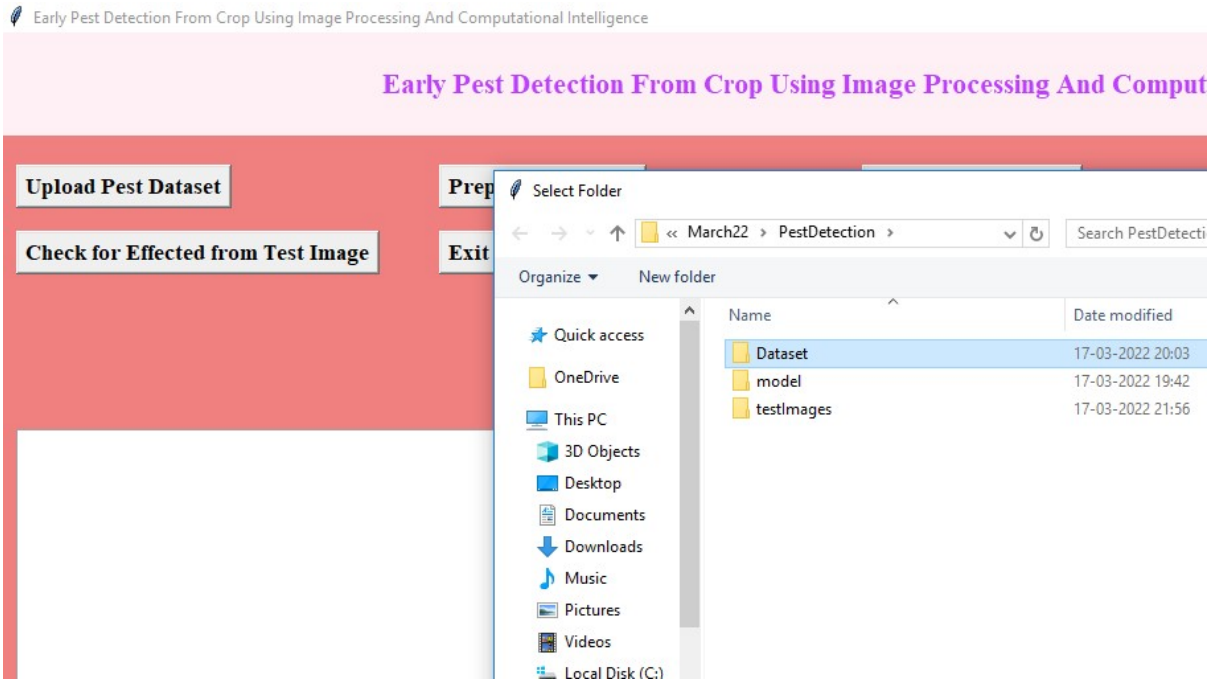
Testing the fix is a critical phase to ensure the problem is resolved and no new issues are introduced. This includes unit testing of individual components, integration testing of combined system elements, functional testing to verify the system meets all requirements, and system testing to ensure it operates correctly in real-world environments. Using a variety of test images, including those with edge cases and challenging scenarios, helps validate the system's robustness.

Finally, reviewing the changes through code reviews and continuous performance monitoring ensures the long-term reliability of the system. Code reviews help verify that the modifications are correct, efficient, and maintainable. Performance monitoring in the field allows for the collection of feedback and adjustments to algorithms and models as necessary, maintaining high accuracy and reliability. This systematic debugging process ensures that the pest detection system remains robust and accurate, effectively aiding in early pest detection and reducing the need for pesticides.

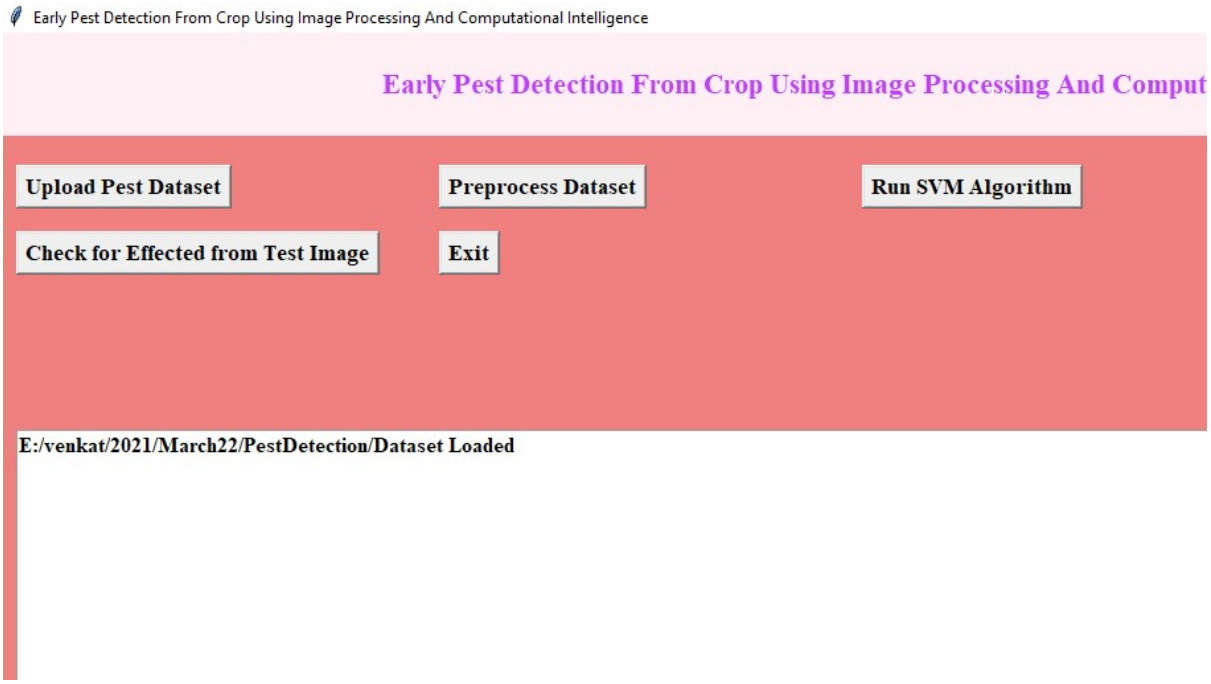
6.RESULTS



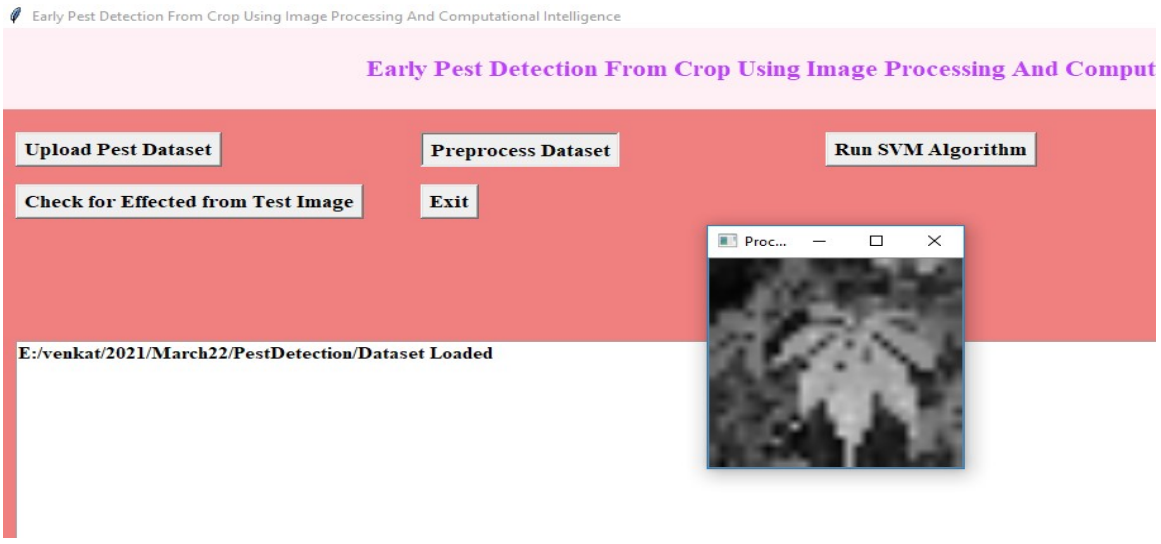
In above screen click on 'Upload Pest Dataset' button to upload dataset and to get below screen



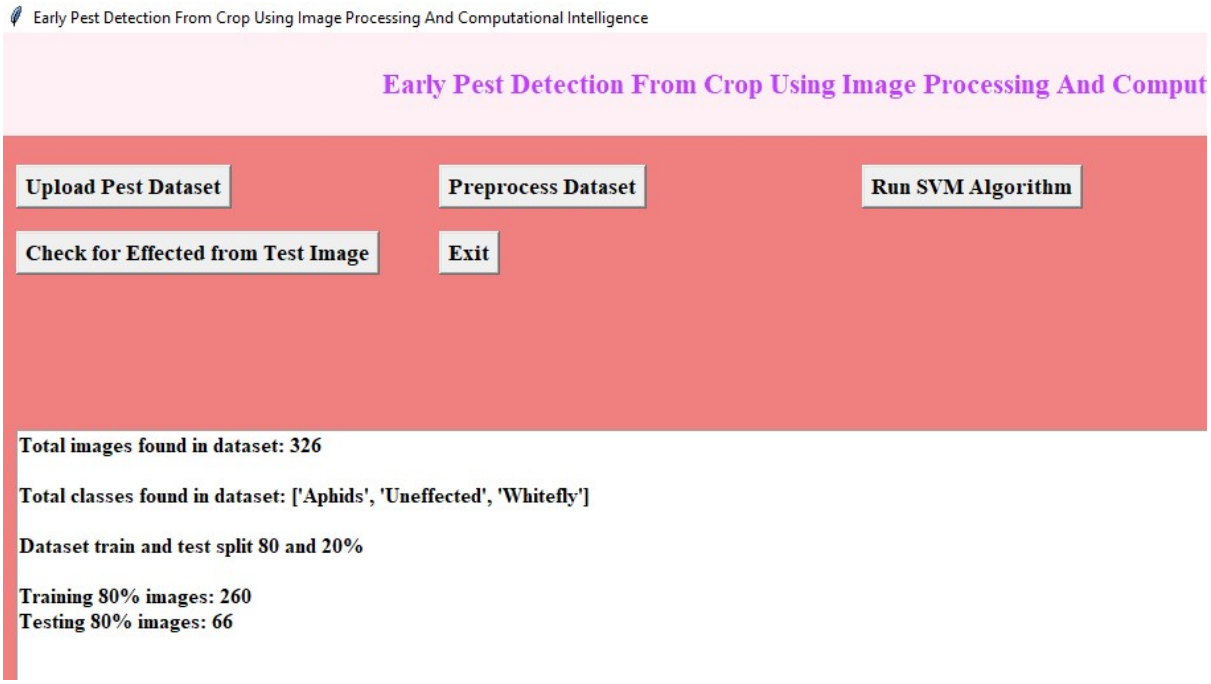
In above screen select and upload ‘Dataset’ folder and then click on ‘Select Folder’ button to load dataset and to get below screen



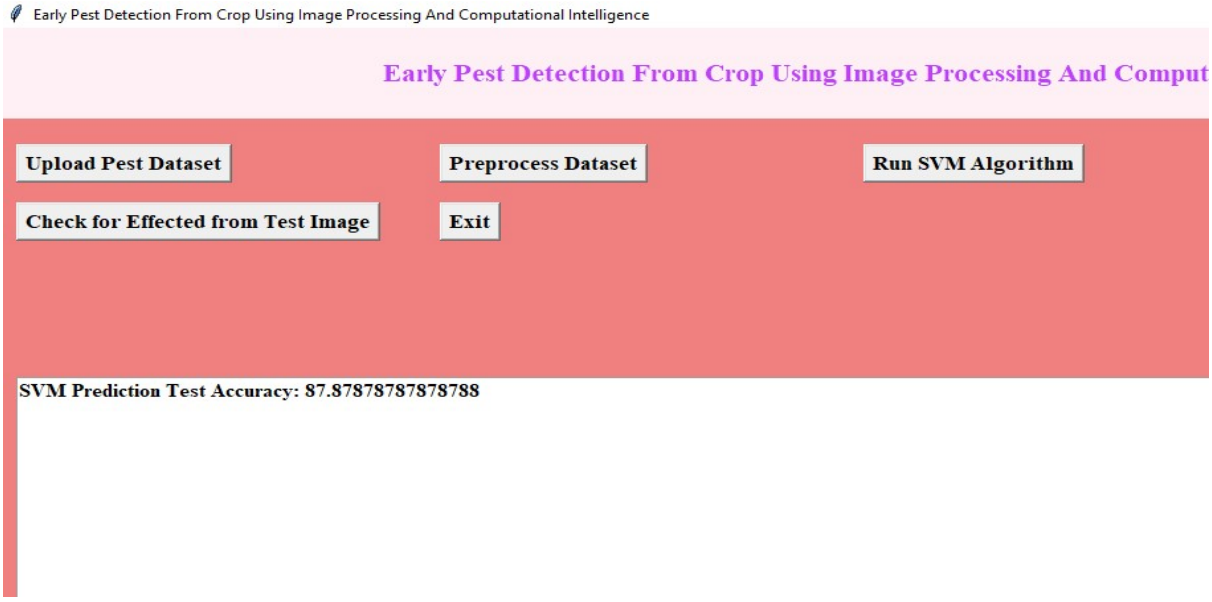
In above screen dataset loaded and now click on ‘Preprocess Dataset’ button to read and normalize images and then split dataset into train and test part.



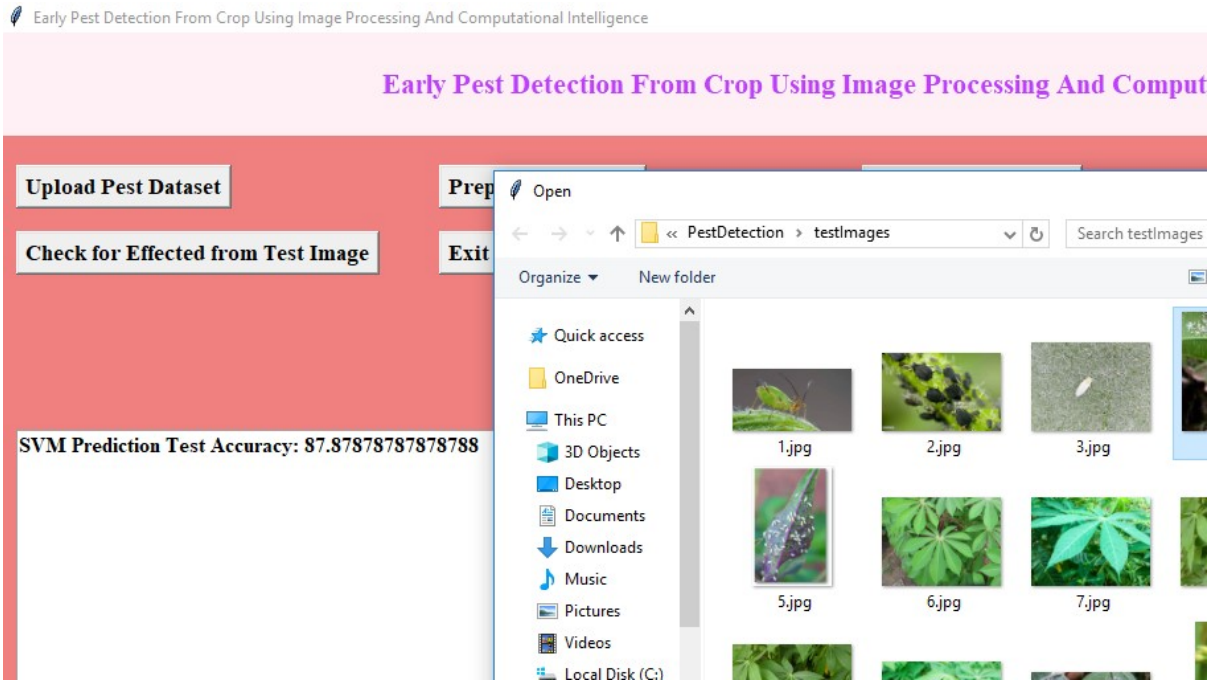
In above screen displaying processed grey image and now close above image to get below screen



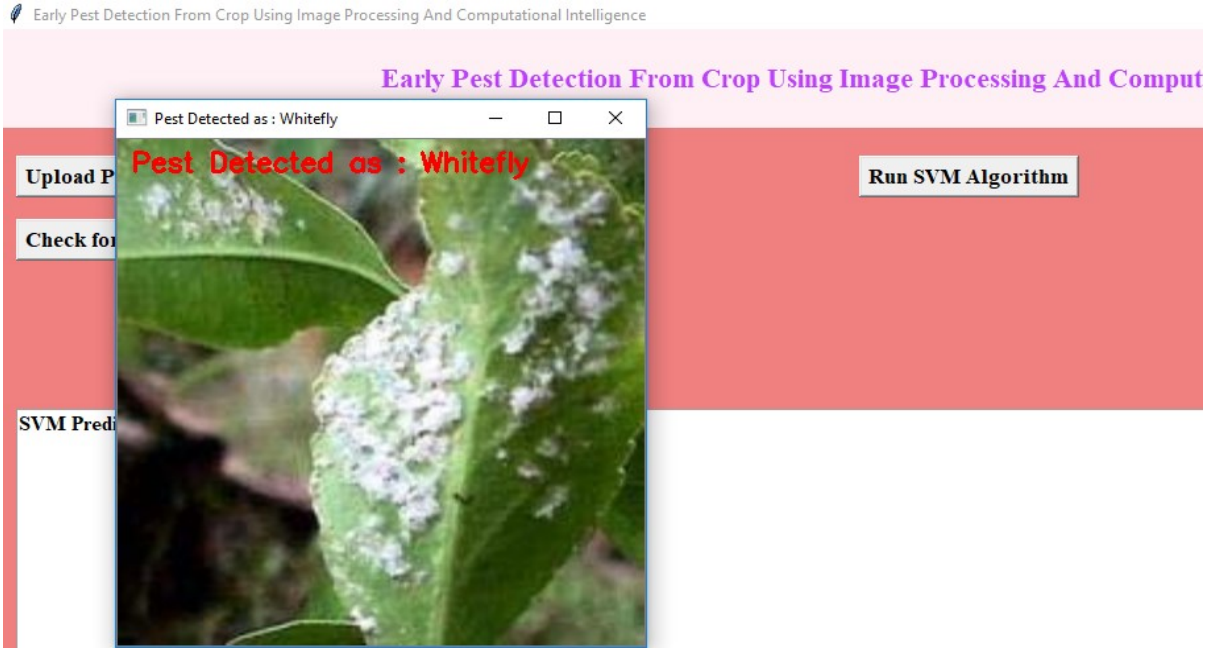
In above screen we can see number of images and classes found in dataset and now click on 'Run SVM Algorithm' button train SVM with processed images and then calculate it's prediction accuracy



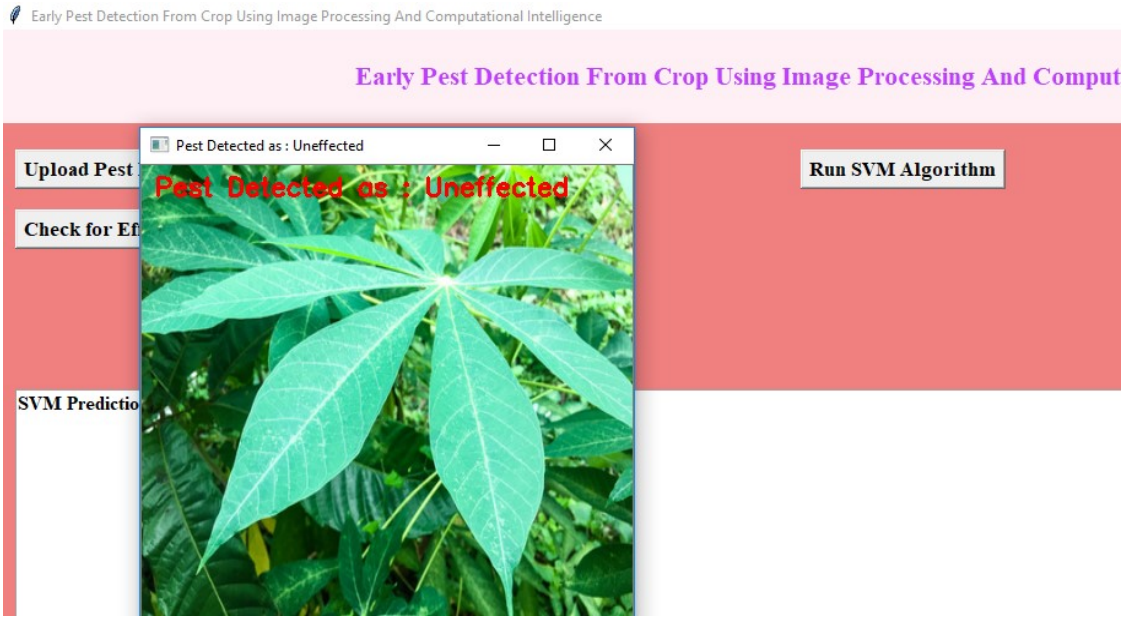
In above screen with SVM we got 87% prediction accuracy and now click on 'Check for Effected from Test Image' button to upload test image like below screen



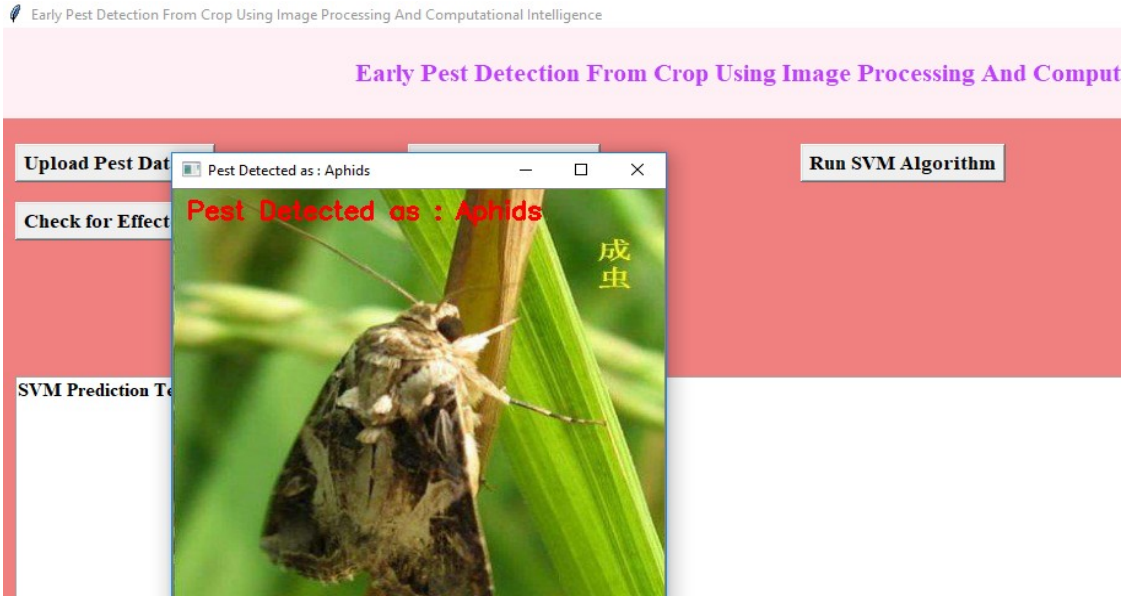
In above screen selecting and uploading 4.jpg file and then click on 'Open' button to get below output



In above screen in red colour text we can see SVM predicted/classified uploaded image as ‘whitefly’ and similarly you can upload and test other images



In above screen uploaded image is predicted as ‘Uneffected’ as it not contains any pest.



In above screen uploaded image is classifier as ‘Aphids’

7.CONCLUSION

Image processing techniques are increasingly important for the detection of pests in greenhouse crops. Our project focuses on the early detection of two common pests: whiteflies and aphids. To achieve this, we have developed a novel approach that employs a pan-tilt camera with zoom capabilities. This advanced camera system allows us to capture high-quality images of the crops without disturbing the pests, ensuring that we can monitor their presence and activity accurately and unobtrusively.

Our approach exemplifies the integration of multiple complementary disciplines and techniques, resulting in a system that is automated, robust, and versatile. The prototype we have developed has been tested extensively and has proven to be reliable for the rapid detection of pests. It is designed to be user-friendly, enabling even those with minimal technical expertise to operate it effectively. Despite its simplicity, our system matches the performance level of classical manual pest detection methods, offering a modern, efficient alternative.

The primary goal of our project is to detect pests as early as possible, which is crucial for effective pest management. Our innovative system thus supports sustainable agriculture practices by providing a practical solution for early pest detection and management.

In summary, our image processing-based pest detection system is a significant advancement in agricultural technology. It combines sophisticated imaging techniques with automated analysis to provide a reliable, easy-to-use tool for greenhouse crop monitoring. This system not only enhances the efficiency of pest detection but also contributes to the broader goals of reducing pesticide usage and supporting sustainable farming practices.

8.REFERENCES

[1]. Martin,V.Thonnat,. “A Learning Approach For Adaptive Image Segmentation.” In Proceedings of IEEE Trans.Computers and Electronics in Agriculture.2008

[2]. Vincent Martin and Sabine Moisan, “Early Pest Detection in Greenhouses”. INRIA Sophia Antipolis M´editerrann´ee, PULSAR team 2004 route des Lucioles, BP93

[3]. Jayamala K. Patil1 , Raj Kumar, “Advances In Image Processing For Detection Of Plant Diseases” Journal Of Advanced Bioinformatics Applications and Research ISSN 0976-2604 Vol 2, Issue 2, June-2011, pp 135-141

[4]. Ikhlef Bechar and Sabine Moisan, “On-line counting of pests in a greenhouse using computer vision”. published in "VAIB 2010 - Visual Observation and Analysis of Animal and Insect Behavior (2010)"

[5]. Paul Boissarda, Vincent Martin, “A cognitive vision approach to early pest detection in greenhouse crops”. computers and electronics in agriculture 6 2 (2 0 0 8) 81–93

[6]. B.Cunha.“Application of Image Processing in Characterisation of Plants.” IEEE Conference on Industrial Electronics.2003.