INTELLIGENT FRAMEWORK FOR MONITORING CROPS IN GREENHOUSES

by

Asmaa Mohamed Al-Maamoun Ali

A thesis submitted to the School of Computing

in conformity with the requirements for

the degree of Doctor of Philosophy

Queen's University

Kingston, Ontario, Canada

(October 2020)

Abstract

Wireless Sensor Networks (WSNs) and Wireless Visual Sensor Networks (WVSNs) are two monitoring technologies that have the potential for use in many application domains, and both are poised for growth in many markets from the farm to the office. Integrating a WSN with a WVSN in a commercial greenhouse setting is an application domain yet to be researched. The integration of WSN and WVSN has the potential to overcome the problems other monitoring systems have encountered. A system combining these two wireless networks will need no human interaction, deliver real-time data indicating an adverse event, be cost-effective, and use less power. Additional efficiencies specific to the greenhouse application are needed due to its clutter and occluded environment, very large area, and restricted energy plan. This thesis presents a framework that combines WSN, WVSN, Machine Learning (ML), deep learning, and image processing to address the challenges faced by operators of commercial greenhouses. The framework achieves three objectives. First, finding the optimal placement of WVSN nodes to minimize the number of installed camera nodes. Second, monitoring the growth of the plants and detect any abnormalities caused by pests or diseases. Third, controlling the microclimate inside the greenhouse and dynamically predicting the duty cycle activities of the monitoring sensors.

Our first objective is achieved by formulating and solving an optimization problem to find the best placement for the camera sensors of the WVSN, maximize the area covered, and minimize the number of camera sensors used with good quality images. The second objective is achieved by using the Hough Forest ML and image processing techniques on the images taken by the WVSN to detect any fungus, monitor the growth of the plant, and to increase crop production and quality. Our third objective is achieved by controlling the microclimate inside the greenhouse using deep learning prediction Long Short-Term Memory (LSTM) model. The prediction model will not only

control the microclimate inside the greenhouse but also predict and control the monitoring sensor's duty cycle to decrease energy consumption and prolong the network's lifetime.

Co-Authorship

- A. Ali, H. S. Hassanein, “A Fungus Detection System for Greenhouses Using Wireless Visual Sensor Networks and Machine Learning”, *IEEE Globecom Workshops (GC Wkshps)*, pp. 1-6. IEEE, 2019. (Published)
- A. Ali, H. S. Hassanein, “Wireless Sensor Network and Deep Learning for Prediction Greenhouse Environments" *IEEE International Conference on Smart Applications, Communications and Networking (SmartNets)*, pp. 1-5., 2019. (Published)
- A. Ali, H. S. Hassanein, “Time-Series Prediction for Sensing in Smart Greenhouses”, *IEEE Global Communications Conference, Dec.* 2020. (Accepted)
- A. Ali, H. Hassanein, “Optimal Placement of Wireless Visual Sensor Cameras”, *IEEE Internet of Things Journal,* 2020 (In progress).
- A. Ali, H. S. Hassanein, “Prediction Sensor’s Duty Cycle Based on LSTM Model”, *IEEE Transactions on Network Science and Engineering,* 2020 (In progress).

Dedication

To my father Dr. Mohamed Al-Maamoun Ali and my mother Faiza Ali
To my husband Dr. Nidal Nasser
To my kids Yousef, Ibrahim, Zainab, and Mariam
... in all love, humility and gratitude

Acknowledgements

This Ph.D. thesis is the result of a challenging journey, upon which many people have contributed and given their support. At the end of this journey, it is a pleasant task to express my thanks to all those people who made this thesis possible and an unforgettable experience for me.

I thank Allah (SWT), the Holy, the Creator, the Most Gracious, the Most Merciful, and the Wise, who helped me and gave me the ability to be at this stage of research.

I am deeply thankful to my supervisor Prof. Hossam H. S. Hassanein. His distinctive supervision and mentorship were a great source of help and guidance throughout the course of my PhD program.

I would like to pay high regards to my husband Prof. Nidal Nasser for his great help and advice during this journey.

I am totally beholden to my father Prof. Mohamed Al-Maamoun Ali and my mother Faiza Ali who continuously supported and encouraged me. They have always been a great source of love, care, motivation, and prayers. They gave me so much that I can never pay them back. They always understood my anxieties and worries and stood beside me in every stage of life. I will owe them gratitude and thankfulness forever.

I express my thanks to my beloved beautiful children Yousef, Ibrahim, Zainab, and Mariam for their patience, happiness and motivation during this journey.

I want to express my deepest gratitude to my sister Maryam and my brother Yahia for their infinite love and support. I want to thank all my brothers and sisters Bilal, Anas, Doaa, and Fatma for their wonderful wishes and prayers.

I have always been blessed with magnificent friends in my life, who kept me going by providing a stimulating and fun filled environment. Words are short to express my gratitude towards my following friends, Abeer Al-Awadhi, Asia Elajnaf, Tahani Al-galahma, Reema Awartani, Abeer Hariri, Manal Jaz, Amatulraheem Abduallah and specially Abeer Alghias, for providing me moral support whenever I required, either directly or indirectly.

I want to thank Basia, Debby, members of the TRL lab and all members of the School of Computing at Queen's University for their support and friendship.

Besides this, I would like to extend my gratitude to one and all who have knowingly or unknowingly, directly, or indirectly helped me in the successful completion of this work.

Asmaa Mohamed Al-Maamoun Ali
Kingston, Ontario
5th, October 2020

Statement of Originality

I hereby certify that all of the work described within this thesis is the original work of the author. Any published (or unpublished) ideas and/or techniques from the work of others are properly referenced.

(Asmaa Mohamed Al-Maamoun Ali)

(October 2020)

Table of Contents

List of Figures

List of Tables

List of Abbreviations

WSN	Wireless Sensor Network
WVSN	Wireless Visual Sensor Network
ML	Machine Learning
LSTM	Long Short-Term Memory
KNN	K- Nearest Neighbor
ReLU	Rectified Linear Units
DT	Decision Tree
NB	Naive Bayes
SVM	Support Vector Machine
BS	Base Station
RF	Random Forest
MLP	Multi-Layer Perceptron
ANN	Artificial Neural Network
RNN	Recurrent Neural Network
FL	Focal Length
AOV	Angle of View
FOV	Field of View
AFOV	Angle Field of View
LFOV	Linear Field of View
CFD	Computational Fluid Dynamics
ILP	Integer Linear Programming
ILP-OPCQ	Integer Linear Programming- Optimal Placement Camera Quality
ROI	Region of Interest

ROC	Receiver Operating Characteristic
AUC	Area Under Curve
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
MSE	Mean Square Error
MAE	Mean Absolute Error
RMSE	Root Mean Square Error
IoT	Internet of Things
MCA-NN	Multivariate Correspondence Analysis Neural Network
B-MAC	Berkeley Medium Access Control
CNN	Cellular Neural Network
SGD	Stochastic Gradient Descent
PDCM	Prediction Duty Cycle Model
TDDCA	Traffic-adaptive Distance-based Duty Cycle Assignment
DDCA	Distance-based Duty Cycle Assignment
PDR	Packet Delivery Ratio
RN	Rely Node
SDC	Short Duty Cycle
LDC	Long Duty Cycle
OEM	Outdoor Environment Monitoring

Chapter 1

Introduction

In the past, traditional crop cultivation was widespread across vast landscapes and remains today essential for the development of human civilization. Crop cultivation takes a tremendous amount of hard work and continuous care and attention. There were many problems involved in traditional crop cultivation: the plant's growth and development depended entirely on the environment and the weather; plants had no protection against pests and diseases. However, the development of modern crop cultivation occurred with the advent of commercial greenhouses [1][2]. A greenhouse allows commercial farmers to grow crops where the changing weather would generally create an unfavourable environment for growing plants. The production of crop plants in greenhouses does not depend on geographic location or even the time of year. A greenhouse gives plants optimum conditions for healthy, vigorous growth. The quality and yield of a crop depend solely on the quality of the environment in the greenhouse.

Given that this environment could negatively impact the plants' growth and yield, it is imperative to have continuous monitoring and control of specific environmental factors to produce maximum crop yield. Temperature, humidity, pressure, wind, and dew point are the most common environmental factors that growers monitor closely. However, these factors alone do not give the grower the complete picture of the condition of the greenhouse ecosystem. Commercial greenhouses are huge and have large outer surfaces where changing weather can vary significantly between different areas inside the greenhouse. It

is common to use sensors and actuators in greenhouses. However, installation requires extensive wiring and maintenance, making the system complex and expensive. Several of the latest wireless and artificial intelligence technologies are promising to contribute to increasing the production of a greenhouse by solving its problems and limitations. Among these technologies are Wireless Sensor Networks (WSNs), Wireless Visual Sensor Networks (WVSNs), intelligent learning (machine and deep learning), and image processing techniques. WSN and WVSN are possible solutions to problems faced by operators of commercial greenhouses. Both networks can operate within a wide range of environments. They are inexpensive, small, require reasonable power to operate, offer flexibility, and exhibit distributed intelligence. The sensing devices in a WSN are called sensor nodes[1] and in a WVSN they are called camera sensor nodes[2]. They used to capture images and sense specific properties of the surrounding environment, including physical and chemical properties, and transmit the sensed data to a central unit called the Base Station (BS) either periodically or on-demand. According to different application requirements, the networks may consist of just a few or as many as thousands of nodes operating collaboratively and coherently for a few days or several years to fulfill a specific task [3]. However, at the same time, excessive use of the nodes will increase energy consumption, consequently decreasing the network's lifetime. This is especially true in applications that need continuous monitoring and observation inside large, and occluded environments like a greenhouse.

[1] sensor node, wireless sensor node, or sensor are terms being used interchangeably in this thesis.
[2] camera sensor node, wireless camera, camera node, or camera sensor are terms being used interchangeably in this thesis.

An intelligent solution comprising machine learning and deep learning has become the most promising and significant technology development in recent years. Machine learning is capable of automatically learning without human interaction and without being programmed. Machine and deep learning extract knowledge from the data that it is given. This knowledge that has been learned can be used to make decisions and generate predictions. Machine and deep learning techniques can be found in WSNs and WVSNs, depending on the labelled data's availability during training.

Merging WSN, WVSN, machine learning, deep learning, with image processing, will help to overcome the problems mentioned earlier, investigate efficient network deployment plans to address the many challenges that have not been addressed and fill the literature gap in this research area.

WSN and WVSN deployment challenges and the motivations behind this research are presented in the following section. Significant research contributions followed by an overview of this thesis and an outline of the remaining contents of this thesis are given in the next sections.

1.1 Motivations and Objectives

Greenhouse crops are susceptible to fluctuations in environmental factors. These factors directly impact plant growth, and poor conditions can spread pests and diseases, ruining an entire crop in the greenhouse. Deploying large scale WSNs in greenhouses becomes necessary to control and monitor the environmental parameters. However, the number of data transmissions between sensor nodes and the BS in such environments increase

significantly along with the network size, promoting data congestion, and a high sensory data loss rate [4-6]. Excessive use of sensor nodes will consume more power, and sensors not operating collaboratively and coherently will shorten the network's lifetime. A short lifetime sensor node will cause instability during the crop life cycle inside the greenhouse. Therefore, there is a crucial need to develop an intelligent solution to predict the weather conditions that affect the greenhouse ahead of time. This solution will help in controlling and stabilizing the microclimate inside the greenhouse. This solution requires historical data and a high degree of accuracy before it can be deployed in a commercial greenhouse. Also, issues related to sensor node's energy consumption and network lifetime must be considered for efficient performance in a greenhouse application. Many investigations looked at changing the duty cycle of the sensors to decrease energy consumption [7], even though reducing the consumed energy for a long duty cycle would result in a considerable amount of unreported data in the network. Therefore, in any proposed intelligent solutions for predicting operational modes of a sensor, it is essential to consider the duty cycle. This will help decrease the wake-up time, which will reduce energy consumption and prolong the lifetime of the network.

Deploying a WSN alone is not enough to overcome all the issues mentioned earlier. There is certainly potential to use camera sensors of a WVSN as a complement to a WSN in monitoring crop growth. Camera sensors are capable of recognizing fungus, pests, and diseases that impact the growth of the plant. If the camera sensor detects increasing fungus on the plant, an alert message that the humidity level is too high inside the greenhouse could be sent. However, the placement of a camera sensor in a greenhouse is challenging.

The placement camera nodes largely influence the operations and performance of WVSN, as camera nodes must be able to observe events of interest and transmit the data to the BS. Finding the best location for camera sensors to cover a large area using the minimum number is an open research problem. The images taken by the camera sensor must be of high quality and high resolution. Thus, the deployment strategy of a WVSN can be modelled as an optimization problem. The problem should aim at maximizing the area to be covered with the least number of camera sensors. This optimization problem becomes challenging because there should be no overlapping field of view between camera sensors. Nonetheless, the camera sensor consumes more energy than any other sensor node, transmits huge amounts of data, and needs a large bandwidth. Thus, the intelligent solution, which predicts sensor node operation modes, can predict the duty cycle of the camera sensor to control the wake-up and sleep modes and hence, decrease the energy consumed during the monitoring time and reduce unnecessary data transmission.

Images from good quality camera sensors are affected by noises and may have unwanted objects in the image, which needs to be cleaned out to be used for recognition processes. With no human interaction, these images need to identify first if the plant has fungus or not. Naturally, the greenhouse is a fully occluded and cluttered environment in which it may be difficult to distinguish and recognize an unhealthy plant. Moreover, the light intensity varies from one area of the greenhouse to another, depending on the direction of the camera when taking images and ambient light through the greenhouse covering. All these issues can affect the image. Combining machine learning with image processing

algorithms can enhance the output images and can program the sensor to recognize the unhealthy plant and locate the fungus on that plant.

Motived by the aforementioned challenges, in this research, we aim to achieve the following objectives:

- Monitoring crop growth in the whole greenhouse area.
- Early detecting of the existence of powdery mildew in a highly occluded and cluttered greenhouse.
- Automating greenhouse operations for reducing the cost and decreasing human interaction.
- Predicting the greenhouse environmental factors to monitor and stabilize the microclimate inside the greenhouse.

1.2 Thesis Contributions

The main contributions of this thesis are as follows:

1. Finding the optimum placement for camera sensors in a commercial greenhouse. To this end, we define an objective function that aims to maximize the covered area and the quality of the image and minimize the number of camera sensors. Since this problem is NP-hard. we resorted to using two sub-optimization problems.
2. We present an automated fungus detection system using WVSN, image processing techniques, and the Hough Forest machine learning algorithm to distinguish between healthy and unhealthy plants and identify the area on the plant that has the fungus with a high degree of accuracy.

3. We propose an intelligent prediction model based on Long Short-Term Memory (LSTM) to control and stabilize the environmental condition in a greenhouse for optimal crop production.
4. We devise a novel approach, based on our proposed intelligent prediction model, to predict the sensor nodes' operational modes (wake-up and sleep) through their duty cycles to decrease the energy consumed and prolong the network lifetime.

1.3 Thesis Overview

This thesis introduces a novel framework that can be applied in a smart commercial greenhouse. The framework includes WSN, WVSN, image processing, machine learning, and deep learning. The framework comprises state of the art approaches on the placement of the camera sensors with maximum image resolution and maximum area covered with a minimal number of cameras to monitor plant growth in commercial greenhouses using WSNs and WVSNs. Lastly, the framework uses state of the art prediction models to operate the sensor node efficiently.

Three phases of research will be described in this thesis, as shown in Figure 1.1. The first phase studies problems related to the deployment of WVSN in the greenhouse. Our aim in this phase is to find the optimal placement for camera sensors, minimize the number of sensors, maximize the area covered, and have better image quality. The second phase involves image processing with machine learning algorithms for dealing with images taken from the camera sensors to recognize the healthy and unhealthy plant. Many image

processing techniques are applied to these images, and then the Hough Forest machine learning algorithm is used for object recognition and segmentation. The Hough Forest was chosen as the detection algorithm for its robustness to occlusion and accuracy. The third phase uses a deep learning LSTM prediction model to predict and control the microclimate inside the greenhouse and communicate with camera sensors to improve monitoring the plants. In this phase, the prediction model will control the duty cycle of the wireless sensor node to achieve less power consumption and, thus, prolong the network lifetime, which results in a good quantity and quality crop production.

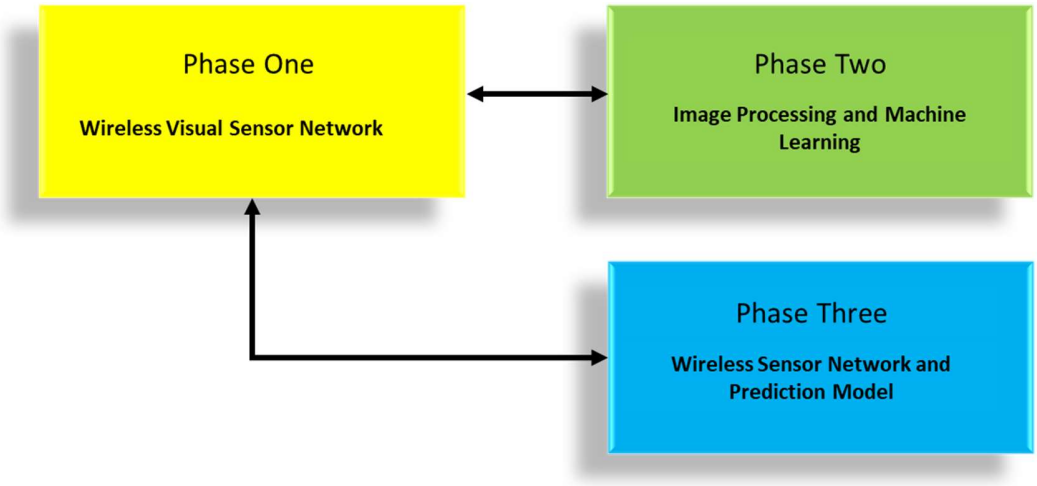


Figure 1.1: Phases of Research

The interaction of the operation of the three phases of the proposed framework can be briefly explained as follows. Initially, during phase one, the camera sensor, placed in the optimum location, will take pictures of the plants and send them to the central servers at the BS for phase two. The images will go through numerous processes to remove noise and

be ready for review. If any anomaly is detected in phase two, the BS sends a message to an actuator to spray the appropriate pesticide. At the same time, sensors send readings of environmental factors. If any measured factor value goes beyond a threshold that is harmful to the crops, then the sensor node will send three messages. One message will be sent to the actuator to perform an action such as open greenhouse windows if too hot, reduce watering if too damp, or water plants if too dry. The second message will be sent to the BS for updating the prediction model and update the duty cycle of the sensor node. The last message will be sent to the camera sensor within the closing proximity of that actuator to take pictures of the crops. Then, those images are sent by WVSN to the BS to take appropriate measures, such as checking the plant's health condition. The full framework is shown in Figure 1.2.

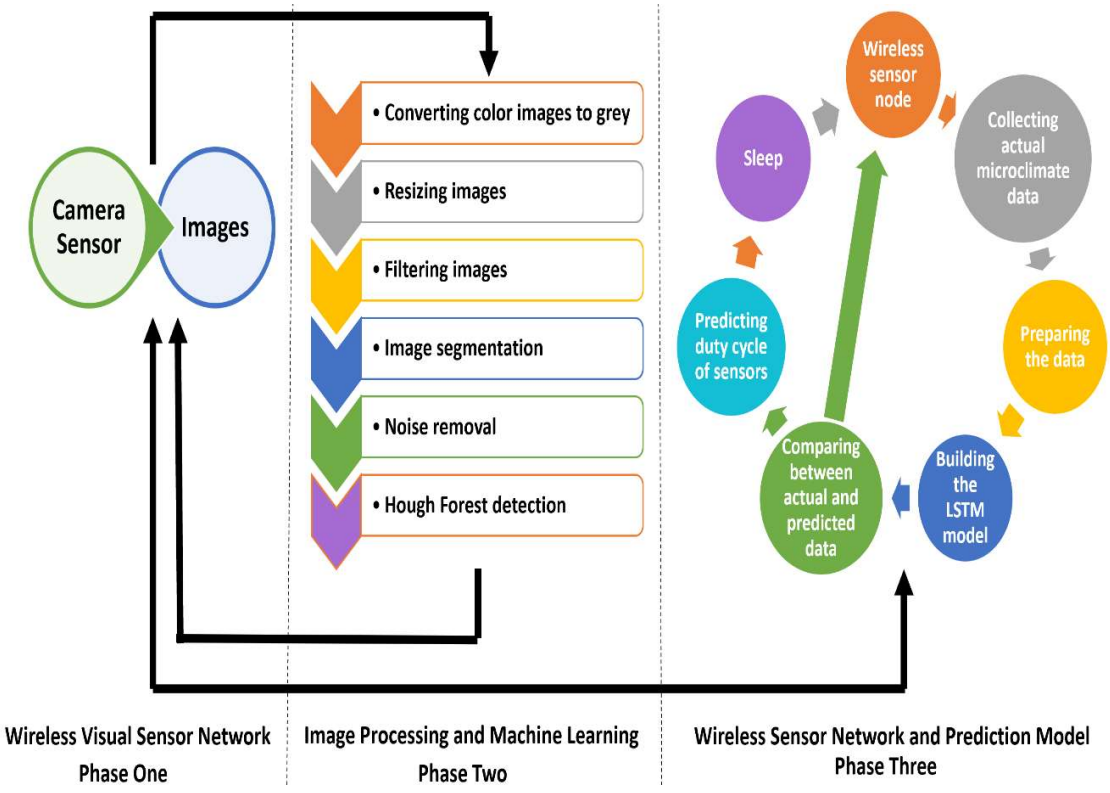


Figure 1.2: Full Framework Interaction

1.4 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 highlights the relevant background literature. Chapter 3 presents our proposed model for the placement of the camera sensors in the greenhouse and our solution to the placement optimization problem. Chapter 4 explains our solution to recognizing fungus and diseases using image processing and machine learning algorithms on images taken from camera sensors. Chapter 5 explains our proposed prediction model based on LSTM for predicting and controlling the microclimate inside the greenhouse and investigate the effect in predicting the duty cycle of the sensors in terms of energy consumption and network lifetime. Lastly, Chapter 6 concludes the thesis and provides directions for future work.

Chapter 2

Background and Overview

In this chapter, we present the relevant background material describing technologies used in this thesis, including wireless sensor networks, wireless visual sensors network, image processing, machine learning, and deep learning.

2.1 Wireless Sensor Networks

A Wireless Sensor Network (WSN) consists of groups of many small devices called sensors, usually distributed indoors or outdoors over a specified area. Each sensor has built-in a communication unit to send and receive data from other connected sensors, a sensing unit for data acquisition, a battery and a microcontroller that will process local data. All sensors have different bandwidths, communication ranges, and can communicate in a single hop or multi hops. The location and placement of these sensors can be mapped or deployed randomly. The sensor's function is to sense and collect environmental data, then send that data to the Base Station (BS) for processing [8]. Typical WSN is shown in Figure 2.1.

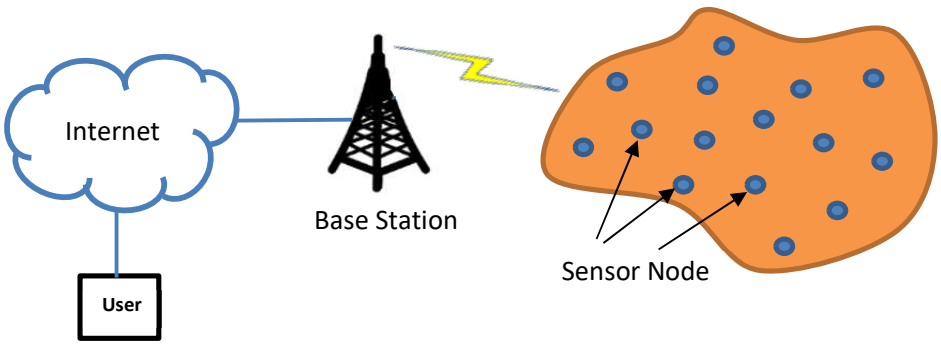


Figure 2.1: Typical WSN

WSNs have been used in many applications, such as: monitoring the environment, healthcare monitoring, target tracking, and industrial process monitoring. These types of applications require the sensors to be active for a long time. Thus, an excessive amount of the sensor's battery will be used, and the battery will drain within a few days. There are many challenges associated with WSNs, including the network's lifetime, limited amount of energy, data loss, overhead costs, latency, process time, and scalability. These challenges are application dependent on WSN deployment, which can be explained as follow:

- Harsh environment: Sensor nodes can be deployed in different environments depending on the application. High humidity, dirt, dust, and corrosive environments are all considered harsh conditions that can reduce their performance and give inaccurate information [9].
- Self-Management: Usually, there is a large number of sensor nodes in a network. A failure in any of the nodes or any additional sensor node will affect the entire network. Sensor networks should adapt to any changes in connectivity [10].
- Redundant Data: A sensor node usually collects data and sends it to the BS. However, when data is sent from many nodes simultaneously, this can cause redundancy of data, which wastes energy [11].
- Real-Time Operation: Many applications required to receive data without delay for reliability and security. However, in a poor network, old data can be mixed with current data resulting in incorrect data being sent to the BS [12].

- Coverage and Connectivity: Network coverage should assure that all the area to be monitored is fully covered not just part of the area. At the same time, network connectivity assures that all nodes should be able to communicate back and forth between each other [13].
- Energy: this is one of the main concerns with sensor nodes. Energy is consumed during node operation and in all processing, transmitting, sensing, and data collecting tasks. However, sensors have limited battery power, the batteries are small and sometimes difficult to change because for example the deployment is over a large area, or in a hazardous environment. Continuous monitoring or tracking will drain the battery very quickly [14].

  With excessive use of sensors, comes an important issue related to sensor lifetime which depends on how much energy is needed and consumed while the sensor is functioning. Sensor nodes must optimize the energy usage for the network's lifetime. One method of lowering or minimizing sensor energy consumption is through controlling its duty cycle activities.

A duty cycle is a period that the sensor is active. Duty cycles are a favorable approach to saving energy in WSNs [15]. Sensors management activities can be classified into two groups: sleep/wakeup protocol and topology control protocol. This classification is based on the topology the sensors were implemented. For the first type (sleep/wake-up protocol), each node works individually. Sensors can be in active (wake-up) mode and then switch to sleep mode when there is no data to send. This type can alternate between wakeup and sleep modes. For the second type (topology control protocol), a select minimum subset of

sensor nodes can be active for network connectivity and the remaining sensor nodes will be in sleep mode to save power and lower the amount of consumed energy by the network.

2.2 Wireless Visual Sensor Networks

A typical Wireless Visual Sensor Network (WVSN), shown in Figure 2.2, consists of a group of sensor cameras that can process images from different vantage points that can create a more useful image that contains more information. Usually, the sensor cameras can process images locally, communicate between other cameras and the BS that processes the collected images from each camera.

The difference between WVSNs and other types of WSNs is the type of data being collected. Most sensors collect measurements as a one-dimensional data signal. However, a sensor camera collects a two-dimensional set of data. Thus, the additional dimensionality of data adds a level of complexity. Another difference is when a sensor collects data, it uses the sensing range, but when a sensor camera collects data, it uses the field of view. These differences are illustrated in Figure 2.3.

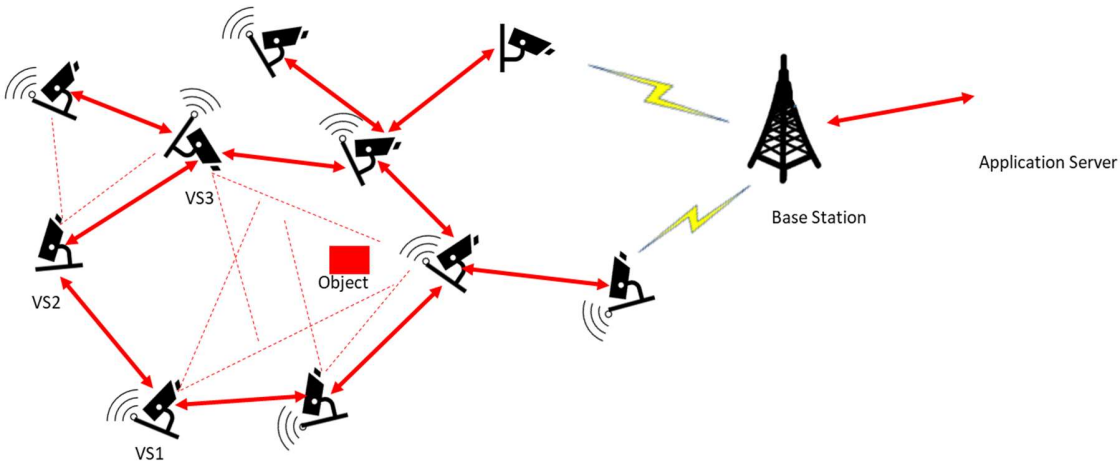


Figure 2.2: Typical WVSN

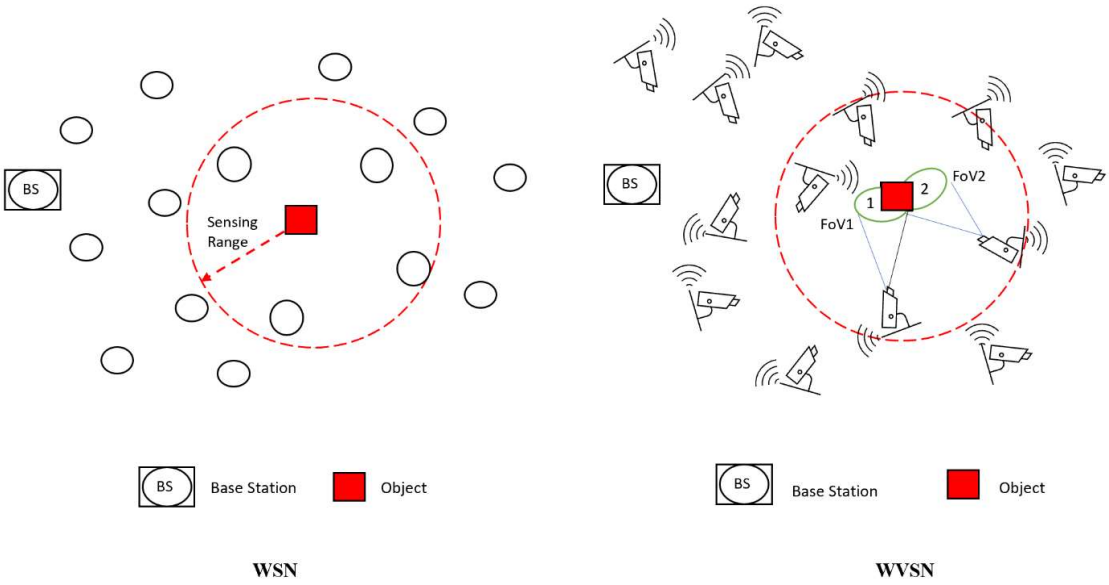


Figure 2.3: Difference Between WSN and WVSN

Listed below are the characteristics of a sensor camera [16]:

- Resource Requirements: The battery in the camera has limited energy for sensing, processing, and transmitting. Sensor cameras generate large amounts of data that need large bandwidth.
- Local Processing: The processes that happen locally in a sensor camera reduce the amount of data that needs to be communicated in the network. All processes, from simple image processing to a complex vision algorithm, depend on the application and the level of connection that the camera node can provide.
- Real-time Performance: Usually, applications need real-time data from the camera sensors, setting a limitation on the length of the delay and the amount of time that data needs to be transmitted from the node to the BS.

- Precise Location and Orientation: Most image processing in WVSNs requires information about the location of the camera node and its orientation.
- Time Synchronization: Time is very important in many applications. Applications that involve multiple cameras depend on the synchronized camera's images for object localization. The information from the image would be meaningless without the correct time that the image was taken.
- Data Storage: Usually camera sensors create very large amounts of data that need to be stored. Monitoring is an application that uses cameras. Therefore, it has more data to capture and transmit, which means cameras will consume more energy. In this case, the camera sensor should be equipped with the capacity to store large data.

2.2.1 Sensor Camera

In addition to the main function of a sensor camera which is captures images, the camera can extract information from the image using machine vision technology. The sensor camera components are an image sensor, image digitization circuitry, image memory, a processor, a communication interface, I/O lines, a lens holder, and a built-in illumination device. Many applications use sensor cameras such as monitoring environment (animal habitats, hazardous area, building, street, train station, airport), smart meeting rooms (visual studios), smart home (elderly care, kindergarten), and surveillance (parking lots, remote area, traffic, public places) [17].

2.3 Image Processing

Image processing is a method used to get an enhanced image or to extract information from the image. It is a type of signal processing where the input is an image, and output may be a clearer image or features associated with that image. Currently, image processing is among the fastest-growing technologies. The basic image processing techniques [18], given in Figure 2.4, are:

Image Acquisition: is capturing an image using a suitable camera. There are different cameras for different applications. In Chapter 4, we used a camera that is sensitive to the visual spectrum.

Grey Level Image: indicates the brightness of a pixel. In greyscale or color image, a pixel value can range between 0 and 255. The minimum grey level is 0. It carries how white and black pixel intensity in the image. To convert any color image to grayscale is by taking the average of the three colors of RGB image. So, adding R, G, B, and then divide by 3. Usually, it is easier to deal with a gray level image than a color image because grayscale is only one channel containing brightness information without any visible color, comparing to three channels (RGB) in a color image, which needs more complicated processing steps. Also, the gray level image size is smaller than the color image size, making it faster for processing.

Noise Removal: is a process to remove unwanted noises using the low-pass filter or any smoothing operation. One of the methods is convolving the image with a Gaussian mask, which will brighten the value of the pixel to be closer to other values of its neighbors.

Edge Detection: one of the image processing techniques for finding the boundaries of an object inside the image and detecting the discontinuities in brightness in the images.

Image Enhancement: is a process for adjusting digital images so the outcome can be more appropriate for a specific application. Such as sharpen the edges, boundaries, or contrast.

Image Segmentation: is a process for portioning digital images into multiple segments. To simplify or change the representation of an image into something more meaningful and easier to analyze.

Captured images using camera sensors require image processing methods to extract specific information from the image. For more detail, a scene can be recorded using image processing from camera sensors for tracking and monitoring applications [18].

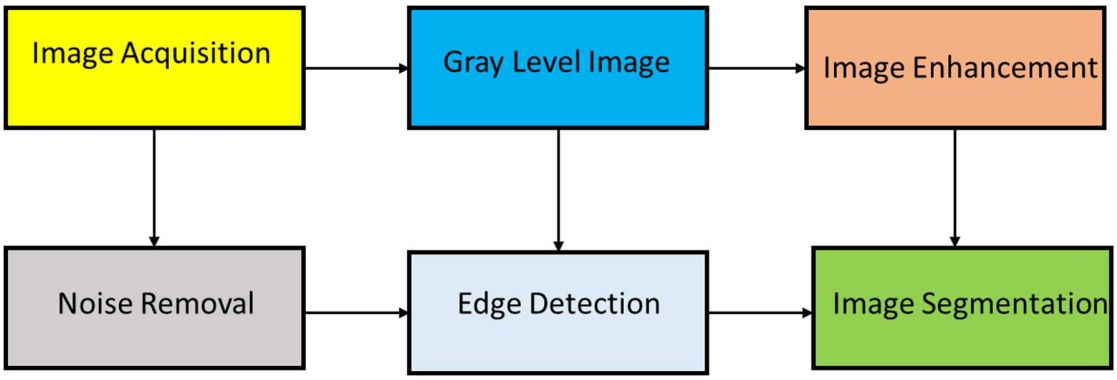


Figure 2.4: Image Processing Techniques

2.4 Machine Learning

Machine learning has become the most promising and significant technology development in recent years. It is capable of automatically learning without human interaction and

without being programmed. Machine learning extracts knowledge from the data it is given. This knowledge that has been learned can be used to make decisions and generate predictions. Machine learning techniques can be found in many types of WSNs and WVSNs, depending on the labelled data's availability during training.

Machine learning approaches can be classified into four categories:

- Supervised machine learning: learn from example data associated with labels or targeted responses, then generate a label to new unseen data.
- Unsupervised machine learning: learn from plain examples without any associated labels. The focus of unsupervised learning is to study the data structure and restructure it into a more meaningful structure.
- Semi-supervised machine learning occurs, when part of the available data comes with labels and the other part does not come with labels.
- Reinforcement learning occurs, when an agent is responsible for taking action in an environment with the purpose to maximize return rewards.

Supervised learning is the most popular category of machine learning due to the abundance of data and storage available nowadays. Supervised learning depends on the availability of the labelled data; that is, there is a set on input data (X) associated with label data (Y). Supervised learning learns a mapping function from input data to output label according to the following equation:

$$F(X) = Y \tag{2.1}$$

where X can be any set of data (features), which are believed to capture the underlying relationship between the input data and output. Y is the label and it can be a discrete value as in classification supervised learning, or real value as in regression supervised learning.

2.4.1 Supervised Learning Algorithms

This subsection introduces the six machine learning algorithms that are used either for classification or regression. Among these algorithms, Random Forest algorithm is used in Chapter 4.

2.4.1.1 Decision Tree

Decision Tree (DT) can be used in regression and classification problems. It is a tree-like graph. DT consists of nodes and branches. The node represents the feature of the dataset. The branches represent the possible value of the feature to trace the tree. The root node represents the whole dataset and the metric like the mean square error when solving the regression problem, or it can be the information gain when solving a classification problem. This is used to determine the split in the data. DT is easy to interrupt but it can be large, and pruning should be done to avoid overfitting [19].

2.4.1.2 K-Nearest Neighbour

K-Nearest Neighbour (KNN) algorithm is based on the distribution free assumption of data. KNN uses K neighbours for a point in the features space to predict the label of this point. For a regression problem, KNN is the average of the K neighbouring values, which are the distances between every training example to the point. Then, K-nearest neighbour is used

to calculate the average, which is the label for this point. For a classification problem, the class of the point is assigned based on the majority class of the K neighbour. KNN is simple and easy to implement. However, KNN can be slow and choosing the value of K is not easy in some applications [20].

2.4.1.3 Linear Regression

In linear regression a real value y is predicted based on a set of variables $x_1, x_2, x_3, \ldots x_n$, it assumes a linear relationship exists between X and Y.

$$Y = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + .. + \beta_n * x_n \quad (2.2)$$

The weights $\beta_0 \ldots \beta_n$ in Equation 2.2 are learned during the training process of a linear regression model such that the prediction error is minimized. Regularization terms can be added to the linear regression equation to ensure that the model is not overfitting to the training data and can perform well on new unseen data.

2.4.1.4 Support Vector Machine

Support Vector Machine (SVM) splits the data to maximize the margin between the different classes, where the closest point to the boundary is called support vectors. The decision boundary or the kernel may be linear or non-linear depending on the complexity of the problem. SVM works effectively in high-dimensional data and complex non-linear data points, although it may be computationally expensive [21].

2.4.1.5 Naive Bayes

Naive Bayes (NB) is based on the Bayesian theorem. NB performs well on most of the classification problems despite its simplicity. NB assumes independence between the different input features and represents the relationship between the label and the input feature in conditional probability relationships. NB may also be used for regression by modelling the probability distribution of the target label with kernel density estimators [19].

2.4.1.6 Random Forest

Random Forest (RF) is like DT in which the decision is based on a set of many trees, not only one as in DT. In a regression problem, the new input will assign a real value based on the mean of all trees. In a classification problem, a class based on the mode of prediction of all trees in the forest is assigned to a new input point. RF implementation is useful on large datasets, but they are prone to overfitting because of the large number of tuning parameters that need to be chosen adequately. The process handles randomness by selecting a random sample of training data to build each tree. At each node, there will be many split points created by choosing each data point's feature value; usually, the one with the highest information gain will be chosen, to generate two child nodes. To speed up the process, a small random subset of features allows the tree to quickly calculate the split point and give the highest information gain. This process will repeat many times until the target is detected, and time and depth are achieved. The randomness makes the trees uncorrelated to each other, allowing the output of the forest to have low bias and low variance, as shown in Algorithm 2.1 [22].

Algorithm 2.1: Random Forest

1. Build forest of T trees, each denoted by the letter t . For each tree t:
   (a) Take a sample n of the training data set N (with replacement);$n << N$
   (b) Repeat the following steps for each node until a target certainty, time, or depth is achieved
      i. Select a subset of f features from the complete feature set F.
      ii. Create split points on each axis by choosing each data point's values .
      iii. Choose a split point that gives the maximum information gain.
      iv. Create two child nodes based on the split.
   (c) Do not prune the tree after it is grown.
2. Classify a new unknown data point by processing it through each tree
3. Calculate the mean or mode of the tree outputs and return the final classification.

2.4.2 Hough Transform

Hough Transform is a technique to extract features in the image. It can identify only certain defined shapes such as lines, circles, and ellipse. The purpose of Hough Transform is grouping edge points into the object candidate performing a voting procedure over a set of a parameterized image object. Since there are many objects that have an arbitrary shape, there is a need for a technique to find an object in an image. Generalized Hough Transform has been used to identify the location of any arbitrary shape.

In Chapter 4, we used Generalized Hough Transform due to the imperfection of edge detector or image data. There can be missing points or pixels on the desired line, or

sometimes there are deviations between the real line and the noisy edge points as they are obtained from the edge detector. Thus, it is better not to depend on the edge detector and depend on the Hough Transform.

2.4.3 Generalized Hough Transform

The generalized Hough transform is a method for estimating the parameters of a shape from its boundary points. It extends the classic version for simple shapes like lines and circles by parameterizing in an R-table, with no need for any analytical form. Any shape is specified by the set of boundary points. An R-table must be created for each different shape. The gradient for each point and the length and the orientation of the vector with reference (centre of gravity) will be generated and will characterize the shape. By increasing the space dimensionality of Hough and adding new factors, the generalized Hough transform will be able to recognize any changes in scale and orientation. Algorithm 2.2 [23] shows the generalized Hough transform process.

There are some limitations in using the generalized Hough transform in some applications. Hough transform works well with a clear target shape that can be generated from an edge detection algorithm which is difficult to guarantee in images of greenhouse plants due to significant clutter. The additional problem is that the generalized Hough transform can only work with shapes that have a very limited variance; this is not the case in greenhouses where plant shapes have large variability.

Algorithm 2.2: Generalized Hough Transform

1: Find the edges in the prototype image.
2: Choose a reference point (x_c, y_c) ; e.g., center of gravity.
3: Compute the distance, r to the reference point and the gradient orientation ϕ of each pixel using a Sobel operator.
4: Using the following equations store the reference point as a function of ϕ, creating the R-table.
5: $r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$
6: $\beta = \arctan(\frac{y-y_c}{x-x_c})$

ϕ_1	(r_1^1, β_1^1)			
ϕ_2	(r_1^2, β_1^2)	(r_2^2, β_2^2)	...	(r_m^2, β_m^2)
$\vdots$	$\vdots$			
ϕ_n	(r_1^n, β_1^n)			(r_m^n, β_m^n)

7: To detect the shape parametrized by the R-table, create a 4 dimensional Hough space$H$ $([x_c], [y_c], [\theta], [s])$where θ is the orientation and$s$ is the scaling factor, and initialize the cells to zero.
8: For each edge point(x, y) in the search image computeϕ.
9: Look up ϕ in the R-table and convert each(r, β) to a potential(x_c, y_c) using the following formulas.
10: $x_c = x + r\cos(\beta)$
11: $y_c = y + r\sin(\beta)$
12: For each value of s' andθ' in their ranges rotate and scale(x_c, y_c) to form$(x_c, y_c)'$ and increment H $([x_c'], [y_c'], [\theta'], [s'])$
13: Examine the Hough space for local maximums, each of which represent a potential object

2.4.4 Hough Forest

Hough Forest is a combination of the generalized Hough transform and Random Forest. Algorithm 2.3 [24] presents the Hough Forest algorithm. The Hough Forest was chosen as the detection algorithm because of several of its unique characteristics: (1) It can robustly deal with stem occlusion; stems partially covered by a leaf can still be detected, (2) Its resistance to noise in training data allows for limited dataset preparation requirements, (3)

Its classification performance is excellent, and (4) It is inherently parallel allowing it to scale in performance to meet time requirements. Further explanation is given in Chapter 4.

2.5 Deep Neural Networks

Artificial Neural Network (ANN) comes from developing intelligent tasks mimicking the tasks performed by the human brain. ANN is a data modelling tool connecting a strong, complex relation between input/output. ANN functions like a human brain in the following ways: neural gets knowledge through learning, then this information will be stored in interneuron connection which is called as synaptic weights. One of the neural functions has the capability to directly identify linear and non-linear relationships from the data being modelled. ANN model is a structure that can create a mapping or relationship from the dataset. The network model is adjusted and then trained from a collection of datasets called the training set. After training, the ANN will be able to predict, estimate, classify or simulate tasks on a new dataset from the same data sources.

Deep learning is a subset of machine learning. It is a neural network with multi-layers and these layers have multi hidden layers. The purpose of adding hidden layers between the input and the output is to improve the accuracy and the performance of the neural. Each layer in the deep network is responsible for extracting features from the input data automatically. As the deep network grows, the more complex the feature and the more abstraction must be extracted [25]. Thus, improving the accuracy and avoiding the excessive processing time. In contrast, when the neural is shallow, less feature to extract.

Thus, less accuracy. The following three subsections present an overview of the most common examples of deep neural networks.

Algorithm 2.3: Hough Forest

Training the Hough Forest

1. Annotate an image set with a bounding box and a centroid for each instance of the target object. All target objects should be scaled to be similar in size.
2. For T trees in forest F:
3. Randomly select N images from a set of images S for training, where $S >> N$
4. Choose P 16x16 pixel patches from N images at random and describe each using appearance information$I$, a class label c, and the displacement vector$d$. From the object centroid $-P = (I, c, d)$. Background patches (outside a bounding box) are not given a displacement vector.
   (a) Appearance Information can include color intensity values, derivative filter responses, histograms of oriented gradients, etc
5. Train the decision tree:
   (a) Feed in a subset of patches to a node and pick the optimal binary test from set of randomly generated binary tests. An optimal test will split the data with minimum uncertainty as defined as:
      i. $U_1(A) = |A|Entropy(c_i)$
      ii. $U_2(A) = \sum_{i:c_i=1}(d_i - d_A)^2$
      iii. where A is a set of patches, c is the impurity of the class label (0-1),and d_A is the average displacement vector to the center of the object.
   (b) split the subset of patches into two child nodes and repeat a)
   (c) if the node is at the tree's maximum depth or the amount of patches left is less than the limit, classify the node as a leaf
      i. record the proportion of patches from the target class contained in the leaf node
      ii. record the displacement vectors for each patch that contributed to the leaf node

Detecting object classes

1. In the test image, use each 16x16 patch as input into each tree of the forest
2. Take the sum up the average outputs of the leaf nodes into the Hough space
3. Examine the Hough space for local maximums, each of which represent a potential object
4. Variable scales and rotations are handled by increasing the dimensionality of the Hough space from 2D to 4D

2.5.1 Multi-Layer Perceptron

Multi-Layer Perceptron (MLP) is the most common form of neural networks. It is a feed-forward neural network. The architecture of MLP is shown in Figure 2.5. It consists of a cascade of three different layers: input layers, hidden layers, and output layer. The number of neurons in the input or output layer depends on the problem that MLP is trying to solve. Also, the number of hidden layers and the number of neurons on them depends on the designer and how the parameters can tune during the MLP training process. Many hidden layers make the network deeper. MLP is called a feed-forward network because the neurons in one layer are not connected to each other, but in the next layer, the neurons are connected to other neurons. In each connection there is a weight value. These weights have been learned during the training process of MLP.

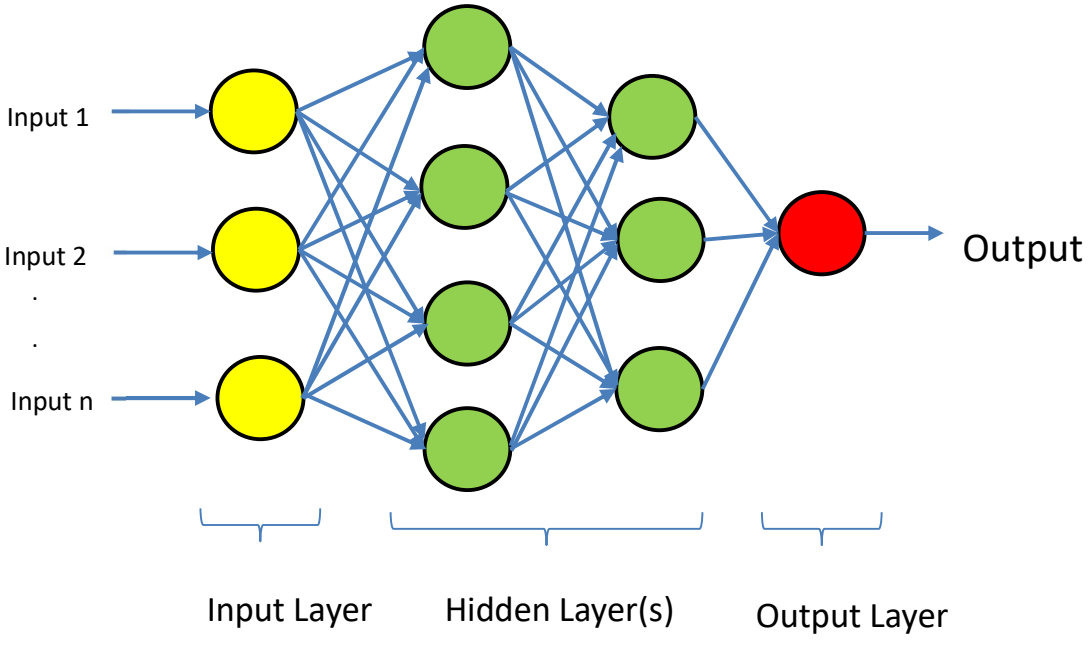


Figure 2.5: Multi-Layer Perceptron Architecture

Figure 2.6 shows the basic elements of MLP. A neuron takes n inputs, a weight will be carried in each connection, then each input will be multiplied by its assigned weight and sum-up the results [21]. An activation function F is applied to the sum to generate the output of the neuron. The equation below summarizes the neuron functionality:

$$Output = f(\sum_i^n(input_i * W_i)) \quad (2.3)$$

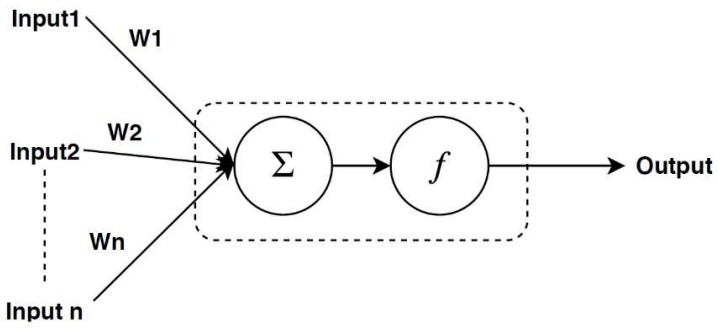


Figure 2.6: Multi-Layer Perceptron

Both regression and classification applications can use MLP depending on the neuron's activation function in the output layer. For binary classification, sigmoid activation function will be used for neuron in the output layer, while regression rectified linear unit (ReLU) activation function will be used in the output layer.

2.5.2 Recurrent Neural Network

A drawback of the ANN is that it cannot capture the sequential information from the input dataset, which is required in many applications such as speech recognition, text generation, voice semantic recognition to name a few. Recurrent Neural Network (RNN) is used to overcome this drawback. RNN has a recurrent relation on the hidden state, as shown in Figure 2.7. This looping constraint ensures the capture of sequential information in the input data, and this is considered as the main difference between RNN and ANN [26].

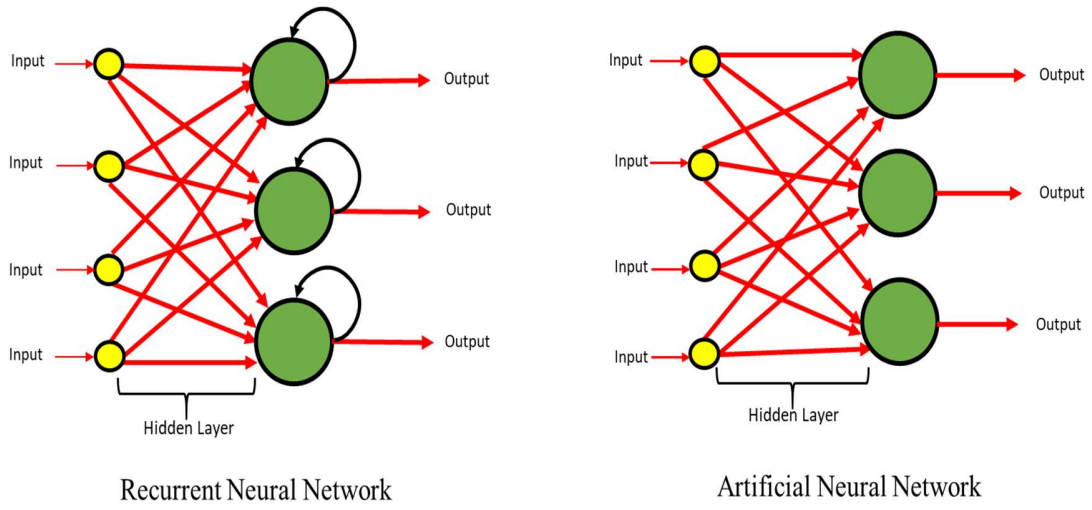


Figure 2.7: Main Difference Between ANN and RNN Architecture

Recurrent Neural Network (RNN) is a type of ANN which works perfectly with time series data, contains loops, and allows the information to be stored within the network. RNNs usually process time series and keep an internal state which summarizes the information in

the connection which was between the nodes that form a direct graph along a temporal sequence [26]. This allows RNN to exhibit temporal dynamic behavior.

RNN steps are:

1. RNN will convert the independent activations into dependent activations. Then assign the same weight and bias on all layers. Thus, reducing the complexity of RNN parameters makes the neural memorize previous output when provided the input of the next layer.
2. Three layers of the same weights and bias merge into one recurring structure (RNN state) as shown in Figure 2.8. Weight and bias 1 denoted as W1 and B1, respectively. Similarly, for the second layer, W2, B2, and the third layer, W3, B3. These layers are separate from each other, which means no memories from the previous output.

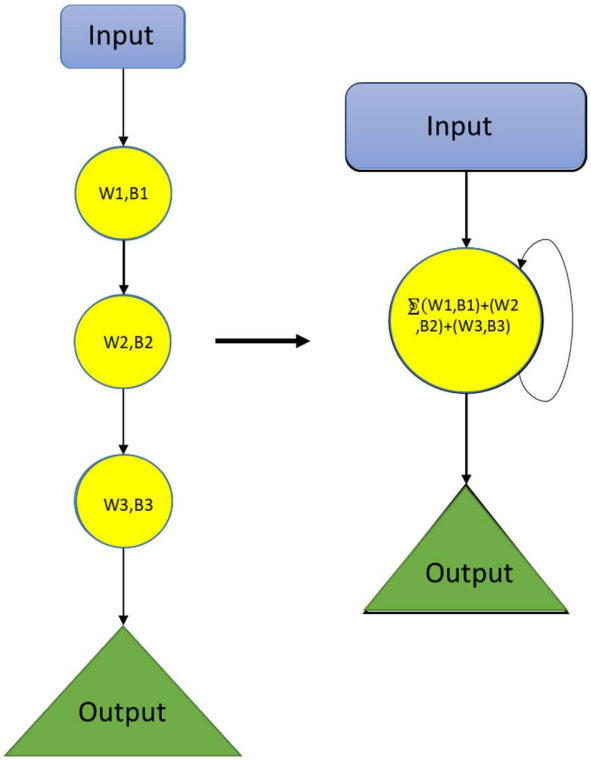


Figure 2.8: RNN State

The following function calculates the RNN state:

$h_t = f\ (\ h_{t-1}\ , x_t)$ (2.4)

where:

$h_t = current\ state$

$h_{t-1} = previous\ state$

$x_t = input\ state$

Apply the activation function:

$h_t = \tanh(\ W_{hh} * \ h_{t-1} + \ W_{xh} * \ x_t)$ (2.5)

where:

$W_{hh} = weight\ at\ recurent\ neuron$

W_{xh}= $weight\ at\ input\ neuron$

To get the output:

$y_t = \ W_{hy} * \ h_t$ (2.6)

2.5.2.1 Long Short-Term Memory

Long Short-Term Memory (LSTM) is an artificial RNN architecture used in the deep learning field of study. LSTM has feedback connections that are different from ANN. Also, LSTM can process all sequences of data not only one single data point. LSTM has a powerful function as RNN in remembering long periods and previous events. Furthermore,

LSTM can efficiently work with scalable processing capabilities (i.e., powerful GPUs) to handle and analyze a massive datasets. This ability alone makes LSTM one of the best commercial artificial intelligent achievements to date [27]. LSTM network applications include processing, classification, and prediction based on time-series data. LSTM has been used for many tasks such as unsegmented, connected handwriting recognition, speech recognition and anomaly detection in network traffic. Some applications are affected by lags of unknown duration between important events in a time series. LSTM can be deployed for this type of application to deal with the exploding and vanishing gradient problems and learn tasks that require memories of events that happened thousands or even millions of discrete time steps earlier.

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell, as shown in Figure 2.9.

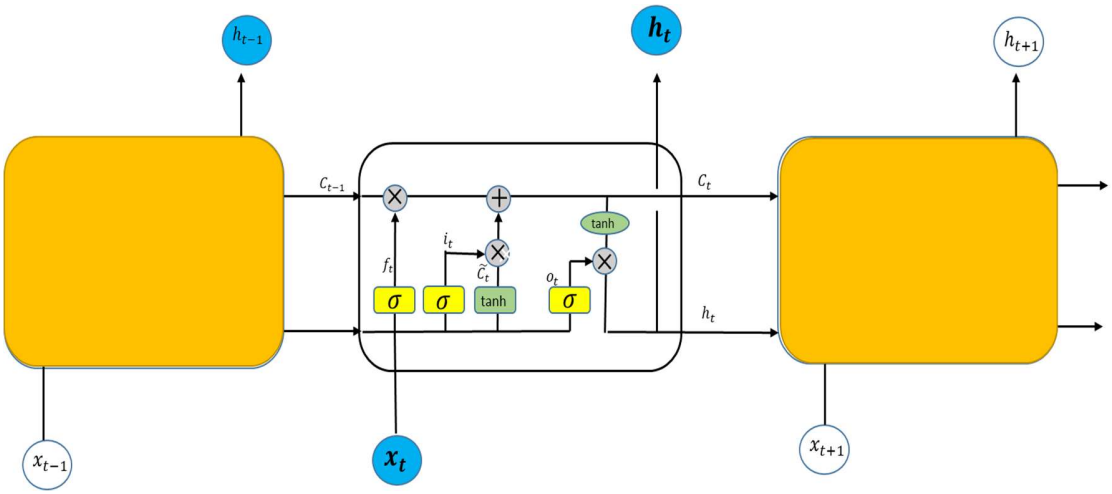


Figure 2.9: LSTM Architecture

The processes in how the LSTM model prediction works is explained as follows. Firstly, know which information should be canceled from the cell state. This depends on the sigmoid layer (forget gate layer) which looks at h_{t-1}, x_t and the output will either 0 or 1 in C_{t-1}. 1 represents "completely keep" and 0 represents "completely leave".

$$f_t = \sigma\,(\,W_f\,.[\,h_{t-1}\,,x_t] + \,b_f) \qquad (2.7)$$

Secondly, information that will be stored in the cell state. Sigmoid layer updating value (input gate layer) and tanh layer to create a new vector with new values.

$$i_t = \sigma\,(\,W_i\,.[\,h_{t-1}\,,x_t] + \,b_i) \qquad (2.8)$$

$$\widetilde{C_t} = \tanh(\,W_C\,.[\,h_{t-1}\,,x_t] + \,b_C) \qquad (2.9)$$

Thirdly, updating the cell state by multiplying the old state with f_t.

$$C_t = \,f_t * \,C_{t-1} + \,i_t * \widetilde{C_t} \qquad (2.10)$$

Finally, the output of the cell state will be:

$$o_t = \sigma\,(\,W_o\,\,[\,h_{t-1}\,,x_t] + \,b_o) \qquad (2.11)$$

$$h_t = o_t\, * \tanh C_t \qquad (2.12)$$

Usually, important information is transferred from short-term memory to long-term memory and kept there.

2.6 Environmental Factors

Plant growth in a greenhouse environment, and in general in an agricultural environment, goes through many different stages. Including germination, sprouting, flowering and fruit development. The development of the plant is impacted by its environment, the nutrients in the soil and the suitability of conditions. A greenhouse environment consists of several factors, including temperature, humidity, air pressure, dew point and wind [28]. Each of these factors plays an important role in the plant's growth. Poor environmental conditions can weaken plants, thereby increasing the risk of disease and pest infestation.

2.6.1 Effect of Temperature

Temperature is an essential environmental factor for plant growth as it will affect germination, sprouting, flowering and fruit development. Moreover, when ambient temperature increases the transpiration rate goes up during the growing stage. Temperature can cause changes impacting other environmental factors such as humidity and soil moisture. Therefore, the temperature should be maintained at an optimum level whenever possible [29].

2.6.2 Effect of Humidity

Humidity is an essential factor for plant growth especially when levels are too high or too low, the plant can show signs of distress [30]. If humidity is high, above 80%, for a long time, fungal diseases can appear and spread to neighboring plants. Moreover, the air

becomes saturated with water vapor which ultimately restricts transpiration, affecting the greenhouse crops significantly.

2.6.3 Effect of Air Pressure

Air pressure is the main factor that affects all other factors. Air pressures control the circulation of the atmosphere. Thus, changes in temperature, rainfall, and winds [30].

2.6.4 Effect of Dew Point

At night, when the air cools to the dew point, the condensation happened, and water droplets are created on any cooler surface like the leaves. This moisture stimulates the germination of fungal such as powdery mildew.

2.6.5 Effect of Wind

The wind passes over all sides of the greenhouse and creates different air pressure, which generates a force that can damage the greenhouse. In addition, the wind will decrease the humidity of the air [28], thus, hindering the plant growth.

2.7 Summary

In this chapter, the necessary background material describing wireless sensor networks, wireless visual sensor networks, image processing, machine learning, and deep learning, was presented. The technical differences between WSNs and WVSNs, in addition to their applications, were explained. In this research, we merge WSN and WVSN for the best of monitoring and controlling inside the greenhouse environment. Image processing methods

for enhancing images were described. These methods are combined with the WSN and the WVSN to design an automated system for early plant disease detection. Standard machine learning and deep learning algorithms were described in this chapter. The Hough Forest machine learning algorithm explained, compared, and chosen, in this research, among other machine learning algorithms for recognizing of powdery mildew on leaves of the plants. The LSTM deep learning model was selected to be used in our research compared to other models due to its high degree of accuracy of predicting the greenhouse environmental factors. Lastly, these factors, which affect greenhouse crop production, were explained.

Chapter 3

Optimal Placement of Wireless Sensor Cameras

3.1 Introduction

There is considerable research on how to control the environment of a greenhouse, the temperature, humidity, wind, pressure and dew point by using sensors [31] [32]. Conversely, there is much less research on the early detection of diseases that can damage greenhouse produced crops. The stability of the ideal environment for growing plants is guaranteed by using a Wireless Sensor Network (WSN), to monitor and control the conditions for optimum plant health and growth. Tracking plant growth is the best method for early detection of plant disease and preventing significant crop losses. A Wireless Visual Sensor Network (WVSN) is an efficient technology for monitoring plant growth with the added feature of wireless sensor cameras. The WVSNs are widely used for surveillance and detecting anomalies [33-35] and are poised to be the best solution for early detection of plant anomalies and diseases in greenhouse crop production.

The area inside a greenhouse that needs to be monitored is very large, and it would take an infinite number of images to cover. To improve the performance in terms of storage and processing and reduce the response time of the image processing unit, we should place the WVSN cameras so that there is no overlap of images taken by the cameras. It is also necessary to capture images with high resolution for better processing and analysis. Hence, optimizing the number of sensor cameras will help in:

(i) Optimizing the limited storage space of the sensor camera nodes.

(ii) Decreasing the processing time to then be able to analyse these images quickly and isolate plants showing signs of disease.

(iii) Producing high-quality images for avoiding false detections.

(iv) Minimizing the project cost since WVSN systems can be expensive to install and maintain.

This chapter presents a mathematical formulation and an optimal solution for the best placement of the WVSN camera nodes to cover a large area, produce high-quality images, and avoid overlapping between cameras.

The remainder of this chapter is organized as follows. In Section 3.2, we review the related work. Section 3.3 defines the problem statement and highlights the research contributions. Section 3.4 outlines the assumption and provides problem formulation. Section 3.5 describes the ILP problem formulation. Section 3.6 presents the implementation and numerical results. Section 3.7 includes the performance evaluation, finally, in Section 3.8, is the summary for this chapter.

3.2 Related Works

Having looked at recent literature on the issue of monitoring greenhouse environments, we can see that the deployment of sensor cameras in many different applications has been increasing rapidly over the past decade. Since a sensor camera placement problem is considered an NP problem, the direction of the research finding the optimal camera

placement has been changed from theoretical analysis to practical application, which increases the complexity of the model with realistic assumptions [36]. Several works had an interest in maximizing coverage, such as [37], where authors studied the problem of maximizing coverage on a set of discrete targets by directional sensors that could turn around. This work was aimed to maximize the network lifetime by maximizing the number of covered targets and minimizing the number of sensors activated at any given moment. The authors ensured that a target must be covered by at least one camera (tolerating overlapping images between the sensors) and did not consider optimal camera placement since they assumed that the cameras were placed randomly. In work presented in [38], the authors proposed a heuristic for the maximum coverage of an area when one of the existing cameras breaks down. The proposed algorithm is a decentralized control system that allows the communication between the cameras in other nearby locations to adjust their direction and field of view.

Authors in [39] solved the camera placement problem using dynamic programming to maximize the coverage area and use it in a surveillance application without considering the quality of the images' resolution. In [40], the authors tried to solve the same problem focusing on maximizing the coverage area and minimizing the cost. Authors in [41] used a graph-based approach to cover a larger area with less time. The authors in [42] model the sensor field as points on a grid (coordinates) and present an Integer Linear Programming (ILP) solution for minimizing the number, and therefore the cost, of the sensors it would take to completely cover the area to be monitored, taking into consideration that sensors vary in terms of field of view ranges and price. The authors did not solve the problem of overlapping between cameras. In [43], the authors address the problem of optimally placing

multiple sensor cameras to cover a given area. They modeled the problem using a linear programming approach which determines the minimum number of cameras needed to cover the monitored area. This approach also determined the exact position of each camera. However, their solution does not manage the problem of overlapping between the cameras and image quality. In [44], authors propose a Computational Fluid Dynamics (CFD) solution using wireless sensor camera nodes and image processing to monitor the temperature in a greenhouse when physical measuring instruments are not available. In [45], the authors propose a global greedy search optimization method to look for the camera's optimal placement. However, the proposed method is very long and complex. It must explore all the possible solutions, and it tolerates overlapping between the cameras. In [46-48], the authors used a different approach to find the camera sensor placement. They solved the problem using Particle Filtering (PF), Resampling Particle Swarm Optimization (RPSO). While they achieved good coverage control, their solution did not consider the resolution of the images.

It is worth noting there remains a gap in research for formulate a general problem for WVSN camera deployment management in greenhouses. Such a problem must consider finding the optimal placement for the camera sensors, reducing the number of cameras, increasing the quality of the images, avoiding overlapping views, covering a large area, and guaranteeing there are no images of the same plant from more than one camera. Our proposed optimization problem manages and satisfies all of these requirements.

3.3 Problem Statement and Contributions

The major challenges in WVSNs deployment in greenhouses is the need to cover a large area of plants, and gather high-quality images while managing with the computation, sensing ability, and communication constraints. As a result, careful wireless sensor camera node placement can be a very effective means of optimizing practical and economical solution goals.

In this chapter, we investigate a solution for solving the following problem:

Given a greenhouse with WVSN infrastructure in place, determine the best placement of the visual sensors to capture images of a desired quality, covering the entire greenhouse area using the minimum number of nodes ensuring connectivity between nodes.

To this end, major contributions of this work are listed as follows:

1. Finding the optimum placement for camera sensors in a commercial greenhouse by formulating an optimization problem that will maximize the covered area and minimize the number of camera sensors with the fact that each point must be seen by one and only one camera.
2. Recent papers tried to solve the placement WVSN problem without considering the importance of image quality. We formulate an optimization problem to consider this feature. To this end, we define an objective function that aims to maximize the

covered area and the quality of the image and minimize the number of camera sensors. However, we found that this problem is an NP-complete problem in which we overcome this challenge by solving it as two sub-optimization problems.

3.4 Optimal Placement Camera Quality Problem Formulation

In this section, we present the optimization problem formulation starting with preliminaries and assumptions.

3.4.1 Preliminaries

In an ideal situation, the camera should have a 360-degree field of view. In this work, we consider the angular field of view of the camera is, $\theta \in]0^o, 180^o]$. Meaning that if the camera is in the centre of a circle, the camera will collect images from the full arch within the angle range $]0^o, 180^o]$.

Each camera has characteristics which can be defined as follows:

- **Focal Length (FL):** is the distance between the lens and the image sensor when the subject is in focus. In other words, it is the distance from the back of the lens to the plane of the image formed of an object placed infinitely far in front of the lens, usually stated in millimetres.
- **Angle of View (AOV):** is the angle subtended by the camera lens, i.e., the visible extent of the scene captured by the camera lens. A wide-angle of view captures a broader area, and vice versa.

- **Resolution**: is the number of pixels per image. The higher the number of pixels, the higher the image quality.

The Field of View (FOV) for the covered area in the WVSN can be specified by the Angular Field of View (AFOV), in degrees, or the Linear Field of View (LFOV), in metres. The AFOV is defined by the focal length, f, and the horizontal dimension of sensor in millimetres, b, as in Equation 3.1. The shorter the FL, the wider the AFOV, see Figure 3.1. Both the AFOV and the LFOV can be measured horizontally, vertically, or diagonally.

$$AFOV = 2\,tan^{-1}\left(\frac{b}{2f}\right) \qquad (3.1)$$

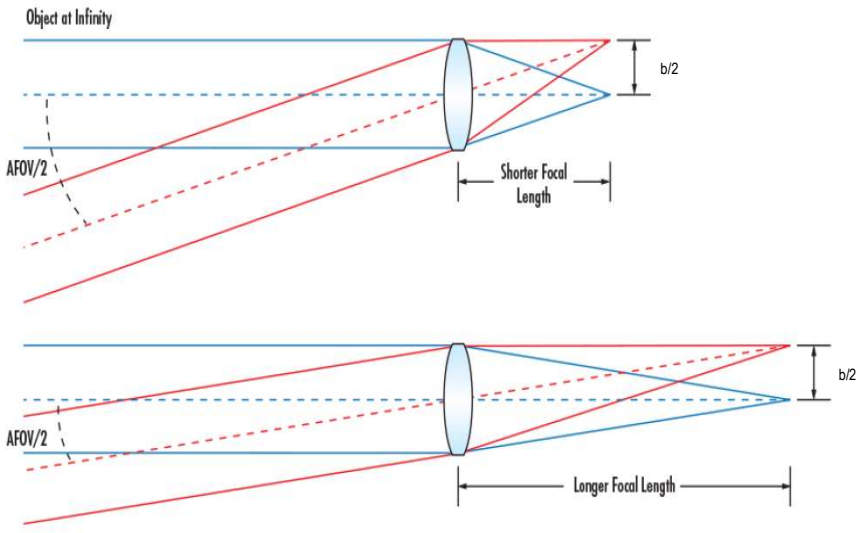


Figure 3.1: The Relation Between the AFOV and the FL

For a given sensor camera and without varying the FL, the AFOV remains constant in contrast to the LFOV which varies depending on the distance between the sensor and the monitored area. The larger the distance between the sensor and the monitored area, the larger the LFOV the poorer the quality of the collected images.

3.4.2 Assumptions and Definition

Assume that the greenhouse is a rectangular area with length L and width W. In the beginning, we assume that there is no obstacle between the sensor camera and the plants, so we can place the cameras wherever we feel we need them.

The assumptions regarding the properties of the WVSN cameras are as follows:

- All the cameras have the same characteristics.
- The camera's field of view is an angle of $\theta \in]0^o, 180^o]$.
- The FL of the camera is fixed; in other words, all the cameras have the same AFOV.
- All the cameras have the same capture resolution $R = R_h * R_v$, where R is the number of pixels of the image. R_h is the horizontal number of pixels. R_v is the vertical number of pixels.
- The cameras can be fixed in a predefined position and placed at the same height.
- The monitoring area is a shape in two-dimensional Euclidean space.

Definition 1. *In this chapter, we define the quality of an image as the number of pixels per unit distance.*

Table 3.1 defines the symbols used in this chapter.

Table 3.1: Glossary of Notation

Symbol	Meaning
C	Set of cameras
θ	Angle of view
R	Image resolution, total number of pixels of the image
R_h	Number of pixels in the horizontal line of an image
R_v	Number of pixels in the vertical line of an image
Q	The image quality, number of pixels per unit distance
Q_{min}	The minimum accepted quality

3.4.3 Optimization Problem Formulation

In this work, we are interested in determining:

1) The optimal placement of cameras to have the desired quality of the image object.
2) The required number of cameras to cover the entire monitored area.
3) The positioning of the cameras so that there is no overlap between images taken by the cameras.

We consider that the monitoring area is a rectangle in the two-dimensional Euclidean space defined by the points ABCD, such that $A = (x_a, y_a), B = (x_b, y_b)$, C= (x_c, y_c), and D = (x_d, y_d). We are interested in the covered area determined by the horizontal field of view of the cameras. In other words, we are interested in the part of the line defined by the points A and D.

Denote the set of cameras by $C = \{1, \ldots, i, \ldots, c\}$. A camera i has an AOV denoted by θ_i, and we have $\forall (i,j) \in C^2, \theta_i = \theta_j$. Denote the coordinates of the camera i by (x_i, y_i). Our aim is to calculate the coordinates of the part covered by the camera, illustrated in Figure 3.2 as a green line [I′ I″], where point I′ is $(x_{i'}, y_{i'})$ and point I″ is $(x_{i''}, y_{i''})$. Coordinates of both points can be calculated as follows.

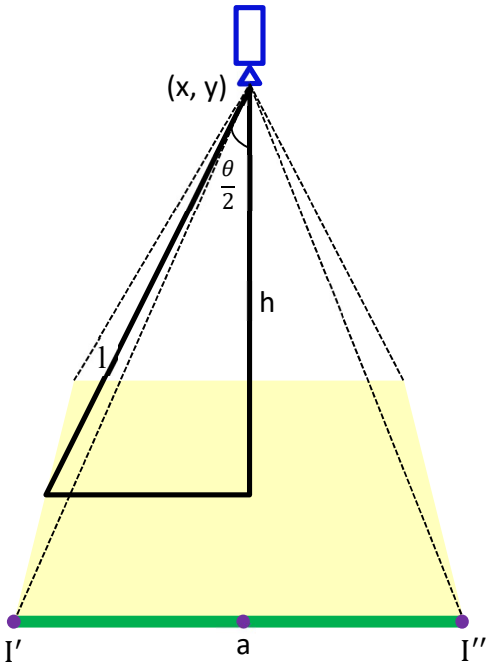


Figure 3.2: The Horizontal Field of View

$$x_{i'} = x_i - \frac{a}{2} \quad (3.2)$$

$$y_{i'} = y_i \quad (3.3)$$

$$x_{i''} = x_i + \frac{a}{2} \quad (3.4)$$

$$y_{i''} = y_i \quad (3.5)$$

The height, h, (or the distance from the camera to the sensed object or the ground), and the AVOF, θ, are known. But $\frac{a}{2}$ is unknown.

Hence,

$$(x_{i'}, y_{i'}) = \left(x_i - h \times \tan\left(\frac{\theta}{2}\right), y_i\right) \quad (3.6)$$

$$(x_{i''}, y_{i''}) = \left(x_i + h \times \tan\left(\frac{\theta}{2}\right), y_i\right) \quad (3.7)$$

Given h and θ, $\frac{a}{2}$ can be found as follows:

$$sin\left(\frac{\theta}{2}\right) = \frac{a/2}{l} \Rightarrow a/2 = sin\left(\frac{\theta}{2}\right) \times l$$

and

$$\cos\left(\frac{\theta}{2}\right) = \frac{h}{l}$$

Hence,

$$a/2 = h \times \tan\left(\frac{\theta}{2}\right) \quad (3.8)$$

Initially, we are interested in finding the optimal placement of the cameras considering the horizontal field of view. For this reason, the value of the coordinates over the Y-axis of the covered area is equal to the value of the Y-axis of the camera. And the covered area on the X-axis is:

$$a = 2 \times h \times \tan\left(\frac{\theta}{2}\right) \quad (3.9)$$

Which is characterized by the part: [I' I"] such that I' and I" are determined by the (6) and (7).

3.4.3.1 Camera Placement Constraints

1) The coordinate on the Z-axis: The position of the cameras cannot be too close to the sensed object as to interfere with the plant's growth and cannot be too far from the sensed object to be able to analyse the captured images. If the camera is far from the object, we get a poor-quality image that will not allow track the growth of the plant and detect diseases.

   The height of the cameras from the ground:

$$\forall i \in C,\ h_{min} \leq d(I, I_0) \leq h_{max} \qquad (3.10)$$

   where d(I,I_0) is the distance of the camera i from the ground or the sensed object. I_0 is the orthogonal projection of the point I on the X- axis, and h_{min} and h_{max} are the minimum and the maximum distance, respectively, of the camera from the object that we can not go beyond.

2) The coordinate on the X-axis: The X-axis coordinate of a camera can take any value from the range determined by the X-axis coordinate of the monitored area. Hence, the X-axis coordinate of a camera i must satisfy:

$$\forall i \in C,\ x_a \leq x_i \leq x_d \qquad (3.11)$$

   where x_a and x_d are the X-axis coordinates of points A and D, respectively.

3) The coordinate on the Y-axis: Since we are interested in monitoring linear area, the Y-axis coordinate for all the cameras is equal to the Y-axis coordinate of the line that we want to monitor,

$$\forall i \in C,\ y_i = y_a \qquad (3.12)$$

4) Covering the entire area: Every point in the monitoring area must be covered by a single camera, i.e., every point is seen at least and at most by one camera.

$$\forall (x,y) \in [AD]\ \exists!\ i \in C\ /\ (x,y) \in [I'\ I''] \qquad (3.13)$$

5) The communication between the cameras: The distance between two consecutive cameras should be less than the transmission range of the cameras in order to guarantee good communication between the wireless cameras. We assume that all the cameras are homogeneous, i.e., the transmission range of a camera $i \in C$ is t.

$$\forall (i, i+1) \in C^2\ ,\ d(i, i+1) \leq t \qquad (3.14)$$

d(i, i +1) is the distance between the cameras i and i +1.

3.4.3.2 Objective Function Definition

We consider the following objectives in the definition of the optimization problem.

- Minimizing the number of cameras needed for covering the entire monitored area:

$$\text{Min } \{|C|\} \tag{3.15}$$

where |C| is the number of cameras.

- Maximizing, over X, the covered area by each camera:

$$\text{Max} \sum_{k=1}^{|C|} 2 \times X \times \tan\left(\frac{\theta}{2}\right) \tag{3.16}$$

- Maximizing, over X, and |C|, the quality of images:

$$\text{Max} \frac{|C| \times R_h}{\sum_{k=1}^{|C|} 2 \times X \times \tan\left(\frac{\theta}{2}\right)} \tag{3.17}$$

By combining the three equations, the objective function of the optimization problem can be written as:

$$\text{Max}\left\{\frac{\sum_{k=1}^{|C|} 2 \times X \times \tan\left(\frac{\theta}{2}\right) + \frac{|C| \times R_h}{\sum_{k=1}^{|C|} 2 \times X \times \tan\left(\frac{\theta}{2}\right)}}{|C|}\right\} \tag{3.18}$$

With considering the constrains, we can rephrase our optimization problem to be written

$$\text{Objective:} \boldsymbol{MAX}\left\{\frac{\sum_{k=1}^{|C|} 2 \times X \times \tan\left(\frac{\theta}{2}\right) + \frac{|C| \times R_h}{\sum_{k=1}^{|C|} 2 \times X \times \tan\left(\frac{\theta}{2}\right)}}{|C|}\right\}$$

Subject to: *Eqs (3.10)-(3.14)*

This problem is cast as an Integer Linear Program (ILP) optimization problem explained in the next subsection.

3.5 ILP Optimization Problem Formulation

We present in this section the ILP formulation of our defined optimization problem above. We call the ILP problem as Integer Linear Programming-Optimal Placement Camera Quality (ILP-OPCQ). To solve the ILP-OPCQ problem we identify the workspace as a grid map; the monitored area (i.e., the monitored green line in Figure 3.2) corresponds to a vector line with $L = E[r2_x]$ lines. The space where the cameras can be placed is viewed as a grid map with $L = E[r2_x]$ lines and $K = E[h_{max}]$ columns, denoted it by $P(k, l)$, where $k = 0, \ldots, K$ and $l = 0, \ldots, L$.

3.5.1 ILP-OPCQ Objective Function Definition

We consider rephrasing the objective function as follows.

- Minimizing the number of cameras needed for covering the entire monitored area:

$$\text{Min} \sum_{k=h_{min}}^{h_{max}} \sum_{l=r1_x}^{r2_x} P(k,l) \qquad (3.19)$$

where $r1_x, r2_x$ are the X-axis coordinates of the horizontal line of the area.

$$P(k,l) = \begin{cases} 1, & if\ a\ camera\ placed\ at\ coordinates\ (k,l) \\ 0, & otherwise \end{cases}$$

- Maximizing k in the Eq. (3.20), which stands for maximizing the covered area by each camera.

$$\text{Max} \sum_{k=h_{min}}^{h_{max}} \sum_{l=x_1}^{x_2} 2 \times k \times \tan\left(\frac{\theta}{2}\right) \times P(k,l) \qquad (3.20)$$

- Maximizing k in the Eq. (3.21), which stands for maximizing the resolution quality of images.

$$\text{Max}\left\{\frac{\sum_{k=h_{min}}^{h_{max}} \sum_{l=r1_x}^{r2_x} P(k,l) \times R_h}{\sum_{k=h_{min}}^{h_{max}} \sum_{l=x_1}^{x_2} 2 * k * \tan\left(\frac{\theta}{2}\right)}\right\} \qquad (3.21)$$

By combining the three equations, the ILP-OPCQ objective function can be written as:

$$\text{Max}\left\{\frac{\sum_{k=h_{min}}^{h_{max}} \sum_{l=x_1}^{x_2} 2 \times k \times \tan\left(\frac{\theta}{2}\right) \times P_{(k,l)} + \frac{\sum_{k=h_{min}}^{h_{max}} \sum_{l=r1_x}^{r2_x} P(k,l) \times R_h}{\sum_{k=h_{min}}^{h_{max}} \sum_{k=x_1}^{x_2} 2 \times k \times \tan\left(\frac{\theta}{2}\right)}}{\sum_{k=h_{min}}^{h_{max}} \sum_{l=x_1}^{x_2} P(k,l)}\right\} \qquad (3.22)$$

3.5.2 Constraints of the ILP-OPCQ Problem

1. The cameras cannot be placed at a height less than h_{min} and greater than h_{max}, see constraint (3.10), which is equivalent to

$$P(k,l) = 0, \ \forall k \in [\![0, E[h_{min}] - 1]\!], \text{ and } l \in [\![0, L]\!] \qquad (3.23)$$

2. We can place at most a single camera in a column

$$\sum_{k=0}^{K} P(k,l) \leq 1, \ \forall l \in [\![0, L]\!] \qquad (3.24)$$

3. The left sight covered area of any camera should be greater than the X-axis coordinate of the beginning of the covered area.

$$\forall(k,\mathrm{l}) \in [\![1,K]\!] \times [\![1,L]\!]$$
$$\mathrm{P(k,l)} \times \left(\mathrm{l} - \mathrm{k} \times \tan\left(\frac{\theta}{2}\right)\right) \geq \mathrm{E[r1_x]} \qquad (3.25)$$

The right sight covered of any camera should be lower than the X-axis coordinate of the end of the covered area.

$$\forall(\mathrm{k},l) \in [\![1,K]\!] \times [\![1,L]\!]$$
$$P(k,l) * \left(\mathrm{l} + \mathrm{k} \times \tan\left(\frac{\theta}{2}\right)\right) \leq \mathrm{E[r2_x]} \qquad (3.26)$$

4. The constraint (13) is equivalent to

$$\forall(\mathrm{k},l) \in [\![1,K]\!] \times [\![1,L]\!]$$

such that

$$\mathrm{P(k,l-1)} \times \left(\mathrm{l-1+k} \times \tan\left(\frac{\theta}{2}\right)\right) \leq \mathrm{P(k,l)} \times \left(\mathrm{l-k} \times \tan\left(\frac{\theta}{2}\right)\right) \times \mathrm{P(k,l)} \times \left(\mathrm{l+k} \times \tan\left(\frac{\theta}{2}\right)\right) \leq \mathrm{P(k,l+1)} \times \left(\mathrm{l+1-k} \times \tan\left(\frac{\theta}{2}\right)\right)$$

$$P(k,l) \times \left(l + \mathrm{k} \times \tan\left(\frac{\theta}{2}\right)\right) \leq P(k,l+1) \times \left(l + 1 - \mathrm{k} \times \tan\left(\frac{\theta}{2}\right)\right) \qquad (3.27)$$

And

$$\sum_{\mathrm{k=h_{min}}}^{\mathrm{h_{max}}} \sum_{\mathrm{l=x_1}}^{\mathrm{x_2}} 2 \times \mathrm{k} \times \tan\left(\frac{\theta}{2}\right) \times \mathrm{P(k,l)} = \mathrm{E\,[r2_x]} - \mathrm{E\,[r1_x]} \qquad (3.28)$$

Consequently, with the aid of the constrains, the optimization problem can be rephrased as follows:

$$\textbf{Objective: } Max\{\frac{\sum_{k=h_{min}}^{h_{max}}\sum_{l=x_1}^{x_2} 2\times k\times tan\left(\frac{\theta}{2}\right)\times P(k,l) \; + \; \frac{\sum_{k=h_{min}}^{h_{max}}\sum_{l=r1_x}^{r2_x} P(k,l)\times R_h}{\sum_{k=h_{min}}^{h_{max}}\sum_{l=x_1}^{x_2} 2\times k\times tan\left(\frac{\theta}{2}\right)}}{\sum_{k=h_{min}}^{h_{max}}\sum_{l=x_1}^{x_2} P(k,l)}\}$$

Subject to: *Eqs. (3.23)-(3.28)*

3.6 Implementation and Numerical Investigation

In general, most attempts from the literature, for example [49] and [50], considered the placement optimization problem to be an NP-complete problem due to its complex constraints. Our formulation adds extra factors and constraints by considering the quality of images and the fact that each point must be seen by one and only one camera which previous works did not address. This adds more complexity to the problem, specifically, the fifth quadratic constraint of our problem Eq. (3.27). Thus, our problem is also considered to be an NP-complete problem due to its complexity. Our approach to solve the ILP-OPCQ problem is by solving it as two sub-problems sequentially:

1) Maximizing the quality of images and the covered area: ILP-OPCQ problem gives us the optimal distance, between the cameras and the object, regarding the quality of the images while maximizing the HFOV (i.e., the horizontal axis of the LFOV) of the cameras. This problem is modelled as the maximization of a function without constraint as explained in the next subsection.
2) Avoiding overlaps between images and determining the number of cameras: ILP-OPCQ problem allows us to determine the exact position of each camera; thus,

minimizing the necessary number of cameras to cover the entire area and avoiding overlaps between the collected images. This problem is modelled as an Integer linear programming problem as detailed in the second part of this section.

3.6.1 Maximizing the Quality of Images

Our aim in this section is to find the cameras' best placement to cover a large area while guaranteeing high-quality images. The objective function of this sub-problem is the same as the objective functions Eq. (3.16) and Eq. (3.17) of the main problem. In other words, the sub-problem definition includes finding the optimum of one variable X, which is the distance between the cameras and the sensed objects. This distance is defined as the perpendicular distance between the cameras and the sensed area. This distance is the optimal distance, which guarantees good quality images while maximizing the area covered by each camera. The following function illustrates the relationship between the covered area and the quality of images.

$$\text{Max } f(X) = 2 \times X \times \tan\left(\frac{\theta}{2}\right) + \frac{R_h}{2 \times X \times \tan\left(\frac{\theta}{2}\right)} \qquad (3.29)$$

where X is the distance between the object and the camera and R_h is the horizontal resolution of the image. The first term of the Eq. (3.29) relates to the maximization of the sensed area, while the second term relates to the maximization of the image resolution. Figure 3.3, illustrates how the function f varies in terms of the value of the distance X. We considered that R_h = 1024 M pixel and θ =160°. We can see from the figure that both the sensed area and the image quality depend on the distance X. As X increases the sensed area increases, and the quality of the images decreases and vice versa.

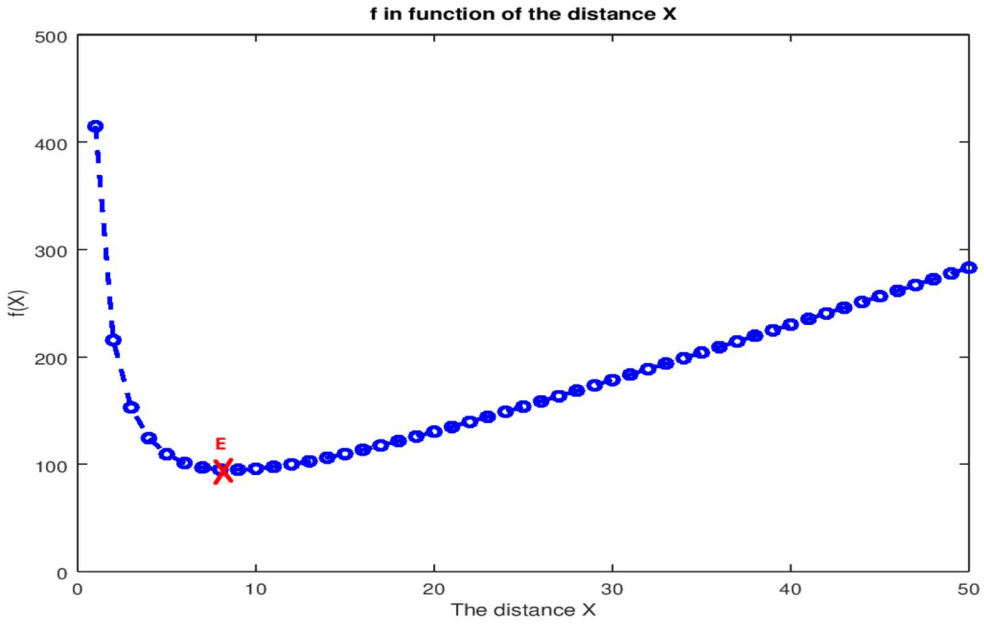


Figure 3.3: **The Relation Between the Distance X and Function f**

Overall, this function accepts one extremum point (a minimum), denote by $E_{(x_e,y_e)}$. Before this point, i.e., when $x < x_e$, as long as x decreases toward 0 the function increases towards +∞. Meaning that we are maximizing the quality of the image and decreasing the covered area. On the other hand, when $x > x_e$, as long as x tends toward +∞ the function f also tends toward +∞ which means that we are increasing the covered area, while the quality of the image is decreasing.

Accordingly, the point E presents a good compromise between the covered area and the quality of images. Thus, either we maximize the covered area and minimize the quality of images or we minimize the covered area and maximize the quality of images.

The point E is the solution of the following equation:

$$f'(X) = 2 \times \tan\left(\frac{\theta}{2}\right) - \frac{R_h}{2 \times \tan\left(\frac{\theta}{2}\right) \times X^2} = 0$$

which is:

$$X = \frac{\sqrt{R_h}}{2 \times \tan\left(\frac{\theta}{2}\right)} \qquad (3.30)$$

Next, we consider studying the relationship between the cameras' optimal position and the quality of the image function. There is a trade-off between the resolution of the image and the covered area. If the quality of the camera is high, then the camera covers a smaller area. Figure 3.4 shows this relationship. The X-axis represents the resolution of the camera which varies from 100 to 1024 pixels. The higher the resolution, the larger the optimal distance and the larger the sensed area.

It is worth mentioning that the optimal distance obtained by maximizing the Eq. (3.29) may give a quality that is lower than the minimum acceptable quality to analyse the plants. To avoid this, we propose Algorithm 3.1 to determine the optimal distance between the cameras and the covered area that guarantees at least a minimum image quality.

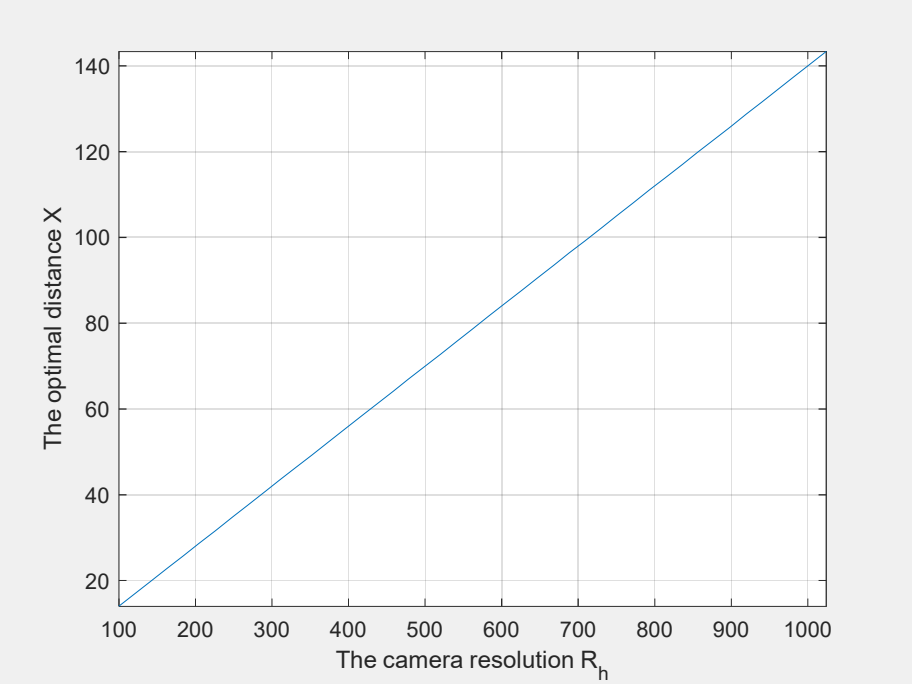


Figure 3.4: The Optimal Distance in Function of the Image Resolution

Algorithm 1 input parameters are the properties of the cameras, namely:

- θ , the AFOV of the cameras in degree
- R_h, the quality of the cameras, i.e., the number of pixels of the horizontal line of images taken by the cameras.
- Q, image quality. We set $Q = Q_{min}$, the minimum quality that we should guarantee to properly analyse the images. It is expressed as the number of pixels per unit distance, i.e., how many pixels represents the information that exists in a unit of distance in the area covered. Higher number of pixels means more detailed image. Q_{min} is used only if the optimal distance X does not guarantee the minimum required resolution.

Algorithm 3.1: Determining the Optimal Distance Between the Cameras and the Area Covered

Input: θ, R_h, $Q = Q_{min}$

Output: X, optimal distance between the cameras and area covered

$$X = \frac{\sqrt{R_h}}{2 \times \tan\left(\frac{\theta}{2}\right)}$$

if $\left(\frac{R_h}{2 \times X \times \tan\left(\frac{\theta}{2}\right)} < Q_{min}\right)$ **then**

$$X = \frac{R_h}{2 \times Q_{min} \times \tan\left(\frac{\theta}{2}\right)}$$

end if

Return X

3.6.2 Optimizing the Number of Cameras

Knowing the optimal distance of the cameras from the ground of the greenhouse (Algorithm 3.1), we can calculate n, the number of cameras necessary to cover the entire greenhouse area as follows:

$$n = H\left(\frac{D}{2 \times X \times \tan\left(\frac{\theta}{2}\right)}\right) \qquad (3.31)$$

where X is the optimal distance calculated using Algorithm 1, $D = x_a - x_d$ is the length of the covered area, and H(x) is the function defined by

$$H(x) = \begin{cases} E(x) + 1, & \text{if } (x - E(x) \geq \frac{1}{2}) \\ E(x), & \text{Otherwise} \end{cases} \qquad (3.32)$$

The function H makes sure that the number of cameras is an integer. The choice of the number ½ in the function H is arbitrary, and in the real world it depends on which is the

case to address: tolerating uncovering the last part of the covered area or not tolerating uncovering the last part of the covered area.

In our case, $x - E(x) \geq \frac{1}{2}$, means that the field of view of the last camera will not fully exploitable. Since it will cover the last part belonging to the covered area and determined by: $D - E(x) \times 2 \times X \times \tan\left(\frac{\theta}{2}\right)$ and the rest of the field view will be inexplicable. In the case where $x - E(x) < \frac{1}{2}$, the last part $D - E(x) \times 2 \times n \times \tan\left(\frac{\theta}{2}\right)$ will be uncovered. Otherwise, if $x - E(x) = 0$. Hence, the entire area will be covered with the n cameras.

Up to this point, we determined the minimum number of cameras to cover the whole area. Next is to determine the position of each camera.

The position, on the X-axis, of camera i is determined by the following sequence:

$$\begin{cases} \mathrm{x_i} = \mathrm{x_{i-1}} + 2 \times \mathrm{X} \times \tan\left(\frac{\theta}{2}\right) \\ \mathrm{x_1} = \mathrm{x_a} + \mathrm{X} \times \tan\left(\frac{\theta}{2}\right) \end{cases} \quad (3.33)$$

Algorithm 3.2 combines the steps solution of the two sub-problems described above. Inputs for Algorithm 3.2 are the same inputs of Algorithm 3.1. The outputs of the algorithm are X, the optimal distance between a camera and the plant (i.e., the Y-axis coordinate of the cameras), and V_p a vector which contains the exact position of each camera (i.e., the X-axis coordinate of each camera).

Algorithm 3.2: Determining the Minimum Number of Cameras and Exact Position of each one

Input: θ, R_h, $Q = Q_{req}$

Output: minimum number of cameras and position of each one

X = call Algorithm 1

$n = H\left(\frac{D}{2 \times X \times \tan\left(\frac{\theta}{2}\right)}\right)$ //n, *the minimum number of cameras to cover the whole area*

// The algorithm to determine the position of each camera

$V_p[n]$; //vector of n elements to store the position of each camera

$V_p[1] = x_a + X \times \tan\left(\frac{\theta}{2}\right)$

For j := 1 to n step 1 **do**

$V_p[i] = V_p[i-1] + 2 \times X \times \tan\left(\frac{\theta}{2}\right)$

end for

In real life, sometimes a specific image quality is required in order to analyze the plants and extract information from the captured images. In this case, therefore, we need to determine the exact position of cameras with respect to a given image quality value, Q_{req}. Steps of Algorithms 3.1 and 3.2 can be implemented to find those positions with using the same input parameters and set $Q = Q_{req}$.

3.7 Performance Evaluation

In this section, we evaluate our proposed optimization problem in practical settings with different input conditions for the AFOV, image resolution, and image quality. We used MATLAB R2019b software for this evaluation.

3.7.1 Experimental Cases and Results

Three test cases are considered in our experiments as explained below. For all the three cases, we consider a greenhouse with a grid area of side length L distance-unit. We set L = 1000 distance-unit. Our goal is to have a big number for L to show better results scale regardless of what unit can be assigned to L. Some camera sensors are deployed in each line of the grid area at the same height. The height will be determined in test case one, and the exact number of cameras will be determined in test case two. Sensor cameras capture images with the desired resolution once a day. Camera sensors transmit captured images to the base station. Single-hop and/or multi-hop communication can be used depending on the number of cameras and their locations. The communication protocol can be based on standard WiFi or Zigbee.

3.7.1.1 Test Case One

In the first case, we determine the optimal distance between the cameras and the plants, while guaranteeing a good quality image and wider area coverage. In this case, we set the cameras AFOV, $\theta = 120^{o}$, and the value of R_h varies from 100 pixels to 1024 pixels, to study the effect of camera quality in terms of the resolution of the collected images on the number of cameras necessary to cover the entire area and the optimal distance between the cameras and the plants.

In Figure 3.5, we plot the necessary number of cameras to cover the area L in function of the quality of cameras. The Y-axis represents the optimal number of cameras and the X-axis represents the quality of cameras, i.e., the resolution (R_h) of the collected images. We

observed that the optimal number of cameras varies from 38 cameras, for a resolution of 100 pixels, to three cameras for resolution of 1024 pixels.

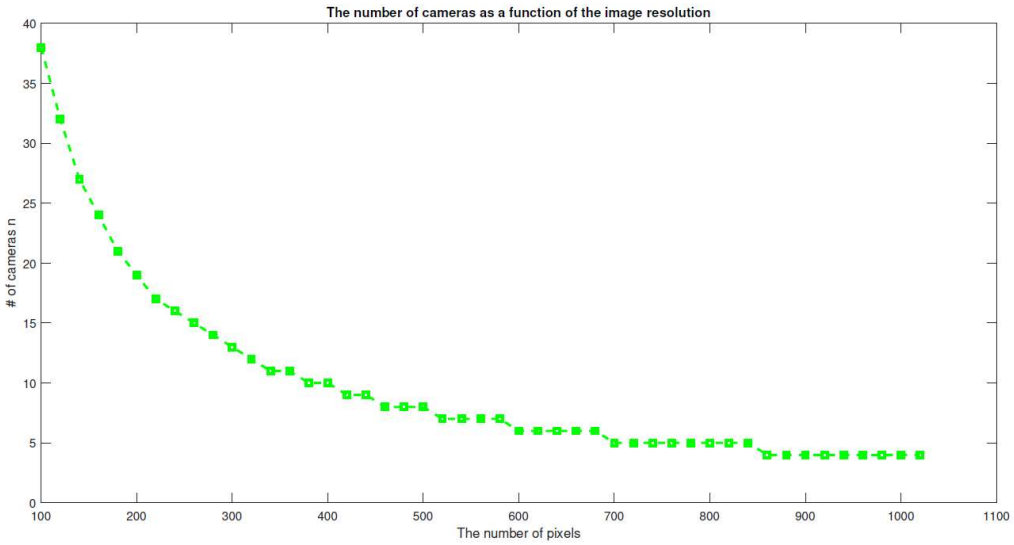


Figure 3.5: The Minimum Number of Cameras to Cover the Area L in Function of the Quality Image

Hence, if the quality of cameras increases, the number of cameras decreases. This is explained by looking at Eq. (3.30), increasing R_h (the quality of cameras) will result in increasing the optimal distance X, therefore, decreasing the number of cameras. Increasing the distance between the cameras and the ground X will allow the coverage of a larger area and hence, decreases the number of cameras.

This result is supported by the results plotted in Figure 3.6. It shows the optimal distance X as a function of the quality of cameras. As we can see, the optimal distance increases by increasing the quality of cameras; it goes from 8.33 distance-unit for the image quality resolution 100 pixels to 85 distance-unit for the image resolution quality 1024 pixels. So,

increasing the cameras' quality will increase the covered area by each camera and hence, decrease the number of cameras.

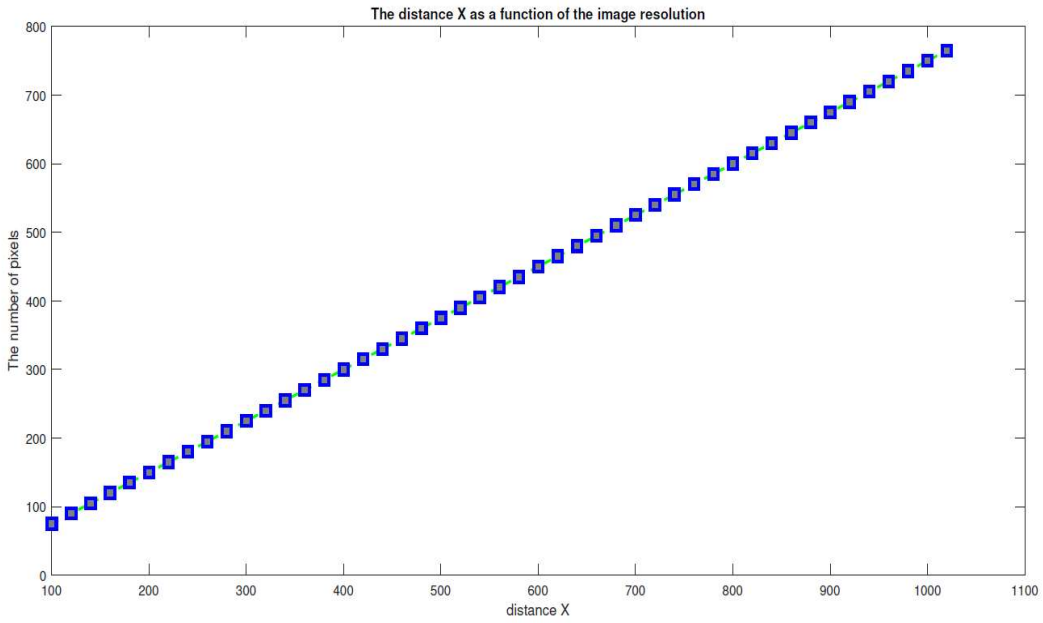


Figure 3.6: The Optimal Distance X in Function of the Quality of Camera R_h

3.7.1.2 Test Case Two

In the second case, we find the exact positions of the needed cameras, n, taking into consideration of avoiding redundancy and overlapping views between the collected images. In this case, we set the horizontal coordinates of the side L to $x_a = 0$ and $x_d = 1000$, the cameras AFOV, $\theta = 160^o$, and the value of R_h to be either 1000 pixels or 1500 pixels.

Figure 3.7 shows the obtained results where the green curve represents the exact position of the cameras that capture images of R_h=1000 pixels, and the blue curve represents the exact position of each of the cameras that capture images of R_h= 1500 pixels.

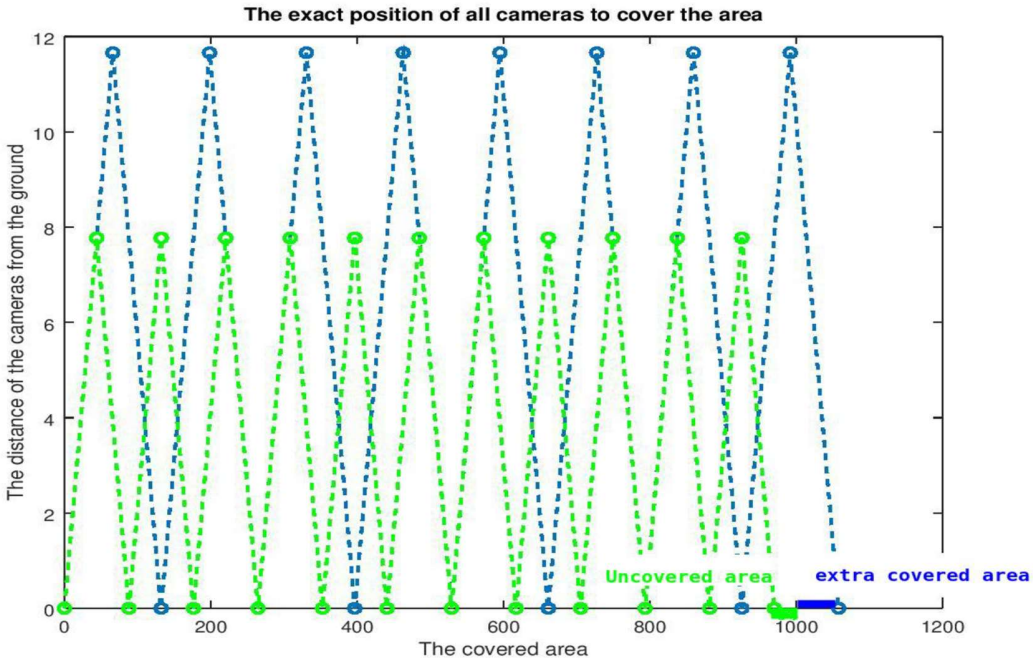


Figure 3.7: The Distance from the Camera to the Ground of the Greenhouse in the Covered Area

From the figure, we can see that in order to cover the entire area, we need either 11 cameras of quality R_h = 1000 pixels (each camera covers an area of 88.16 distance-unit) or eight cameras of quality R_h = 1500 pixels (each camera covers an area of 132.25 distance-unit). It is worth mentioning that the 8th camera will cover an extra part of area of approximately 57.96 distance-unit (the blue highlighted section in Figure 3.7). This is explained by the fact that the number of cameras n is equal to the integer part of ($\frac{1500}{2\times11.66\times\tan 80}$) plus 1,

which is $E(7.56) + 1 = 8$. Whereas, if choosing the second type of camera, i.e., cameras of quality R_h = 1000 pixels, an area of about 30.20 distance-unit will not be covered (the green highlighted section in Figure 3.7), since the number of cameras n is equal to the integer part of $\frac{1000}{2 \times 7.77 \times \tan 80}$ = 11.34 which is 11. Summary of the obtained results is given in Table 3.2.

Table 3.2: Cameras Position Properties

R_h (pixels)	X (distance-unit)	Coverage area per camera (distance-unit square)	Number of cameras
1000	7.77	88.16	11
1500	11.66	132.25	8

3.7.1.3 Test Case Three

In the third case, we consider finding the camera's position with respect to a given image quality to satisfy the purpose. For this case, we set the image quality to a specific value, Q_{req} = 20 pixels per distance-unit. Other input parameters, θ and R_h are set to the same values used in test case two (i.e., θ = 160°, and R_h = 1000 pixels or 1500 pixels). Based on the R_h value we consider two types of cameras:

- Type 1: cameras that can capture images of R_h = 1000 pixels
- Type 2: cameras that can capture images of R_h = 1500 pixels

Figure 3.8 shows the exact position of type 1 cameras in order to cover the area of L=1000 distance-unit. Each camera is placed at X = 4.41 distance-unit from the ground of the greenhouse, and the necessary number of cameras needed to cover the whole area is 20. Thus, each camera covers an area of 50 distance-unit.

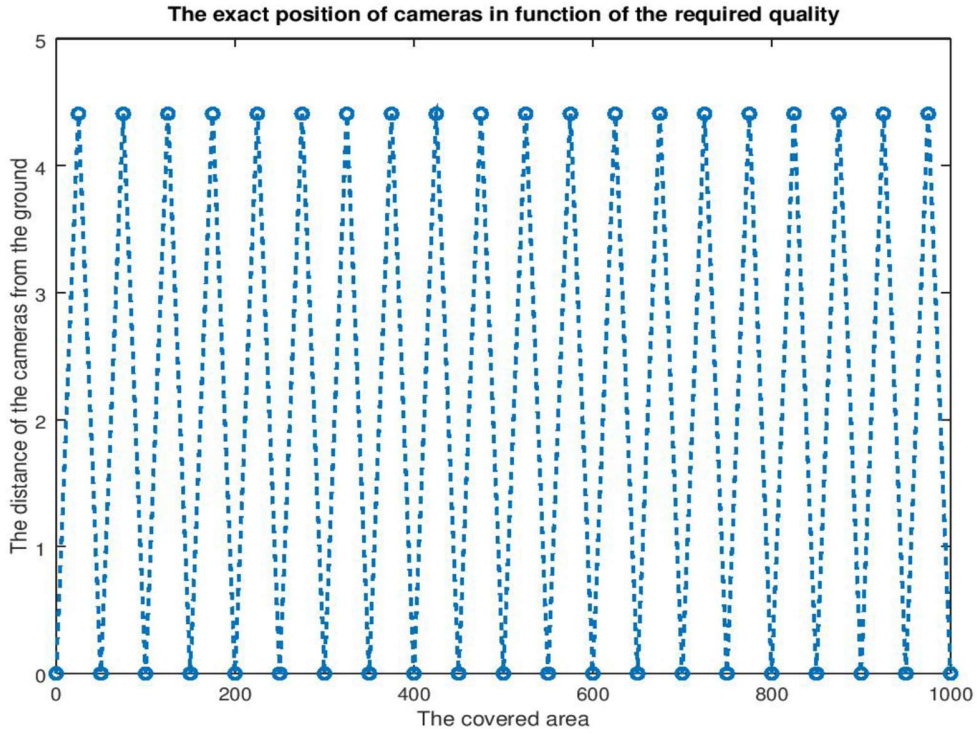


Figure 3.8: The Exact Position of Cameras

Figure 3.9 shows the exact position of type 2 cameras in order to the same area of L = 1000 distance-unit. For this camera, type 2, X, for each camera, is found to be at X = 6.61 and 13 cameras are needed to cover the whole area. This gives 75 distance-unit coverage area for each camera.

Comparing the two types of cameras, the first type of camera is closer to the ground of the greenhouse and can cover smaller areas compared to the second type of camera, which is higher up from the ground of the greenhouse and can cover larger areas. Hence, this confirms that the camera's quality and the required image quality are two important factors for determining the cameras' exact position.

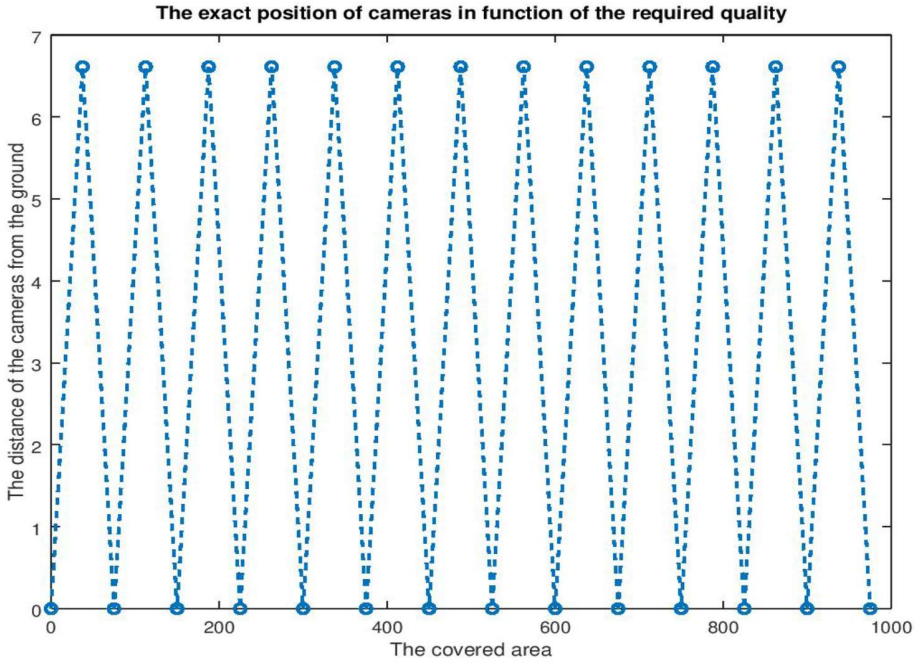


Figure 3.9: The Exact Position of Cameras in Function of the Required Quality

3.8 Summary

In this chapter, we discussed deploying WVSN in a greenhouse, aiming at finding the optimal number and position of the sensor cameras to cover the entire greenhouse area while maximizing both the quality of the images and the area covered by each camera. We formulated the problem as an ILP. We proposed two algorithms. Algorithm 1 for finding

the optimal position for the cameras and Algorithm 2 for determining the number of cameras needed with high-quality of images. Experimental results show the effectiveness of the proposed solution in finding the minimum number of cameras, the exact placement of each camera to cover the entire area being monitored in the greenhouse with the required image quality resolution to pick up any signs of plant disease.

Chapter 4

Fungus Detection System Using Wireless Visual Sensor Network and Machine Learning

4.1 Introduction

Statistics Canada released the 2018 census data that "In 2018, there were a total of 866 commercial greenhouse vegetable operations with 17.5 million square meters of production area, which produced over 660,535 metric tons of vegetables. There has been a steady increase in greenhouse vegetable production in Canada and it is anticipated that acreages in greenhouse vegetable production will continue the multi-decade growth trend. Ontario continued to lead the greenhouse vegetable sector in 2018, representing 68% of the total production in Canada, followed by British Columbia and Quebec with 18% and 6%, respectively." [116], shown in Table 4.1 and Figure 4.1. Also, the problem that concerns the labour cost of commercial greenhouse, as stated by Canada Agriculture, "Greenhouse operating expenses were up 1.6% in 2019 to $2.7 billion, largely driven by higher labour costs. Although the number of employees fell by 205 from a year earlier to 32,373 in 2019. As reported by *Canada's Agriculture Sector Labour Market Forecast to 2025*, the greenhouse industry was identified as the most problematic agriculture sector in terms of labour shortage." [51]. There is a growing demand to automate greenhouse crop production due to the recent COVID-19 pandemic.

Canada	2014	2015	2016	2017	2018
Greenhouse (square meter)	**14,216,767**	**14,592,933**	**15,928,094**	**16,878,194**	**17,438,325**

Table 4.1: Harvested Greenhouse in Canada (Square Metre)

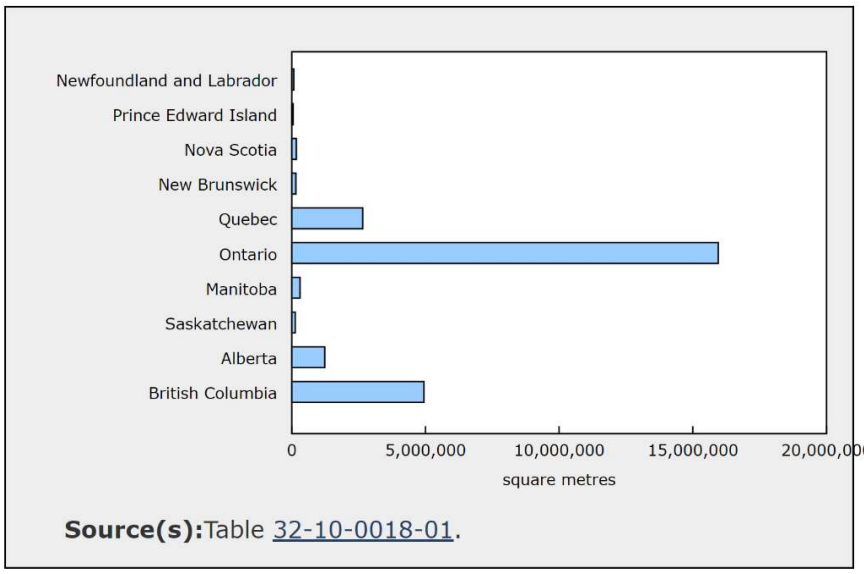


Figure 4.1: Provincial Distribution of Total Greenhouse Area, 2019

Large production crop greenhouses are growing rapidly. With this growth, comes a significant need to maintain production for economic reasons. One efficient way to maintain production growth is by controlling the greenhouse atmosphere and monitoring the plants to remain healthy throughout the life cycle.

Because the environment of a greenhouse is warm and humid, disease and pest issues can become particularly challenging for greenhouse growers. From fungus and rusts to viruses and root rots, disease can cause damage to plants. In this study and without loss of

generality, we will focus on detecting powdery mildew, which grows on the plant leaves. Powdery mildew is a fungal disease that looks like a dusty white coating on the leaves and stems of infected plants as shown in Figure 4.2. A powdery mildew infection usually begins as a few spores on the leaves but quickly spreads. The white powdery surface is a thick coating of the fungi spores [52]. This type of fungus increases in 99% humid conditions with moderate 25°C temperatures. In a greenhouse when the summer is humid, powdery mildew almost always makes an appearance. It can affect any plant. In extreme cases, it results in leaf yellowing and dropping, stunted plant growth, distortion of buds, blooms, and fruit, and eventually, overall weakening of the plant.

Figure 4.2: Powdery Mildew Fungus Disease

It is well documented that powdery mildew diseases, which are caused by several species of fungi, will affect most if not all plants. These diseases can reduce crop production, which leads to economic losses. Several methods can be used to diagnose and determine what harmful agent is affecting the plant leaves. These methods are the following [53]:

- Visual inspection: methods to discover certain deficiencies of the crop; however, not all damage can be seen by the naked eye.
- Soil analysis: measuring the amount of nutrient level in the soil.
- Tissue analysis: measuring the nutrient level in the plant leaves, and stem.
- Bioassays: a method for diagnosing nutrient deficiencies that combine tissue analysis and testing in pots.
- Field Tests: are one of the efficient methods to diagnose nutrient deficiencies, but it is an expensive procedure.

These methods are used as the first step in exploration; however, since field tests are expensive, difficult to administer, and can be done only in a laboratory we need an in-house, inexpensive technology to detect and diagnose different diseases with minimal human interaction. The use of technology in agriculture has been increasing. Depending on its nature agriculture technology can be biochemical (pesticides and fertilizers) or implemented into farm management. Mechanical and information technology can be applied to agriculture, such as in monitoring growth and controlling pests, geophysical measurement systems, flood detection and precision agriculture [54][55]. Moreover, there are sensor systems for monitoring the environment, such as ambient temperature, humidity, wind [56]. Monitoring systems are based on WSN technology [57]. WSN technology will not create a new agricultural product but will help improve existing techniques to improve the diagnosis of plant diseases and ensure final product quality [58].

WSNs have been used in countless applications [55]. One of these applications is measuring environmental parameters inside a greenhouse. Measuring these parameters is not enough to maintain healthy plants. To have healthy plants in greenhouses, we must carefully monitor the plants' growth and the environmental parameters by using reliable and affordable technologies. Among these technologies are the Wireless Visual Sensor Networks (WVSNs) and intelligent detection techniques, including image processing methods.

This chapter proposes developing an automated detection system to detect powdery mildew fungal disease and monitor the plant growth in a smart greenhouse, shown in Figure 4.3. We consider a greenhouse that is fully occluded and cluttered with varying degrees of light. The system utilizes four individual technologies, WVSN, WSN, a machine learning technique, and image processing methods. A WSN is deployed in a greenhouse to monitor the atmospheric conditions. A WVSN is deployed to monitor the plant growth via cameras, while image processing methods and machine learning techniques detect disease in the plant from the captured images from the camera sensors of the WVSN.

The reminder of this chapter is organized as follows. In Sections 4.2, we review the related work. Section 4.3 defines the problem statement and highlights the research problem. Section 4.4 describes the automated fungus detection system. Section 4.5 includes the experiment results and performance evaluation for the system, lastly, Section 4.6, present our summary of the chapter.

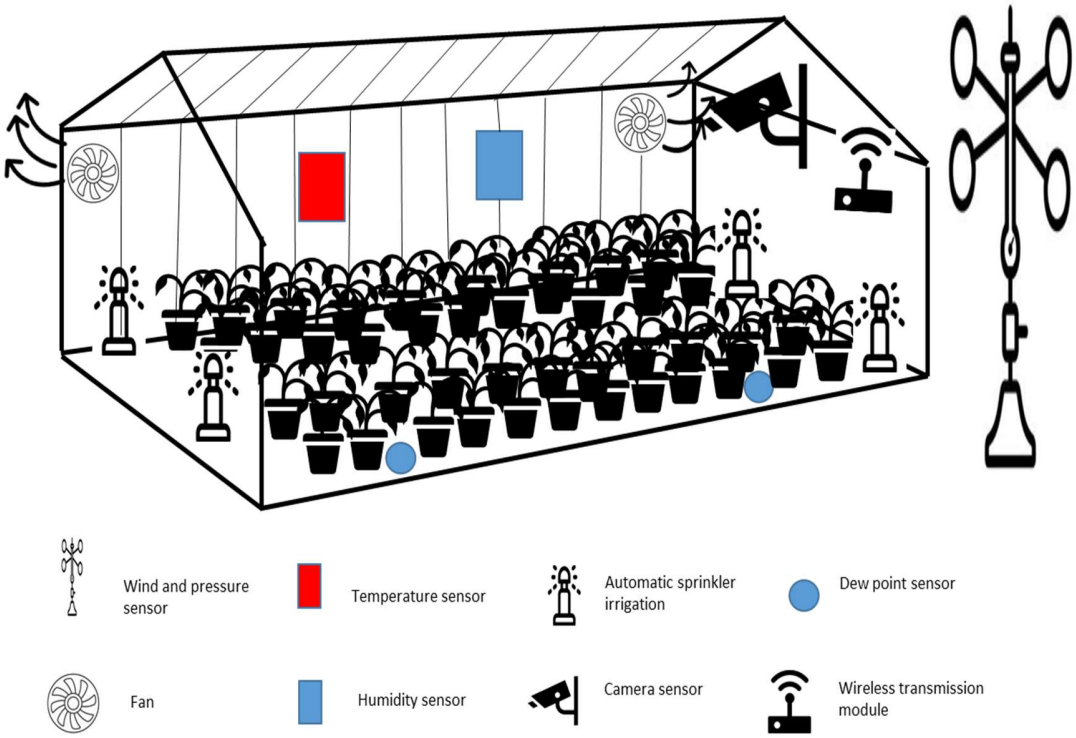


Figure 4.3: Typical Smart Greenhouse

4.2 Related Works

There is very little research in combining WSNs with image processing and pattern recognition in agriculture. The authors in [59] mentioned two systems for image recognition. They explained the structure, the recognition algorithms, and the neural classifier. One of their applications was image recognition based on an adaptive control system for micromechanics where a neural classifier was used for texture recognition of metal surfaces. The authors also used pesticides to kill insects by using a web-camera based computer vision system to automate the recognition of larva. Their system sought to locate the insect and larvae early so that they could reduce the use of pesticides. The system

consisted of neural classifiers, which would detect the insect from a captured image. Recognizing the larvae and sensing warmth to indicate the larvae were active are not easy tasks because different species of larva have different colors, shapes, and sizes and can be under a leaf. They used pre-processing techniques, then trained the system. Their system was not able to distinguish between textures related to the larvae and those related to the background of the image.

Research done by the authors in [60] had the same type of the system as in the previous work [59], but the purpose of this work was to use a back-propagation ANN model to distinguish between weeds and young corn plants. The authors used a series of cameras to obtain high-quality images. Each image was pre-processed from the bitmap format with image processing to indexed images based on the RGB color system. Then, each index color acted as input for the ANN. The output value was 0 or 1, representing whether the image was weeds or young corn plants. The processing time was 20 hours for training the network. This process can help reduce the use of herbicide sprays if it decreases the training time.

Another work involved in recognizing weeds [61] used a fuzzy logic system to create a weed map to determine the location of the weed to use the right amount of herbicide. The authors also used a digital camera and a personal computer for more testing. Their system was able to locate some of the weeded areas, which resulted in using less herbicides, reduced soil and water pollution, and some cost savings.

The authors of [62] used machine vision to detect a worm in maize plantings. They used a pre-processing technique that converted the image from grayscale to binary images using an iterative algorithm. First, the system segmented the leaves and divided them into pixels.

Second, the images were divided into blocks. Blocks that contained a more significant amount of leaf surface were selected. These selected blocks were recognized as damaged or undamaged by counting the objects in each block. Their system performed well overall. In [63], the authors merged three thresholding strategies, fuzzy method, Otsu method, and Isodata algorithm, to determine whether the field was covered with oat or frost. They stated that this merger provided better results than using each method separately.

The work in [64] presented the use of image processing to measure the water droplet size and distribution of agricultural sprinklers. They used the properties of Fourier analysis and correlation in the frequency domain. The purpose of this paper was to obtain a direct measurement of sprinkler drops, which would help avoid exceeding the size of the drop that would lead to soil erosion, surface sealing, and infiltration, as well as to minimize the size of the drop not to be affected by wind drift and that alters the pattern of irrigation. This study would help the farmer control the size of the drops and maintain the right amount of water.

Another use of visible light image processing and machine vision system was presented in [65] and [66] to detect diseases in the field. Their systems achieved a good detection rate with some restrictions on input, such as taking images only from the top view of the plant with uniform background and taking images only of a single centred leaf. These restrictions make the system unsuitable for autonomous detection.

Using a camera provides more information and benefits over sensor networks alone as in [67]. The authors used a camera sensor network for recognition, tracking, and detection. Their work introduced low-latency detection, low power, and efficient recognition. However, their work depended on using light image processing, which would not be

efficient in detecting pests or disease. In [115], the authors used image recognition based on the Convolution Neural Network (CNN) to detect powdery mildew of tomato crop. They create artificial powdery images using image fusion techniques to prepare various forms of CNN learning data. The testing was applied in real images taken from the greenhouse. The performance of their system achieved 93%. The weakness of using CNN in this task is, CNN does not encode the position and the orientation of the powdery from the images; the noise in the image can affect the performance of CNN. Also, CNN is not invariant to large transformations of the input data, which means that CNN cannot distinguish the object when it is in different viewing angles. It is also highly computational cost, needs good GPU and large training data.

Most of the previous works done on detecting diseases and pests used a digital camera with image processing. To the best of our knowledge, no work such as our proposed automated system has been done in a greenhouse.

4.3 Problem Statement and Contributions

Early and fast detection of any diseases or pests in a greenhouse is an essential step and part of an integrated management strategy needed to maintain the health of the plants and increase crop production. Automated plant disease detection in an environment like a greenhouse is complex because the surroundings are a fully cluttered, large-scale, and uncontrolled environment. In this chapter, we investigate a solution for solving the following problem:

Design and develop an automated detection system for observing disease, pest, or deficiency present on the leaves of plants in an occluded and/or cluttered greenhouse.

To address this problem, we propose an automated detection system that utilizes WVSN, WSN, the Hough Forest machine learning technique, and image processing methods. WVSN and WSN are deployed to continually monitor the plant growth and the atmospheric conditions of the greenhouse for healthy crop production. The image processing methods and the Hough Forest technique are used to efficiently detect powdery mildew fungus in images of the plant's leaves taken from a WVSN. Our contribution can be summarized as follows.

Proposing an automated fungus detection system using WVSN, WSN, the Hough Forest machine learning technique, and image processing methods, to distinguish between healthy and unhealthy plants, and identify the powdery mildew fungus on the plants with a high level of accuracy.

4.4 Automated Fungus Detection System

The block diagram of the proposed detection system is presented in Figure 4.4. The system has the following four units: WVSN unit, WSN unit, image processing unit, and machine learning detection unit. In the WVSN unit, camera sensors are installed inside the greenhouse to capture images and transmit them to the image processing unit. The image processing unit processes the captured images using different methods: resizing, filtering,

segmentation, and noise removal. The image processing unit's output images are used as input to the machine learning unit to detect mildew fungus. If fungus is detected, then the WVSN unit communicates with the WSN to send a job request to an actuator to do an action such as start the fan or open a window to reduce humidity. In the following subsections, we present the details of each unit.

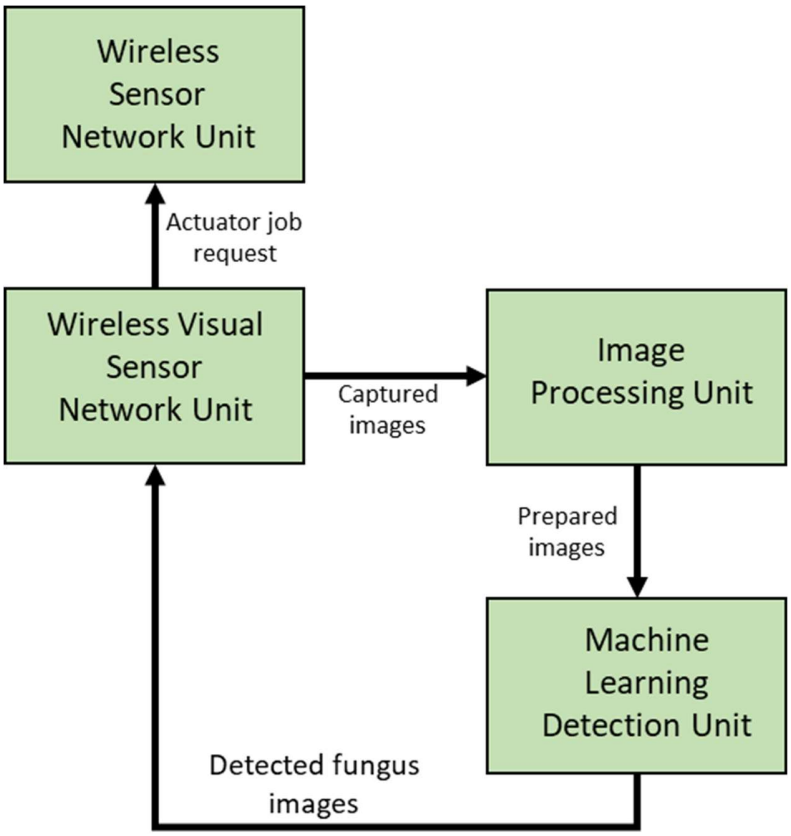


Figure 4.4: Block Diagram of the Proposed System

4.4.1 Wireless Visual Sensor Network Unit

A WVSN has camera-equipped sensor nodes. The very small sensor camera nodes can capture visual data, and process and transmit image/video information [68]. The nodes capture images to monitor the growth of the plants, detect any fungal diseases, or pests that can adversely affect the plant. This type of communication network will minimize human interaction. In general, the monitored area is an immense place, which means we must deal with a large number of images. To improve the performance, in terms of storage and processing, and reduce the response time of the image processing unit, we consider placing the sensor camera nodes such that there is no overlap between images taken by those nodes. Optimum locations for the nodes are determined and explained in Chapter 3.

Images are taken from different locations and angles of different camera sensor nodes in a greenhouse during the daytime. The sensor camera's specifications are (12MP, 50mm focal length, 1/2.55-inch sensor, dual-pixel PDAF, and f/1.5-2.4 variable-aperture lens), as well as another camera sensor (12MP, 2x focal length, f/2.4 lens, 1/3.6-inch sensor, AF). Also, a single LED flash is in both cameras. The distance between the camera sensor and the plant achieves better resolution at the same time with no overlap explained in Chapter 3.

Samples of the plant leaves in the greenhouse images are shown in Figure 4.5. The dataset consisted of 282 images at 1960 x 4032 pixels/image obtained from a greenhouse in Surrey, British Columbia, Canada [107]. Images were taken with different levels of occlusion. The levels varied between images from low to highly occluded and cluttered. The images were

visually inspected for powdery mildew fungus, which was observed in the images. Note that these images have different resolutions from the images used in Chapter 3.

Figure 4.5: Plants Images Captured by Sensor Camera Nodes in the Greenhouse

4.4.2 Image Processing Unit

In this unit, different image processing methods are used to prepare the images for the machine learning unit. One of the most prominent issues in disease detection techniques is background clutter in a greenhouse setting. The problem with background clutter is the high possibility of false detection, which will decrease efficiency and accuracy by searching to cover an area that does not contain the object of interest. The use of segmentation image processing for the fungus will remove the background clutter by applying the following six steps:

1) Find the RGB color spaces from the image. RGB stands for red, green, blue channels. RGB is a composite of the independent grayscale images that correspond to the intensity of red, green, and blue light.
2) Determine the difference between red color image and green color image. Get the difference between red channel and green channel
3) Convert the images to grayscale.
4) Create the mask for the foreground (i.e., the plant leaves)
5) Apply a median filter over the mask to remove the noise. First sorting all pixel values from the window into numerical order, and then replacing the pixel being considered with the middle (median) pixel value.
6) Extract the foreground from the image.

Final results will have the Region of Interest (ROI) image, as shown in Figure 4.6. The prepared images will be used as input into the machine learning detection unit.

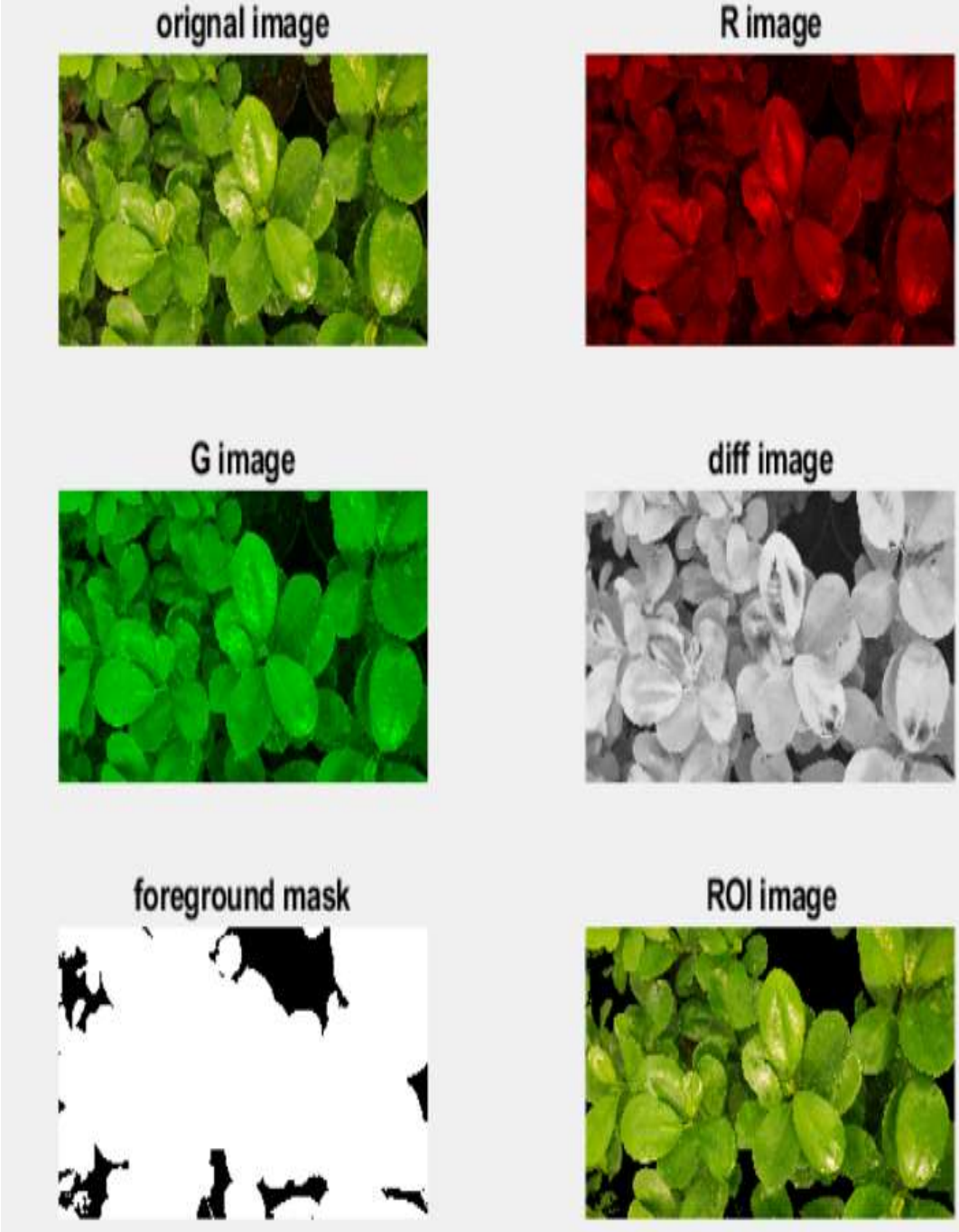


Figure 4.6: Preparing Captured Images

4.4.3 Machine Learning Detection Unit

In this section, we propose using Hough Forest machine learning to detect powdery mildew fungus on the plant leaves. Hough Forest is chosen as the detection algorithm because of its robustness to occlusion and noise. Hough Forests combine random forests' learning properties with the detection properties of Hough transforms [69]. Prepared images, from the image processing unit, are used to create labelled image samples in which each image sample is labelled either has fungus or no fungus. The sampling is performed using a cropping tool that is developed to crop and label images successfully. The tool allows a user to left-click-drag-and release or right-click, using a two-button computer mouse to point a portion of the image as leaves. We note that since the samples are picked from training images that are selected randomly, the number of training samples will vary among the various folds. After sampling the training set, patches are created using a semi-automatic approach. P_i training batches are created with a size of 16x16 pixels. Each batch is then represented by a feature $P_i = (I_i, C_i, d_i)$: I_i is an appearance information which can includes image features (L*a*b color space), derivative filter (first and second differentials using Sobel operator), and three histograms of the oriented gradient, C_i is an image class (leave, background), and d_i is a distance vector from center of image to center of batch, as described in [70]. After that, the implementation of the Hough Forest described in [69] is used to train the classifier. Samples of the patch images used for training the decision tree. Each tree generates an output to create a prediction. A random subset of features was chosen for each split of tree branches. A training process is shown in Figure

4.7, and a detection process is shown in Figure 4.8. All the Hough Forest approach in Figure 4.9.

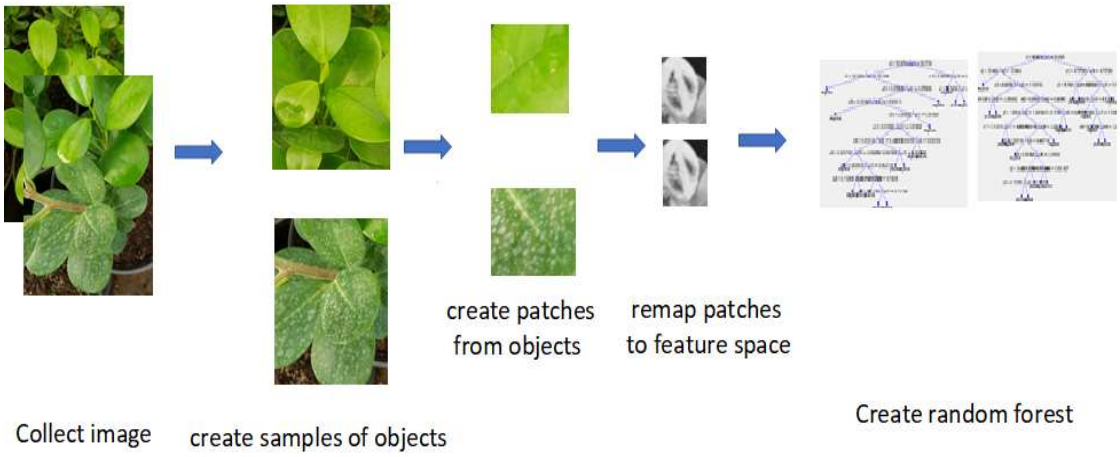


Figure 4.7: Hough Forests Training

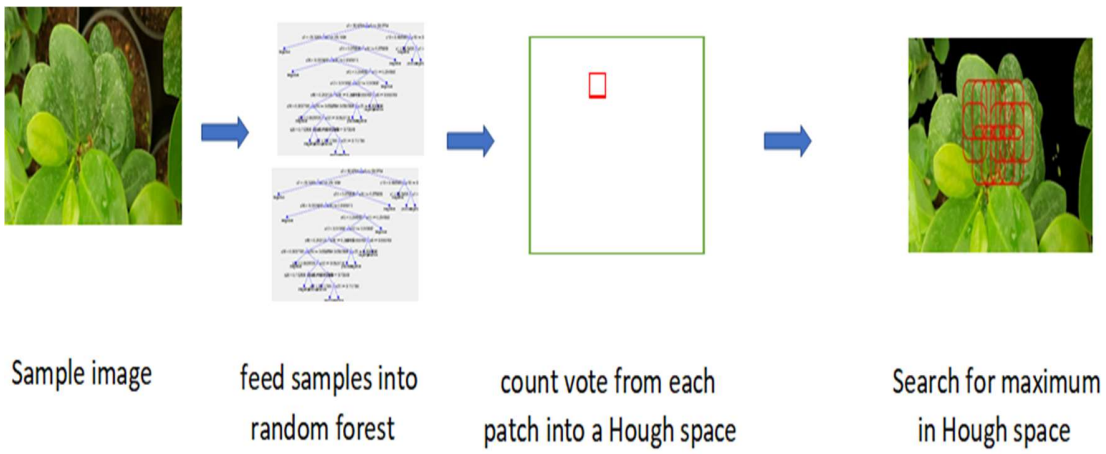


Figure 4.8: Hough Forests Detection

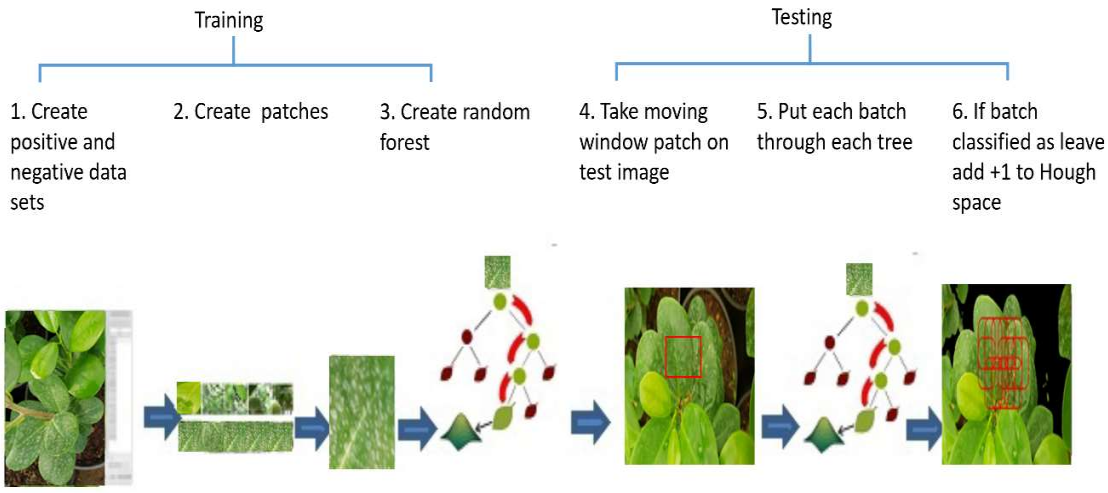


Figure 4.9: Hough Forest Approach

4.4.4 Wireless Sensor Network Unit

The greenhouse will have a WSN with actuators that communicate with the WVSN to control the greenhouse atmosphere. When the WVSN detects an unusual status in the plant leaves received from the machine learning detection unit, it will send a message to the BS of the WSN requesting the wireless sensor nodes to measure the humidity inside the greenhouse. If the humidity is high, the BS sends a message to the actuator to turn on a fan or open a window, for example, to decrease the humidity in the greenhouse (see Figure 4.3).

4.5 Experimental Results

In this section, we evaluate the performance of our proposed automated detection system. We start by describing the dataset used. Then we explain the training process following by performance evaluation for the system, statistic results, and comparison and discussion.

4.5.1 Dataset

Our dataset consists of 282 images at 1960 x 4032 pixels/image. Labelled sample images are prepared using a semi-automatic approach to create patches. Five hundred and two patches are created; 260 positive patches have fungus and 242 negative patches did not. All patches are re-sized to 256 x 256 pixels/image. A Hough Forest is trained with positive fungus images with the negative background removed. Samples of the patch images used for training are shown in Figure 4.10 and Figure 4.11. These images are not part of the testing set. More detailed results are included in the Appendix A.

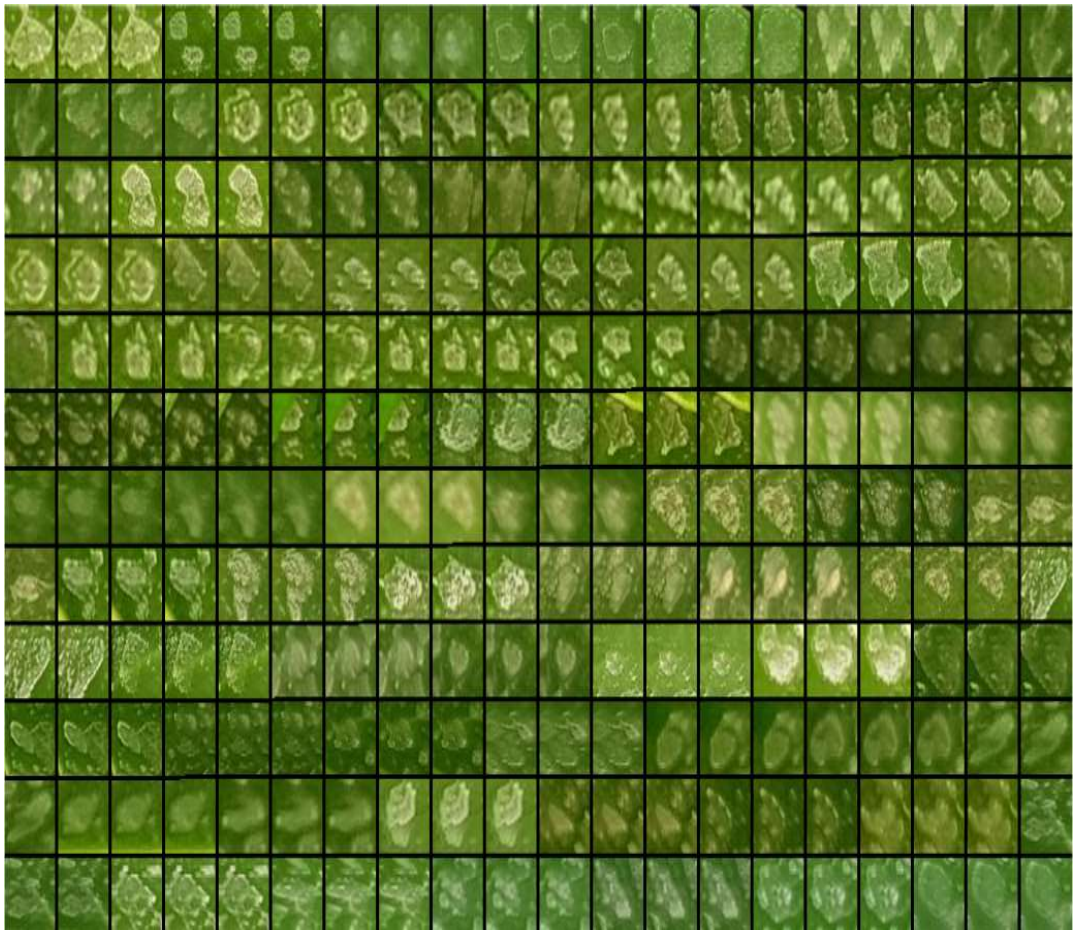


Figure 4.10: Positive Training Patches

Figure 4.11: Negative Training Patches

4.5.2 Training Process

General outlines of training and detection are shown in Figure 4.7 and Figure 4.8, respectively. Patches are extracted randomly from each image sample and carried different features. These features have the information used for constructing each tree that included each channel of the L*a*b color space, first and second discrete differentials, using the Sobel operator, as well as nine histograms of gradients, as described by Leibe et al. in [70]. J. Gall's re-implementation of the Hough Forest described in [69] is used to train the classifier. The patches are selected randomly with their location and image classification. Then they are passed along to the root node of the decision tree. Each patch is processed

through a tree until it reached a leaf node. The tree will split into two new nodes, which will maximize the information gain. Every node knows the position of the patches relative to the center of the image and image classification. The leaf node had the position and the class information about that patch, which would be used to create a vote into a Hough space. All trees have votes in Hough space. The highest number of votes indicates the correct location of the object (i.e., fungus). The process will keep repeating until it reaches the stopping point. Many trees are trained using the same steps, thus, creating a forest. The forest contained ten trees, each with a depth of 18 nodes, as shown in Figure 4.12.

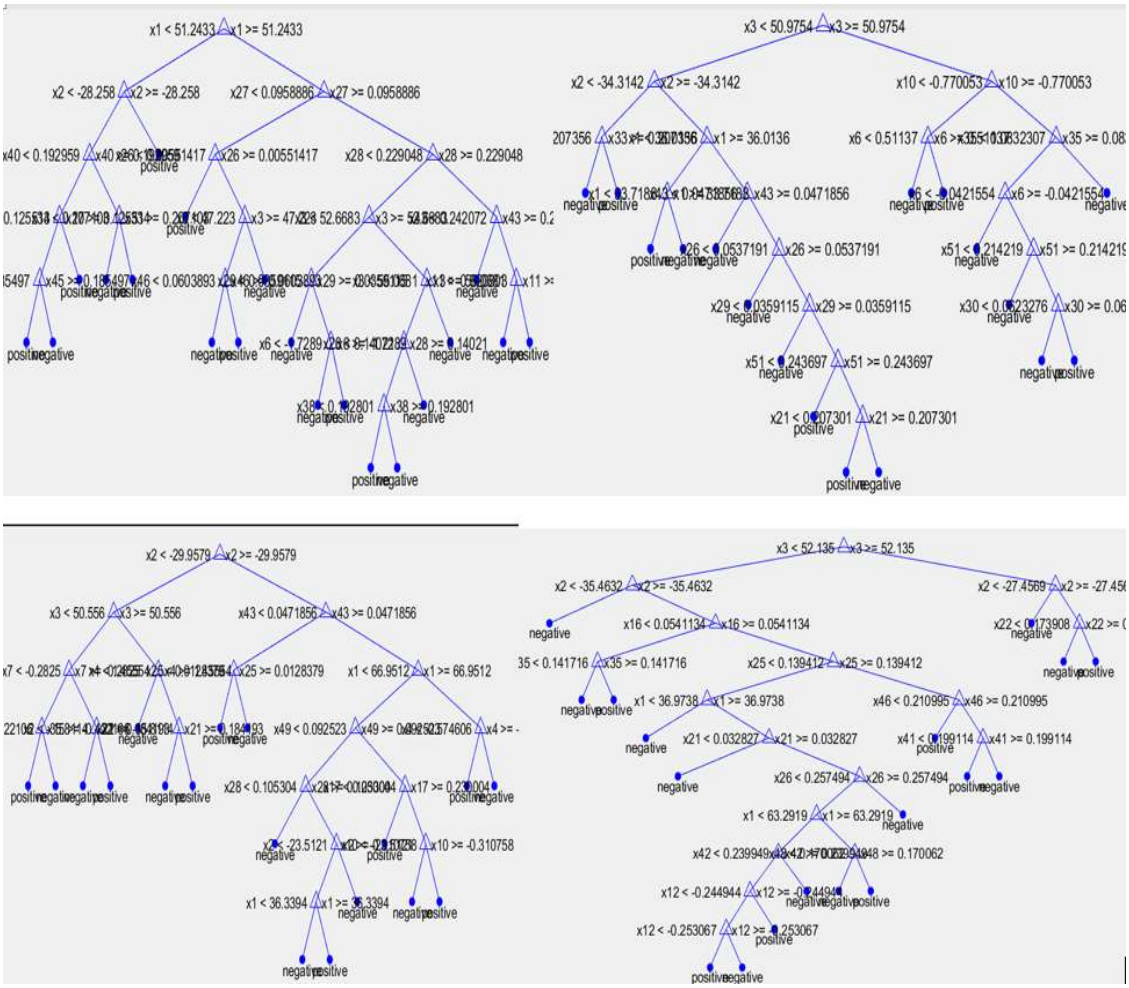


Figure 4.12: Hough Tree Forest

4.5.3 Performance Evaluation

With a machine learning algorithm, it is necessary to divide the dataset into training and testing subsets. This process will be repeated many times. Each time, the data is randomly selected to create different training and testing subsets. We use k-fold cross-validation to divide the data into training and testing, with k = 5. This approach ensures that every image sample will be tested, and the testing subset will not overlap. We use MATLAB R2019b software with intel ® Core ™ i7-7500 CPU@ 2.70GHZ to apply 5-fold cross-validation on 282 images taken from inside the greenhouse. The results of the 5-fold cross-validation are presented in Figure 4.13. The results are averaged over 282 images. We use the Receiver Operating Characteristic (ROC) parameter. The ROC is calculated by comparing the True Positive (TP) rate to the False Positive (FP) rate. Also, in Figure 4.13, we calculate the area under ROC curve (AUC), which evaluates how good the classifier is, and how accurate the output is. In our case, the AUC is 96.96%.

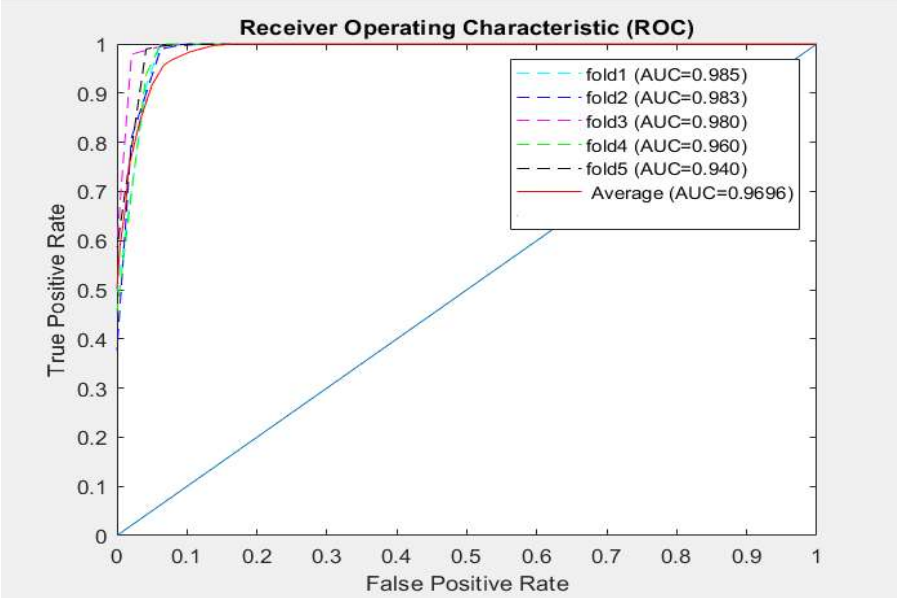


Figure 4.13: ROC for Hough Forest Trained with Fungus Image Patches

The results of testing 100 images are shown in Table 4.2. The results show detection rates of 71% True Positive (TP), and 23% True Negative (TN). The total of all true detection rates is 94%. The rate of False Negatives (FN) is low at 5%, and the rate of False Positives (FP) is much lower, at 1%.

Table 4.2: Results of Fungus Detection on the Testing Dataset

Test	Fungus in image %	No Fungus in image %
Detected fungus	TP (71)	FP (1)
No detected fungus	FN (5)	TN (23)

Sample output results from applying the Hough Forest machine learning on the images were TN detection (healthy plant) and TP detection (fungus found), as shown in Figures 4.14 and 4.15, respectively. FN detection and FP detection are shown in Figure 4.16 and 4.17, respectively.

Figure 4.14: True Negative Detection

Figure 4.15: True Positive Detection

Figure 4.16: False Negative Detection

Figure 4.17: False Positive Detection

4.5.4 Statistical Results

In Table 4.3, statistical results show how well our system performed in predicting powdery mildew fungus based on the images.

Table 4.3: Statistical Results

Sensitivity	93.4%	**Specificity**	95.8%
Positive likelihood ratio	22	**Negative likelihood ratio**	0.06
Positive predictive value	98.6%	**Negative predictive value**	82.1%

- Sensitivity shows the probability that our test results are positive when the fungus is present. In our case, we have a high probability of 93.4%.
- Specificity shows the probability that our test results are negative when the fungus is not present. In our case, we have a high probability of 95.8%.
- A positive likelihood ratio of greater than 1 indicates that the test result is associated with fungus. In our case, the result was 22, which conforms our output results.
- A negative likelihood ratio of less than 1 indicates that the test result is associated with an absence of fungus. In our case, the result was 0.06, which confirms with our output results.
- Positive predictive value shows the probability that the fungus is present in the images when the test is positive. In our case, the probability value was 98.6% (very high).
- Negative predictive value shows the probability that the fungus is not present in the images when the test is negative. In our case, the probability value was 82.1% (very high).

4.5.5 Comparison and Discussion

Table 4.4 shows a comparison of our proposed system applying Hough Forest machine learning on images taken from the WVSN, from different angles and placement, against each image process used in previous works [65], [66] and [115]. Images in [65] are taken from the top view, which minimizes the clutter from the background images. The work in [66] cropped the leaf images before applying color-texture detection, which reduced the

clutter of background images. In [115], they used artificial images in training while real images in testing; they could not deal with the real images that have noises and different viewing angles. Also, they used GPU to speed the training processes, but it was costly.

Table 4.4: Comparison with Other Works

Authors	Method	Images	Detection Rates (%)
L. Velzquez et al. [65]	Color feature detection	Images with top view	70
M. Zhang et al. [66]	Color texture detection	Images contain leaves only	67-88
H. Jung et al. [115]	CNN	Artificial Images	93
A. Ali and H.S. Hassanein [71]	Hough Forest, color and background removal	Images with different view and varying light	94

4.6 Summary

This chapter proposed an automated detection system for any disease or pests in an occluded and cluttered greenhouse. The system is designed to detect any type of diseases or pests on plant leaves. However, and without loss of generality, our study focused on detecting powdery mildew fungus disease as a proof of concept. Hough Forest machine

learning technique is applied to detect the powdery mildew fungus in images of the plant leaves taken from a WVSN. A detection rate of 94% is obtained, which confirms the performance strength of our proposed system. Also, our system obtained a low false positive rate, which is very important for maintaining a successful detection system, as each positive detection would require sending messages to sensor nodes to measure the humidity of the greenhouse and re-set accordingly.

Chapter 5

Intelligent Framework for Predicting and Controlling the Greenhouse Microclimate

5.1 Introduction

Continuous monitoring of plant crops and frequent controlling of the microclimate inside a greenhouse using WSN increases the node's energy consumption and, thus, shortens the network lifetime. Also, given the scale of commercial greenhouses and the scale of the network, the number of data transitions between sensor nodes and the BS in this specific environment will increase significantly, causing data congestion, a high rate of sensory data loss, and a low signal-noise ratio [72-76]. Moreover, the deployment and maintenance cost of a WSN is expensive. A smart greenhouse microclimate is a complex nonlinear system that provides an optimum environment for plant growth. Because of the complexity of the factors involved, the slow time variation, and the non-linearity of the smart greenhouse microclimate, it is challenging to build a precise mathematical model. The widely used current greenhouse microclimate modelling has the following three approaches: computational fluid dynamics model method, the mechanism by design modelling method, and the system identification method. These methods have several limitations and unknown parameters. They also require expensive instruments and tests [77-81]. A smart greenhouse requires intelligent technologies and tools to process data at a reasonable cost and translate it into better decisions and actions [82]. Thus, it is important to accurately predict a greenhouse microclimate for environmental control and crop management. Likewise, controlling the duty cycle (i.e., operational activities, wake-up and

sleep) of monitoring sensor nodes will reduce the consumed energy and prolong the network lifetime. When the weather background and the composition of the system's greenhouse components are determined, the system's unique agricultural microclimate characteristics will be relatively stable, which is conducive to prediction. Data prediction helps improve data quality, reduces unnecessary data transmission, and saves sensor nodes energy. Microclimate prediction is useful in the thermal analysis of a greenhouse for enabling the cooling and heating load calculation. The prediction and control of all the microclimate parameters will help reduce plant stress, decrease fungus growth, decrease the number of pests, foster an appropriate environment for growing crops, and prolong the network lifetime. However, the prediction of the microclimate in a greenhouse is a challenging task for researchers. Many methods use redundant and periodic sensory data based on historical data; this usually results in low prediction accuracy [83-85].

This chapter proposes an intelligent framework to predict and control the microclimate of a greenhouse and maintain its deployed WSN energy as efficiently as possible for a long time. The framework uses a deep learning model, Long Short-Term Memory (LSTM), to collect and predict five environmental factors: air temperature, relative humidity, air pressure, dew point, and wind data daily. LSTM is an artificial Recurrent Neural Network (RNN) architecture well-suited to classifying, processing, and making predictions based on time series data. There are many different types of classical time series prediction, but these techniques are not suitable in our case because they work well on short-term prediction and does not show its effectiveness for long term data. Also, these techniques are based on existing patterns that will continue in the future. But in the real dynamic nature

of time series data, these assumptions are not valid, which indicates to use deep learning techniques as LSTM. Besides, the input data sometimes in time series suffers from sequence dependence problem which LSTM can resolve it because it is easier to be trained on large hidden architecture and get better results.

The remainder of this chapter is organized as follows. Section 5.2 reviews the recent related works. Section 5.3 states the research problem and lists our contributions. Section 5.4 introduces our proposed framework. Section 5.5 explains the first phase of building a prediction model. The performance evaluation of the LSTM model is covered in Section 5.6. Section 5.7 explains the second phase of the prediction duty cycle, based on the LSTM model, and gives our proposed algorithm. Lastly, in Section 5.8, the performance evaluation for the proposed algorithm is discussed.

5.2 Related Works

In recent years, intelligent solutions based on using machine learning and deep learning technologies have developed rapidly and have significantly contributed to the advancement of prediction models. These models were shown to enhance the quality, accuracy, generalization ability, and robustness of the conventional time series prediction tools. Many models based on regression and the neural network have been built [86-87]. Recurrent Neural Network (RNN) has many applications in speech recognition, machine translation, and time-series data prediction due to its memory capability. The Long Short-Term Memory (LSTM) neural network is based on the development of RNN. LSTM is based on time series of connecting previous information to the present task and having a

very large memory. LSTM can remember information for long periods, making LSTM a good candidate model to forecast the greenhouse microclimate. LSTM performs well when processing long-term dependencies of time series data and predicting long-interval events as in [88]. With the existence of the Internet of Things (IoT) and cloud services, a large amount of environmental data can be saved and accessed, facilitating LSTM model accuracy.

In [89], the authors proposed a predictive solution for disaster monitoring using a neural network-based Multivariate Correspondence Analysis (MCA-NN). The MCA-NN model aims to improve the detection results by combining features from multivariate shallow learning models as described in [89]. Others utilized Cellular Neural Networks (CNN) to monitor desertification. Authors in [90] used CNN to predict the trend of land desertification from 2000 to 2011; the experiment showed that the CNN model is better when they used an exponential smoothing model first before the prediction. The authors in [91] proposed a method based on the Artificial Neural Network (ANN) to predict irrigation requirements using the multi-layer perceptron model to extract the climate information retrieved from the public weather forecast to predict current crop evapotranspiration. In [92], the authors built an Autoregressive Neural Network (AR-NN) model for the seasonal weather, to map the nonlinear relationship of the data collected to get reliable prediction results.

All the previous works and applications mentioned above, depend basically on sensors for collecting data. The sensors deployed in various environments for numerous applications must have a long lifetime, long enough to fulfill the application requirements with high accuracy and efficiency to produce reliable predictions. There are many factors that affect

the quality of the wireless sensor network. One of these factors is energy consumption. Authors of [93] used experimental measurements to show that transmitting data will consume more energy than processing data. While authors in [94], claimed energy could be consumed by sensor components such as the CPU, radio, and microprocessor. Many schemes are proposing to conserve energy via communication [95]. Another approach to energy conservation was to reduce the amount of data transmitted either by compression [96] or aggregation [97]. In addition, many proposed works have been done to save energy by scheduling the sleep/wake-up duty cycle among sensor nodes in the network. In [98], the authors derived an algorithm to increase energy efficiency based on the node's location and the scheduling of node activities. The simulation indicated that the network design was maximizing the lifetime of the sensor. In [99], the authors discussed the parameters that can affect the energy consumption and lifetime, their experiment focused on the effects of different data sizes and changing the duty cycle (sleep, idle, sleep). They concluded that sleep current is an important parameter that reduced the lifetime of the battery by 193 days. Furthermore, they reported that the increase in data packet size would decrease the lifetime of the battery. The authors in [100] proposed a method based on local traffic for the derived distance-duty cycle. They compared three methods: Traffic-adaptive Distance-based Duty Cycle Assignment (TDDCA), Distance-based Duty Cycle Assignment (DDCA) and Constant Duty Cycle (CDC), based on each method's packet delivery ratio (PDR). The results proved that the PDR is almost the same in the three methods for light traffic loads. However, for heavy traffic loads, each method performed differently. Another approach in [101] the authors provided different duty cycles based on the distance between the node and the BS to show the effect of traffic load on the amount of consumed energy. In [102],

the authors found node density impacted energy consumption with different numbers of nodes and duty cycles. In [103], the authors attempted to prolong the lifetime of the node; their experiment concluded that the node is in a wake-up mode when the energy level is above a certain threshold and can transmit, receive, and process data. When the energy level is under the threshold, the node enters sleep mode and is inactive. In [104], the authors determined that device placement cannot be feasible because of the environment or if the number of devices is large. The work involved many attempts that assumed a device could be placed in the sensing field with the goal of optimizing the device placement with respect to system lifetime. In [105], the authors attempted to find the minimum number of relay nodes (RNs) and where the best location would be to meet the constraints of the network lifetime. They used a recursive algorithm by placing the RNs in the intersections of the communication range of the largest number of sensors. In [106], the authors proposed a way to decrease the energy consumption by using B-MAC carrier sense media protocol which reduced the duty cycle to achieve a low power operation.

Some drawbacks we noted in the literature work above are as follows. Most of the works did not include all the microclimate factors that affect the growth of a plant. Most did not consider the prediction of the maximum, minimum, and average of all the microclimates, which helps establish boundaries for the prediction of weather values to enhance the accuracy of the prediction model. Besides, all the proposed works for decreasing the energy consumption of sensors were mostly covering location, the components in the sensor, communication, duty cycle, data size and data type that need to be transmitted. Our proposed framework uses a deep learning model to collect and predict the sensor's duty

cycle which will change and control the sensor's operation that monitors the microclimate inside the greenhouse, saving the energy and prolonging the lifetime of the network.

5.3 Problem Statement and Contributions

The microclimate inside each part of the greenhouse must be monitored frequently and kept under control to avoid any sudden environmental changes that affect the growth of the crop. However, such intensive use of sensors for monitoring will consume much energy and decrease the networks' lifetime. In this chapter, we investigate a solution for solving the following problem:

Determine an efficient solution for monitoring the microclimate to protect crop growth while increasing the lifetime of the WSN

To address this problem, we propose an efficient two-phase framework. The first phase uses the LSTM model with data collected from WSN to control and stabilize the greenhouse atmosphere to ensure good quality crop production. In the second phase, the LSTM model will be used to control the duty cycles of the sensors which will decrease the energy consumption and cost production and increase the network's lifetime.

To this end, the major contributions of this work are listed as follows:

- Proposing an intelligent prediction model Long Short-Term Memory (LSTM) to control and stabilize the microclimate in a greenhouse for better crop production.

The model succeeds in predicting the microclimate of the greenhouse for seven days, 30 days, and 90 days, ahead.

- Designing a novel algorithm, based on our proposed intelligent prediction LSTM model, to predict the sensor nodes operational modes (wake-up and sleep) through their duty cycles. The algorithm succeeds in decreasing the consumed energy and prolonging the network lifetime.

5.4 Proposed Framework

In this section we present the two phases involved in the implementation of the proposed framework. Phase one, building an intelligent prediction model to control the microclimate inside the greenhouse is explained in Section 5.5. The second phase uses the predicted model to control the duty cycle of sensors and is described in Section 5.7. The proposed framework is shown in Figure 5.1.

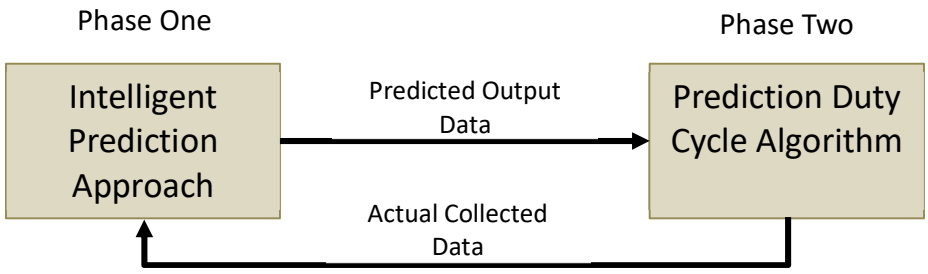


Figure 5.1: Proposed Framework

5.5 Phase One: Intelligent Prediction Approach

In this section, we present our novel approach for building an intelligent prediction model based on Long Short-Term Memory (LSTM) to control the microclimate inside a

greenhouse. Phase one of our proposed approach has four stages as given in Figure 5.2. The first stage involves wireless sensor nodes to sense and transmit microclimate data from inside the greenhouse. The second stage involves collecting all the microclimate data and pre-processing these data. The third stage involves, building the prediction model (training, testing, and validation) until the model reaches a high level of accuracy and meets the requirements to be used inside the greenhouse. The fourth stage involves the prediction values used to control the greenhouse when the sensors are in sleep mode of their cycle and as input values to phase two. All four stages are described in more detail in the following subsections.

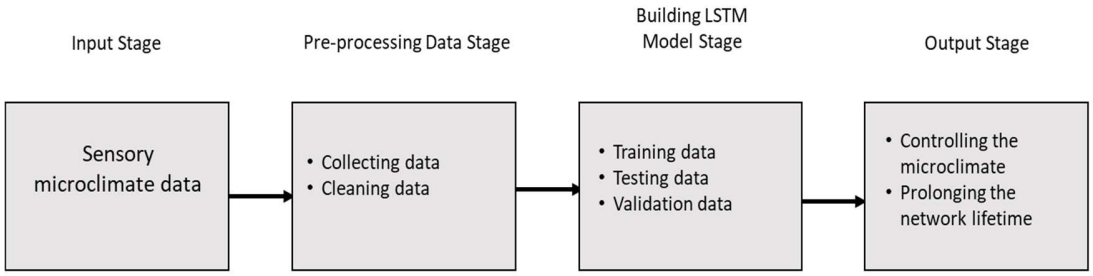


Figure 5.2: Phase one Stages

5.5.1 Deployment of the WSN in the Greenhouse

We consider a two-dimensional greenhouse area of size A = L x W, where L and W represent the length and the width of the greenhouse. The WSN has a group of n sensors S = $\{S_1, S_{2,\ldots,} S_n\}$ connected and deployed in the greenhouse at different locations as shown in Figure 5.3. Without loss of generality, each sensor is connected to a BS in single hop

and transmit data using the Zigbee communication protocol[3]. All sensors have the following characteristics: a limited power supply, equal initial energy, and the same lifetime. We assume that the lifetime for all sensor nodes is known in advance. In our study, we assume that each sensor has the following two modes of operations: wake-up (or active) and sleep, during its duty cycle. Each sensor is responsible for sensing, processing, and transmitting the microclimate data.

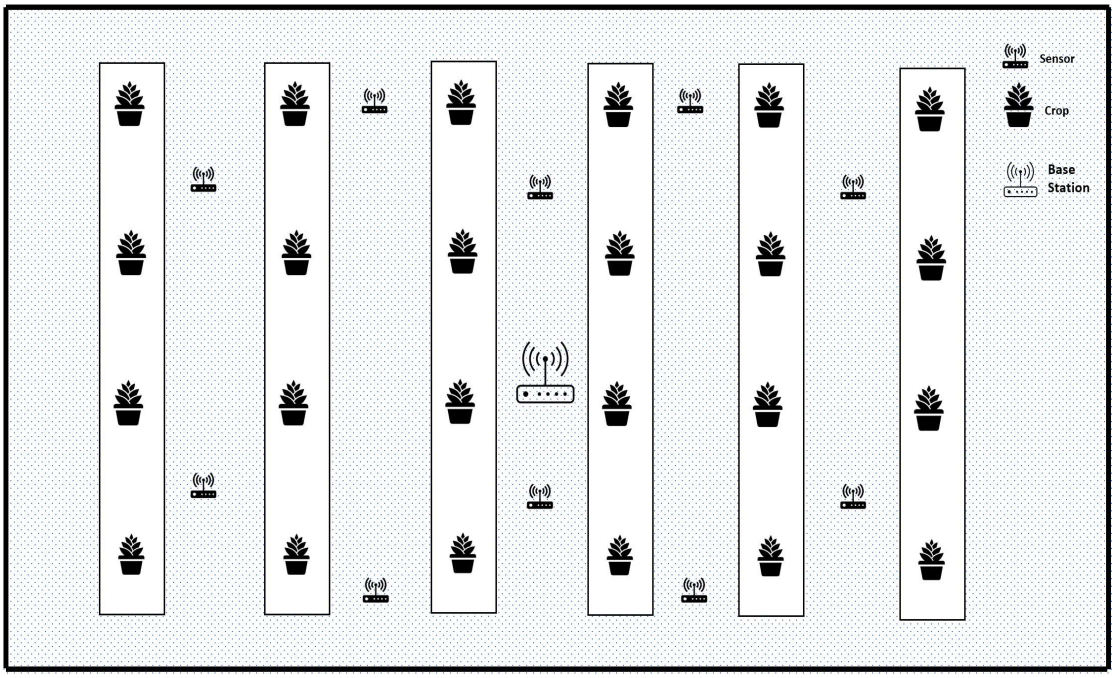


Figure 5.3: WSN Deployment in a Greenhouse

5.5.1.1 Data Collection and Preparation

Our proposed approach is based on a very large dataset of historical weather records. Data collection is the second operational stage of phase one. We utilize a dataset of 1826 records collected by many types of wireless sensor nodes that have been installed inside and outside

[3] Other network architectures/protocols could have been used.

a greenhouse in Surrey, British Columbia, Canada to monitor the microclimate [107]. From this point forward we will refer to air temperature, humidity, air pressure, dew point, and wind as five environmental factors, were recorded and analyzed with the help from Weather Underground, a community of volunteers reporting data taken from specific weather sensors, located in British Columbia. This data has been collected for five years from January 1, 2015 to December 30, 2019, on an hourly, daily, and weekly basis. From that dataset we calculate the maximum, minimum, and mean data for all five factors.

In this study, we used the collected data to train and test our proposed model for weather prediction for 7/30/60 days. The prediction for 60 days ahead is an effective indicator before the sensor run out of battery and give the greenhouse manager the chance to replace the sensor without any damage can affect the greenhouse. The inputs for the model were the maximum, minimum, and mean of the five factors. The output weather predictions were for 7/30/60 days in the future. The sample of the microclimate data is shown in Figure 5.4.

After collecting data from sensor nodes, we prepare the dataset to be fed to the model using a cleaning process. All records should not have missing values. The dataset must be in numerical value. Applying scaling transformation and then normalization on the dataset. LSTM generally improved its performance with the normalized data. We prepare the dataset to LSTM by normalizing the input variables. Normalization using Gaussian distribution is a rescaling of the data from the original range so that all values are within the range of 0 and 1. We accurately estimate the minimum and maximum values. The dataset already comes pre-processed by the community. Any data that has a missing value was replaced with NaN. Also, any duplicated data were discarded. All data are transformed

and scaled for easier use. Following data preparation, cleaned data is input into the LSTM model. The model consists of three steps explained next and illustrated in Figure 5.5.

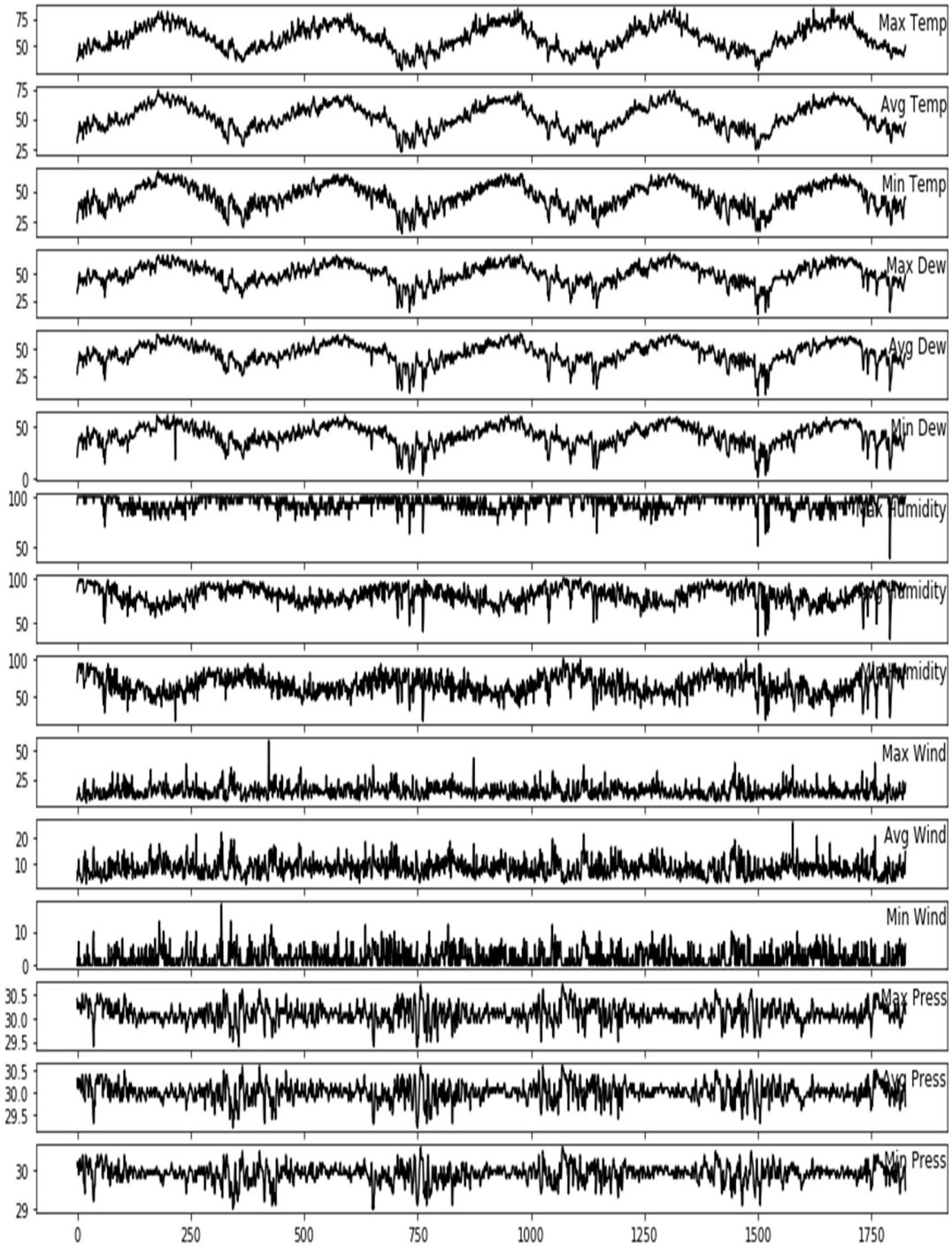


Figure 5.4: **Sample Microclimate Data**

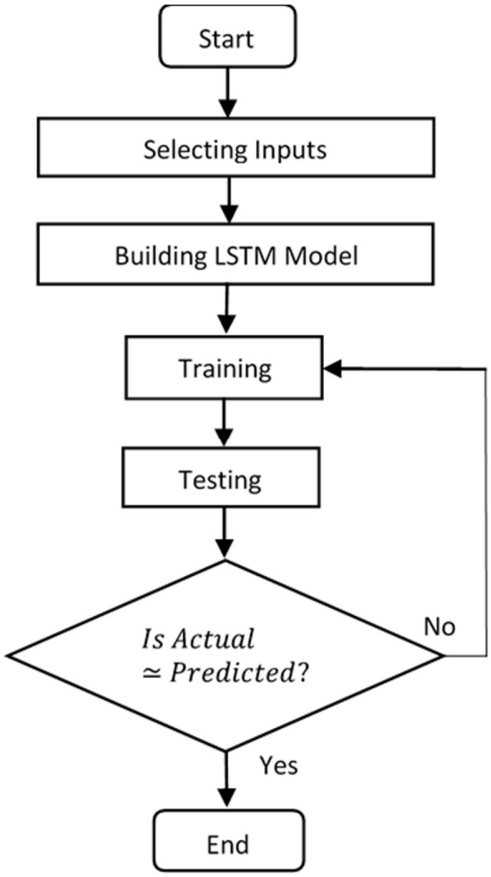


Figure 5.5: Flowchart for Building the LSTM Model

5.5.1.2 Building LSTM Model

LSTM is a type of deep learning RNN architecture. The input of LSTM can be current data and data previously collected. Thus, the input of the LSTM model at time t is the model output at time t-1 along with new input at time t. The model depends on time series to predict the future microclimate in the greenhouse, which is dependent on the previous microclimate of the greenhouse. Data has been collected and used to determine which

prediction is more accurate for the next 7/30/60 days. The prepared dataset is divided into training, testing, and validation sets (80% for training, 10% for testing, and 10% validation). These sets are then split into input and output variables. In LSTM the input changed to a 3D format [features, samples, timesteps]. For predicting all the weather conditions, we define the LSTM with 50 neurons in the first hidden layer and 15 neurons in the output layer. The input shape is a one-time step with 15 features. The model fits for 1500 training epochs with different batch sizes and activation function (Relu).

We can forecast for all the test dataset after fitting the model. We combine the forecast with the test dataset and invert the scaling. Calculating the error score for the model by using original scale of forecasts and actual values. We calculate the Mean Absolute Error (MAE) and the Mean Square Error (MSE). Based on our predictions, and to increase the accuracy of our proposed model, we apply two different gradient descent optimization algorithms Adam [108] and the Stochastic Gradient Descent (SGD) [109]. The LSTM model with those two optimization algorithms is tested, and the results compared by calculating the Root Mean Square Error (RMSE). The best-suited model is selected based on the minimized values of MSE and MAE and used to measure the performance of the model.

After prediction, we can change any environmental factor inside the greenhouse to the desired level and control it by requesting an actuator to initiate an action such as open the windows, switch on the heater, or switch on the fan. Our model was tested on different datasets collected from wireless sensor monitoring. Additionally, our model can predict distinct lengths of time.

There are two parts of the LSTM model: the training part and the prediction part.

A. Training Part

The LSTM training model, shown in Figure 5.6, is used to predict the microclimate inside a greenhouse. Our LSTM model has three layers: an input layer, a hidden layer, and a dense layer. The input layer has 50 neurons and is used to provide input to the LSTM model. The LSTM model input is a vector containing the current (hour, day, or week) for the weather forecast data. This feature vector is denoted by $x_i, 0 \leq i \leq n,$ in the diagram at time t. Our model has 16 hidden layers. The LSTM model's output at time t is an initial parameter vector which is also an input for the model for time t+1. The hidden units are internally connected, where output h_i of LSTM at time t is the input of the next hidden unit, h_{i+1}. The hidden layer is used to adjust the weights assigned to the initial parameters based on the gradient descent difference. The LSTM model output at time t is also the input for the model for time t+1. This is because the LSTM behavior for that next hour's, day's, or week's output is dependent on the previous hour's, day's, or week's output. The last layer is the dense layer. The output of all units in the hidden layer h_i is connected to a dense layer whose output p_i has 15 units, representing each microclimate forecast. These predicted values are then compared with the actual weather forecast's value at that time.

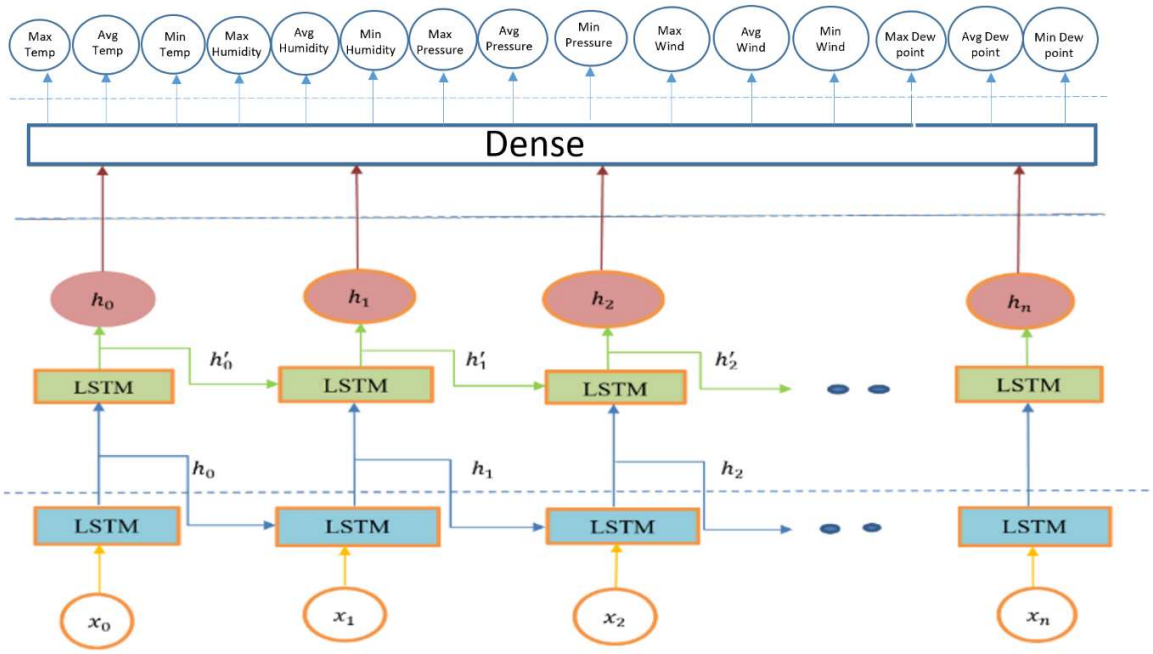


Figure 5.6: LSTM Training Model with Input, Hidden, and Dense Layers

B. Prediction Part

In the prediction phase, we use the trained LSTM model to predict the microclimate forecast for the next 7/30/60 days. Let P_i^t denote the microclimate i at time t. Let $\{F_i^t \ldots F_n^t\}$ be the climate features. Given a tuple (P,F), our goal is to predict $P_i^{t+1}, P_i^{t+2}, \ldots P_n^{t+m}$, where m is the number of days for which prediction is required. The feature vector at time t is the input to the trained LSTM model that predicts the microclimate for the number of days, passed as an argument in the function. The next day's predicted values are appended with the corresponding day's weather forecast data to predict the next day's microclimate. The whole function is recursively called n times, where n is the number of days for which prediction is required. We predict the next m number of days microclimate; instead of training a separate LSTM model for different values of n, one model is trained to predict

the next day's values that are extrapolated for the next day's prediction, as shown in Figure 5.7.

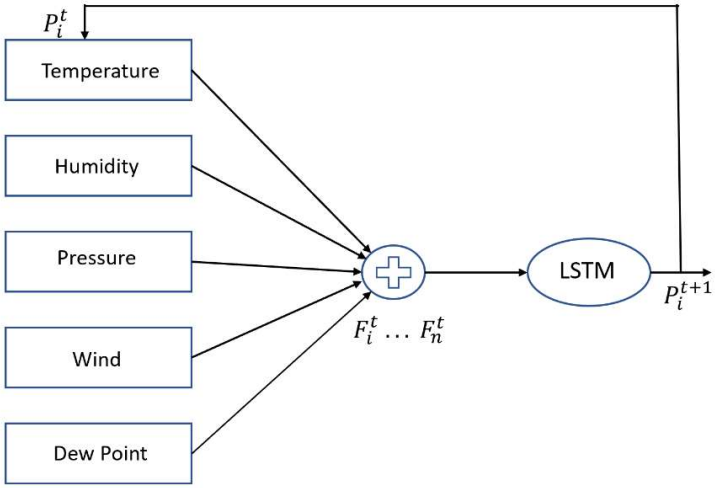


Figure 5.7: LSTM Prediction Model

5.6 Performance Evaluation

In this section, we present the performance evaluation and results' comparisons of our intelligent LSTM model built in phase one. Our proposed model is implemented with multiple python packages including TensorFlow [110] and Keras [111] to perform label encoding and scaling on our dataset, respectively. The LSTM model predicts the five environmental factors over the next number of days. There are multiple parameters (number of epochs, hidden layers, hidden neurons) on which the LSTM model works. Tuning all these parameters results in different RMSE values. We performed several experiments to find the optimal value of the parameters to achieve the least RMSE.

The data is split into training, testing, and validation. To assess the performance of the proposed model, we compare the training data results and validation data in loss function. We used three loss functions to measure how accurately our model can predict the expected outcome. The loss function is a measure of how well our model did at predicting the outcome. A high value for the loss means our model performed poorly. A low value for the loss means our model performed very well. The three loss functions are explained below.

5.6.1 Evaluation Metrics

We used the following three loss functions, MAE, MSE, and RMSE, per Equations 5.1, 5.2, and 5.3, respectively. RMSE is the difference between the microclimate weather values predicted by a model and the values observed. Because the model is trained on past data, we report the RMSE for future microclimate weather prediction values. In Equations 5.1-5.3, s represents the test sample size. The MAE and MSE results are shown in Figure 5.8 and Figure 5.9, respectively. From here on, we use the MAE in our results. Because the MSE loss function square the error and it will take time to reach the minimum, rather than MAE loss function is subtracting the error and will be faster to reach to the minimum.

$$MAE = \frac{1}{s}\sum_{i=1}^{s}|Predicted_i - Actual_i| \quad (5.1)$$

$$MSE = \frac{1}{s}\sum_{i=1}^{s}(Predicted_i - Actual_i)^2 \quad (5.2)$$

$$RMSE = \sqrt{\frac{1}{s}\sum_{i=1}^{s}(Predicted_i - Actual_i)^2} \qquad (5.3)$$

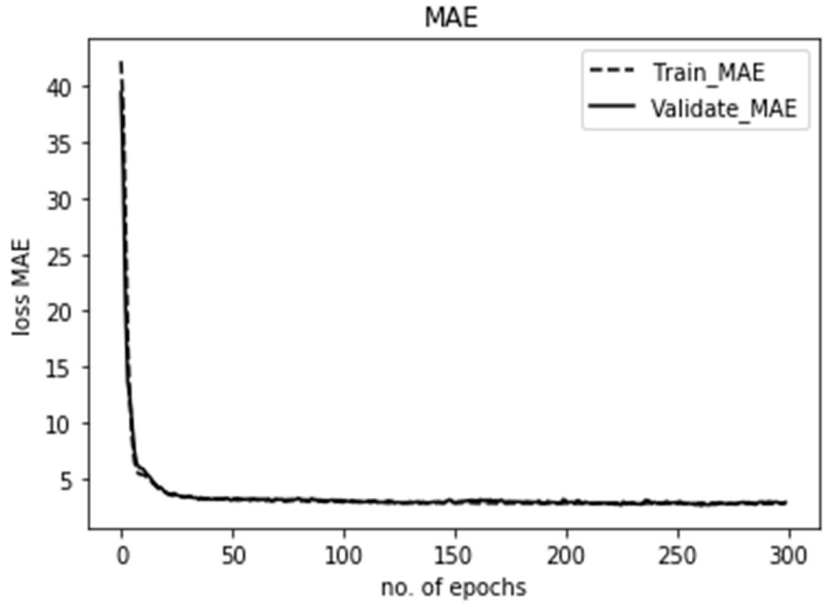


Figure 5.8: MAE Result.

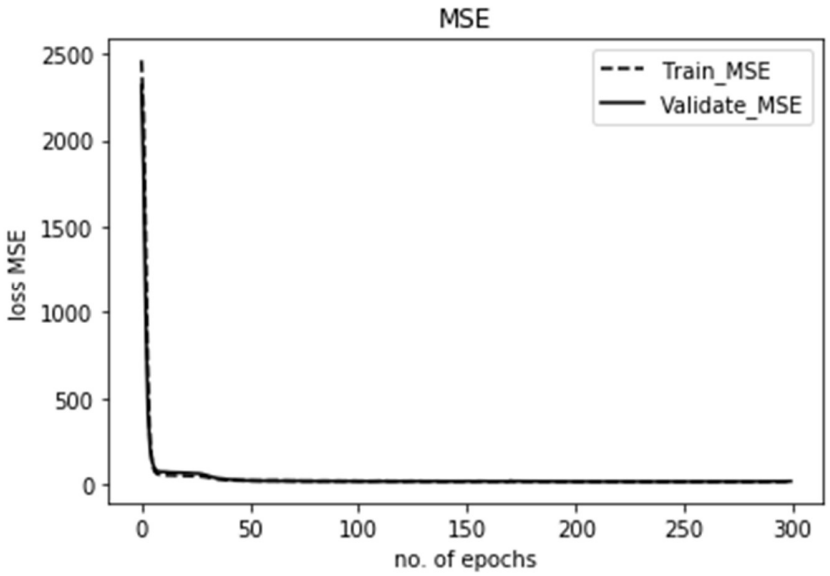


Figure 5.9: MSE Result.

5.6.2 Results and Discussion

Based on our predictions, and to increase the accuracy of our proposed model, we apply two different gradient descent optimization algorithms Adam [108] and the Stochastic Gradient Descent (SGD) [109]. The advantage of the Adam algorithm over SGD is that global minima are achieved faster with fewer epochs, as shown in Figures 5.10-5.13.

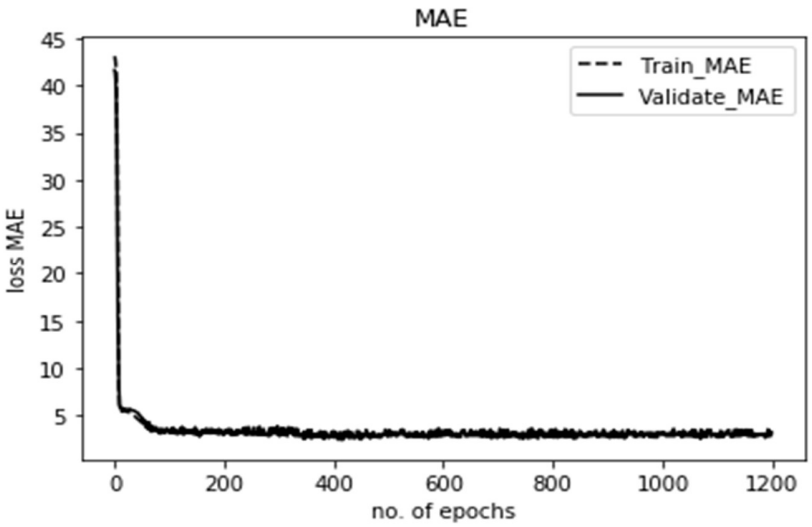


Figure 5.10: 50 Neurons, SGD

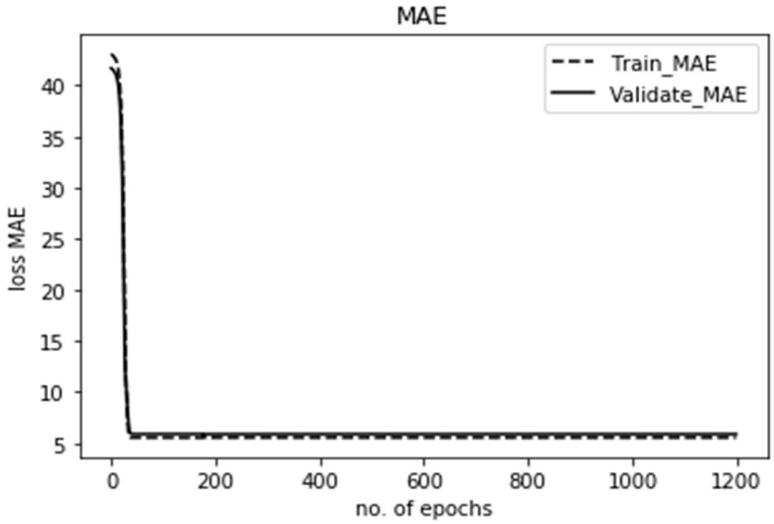


Figure 5.11: 5 Neurons, SGD

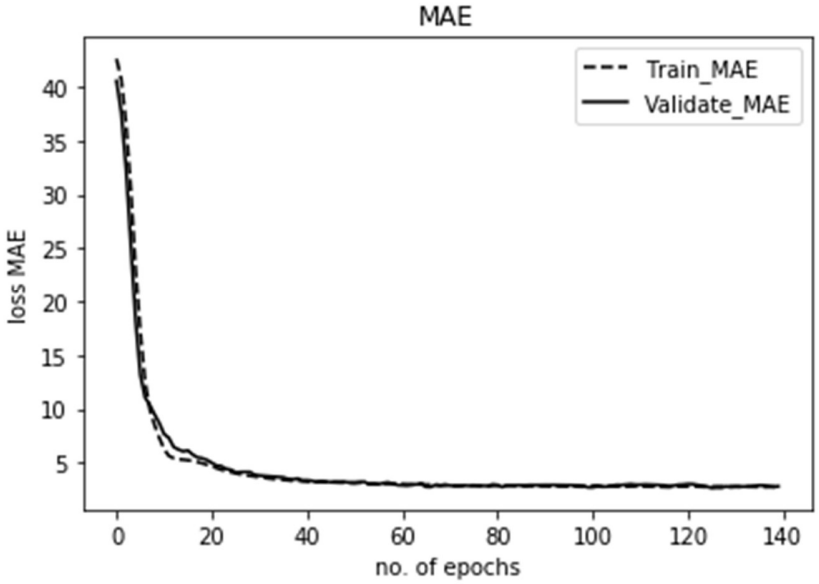


Figure 5.12: 50 Neurons, Adam

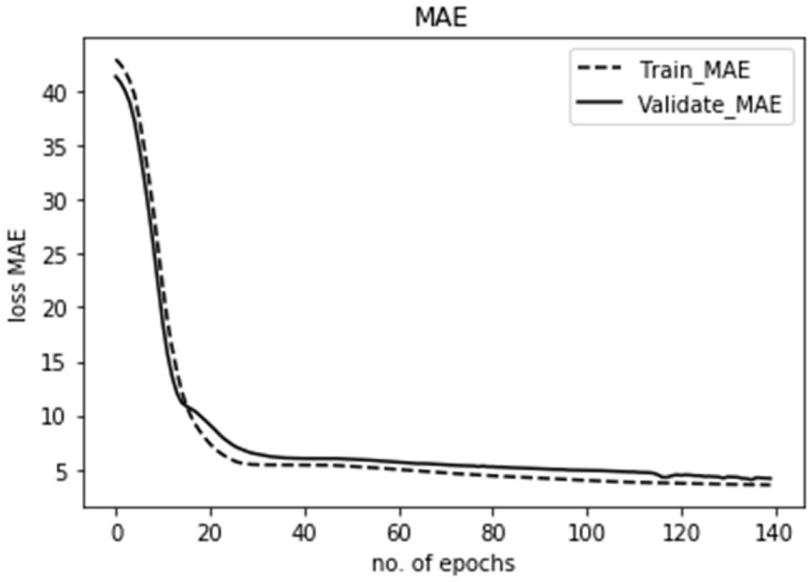
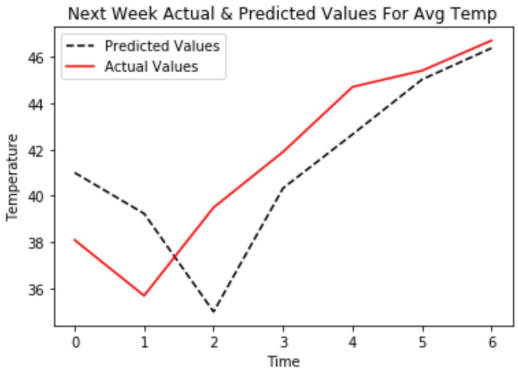


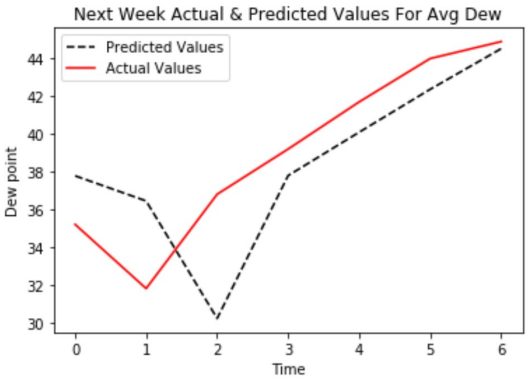
Figure 5.13: 5 Neurons, Adam

Figures 5.10, 5.11, 5.12 and 5.13 show which optimization algorithm reaches global minima, with fewer epochs, faster when there is different neurons input. From the figures, the global minima are achieved with the least number of epochs when the number of neurons is higher. The figures also illustrate that 50 neurons perform better than five neurons in terms of achieving global minima. In Figure 5.12, the use of 50 neurons and the Adam algorithm achieved the global minima in fewer than 30 epochs, while in Figure 5.10, the use of 50 neurons and the SGD algorithm achieved the global minima in 200 epochs. We used the Adam algorithm with one hidden layer, 50 hidden units, and 150 epochs in our LSTM model to obtain the global minima. We did not define the learning rate in the Adam algorithm since it already calculates the individual adaptive learning rate for each parameter.

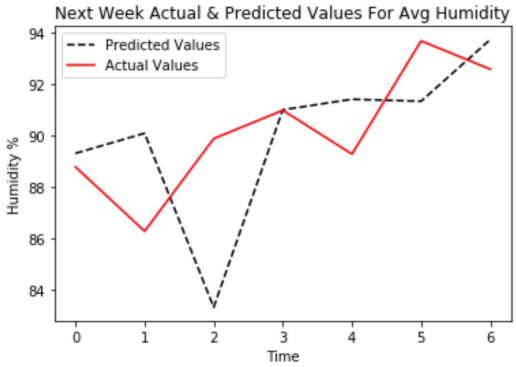
The RMSE results for 7/30/60 days are shown in Figures 5.14, 5.15 and 5.16. Each figure represents the five environmental factors, for the predicted and actual data. From the results [112], we can see that the proposed model can predict the future weather inside the greenhouse with a high level of accuracy because the performance of the model is good on both the training and validation sets (good fit) as shown in Figure 5.8. It is noticeable that the prediction accuracy is almost the same in all the three figures. More detailed results are included in the Appendix B.



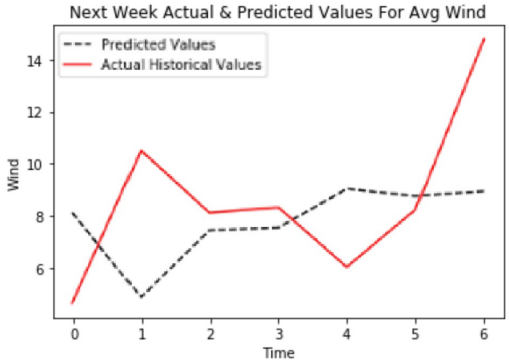
Test RMSE: 2.619



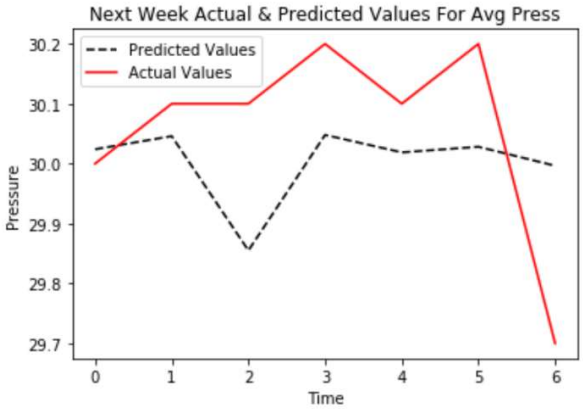
Test RMSE: 3.360



Test RMSE: 3.143 %

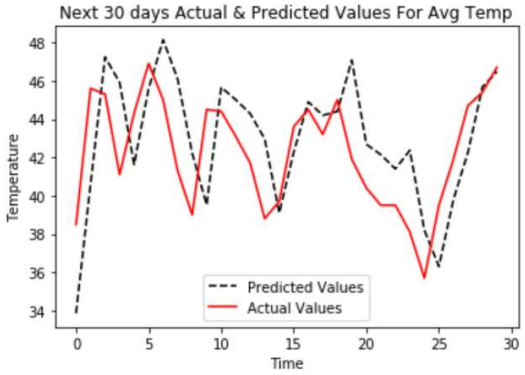


Test RMSE: .281

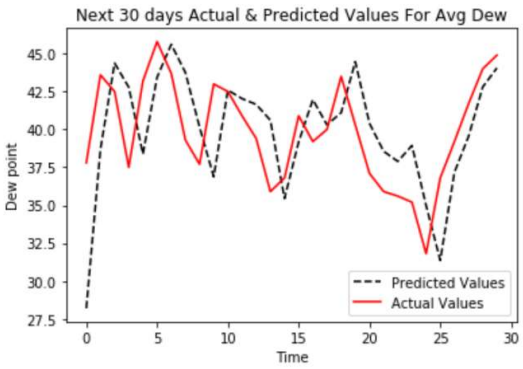


Test RMSE: 0.173

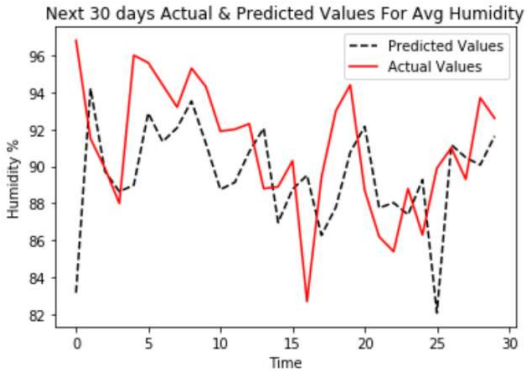
Figure 5.14: Prediction for Seven days Ahead



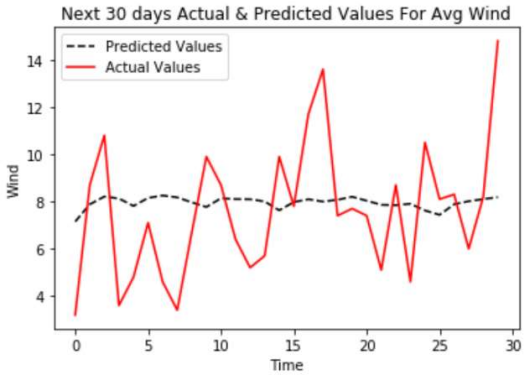
Test RMSE: 3.005



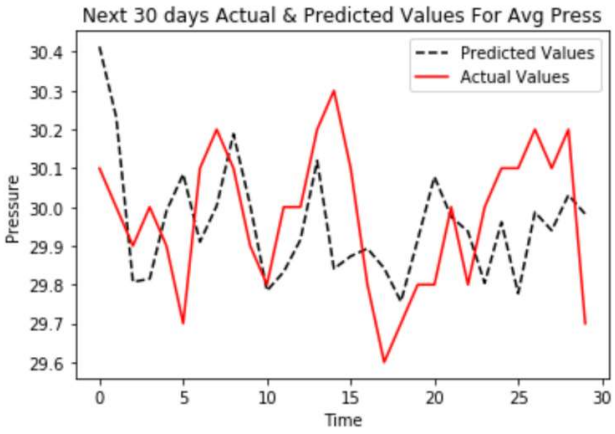
Test RMSE: 3.632



Test RMSE: 4.146 %

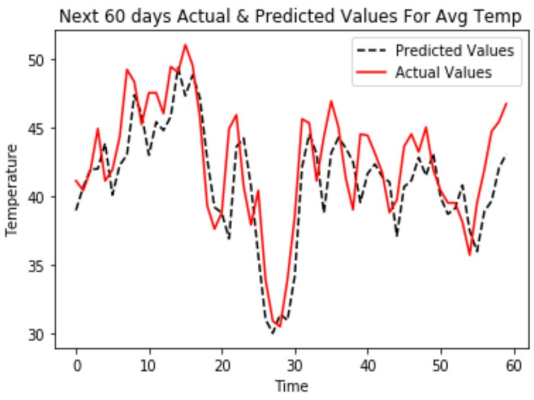


Test RMSE: .826

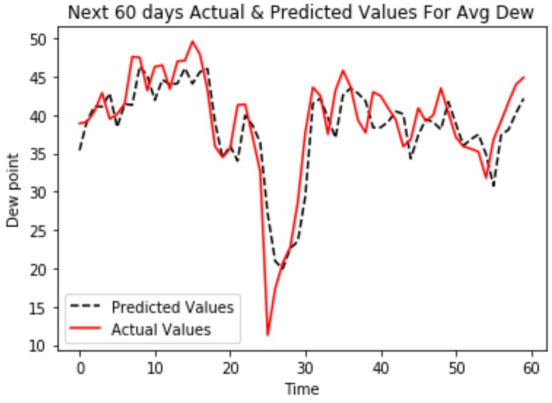


Test RMSE: 0.206

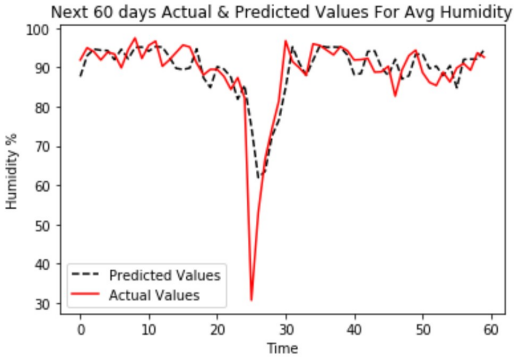
Figure 5.15: Prediction for 30 days Ahead



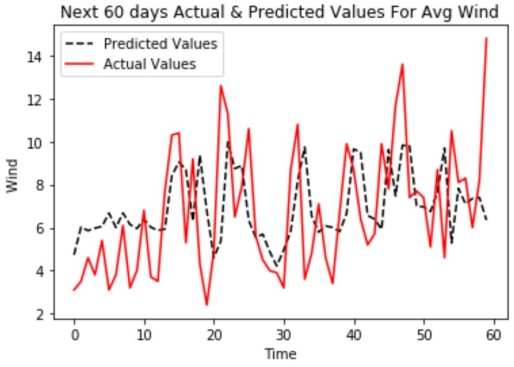
Test RMSE: 2.969



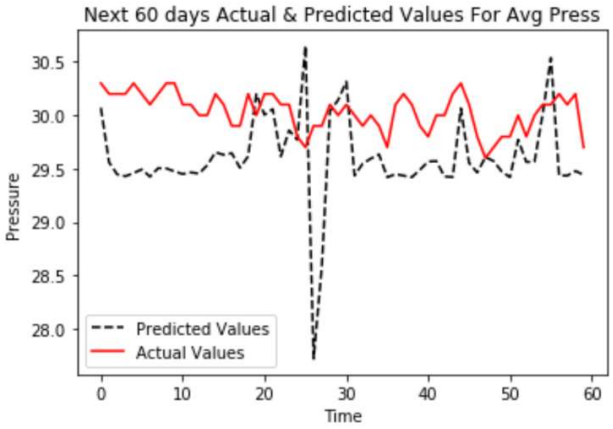
Test RMSE: 3.835



Test RMSE: 6.857 %



Test RMSE: .873



Test RMSE: 0.602

Figure 5.16: Prediction for 60 days Ahead

5.7 Phase Two: Prediction Wireless Sensor Nodes Duty Cycle

In phase two of our proposed framework, we propose an algorithm called Prediction Duty Cycle Algorithm (DCPA) to decrease the consumed energy of the wireless sensor and camera sensor nodes and prolong the network lifetime by predicting and controlling the duty cycle of the nodes during the crop life cycle in a greenhouse. Data input for the DCPA is taken from the LSTM model, which is developed in phase one.

In the remainder of this section, we first explain the plant growth cycle in a greenhouse to understanding of the natural growth cycle of a plant which is important when making decisions that affect the overall efficiency of greenhouse crop production. Without loss of generality, we explain this cycle using the growth of a tomato plant. We then explain in detail the steps of our proposed algorithm, DCPA.

5.7.1 Case Study: Tomato Plant

In this research, we focus on the tomato crop as a use case for understanding its growth cycle that help in building our model. The optimum microclimate levels for the best greenhouse cultivation of tomatoes depend on different growth stages and conditions. There are five stages of growth in a tomato: germination and early growth with initial leaves take between 25-35 days, the vegetative period between 20-25 days, the flowering period 20 to 30 days, the early fruiting period between 20 to 30 days, and the mature fruiting period between 15-20 days [113]. The exact period of days depends on the atmosphere inside the greenhouse. For most greenhouse tomatoes to reach maturity and ripeness is between 65 to 100 days. Shortening the production time can be done by

changing the conditions inside the greenhouse. The growth stages of a tomato plant are graphically presented in Figure 5.17 along with fruit maturity and ripeness levels. It should be noted that tomatoes are harvested when they have reached the mature green stage (vine-ripe), which is just as they start to ripen.

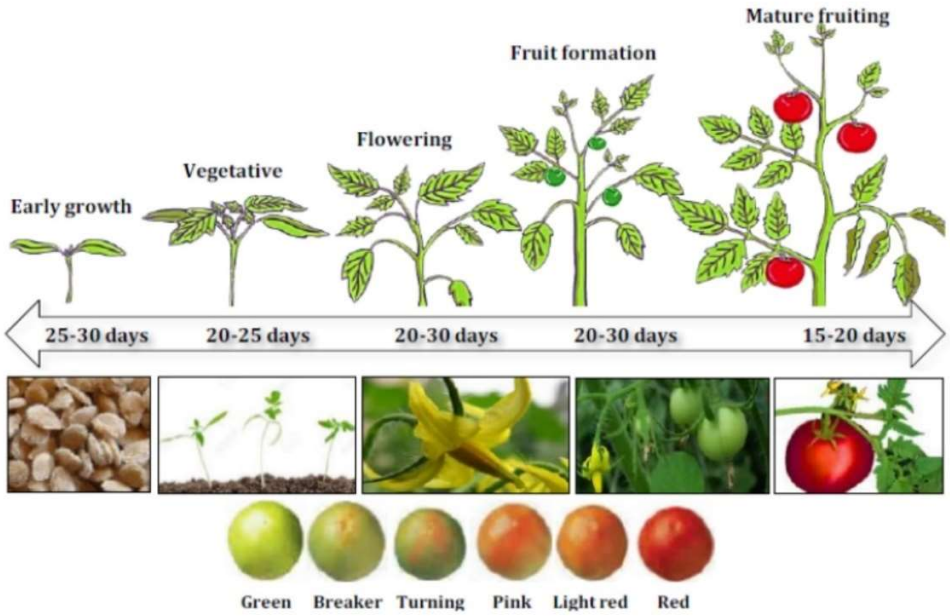


Figure 5.17: Five Growth Stages for Tomato Plants

Air temperature higher or lower than the optimal values affects different stages of tomato growth and development. High temperatures cause the fruit to die because of improper transpiration and can destroy the plant. Tomato plants are notably sensitive to above optimal air temperatures during the reproductive stage and may face a reduction in the percentage of fruit set, which triggers a significant yield decrease in commercial cultivation. On the other hand, the decreasing temperature will affect respiration and photosynthesis, causing a hormone imbalance in the tomato plant. In addition, a very high

level of humidity in the greenhouse environment causes diseases and fungal pathogens to grow and spread rapidly infecting the plants. Pests also spread faster in high temperatures and humidity. Some of the diseases and physiological abnormalities associated with high humidity in greenhouse production include black spot, powdery mildew, leaf-edge burn and blossom-end rot of tomatoes. As well, plants evapotranspiration may be limited. Barometric pressure directly affects the water uptake by tomato plants and likewise the overall tomato fruit yield. Extremely high- or low-pressure values can result in leaf physiology disorders and the fruit to die.

By knowing the duration of the plants' growth, we can control the greenhouse environment to speed up the growth of the plants and also to protect them from diseases. Predicting greenhouse optimal parameters for seven days after seeding, which is when the seeds germinate (sprout), and 30 days is the duration of each stage in tomato growth, allowing growers to control the climate inside the greenhouse during these periods of growth. It will also reduce energy consumption for wireless sensor nodes which in return will reduce network deployment and maintenance costs.

5.7.2 Proposed Duty Cycle Predicting Algorithm (DCPA)

Based on the literature reviewed, there remains a gap in the research that answers the question: How can you decrease the amount of energy consumed by wireless sensor nodes. To solve this problem. We build the LSTM model as explained in section 5.5. Then, use our novel algorithm DCPA based on LSTM to predict and control the monitoring sensors'

duty cycle. Using our proposed DCPA will regulate between wake-up/sleep mode for the sensors depending on the prediction model.

Our proposed algorithm's primary goal is to predict the operation of the sensor nodes through the duty cycle to make sure that the nodes are not active continuously when monitoring the greenhouse. Thus, reducing the consumed energy and prolonging the WSN lifetime. DCPA uses the output features' values from the LSTM prediction model as an input. Each sensor can be in active mode or in sleep mode depending on the difference between the actual measured value and the predicted value. DCPA works as given in Algorithm 5.1.

Sensors at time t start in an active mode and sense any of the features (i.e., environmental factors), A_t, inside the greenhouse and compare it with predicted value, P_t, from the model. All predicted values are stored in an array of size n, *P[n]*. The sensor node will calculate the absolute difference between actual value and predicted value at time t. If the difference is greater than a predefined threshold value, then the sensor will continue to be active and will send the new value to the BS to update the prediction model. For example, the threshold value equal to an acceptable temperature range as in [114]. The BS then will send a job request to an actuator to do an action to reduce the current feature value inside the greenhouse. For example, switch on the fan for a specific period of time if the air temperature or humidity levels exceed the threshold. The sensor node will continue sensing and comparing until sensor node runs out of energy or the difference is less than the threshold. If the difference is less than then threshold, then the sensor node will sleep for a period of time, ST_t. The value of ST_t is determined based on the feature value increase or decrease in the greenhouse. This can be calculated using a sleeping time indicator *STI*,

which is the difference between A_t and P_{t+PST_t}, where PST_t is the proposed sleeping time at t and P_{t+PST_t} is the predicted feature value at $(t + PST_t)$ from $P[n]$. If STI is less than zero, this means the feature value is decreasing and thus, the sensor node can sleep as proposed, i.e., sleeping time is equal to proposed sleeping time ($ST_t = PST_t$). While, if STI is greater than zero, this means the feature value is increasing and the sleeping time should be less than the proposed time as follow $ST_t = \left\lfloor e^{-\frac{STI}{10}} \times PST_t \right\rfloor$.

Table 5.1 defines the symbols used in Algorithm 5.1.

Table 5.1: Algorithm 5.1 Symbol Definitions

Symbol	Definition
$P[n]$	Array of predicated features' values for n hours
P_t	Predicted feature value at t
A_t	Actual feature value at t
$Diff_t$	Feature difference value between A_t and P_t
PST_t	Proposed sleeping time at t
P_{t+PST_t}	Predicted feature value at $(t + PST_t)$
STI	Sleeping time indicator, which is the difference between A_t and P_{t+PST_t}
ST_t	Sleeping time at t
Th	Predetermined threshold value

Algorithm 5.1: Predict Duty Cycle

Input: Array of predicted features' values for n hours, $P[n]$, and actual feature value, A_t, at time t

Output: Sleeping time at t

// Do for each sensor at time t = 1:n

do

Determine P_t from $P[n]$

Sense and measure the current actual feature value, A_t

Calculate $Diff_t = A_t - P_t$

if ($Diff_t <= Th$) **then** // Sensor sleeping mode

Calculate $PST_t = Trunc \mid A_t - P_t \mid$

Determine P_{t+PST_t} from $P[n]$

Calculate $STI = P_{t+PST_t} - A_t$

if ($STI > 0$) **then** // Feature value will increase

$$ST_t = \left\lfloor e^{-\frac{STI}{10}} \times PST_t \right\rfloor$$

else // Feature value will decrease

$$ST_t = PST_t$$

$$t = t + ST_t$$

end if-else

else // Sensor active mode

Transmit A_t to the BS

BS sends a job request to the actuator to adjust the feature value inside the greenhouse

Update the prediction model based on the value of A_t

$t = t + 1$

end if-else

while ($t <= n \;||\; Diff_t > Th$)

end do-while

5.8 Performance Evaluation

In this section, we evaluate the performance of sensor node operation using our proposed algorithm, DCPA, and compare it with two different operational methods: Short Duty Cycle (SDC) and Long Duty Cycle (LDC). The sensor nodes in both methods, SDC and LDC, will alternate between the two operational modes, active and sleep, for different lengths of time. During the active period, the nodes can sense the microclimate factors of the greenhouse, process, and send data to the BS. In our study, we consider that the nodes in SDC in an active mode for an hour and then in sleep mode for the next hour. While, the nodes in the LDC sleep for six hours and wake up for one hour. Thus, on a period of 24 hours, the SDC nodes are active for 12 hours and sleep for another 12 hours in total. On the other hand, the LDC nodes will sleep for 20 hours and be active for 4 hours.

A WSN enabled greenhouse, shown in Figure 5.3, is considered in this simulation with the defined parameters and values given in Table 5.2. Figures 5.18-5.20 show the simulation comparison results between the three methods in terms of the following performance metrics running for 24 hours:

- Total energy consumption: The estimated consumed energy, E_c, for sensing, computing, sending, receiving, and sleeping as in [41]. Thus, E_c is the summation of estimated energy consumed in sensing (SsE), energy consumed in computing (CpE), energy consumed in sending (SnE), energy consumed in receiving (RcE), and energy consumed in sleeping (SpE), for all n sensors. It can be calculated using the simple model in Eq. 5.6. Note that other more elaborate energy consumption

models could have been used, but we are more concerned with relative energy consumption as opposed to absolute values.

$$E_c = \sum_{i=1}^{n} (SsE + CpE + SnE + RcE + SpE)_{S_i} \quad (5.6)$$

- Network lifetime (or remaining network energy): is defined as the total remaining energy of all sensor nodes of the network over a certain period of time. Let us assume that the initial network energy or energy of all nodes is equal to E_{in}. We estimate the energy consumptions of all sensor nodes in the network to be E_c for a certain period of time. Then, the network lifetime is estimated as $E_{in} - E_c$, the difference between E_c and E_{in}.
- Unreported data: is number of significant changes in the environmental factor value that is not reported to the BS since the sensor node was in sleep mode according to its duty cycle activities.

Table 5.2: Simulation Parameters.

Parameters	Values
Greenhouse area	100m x 100m
Number of sensor nodes	10
Base station position	(50, 50)
Initial node energy	2 J
Sensing energy	0.15 J
Computing energy	0.2 J
Sending energy	1.05 J
Receiving energy	0.5 J
Sleeping energy	0.01 J

5.8.1 Results and Discussion

Figure 5.18 illustrates the total estimated energy consumption using the three methods. We note that the SDC operational method's energy consumption is the highest, while energy consumption is the lowest for the LDC operational method. Sensor nodes consume more energy in the SDC method due to the frequent active every other hour for one hour for the sensor nodes where they have to sense, process, and transmit. While less energy is consumed using the LDC method since the sensor nodes are in sleep mode most of the time and thus, save more energy. It is noted that, in Figure 5.18, our proposed algorithm, DCPA, performs much better than the SCD and very close to the LDC method. The reason is that the sensor nodes will be active only when there are significant changes in the greenhouse microclimate conditions. Thus, more control over the operation of the nodes and thus, saving more energy.

Figure 5.19 shows the total remaining energy using the three methods. The DCPA achieves a better network lifetime compared to the SDC method and very close to the LDC method. The results in this figure complement the results obtained in Figure 5.18 and based on our assumptions that all sensor nodes have equal initial energy and same lifetime, as explained in Section 5.5.1. The more time the sensor nodes sleep, as in the LDC case, the more energy is saved; thus, longer network lifetime is achieved. This is not the case when using the SDC method where more energy is consumed which shortens the network lifetime. Our algorithm shows reasonable results based on its ability to predict and control the sensor nodes' operation based on real-time collected data making it a more practical solution to implement.

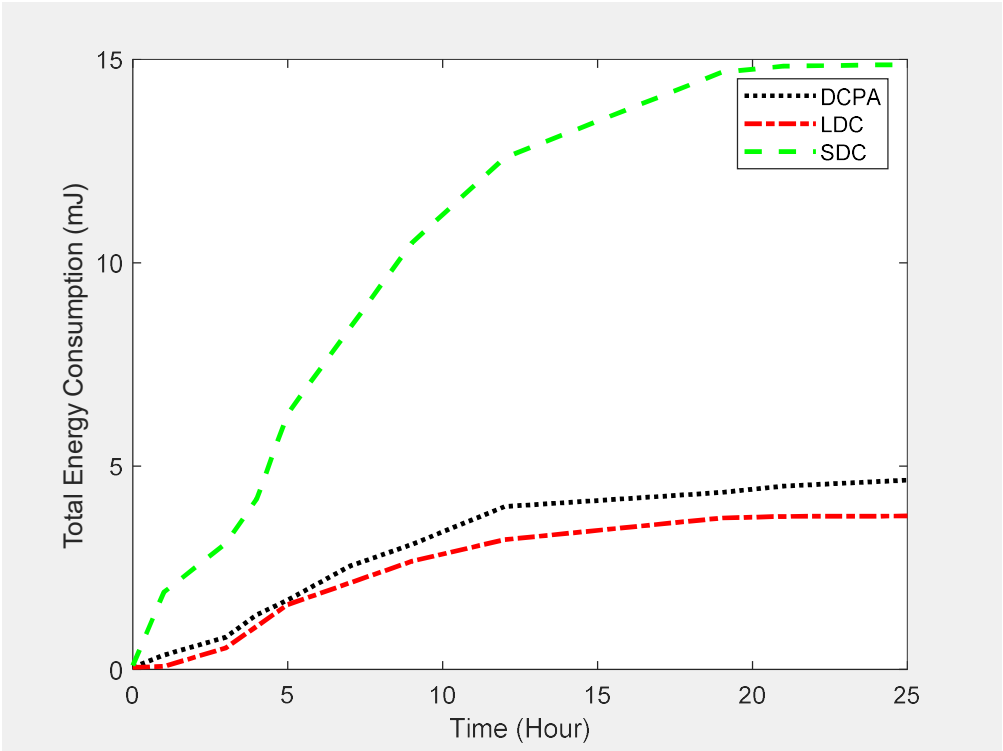


Figure 5.18: Energy Consumption of Node in each Round

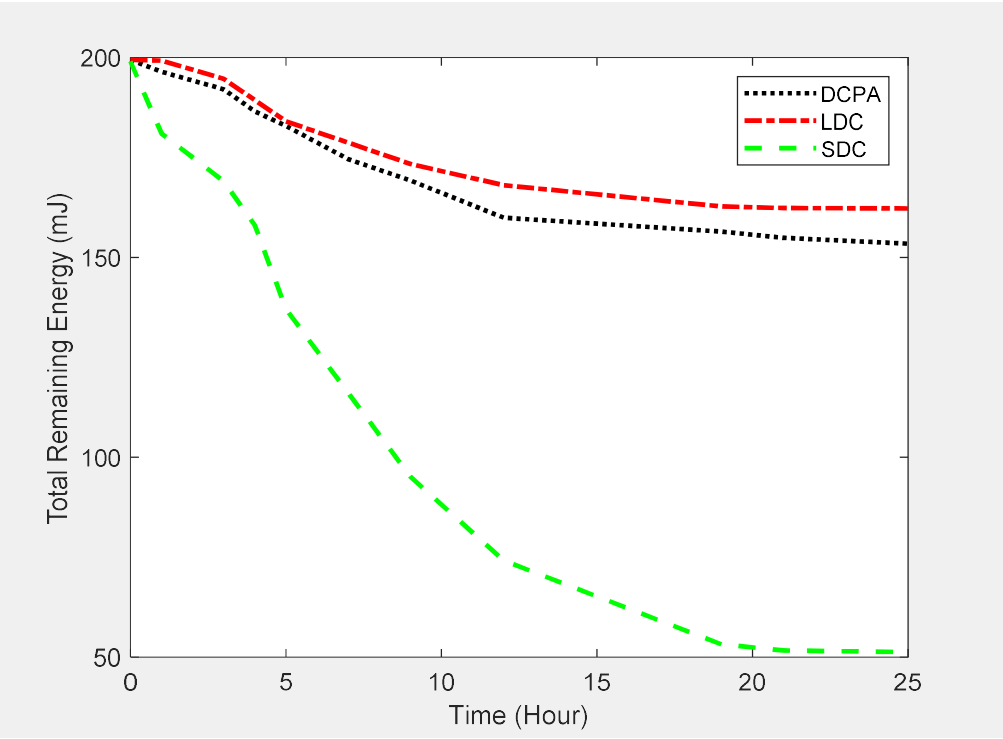


Figure 5.19: Lifetime of a Sensor Network

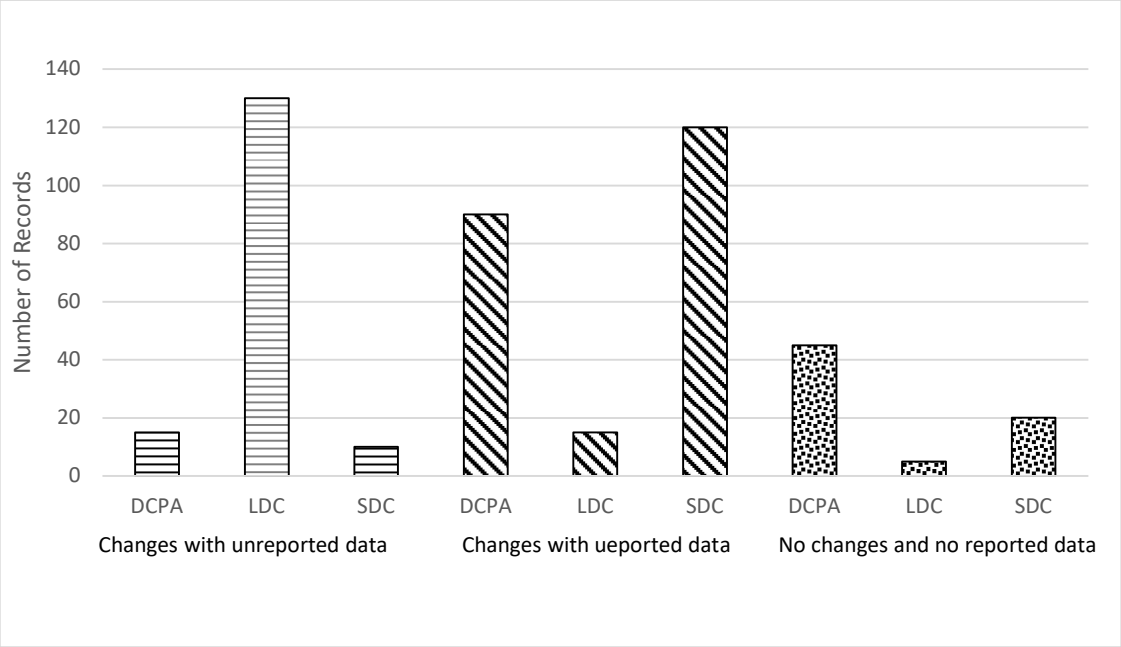


Figure 5.20: Reported and Unreported Comparison Results Between DCPA, SDC, LDC

Figure 5.20 illustrates the comparison results between the methods based on three cases: (1) significant change occurs in the greenhouse microclimate condition but this change is not reported to the BS since the sensor node is in sleep mode, (2) significant change occurs in the greenhouse microclimate condition and this change is reported to the BS since the sensor node is in an active mode, and (3) no change occurs in the greenhouse microclimate condition when the sensor did the comparison between actual and predicted values, and thus, there is nothing to report to the BS. In case one, unreported data, the LDC method achieved the highest value since the sensor nodes are in sleep mode most of the time making them fail to report any significant changes in the greenhouse. While, the SDC has the lowest unreported data because the nodes are active most of the time and can report any significant changes that are happening in the greenhouse. Interestingly, the DCPA has

fewer missing data, since the nodes are operated by executing our intelligent algorithm with high accuracy to predict more realistic sleeping time based on the greenhouse microclimate conditions. The nodes using the DPCA are awake when there are significant changes in the greenhouse and sleep mode when no significant changes are happening. Results obtained in cases 2 and 3 are straightforward and confirm the ability of the three methods to report significant changes when sensors are in active mode and save energy when there are no significant changes to report efficiently. The DCPA result in case 2 is close to the result of the SDC. In addition, the summation of reported data and unreported data when there is a change for both methods, DCPA and SDC, have almost the same value. High result value for the DCPA in case 3 confirms the ability of the LSTM model to predict the microclimate changes in the greenhouse accurately.

We conclude from the above results, Figures 5.18-5.20, that our proposed algorithm, DCPA, achieves better realistic and more robust performance results in terms of energy consumption and network lifetime than both SDC and LDC methods. This is because sensor nodes using DCPA consume energy in an optimistic way; they are only in an active mode when there are changes in the greenhouse that require the nodes to send messages to the BS. As sensor nodes using DCPA know ahead what will happen in the greenhouse microclimate condition based on the proposed LSTM prediction model. In phase one, we showed that the LSTM model has a high level of accuracy where predicted values are almost the same as actual values most of the time. By using the predicted values from the LSTM model as input for the DCPA this ensures that sensor nodes will not miss reporting significant changes in the greenhouse microclimate conditions to the BS. This is because

the nodes will be for the majority of the time in the right mode of operation. Also, it is noted that sensor nodes that use DCPA consume energy wisely which means prolonging the lifetime of the sensors and thus, they can be used for longer periods of time ideally for longer than the lifecycle of the growing crop. This will reduce the cost of both WSN deployment and maintenance. Keep in mind that each location for each sensor is different. This means if some sensors are in sleep mode because they did not sense any changes, other sensors can detect the changes and report it and update the prediction model. This increases the efficiency of the network that have DCPA.

These results confirm the ability of the sensor nodes to use DCPA to operate efficiently in different conditions and save time to fix any critical issues that may happen inside the greenhouse. Making it more practical than the other two methods SDC and LDC.

5.9 Summary

In this chapter, we introduced a prediction-based framework for monitoring microclimate parameters and controlling the sensor nodes in a greenhouse for having high-quality production of crops with less cost. WSN is deployed inside the greenhouse to collect the microclimate data. The LSTM model with different optimization algorithms are applied to the training and testing of environmental data that is collected over a five-year span. The accuracy of the model performance, as evaluated by measurements of AME, MSE and RMSE, is high. The obtained prediction results for 7/30/ 60 days ahead are promising and are significantly useful in predicting and controlling the sensor nodes' duty cycles (active

mode and sleep mode) inside the greenhouse. The proposed DCPA is used to control and predict the duty cycle of the nodes during the crop lifecycle. We showed through simulation that our proposed algorithm, DCPA, performance is robust and more practical than the existing two methods, SDC and LDC. The estimated energy consumption and network lifetime are better in DCPA compared to the SDC method. Also, the DCPA has fewer missing data reported compared to the LDC method.

This research can efficiently facilitate the deployment of WSN and using intelligent prediction solution, in the greenhouses, not only to decrease the energy consumption of the WSN nodes and the production cost but also will predict, control, and moderate the microclimate inside the greenhouse and thereby prevent any incident that would adversely affect the crop from happening. We, therefore, advocate adopting the deployment of this framework whenever possible in practice.

Chapter 6

Conclusions and Future Works

The last decade has witnessed an increasing implementation of greenhouse applications using WSNs and WVSNs due to the combined potential of these two technologies, especially in monitoring, detection, and controlling applications. However, these essential tasks need particular efficiency in a greenhouse setting to ensure data availability, high-quality image resolution, maintain the microclimate stability, and timeliness delivery, within a realistic cost. In literature, existing schemes in this research area are limited to specific cases and deliver results that are far from optimum. To the best of our knowledge, no cohesive framework exists that considers all the issues involved in a greenhouse application, including uncertainty in detecting plant diseases, prediction accuracy, optimizing placement, coverage, and resolution for monitoring and controlling devices. This chapter concludes the thesis with a summary of our research contributions in Section 6.1, as well as future research directions outlined in Section 6.2.

6.1 Research Contributions Summary

The work in this thesis proposed a realistic and practical framework that utilizes WSN, WVSN, machine learning, deep learning, and image processing technologies to satisfy greenhouse-specific network properties.

The framework is divided into three phases. In the first phase, we succeeded in finding the minimum number of camera sensors, and the optimal placement of each camera sensor, to

cover a large area in the greenhouse with high-resolution quality and no overlap between images. High-resolution images were enhanced, then image processing techniques were applied for the segmentation process and machine learning to recognize and identify powdery mildew fungus in our proposed automated system. Phase two achieved high levels of accuracy that confirmed the system's ability for its intended purpose. In the last phase, our proposed prediction system efficiently demonstrated saving time and effort in preventing fungus or an infestation of pests by controlling and predicting the microclimate ahead of time. Thus, increasing the quality and quantity of the greenhouse crop. Also, our novel proposed algorithm, DCPA, at this phase, proved decreasing the extensive work of the wireless sensors, by predicting and controlling their duty cycle activities, which effectively decreases the consumed energy, prolong the network lifetime, and reduce deployment and maintenance costs.

Our proposed framework is composed of three phases as presented in Chapter 3 (Phase 1), Chapter 4 (Phase 2), and Chapter 5 (Phase 3). In Chapter 3, we formulated an optimization problem for defining the optimal placement for the sensor cameras to satisfy four main criteria, minimizing the number of sensor cameras, maximizing the covered area, maximizing image resolution, and avoiding overlap image views. This problem is considered an NP-complete. We used ILP formulation and divided the objective function into two sub-functions to solve the problem. The first function considers maximizing the quality of the image and the area being covered. The second function considers avoiding the overlap between images and minimizing the number of deployed sensor cameras. The

experimental results for both functions were satisfactory, showing great promise as future applications.

In Chapter 4, we proposed a novel automated detection system that used the high-resolution output images taken from the sensor cameras and good coverage of the area. The system used the Hough Forest machine learning technique and image processing methods to enhance these images. The system was able to identify and recognize the early signs of powdery mildew fungus from the images with high accuracy levels. The cross-validation result is 96.69%, which is high enough to show the excellent system performance. The fact that we achieved statistical results with a detection rate of 94% is another indication of the strength of the proposed system.

In Chapter 5, we introduced a framework with two stages. In the first stage, we used deep learning with WSN to control and predict the microclimate. We proposed building the LSTM prediction model based on collected data from wireless sensors. Our proposed model's accuracy was tested and validated using MAE, MSA, RMSE, and this showed that the data were accurately trained with great validation accuracy. The model examined two optimization functions with a different number of neurons to achieve the most accurate results in less time.

In the second stage, we used the LSTM prediction model, from stage one, to propose our novel algorithm, DCPA, to predict and control the sensor duty cycle activities. We compared our proposed DCPA with SDC and LDC. The performance of the DCPA was better than the SDC in consuming energy and lifetime and better than the LDC in having fewer missing data. These results assure maintaining stability inside the greenhouse ahead

of time with minimum human interaction and saving the sensor's power extending its lifetime considerably during its deployment.

Merging WSN, WVSN, deep learning, machine learning, image processing technologies creates a complete monitoring framework for a greenhouse to increase the quality and quantity of crop production without human interaction and for less cost. The significance of the proposed framework deployment in this thesis moves beyond the scientific data collection to enable a smart, intelligent, and safe, productive environment. The framework provides automatic, fast and accurate detection capabilities with long life network interaction, which applies to greenhouses deployment and can be used in different locations, including:

(1) Nursing homes, and home care for monitoring and reporting an emergency,

(2) Airports, trains, and transportation systems to provide information such as identification and recognition in real-time for security purposes, and

(3) Outdoor environment monitoring such as farms to monitor grazing animals and report their health.

6.2 Future Work

Several future research directions and open issues can be derived from our work thus far. In this section, we outline these directions.

- We used the Hough Forest machine learning technique and image processing methods to recognize powdery mildew fungus in a tomato crop. We think using the

same technique and generalizing it to recognize any diseases that affect crops is possible.

- We intend to expand our work in a more dynamic WSN and WVSN, where each smart camera and sensor node has a mobility feature and checks the effect on the network efficiency.
- Deep learning networks proved their ability to learn any complex function between a given input and an output. This potential may be further investigated to understand the timing requirements and the delay model of connections.
- Deep learning can be trained to solve the best placement, given an objective function. A deep learning model can be trained on the proposed problem, ILP-OPCQ, and their corresponding placement solutions. Later this model could be generalized to generate a placement solution to new unseen data.
- Developing a hybrid model including the CNN and ARIMA models for predicting the microclimate and diseases in a greenhouse.

Bibliography

[1] L. Kenneth, J. Joe, W.D.H. Hanan, "Goldsberry greenhouse management," *Advance series in agriculture sciences.* New York: Springer-Verlag, 1978.

[2] J. Mitchell, "The greenhouse effect and climate change," *Reviews of Geophysics*, 27(1), pp. 115-139, 1989.

[3] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine,* 40(8), pp. 102-114, 2002.

[4] H. Xiao, S. Lei, Y. Chen, H. Zhou, "WX-MAC: An energy efficient MAC protocol for wireless sensor networks," *IEEE 10th International Conference on Mobile Ad-Hoc and Sensor Systems,* pp. 423-424, 2013.

[5] M.A. Razzaque, C. Bleakley, S. Dobson, "Compression in wireless sensor networks," *ACM Transactions on Sensor Networks,* 10(1), pp. 1-44, 2013.

[6] C.P. Chen, S.C. Mukhopadhyay, C.L. Chuang, M.Y. Liu, J.A. Jiang, "Efficient coverage and connectivity preservation with Load Balance for Wireless Sensors Networks," *IEEE Sensors Journal*, 15(1), pp. 48-62, 2015.

[7] Liming Qiu, K. I. Wang, Z. Salcic, "Dynamic duty cycle-based Wireless Sensor Network for underground pipeline monitoring," *International Conference on Sensing Technology (ICST)*, 2015, pp. 116-121, 2015.

[8] C.P. Chen, C.L. Chuang, J.A. Jiang, "Ecological Monitoring Using Wireless Sensor Networks - Overview, Challenges, and Opportunities," Edited by Mukhopadhyay S., Jayasundera K., Fuchs A., *Advancement in Sensing*

Technology. Smart Sensors, Measurement and Instrumentation, vol 1. Springer, Berlin, Heidelberg, 2013.

[9] R. Pahuja, H.K. Verma, M. Uddin, "A wireless sensor network for greenhouse climate control," *IEEE Pervasive Computing*, 12(2), pp. 49-58, 2013.

[10] H. Anahita, B. Chris, H. Sepideh, H. Hannaneh, K. Haik, S. Costas, B. Alex, K. Christine, S. Majid, "Feasibility of a Secure Wireless Sensing Smartwatch Application for the Self-Management of Pediatric Asthma," *Sensors*, vol.17, pp. 1780-1786, 2017.

[11] G. Ahmed, X. Zhao, M.M.S Fareed, M.R. Asif, S.A. Raza, "Data Redundancy-Control Energy-Efficient Multi-Hop Framework for Wireless Sensor Networks," *Wireless Personal Communications*, 108(4), pp. 2559-2583, 2019.

[12] J. Lee, B. Shah, G. Pau, J. Prieto, K.-I. Kim, "Real-Time Communication in Wireless Sensor Networks," *Wireless Communications and Mobile Computing,* vol. 2018, p. 9612631, 2018.

[13] S.N. Alam, Z.J. Haas, "Coverage and connectivity in three-dimensional networks with random node deployment," *Ad Hoc Networks*, vol. 34, pp. 157-169, 2015.

[14] V.K. Sachan, S.A. Imam, M.T. Beg, "Energy-efficient communication methods in wireless sensor networks: A critical review," *International Journal of Computer Applications*, 39(17), pp. 35-48, 2012.

[15] Y. Zhang, C. Feng, I. Demirkol and W. B. Heinzelman, "Energy-Efficient Duty Cycle Assignment for Receiver-Based Converge cast in Wireless Sensor Networks," *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, Miami, FL, pp. 1-5, 2010.

[16] S. Soro, W. Heinzelman, “A Survey of Visual Sensor Networks,” *Advances in Multimedia,* vol. 2009, p. 640386, 2009.

[17] K. Aphetsi, C. Ozoemena, A. Mohammed, “A Surveillance Wireless Camera Sensor Network System for Intrusion Detection using Image Processing Techniques,” *Journal of Engineering and Applied Sciences*, vol.10, 2015.

[18] X. Zhu, B. Ding, W. Li, L. Gu, and Y. Yang, “On development of security monitoring system via wireless sensing network,” *EURASIP Journal on Wireless Communications and Networking*, *2*(1), p. 221, 2018.

[19] N. Williams, S. Zander, G. Armitage, “A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification,” ACM SIGCOMM Computer Communication Review, 36(5), pp. 5-16, 2006.

[20] Q. Kuang, L. Zhao, “A practical gup based KNN algorithm,” *International Symposium on Computer Science and Computational Technology (ISCSCI 2009)*, p. 151, 2009.

[21] S. B. Kotsiantis, “Supervised machine learning: A review of classification techniques,” *Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*. pp. 3-24, 2007.

[22] A. Liaw, M. Wiener, “Classification and regression by random forest,” *R News*, 2(3), pp. 18-22, 2002.

[23] D.H. Ballard, “Generalizing the Hough transform to detect arbitrary shapes,” Pattern Recognition, 13(2), pp. 111-122, 1981.

[24] A. Tran and A. Manzanera, "Fast growing hough forest as a stable model for object detection," 2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA), Oulu, pp. 1-6, 2016.

[25] Y. LeCun, Y. Bengio, G. Hinton, "Deep learning," *Nature*, 521(7553), p. 436, 2015.

[26] S. Dupond, "A thorough review on the current advance of neural network structures," *Annual Reviews in Control*, vol.14, pp. 200-230, 2019.

[27] A. Felix, N. Schraudolph, J. Schmidhuber, "Learning Precise Timing with LSTM Recurrent Networks," *Journal of Machine Learning Research*, vol.3, pp.115-143, 2002.

[28] J.L. Iga, C.L. Iga, R.A. Flores, "Effect of air density variations on greenhouse temperature model," *Mathematical and Computer Modelling*, 47(10), pp.855-867, 2008.

[29] P.A. Thomas, "Greenhouse operation and management," *HortScience*, 40(3), pp.503-504, 2005.

[30] D. Mears, "Greenhouse climate control - An integrated approach," Edited by J. C. Bakker, G. P. A. Bot, H. Challa and N. J. Van de Braak. Wageningen Pers, The Netherlands, 279 pp. ISBN 90 74134 17 3. Agricultural Systems, Elsevier, 51(3), pp. 370-372, 1996.

[31] N. Harris, A. Cranny, M. Rivers, K. Smettem, E.G. Barrett-Lennard, "Application of distributed wireless chloride sensors to environmental monitoring: Initial results," *IEEE Transactions on Instrumentation and Measurement*, 65(4), pp. 736-743, 2016.

[32] M. Kang, X. Fan, J. Hua, H. Wang, X. Wang, F. Wang, "Managing traditional solar greenhouse with CPSS: A just-for-fit philosophy," *IEEE Transactions on Cybernetics*, 48(12), pp. 3371-3380, 2018.

[33] European Commission, "Directorate-General for Agriculture and Rural Development, Ddirectorate H. Sustainability and Quality of Agriculture and Rural Development, H.3," *Organic Farming*, 2013.

[34] E. Eriksson, G. Da´n, V. Fodor, "Predictive distributed visual analysis for video in wireless sensor networks," *IEEE Transactions on Mobile Computing*, 15(7), pp. 1743-1756, 2016.

[35] Y. Wang, Z. Liu, D. Wang, Y. Li, J. Yan, "Anomaly detection and visual perception for landslide monitoring based on a heterogeneous sensor network," *IEEE Sensors Journal*, 17(13), pp. 4248-4257, 2017.

[36] J. Liu, S. Sridharan, C. Fookes, "Recent advances in camera planning for large area surveillance: A comprehensive review," *ACM Computing Surveys*, 49(1), 2016.

[37] A.I. Jing, A. Abouzeid, "Coverage by directional sensors in randomly deployed wireless sensor networks," *Journal of Combinatorial Optimization*, 11(02), pp. 21-41, 2006.

[38] M. Hoffmann, M. Wittke, J. Hahner, C. Muller-Schloer, "Spatial partitioning in self-organizing smart camera systems," *IEEE Journal of Selected Topics in Signal Processing*, 2(4), pp. 480-492, 2008.

[39] A. Altahir, V. Asirvadam, "Optimizing Visual Surveillance Sensor Coverage Using Dynamic Programming," *IEEE Sensors Journal.* 17(11), pp. 3398-3405, 2017.

[40] P. Natarajan, K. Pradeep, Atrey, M. Kankanhalli, “Multi-camera coordination and control in surveillance systems: A survey,” *ACM Transactions on Multimedia Computing, and Applications,* 11(4), 2015.

[41] P. Munishwar, N. Abu-Ghazaleh, “Coverage algorithms for visual sensor networks,” *ACM Transactions on Sensor Networks,* 9(4), 2013.

[42] K. Chakrabarty, S.S. Iyengar, H. Qi, E. Cho, “Grid coverage for surveillance and target location in distributed sensor networks,” *IEEE Transactions on Computers*, 51(12), pp. 1448-1453, 2002.

[43] E. Horster, R. Lienhart, “Approximating optimal visual sensor placement,” *IEEE International Conference on Multimedia and Expo*, pp. 1257-1260, 2006.

[44] C. Valdivia, J. Escobedo, H. Mun˜oz, J. Berumen, A. Herna´ndez, O. Guirette, A. Arroyo, J. Mendoza, F. Becerra, A. Ortega, H. Azcaray, M. Ortiz, “Implementation of virtual sensors for monitoring temperature in greenhouses using CFD and control,” *Sensors*, 19(01), pp. 1-13, 2019.

[45] A.A. Altahir, V.S. Asirvadam, N.H. Hamid, P. Sebastian, N. Saad, R. Ibrahim, S.C. Dass, “Modeling multicamera coverage for placement optimization,” *IEEE Sensors Letters*, 1(6), pp.1-4, 2017.

[46] X. Wang, H. Zhang, S. Fan, H. Gu, “Coverage Control of Sensor Networks in IoT Based on RPSO,” *IEEE Internet of Things Journal*, 5(5), pp. 3521-3532, 2018.

[47] X. Wang, H. Gu, H. Zhang, H. Chen, “Novel RPSO Based Strategy for Optimizing the Placement and Charging of a Large-Scale Camera Network in Proximity Service,” *IEEE Access,* vol.7, pp. 16991-17000, 2019.

[48] X. Wang, H. Gu, Y. Liu, H. Zhang, "A Two-Stage RPSO-ACS Based Protocol: A New Method for Sensor Network Clustering and Routing in Mobile Computing," *IEEE Access*, vol.7, pp. 113141-113150, 2019.

[49] C. Papadimitriou, K. Steiglitz, "Combinatorial Optimization: Algorithms and Complexity," *Prentice-Hall*, 1982.

[50] L.A. Wolsey, "Integer Programming," *Wiley Series in Discrete Mathematics and Optimization*. Wiley, 1998.

[51] https://www150.statcan.gc.ca/n1/daily-quotidien/200506/dq200506a-eng.htm, The Daily, Greenhouse, Sod and Nursery Industries, 2019.

[52] D. Jacob, D.R. Sztjenberg, D. Rav David, Y. Elad, "Conditions of development of powdery mildew of tomato caused by Oidium neolycopersici," *Phytopathology,* 98(3), pp. 270-281, 2008.

[53] R.N. Van Den Driessche, "Prediction of mineral nutrient status of trees by foliar analysis," *The Botanical Review,* vol. 40, pp. 347-394, 1974.

[54] M. Garcia, D. Bri, S. Sendra, J. Lioret, "Practical deployments of wireless sensor networks: A survey," *International Journal on Advances in Networks and Service,* vol. 3. pp. 136-178, 2010.

[55] J. Lloret, M. Garcia, D. Bri, S. Sendra, "A wireless sensor network deployment for rural and forest fire detection and verification," *Sensor,* vol. 9, pp. 8722-8747, 2009.

[56] C. Anand, S. Sadistap, S. Bindal, B.A. Botre, K.S.N. Rao, "Wireless multi-sensor embedded system for Agro-industrial monitoring and control," *International Journal on Advances in Networks and Service,* vol. 3, pp. 1-10, 2010.

[57] Di. D. Palma, L. Bencini, G. Collodi, G. Manes, F. Chiti, R. Fantacci, A. Manes, "Distributed monitoring systems for agriculture based on wireless sensor network technology," *International Journal on Advances in Networks and Service ,* vol. 3, pp. 11-21, 2010.

[58] D. Sunding, D. Zilberman, "The agricultural innovation process: Research and technology adoption in a changing agricultural industry Handbook of Agricultural and Resource Economics," Edited by Gardner, B., Rausser, G.C., *North-Holland: Amsterdam, The Netherlands*, 2000.

[59] T. Baidyk, E. Kussul, O. Makeye, A. Vega, "Limited receptive area neural classifier-based image recognition in micromechanics and agriculture," *International Journal of Applied Mathematics and Informatics*, vol. 2, pp. 96-103, 2008.

[60] C.C. Yang, S.O. Prasher, J.A. Landry, H.S. Ramaswamy, A. Ditommaso, "Application of artificial neural networks in image recognition and classification of crop and weeds," *Canadian Biosystems Engineering*, 42(3), pp. 147-152, 2000.

[61] C.C. Yang, S.O. Prasher, J.A. Landry, J. Perret, H.S. Ramaswamy, "Recognition of weeds with image processing and their use with fuzzy logic for precision farming," *Canadian Biosystems Engineering*, 42(4), pp. 195-200, 2000.

[62] D.G. Sena, F.A.C. Pimto, D.M. Queiroz, P.A. Viana, "Algoritmo de processamento de imagens para controle localizado de pragas na cultura do milho," *International Symposium on Precision Agriculture*, p. 1214, 2002.

[63] A. Macedo-Cruz, G. Pajares, M. Santos, I. Villegas-Romero, "Digital image sensor-based assessment of the status of Oat (Avena sativa L.) Crops after frost damage," *Sensors*, vol. 11, pp. 6015-6036, 2011.

[64] P.E. Cruvinel, E.R. Minatel, M.L. Mucheroni, S.R. Vieira, S. Crestana, "An Automatic Method Based on Image Processing for Measurements of Drop Size Distribution from Agricultural Sprinklers," *Simpsio Brasileiro de Computao Grfica e Processamento de Imagens* (SIBGRAPI), vol. 9, pp. 39-46, 1996.

[65] L. Velzquez, N. Sasaki, Y. Nakano, K. Meja-Muoz, J.M. Romanchik, E. Kriuchkova, "Detection of powdery mildew disease on rose using image processing with Open CV," *Rev. Chapingo. Ser. Horticult,* vol. 17, p. 151160, 2011.

[66] M. Zhang, Q. Meng, "Automatic citrus canker detection from leaf images captured in field," *Pattern Recognition Letters*, vol. 32(15), pp. 2036-2046, 2011.

[67] P. Kulkarni, D. Ganesan, P. Shenoy, Q. Lu, "SensEye: A Multi-tier Camera Sensor Network," *ACM International Conference on Multimedia (MM05)*, p. 611, 2005.

[68] G. Yap, H. Florence, Y. Hong-Hsu, "A survey on sensor coverage and visual data capturing/processing/transmission in wireless visual sensor networks," *Sensors*, Switzerland, vol. 14, pp. 3506-3527, 2014.

[69] J. Gall, V. Lempitsky, "Class-specific Hough forests for object detection," pp. 1022-1029, 2009.

[70] K. Rematas, B. Leibe, "Efficient object detection and segmentation with a cascaded Hough Forest ISM," *IEEE international conference on computer vision workshops (ICCV workshops),* pp. 966-973, 2011.

[71] A. Ali, H.S. Hassanein, "A Fungus Detection System for Greenhouses Using Wireless Visual Sensor Networks and Machine Learning," *IEEE Globecom Workshops (GC Wkshps)*, pp. 1-6, 2019.

[72] P. Rawat, K.D. Singh, H. Chaouchi, J.M. Bonnin, "Wireless sensor networks: a survey on recent developments and potential synergies," *Journal of Supercomputing,* 68(1), pp. 1-48, 2014.

[73] Q. Wang, K. Xu, G. Takahara, H.S. Hassanein, "On lifetime-oriented device provisioning in heterogeneous wireless sensor networks: approaches and challenges," *IEEE Network Magazine*, 20(3), pp. 26-33, 2006.

[74] H. Xiao, S. Lei, Y. Chen, H. Zhou, "WX-MAC: An energy efficient MAC protocol for wireless sensor networks," *IEEE International Conference on Mobile Ad-Hoc and Sensor Systems,* pp. 423-424, 2013.

[75] K. Xu, Q. Wang, H.S. Hassanein, G. Takahara, "Relay node deployment strategies in heterogeneous wireless sensor networks: Single-hop communication case," *IEEE Global Communication Conference (GLOBECOM),* pp. 21-25, 2005.

[76] K.J. Yang, Y.R. Tsai, "WSN18-2: Link Stability Prediction for mobile ad hoc networks in shadowed environments," *IEEE Global Communication Conference (GLOBECOM),* 2006.

[77] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, A. Torralba, "Object detectors emerge in deep scene CNNS," *arXiv preprint arXiv,* 2014.

[78] K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-778, 2016.

[79] J.K. Zhu, “Abiotic stress signaling and responses in plants,” *Cell*, 167(2), pp.313-324, 2016.

[80] S.L. Patil, H.J. Tantau, V.M. Salokhe, “Modelling of tropical greenhouse temperature by auto regressive and neural network models,” *Biosystems Engineering*, 99(3), pp. 423-431, 2008.

[81] I.L. Lópezcruz, L. Hernándezlarragoiti, “Neuro fuzzy models for air temperature and humidity of arched and Venlo type greenhouses in central Mexico,” *Agrociencia*, 44(7), pp. 791-805, 2010.

[82] R. Guzmán-Cruz, R. Castañeda-Miranda, J.J. García-Escalante, I.L. López-Cruz, A. Lara-Herrera, J.I. De la Rosa, “Calibration of a greenhouse climate model using evolutionary algorithms,” *Biosystems Engineering*, 104(1), pp. 135-142, 2009.

[83] Q. Liu, D. Jin, J. Shen, Z. Fu, N. Linge, “A WSN-Based Prediction Model of Microclimate in a Greenhouse Using an Extreme Learning Approach,” *International Conference on Advanced Communication Technology (ICACT),* 2016.

[84] A. Sinha, D.K. Lobiyal, “Prediction models for energy efficient data aggregation in wireless sensor network,” *Wireless Personal Communications*, 84(2), pp. 1325-1343, 2015.

[85] D. Spenza, C. Petrioli, A. Cammarano, “Pro-Energy: A novel energy prediction model for solar and wind energy-harvesting wireless sensor networks,” *IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (MASS 2012)*, pp. 75-83, 2012.

[86] M. Taki, Y. Ajabshirchi, S.F. Ranjbar, A. Rohani, M. Matloobi, “Heat transfer and MLP neural network models to predict inside environment variables and energy lost in a semi-solar greenhouse,” *Energy & Buildings*, 110, pp. 314-329, 2016.

[87] W. Zou, F. Yao, B. Zhang, C. He, Z. Guan, “Verification and predicting temperature and humidity in a solar greenhouse based on convex bidirectional extreme learning machine algorithm,” *Neurocomputing,* vol. 249, pp. 72-85, 2017.

[88] F. Karim, S. Majumdar, H. Darabi, S. Chen, “LSTM fully convolutional networks for time series classification,” *IEEE Access,* vol. 6, pp. 1662-1669, 2017.

[89] H. Tian, S. Chen, “MCA-NN: Multiple Correspondence Analysis Based Neural Network for Disaster Information Detection,” *IEEE International Conference on Multimedia Big Data (BigMM),* pp. 268-275, 2017.

[90] F.X. Zhang, G.D. Li, W.X. Xu, Xinjiang, “Desertification disaster prediction research based on cellular neural networks,” *International Conference on Smart City and Systems Engineering (ICSCSE),* pp. 545-548, 2016.

[91] S. Traore, Y.F. Luo, G. Fipps, “Deployment of artificial neural network for short-term forecasting of evapotranspiration using public weather forecast restricted messages,” *Agriculture Water Management*, 163(1), pp. 363-379, 2016.

[92] S.K. Biswas, N. Sinha, B. Purkayastha, L. Marbaniang, “Weather prediction by recurrent neural network dynamics,” *International Journal of Intelligent Engineering Informatics* 2(2/3), PP. 166-180, 2014.

[93] F. Engmann, F.A. Katsriku, J.D. Abdulai, K.S. Adu-Manu, F.K. Banaseka, “Prolonging the lifetime of wireless sensor networks: a review of current

techniques," *Wireless Communications and Mobile Computing,* vol. 2018, p. 8035065, 2018.

[94] V. Raghunathan, C. Schurghers, S. Park, M. Srivastava, "Energy-aware Wireless Microsensor Networks," *IEEE Signal Processing Magazine*, pp. 40-50, 2002.

[95] V.K. Sachan, S.A. Imam, M.T. Beg, "Energy-efficient communication methods in wireless sensor networks: A critical review," *International Journal of Computer Applications*, 39(17), pp. 35-48, 2012.

[96] S. Liu, Y. Liu, X. Chen, X. Fan, "A new scheme for evaluating energy efficiency of data compression in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 14, p. 155014771877692, 2018.

[97] M. Ambigavathi, D. Sridharan, "Energy-aware data aggregation techniques in wireless sensor network," *Advances in Power Systems and Energy Management*, pp. 165-173, Springer, Singapore, 2018.

[98] H. Zhang, J.C. Hou, "Maximizing Lifetime of Wireless Sensor Network," *International Journal of Sensor Networks* 1(1/2), pp. 64-71, 2006.

[99] M.S. Pawar, J.A. Manore, M.M. Kuber, "Lifetime Prediction of Battery-Operated Node for Energy Efficient WSN Applications," *International Journal of Computer Science and Technology*, 2(4), 2011.

[100] Y. Zhang, CH. Feng, I. Demirkol, W. Heinzelman, "Energy-Efficient Duty Cycle Assignment for Receiver-Based Convergecast in Wireless Sensor Networks," pp. 1-5, 2010.

[101] M. Medidi, Y. Zhou, "Extending Lifetime with Differential Duty Cycles in Wireless Sensor Networks," *IEEE Global Telecommunications Conference*, pp. 1033-1037, 2007.

[102] F. Zorzi, M. Stojanovic, M. Zorzi, "On the Effects of Node Density and Duty Cycle on Energy Efficiency in Underwater Networks," *OCEANS'10 IEEE Conference and Exhibition*, 2010.

[103] F.M. Al-Turjman, H.S. Hassanein, M. Ibnkahla, "Towards prolonged lifetime for deployed WSNs in outdoor environment monitoring," *Ad Hoc Networks*, vol. 24, pp. 172-185, 2015.

[104] Q. Wang, K. Xu, G. Takahara, H.S. Hassanein, "On lifetime-oriented device provisioning in heterogeneous wireless sensor networks: Approaches and challenges," *IEEE Network*, 20(3), pp. 26-33, 2006.

[105] Q. Wang, K. Xu, H.S. Hassanein, G. Takahara, "Minimum cost guaranteed lifetime design for heterogeneous wireless sensor networks (WSNs)," *IEEE International Performance, Computing, and Communications Conference (PCCC 2005),* pp. 599-604, 2005.

[106] J. Polastre, J. Hill, D. Culler, "Versatile Low Power Media Access for Wireless Sensor Network," 2004.

[107] https://www.wunderground.com/wundermap/

[108] P. Diederik, J. Ba, "Adam: A Method for Stochastic Optimization," *CoRR abs/1412.6980* (2014), arXiv:1412.6980 http://arxiv.org/abs/1412.6980.

[109] L. Bottou, Stochastic Gradient Descent Tricks. In: Montavon G., Orr G.B., Müller KR., "Neural Networks: Tricks of the Trade," *Lecture Notes in Computer Science, Springer*, vol. 7700, 2012.

[110] M. Abadi, et al., "TensorFlow: A System for Large-scale Machine Learning," *USENIX Symposium on Operating Systems Design and Implementation (OSDI'16),* pp. 265-283, 2016.

[111] C. François et al., Keras, https://github.com/fchollet/keras, 2015.

[112] A. Ali, H.S. Hassanein, "Wireless Sensor Network and Deep Learning for Prediction Greenhouse Environments," *IEEE International Conference on Smart Applications, Communications and Networking (SmartNets)*, pp. 1-5, 2019.

[113] R. Shamshiri, P. van Beveren, H. Che Man, A.J. Zakaria, "Dynamic assessment of air temperature for tomato (Lycopersicon esculentum Mill.) cultivation in a naturally ventilated net-screen greenhouse under tropical lowlands climate," *Journal of Agricultural Science and Technology* , 19(1), pp. 59-72, 2017.

[114] F. Rodríguez, M. Berenguel, J.L. Guzmán, A. Ramírez-Arias, "Modeling and control of greenhouse crop growth," *Switzerland: Springer International Publishing*, 2015.

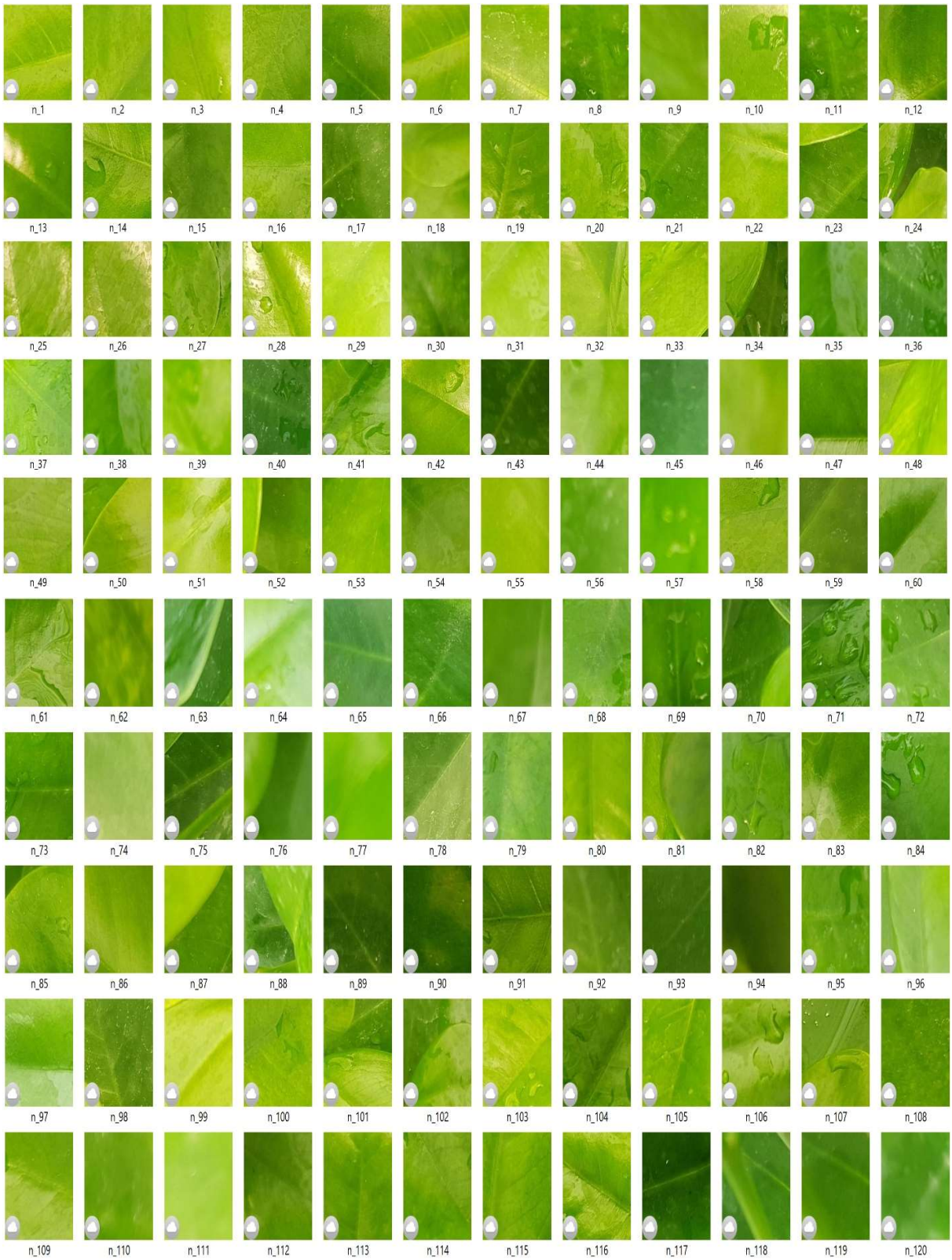
[115] H. Jung, A. Moon, S. An, Y. Song, "CNN-based Tomato Powdery Mildew Recognition Method," *Journal of Institute of Control, Robotics and Systems*, 24(7), pp. 617-623, 2018.

[116] https://www.agr.gc.ca/eng/horticulture/horticulture-sector-reports/statistical-overview-of-the-canadian-greenhouse-vegetable-industry-

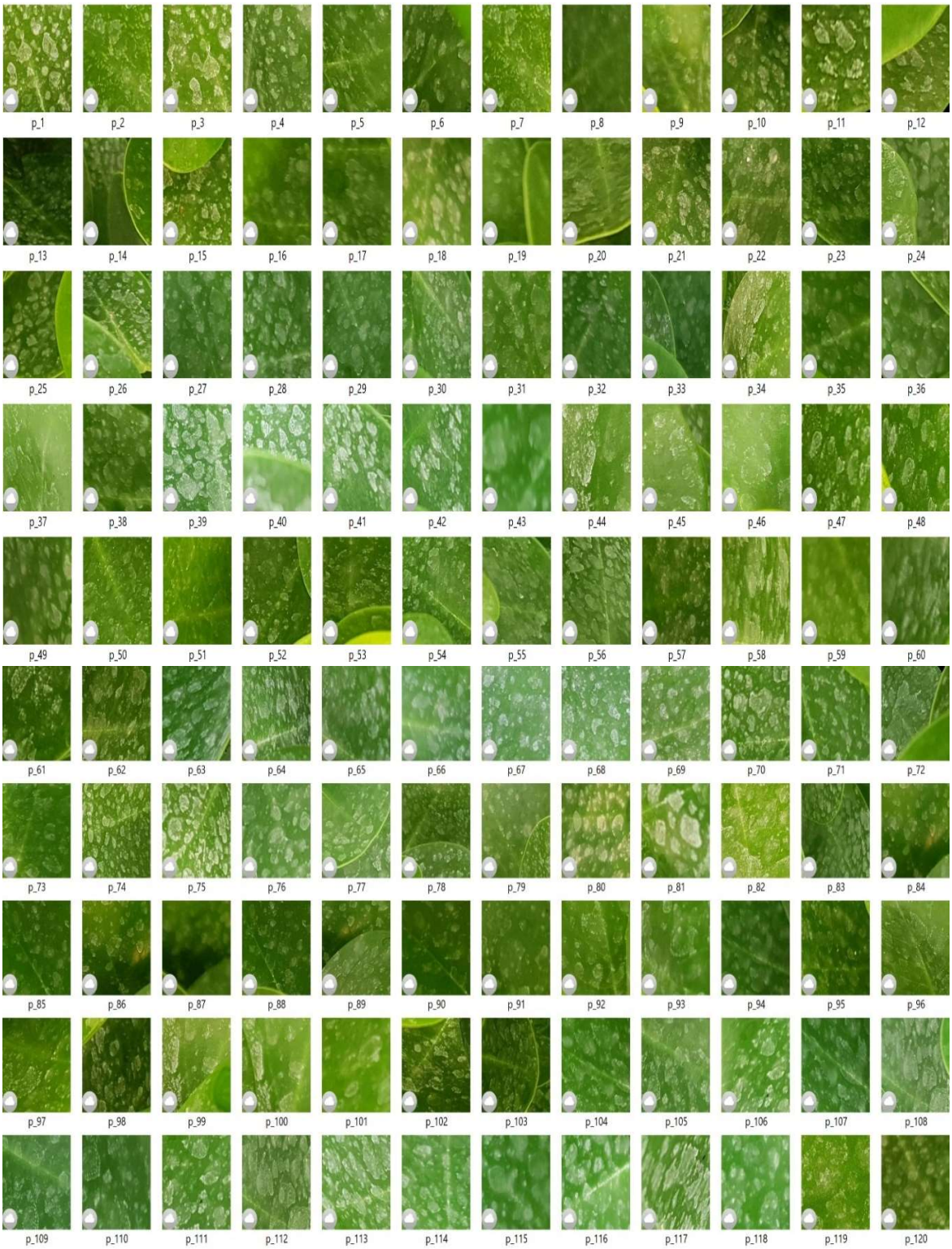
2018/?id=1578950554200, Overview of the Canadian Greenhouse Vegetable Industry, Agriculture and Agri-Food Canada (AAFC), 2018.

Appendix A

Samples of Greenhouse Images



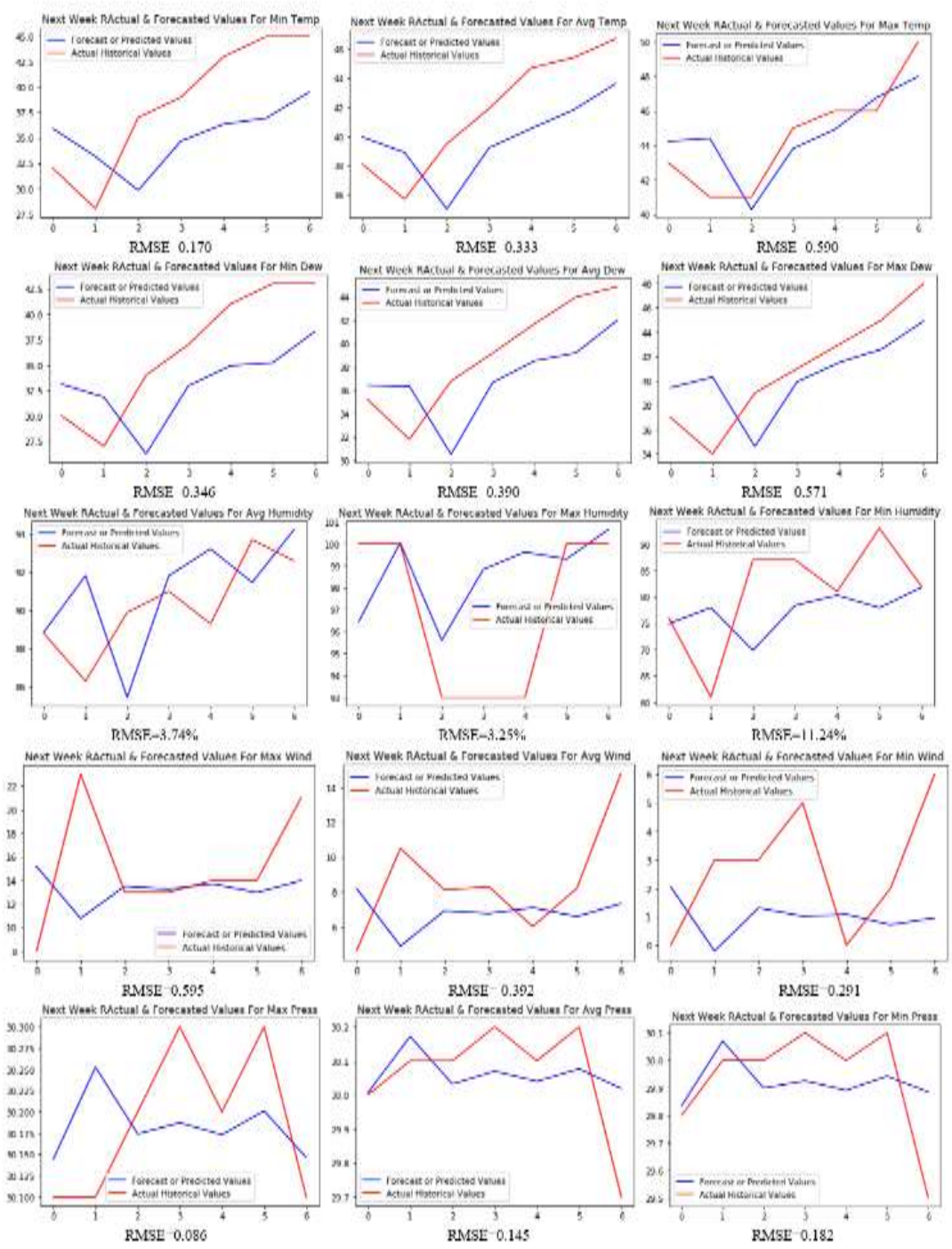
Sample patches (no fungus)



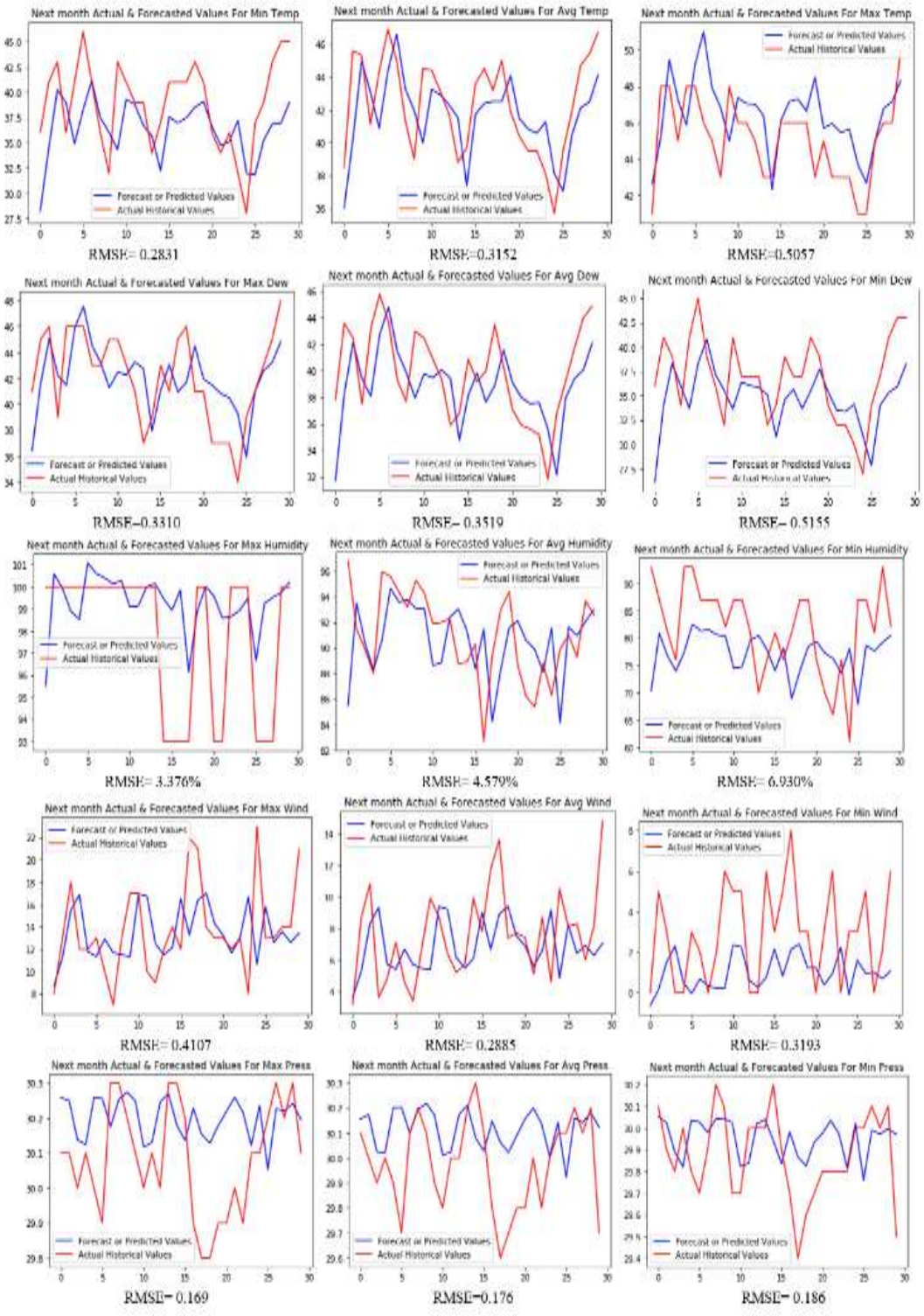
Sample patches (no fungus)

Appendix B

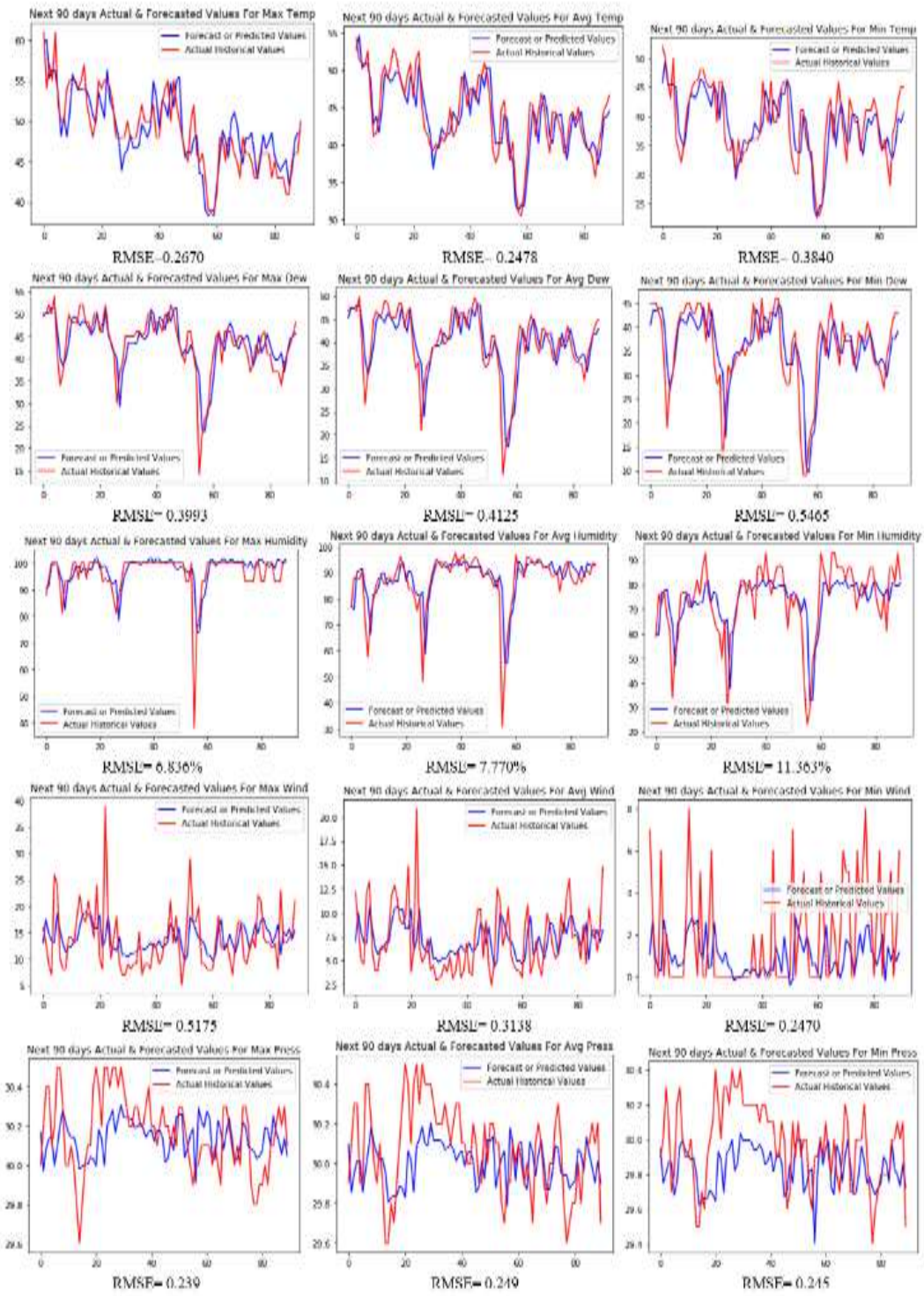
Prediction Microclimate



Prediction microclimate for seven days a head (min, max, avg)



Prediction microclimate for 30 days ahead (min, max, avg)



Prediction microclimate for 90 days ahead (min, max, avg)