12-2020

Acquisition, Processing, and Analysis of Video, Audio and Meteorological Data in Multi-Sensor Electronic Beehive Monitoring

Sarbajit Mukherjee
Utah State University

ACQUISITION, PROCESSING, AND ANALYSIS OF VIDEO, AUDIO AND METEOROLOGICAL DATA IN MULTI–SENSOR ELECTRONIC BEEHIVE MONITORING.

by

Sarbajit Mukherjee

A dissertation submitted in partial fulfillment
of the requirements for the degree

of

DOCTOR OF PHILOSOPHY

in

Computer Science

Approved:

Vladimir A. Kulyukin, Ph.D. Major Professor	Xiaojun Qi, Ph.D. Committee Member
Jacob Gunther, Ph.D. Committee Member	Nicholas Flann, Ph.D. Committee Member
Haitao Wang, Ph.D. Committee Member	D. Richard Cutler, Ph.D. Interim Vice Provost of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2020

ABSTRACT

Acquisition, Processing, And Analysis of Video, Audio And Meteorological Data In Multi–Sensor Electronic Beehive Monitoring.

by

Sarbajit Mukherjee, Doctor of Philosophy

Utah State University, 2020

Major Professor: Vladimir A. Kulyukin, Ph.D.
Department: Computer Science

The decline in the honey bee (*Apis mellifera*) population in recent years has prompted the need to gather more fruitful information to understand the many aspects of the behavior and ecology of bees. Electronic Beehive Monitoring is a method of gathering critical information regarding a colony's health and behavior. In this dissertation, we have presented an electronic beehive monitoring system called BeePi that requires no structural modifications to a standard beehive (Langstroth or Dadant beehive), thereby preserving the sacredness of the bee space without disturbing the natural beehive cycles.

In our research, we have conducted various tests and have proposed diagnostic models to address different electronic beehive monitoring aspects. We have developed a bee motion count algorithm that analyzes the video recordings of forager traffic over the landing pad without additional costly intrusive hardware and returns the number of bees that have moved in each video. The algorithm is also able to distinguish between incoming, outgoing, and lateral bee movements. The above processing is done under 2.5 minutes insitu on the Beepi consisting of a low-cost raspberry pi computer. We introduced a dataset of 32 videos (744*32=23808 frames) along with manual counts for the number of bees that moved in successive frames.

We also discussed the design of a convolutional neural network architecture that processes raw audio waveforms and compares the performance of different deep learning models toward classifying audio samples recorded by microphones on beehives.

Next, we studied the effect of 21 different meteorological variables on the foraging activity. By fitting a linear regression model, we found that 71% of the observed variation in the forager activity can be explained by the variation in evapotranspiration, visibility, temperature, pressure, and humidity.

The algorithms proposed in this dissertation have been developed using open-source software tools capable of running on low-power devices such as a raspberry pi or Arduino and have been tested on a structure built with off-the-shelf hardware components. To ensure that interested research and citizen science communities can reproduce our findings and algorithms, we have made our datasets of bee images, videos, audios and source codes public.

(269 pages)

PUBLIC ABSTRACT

Acquisition, Processing, And Analysis of Video, Audio And Meteorological Data In Multi–Sensor Electronic Beehive Monitoring.

Sarbajit Mukherjee

In recent years, a widespread decline has been seen in honey bee population and this is widely attributed to colony collapse disorder. Hence, it is of utmost importance that a system is designed to gather relevant information. This will allow for a deeper understanding of the possible reasons behind the above phenomenon to aid in the design of suitable countermeasures.

Electronic Beehive Monitoring is one such way of gathering critical information regarding a colony's health and behavior without invasive beehive inspections. In this dissertation, we have presented an electronic beehive monitoring system called BeePi that can be placed on top of a super and requires no structural modifications to a standard beehive (Langstroth or Dadant beehive), thereby preserving the sacredness of the bee space without disturbing the natural beehive cycles. The system is capable of capturing videos of forager traffic through a camera placed over the landing pad. Audio of bee buzzing is also recorded through microphones attached outside just above the landing pad. The above sensors are connected to a low-cost raspberry pi computer, and the data is saved on the raspberry pi itself or an external hard drive.

In this dissertation, we have developed an algorithm that analyzes those video recordings and returns the number of bees that have moved in each video. The algorithm is also able to distinguish between incoming, outgoing, and lateral bee movements. We believe this would help commercial and amateur beekeepers or even citizen scientists to observe the bee traffic near their respective hives to identify the state of the corresponding bee colonies.

This information helps those mentioned above because it is believed that honeybee traffic carries information on colony behavior and phenology.

Next, we analyzed the audio recordings and presented a system that can classify those recordings into bee buzzing, cricket chirping, and ambient noise. We later saw how a long–term analysis of the intensity of bee buzzing could help us understand the hive's development through an entire beekeeping season.

We also investigated the effect of local weather conditions using 21 different meteorological variables on the forager traffic. We collected the meteorological data from a weather station located on the campus of Utah State University. Through our study, we were able to show that without the use of additional costly intrusive hardware to count the bees, we can use our bee motion counting algorithm to calculate the bee motions and then use the counts to investigate the relationship between foraging activity and local weather.

To ensure that our findings and algorithms can be reproduced, we have made our datasets and source codes public for interested research and citizen science communities.

To my parents, who always inspire me to learn and grow as an individual, my beautiful wife for her continued encouragement and support, and 'dima' whom I dearly miss

ACKNOWLEDGMENTS

I want to express my sincere gratitude to many wonderful people who have supported me in different capacities throughout the journey of my Ph.D. program at Utah State University.

First, I would like to thank my major professor Dr. Vladimir Kulyukin for his continual advice, encouragement and productive feedback that has helped me grow as a researcher since 2015 when I became his Ph.D. student. I am grateful to Dr. Kulyukin who introduced me to the world of bees, along with guiding me in both academia and life. Second, I would like to thank my Ph.D. committee members: Dr. Xiaojun Qi, Dr. Jacob Gunther, Dr. Nicholas Flann, and Dr. Haitao Wang, for their valuable comments and feedback that helped me expand the horizon of my research.

I am grateful for my friends and colleagues at our CS laboratory in Old Main 405 for making my Ph.D journey pleasurable. Outside of the lab, I am grateful for the presence of many wonderful human beings in my day-to-day life. Thomas and Martha Stoddard as well as Trent and Annette Peterson for being parent figures and always being there with their helping hands; Joe Loveless, Trevor Landeen, Ben Fred, Jason Peterson, Avik Mukherjee, Jacob Adams for being such wonderful friends; Abigail, for helping me during challenging times as a friend and a sister. Bob and Mary Jones for accepting me as their son-in-law and always acknowledging me for my achievements and encouraging me to learn more.

Most importantly, I am very fortunate to have the support and blessings of my parents from thousands of miles away. I want to express my deepest gratitude to my parents Tarak Mukherjee and Sipra Mukherjee, for believing in me and always encouraging me to be a better person. Last but not least, I am grateful to my beautiful wife, Jacquelyn Anne Mukherjee, for her unconditioned love and continued support. Her dedication and professionalism as a high school science teacher has motivated me through our years together.

I am also very grateful to the very patient department secretaries, Genie Hanson, Vicki Anderson and Cora Price who answered all my questions and helped me sail smoothly

through the various paper work. Also, special thanks to Craig Huntzinger and Richard Mueller for letting us use their property in Northern Utah for our longitudinal electronic beehive monitoring tests.

Finally, I would like to express my gratitude to the Kickstarter backers who contributed to the two successful BeePi Project Kickstarter fundraisers in 2017 and 2019. All video and audio data acquired for the research in this thesis were acquired and stored on the hardware purchased with the Kickstarter backers' funds. Their funds were also used to purchase most of the bee packages in 2018 and 2019.

Sarbajit Mukherjee

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

1.1 Introduction

Recently, global food production faces significant challenges in terms of food security and quality. Bees and other pollinators play a significant role in the food production cycle and in providing balance to the environment, since they are essential for diet diversity, biodiversity, and the maintenance of natural resources [4,5]. A study performed by the Food and Agriculture Organization of the United Nations (FAO) in [6] stated that with the help of bees and other pollinators, global food production has increased by 24%. Furthermore, bees are often regarded as the most important among insect pollinators due to their ability to transport and store pollen efficiently [7]. Honeybees (*Apis mellifera*) especially contribute to approximately 14% of pollination services for agricultural production [8]. Without bees and other pollinators, a variety of crops (such as almonds, coffee, apples etc.) that depend upon the pollination services would face a severe shortage. To raise awareness regarding the vital role bees play in human welfare and keeping the planet healthy, the United Nations has named May 20 as World Bee Day. Hence it is of utmost importance to devise solutions that aim to preserve the honey bees' healthy existence.

A honey bee colony's health is influenced by different external factors such as pollution, pesticides, an increase of pathologies, etc. To a beekeeper, the wellbeing of a honeybee colony depends on the robustness of forager traffic. Therefore, visual estimates of forager traffic are often used by professional and amateur beekeepers to evaluate a honeybee colony's health. When the forager traffic activity is higher than usual, it might indicate onsets of swarm or other hive threatening events such as colony robbing; on the other hand, lower levels of forager traffic might indicate to a beekeeper that there could be mite infestations, failing queens, or chemical poisonings [9]. Hence continuous monitoring of honeybee colonies

has been the subject of interest to beekeepers, researchers and apiologists for a long time [10], since early detection of the health status of the beehives could be a crucial element in ensuring the survival of the colonies. Although experienced beekeepers can easily infer the health of honeybee colonies from forager traffic patterns, it might not always be possible for them to be physically present to manually inspect every hive, especially when commercial beekeepers have to manage several hundreds of beehives. The reason could be fatigue, or even logistics as some beekeepers have to drive long distances to their apiaries due to ever–increasing urban and suburban sprawl.

A consensus is emerging among beekeepers and researchers regarding the usefulness of electronic beehive monitoring (EBM) to collect valuable information on honeybee colony behavior and phenology without invasive beehive inspections, thereby reducing labor costs [11]. In general, special attention has been given to monitoring the effects of weather variables such as temperature, humidity and solar radiation along with sounds from a beehive, vibrations, beehive weight and gas contents [12].

Recently, there have been continuous and rapid advancements in consumer electronic sensors that have made it feasible to transform hives along with entire apiaries into networks of sensors that could monitor in situ the health of bee colonies [13]. The reduction in the cost and size of different sensors has made it possible to be deployed and collect useful data from hives located in the countryside. Therefore, practical EBM applications have started to be adopted by beekeepers which provides them with useful information from remote hives without having to manually inspect the above. However, we need to keep in mind that monitoring and data acquisition of a biological process such as the natural cycle of the bees is quite complicated because the behavior of a natural system such a beehive in response to human intervention, is not predictable. Additionally, the bees tend to neutralize any foreign objects (such as sensors) that might appear as a threat to them. Hence, one of the primary goals of this dissertation is to design an EBM system, called BeePi, that requires no structural modifications to a standard beehive (such as the Langstroth beehive [14] or the Dadant beehive [15]), thereby preserving the sacredness of the bee space without disturbing

the natural beehive cycles.

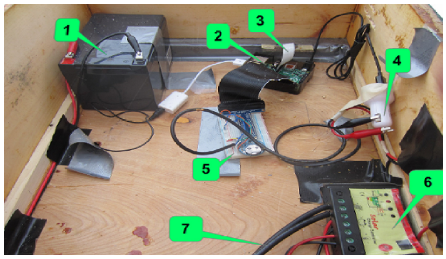
A fundamental objective in the BeePi system design was reproducibility; i.e., the goal was to create a suite of replicable hardware and software tools that researchers and citizen scientists can easily use to build their EBM systems at minimum cost and time commitments. Even the algorithms proposed in this dissertation have been developed using open-source software tools capable of running on low power devices such as a raspberry pi or arduino, and have been tested on a structure built with off-the-shelf hardware components.

1.2 Deployment of BeePi Monitors

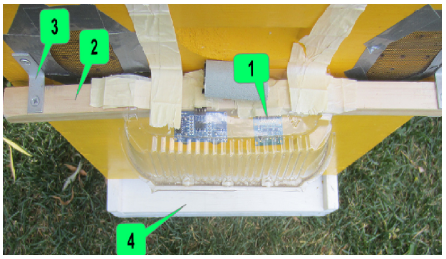
BeePi monitors have so far had seven field deployments and three different iterations of hardware and software improvements [1, 2, 16–22]. The first deployment was on private property in Logan, UT in early fall (September) 2014. A BeePi monitor was placed inside an empty hive and ran exclusively on solar power for two weeks. The second deployment was in Garland, UT (December 2014–January 2015), when a BeePi monitor was placed in a hive with overwintering honeybees (refer to Figure 1.1a) and successfully operated on solar power for nine out of the fourteen days of deployment to capture ≈3 GB of data (images, audio recordings and temperature readings). The third deployment was in North Logan, UT (April 2016–November 2016) where four BeePi monitors were placed into four beehives at two small apiaries and captured ≈20 GB of data running exclusively on batteries by harvesting solar energy (refer to Figure 1.1b).

(a) January 2015. (b) August 2016.

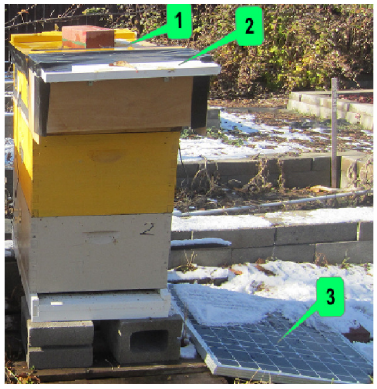
Figure 1.1: BeePi monitors with solar panels.



(a) Solar BeePi hardware components: 1) battery; 2) raspberry pi; 3) raspberry pi camera; 4) car charger; 5) breadboard; 6) solar charge controller; 7) solar panel wires



(b) Camera (1) under plastic cover attached to wooden plank (2); plank is attached with metallic brackets (3) to box with BeePi hardware; camera looks down on landing pad (4)



(c) Hive lid (1) with protection box (2) on hive; solar panel (3) on ground

Figure 1.2: Deployment of BeePi monitor with solar panel.

The hardware components of the solar-powered BeePi monitors is shown in Figure 1.2a. The setup included a raspberry pi 3 Model B with 1GB RAM, a T-Cobbler board, a full sized breadboard for sensor integration, a waterproof DS18B20 digital temperature sensor, a raspberry pi camera and a USB microphone hub where three Neewer 3.5 mm mini lapel microphones were plugged in. The microphones have a frequency range of 1520 KHz, an omnidirectional polarity, and a signal-to-noise ratio greater than 63 dB. The wired microphones were connected to the raspberry pi and placed into small holes drilled in on the side of the hive walls to collect audio samples from inside the hive. To prevent the propolisation of microphones, the above holes were covered using mesh nets. Renogy 50 watts 12 Volts monocrystalline solar panels were used for harvesting solar energy into UPG 12V 12Ah F2 sealed lead acid AGM deep-cycle rechargeable batteries via Renogy 10 Amp PWM solar charge controllers. It took us approximately 25 minutes to wire and assemble a BeePi monitor before deployment.

The raspberry pi camera was attached to a wooden plank to improve the camera's stability during high winds. The camera was positioned in a way such that it looked down straight on the landing pad. Furthermore, the wooden plank is attached with screws and metallic brackets to the box holding the BeePi hardware components, as shown in Figure 1.2b. The box was then placed over the topmost super, as we can see in Figure 1.2c. The camera was further protected from the elements by a wooden hive lid as seen in Figure 1.2c. The lid also protected other hardware components in the box against the elements from above and the four sides. In the third deployment of the BeePi [18], the data collection software was written in Python 2.7.9. Three separate data collection threads were written in python that started on system startup. The first thread collected temperature data every 5 minutes and saved them to a text file. The second thread recorded 30 seconds of audios in wave format every 5 minutes. The third thread captured pictures of the landing pad every 5 minutes and saved them in png format. The data from all three threads were saved on a 32 GB micro sd card inserted into the raspberry pi. In Chapter 3, we propose an algorithm based on 1D Haar Wavelet Transformation and apply it to the above data to

compute omnidirectional bee counts from static images of beehive landing pads.

In the fourth deployment, four BeePi units were placed into four beehives with Italian honey bee colonies at two small apiaries in Logan and North Logan, UT (April 2017–September 2017). We were able to collect ≈220 GB of audio, video, and temperature data. In the fifth deployment, four BeePi monitors were placed in four new beehives freshly initiated with Carniolan honeybee colonies at a small apiary in Logan, UT, during the beekeeping season of 2018 in April. In 2018 during the fifth deployment, we decided to keep the monitors deployed through the winter to stress test the equipment in the harsh weather conditions of northern Utah. One of the hives survived the harsh winter and we were able to record ≈400 GB of data. For the fourth and fifth deployment, we updated our data collection software by making three changes. First, we replaced the data collection thread that captured static images of landing pad with a separate thread that ran every 15 minutes and recorded videos (duration 30 seconds with 25 fps) of the the landing pad. Secondly, we updated the audio data recording thread, which was previously written in Python, with a bash script. This helped us get reliable audio data without little to none software issues, which was often the case during the third deployment. Finally, we made a small hardware change to the BeePi setup, wherein the microphones were placed 10 cm above the landing pad with microphones on each side as seen in Figure 1.3 rather than being embedded into the hive walls. The above change in the microphones' position helped us get good quality audio data compared to the previous iterations where the embedded microphones were often propolized by the bees which resulted in the bad quality of audio data. Another important update was regarding the power provided to the BeePi monitors. There were two versions of BeePi in the fifth deployment: regular and battery. The BeePi operated either on the grid or on a rechargeable battery. We used two types of rechargeable batteries for power storage: the UPG 12 V 12 Ah F2 lead acid AGM deep cycle battery and the Anker Astro E7 26,800 mAh battery.

In Chapter 6, we will use the above audio data from the fourth and fifth deployment and discuss the design and compare different deep learning models towards classifying the

(a) Bee hive in an apiary (b) Closeup image of microphones

Figure 1.3: Bee hive with microphones attached approximately 10cm above the landing pad. The microphones are not affected by rain or snow.

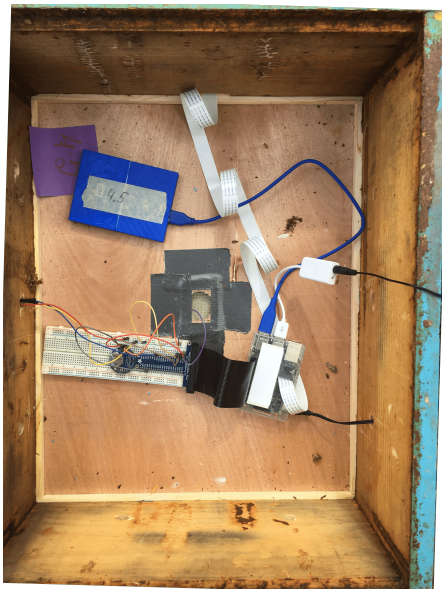
audio samples into bee buzzing, cricket chirping and ambient noises. To aid in our investigation, we introduced two different datasets of labeled audio samples. In the first dataset BUZZ1 [23] with 10,260 audio samples, the training and testing samples were separated from the validation samples by beehive and location. In the more challenging second dataset, BUZZ2 [23] with 12,914 audio samples, the training and testing samples were separated from the validation samples by beehive, location, time and bee race. We also introduced three image datasets (BEE1 [24], $BEE2\_1S$ [25], $BEE2\_2S$ [26]) using the video data from the fourth and fifth deployment.

The sixth deployment started in May 2019 with four freshly installed bee packages and the one surviving hive from the fifth deployment in 2018. The monitors were left running through the winter of 2019 and only one of the five hives survived the winter. In Chapter 4 and Chapter 5, we will use the above video data and propose a bee movement localization technique combined with a digital particle image velocimetry (dpiv) [27, 28] based algorithm to count bee motions between successive frames. We will show how we can use dpiv to analytically classify and count different bee motions into incoming, outgoing or lateral and then use those individual counts as measurements of directional traffic levels. To aid in our investigation, we created an evaluation dataset of 32 videos selected from different times during the beekeeping season of 2017, 2018 and 2019 with varying levels of bee traffic across different backgrounds. To generate the ground truth data, each of the

frames in the 32 videos (744*32=23808 frames) were individually evaluated beforehand and the number of bees that moved in successive frames was counted from them.

The seventh field deployment started in May 2020 when we installed four new bee packages in the same apiary in Logan, UT. The deployment is ongoing, with ≈450 GB of data collected so far. For the current deployment, we updated and improved our data collection software. We updated the video and temperature data collection threads, which were written in Python 2.7.9, with individual bash scripts. The reason for the update was primarily driven by the fact that previously the data collection threads were dependent on each other, which meant that a failure in one thread often resulted in the whole BeePi monitor to break down without recording any data. Hence hardware issues were quite common in the previous iterations. In the current deployment, three separate bash scripts were written to record audio, video and temperature data and save them on an external harddrive. A cronjob was written to monitor the scripts and to restart them individually every 15 minutes. This meant that if one sensor failed, the other sensors kept on recording seamlessly. By using the above updated data collection method, we believe that we are able to eliminate the random data loss due to hardware failures. Our claim is backed up by the recorded data from this current beekeeping season of 2020, where we have not experienced any data loss from whole BeePi monitors due to the hardware failure of a single sensor. The final setup of the BeePi monitor is shown in Figure 1.4.

(a) A BeePi monitor box on top of a Langstroth beehive that consists of 3 supers; the camera is protected against the elements with a specially constructed wooden box.



(b) A BeePi monitor consists of a raspberry pi 3 model B v1.2 computer, a pi T-Cobbler, a breadboard, a waterproof DS18B20 temperature sensor, a pi v2 8-megapixel camera board, a v2.1 ChronoDot clock, and a Neewer 3.5 mm mini lapel microphone placed above the landing pad.

Figure 1.4: Setup of the latest BeePi deployment running on the grid.

1.3 Outline of Contributions

In this dissertation we explore different techniques using computer vision, machine learning, deep learning and audio processing in the design of an electronic beehive monitoring (EBM) system, called BeePi. BeePi has been designed to be a multisensor electronic beehive monitor, all of whose components fit in a standard Langstroth box commonly used by thousands of beekeepers worldwide. In this dissertation, four main hypothesis will be addressed:

1. We can estimate bee traffic patterns and count the number of bees moving in and out of the hives by analyzing bee traffic videos recorded by a BeePi monitor in-situ on a low power device, such as a raspberry pi, efficiently within a time period.

2. Convolutional Neural Networks can operate in situ in electronic beehive monitors that use low voltage devices such as Raspberry Pi to classify audio samples recorded by microphones deployed approximately 10 cm above Langstroth beehives' landing pads.

3. It is possible to infer about hive development by analyzing audio and video recordings.

4. Hive activity is correlated to local weather conditions and observed variation in forager traffic can be explained by variation in the weather variables.

The first hypothesis states that it is possible to get a count of the forager traffic that have moved in and out of the hives by analyzing the 30 seconds of video recording. We support our claim by proposing a bee movement localization technique in Chapter 4 and then combined it with a dpiv based algorithm in Chapter 5, to count bee motions between successive frames. We also demonstrate in Chapter 5, that the combined algorithm runs on a raspberry pi 3 and generates the counts on an average in $\approx$2.15 minutes.

We support our claim in the second hypothesis in Chapter 6, by proposing the design of a convolutional neural network (CNN) to classify raw audio signals and then demonstrating that the CNN trained to classify raw audio can successfully operate in situ on a low voltage raspberry pi computer.

The third hypothesis is built upon the first and second hypothesis where in we used the bee motion counts and the buzzing intensity of the classified bee audio from an entire bee keeping season in order to support our claim. In Chapter 5, we compare the time series data for the bee motion counts during the entire beekeeping season of 2018 from May to November for two of our hives R_4_5 and R_4_7 and report on some interesting observations regarding the health of the hives. Similarly in Chapter 7, we analyze the power/intensity of the bee buzzing over the entire beekeeping season of 2018 from May to November and investigate how two of our hives(R_4_5 and R_4_7) had progressed through the 2018 beekeeping season.

The final hypothesis is addressed in Chapter 8, wherein we study the effects of 21 different meteorological variables on the foraging activity during the period of May–November of our beekeeping season in 2018 and during May–July 2019. Through our study we are able to

show that without the use of additional costly intrusive hardware to count the bees, we can use our bee motion counting algorithm to count the bee motions and then use the counts to investigate the relationship or correlation between foraging activity and local weather.

CHAPTER 2

RELATED WORKS

Although there has been more than a century devoted to academic research on investigating bee behavior, still many aspects of the behavior and ecology of bees remain less understood [29, 30] due to difficulties in collecting quantitative data of such small and fast moving insects. To address the above difficulties, different studies have been performed by using experimental hives and equipping bees with different kinds of tags.

2.1 Beehive Monitoring

The first known application of beehive monitoring was published in 1914 [31], where the author published data regarding the internal temperature of a beehive which was collected manually every hour for several days in 1907. Following that in 1926, there was a study using thermocouples to measure the temperature inside a beehive [32]. One of the first electronic system designed to monitor bees was Apidictor [33], which consisted of a low-pass filter to detect the changes in the frequencies of bee buzzing that took place inside the hive up to two to three weeks before swarming. In [34], several hives were equipped with accelerometers to observe the increasing amplitudes during days prior to swarming.

In another study, the authors in [35], placed hives on electronic balances connected to a 12-bit resolution datalogger to record and analyze the evolution of the weight of the beehives for over sixteen months. The system was powered by solar panel and the weight was recorded hourly. Following the development of sensors and microelectronics in recent times, different types of monitoring tool have been designed from observing data in the hive [36] to building systems able to analyze those data [37]. In [10, 38], some of the beehive monitoring systems have been reviewed and summarized.

On the other hand, advancements in the performance and size of microcontrollers have helped in the development of low-cost electronic beehive monitoring systems based on

Arduino® , Raspberry Pi® etc. In one such designed system [39] based on Arduino, the authors recorded temperature and relative humidity data and saved them on to a microSD memory card. In order to analyze those data, a beekeeper had to go to the location of the hive and manually download the content from the microSD card of each hive onto his/her hand held device.

Some commercial systems have also been designed to monitor beehives, for example Arnia [40] and HiveMind [41], which provide access to data from individual hives through Internet-enabled devices.

In a more recent study [42], the authors have presented a method based on supervised machine learning, that uses the data from sensors placed in the hive (internal temperature and hive weight), as well as weather data (temperature, dew point, wind direction, wind speed, rainfall, and daylight) and data regarding apiary inspections in order to forecast the health of a honeybee colony.

2.2 Counting Bees

The importance of counting forager traffic close to the hive entrance have prompted multiple research and commercial attempts since it is an indicator of the colony's health, honey flow and pollination. Hence accurate estimates of forager traffic levels is important to apiarists, fruit growers and also researchers.

One of the first electrical bee counters was proposed in [43], and later on that design was subsequently adopted and improved upon in [44], where the bees were counted by analyzing electrical impulses generated by bees tripping a balance arm. Similar electrical bee counters were also later presented in [45, 46]. In [47], the authors proposed the design of a box like extension that was attached to the hive entrance. The box was equipped with special tube like structure that were coated with paraffin, through which each bee passed. There was also a mesh bag attached to the end of the tubes in order to collect the crawling bees and weigh them.

In earlier electrical bee counters it was not possible to distinguish between incoming and outgoing forager traffic. Hence in subsequent research a bi-directional bee counter was

presented in [48]. Later on the design was adopted by Lowland Electronics in Belgium to manufacture bee counters, in where the bees were counted when they passed through special portals equipped with infrared (IR) sensors. In a later research in [49], the authors presented their *SmartHive*® system equipped with IR bee counters. They argued that their IR based counters were more robust and accurate in comparison to capacitance and video based systems. But the IR counters required regular maintenance and hence they also developed a self diagnostic program that checked the proper functioning of the IR sensors.

In another study [50], the authors attached a tag to dorsum of the bees thoraxes and then presented an imaging system located at the entrance to the hive to count the number of times each tagged bee entered and exited the hive. Following that, there have also been studies where some researchers used radio frequency identification (RFID) to tag each bee in order to solve the bee counting problem. In one such study [51], the authors used the RFID tags to get the bee count and study the effect of pesticides on honeybee colonies by exposing workers from a colony of approximately 2000 bees to contaminated sugar syrup at a feeder. RFID based tagging was also used in [52], where the authors presented a technique where a RFID based system was used to find the frequency and length of the nuptial flights of honeybee queens.

There have also been studies that use computer vision based approach to analyze the behavior of forager traffic and their flight activities outside the hive. In [53], the authors proposed a model that was able to track 99% of single flying bees and 70% of merged bees from 2D videos recorded at close proximity to the hive entrance. Their model is based on bees blobs segmentation and suffers from a number of limitations such natural color variation in videos and assumption of bees flying at a constant velocity. An improvement to the above model was subsequently presented in [54, 55].

In [56], the authors used digital video cameras to study the social interactions among bees. They were able to track up to 16 bees that walked freely on a flat surface and was able to correctly extract more than 95% of those bees' locations. Their system worked offline on pre-recorded videos and thus was able to monitor the behavior of the bees only within a

very limited area. Hence it was not suitable for long term monitoring concerning the health of the bees.

In [57], the authors presented a real–time imaging system for tracking honey bees leaving and entering the beehives from images acquired at the hive entrance. Honeybees were first detected and segmented using background subtraction method and then they were tracked using integrated Kalman Filter and Hungarian algorithm tracking method for counting the honey bees that entered and exited the beehive. To achieve faster performance they used an embedded GPU system and NVIDIA Jetson TX2 as the main computing device. They used a 4G LTE router for remote connectivity via the internet and also to transfer data. They reported their proposed approach was able to generate counts that had an accuracy of 93.9% when compared to ground truth data generated by manual counting. The evaluation was performed on 11 videos and since the data and also the process behind labeling each video was not made public, we were not able to replicate their results.

2.2.1 Use of Dpiv To Analyze Insect Motions

In [58] dpiv was used to measure the turbulence levels in a low turbulence wind tunnel. In that same study, by extending their findings, the authors suggested that dpiv can also be tuned to measure the aerodynamic performance of a small-scale flying device. When an insect, animal or a bird flies through the air, the traces of air particles that move as a result can be studied and used to analyze insect and animal flight patterns. The recent advancements in dpiv [27, 28, 59] has enabled researchers to start investigating the above flight patterns to understand how certain animals or insects fly and the use the relevant information to design micro-air vehicles. In [60], the authors used a high speed camera and applied dpiv to measure the vortex wake and kinematics of a swift's flight through a wind tunnel. In [61], the author's demonstrated the difference between the wakes of birds and small bat species by using dpiv to analyze the recorded wake images. It was shown that each wing of the bat generated it's own vortex loop along with a sign difference between the circulation on the outer wing and the arm wing during upstroke. Spedding et al. in [62] used dpiv to analyze the flight of thrush nightingales. They used dpiv to measure the

balance of forces during the upstroke and the downstroke of the bird's flight. Through their research they showed that it is possible to track the momentum in the wake of a flying animal. There has also been past research such as in [63] and [64], where the authors have used dpiv to study and analyze the flight patterns and the wake structure of nectar-feeding bats and dog-faced fruit bats respectively. The use of dpiv to investigate insect flight was first demonstrated in [65] in which the authors measured fluid velocities of a fruit fly and then examined the contribution of the leading–edge vortex in the overall force production during flight. The authors in [66] were the first to use dpiv to analyze the flow field around the wings of a tobacco hawkmoth while it flew through a wind tunnel. The authors also experimentally showed the flow separation near the leading edge of the wing during the insect's downstroke. In a different research [67], it was shown that dpiv could also be used to examine and map the air flows generated by dancing honeybees.

There have been past research projects that involved electronic beehive monitoring. One such study was reported by Rodriguez et al. in [68]. They proposed a system that would detect and track multiple insects and animals, with a special interest for monitoring the traffic of honeybees and mice. They designed their system based on deep neural network that associated detected body parts to whole insects and animals. The network initially predicts a set of 2D confidence maps of detected body parts along with a set of vectors that hold the associations among the detected body parts. Next they use greedy inference to select the most likely predictions for each part and then compiles the predictions into larger insects or animals. To detect the bee traffic that leave and enter the hive, the authors use trajectory tracking. The above gives reliable results under smaller traffic levels. The authors reported that the dataset used in the above study consisted of 100 fully annotated frames with 6–14 honeybees per frame. Since the dataset was not made public, we were not able to independently replicate the results. Furthermore in the above research, a standard Langstroth or Dadant hive, which is commonly used by beekeepers, was not used. Rather a smaller laboratory grade hive was used in the experiments. The proposed system also requires some modifications to the beehive in that a transparent acrylic plastic cover had

to be used in order to ensure that the bees remained in the focal plane of the camera.

In another research Babic et al. [69] proposed a system that would detect pollen bearing honeybees from videos recorded at the entrance of a hive. The proposed system included a specially designed wooden box mounted above the hive entrance, and also had a raspberry pi camera attached inside it. The wooden box was specially designed in a way to restrict bees from flying when in the field of view of the camera. To achieve the above, the authors put a glass plate on the bottom of a box just 2cm above the landing pad and in effect force the bees to crawl a distance of ≈11 cm while entering or leaving the hive. The authors reported the training dataset to be composed of 50 images of pollen bearing honeybees and 50 images of honeybees without pollen. The testing data set on the other hand consisted of 354 images of honeybees. Since the dataset was not made public, we were not able to independently replicate the reported results.

We found in [70], the review of various electronic, remote-sensing, and computer-based techniques for observing and monitoring insect movements in the field and the laboratory. In particular, they have reported in their review, the use of electronic bee counters to record bees entering and leaving the hive. Following that we have seen in [71] a practical use of bee counters in modeling the flight activity of honey bees at hive entrance. We also see the use of bee counters in a much recent study in [72], where the authors correlate foraging activity to weather conditions. Although the reported results were promising but in each of those cases, the use of a bee counter needed additional hardware and attachment to a standard hive. The reason is that to detect bee movements at the hive entrance 32 bi-directional channels of 10 mm × 6.5 mm × 6.5 mm (length × width × height) each and separated by 12.7 mm is required for the setup. The detection of bee inside each channel is achieved via an infrared diode that excites two photoreceptors. Following that, the break order of the receptors indicates whether a bee is entering (in) or leaving (out) of the hive. Thus we can see hardware modifications are needed to be done on the standard hive in order to be able to use the bee counters.

2.3 Analyzing Hive Audio

In [73], a distributed audio monitoring platform was presented where the system consisted of nodes and wireless sensor units with one node per apiary and one sensor unit per hive. The sensor unit consisted of an omnidirectional microphone and a temperature sensor. They monitored 10 hives continuously by taking 8 s audio samples every hour with a sampling rate of 6250 Hz. They reported on their observations of the daily patterns found in beehive sound volume and sound intensity at medium and low frequencies.

In [74], the authors studied honeybee swarms by designing five custom build observation hives sealed with glass cover along with cameras and microphones to detect movement and audio of bees within the hive. Swarming is the process by which a new honeybee colony is formed when the queen bee leaves the colony with a large group of worker bees. The video and audio data from the hives were monitored daily by human observers. The authors reported that in three different colonies where bees had swarmed, the production of piping signals gradually increased starting one hour before swarm exodus and ultimately peaking during swarm departure. Similar patterns were also seen in the bee movement in all the three colonies.

In [75], a monitoring system was designed by the authors to analyze bee buzzing during swarming. They recorded sound through three omnidirectional microphones and also monitored the temperature and relative humidity inside the hive by means of a datalogger. The system was connected to a computer in a nearby barn via underground cables. They observed the hives for 270 hours and reported that when swarming happened, there was a gradual increase in the buzzing frequency from 110 Hz to 300 Hz. The audio from the beehive was recorded at 2 Khz and was analyzed using Matlab®. In general they proposed a method to predict swarming period based on labeling the sounds.

In [34], the authors attached accelerometers on the hive walls to predict swarming activity by analyzing the vibrations. They logged data from the accelerometers for a period of 8 months and reported the finding of a multi-dimensional time vector from the data which had a signature specific to swarming activity.

In a similar approach as above in [76], the authors placed accelerometers on the outside walls of the hives in an attempt to capture and analyze the acoustic vibrations that bees use to communicate. They studied one such pulse named 'stop signal' for a period of 9 months. They concluded that monitoring changes in the 'stop signal' could provide us valuable information regarding the health of the colony. They also reported that analyzing the frequencies associated with the vibration could be helpful in detecting swarming several day in advance.

The authors in [3] presented a system capable of detecting stress in a honeybee colony by comparing the acoustic fingerprint of the beehive with that of a healthy hive. They extracted Peak Frequency, Spectral Centroid, Bandwidth and Root Variance Frequency as the four features and performed feature engineering by using Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) .

In [16], the authors presented an audio processing algorithm that was capable of digitizing bee buzzing into sequences of A440 piano notes. Their goal was to view the note sequences as a time series data and the correlate them to other timestamped data for pattern recognition.

2.3.1 Use of CNN To Classify Audio Samples

Through representation learning it is possible to automate the acquisition of patterns from raw signals for detection and classification [77]. Standard machine learning techniques are limited in their abilities to learn from raw data since they require considerable feature engineering to convert raw signals into feature vectors which could then be used for classification. The technique of deep learning is often more suitable in acquiring multi-layer representations from raw data because real life data often contain non-linear representations which low-order functions have a hard time modeling. There have been several defining studies where deep learning methods have been successfully applied to image classification [78–80], speech recognition and audio processing [81, 82], music classification and analysis [83–85], environmental sound classification [86], and bioinformatics [87, 88].

In the field of audio processing, convolutional neural networks (CNNs) have been

trained to classify audio signals either by training on raw audio waveforms or by extracting feature vectors from them. For examples in [86], the author discussed the design of a CNN to classify short audio samples of environmental sounds. The CNN consisted two convolution layers with maxpooling and two fully connected layers and was trained on the segmented spectrograms generated from the audio data. They reported that on three public datasets of environmental and urban recordings, their designed CNN architecture outperformed the results obtained by using random forests with mel frequency cepstral coefficients (MFCCs) and zero crossing rates as the features. Their proposed method also performed par with some state-of-the-art audio classification approach as well.

On the other hand, the authors in [89], developed a new CNN model capable of learning from the raw audio waveform directly. They trained their model by transferring knowledge from computer vision that classified events from unlabeled videos. The representation learned by their propose CNN model was evaluated and it obtained state-of-the-art accuracy on three standard acoustic datasets. In [90], the authors showed that it is possible to predict the next sample in an audio sequence by training a generative network. The proposed architecture of their model contained 60 layers and it sampled raw audio at a rate of 16 kHz to 48 kHz. In [91], the authors developed a chord recognition system by designing a CNN to classify 5-s tiles of pitch spectra. Their model produced state-of-the-art performance across a variety of benchmarks.

2.4 Effect of Weather On Foraging Activity

There have been different studies, where it has been reported that various weather conditions play a role in the foraging activity across different bee species. The authors in [92] have reported the effect of rainfall on honey bee foraging activity. They observed a higher level of foraging activity prior to heavier rainfall. They suggested that honey bees are able to sense the likelihood of upcoming rainfall. In their experiments they used 3 honeybee colonies each having 6000 worker bees and 1 laying queen. They attached an RFID tag to newly emerged workers (n=300) of each experimental colony and monitored them for 34 days. RFID detectors and tag systems can be rather expensive, and utmost care must be

taken so that the tags or the glues does not affect the bee behavior or their survivorship. Our BeePi monitors on the other hand does not require the tagging of individual bees and the sensors that we use are non-obtrusive. The dpiv based algorithm described in Chapter 5 is able to count the bee motions which is on par with human counts, thus eliminating the need for RFID based tagging of bees in our electronic beehive monitoring system.

A novel study by the authors in [93] found out the relationship between weather variables, such as temperature and solar radiation to the foraging activity of bee colonies. They gathered foraging data by using a photoelectric beecounter ('Apicard') placed at the hive entrance, for 30-min periods (data was recorded every 15s and was pooled) over 23 consecutive days from a single colony of bees and then analyzed those data with respect to ambient weather conditions. They reported that a positive correlation was found between the foraging activity and both temperature and solar radiation. It was also reported that the positive correlation between the bee activity and solar radiation was only till a certain threshold, after which the bee activity went down as the solar radiation increased.

There has also been research such as in [94], which showed the effects of temperature on the pollination activity of two separate bee species in an apple orchard in Girona (northeast Spain). It was a controlled study, that was performed during the years 1993, 1994 and 1995 in an apple orchard by monitoring 20 honey bee hives. They installed a weather station during 1995 and studied the effects of ambient temperature, relative humidity, solar radiation, and wind speed by recording data every 10 min. The weather station was located 2m above the ground and 800m away from the orchard on a flat terrain. They studied how weather conditions affect pollination and along with that they investigated the design of pollination strategies to optimize fruit yield when the weather conditions were suboptimal. Since the above investigation was performed in a controlled environment, it is challenging to replicate the findings under natural conditions.

Following that, various research, such as in [95] and [72] have been conducted on studying the correlations between foraging behavior of honeybees and climatic conditions such as temperature, humidity and wind velocity. In [95], the investigation was performed during

the 2002 growing season at Kneevi Vinogradi location, Baranja County (northeast Croatia). They performed the study on honey bees visiting sunflower inflorescences at 100, 200 and 300 meters, by recording and counting the individual bees present four times a day (9am, 11am, 1pm and 5pm). They studied the effect of temperature, humidity, precipitation and wind speed and reported the correlation between weather and foraging activity by using Spearman correlation coefficient. They reported a positive correlation between temperature and foraging activity and negative correlation between foraging activity and humidity, precipitation and wind speed. However the authors did not report the number of hives that were used for the experiments nor the data was made public.

In [72], the authors performed their analysis on data from two different bee hive colonies during the foraging seasons of July–September 2013 and June–September 2014. They presented their analysis based on 42 days of data in 2013 and 74 days of data in 2014, with data recorded with a time resolution of 1samples/min. They used a multichannel electro-optical beecounter to measure the bee egress rate and placed a weather station near the hives to record the meteorological data related to rain, solar radiation, temperature, humidity, wind speed and wind direction. A multichannel electro-optical beecounter requires hardware modifications if used on a standard hive and thus can effect the natural surroundings for the bees. They presented the effect of different meteorological variables on foraging activity and their correlation was in line with the results from previous studies in the literature. They also explored the utility of predictive modeling by fitting a generalized linear model to their data using meteorological data as the predictors and then used it to predict bee egress rate. They reported promising results and also suggested the use of other meteorological variables and even bee audio to further understand their combined effect on foraging activity. We were not able to replicate or use their models as their data was not made public. Although a clear understanding of the different factors that might effect foraging activity is difficult, the above studies suggest that local weather conditions do play a significant role. Since weather conditions differ from place to place, we believe there is indeed scope for further analyzing the effect of different meteorological variables on bee foraging effort.

CHAPTER 3
OMNIDIRECTIONAL BEE COUNTING FROM LANDING PAD IMAGES

3.1 Goal

In this chapter we will be discussing a method to compute bee counts from static images of beehive landing pads. The camera in our BeePi system points vertically on the landing pad over which the bees enter or leave their respective hives. From a static image, the proposed method will first locate the landing pad and then determine and remove any skew from the landing pad images. We have proposed two algorithms to detect bees from the localized landing pad images. One of the algorithm is based on 1D Haar Wavelet Transformation and the other is based on the technique of contour analysis. The flowchart of proposed method in this chapter is show in Figure 3.1. The proposed method does not distinguish between incoming and outgoing bee traffic. In other words, the proposed method gives us omnidirectional bee counts from static landing pad images. Due to the hardware limitations of the camera, the method also does not distinguish between different types of bees like, worker, drone or queen bee. By the end of this chapter we will be able to see that computer vision can act an important tool in determining the health of a bee hive by helping us determine bee counts in static images; thereby acting as a strong contender to be included in EBM software systems.

3.2 Landing Pad Localization

The process of localizing a landing pad consists of three steps, adjusting the brightness of the image, determining the skewness in the images by locating the bounding areas of the landing pad and finally calculating the skew angle and adjusting the landing pad image accordingly.

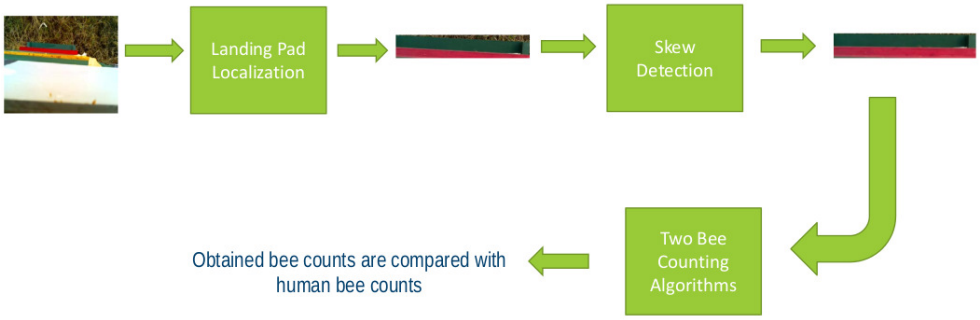


Figure 3.1: Flowchart of Proposed Model

(a) Image Taken In The Morning (b) Image Taken During Middle of The Day

Figure 3.2: Landing pad images from two separate beehives recorded by the BeePi camera module at different times of the day

3.2.1 Adjustment of Image Brightness

The beehives are located outdoors in areas where during certain times of the day the sun shines directly on. As a result of that, the brightness in the images taken by the BeePi camera during those times is significantly higher. As an example we see two images taken by the BeePi camera during different times of the day in Figure 3.2. Image in Figure 3.2a was taken in the morning before noon and image in Figure 3.2b was taken when the sun was shining directly on the landing pad.

To adjust the brightness of the image, Img as in Figure 3.2b, the first step is to convert it to a grayscale image and calculate the average brightness. Next if the average brightness is less than a predefined threshold, then Img is converted from `RGB` to `YCrCb` color space, where `Y` is the luminance component and `Cr` and `Cb` are the blue difference and red difference chroma components, respectively. The `Y`, `Cr` and `Cb` channels are separated out and intensity

(a) Image Before Brightness Adjustment (b) Image After Brightness is Adjusted

Figure 3.3: Images before and after adjustment of brightness

histograms are computed for each channel. Next, all the histograms are equalized so that the intensity spread is uniform across the three channels. Finally the three adjusted channels are combined and the combined image is converted back to `RGB` color space. Figure 3.3 shows the images before and after the brightness adjustment. In Figure 3.3b we see that the brightness of the image is more uniform and the bees on the landing pad are more distinct and visible compared to the bees on the landing pad in Figure 3.3a.

3.2.2 Locating The Bounding Areas of The Landing Pad

We will design our bee counting algorithms such that it counts the number of bees present on the landing pad at any given time. But rather than processing the entire image as in Figure 3.3b and trying to count the bees, it would be better if we could locate the bounding region of the landing pad for faster analysis. But in Figure 3.2a we can see that there might be cases where the landing pad is crooked in the images taken by the BeePi camera. In other words, the landing pad in the image is skewed. To determine the skew angle of the landing pad, we first need to determine the portion of the image which tightly bounds the landing pad. The skew in the image may be caused by camera movement during wind gusts or even during physical hive inspections by a beekeeper. In Figure 3.2a, if we observe closely then we can see the presence of grass around the landing pad region. Grass has significantly more edges or corners than the area of the beehive around and above the landing pad. By capturing this texture difference we will be able to separate out the

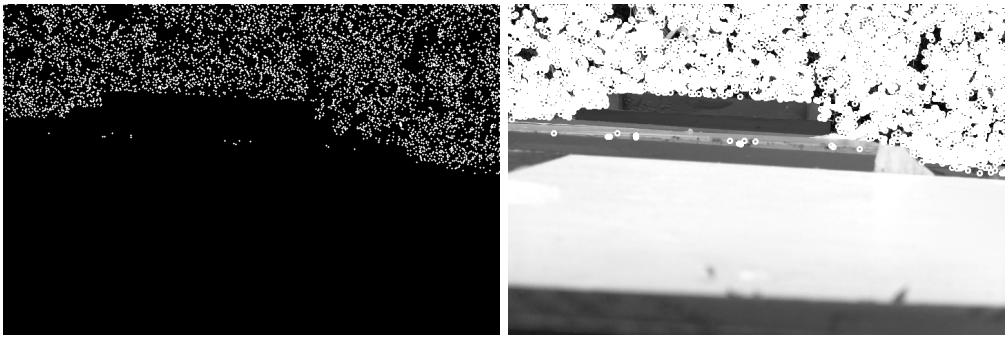
(a) Original Image (b) I_{corner} With Detected Corners

Figure 3.4: Images before and after the application of Harris Corner Detection

grassy region from the landing pad. The morphological operation that helps us to capture that texture difference is corner detection. Harris Corner detection [?cite?] is one such technique to detect corners in an image. Figure 3.4 shows the result after applying Harris Corner detection. The result of Harris Corner detection is image I_{corner} (Figure 3.4b), which has roughly the raw corners in the image in Figure 3.4a. We can see that the corner points are not distinct. So in the next following steps, we will work towards generating distinct corner points from I_{corner}.

Towards that end, we create a clone of I_{corner} and perform the morphological operation of erosion on it. Erosion helps in eroding or shrinking the regions of detected corner regions. Thus erosion helps in making the gaps between the detected corner regions larger and eliminating small details on the boundary areas of each region. The final result of erosion is an image, $I_{erosion}$, where the foreground pixels have shrunk in size, and holes within those areas have become larger thereby emphasizing on individual corner points.

Next $I_{erosion}$ is compared with I_{corner} by performing a pixel wise AND operation to retain only the unmodified pixels between $I_{erosion}$ and I_{corner}. This results in an image (I_{final}) with distinct corner points as shown in Figure 3.5a. Figure 3.5b shows the original image with the final distinct corner points plotted on. For the above example, we can see that our method of detecting the corner points has been successfully able to detect all the grassy region above the landing pad. Next we will use the above information about the positions of the corner points to determine the skewness of the landing pad.

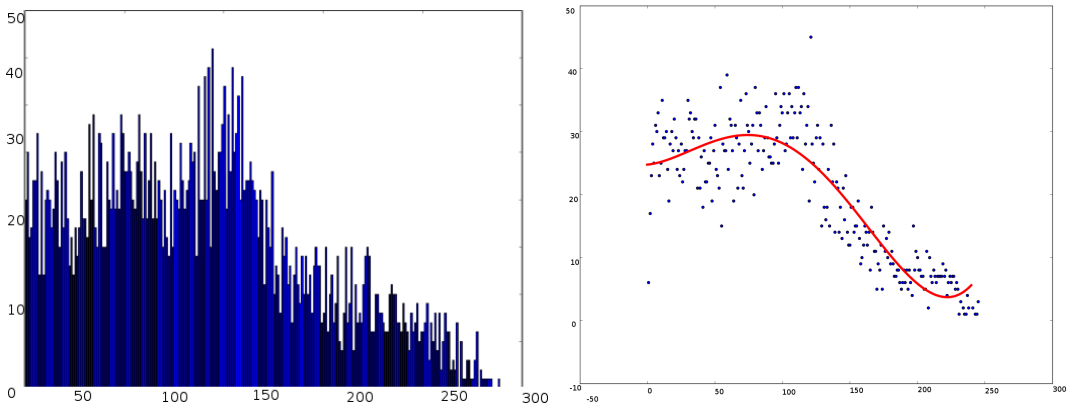


(a) I_{final} With Distinct Corners (b) Original Image With Corner Points

Figure 3.5: Images with distinct corner points

Using the corner point coordinates, we create two separate hash maps, one for the rows and one for the columns. Each hash map corresponding to a row i, will hold the number of the columns in that row, where the corners were detected. Similarly each hash map corresponding to a column j, will hold the number of the rows in that column, where the corners were detected. Mathematically, let there be P rows and Q columns in I_{final}. Let (i, j) be the coordinate in the image I_{final} which is detected as a corner point. We represent the presence or absence of a corner point at the location (i, j) by (r_i, c_j), where $(r_i, c_j) = (1, 1)$ if it is a corner point and (0,0) otherwise. Both values (r_i and c_j) need to be 1 in order to be classified as a corner point. So the hashmap ($M_r(i)$) corresponding to row r_i will hold the count of the y–coordinate elements in the set $(r_i, c_0), (r_i, c_1), (r_i, c_2), \ldots, (r_i, c_Q)$ where $r_i = 1$, i.e. $M_r(i) \rightarrow \Sigma c_{iy}$, where $0 \leq y \leq Q$ and c_{iy} represents the y–coordinate (c_y) corresponding to (r_i, c_y) and $r_i = 1$. In the above equation, $c_{iy} = 1$, if (i, y) is a corner point or else $c_{iy} = 0$. In a similar way, the hashmap ($M_c(j)$) corresponding to column c_j will hold the count of the x–coordinate elements in the set $(r_0, c_j), (r_1, c_j), (r_2, c_j), \ldots, (r_P, c_j)$ where $c_j = 1$, i.e. $M_c(j) \rightarrow \Sigma r_{xj}$, where $0 \leq x \leq P$ and r_{xj} represents the x–coordinate (r_x) corresponding to (r_x, c_j) and $c_j = 1$. In the above equation, $r_{xj} = 1$, if (x, j) is a corner point or else $r_{xj} = 0$.

Either of the hash maps could could be used to start our analysis. In our study we decided to use the hash map, M_r to construct histogram H_r which is the horizontal projection of the corner counts across each row. For H_r, the bins represent the row numbers



(a) Horizontal Histogram (b) Scatter Plot with Fitted Line

Figure 3.6: Scatter plot of horizontal histogram projections with the best fitted line

from the image I_{final} and the bin value is the count of the corners for that row, i.e, the value at bin i is $M_r(i)$. The histogram is shown in Figure 3.6a. We will use the histogram to detect and identify positions of significant changes in the counts of the corner points.

If we observe Figure 3.5a, we can see that the number of detected corner points are gradually reduced after a certain section as we move down the image row by row or horizontally from the top. We can see the replication of the above scenario in the histogram in Figure 3.6a. As we move from left to right, the count of the number of corners is reduced. We should keep in mind that in an Image, the origin is at the top left corner. If we compare Figure 3.4a and Figure 3.5a, we can safely say that the area of the landing pad has fewer corners detected than the grassy area above the landing pad. Thus the decline in the corner counts would help us identify the upper coordinate positions of the landing pad.

To have a better understanding and visualization of the spread of corner points in the histograms in Figure 3.6a, we convert the histograms to 2D scatter plots and work towards finding the best fit line through those points. But before proceeding, we remove the outliers in the scatter plots by considering the L1 distance between corresponding points. This outlier removal is necessary as it might skew the best fit line. We found that the best fit line is a 4^{th} degree polynomial that is able to represent the distribution of the corner points in the scatter plot as shown in Figure 3.6b.

Equation (3.1) shows the 4^{th} degree polynomial line which represents the horizontal projections the best as seen in Figure 3.6b; the 3^{rd} and 4^{th} degree coefficients are very small and have been discarded.

$$r(x) = 0.0025x^2 + 0.017x + 24.7 \tag{3.1}$$

The above curve in equation (3.1) is best fit line for the projection of the corner counts for each row. Thus we will use it to identify the location of the upper and lower bounds of the landing pad in the image. Towards that end we create a sliding window W^i_{row} of 10 points along the points on $r(x)$. At each step W^i_{row} is shifted by 5 points along the axis of the curve. Let M^i_{row} be the median of the sliding window W^i_{row} at step i and similarly M^{i-1}_{row} is the median of the sliding window W^{i-1}_{row} at step $(i-1)$ respectively. Let the change between the corresponding medians of two sliding windows at step $(i-1)$ and i be defined as, $\Delta^i_{row} = M^i_{row} - M^{i-1}_{row}$. If we observe the curve in Figure 3.6b we can see when $\Delta^i_{row} < 0$, it means that the slope of the curve is decreasing and so in order words it means that the number of corner points detected is reducing. We can see from Figure 3.5b, that the corner points start reducing as we approach the landing pad from the top. Thus the sliding window,W^i_{row} at step i for which $\Delta^i_{row} < 0$ indicates the location for the upper bound of the landing pad. To get a more accurate approximate positioning of the upper ordinate of the landing pad, we calculate the first quartile $Q1_{W^i_{row}}$ of the 10 points in W^i_{row}. Thus row number $Q1_{W^i_{row}}$ is the upper starting point of the landing pad.

Based on the positioning of the BeePi camera, the width of the landing pad is calculated as 50 pixels wide. This width changes as we move further into the beekeeping season while adding more supers to the beehive. Thus given the width and the starting row number of the landing pad, end row number of the landing pad is calculated as $Q1_{W^i_{row}} + 50$. Next we take these two points and crop the respective portion from Figure 3.7a. We can see in Figure 3.7b that the above cropping resulted in a new image $I_{croppedRow}$ the landing pad has correctly been identified.

Although in Figure 3.7b, we have been able to detect the upper and lower bounds of

(a) Original Image

(b) $I_{croppedRow}$

Figure 3.7: Result after row based cropping of the landing pad

(a) Corner points drawn on $I_{croppedRow}$

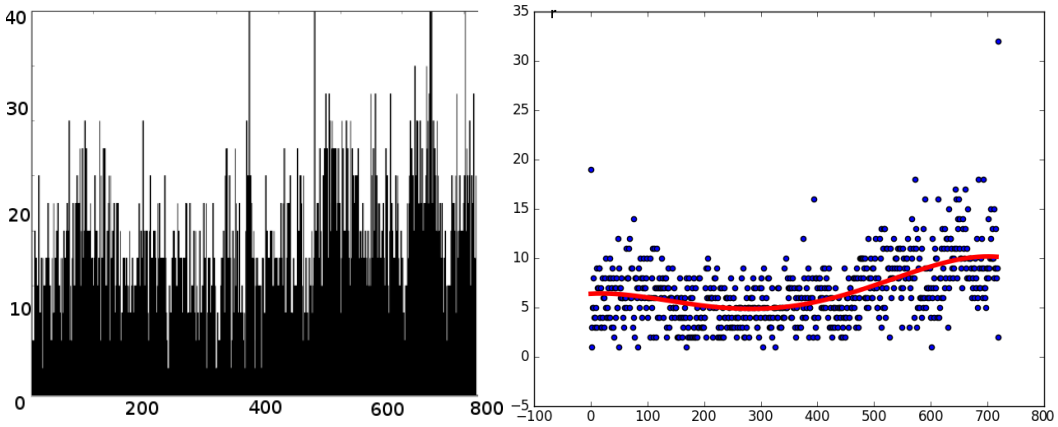
(b) Corner points detected

Figure 3.8: Corner detection in $I_{croppedRow}$

the landing pad but there is still some portion of the grassy area that needs to be removed on the either side, before we can find the skew angle of the landing pad. We will follow the similar steps of detecting the corner points and creating a hash map $H_c(j)$ of the column projections. The detected corner points are shown in Figure 3.8b. Next we use $H_c(j)$ and generate the histogram as shown is Figure 3.9a. We will start by finding the best fit line that represents the scatter plot of the corner counts in different columns. Equation (3.2) shows the best fitted 4^{th} degree polynomial line for vertical projections as seen in Figure 3.9b; the 3^{rd} and 4^{th} degree coefficients are very small and are discarded.

$$c(x) = -0.0001x^2 + 0.004x + 6.4 \tag{3.2}$$

From the best fit line in Figure 3.9b, we can see that there is a decline in the slope of the curve followed by a rise. Now if we look at Figure 3.8b and observer the variation of corner points from left to right of the image, we can see that a decline in the number of corner points indicates that, within that particular region is the vertical starting point of the landing pad. On the other hand, we can see a steady rise in the corner points near the right end of Figure 3.8b, which tells us that the vertical ending point of the landing

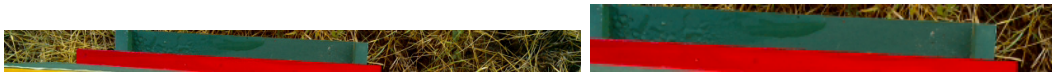


(a) Vertical Histogram (b) Scatter Plot with Fitted Line

Figure 3.9: Scatter plot of vertical histogram projections of column based corner points with the best fitted line

pad is in that region. Thus looking back at the best fit line in Figure 3.9b tells us that the region where the curve slope decreases is the starting region of the landing pad and where the slope steadily increases indicate the ending region of the landing pad.

Towards that end we create a sliding window W^i_{col} of 10 points along the points on $c(x)$. At each step W^i_{col} is shifted by 5 points along the axis of the curve. Let M^i_{col} be the median of the sliding window W^i_{col} at step i and similarly M^{i-1}_{col} is the median of the sliding window W^{i-1}_{col} at step $(i-1)$ respectively. Let the change between the corresponding medians of two sliding windows at step $(i-1)$ and i be defined as, $\Delta^i_{col} = M^i_{col} - M^{i-1}_{col}$. If we observe the curve in Figure 3.9b we can see when $\Delta^i_{col} < 0$, it means that the slope of the curve is decreasing and so in order words it means that the number of corner points detected is gradually reducing. We can see from Figure 3.8b, that the corner points start reducing as we approach the landing pad from the left. Thus the sliding window,W^i_{col} at step i for which $\Delta^i_{col} < 0$ indicates the location for the vertical starting point region of the landing pad. To get a more accurate approximate positioning of the starting ordinate of the landing pad, we calculate the first quartile $Q1_{W^i_{col}}$ of the 10 points in W^i_{col}. Thus column number $Q1_{W^i_{col}}$ is the vertical starting point of the landing pad. Similarly when $\Delta^j_{col} > 0$, it means that the slope of the curve is increasing which means that the number of corner points detected is gradually increasing too. We can see from Figure 3.8b, that the



(a) $I_{croppedRow}$ (b) $I_{croppedColumn}$

Figure 3.10: Result after column based cropping of the landing pad

Figure 3.11: Result after detecting corners in Figure 3.10b

corner points start increasing as we approach the end of the landing pad on the right. Thus the sliding window,W^j_{col} at step j for which $\Delta^j_{col} > 0$ indicates the location for the vertical ending point region of the landing pad. To get a more accurate approximate positioning of the end ordinate of the landing pad, we calculate the first quartile $Q1_{W^j_{col}}$ of the 10 points in W^j_{col}. Thus column number $Q1_{W^j_{col}}$ is the vertical end point of the landing pad. Using the two points we crop the respective portion out of Figure 3.10a. This results in the new image $I_{croppedColumn}$ as shown in Figure 3.10b.

We can see in Figure 3.10b, that there could be still some portions in the image which could still have some grassy areas. This happens when the landing pad in the image is skewed. We will use the grassy region on the top to determine the skew angle. But before that we need to remove the additional grassy region that might occur on either side of the landing pad after our first iteration. So in the next few steps we will be performing another round of corner detection on Figure 3.10b and localizing the ending points of the landing pad using the best fit line. Figure 3.11 shows the result of the corner detection on Figure 3.10b. We then take those corner points and generate the scatter plot to find the line that best represent the spread of corner points. Figure 3.12 shows the line obtained by fitting a fourth degree polynomial line. We can discard the initial points on the line till we get the first non zero value. We can see that towards the end of the line, the corner counts increase and thus as a result the slope of the line increases gradually.

Now if we look at Figure 3.11 we can see that the increase of corner counts near the end indicates the presence of the grassy region. So we will use this information to localize the right end of the landing pad. We will look for the region where there is a continuous

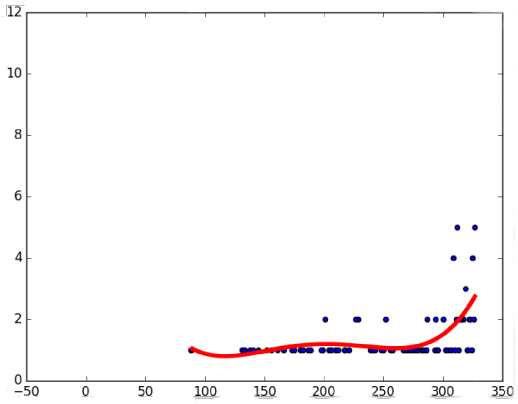
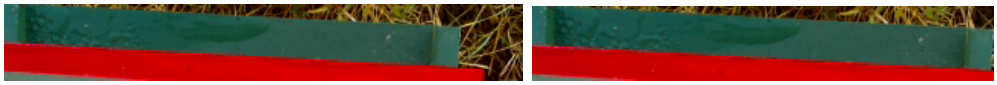


Figure 3.12: Best fit line for the column projections of corner points from Figure 3.11

increase in the slope of the line. Towards that end we create a sliding window W^i_{col} of 10 points along the points on the best fit line in Figure 3.12. At each step W^i_{col} is shifted by 5 points along the axis of the curve. Let M^i_{col} be the median of the sliding window W^i_{col} at step i and similarly M^{i-1}_{col} is the median of the sliding window W^{i-1}_{col} at step $(i-1)$ respectively. Let the change between the corresponding medians of two sliding windows at step $(i-1)$ and i be defined as, $\Delta^i_{col} = M^i_{col} - M^{i-1}_{col}$. If we observe the curve in Figure 3.12 we can see when $\Delta^i_{col} > 0$, it means that the slope of the curve is increasing and so in order words it means that the number of corner points detected is gradually increasing. We can see from Figure 3.11, that the corner points start increasing as we approach the grass on the right side of the landing pad. Specifically, we would like to find the region on the curve where the increase in slope is above a certain threshold θ. If we look at Figure 3.11, we can see that although the corner counts have started to increase from the middle portion of the image, but it is only at the right end when the corner counts increases sharply and that indicates the vertical end of the landing pad. Thus our use of a threshold value, helps us to capture that specific region where the curve starts increasing rapidly. Let there be N points on the curve, then $W_{col} = \lceil \frac{N}{5} \rceil$ is the number of windows or steps over the curve. Let after step or window j, the slope of the curve starts to increase, so we have $I_{col} = \{I^k_{col}\}$, where $I^k_{col} = \{k | M_k - M_{k-1} > 0\}$ and $j \leq k \leq W$. From the set of points in I_{col}, we want to find the point p such that $p = min\{i | i \in I^k_{col} \wedge \Delta^i_{col} > \theta\}$. So points in the sliding window W^p_{col} gives us the location of the vertical end of the landing pad on the right side. To get a more



(a) $I_{croppedColumn}$ (b) $I_{croppedFinal}$

Figure 3.13: Result after second iteration of column based cropping of the landing pad

(a) Corners detected in left half of cropped image in Figure 3.13b

(b) Corners detected in right half of cropped image in Figure 3.13b

Figure 3.14: Corners detected in Figure 3.13b

accurate approximate positioning of the starting ordinate of the landing pad, we calculate the first quartile $Q1_{W^p_{col}}$ of the 10 points in W^p_{col}. Thus column number $Q1_{W^p_{col}}$ is the vertical end point of the landing pad on the right side. Next we crop the respective portion out of Figure 3.13a. This results in the new image $I_{croppedFinal}$ as shown in Figure 3.13b.

3.2.3 Finding the skew angle

After we have been able to find the bounding coordinates of the landing pad in Figure 3.13b, the final step is to find the skew angle and rotate the image accordingly. We start by following the similar procedure of finding the corners in Figure 3.13b. After we generate the corners we divide the image into two halves and count the number of corners detected in each half. The larger the count, the more grassy region in that particular half. In our example, the right half had the larger corner counts as shown in Figure 3.14b.

From the corners detected, we create a hash map for corner counts in each row. Each hash map corresponding to a row i, holds the number of the columns in that row, where the corners were detected. Mathematically, let there be P rows and Q columns in Figure 3.14b. Let (i, j) be the coordinate in the image which is detected as a corner point. We represent the presence or absence of a corner point at the location (i, j) by (r_i, c_j), where $(r_i, c_j) = (1, 1)$ if it is a corner point and (0,0) otherwise. Both values (r_i and c_j) need to be 1 in order to be classified as a corner point. So the hashmap $(M_r(i))$ corresponding to row r_i will hold the count of the y–coordinate elements in the set $(r_i, c_0), (r_i, c_1), (r_i, c_2), \ldots, (r_i, c_Q)$

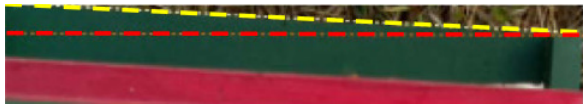


Figure 3.15: Skew angle detection in Figure 3.13b

Figure 3.16: Pad image in Figure 3.13b rotated counterclockwise to eliminate skew

where $r_i = 1$, i.e. $M_r(i) \rightarrow \Sigma c_{iy}$, where $0 \leq y \leq Q$ and c_{iy} represents the y–coordinate (c_y) corresponding to (r_i, c_y) and $r_i = 1$. In the above equation, $c_{iy} = 1$, if (i, y) is a corner point or else $c_{iy} = 0$. Next we find row p_{right} where $p_{right} = min\{i | i \in M_r(i) \wedge M_r(i) = 0\}$. Thus row p from the top, gives use the location where the landing pad starts in Figure 3.14b. Similarly for the left half of the corners detected in Figure 3.14a, we find row p_{left} where $p_{left} = min\{i | i \in M_r(i) \wedge M_r(i) = 0\}$. Thus we have the 3 required points to find the skew angle of the landing pad in Figure 3.13b. The three points are $(p_{left}, 0)$, $(p_{right}, 0)$ and $(p_{right}, Q-1)$.

A line is drawn between the points $(p_{left}, 0)$ and $(p_{right}, Q-1)$. This virtual line is depicted by a yellow dash line in Figure 3.15. If the corner counts increase from left to right of the image then the skew angle is $\angle(p_{left}, 0)$, $(p_{right}, Q-1)$, $(p_{right}, 0)$. If the counts decrease from left to right of the image, the skew angle is $\angle(p_{left}, Q-1), (p_{right}, 0), (p_{right}, Q-1)$. Figure 3.16 shows the rotated image after adjusting for the skew angle.

3.3 Counting Bees On The Detected Landing Pad

In this section we will present an algorithm to count the bees present on the cropped landing pad as seen in Figure 3.16.

3.3.1 Using Haar Wavelet Transform To Count The Bees

We used 1D Haar Wavelet Transform [??] to design the first algorithm to count bees on the landing pad. In 1D HWT, a signal is represented as a vector in R^n where $n = 2^k$

and $k \in N$. As formulated in [??], let $W_a^{(k)}$ be a $2^k \times 2^k$ matrix for computing k scales of the 1D HWT. We can compute the above matrix using the n canonical base vectors of R^n. Let $x = (x_0, ..., x_{2^k-1})$ be a signal in R^n, then the k^{th}-scale 1D HWT of x is defined as in Equation (3.3).

$$W_a^{(k)} \cdot x^T = y \tag{3.3}$$

Equation (3.4) shows the corresponding values in column vector y.

$$y^T = (a_0^{(0)}, c_0^{(0)}, c_0^{(1)}, c_1^{(1)}, ..., c_{2^{k-1}-1}^{(k-1)}) \tag{3.4}$$

In the above Equation 3.4, $a_0^{(0)} = \mu(y)$ and c_i^j is the coefficient of the i^{th} Haar wavelet at scale j. Haar Wavelet Transformation is primarily used to detect significant changes and detect spikes in signal values. We will be working with four types of spikes to detect and classify signal changes: up-down triangle, up-down trapezoid, down-up triangle, and down-up trapezoid. Figure 3.17 shows up-down triangle and trapezoid spikes and Figure 3.18 shows down-up triangle and down-up trapezoid spikes.

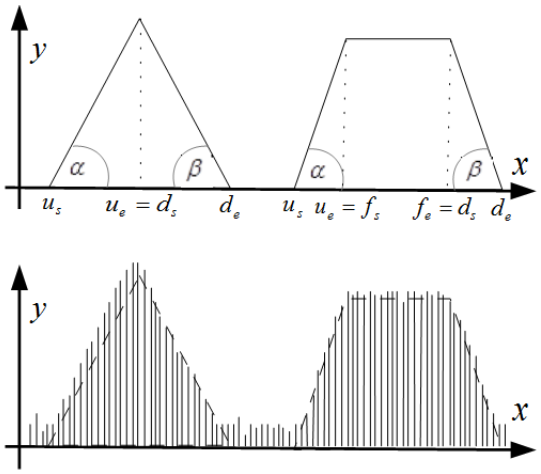


Figure 3.17: Up-down spikes

In Figures 3.17 and 3.18, the lower graph represents the possible values of probable values

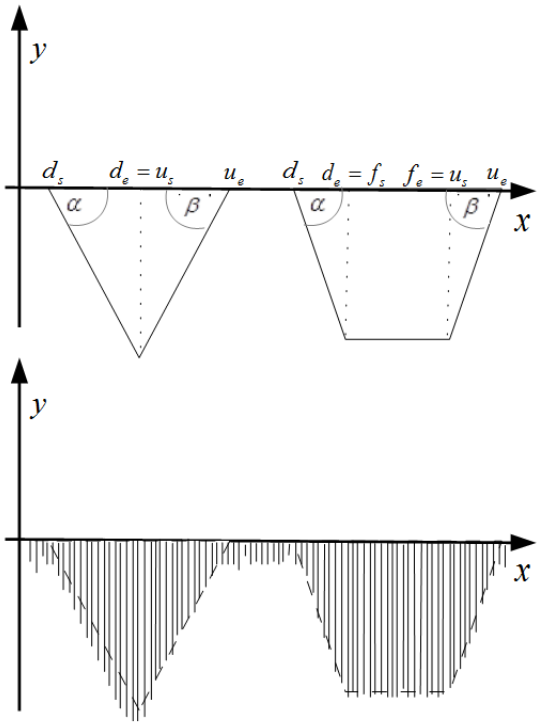


Figure 3.18: Down-up spikes

of Haar Wavelets at scale k. A spike S is represented by 9-tuples of real numbers as shown in Equation (3.5).

$$S = (u_s, u_e, \alpha, f_s, f_e, \gamma, d_s, d_e, \beta) \tag{3.5}$$

The first two elements of the 9-tuple, u_s and u_e, represent the starting and the end points of the spike's climb when the wavelet coefficients of the 1D HWT increase, as shown in Figure 3.17 and 3.18. The third element of the tuple, α represents the sharpness of the spike's climb. If $cu_s^{(k)}$ and $cu_e^{(k)}$ are the k^{th} scale wavelet coefficient ordinates at u_s and u_e, then the angle or the sharpness of the spike's climb is measured as $\alpha = tan^{-1}(u_e - u_s + 1, cu_e^{(k)} - cu_s^{(k)})$.

The flat part in the trapezoidal spike in Figure 3.17 and 3.18 is represented by the fourth, fifth, and sixth elements of the 9-tuple in Equation (3.5). f_s and f_e, represent the starting and the end points of the flat segment, where the wavelet coefficients of the 1D

HWT either remains the same or shows minor fluctuations. If $cf_s^{(k)}$ and $cd_e^{(k)}$ are the k^{th} scale wavelet coefficients corresponding to f_s and f_e, respectively, the spike's flatness is estimated as $\gamma = tan^{-1}(f_e - f_s + 1, cf_e^{(k)} - cf_s^{(k)})$. From the Figures 3.17 and 3.18 we can see that the absolute value of γ tends to be close to 0.

Finally the segment of the spike where the descend happens as seen in Figure 3.17 and 3.18 is represented by the seventh, eighth, and ninth elements of the 9-tuple in Equation (3.5). d_s and d_e, represent the starting and the end points of the spike's descend when the wavelet coefficients of the 1D HWT decreases due to reduction in the samples from the analyzed signal. The last element of the tuple, β represents the sharpness or angle of the spike's descend. If $cd_s^{(k)}$ and $cd_e^{(k)}$ are the k^{th} scale wavelet coefficient ordinates at d_s and d_e, respectively, then the sharpness or the angle of the spike's decline is measured as $\beta = tan^{-1}(d_e - d_s + 1, cd_e^{(k)} - cd_s^{(k)})$.

An image can be interpreted as a signal. Thus given an image, the spikes can be computed for either each row or column depending upon the application domain. In our application, we will be computing the spikes along each row r. For each row r, the column indices for the actual pixels covered by each spike at scale j is represented using Equation (3.6).

$$p(j, s, e) = \{i | 2^j \cdot s \leq i \leq 2^j \cdot (e + 1) - 1\} \tag{3.6}$$

In the above equation, s and e are the starting and ending wavelet coefficients at scale j. For up-down spikes as in Figure 3.17, $s = u_s$ and $e = d_e$, whereas, for down-up spikes as in Figure 3.18, $s = d_s$ and $e = u_e$. Let n be the number of rows in the image and the total number of up-down spikes in row r be U_r. Then set of pixel columns covered by up-down spikes in row r is represented in Equation (3.7), where the scale is j and s_z and e_z are the beginning and end positions of spike z in row r respectively.

$$Z_{U,j}^r = \bigcup_{z \in U_r} p(j, s_z, e_z), \text{ where } 0 \leq r \leq n - 1 \tag{3.7}$$

Similarly, if D_r be the number of down-up spikes in row r. Then the set of pixel columns

covered by down-up spikes in row r is represented in Equation (3.8), where the scale is j and s_z and e_z are the beginning and end positions of a down-up spike z in row r respectively.

$$Z^r_{D,j} = \bigcup_{z \in D_r} p(j, s_z, e_z) \tag{3.8}$$

As you can infer from Equation (3.7) and (3.8), certain column pixels would be counted more than once. Thus the number of unique pixels covered by the up-down and down-up spikes in row r is represented as follows in Equation (3.9).

$$Z^r_j = (Z^r_{U,j} - Z^r_{D,j}) \cup (Z^r_{D,j} - Z^r_{U,j}) \cup (Z^r_{D,j} \cap Z^r_{U,j}) \tag{3.9}$$

Thus in an image I with n rows, the total number of pixels covered by the up-down and down-up spikes is formulated as follows in Equation (3.10).

$$X_j(I) = \sum_{r=0}^{n-1} |Z^r_j| \tag{3.10}$$

Let us use an example to see the workflow of the HWT in detecting and counting bees. We start by analyzing a small 16x16 portion cropped out from a landing pad image with a single bee as seen in Figure 3.19.

Figure 3.19: Bee Image

Next, in order to separate the bee pixels from the non-bee pixels in Figure 3.19, we apply HWT. We would refer a pixel in the image to be a bee pixel if it is covered by the up-down or down-up spike in each row as seen in Figure 3.20. The image on the left has only 1 up-down triangle spike detected and image on the right has 1 down-up triangle spike detected on row 8 after application of a single scale 1D HWT.

Figure 3.20: Up-down triangle spike in row 8 (left); down-up triangle spike in row 8 (right); up-segments are red; down-segments are blue

We refer back to Equation (3.5) to represent the spikes in Figure 3.20. Towards that end, the up-down spike S_{ud} in the left image is represented as follows in Equation (3.11).

$$S_{ud} = (3, 4, \pi/4, -1, -1, 0, 5, 5, \pi/3) \tag{3.11}$$

In the above equation, the starting and ending points of the curve are the first two elements in S_{ud}, i.e. $u_s = 3$ and ends at $u_e = 4$. The third element is the angle of the climb for the spike, $\alpha \approx \pi/4$. The fourth, fifth, and sixth elements represent the flat part of the spike. But since the up-down spike is a triangle spike there will be no flat segment, thus $f_s = f_e = -1$ and $\gamma = 0$. Finally the seventh, eighth, and ninth elements of S_{ud} represent the segment where the spike descends. Thus the starting and end points of the decline segment are the seventh and eighth element respectively with the angle of descend being the ninth element of S_{ud}. Hence $d_s = 5$ and $d_e = 5$ and the angle $\beta \approx \pi/3$.

Similarly, the the down-up spike S_{du} in the image on the right in Figure 3.20 is represented as follows in Equation (3.12).

$$S_{du} = (3, 4, \pi/4, -1, -1, 0, 1, 2, \pi/3) \tag{3.12}$$

S_{du} is a down-up spike, thus the starting point of the climb will be after the ending point of the decline segment. In the above equation, the starting and ending points of the curve are the first two elements in S_{du}, i.e. $u_s = 3$ and ends at $u_e = 4$. The third element is the angle of the climb for the spike, $\alpha \approx \pi/4$. The fourth, fifth, and sixth elements represent the flat part of the spike. But since the up-down spike is a triangle spike there will be no flat segment, thus $f_s = f_e = -1$ and $\gamma = 0$. Finally the seventh, eighth, and ninth elements

of S_{du} represent the segment where the spike descends. Thus the starting and end points of the decline segment are the seventh and eighth element respectively with the angle of descend being the ninth element of S_{du}. Hence $d_s = 1$ and $d_e = 2$ and the angle $\beta \approx \pi/3$.

Next, in order to find the set of pixel columns in row 8 covered by the up-down S_{ud} and down-up S_{du} spike we refer to Equations (3.6), (3.7), and (3.8). The result is shown in Equation (3.13) and (3.14)

$$Z^8_{U,1} = \{p(1,3,5)\} = \{i | 6 \leq i \leq 11\} \tag{3.13}$$

$$Z^8_{D,1} = \{p(1,1,4)\} = \{i | 2 \leq i \leq 9\} \tag{3.14}$$

The number of unique column pixels covered by the up-down and down-up spike in row 8 is given in Equation (3.15) following the formulation in Equation (3.9).

$$Z^8_1 = (Z^8_{U,1} - Z^8_{D,1}) \cup (Z^8_{D,1} - Z^8_{U,1}) \cup (Z^8_{D,1} \cap Z^8_{U,1}) \tag{3.15}$$

Thus the unique pixels that represent the bee can be found out by solving for Equation (3.15). $Z^8_1 = \{10,11\} \cup \{2,3,4,5\} \cup \{6,7,8,9\} = \{2,3,4,5,6,7,8,9,10,11\}$. Since the scale of the 1D HWT used in this example is $j = 1$, thus the total number of bee pixels in Figure 3.19 covered by the spikes can be calculated following the formulation in Equation (3.10). The result is shown in Equation (3.16).

$$X_1(I) = \sum_{r=0}^{15} |Z^r_1| = 44 \tag{3.16}$$

A normalizer N is then used to count the actual number of bees present in the entire image. This normalizer is the average number of pixels occupied by individual bees. The value of N is found experimentally and it depends upon the distance of the camera from the landing pad. The combined process of counting bees on the landing pad is presented in Algorithm 3.1. The algorithm takes as input an image I, normalizer N, scale of 1D

HWT j, thresholds for the angles of climb α, descend β and flat area γ of the spike. In our experiments $\alpha = \beta = 60°$ and $\gamma = 5°$.

Algorithm 3.1 Localize Landing Pad And Count Bees

Input:

Image (I),
Normalization Factor (N),
Scale of 1D HWT (j),
Climb Angle (α),
Descend Angle (β),
Angle of Flat Segment (γ)

Output:

Count of Bees on the Landing Pad (**count**)

Begin

L = Localize landing pad using method discussed in Section 3.2
GL = Perform Gaussian Blur on L with a 7x7 kernel
PL = Apply Pyramid Mean Shift Filter on GL
RL = Apply Max RGB Filtering on PL
BL = Blue pixels in RL are set to white. This is done to eliminate the effect of shadows being turned blue after applying max RGB filtering in the previous step
$GrayL$ = Convert BL to grayscale
$X_j(GrayL)$ = Apply 1D HWT with scale j to $GrayL$ and then count all pixels covered by up-down and down-up spikes in all rows
$count = \lfloor X_j(GrayL)/N \rfloor$

End

The output from various steps of the Algorithm 3.1 is shown below.

Figure 3.21: Landing Pad L localized from Image I using methods discussed in Section 3.2

Figure 3.22: Image PL after applying Gaussian Blur on Figure 3.21 and then shifting the mean by applying pyramid mean shift filter

Figure 3.23: Image RL after applying max RGB filtering on Figure 3.22

Figure 3.24: Image BL after bleaching the pixels in Figure 3.23

Figure 3.25: Final grayscaled image $GrayL$

3.4 Evaluation on Sample Images

We compared the accuracy of the algorithm by selecting 793 images with bees on the landing pad. We tried to select images which had some amount of skewness. Three human observers went through our selected set and counted the number of bees present on the landing pad in each image. The above counts served as a ground truth for our evaluation. Towards that end, we also evaluated another bee counting algorithm proposed in [18] on the same sample of images. We will refer to this algorithm as Algorithm Omni. Table 3.1 summarizes the bee count results from the two algorithms.

Algo/Error	3	5	7	10	15
Algorithm 3.1	39.97	53.72	63.56	73.77	82.10
Algorithm Omni [18]	63.18	73.77	80.10	84.99	90.29

Table 3.1: Evaluation of two bee counting algorithms: column names are error margin values; other values are percentages

In Table 3.1, the columns represent the allowable error margin between the ground truth bee counts and the bee counts from the two algorithms. For example, let us consider the case when the value of *Error Threshold* is 3. The value in the second column corresponding to Algorithm 3.1, which is 39.97, suggests that the bee counts while using Algorithm 1 where within an absolute margin of 3 with the ground truth data for 39.97% of the sample images. Similarly the values in the last column which are 82.10 and 90.29 respectively,

suggests that when the *Error Threshold* is 15, the number of times the bee count results from Algorithm 3.1 were within an absolute difference of 15 from the ground truth data was for 82.10% of the sample images, whereas bee count results from Algorithm 2 were within the threshold for 90.29% of the sample images. The under performance of the Algorithm 3.1 can be attributed to the fact that in certain images the landing pad Localization was not perfect and there were grassy regions left above or around the landing pad. That resulted in 1D HWT generating spikes on the grassy regions and that resulted in false positives. Algorithm 2 on the other hand adds an additional step to remove any excess grassy areas above the landing pad. Figure 3.26 shows an example of one such image where the landing pad localization did not work perfectly and there were grassy regions left. False positives also occurred when shadows, blades of grass or leaves blown by wind were counted as bees. In cases where the landing pad was localized correctly, both the algorithms gave similar bee counts.

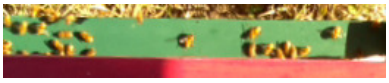


Figure 3.26: Inaccurately localized landing pad

3.5 Discussion

Static images of landing pads, may not always be a sufficient tool to study the foraging activity of a hive. In order to understand the behavior, the images need to be taken at a very small interval. Analyzing flight patterns of foraging traffic would help us understand how many bees move in and out of the hives at a given time. Improvement in the hardware of raspberry pis allows us to take videos through the raspberry pi camera. Hence in the following chapter we will analyze the videos of forager traffic and work towards understanding how foraging traffic activity can provide information regarding the health of a hive.

CHAPTER 4

TRACKING DIRECTIONAL BEE MOTIONS

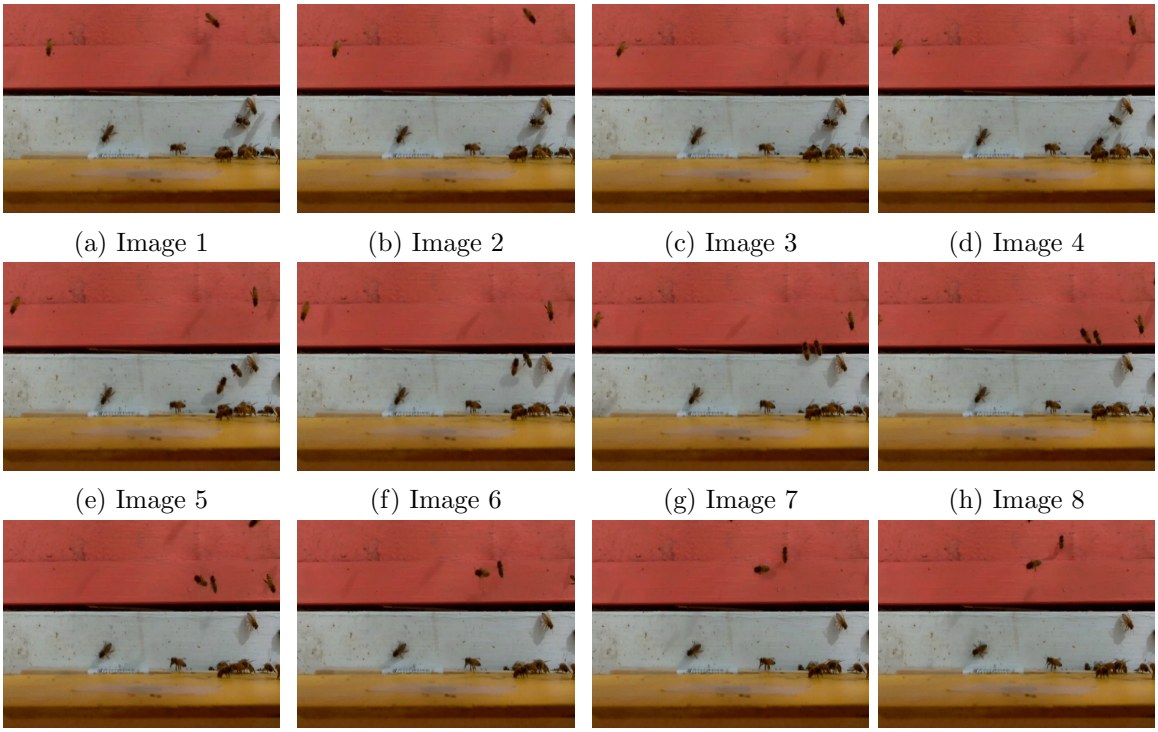
4.1 Goal

In this chapter we will be discussing a method to detect bee motions from extracted frames of a video. The goal of the method proposed in this chapter, is to analyze each frame and then find the coordinates of the positions in the frame where bee movement are detected. Next we take those positions and project them on a separate frame (size is the same as the original frame) with a white background. This helps us to take a frame with bees and convert them into a frame with white background having only the positions of interest marked, i.e. where there are bee movements. To follow along in this chapter, lets take for example 12 consecutive frames from a video. Figure 4.1 shows the twelve frames.

Let us assume that the frame rate of the video is 7 frames per second. Figure 4.1a is the first frame and Figure 4.1l is the last frame of the video. For the above sample frames the end objective of our proposed method are the images with the white background shown in Figure 4.2. For example, Figure 4.2b is the result of applying our method to Figure 4.1b. In the resultant image we can see that only the bees which show movements have been marked. Similarly Figure 4.2j is the result of applying our method to Figure 4.1i. In the following sections in this chapter we will see the various methodologies used in designing our proposed approach.

4.2 Background Difference

Background difference method is suitable for motion detection in scenarios where the camera and the background is fixed. This model suits our BeePi system since our cameras are fixed. It's advantages include being simple and having more accurate description of the target positioning. It does have some shortcomings where the background information may

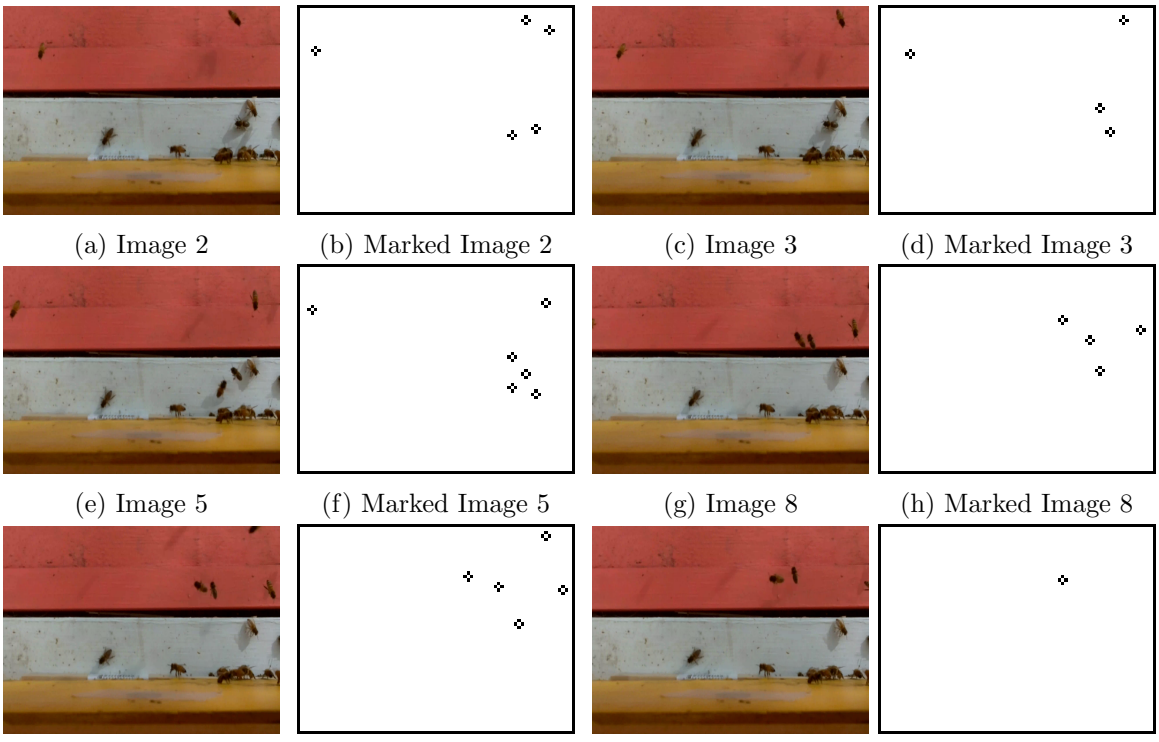


(a) Image 1 (b) Image 2 (c) Image 3 (d) Image 4

(e) Image 5 (f) Image 6 (g) Image 7 (h) Image 8

(i) Image 9 (j) Image 10 (k) Image 11 (l) Image 12

Figure 4.1: 12 Sampled Consecutive Frames from a Video



(a) Image 2 (b) Marked Image 2 (c) Image 3 (d) Marked Image 3

(e) Image 5 (f) Marked Image 5 (g) Image 8 (h) Marked Image 8

(i) Image 9 (j) Marked Image 9 (k) Image 11 (l) Marked Image 11

Figure 4.2: Sample original frames and the corresponding images with bees movements marked on a white background

be influenced by variable illumination, shadows and noise.

4.3 Creation of a Dynamic Background Image

The first step of the algorithm is to create a dynamic background average image to compare against each frame of the video. The cameras in our BeePi system record video at 25 frames per second. We found during our experiments, that examining frames between half a second interval before and after the current frame in context gives us meaningful results in examining bee movements. Thus in order to create a dynamic background image for each frame, we calculate a running average of the 12 preceding and 12 successive frames (i.e. 0.5 seconds of video before and after the current frame). Algorithm 4.1 describes the method to find the background image.

Algorithm 4.1 Compute Initial Dynamic Background Image (*ComputeBackground*)

Input:

Total Number of Frames in The Video (n),
Index of Current Frame (pos),
Average Window Size ($avgWinSize$),
Sum of All Images in the Current Window ($avgSum$),
Number of Images Before Current Frame ($Images_{beforeCurrent}$),
Number of Images After Current Frame ($Images_{afterCurrent}$)

Output:

Average Background Image (Avg)

Begin

$ImagesCount_{inBackground} = avgWinSize$ - [max(0, $Images_{beforeCurrent}$ - pos)
+ max(0, $Images_{afterCurrent}$ - (n - 1 - pos))]

$Avg = avgSum$ / $ImagesCount_{inBackground}$

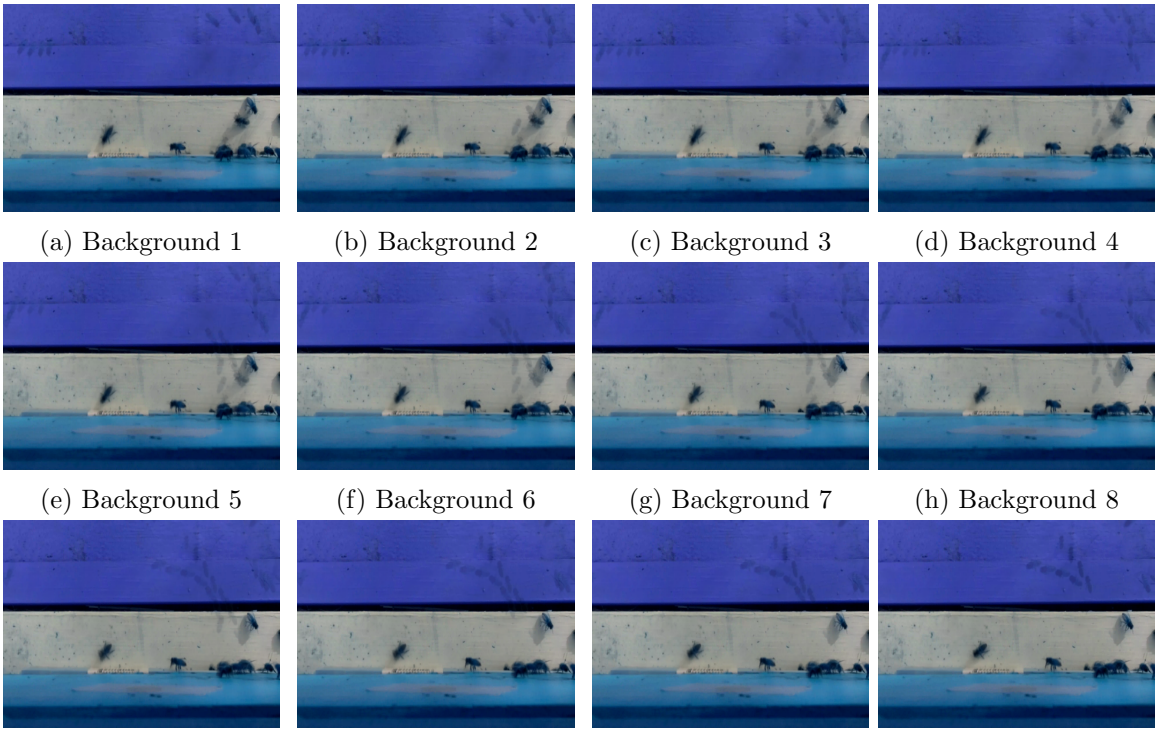
End

The longer an object is in a particular place on screen in the source video, the more distinct it will tend to be in the resulting average image. The average background image displays the portions of the image frame that remain approximately the same over the 1 sec of the video. The intuition behind generating an average background image comes from the fact that static bees on the landing pad and static background should not be considered

while investigating bee motions.

In our example with frames in Figure 4.1, let us assume the fps of the video is 7, which means there are 7 frames in 1 second of the video. Thus in order to create a average background image for a particular frame, we have to consider its 3 preceding and 3 successive frames. So we create a sliding window of 7 frames (3 preceding + current frame + 3 successive), and we call the middle frame as the frame in context. At the beginning when the first frame is in consideration, there are no preceding 3 frames. In that scenario, we assign 3 null values to the beginning of the sliding window array and so on. Below is the detailed list of frames that were averaged to find the background image for the frame in context in this current example. Background1 represents average background when Frame 1 (Figure 4.1a) is in context. All corresponding frames in context are marked in bold. *Empty* below refers to a single null value, whereas Image1 is an image matrix.

1. Background1 $\Longrightarrow$ *Empty*, *Empty*, *Empty*, **Image1**, Image2, Image3, Image4
2. Background2 $\Longrightarrow$ *Empty*, *Empty*, Image1, **Image2**, Image3, Image4, Image5
3. Background3 $\Longrightarrow$ *Empty*, Image1, Image2, **Image3**, Image4, Image5, Image6
4. Background4 $\Longrightarrow$ Image1, Image2, Image3, **Image4**, Image5, Image6, Image7
5. Background5 $\Longrightarrow$ Image2, Image3, Image4, **Image5**, Image6, Image7, Image8
6. Background6 $\Longrightarrow$ Image3, Image4, Image5, **Image6**, Image7, Image8, Image9
7. Background7 $\Longrightarrow$ Image4, Image5, Image6, **Image7**, Image8, Image9, Image10
8. Background8 $\Longrightarrow$ Image5, Image6, Image7, **Image8**, Image9, Image10, Image11
9. Background9 $\Longrightarrow$ Image6, Image7, Image8, **Image9**, Image10, Image11, Image12
10. Background10 $\Longrightarrow$ Image7, Image8, Image9, **Image10**, Image11, Image12, *Empty*
11. Background11 $\Longrightarrow$ Image8, Image9, Image10, **Image11**, Image12, *Empty*, *Empty*
12. Background12 $\Longrightarrow$ Image9, Image10, Image11, **Image12**, *Empty*, *Empty*, *Empty*



(a) Background 1 (b) Background 2 (c) Background 3 (d) Background 4

(e) Background 5 (f) Background 6 (g) Background 7 (h) Background 8

(i) Background 9 (j) Background 10 (k) Background 11 (l) Background 12

Figure 4.3: Background Images

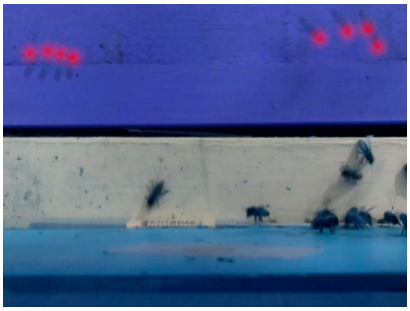
We can see Background1 has 7 values in its list, with 3 of them being *Empty* and 4 of them being image frames. Since Image1 is the middle frame, we call Image1 as the frame in context. When we calculate the first average image Background1, we take into consideration only 4 valid images as the count. In this case, input variable *avgSum* in Algorithm 4.1 refers to the pixel wise sum of Image1, Image2, Image3 and Image4. We can refer to the calculation of $ImagesCount_{inBackground}$ in Algorithm 4.1. In the case of Background1 ($pos = 0$), $ImagesCount_{inBackground}$ = 7 - (max(0, 3-0) + max(0, 3-(12-1-0))) = 4. Similarly in the case of Background11 ($pos = 10$), $ImagesCount_{inBackground}$ = 7 - (max(0, 3-10) + max(0, 3-(12-1-10))) = 5. Figure 4.3 shows the corresponding background images.

(a) **Image 1**

(b) Image 2

(c) Image 3

(d) Image 4



(e) Background 1

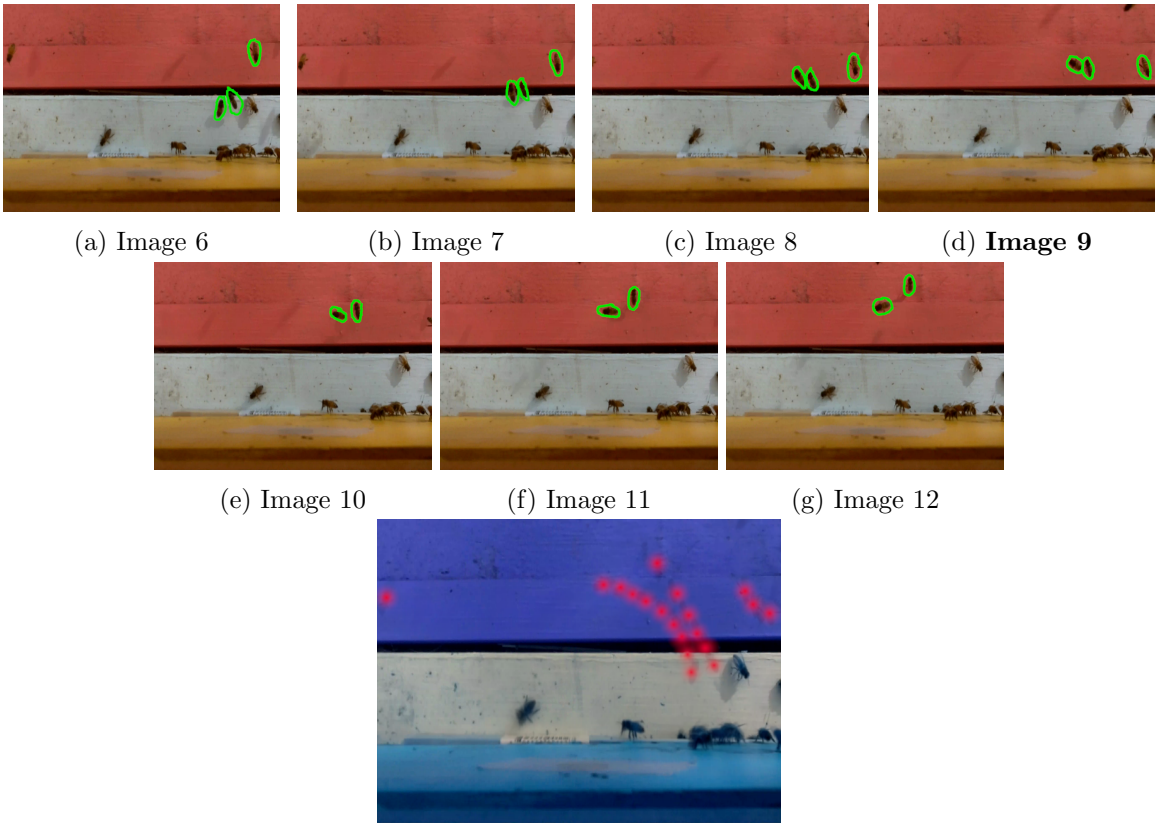
Figure 4.4: 4 Frames Used to Generate Background1. Red dots in Figure 4.4e are marked only for presentation or visual reference of the reader. Image 1 is the frame in context i.e. the middle frame of the 1 second window

4.3.1 Visual Description of Background Generation

In this part we will compare the background images in Figure 4.3 with the corresponding frames in context. Specifically we will try to find out visually if our approach generated the correct average background image. We will start with Background1 (Figure 4.3a), which was generated using Frames1 (Figure 4.1a), Frames2 (Figure 4.1b), Frames3 (Figure 4.1c) and Frames4 (Figure 4.1d).

In Figures 4.4a, 4.4b, 4.4c and 4.4d we see that the bees which have moved in the 4 consecutive frames have been marked. Since the background average image is generated using the above 4 frames, we see that the Figure 4.4e was correctly able to pick up the bee movements which have been marked by red dots. The other static bees in the frames have remained in the same position in the background image too.

Another example, in Figures 4.5a, 4.5b, 4.5c, 4.5d, 4.5e, 4.5f and 4.5g we see that the bees which have moved in the 7 consecutive frames have been marked. Since the background average image is generated using the above 7 frames, we see that Figure 4.5h was correctly able to pick up the bee movements which have been marked by red dots for our reference.



(a) Image 6 (b) Image 7 (c) Image 8 (d) **Image 9**

(e) Image 10 (f) Image 11 (g) Image 12

(h) Background 9

Figure 4.5: 7 Frames Used to Generate Background9. Red dots and green markings in Figure 4.5h are used only for presentation or visual reference of the reader. Image 9 is the current frame in context i.e. the middle frame of the 1 second window

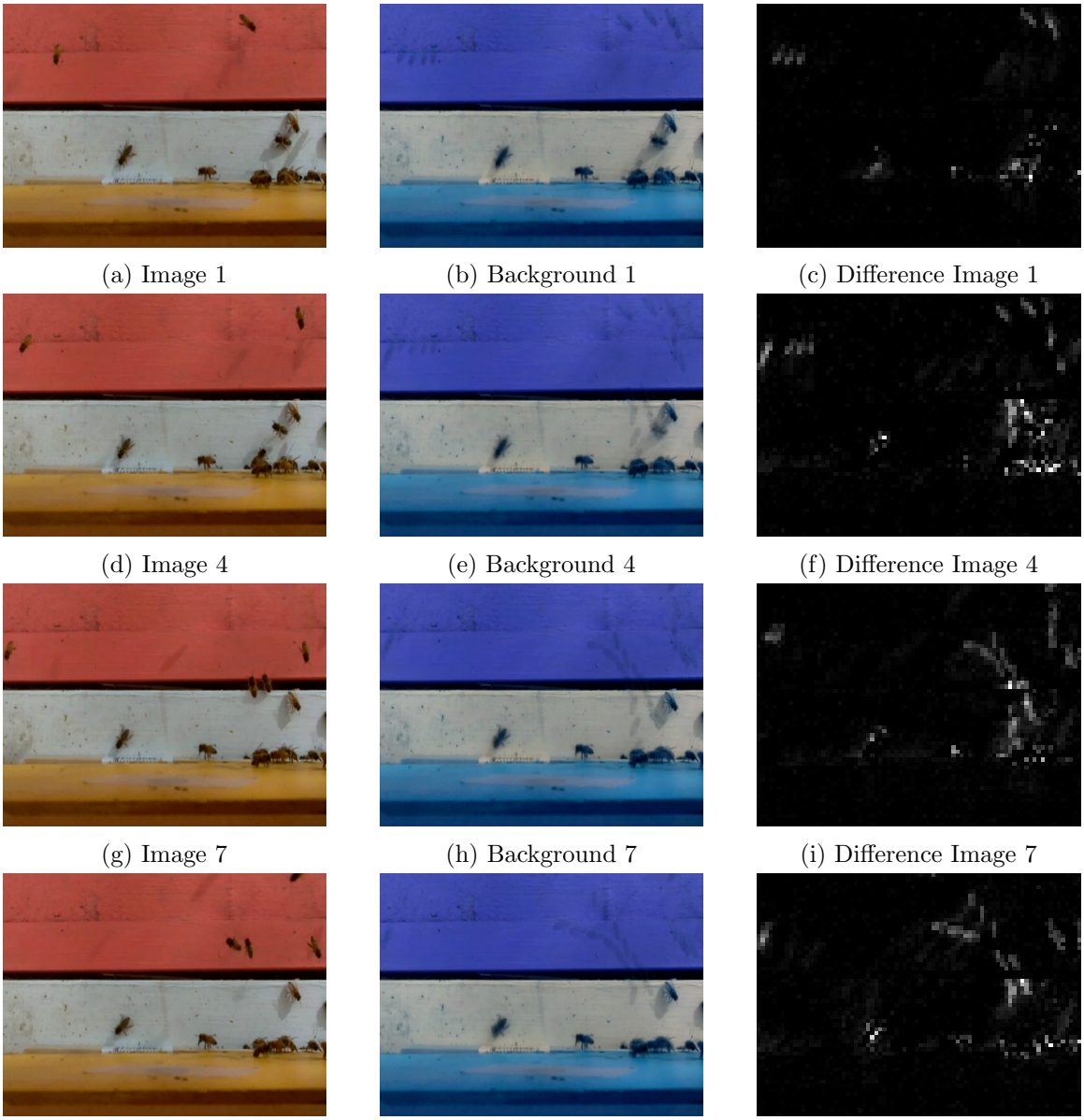
The other static bees in the frames have remained in position in the background image too. From the two examples we can see that if we subtract the background image from the frame in context, we would be able to find the positions in the frame where there are possibilities of bee movements being detected.

4.4 Subtracting Background Image From The Video Frame in Context

The following step involves subtracting the average background image from the video frame in context. By doing so, we will be able to find the positions in the frame that have moved or have shown some sort of motion over the 1 second of the video. The intuition behind this approach is that the resultant image after subtraction would give us the positions in the frame in context, which is the farthest away (in terms of distance) from the average background image. The positions in the frame which are closest to the background image are the portions of the frame which have remained static over that 1 second of the video (the frame in context, is the middle frame of the 1 second window). Thus the locations where the distance was maximum will help us to identify the bee movements. The camera in our BeePi system points vertically on the landing pad over which the bees enter or leave their respective hives. Thus we can safely say that any recorded movement would most likely correspond to a bee motion.

In our example with 12 Frames in Figure 4.1, we have seen in Section 4.3 that there are 12 average background images generated. In Figure 4.6, we see some of the resultant images after subtracting the average background images. The first column is the current frame in context, i.e. the middle frame of the 1 second period of the video, the second column represents the corresponding average background image. The images in the third column are generated by subtracting the background image from the frame in context in the first column.

We can see in Figure 4.7, that the images generated after subtracting the background image from the current does not only contain the bees which have moved in the current frame, it also contains traces of bees which have moved or shown movement over the 1 second of frames (7 frames) used to generate the background image. Failure to remove



(a) Image 1 (b) Background 1 (c) Difference Image 1

(d) Image 4 (e) Background 4 (f) Difference Image 4

(g) Image 7 (h) Background 7 (i) Difference Image 7

(j) Image 9 (k) Background 9 (l) Difference Image 9

Figure 4.6: Frames with corresponding average background to be compared against and the image generated after subtraction. The images in the third column are generated by subtracting the image in the second column from the image in the first column. We are trying to find out the portion in the current frame in context that have moved over a 1 second of the video. The current frame occurs in the middle of the examining period of 1 second of the video.

(a) Difference Image 1 (S1) (b) Difference Image 4 (S4) (c) Difference Image 9 (S9)

Figure 4.7: Difference Image

those additional spots from the resultant images would create problems of duplicate counts. So in the following steps we will focus on trying to remove those additional bee traces from those images. The end goal would be to find only those spots in the difference image which represent bees that have moved in the frame in context.

4.5 Smoothing Out Difference Image

This step involves smoothing the resultant image generated by subtracting the current frame from its corresponding average background image. It helps to turn detected movement regions into local maxima in the difference image. Smoothing helps in the better measurement of a variable which is varying slowly and is corrupted by random noise. In our case, the difference image may be subjected to occasional spikes or random noise as we see in Figure 4.7. Such random noises may be generated by shadows of flying bees or blurry images due to camera movement during wind gusts etc. Thus it can sometimes be useful to replace each data point by some kind of local average of surrounding data points. Since nearby points measure very nearly the same underlying value, averaging can reduce the level of noise without (much) biasing the value obtained. To that we can say that smoothing reduces noise, giving us (perhaps) a more accurate intensity surface.

We smooth the detected movement regions by applying a mask convolution over the difference image. For every detected maxima point in the difference image, we take into consideration its 4 neighboring points. For example if the maxima point is at (i,j), then the 4 neighboring pixels for consideration would be (i-1, j), (i+1, j), (i, j-1), (i, j+1) re-

spectively. We decided to smooth the image using 4 neighboring points for simplicity and faster processing. Next we blur each resultant image frame by averaging each pixel by its 4 nearest pixels. Doing so each bee becomes a local maxima in the difference image. The approach is explained in Algorithm 4.2. The central pixel is weighed more than the others, and the others are weighted unequally for different degrees of smoothing. This would help in eliminating the false detections which may occur due to random noises.

Algorithm 4.2 Smooth Image ($SmoothImage$)

Input:

Difference Image ($image$). Shape is (r,c),
Smoothing Steps (n)
Smoothing Factor ($factor$)

Output:

Smoothed Image ($image$)

Begin

r = Number of rows in $image$
c = Number of columns in $image$
Initialize a zero or empty matrix eIm with same dimension of $image$
Populate matrix $eImT$ with values from rows 1 to $(r-1)$ of eIm. Shape is (r-1,c)
Populate matrix $eImB$ with values from rows 0 to $(r-2)$ of eIm. Shape is (r-1,c)
Populate matrix $eImR$ with values from columns 1 to $(c-1)$ of eIm. Shape is (r,c-1)
Populate matrix $eImL$ with values from columns 0 to $(c-2)$ of eIm. Shape is (r,c-1)

Populate matrix imT with values from rows 1 to $(r-1)$ of $image$. Shape is (r-1,c)
Populate matrix imB with values from rows 0 to $(r-2)$ of $image$. Shape is (r-1,c)
Populate matrix imR with values from columns 1 to $(c-1)$ of $image$. Shape is (r,c-1)
Populate matrix imL with values from columns 0 to $(c-2)$ of $image$. Shape is (r,c-1)

while $i < n$
Begin
Step 1: multiply every value in $image$ by 4 and use those values to update eIm
Step 2: add corresponding entries in $eImT$ with imB to update $eImT$ inplace
Step 3: add corresponding entries in $eImB$ with imT to update $eImB$ inplace
Step 4: add corresponding entries in $eImR$ with imL to update $eImR$ inplace
Step 5: add corresponding entries in $eImL$ with imR to update $eImL$ inplace
Step 6: multiply every value in eIm by $\frac{1}{factor}$ and use those values to update $image$
$i = i + 1$
End

End

In Figure 4.8, we can see an example of how smoothing helps in removal of occasional spikes or random noises. We started with the matrix : $\begin{pmatrix} 5 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 1 & 7 \end{pmatrix}$, which had random noises in the first and third rows. After 1 iteration (refer to Figure 4.8k), we see that the *image* matrix still has some spikes. But as we increase the number of smoothing steps, we see that spikes in the *image* matrix are reduced (Figure 4.8l, Figure 4.8m). Thus after 4 rounds of smoothing (refer to Figure 4.8n), the *image* matrix is transformed to a more uniform spread of values: $\begin{pmatrix} 0.870 & 0.996 & 0.738 \\ 0.972 & 1.343 & 1.223 \\ 0.667 & 1.136 & 1.161 \end{pmatrix}$, with the center pixel being weighted more than the others in the final smoothed matrix.

The Algorithm 4.2 can be expressed as a formula as shown in Equation 4.1. Each pixel (i, j) in *image* is replaced in place by Equation 4.1. In cases of boundary pixels when $image(i-1, j)$, $image(i+1, j)$, $image(i, j-1)$ and $image(i, j+1)$ does not exist, then they are replaced by 0.

$$\begin{aligned} image(i,j) = &[image(i-1,j) + \\ &image(i,j-1) + 4 * image(i,j) + image(i,j+1) + \\ &image(i+1,j)]/factor \end{aligned} \tag{4.1}$$

Let us take a look at our example in Figure 4.8 and evaluate the smoothed image matrix in Figure 4.8k using the Equation 4.1. Let the smoothing factor $factor$ be 8. The middle pixel $(1, 1)$ is evaluated as follows:

$$image(1,1) = [image(0,1) + image(1,0) + 4 * image(1,1) + image(1,2) + image(2,1)]/8$$

$$image(1,1) = [1 + 1 + 4 * 1 + 2 + 1]/8 = 1.125$$

Similarly for the border pixel $(0, 0)$, $image(0, 0)$ is evaluated as follows:

$$image = \begin{bmatrix} 5 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 1 & 7 \end{bmatrix}$$

a. **Sample *image* matrix**

$$eIm = \begin{bmatrix} 20 & 4 & 4 \\ 4 & 4 & 8 \\ 4 & 4 & 28 \end{bmatrix}$$

b. **Step 1**

$$eImT = \begin{bmatrix} 9 & 5 & 9 \\ 5 & 5 & 30 \end{bmatrix}$$

c. **Step 2**

$$eIm = \begin{bmatrix} 20 & 4 & 4 \\ 9 & 5 & 9 \\ 5 & 5 & 30 \end{bmatrix}$$

d. **Updated *eIm* matrix**

$$eImB = \begin{bmatrix} 21 & 5 & 6 \\ 10 & 6 & 16 \end{bmatrix}$$

e. **Step 3**

$$eIm = \begin{bmatrix} 21 & 5 & 6 \\ 10 & 6 & 16 \\ 5 & 5 & 30 \end{bmatrix}$$

f. **Updated *eIm* matrix**

$$eImR = \begin{bmatrix} 10 & 7 \\ 7 & 17 \\ 6 & 31 \end{bmatrix}$$

g. **Step 4**

$$eIm = \begin{bmatrix} 21 & 10 & 7 \\ 10 & 7 & 17 \\ 5 & 6 & 31 \end{bmatrix}$$

h. **Updated *eIm* matrix**

$$eImL = \begin{bmatrix} 22 & 11 \\ 11 & 9 \\ 6 & 13 \end{bmatrix}$$

i. **Step 5**

$$eIm = \begin{bmatrix} 22 & 11 & 7 \\ 11 & 9 & 17 \\ 6 & 13 & 31 \end{bmatrix}$$

j. **Updated *eIm* matrix**

$$image = \begin{bmatrix} 2.750 & 1.375 & 0.875 \\ 1.375 & 1.125 & 2.125 \\ 0.750 & 1.624 & 3.875 \end{bmatrix}$$

k. **Step 6: *image* after 1 iteration**

$$image = \begin{bmatrix} 1.718 & 1.281 & 0.875 \\ 1.265 & 1.375 & 1.796 \\ 0.750 & 1.531 & 2.406 \end{bmatrix}$$

l. **Smoothed *image* after 2 iterations**

$$image = \begin{bmatrix} 1.177 & 1.136 & 0.822 \\ 1.113 & 1.421 & 1.480 \\ 0.724 & 1.332 & 1.619 \end{bmatrix}$$

m. **Smoothed *image* after 3 iterations**

$$image = \begin{bmatrix} 0.870 & 0.996 & 0.738 \\ 0.972 & 1.343 & 1.223 \\ 0.667 & 1.136 & 1.161 \end{bmatrix}$$

n. **Smoothed *image* after 4 iterations**

Figure 4.8: An example to demonstrate the step by step procedure of Algorithm 4.2. Each step in the caption represents the result from the corresponding step of the algorithm.

$$image(0,0) = [image(-1,0) + image(0,-1) + 4 * image(0,0) + image(0,1) + image(1,0)]/8$$

$$image(0,0) = [0 + 0 + 4 * 5 + 1 + 1]/8 = 2.75$$

In the second step of smoothing in Figure 4.8l, Pixel $image(2,2)$ is evaluated as:

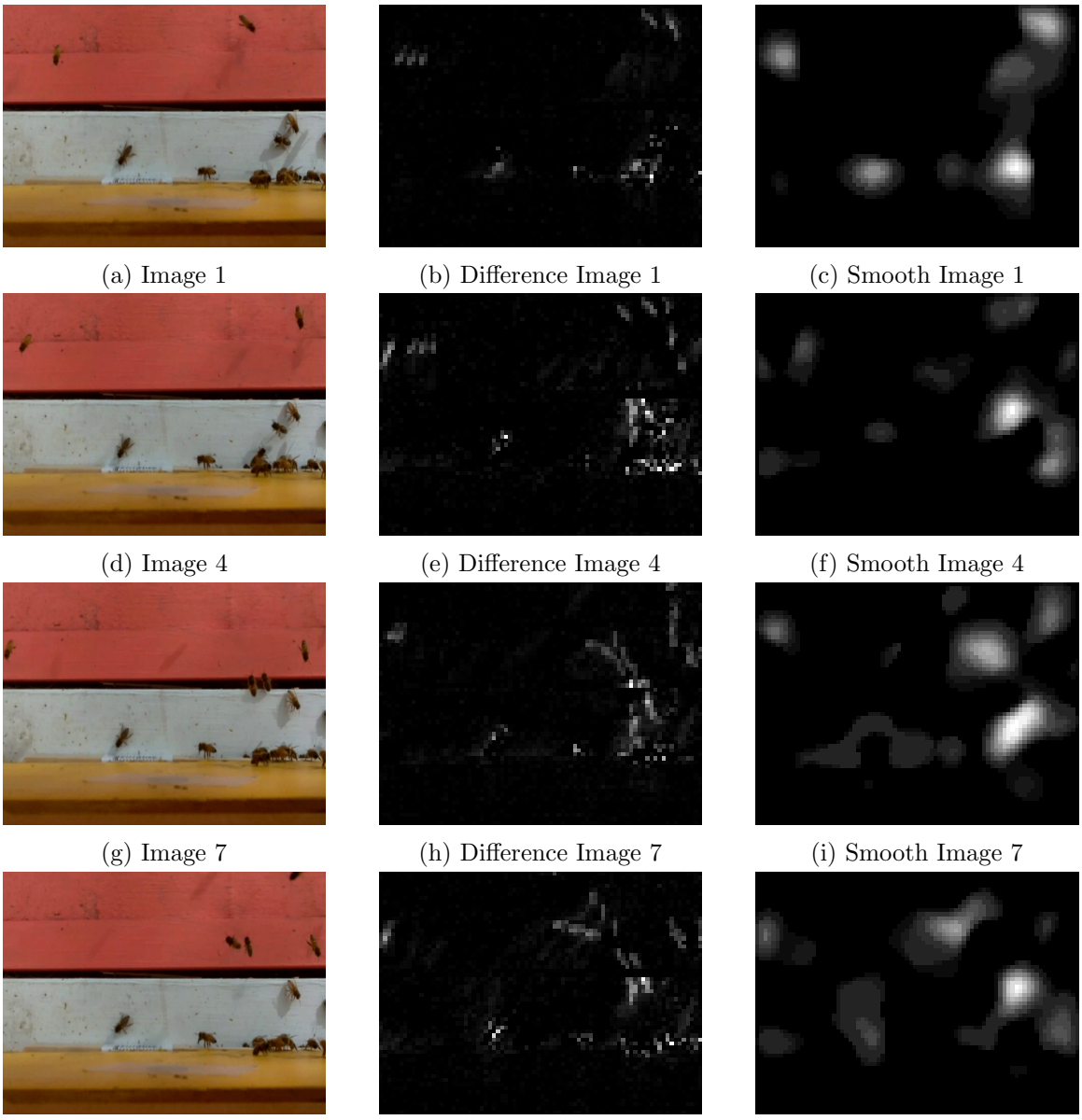
$$image(2,2) = [image(1,2) + image(2,1) + 4 * image(2,2) + image(2,3) + image(3,2)]/8$$

$$image(2,2) = [2.125 + 1.624 + 4 * 3.875 + 0 + 0]/8 = 2.406$$

From the above examples we can see that in our smoothing algorithm the central pixel is always weighed more than the neighboring pixels. The result of the smoothing step is shown in Figure 4.9. A visual comparison of the images in the second and third column in Figure 4.9 shows how smoothing has helped to turn multiple maxima values in a region to local maxima points. Along with that, if we observe closely the images in the first and third column, we will be able to see that the positions of those local maxima are closely associated to the positions of corresponding bee movements in the images in the first column. The higher the above association the more distinct is the corresponding local maxima region, thus resulting in brighter spots in the smoothed image.

4.6 Finding Maxima Points In The Smoothed Image

The result of the smoothing step in the above subsection is a frame with positions/points which are local maximas. But we saw in Figure 4.7 and Figure 4.9, not all maxima points that are detected correspond to bee movements. If we refer to the smoothed images in Figure 4.9, specifically we would like to remove those points which correspond to the lighter spots on the smoothed images. Therefore, we need to consider a threshold below which all the points will be removed. We can define this threshold to be the minimum acceptable



(a) Image 1 (b) Difference Image 1 (c) Smooth Image 1

(d) Image 4 (e) Difference Image 4 (f) Smooth Image 4

(g) Image 7 (h) Difference Image 7 (i) Smooth Image 7

(j) Image 9 (k) Difference Image 9 (l) Smooth Image 9

Figure 4.9: The images in the first column are the original frames. The second column is the result of subtracting the average background image from the corresponding frame in the first column. The images in the third column are generated as a result of smoothing the difference image in the second column.

color variation between the frame in context and the corresponding background. The above threshold will not be a one size fits all kind of value. In our experiments we have seen, the threshold value is different depending upon a video and the background in where the bee hive is located. The procedure to select a threshold value based upon a video is explained in Section 4.10. The method to find the coordinates of those maxima points is explained in Algorithm 4.3. While there might be several maxima points above the threshold, but not all represent the bees. Also the algorithm takes into consideration that a single bee can generate multiple maxima points close to each other (head, body, wings etc). Thus the above algorithm takes all those disjointed clusters of maxima points and works towards reducing them to a single local maxima point.

In the example in Figure 4.10, we see that the initial coordinates of the maxima points are (0,0), (0,2), (1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), (3,3). At each step, the algorithm compares an element against its 4 neighboring elements. For example in 'Step2' of the algorithm, each element of row 'i' in the matrix is compared against the elements in row 'i+1'. Similarly in 'Step5' of the algorithm, each element of column 'i' in the matrix is compared against the elements in column 'i-1'. Boolean matrix '*max*' is populated by comparing the corresponding entry in *image* against the threshold. Next computing the element-wise truth value at each step of the algorithm, gives us the position of the elements which are greater than the threshold and also a local maxima among its neighboring elements. In the above example in Figure 4.10n, we see that the number of local maxima points in the final '*max*' matrix is lower than the number of original maxima points in the initial '*max*' matrix in Figure 4.10b. For (2,1), we see that the corresponding value 136 is higher than its surrounding elements. Another important point of observation is that, (3,3) has been selected to be a local maxima; but the value at (3,3) which is 125.50 is lower than the value at (2,2) which is 126.89. The reason behind the selection is apparent when we observe the surrounding elements of (3,3) and (2,2). The surrounding element of (2,2) has the element 136, which is higher than 126.89. Thus (2,2) was not selected as a local maxima point.

Algorithm 4.3 Find Coordinates of Maxima Points ($FindMaxima$)

Input:

Smoothed Image ($image$). Shape is (r,c),
Threshold ($threshold$)

Output:

Coordinates of Local Maxima Points (x, y)

Begin

r = Number of rows in $image$
c = Number of columns in $image$
Step 1: Initialize a boolean matrix max. Each entry in max is True,
when the corresponding entry in $image$ is greater than $threshold$ else it is False

Populate matrix $maxT$ with values from rows 1 to $(r-1)$ of max. Shape is (r-1,c)
Populate matrix $maxB$ with values from rows 0 to $(r-2)$ of max. Shape is (r-1,c)
Populate matrix $maxR$ with values from columns 1 to $(c-1)$ of max. Shape is (r,c-1)
Populate matrix $maxL$ with values from columns 0 to $(c-2)$ of max. Shape is (r,c-1)

Populate matrix $imageT$ with values from rows 1 to $(r-1)$ of $image$. Shape is (r-1,c)
Populate matrix $imageB$ with values from rows 0 to $(r-2)$ of $image$. Shape is (r-1,c)
Populate matrix $imageR$ with values from columns 1 to $(c-1)$ of $image$. Shape is (r,c-1)
Populate matrix $imageL$ with values from columns 0 to $(c-2)$ of $image$. Shape is (r,c-1)

Step 2: Initialize a temporary boolean matrix $temp1$. Entry in $temp1$ is True,
if corresponding entry in $imageB > imageT$, else False.
Compute the truth value of $temp1$ and $maxB$ element-wise and update $maxB$ inplace.

Step 3: Initialize a temporary boolean matrix $temp2$. Entry in $temp2$ is True,
if corresponding entry in $imageT > imageB$, else False.
Compute the truth value of $temp2$ and $maxT$ element-wise and update $maxT$ inplace.

Step 4: Initialize a temporary boolean matrix $temp3$. Entry in $temp3$ is True,
if corresponding entry in $imageL > imageR$, else False.
Compute the truth value of $temp3$ and $maxL$ element-wise and update $maxL$ inplace.

Step 5: Initialize a temporary boolean matrix $temp4$. Entry in $temp4$ is True,
if corresponding entry in $imageR > imageL$, else False.
Compute the truth value of $temp4$ and $maxR$ element-wise and update $maxR$ inplace.

Step 6: (x, y) are positions where the corresponding entry in max is True

End

$$\begin{bmatrix} 134.00 & 71.75 & 143 & 64.34 \\ 93.64 & 102.96 & 107.66 & 100.70 \\ 99.10 & 136 & 126.89 & 124.69 \\ 83.48 & 103.32 & 120.09 & 125.50 \end{bmatrix}$$

a. **Sample *image* matrix**

$$max = \begin{bmatrix} \mathbf{T} & F & \mathbf{T} & F \\ F & \mathbf{T} & \mathbf{T} & \mathbf{T} \\ F & \mathbf{T} & \mathbf{T} & \mathbf{T} \\ F & \mathbf{T} & \mathbf{T} & \mathbf{T} \end{bmatrix}$$

b. **Step 1 (threshold = 100)**

$$\begin{bmatrix} T & F & T & F \\ F & F & F & F \\ T & T & T & F \end{bmatrix}$$

c. $temp1 = imageB > imageT$

$$maxB = \begin{bmatrix} T & F & T & F \\ F & F & F & F \\ F & T & T & F \end{bmatrix}$$

d. **After Step 2**

$$max = \begin{bmatrix} T & F & T & F \\ F & F & F & F \\ F & T & T & F \\ F & T & T & T \end{bmatrix}$$

e. **Updated** max

$$\begin{bmatrix} F & T & F & T \\ T & T & T & T \\ F & F & F & T \end{bmatrix}$$

f. $temp2 = imageT > imageB$

$$maxT = \begin{bmatrix} F & F & F & F \\ F & T & T & F \\ F & F & F & T \end{bmatrix}$$

g. **After Step 3**

$$max = \begin{bmatrix} T & F & T & F \\ F & F & F & F \\ F & T & T & F \\ F & F & F & T \end{bmatrix}$$

h. **Updated** max

$$\begin{bmatrix} T & F & T \\ F & F & T \\ F & T & T \\ F & F & F \end{bmatrix}$$

i. $temp3 = imageL > imageR$

$$maxL = \begin{bmatrix} T & F & T \\ F & F & F \\ F & T & T \\ F & F & F \end{bmatrix}$$

j. **After Step 4**

$$max = \begin{bmatrix} T & F & T & F \\ F & F & F & F \\ F & T & T & F \\ F & F & F & T \end{bmatrix}$$

k. **Updated** max

$$\begin{bmatrix} F & T & F \\ T & T & F \\ T & F & F \\ T & T & T \end{bmatrix}$$

l. $temp4 = imageR > imageL$

$$maxR = \begin{bmatrix} F & T & F \\ F & F & F \\ T & F & F \\ F & F & T \end{bmatrix}$$

m. **After Step 5**

$$max = \begin{bmatrix} \mathbf{T} & F & \mathbf{T} & F \\ F & F & F & F \\ F & \mathbf{T} & F & F \\ F & F & F & \mathbf{T} \end{bmatrix}$$

n. **Updated** max

Figure 4.10: An example to demonstrate the step by step procedure of Algorithm 4.3. Each step in the caption represents the result from the corresponding step of the algorithm. The final result of Step 6 would be the coordinates of the 'True' or 'T' positions in max, i.e. (0,0), (0,2), (2,1), (3,3)

$$max(i,j) = \begin{cases} True \text{ or } T, & \text{if } image(i,j) > image(i-1,j) \text{ AND} \\ & image(i,j) > image(i,j-1) \text{ AND} \\ & image(i,j) > image(i+1,j) \text{ AND} \\ & image(i,j) > image(i,j+1) \\ False \text{ or } F, & \text{if } image(i,j) < image(i-1,j) \text{ OR} \\ & image(i,j) < image(i,j-1) \text{ OR} \\ & image(i,j) < image(i+1,j) \text{ OR} \\ & image(i,j) < image(i,j+1) \end{cases} \tag{4.2}$$

Algorithm 4.3 can be formally represented using the Equation 4.2. Let us look at our example in Figure 4.10 and evaluate the final matrix max in Figure 4.10n using the Equation 4.2. For position $(i,j) = (0,0)$ in $image$, $image(0,0) = 134$ is greater than $image(i-1,j) = 0$ and $image(i,j-1) = 0$ and $image(i+1,j) = 93.64$ and $image(i,j+1) = 71.75$. Thus position (0,0) in the final max matrix is $True$. Similarly for position $(i,j) = (2,2)$ in $image$, $image(2,2) = 126.89$ is greater than $image(i-1,j) = 107.66$ and $image(i,j+1) = 124.69$ and $image(i+1,j) = 120.09$ but $image(2,2)$ is less than $image(i,j-1) = 136$. Thus position (2,2) in the final max matrix is $False$. Thus we see from the Algorithm 4.3 and Equation 4.2, that in the final max matrix, position (i,j) is labelled as $True$ only if the pixel value at position (i,j) is greater than the pixel value at it's 4 neighboring pixels $(i-1,j)$, $(i,j-1)$, $(i+1,j)$ and $(i,j+1)$. If the pixel value at position (i,j) is lesser than any of it's 4 neighbors, then the corresponding location in the final max matrix is labeled as $False$.

Hence, we are able to say that the Algorithm 4.3 has been able to reduce the number of maxima points detected by reducing a cluster of points to a single local maxima point by doing a local neighborhood comparison.

4.7 Removal of Very Close Maxima Points

The BeePi monitors record video every 15 minutes with a frame rate of 25 fps. To

facilitate a faster processing of those videos, we extract frames from them with a resolution of 80x60. By our manual inspection of those frames, we have come to the conclusion that the size of majority of the bees captured in those frame is between 8–12 pixels. Thus in order to reduce a single bee with multiple maxima points, to only one local maxima point, we need to extend our Algorithm 4.3 to further process its final result and remove very close points. The spacing distance is determined as explained in Section 4.10. Algorithm 4.4 describes the method to remove maxima points that are within the minimum spacing distance from the nearest position of another maxima value.

Algorithm 4.4 Remove Close Maxima Points (*RemoveClose*)

Input:

Coordinates of Maxima Points from Algorithm 4.3 (*points*),
Min Allowable Spacing Between Consecutive Maxima Points (*spacing*),
Smoothed Image (*image*)

Output:

Coordinates of Local Maxima Points (x, y)

Begin

imagePixels = pixel values in *image* corresponding to the *points*
indexPos = index positions of lowest to highest pixel values in *imagePixels*
n = length of *points*
Initialize a **set**, *discard* /* will hold the discarded coordinates from *points* */
while $(i < n)$ **and** $indexPos[i]$ **not in** *discard*
Begin
j = i+1
while $(j < n)$ **and** $indexPos[j]$ **not in** *discard*
Begin
$dX, dY = points[indexPos[j]] - points[indexPos[i]]$
$sumSqr = (dX * dX) + (dY * dY)$
$spacingSqr = (spacing * spacing)$
add $indexPos[j]$ to *discard* if $sumSqr < spacingSqr$
$j = j + 1$
End
$i = i + 1$
End
/* *discard* now contains the index positions of the *points* that will be removed */
$(x, y) = points[k]$, where $0 <= k < n$ and k not in *discard*

End

To understand the workings of Algorithm 4.4, let's continue from the results in the example of Figure 4.10. The maxima positions were at (0,0), (0,2), (2,1), (3,3).

$$\begin{array}{c} \\ positions \\ pixelvalues \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{pmatrix} (0,0) & (0,2) & (2,1) & (3,3) \\ 134 & 143 & 136 & 125.50 \end{pmatrix} \end{array}$$

Now let us set the spacing between two maxima points to be 4. The algorithm first creates a lookup array, *indexPos*, which holds the index positions of the sorted pixel values for the above coordinate points. So *indexPos* will hold the following values, [3,0,2,1]. This means the pixel value at (3,3) is the lowest and pixel value at (0,2) is the highest. We can verify this from Figure 4.10a, where pixel value at (3,3) is 125.50 and at (0,2) is 143. The algorithm works as follows; for each value in *indexPos*, we compare its coordinate with the coordinates of the points that succeeds the current index position. For example, when the index position is at 0, we will compare against the coordinates at positions 1,2 and 3. Again when the index position is at 2, we will only compare against the coordinate at position 3.

The comparison between two points is found using the squared distance formula as follows: $d = (x_1 - x_2)^2 + (y_1 - y_2)^2$. Next we check if d is less than the squared spacing, i.e. $d < 4 * 4$. If the condition holds then we add the point at the next index position to a *discard* **set**. Let us follow along with our example to make things clear. The first value in *indexPos* is 3. The corresponding coordinate for position 3 is (3,3). We will be comparing it to the next value in *indexPos* which is 0; and its corresponding coordinate is (0,0). The squared distance between (3,3) and (0,0) is $d = 18$. Next we will compare (3,3) against the next value in *indexPos* which is 2; and its corresponding coordinate is (2,1). The squared distance between (3,3) and (2,1) is $d = 5$. Now since $d < 16$, we will be moving $indexPos = 2$ to the *discard* set. Next we will compare (3,3) against the next value in *indexPos* which is 1; and its corresponding coordinate is (0,2). The squared distance between (3,3) and (0,2) is $d = 10$. Now since $d < 16$, we will be moving $indexPos = 1$ to the *discard* set. So the *discard* set holds {2,1} after the first iteration. In this way we loop through the length of the *indexPos*, skipping the indices which are already in the *discard*

$$\begin{bmatrix} 134.00 & 71.75 & 143 & 64.34 \\ 93.64 & 102.96 & 107.66 & 100.70 \\ 99.10 & 136 & 126.89 & 124.69 \\ 83.48 & 103.32 & 120.09 & 125.50 \end{bmatrix} \qquad max = \begin{bmatrix} \mathbf{T} & F & F & F \\ F & F & F & F \\ F & F & F & F \\ F & F & F & \mathbf{T} \end{bmatrix}$$

a. **Sample *image* matrix** b. **Final Result**

Figure 4.11: The final results of the example after removing maxima points close to each other. 'True' or 'T' positions, i.e. (0,0), (3,3) are the final maxima points

set. Thus at the end of the all the iterations the *discard* set holds {2,1}. So the final result will only have the coordinates at index positions 0 and 3 which are (0,0) and (3,3) respectively. Thus we have been able to remove the maxima points that were very close to each other. The final result for the example is given in Figure 4.11. Therefore combining Algorithm 4.3 and Algorithm 4.4, we are able to reduce multiple maxima points for a bee into a single point.

One important thing to note here is that, the final coordinate position does not necessarily mean the highest pixel value was selected. As in the example we saw the highest pixel value was 143 which was at location (0,2). But it was not selected as it was within the maxima spacing and close to (0,0). Thus it is very important to note that the positions of the detected maximas matter and not the corresponding pixel values in the *image*.

4.8 Combining All The Above Algorithms

In this section, we will be combining all the algorithms discussed in the above sections and see how a video is processed frame by frame and an individual bee in a frame is represented using a single point.

Algorithm 4.5 describes how we are able to combine our discussions in the previous sections into one entire algorithm. We start by creating an array of initial background image files $imFiles_{backgnd}$. At the beginning of the algorithm, when the first frame is in consideration, there is no preceding 12 frames. In that scenario, we assign 12 blank images to replicate the above. Next we start iterating through the frames of the video and computing the corresponding average background image at each step. Next we subtract the

Algorithm 4.5 Final Algorithm To Find The Maxima Points Across All Frames

Input:

Image Files ($imFiles$),
Number of Image Files (n),
Averaging Window Size ($avgWinSize$),
Threshold ($thresh$),
Smoothing Steps ($smooth$),
Smoothing Factor ($factor$),
Min Allowable Spacing Between Consecutive Maxima Points ($spacing$)

Output:

Array of Maxima Points For All Frames (**maximaArray**)

Begin

$Images_{afterCurrent} = \lceil avgWinSize/2 \rceil$
$Images_{beforeCurrent} = (avgWinSize$ -1) - $Images_{afterCurrent}$
$imFiles_{backgnd} = [Empty]$ * $Images_{beforeCurrent}$ + $imFiles[: Images_{afterCurrent}]$
$avgSum$ = Sum of all Images in $imFiles_{backgnd}$
while $(i < n)$
Begin
$avgImg =$ **ComputeBackground**$(n, i, avgWinSize, avgSum,$
$Images_{beforeCurrent}, Images_{afterCurrent})$
$subImage = Subtract(imFiles[i], avgImg)$
$smoothImage =$ **SmoothImage**$(subImage, smooth, factor)$
$absImageSqrd = Multiply(smoothImage, smoothImage)$
$sumImSqrd = Sum(absImageSqrd, axis = 2)$
$maximaPoints =$ **FindMaxima**$(sumImSqrd, threshold * threshold)$
$mxPtCloseRm =$ **RemoveClose**$(maximaPoints, spacing, sumImSqrd)$
Append $mxPtCloseRm$ To **maximaArray**
$imLater = i + Images_{afterCurrent} + 1$
if $imLater < n$
Begin
$nextImage = imFiles[imLater]$
$avgSum = avgSum + nextImage$
End
else
Begin
$nextImage = [Empty]$
End
if $imFiles_{backgnd}[0]$ **is not** $Empty$
Begin
$avgSum = avgSum$ - $imFiles_{backgnd}[0]$
End
$imFiles_{backgnd} = imFiles_{backgnd}[1 :] + nextImage$
$i = i + 1$
End

End

background image from the current frame and then we apply smoothing to the resultant image. At this point the image $smoothImage$ represents the color difference between the current frame and the background image in the R, G and B channels. We apply sum of squared technique to find the maximum absolute variation or difference between the current frame and the background image. For that purpose we multiply the image with itself and we have $absImageSqrd$. The image $absImageSqrd$ has 3 channels. So we convert the image to a single channel by creating a new image $sumImSqrd$, which is basically the sum of the three channels at each index. For example: $sumImSqrd[i,j] = absImageSqrd[:,:,0][i,j] + absImageSqrd[:,:,1][i,j] + absImageSqrd[:,:,2][i,j]$. Then we take $sumImSqrd$ and find out the coordinates of the points where the pixel value is greater than $threshold*threshold$. This gives us a set of points which are local maximas in their surroundings. But sometimes two local maxima points are very close to each other. This might be the case where the head and the tail of a single bee have both been detected as maxima points. To counter this scenario, we apply a rule for the points to have a minimum distance/spacing between them. In the next step we keep only those points, the distance between whom satisfies our spacing rule. And finally we add those points, $mxPtCloseRm$ to **maximaArray**.

After that we select the next image in $imFiles$ add it to $imFiles_{backgnd}$ and update average background sum, $avgSum$ with it. Along with that we remove the first image in $imFiles_{backgnd}$ and subtract the corresponding value from $avgSum$, if the first image to be removed is not an $Empty$ value . At the end when there is no image left in $imFiles$, then we add $[Empty]$ to $imFiles_{backgnd}$ and in that case, $avgSum$ is not updated.

The final result of the Algorithm 4.5 is **maximaArray**. It holds the coordinates of the corresponding maxima points for each frame in the video that we have processed.

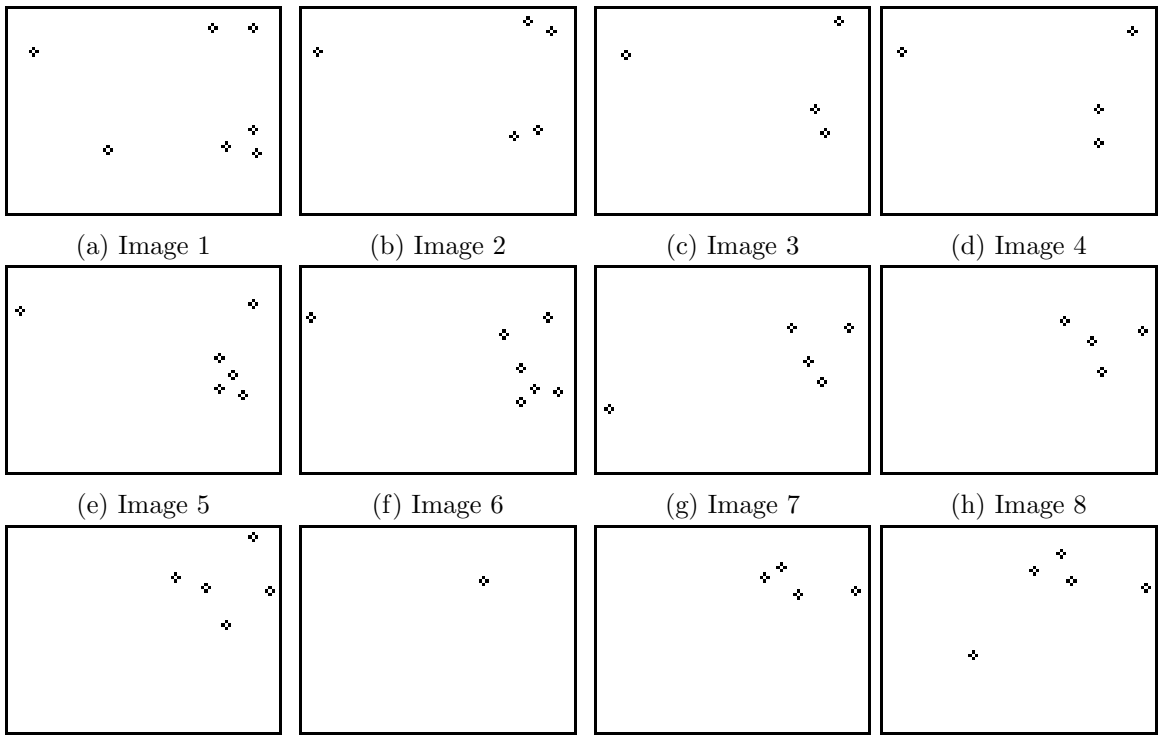
4.9 Putting The Maxima Points Against White Background

The **maximaArray** from Algorithm 4.5 holds the location of the bees detected in each frame by our algorithm. Thus we now know the positions in a frame which are important for us. So rather than looking at an entire frame, we need to focus our attention to only those detected maxima points. We do this by generating an image image W with

a white background, i.e. all the pixel values in that image are 255. Image W has the same dimension of 80x60, similar to each extracted frame. Now for each frame, we update W by drawing a small circle at the position of a detected maxima point for that frame. The Algorithm 4.6 explains our method. Figure 4.12 and Figure 4.13, shows the results of our proposed algorithm against white background and the original image background respectively.

Algorithm 4.6 Putting Maxima Points On a White Background

Input:
Image Files ($imFiles$),
Array Holding Maxima Points For Each Frame ($maximaArray$)
Output:
Frames With Maxima Points Marked On White Background
Begin
n = Number of $imFiles$
while ($i < n$)
Begin
$image = imFiles[i]$
$width$ = number of columns in $image$
$radius = width/50$
Convert $image$ to grayscale
Modify all the pixel values in $image$ to 255.
for $points$ **in** $maximaArray[i]$
Begin
Draw circles on $image$ at $points$ with radius=$radius$ and thickness=$(radius/5)$
Save $image$
End
End
End

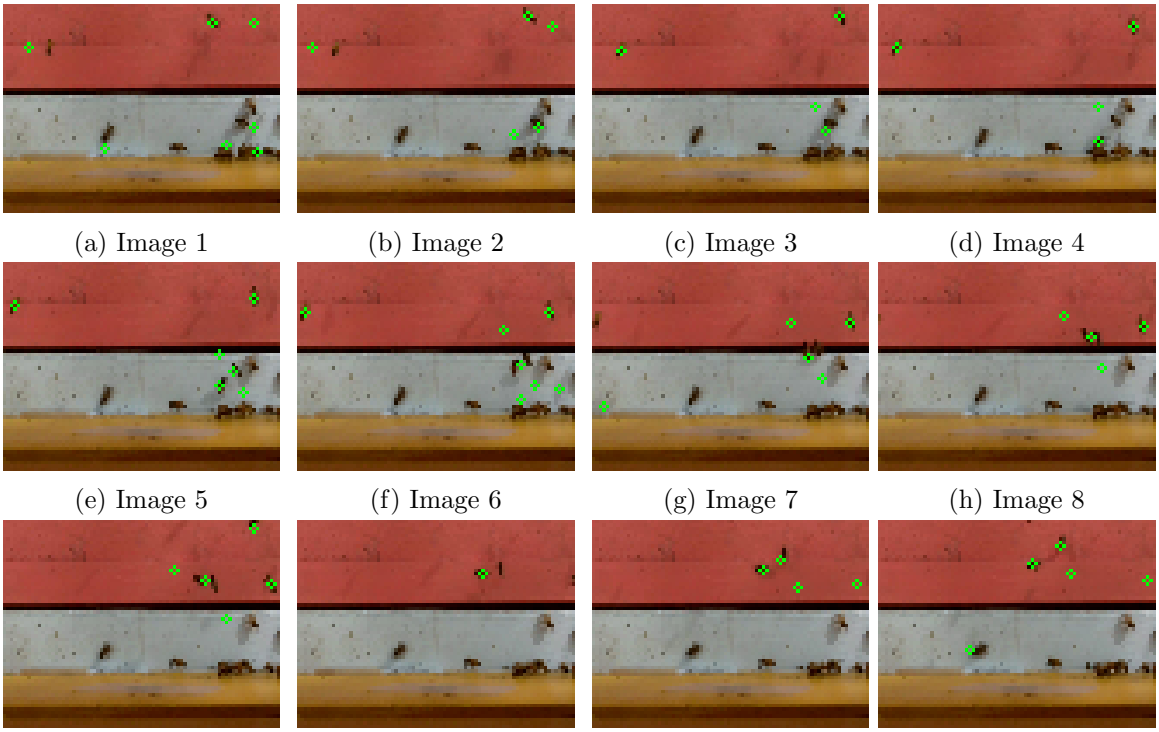


(a) Image 1 (b) Image 2 (c) Image 3 (d) Image 4

(e) Image 5 (f) Image 6 (g) Image 7 (h) Image 8

(i) Image 9 (j) Image 10 (k) Image 11 (l) Image 12

Figure 4.12: Maxima points for the 12 frames in Figure 4.1 on a white background



(a) Image 1 (b) Image 2 (c) Image 3 (d) Image 4

(e) Image 5 (f) Image 6 (g) Image 7 (h) Image 8

(i) Image 9 (j) Image 10 (k) Image 11 (l) Image 12

Figure 4.13: Maxima points for the 12 frames in Figure 4.1 on the actual background

4.10 Threshold For Smoothed Image And Spacing Between Maxima Points

In this section, we will first discuss regarding the choice of threshold value from an entire video. The intuition behind this threshold selection comes from the fact that it is useful to be able to separate out the regions of the image corresponding to objects in which we are interested, from the regions of the image that correspond to background. As you can see in the smoothed images of Figure 4.9, there are certain portions in the images, which if not separated out may have a negative impact in detecting bee movement. Our beehives are located in areas where the background in the recorded videos differ from each other. Thus it is necessary to automate the threshold selection based upon the recorded videos. Towards that end, we first create a global background image I_B from the video. This global background image will have the portions of the video which have remained the same across the video time length. This may include the background landscape in the video, or portions where the sun shines directly for the video time length. The above background image is generated by adding all the frames of the video along the 3 channels and then dividing the result by the number of frames in the video. Next we calculate the color variation between each frame of the video against the global background average image I_B across all three channels. Algorithm 4.7 explains the process of calculating the mean variability of color in a video and subsequently the standard deviation of the mean color variability. A higher value of *meanStd* would suggest the presence of a lot of bee movements in the video or bee movements against plants or vegetation in the background or could be even bee movements in direct sunlight which could create a lot of shadows. In all the above cases, the color variation would be higher. Similarly when the bee traffic level is low, the color variations would be small and thus *meanStd* would be comparatively smaller.

Based upon our observations from different videos collected during various stages of a bee keeping season with varying levels of traffic, we assert that the minimum value of *meanStd* will be close to 5 and the maximum value of *meanStd* will be close to 210. Next, based upon the value of *meanStd* we design a strategy that would decide the threshold and spacing distance for each video. We can recall that the goal of choosing a threshold

Algorithm 4.7 Algorithm To Find The Mean Color Variability

Input:

Image Frames of a Video ($imFiles$),
Number of Image Frames (n),
Global Background Image (I_B),
Smoothing Steps ($smooth$),
Smoothing Factor ($factor$)

Output:

Mean Standard Deviation of color variability (**meanStd**)

Begin

r = Number of rows in $imFiles[1]$
c = Number of columns in $imFiles[1]$
Initialize an empty array $colorVar$. Shape is (r,c)
while ($i < n$)
Begin
$subImage = Subtract(imFiles[i], I_B)$
$smoothImage =$ **SmoothImage**$(subImage, smooth, factor)$
$absImageSqrd = Multiply(smoothImage, smoothImage)$
$sumImSqrd = Sum(absImageSqrd, axis = 2)$. Shape is ($r$,$c$)
The coordinates (m, n) where $sumImSqrd$ is greater $colorVar$ are noted.
Pixel values at $colorVar(m, n)$ is replaced by $sumImSqrd(m, n)$ for only those positions where $sumImSqrd > colorVar$.
End
$colorVar$ holds the positions along with the values of the maximum color.
variations that has occurred across all the frames in the video in comparison to I_B.
$meanVar$ = mean of all the pixel values in $colorVar$
$meanStd$ = Standard Deviation of the $meanVar$

End

Table 4.1: Suggested Spacing Distance

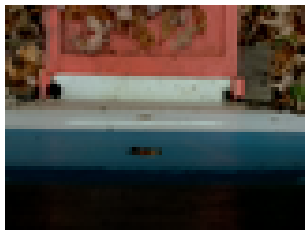
Normalized Threshold ($thresholdNorm$)	Spacing
$9 \leq thresholdNorm < 22$	5
$22 \leq thresholdNorm \leq 65$	3

is to find the minimum acceptable color variation between the frame in context and the corresponding average background image. This color variation would be small for videos with low bee traffic and higher in videos with high bee traffic. Thus we need to choose the upper and lower limits of the threshold in a way that we would be able to eliminate noise in the smoothed image for lower traffic videos and also retain useful information in smoothed images for videos with higher traffic. We assert the lower and upper bounds for the threshold selection to be 9 and 65 respectively. Based upon our experiments we believe that the above is a generous range which would work across different backgrounds and surroundings. Next we take the $meanStd$ values for each video and normalize them between 9 and 65. Without the normalization we would miss a lot of bee movements for higher bee traffic videos, and also incorporate a lot of noisy values for detecting bee movements in low bee traffic videos. The process of normalizing the $meanStd$ values in shown in Equation (4.3):

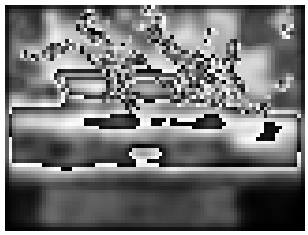
$$thresholdNorm = \frac{meanStd - 5}{210 - 5} \times (65 - 9) + 9 \tag{4.3}$$

A higher value of $thresholdNorm$ will be accompanied by a smaller spacing value between detected maxima points. The reason behind the smaller choosing a smaller value of spacing is due to the fact that a higher value of $thresholdNorm$ signifies a higher level of bee traffic. Thus there is possibilities of multiple bees close to one another and so we choose a smaller value of spacing to take all the bees into consideration as possible. Similarly in case of a lower value of $thresholdNorm$, we would choose a higher value for spacing. Since a lower value of $thresholdNorm$ would signify a lower bee traffic level, thus choosing a larger spacing value, would help us process each frame in the video faster.

Table 4.1 gives us some suggestive values for spacing that could be used if $thresholdNorm$

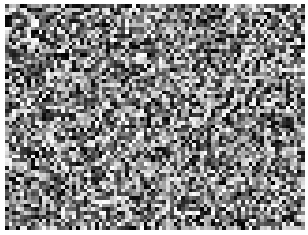


(a) Background Image of a video with Low Traffic



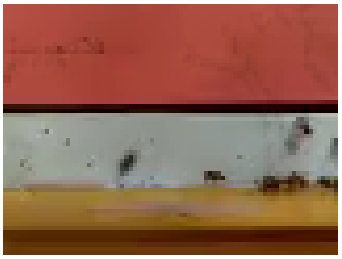
(b) Image *colorVar* for images used to generate Figure 4.14a

(c) Background Image of a video with High Traffic

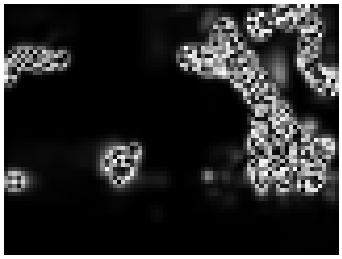


(d) Image *colorVar* for images used to generate Figure 4.14c

Figure 4.14: Sample global background image along with $colorVar$, which holds the values and locations of the highest color variations across all frames in a video



(a) Background Image

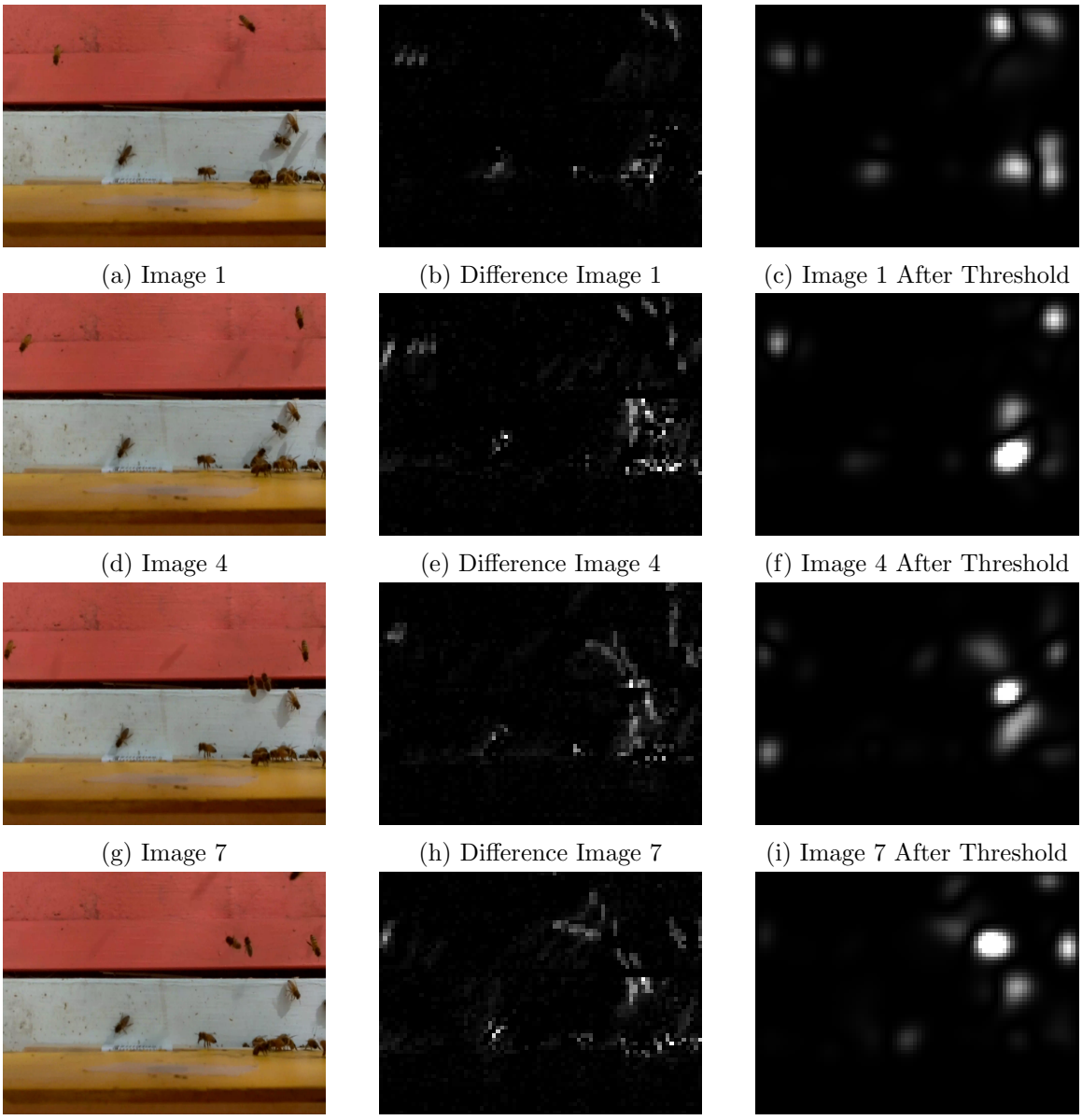


(b) $colorVar$ for Figure 4.1

Figure 4.15: Global background image for the images in Figure 4.1 along with $colorVar$, which holds the values and locations of the highest color variations across all frames. The $meanStd$ for the image in Figure 4.15b is 13.592617488995614 and the correspond $thresholdNorm$ is 11.34725160675 using Equation (4.3)

falls within the defined range. The values shown in Table 4.1 were found out during our experiments with videos with different levels of bee traffic across different times in the bee keeping season between May to September. For our example with sample images as in Figure 4.1, the global background image and the color variability is shown in Figure 4.15.

From Figure 4.15 we can see the regions where the majority of the bee movement was detected matches our observations regarding the bee movements from the images in Figure 4.1. But the bee traffic level was low, which is also evident from the $meanStd$ value which is 13.592617488995614. Thus in this case the $thresholdNorm$ is 11.34725160675 and a minimum spacing distance (the minimum distance between two detected maxima points in the smoothed image) is 5. The result of the application of thresholding is seen in Figure 4.16.



(a) Image 1 (b) Difference Image 1 (c) Image 1 After Threshold

(d) Image 4 (e) Difference Image 4 (f) Image 4 After Threshold

(g) Image 7 (h) Difference Image 7 (i) Image 7 After Threshold

(j) Image 9 (k) Difference Image 9 (l) Image 9 After Threshold

Figure 4.16: The images in the first column are the original frames. The second column is the result of subtracting the average background image from the corresponding frame in the first column. The images in the third column are generated as a result of applying threshold to the result of smoothing the difference image in the second column.

CHAPTER 5
ANALYSIS OF BEEHIVE VIDEO SAMPLES

5.1 Goal

Video bee traffic analysis can be used to assess a honeybee colony's health by automating the bee traffic level assessment. A healthy hive will have little discrepancies in the forager traffic level across multiple days. But consistent fluctuations in the bee traffic levels can indicate that a hive is under-performing and may require special interventions by the beekeeper. Thus accurate measurement of forager traffic level is important in automated bee hive monitoring systems that are able to detect any deviations of bee traffic from norm. We captured videos of forager traffic leaving and entering the hive by placing a camera just above the landing pad of a Langstroth bee hive as seen in Figure 5.1. In our continuing research ([1, 2]), we have studied and watched a number of honeybee traffic videos that involved 3 different bee races: carniolan, italian and buckfast. Our study suggests that honey bee traffic movement can be classified into three types: incoming, outgoing and lateral. Incoming traffic consists of bees entering into the hive through the landing pad or through holes drilled in the supers. Outgoing traffic consists of bees leaving the hive from the landing pad or through holes drilled in the supers. We refer to a bee movement as a lateral movement, when the bees fly parallel to the field of view of the camera just above or near the landing pad. In this chapter, we will be presenting an algorithm based on digital particle image velocity (dpiv) [27], [28] to count honey bee motions and also detect their flight directions from frames extracted from videos recorded by our BeePi electronic beehive monitoring system. In the previous Chapter 4 we have seen how we could take frame with bees and convert them into a frame with white background having only the positions of interest marked, i.e. positions where there are bee movements. In this chapter we will be using those above frames with white background and use our algorithm to count

bee motions between successive frames. Thus we will see how we can use dpiv to analytically classify and count different bee motions into incoming, outgoing or lateral and then use those individual counts as measurements of directional traffic levels. Towards that end we also introduced a dataset of 32 recorded video samples from different times in the bee keeping season and from different hives to aid us and validate our investigation [96].

Figure 5.1: A closer look at the camera looking down on the landing pad

5.2 Related Research

In [58] dpiv was used to measure the turbulence levels in a low turbulence wind tunnel. In that same study, by extending their findings, the authors suggested that dpiv can also be tuned to measure the aerodynamic performance of a small-scale flying device. When an insect, animal or a bird flies through the air, the traces of air particles that move as a result can be studied and used to analyze insect and animal flight patterns. The recent advancements in dpiv [27, 28, 59] has enabled researchers to start investigating the above flight patterns to understand how certain animals or insects fly and the use the relevant information to design micro-air vehicles. In [60], the authors used a high speed camera and applied dpiv to measure the vortex wake and kinematics of a swift's flight through a wind tunnel. In [61], the author's demonstrated the difference between the wakes of birds and

small bat species by using dpiv to analyze the recorded wake images. It was shown that each wing of the bat generated it's own vortex loop along with a sign difference between the circulation on the outer wing and the arm wing during upstroke. Spedding et al. in [62] used dpiv to analyze the flight of thrush nightingales. They used dpiv to measure the balance of forces during the upstroke and the downstroke of the bird's flight. Through their research they showed that it is possible to track the momentum in the wake of a flying animal. There has also been past research such as in [63] and [64], where the authors have used dpiv to study and analyze the flight patterns and the wake structure of nectar-feeding bats and dog-faced fruit bats respectively. The use of dpiv to investigate insect flight was first demonstrated in [65] in which the authors measured fluid velocities of a fruit fly and then examined the contribution of the leading–edge vortex in the overall force production during flight. The authors in [66] were the first to use dpiv to analyze the flow field around the wings of a tobacco hawkmoth while it flew through a wind tunnel. The authors also experimentally showed the flow separation near the leading edge of the wing during the insect's downstroke. In a different research [67], it was shown that dpiv could also be used to examine and map the air flows generated by dancing honeybees.

There have been past research projects that applied computer vision to monitor various aspects of honeybee traffic. One such study was reported by Rodriguez et al. in [68]. They proposed a system that would detect and track multiple insects and animals, with a special interest for monitoring the traffic of honeybees and mice. They designed their system based on deep neural network that associated detected body parts to whole insects and animals. The network initially predicts a set of 2D confidence maps of detected body parts along with a set of vectors that hold the associations among the detected body parts. Next they use greedy inference to select the most likely predictions for each part and then compiles the predictions into larger insects or animals. To detect the bee traffic that leave and enter the hive, the authors use trajectory tracking. The above gives reliable results under smaller traffic levels. The authors reported that the dataset used in the above study consisted of 100 fully annotated frames with 6–14 honeybees per frame. Since the dataset was not made

public, we were not able to independently replicate the results. Furthermore in the above research, a standard Langstroth or Dadant hive, which is commonly used by beekeepers, was not used. Rather a smaller laboratory grade hive was used in the experiments. The proposed system also requires some modifications to the beehive in that a transparent acrylic plastic cover had to be used in order to ensure that the bees remained in the focal plane of the camera.

In another research Babic et al. [69] proposed a system that would detect pollen bearing honeybees from videos recorded at the entrance of a hive. The proposed system included a specially designed wooden box mounted above the hive entrance, and also had a raspberry pi camera attached inside it. The wooden box was specially designed in a way to restrict bees from flying when in the field of view of the camera. To achieve the above, the authors put a glass plate on the bottom of a box just 2cm above the landing pad and in effect force the bees to crawl a distance of ≈11 cm while entering or leaving the hive. The authors reported the training dataset to be composed of 50 images of pollen bearing honeybees and 50 images of honeybees without pollen. The testing data set on the other hand consisted of 354 images of honeybees. Since the dataset was not made public, we were not able to independently replicate the reported results.

We found in [70], the review of various electronic, remote-sensing, and computer-based techniques for observing and monitoring insect movements in the field and the laboratory. In particular, they have reported in their review, the use of electronic bee counters to record bees entering and leaving the hive. Following that we have seen in [71] a practical use of bee counters in modeling the flight activity of honey bees at hive entrance. We also see the use of bee counters in a much recent study in [72], where the authors correlate foraging activity to weather conditions. Although the reported results were promising but in each of those cases, the use of a bee counter needed additional hardware and attachment to a standard hive. The reason is that to detect bee movements at the hive entrance 32 bi-directional channels of 10 mm × 6.5 mm × 6.5 mm (length × width × height) each and separated by 12.7 mm are required for the setup. The detection of bee inside each channel is achieved

via an infrared diode that excites two photoreceptors. Following that, the break order of the receptors indicates whether a bee is entering (in) or leaving (out) of the hive. Thus we can see hardware modifications are needed to be done on the standard hive in order to be able to use the bee counters.

The dpiv based algorithm that we present in this chapter, improves upon our previous research [1] where we proposed a two tier method for bee motion counting based on motion detection and motion region classification using a neural network. The improved bee motion counting method proposed in this chapter is an improvement to our continuing research [2] on using dpiv in counting bee motions. This current method also encapsulates the improvement stated in [2] over [1], such as measuring directional honey bee traffic, not requiring to train a neural network to classify bees and finally providing an insect independent motion detection method. This current method improves the bee motion counts by first localizing the bees that have moved between successive frames and then separating the corresponding positions on a different frame with a white background (refer to Chapter 4) and then applying dpiv to the frames with only the bees that have moved, marked on a white background. The various stages of the proposed algorithm are given in Figure 5.2.

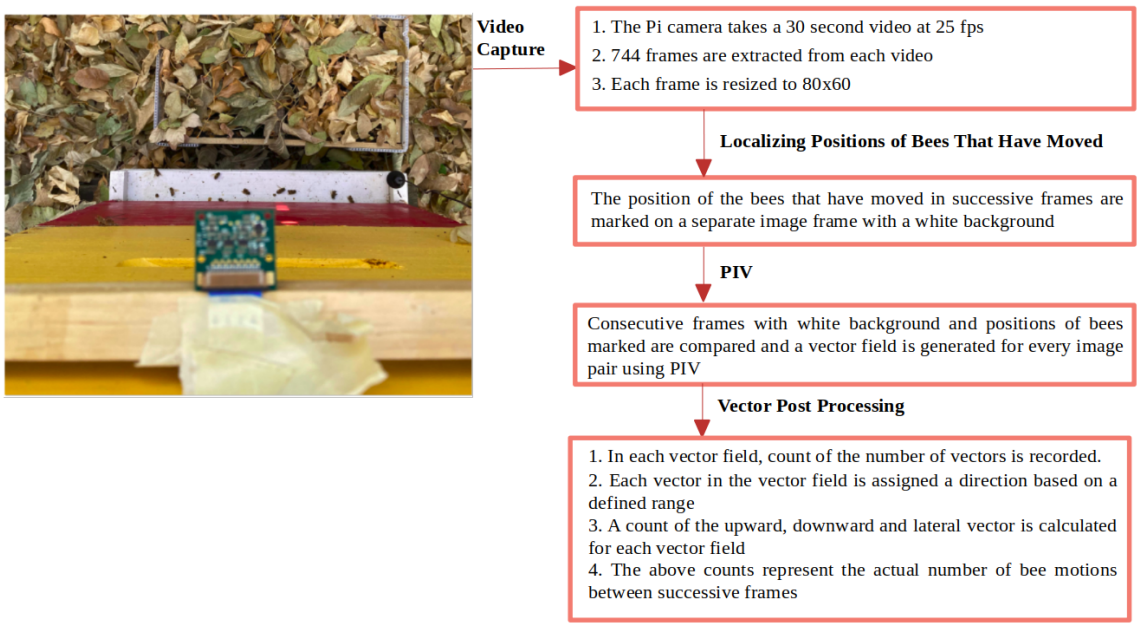


Figure 5.2: Directional dpiv-based bee motion counting algorithm

5.3 Video Data

The video data used for experiments in this chapter have been collected from bee hive monitors from two different bee keeping seasons across the same apiary. Figure 5.1 shows the relative position of the camera above the landing pad on the beehive. Each BeePi monitor recorded 30 seconds of video samples with an interval of 15 minutes. For this chapter, we chose 32 videos across different times during the bee keeping season of 2018 and 2019. The videos were chosen in way such that they had varying levels of bee traffic across different backgrounds. Next, we extracted 744 frames from each video. Hence our dataset contained a total of 23808 (744*32) frames. The data is publicly available at [96]. The bee count in the first frame for every video was 0. After that each of those frames were individually studied and the number of bees that moved between successive frames in a video were counted. A bee was considered to have made a motion and was accounted for if it its position changed between successive frames. We also counted bees that were not in a previous frame but appeared in the following frame (when the bee flew into the camera's field of view). The above count for the number of bees that moved between successive

frames acted as our validation data. In this chapter we will refer to the above counts as bee motion counts or count of the bees interchangeably.

5.4 Digital Particle Image Velocimetry (dpiv)

We will be working with two consecutive image frames from a video, F_t and F_{t+1} in order to explain the workings of dpiv. Frame F_t represents a frame at time t and F_{t+1} represents a frame at time $t+1$. We select a $D \times D$ window, IA_1 from F_t which is centered at position (i, j). The $D \times D$ window is often referred to as *interrogation area* in the dpiv literature. We also select another $D' \times D'$ window, IA_2 from F_{t+1} such that $D \leq D'$. The position of the interrogation window IA_2 in frame F_{t+1} is a function of the position of IA_1 in frame F_t, since we move IA_2 relative to IA_1 in order to find the maximum correlation peak. As we move IA_2 around the $D' \times D'$, we calculate for each possible position of IA_1 in F_t, a corresponding position of IA_2 is computed in F_{t+1}. This is done in order to compare the two windows and find the corresponding positions of the particles. The comparison is performed mathematically by calculating a 2D matrix correlation between IA_1 and IA_2 using the correlation formula in Equation 5.1.

$$\begin{gathered} C(r,s) = \sum_{i=0}^{D-1} \sum_{j=0}^{D-1} IA_1(i,j) IA_2(i+r, j+s), \\ \textit{where } \mathrm{r,\ s} \in [-\lfloor \tfrac{D+D'-1}{2} \rfloor, \ldots, \lfloor \tfrac{D+D'-1}{2} \rfloor] \end{gathered} \tag{5.1}$$

In Equation 5.1, $IA_1(i, j)$ is the pixel intensity at location (i, j) in image frame F_t and $IA_2(i + r, j + s)$ is the pixel intensities at location $(i + r, j + s)$ in frame F_{t+1}. If the size of IA_1 is $M \times N$ and the size of IA_2 is $P \times Q$, then the size of the final correlation matrix C is $(M + P - 1) \times (N + Q - 1)$. The correlation matrix C records the correlation coefficient for each possible alignment of IA_1 with IA_2 as we iterate through different values of r and s. Let $C(r_h, s_h)$ be the highest value in correlation matrix C. If (i_c, j_c) is the center of the interrogation window IA_1, then the positions of (i_c, j_c) and (r_h, s_h) define a displacement vector $\vec{v}_{i_c,j_c,r_h,s_h}$ from location (i_c, j_c) in frame F_t to $(i_c + r_h, j_c + s_h)$ in frame F_{t+1}. This vector is a representation of where the particles in frame F_t have moved to in the next

frame F_{t+1}. All such displacement vectors between successive frames form a vector field that can be used to estimate possible flow patterns of the particles. Figure 5.3 is a pictorial representation of how the cross correlation coefficients are computed between successive frames. One thing to keep in mind is that Equation 5.1 leads to a time complexity of $O(n^2)$ and hence it is a slow process. A faster way to compute the cross correlation between two successive frames is to use Fast Fourier Transform (FFT), as shown in Equation 5.2. The important criteria for solving Equation 5.2 is that the two interrogation windows IA_1 and IA_2 must be of the same size i.e. $D = D'$.

$$C(r, s) = \Re[FFT^{-1}(FFT^{*}(IA_1) \cdot FFT(IA_2))] \tag{5.2}$$

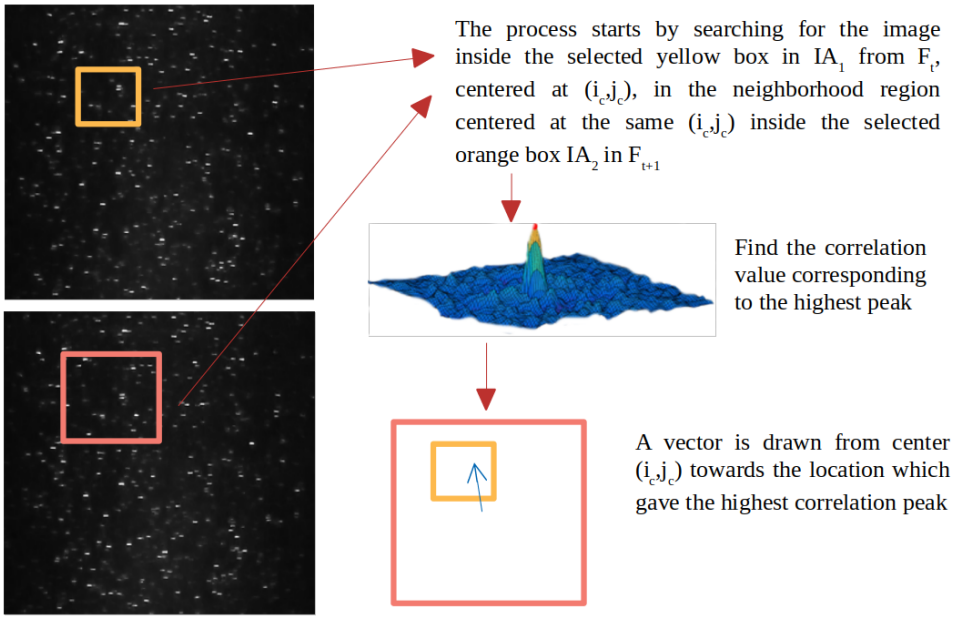


Figure 5.3: 2D Cross Correlation Algorithm. The upper image with particles on the left is F_t. The lower image on the left is F_{t+1}. The region with the light orange borders in the upper image is IA_1. The region with the pink borders in the lower image is IA_2. The 3D plot on the right plots all 2D correlation values. The highest peak helps to estimate the general flow direction.

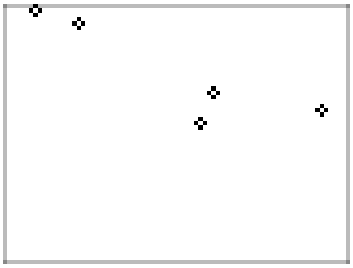
For many real world images, the displacement vectors generated have background noise or other real-life imperfections. These vectors must be eliminated from the vector field in

order to avoid large errors. This can be caused by small interrogation window size or due to the quality of the images. We have information about the signal to noise ratio of the vectors from the cross-correlation function, and so we can classify a vector as an outlier if it's signal to noise ratio exceeds a certain threshold. The final step is to replace the missing/outlier vector by computing the average of it's immediate neighbors. The position of the highest correlation peak can also be fine tuned as function of the second and third highest peaks in the correlation plane to achieve subpixel accuracy.

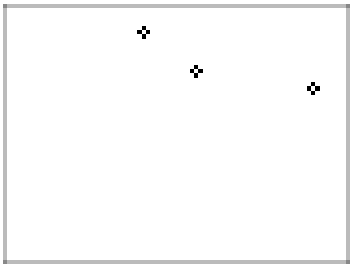
Figure 5.4e shows the directional vectors computed by the dpiv based algorithm from two consecutive image frames of size (80×60) from a 30-s video captured by a deployed BeePi monitor. The above vectors were generated by using an interrogation window of size (27×27) and an overlap of 40%. In Figure 5.4a, two moving bees are marked using two green circles. The bees were marked by comparing the immediate predecessor image frame (not shown here) to Figure 5.4a. In Figure 5.4b we can see that the same two bees have moved again and along with that another bee on the right end has moved too. Hence those three bees have been marked with green circles for our reference. Figures 5.4c and 5.4d are generated by applying the method presented in Chapter 4. We can recall from Chapter 4 that Figures 5.4c and 5.4d are not generated by comparing only the immediate successor and predecessor frames, rather each of the above two frames are generated by comparing the image frames which are 1/2 second before and after the corresponding frame. In other words, Figure 5.4d was generated by analyzing the 12 frames before and 12 frames after Figure 5.4d. In Figure 5.4c we can see that 5 bee movements have been detected whereas in Figure 5.4d, 3 bee movements have been detected. This tells us that between the time period of two image frames, 3 bees have moved. This is also reflected in Figure 5.4e, where we can see 3 vectors have been detected; one vector for each bee movement. We can also see that the three vectors are of different lengths. If we observe Figures 5.4a and 5.4b, we can see that the bee at the top of the image has moved the most and the bees in the middle showed small movements between successive frames. The bee at the top had moved more or less parallel to the landing pad, whereas the bees in the middle moved upward. We can

(a) Image 1

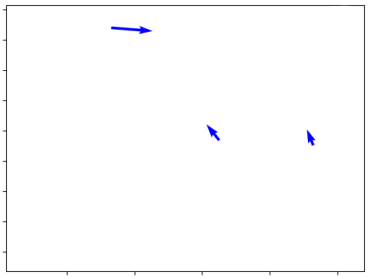
(b) Image 2



(c) Localized bees in Image 1



(d) Localized bees in Image 2



(e) Vectors generated by comparing Images (**c**,**d**)

Figure 5.4: Image **a** and Image **b** are two consecutive Images selected from a 30-s video. We have marked the bees that have moved with green circles. Image **c** and Image **d** are generated using the algorithm described in Chapter 4. They individually represent the bees that have moved by comparing 1/2 sec of video before and after Image **a** and Image **b** respectively. The vector field in (**e**) is generated from the images in (**c**,**d**) using dpiv.

see the above directional characteristics are reflected in the vectors in Figure 5.4e as well.

Once the dpiv vectors are computed for a pair of consecutive image frames, the angles of individual vectors are used in the calculation of directional bee traffic levels. Specifically, each vector is classified as *lateral*, *incoming*, or *outgoing* according to the corresponding angle value as shown in Figure 5.5. A vector $\vec{v}$ is classified as *outgoing* if it's corresponding angle is in the range $[11°, 170°]$, as *incoming* if it's angle is in the range $[-11°, -170°]$, and as *lateral* if it's angle is in the ranges $[-10°, 10°]$, $[171°, 180°]$, or $[-171°, -180°]$.

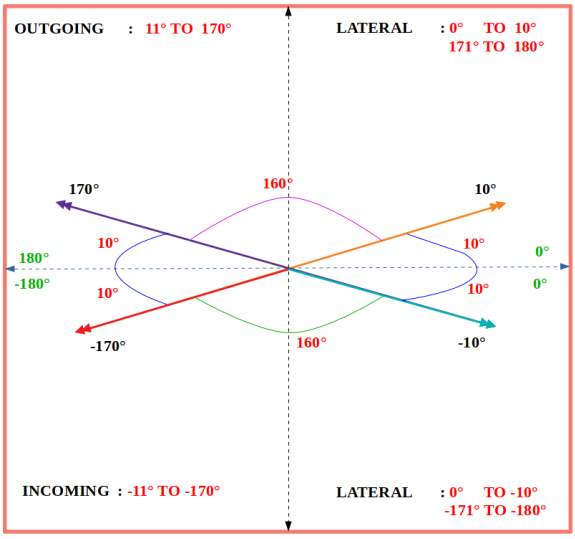


Figure 5.5: Degree ranges used to classify dpiv vectors as lateral, incoming, and outgoing.

Let F_t and F_{t+1} be two consecutive image frames from a video V at time t and $t+1$ respectively. Let us also refer to $I_f(F_t, F_{t+1})$, $O_f(F_t, F_{t+1})$, and $L_f(F_t, F_{t+1})$ as the counts of incoming, outgoing, and lateral vectors when dpiv is applied to frames F_t and F_{t+1}. For each pair of consecutive image frames F_t and F_{t+1}, we compute three non-negative integers: $I_f(F_t, F_{t+1})$, $O_f(F_t, F_{t+1})$, and $L_f(F_t, F_{t+1})$. If a video V is a sequence of n frames $(F_1, F_2, \ldots, F_n)$, then we can use I_f, O_f, and L_f to define the functions $I_v(V)$, $O_v(V)$, and $L_v(V)$ that return the total counts of the incoming, outgoing, and lateral vector counts for video V, as shown in Equation (5.3).

$$
\begin{aligned}
I_v(V) &= \sum_{i=1}^{n-1} I_f(F_i, F_{i+1}) \\
O_v(V) &= \sum_{i=1}^{n-1} O_f(F_i, F_{i+1}) \\
L_v(V) &= \sum_{i=1}^{n-1} L_f(F_i, F_{i+1})
\end{aligned}
\tag{5.3}
$$

For example, let $V = (F_1, F_2, F_3)$ such that $I_f(F_1, F_2) = 10$, $O_f(F_1, F_2) = 4$, $L_f(F_1, F_2) = 3$ and $I_f(F_2, F_3) = 2$, $O_f(F_2, F_3) = 7$, $L_f(F_2, F_3) = 5$. Then, $I_v(V) = I_f(F_1, F_2) + I_f(F_2, F_3) = 10 + 2 = 12$, $O_v(V) = O_f(F_1, F_2) + O_f(F_2, F_3) = 4 + 7 = 11$, and $L_v(V) = L_f(F_1, F_2) + L_f(F_2, F_3) = 3 + 5 = 8$. For each pair of consecutive image frames F_t and F_{t+1}, the omnidirectional motion vector count is defined as the sum of the values of I_f, and O_f, and L_f, as shown in Equation (5.4). Then, for each video V, the omnidirectional vector count $T_v(V)$ for the video is the sum of the three directional counts, as also shown in Equation (5.4).

$$
\begin{aligned}
T_f(F_t, F_{t+1}) &= I_f(F_t, F_{t+1}) + O_f(F_t, F_{t+1}) + L_f(F_t, F_{t+1}) \\
T_v(V) &= I_v(V) + O_v(V) + L_v(V)
\end{aligned}
\tag{5.4}
$$

5.5 Experiments

5.5.1 Interrogation Window and Overlap

In Chapter 4, we learned about the selection of an appropriate threshold value from a video in order to be able to separate out the regions of the image corresponding to objects in which we are interested, from the regions of the image that correspond to background. To achieve the above, we first created a global background image and then calculated the color variation between each frame of the video against the global background average image in order to find the mean variability of color in a video. A higher value of *meanStd* (i.e.

standard deviation of the mean color variability) suggested the presence of a lot of bee movements in the video or bee movements against plants or vegetation in the background or even bee movements in direct sunlight which could create a lot of shadows. In all the above cases, the color variation would be higher. Similarly when the bee traffic level is low, the color variations would be small and thus $meanStd$ would be comparatively smaller. We observed and investigated various video recordings from the Beepi monitors over the course of the bee keeping season of 2018 and 2019 and came to the conclusion that minimum value of $meanStd$ is close to 5 and the maximum value of $meanStd$ is close to 210. Next, we normalized the $meanStd$ values for each video to be between 9 and 65 and generated $thresholdNorm$. Following that we learned in Chapter 4, how $thresholdNorm$ can be used to determine the spacing value between detected maxima points. In this section we will see how $thresholdNorm$ is used to determine the interrogation window size and the overlap between two interrogation windows, which are the necessary parameters to perform dpiv.

Before we design a strategy to determine an optimal interrogation window size and amount of overlap, it is important to know that their values greatly depend upon the type of experiments we perform. The information that we have regarding our experiments are that the videos have either high traffic, low traffic and medium traffic. We explain the process of designing a bee motion counting algorithm **DPIV_A** in Appendix A.

The video data used for experiments in this chapter have been collected from bee hive monitors from two different bee keeping seasons across the same apiary. Each BeePi monitor recorded 30 seconds of video samples with an interval of 15 minutes. For this chapter, we chose 32 videos across different times during the bee keeping season of 2018 and 2019 [96]. The videos were chosen in way such that they had varying levels of bee traffic across different backgrounds. We first used 20 videos for our analysis in order to determine a strategy to choose the interrogation window size and the overlap based upon $thresholdNorm$. Then we tested our designed strategy on the remaining 12 videos. In total our training dataset contained a total of 20 videos, i.e. 14880 (744*20) frames and the testing set had 12 videos, i.e. 8928 (744*12) frames. To generate the ground truth data, each of those frames were

Normalized Threshold ($thresholdNorm$)	Interrogation Window	Overlap
$0 \leq thresholdNorm < 11$	13	1
$11 \leq thresholdNorm < 11.4$	13	3
$11.4 \leq thresholdNorm < 11.8$	13	5
$11.8 \leq thresholdNorm < 12$	15	1
$12 \leq thresholdNorm < 12.2$	15	3
$12.2 \leq thresholdNorm < 12.4$	15	5
$12.4 \leq thresholdNorm < 12.6$	17	1
$12.6 \leq thresholdNorm < 12.8$	17	3
$12.8 \leq thresholdNorm < 13$	17	5
$13 \leq thresholdNorm < 13.2$	19	1
$13.2 \leq thresholdNorm < 13.4$	19	3
$13.4 \leq thresholdNorm < 13.6$	21	1
$13.6 \leq thresholdNorm < 14.4$	21	3
$14.4 \leq thresholdNorm < 15$	21	5
$15 \leq thresholdNorm < 15.4$	23	1
$15.4 \leq thresholdNorm < 16$	23	3

Table 5.1: Suggested values for interrogation window and overlap

individually evaluated before hand and the number of bees that moved in successive frames were counted from them.

We saw in Appendix A that there are different strategies possible to select the interrogation window size and overlap and we saw that fixing the interrogation window size was not an ideal choice. Hence, here we present two tables, Tables 5.1 and 5.3 that show the suggested values with varying interrogation window size and overlap. Table 5.1 represents videos with smaller $thresholdNorm$ value. In other words we can say Table 5.1 represents the suggested values for videos with low to mid-low bee traffic. Since the bee traffic level is low, the overlap between two interrogation windows is chosen to be small as we can see in Table 5.1. We chose the overlap to be approximately 10%, 20% and 30% respectively between two interrogation windows. We started with an interrogation window that was almost $\frac{1}{5}th$ of the height of each image frame and then increased the size by 2 pixels after each iteration of 10%, 20% and 30% overlap for an interrogation window.

In Table 5.3 we can see that the overlap between two interrogation windows were chosen to be approximately 1%, 10%, 20%, 30%, 35%, 45%, 50%, 60% and 70% respectively. The reason behind the choice of the above overlap percentages is due to the fact that we wanted

to capture all of the information from videos that had mid to high bee traffic levels. We started with an interrogation window that was almost $\frac{2}{5}th$ of the height of each image frame. We found through our experiments that interrogation windows of size 25 and 27 worked well for videos with mid to high traffic levels. We also saw that for really high traffic videos interrogation window of size 27×27 and an overlap of 70% pixels were able to gather most of the information regarding bee movement. Table 5.2 shows the remaining important dpiv parameters that were used for the experiments. We used the above strategy to design the bee motion counting algorithm **DPIV_B**.

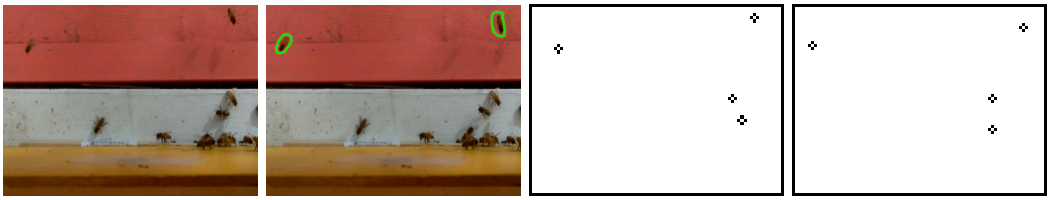
Parameter	**Value**
Img. Size	80×60
Inter. Win. Correlation	FFT
Signal to Noise Ratio	peak1/peak2 with a threshold of 0.05
Spurious Vector Replacement	local mean with kernel size of 2 and max. iter. limit of 15

Table 5.2: Additional dpiv parameters used for the experiments.

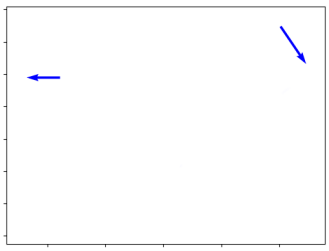
We used some of the images that we tested with in Chapter 4 for visualizing the workings of the combined algorithm, the one proposed in Chapter 4 and then adding the dpiv based analysis to it. For the convenience of the reader, we have marked in green circles the bees that have moved between successive frames. In Figure 5.6, we can see that between Figures 5.6a and 5.6b, two bees have moved, one downward and one sideways (or lateral). We can see that the above bee movements between successive image frames in Figures 5.6a and 5.6b have been correctly depicted by the vectors in Figure 5.6e.

Normalized Threshold ($thresholdNorm$)	Interrogation Window	Overlap
$16 \leq thresholdNorm < 16.4$	25	1
$16.4 \leq thresholdNorm < 17$	25	3
$17 \leq thresholdNorm < 17.4$	25	5
$17.4 \leq thresholdNorm < 17.8$	25	7
$17.8 \leq thresholdNorm < 18.2$	25	9
$18.2 \leq thresholdNorm < 18.6$	25	11
$18.6 \leq thresholdNorm < 19$	25	13
$19 \leq thresholdNorm < 19.4$	25	15
$19.4 \leq thresholdNorm < 20$	25	17
$20 \leq thresholdNorm < 20.5$	27	1
$20.5 \leq thresholdNorm < 21$	27	3
$21 \leq thresholdNorm < 21.5$	27	5
$21.5 \leq thresholdNorm < 22$	27	7
$22 \leq thresholdNorm < 24$	27	9
$24 \leq thresholdNorm < 30$	27	11
$30 \leq thresholdNorm < 36$	27	13
$36 \leq thresholdNorm < 42$	27	15
$42 \leq thresholdNorm < 48$	27	17
$ThresholdNorm \geq 48$	27	19

Table 5.3: Suggested values for interrogation window and overlap



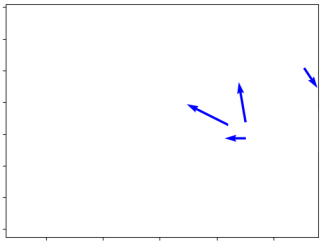
(a) Image 1 (i1) (b) Image 2 (i2) (c) Localized bees i1 (d) Localized bees i2



(e) Vectors

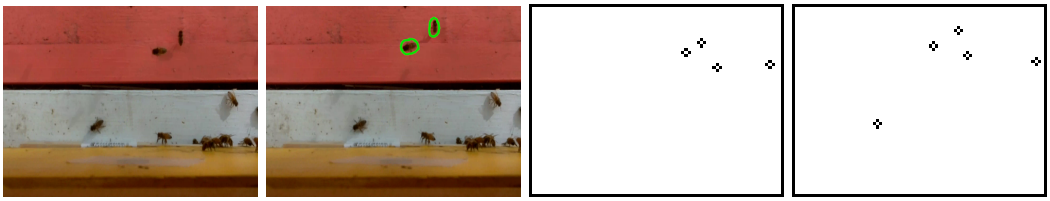
Figure 5.6: Image **a** and Image **b** are two consecutive Images selected from a 30-s video. We have marked the bees that have moved with green circles in Image **b**. Image **c** and Image **d** are generated using the algorithm described in Chapter 4. The vector field in (**e**) is generated from the images in (**c**,**d**) using dpiv.

(a) Image 3 (i3) (b) Image 4 (i4) (c) Localized bees i3 (d) Localized bees i4

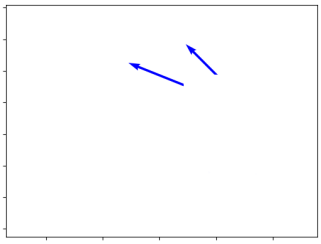


(e) Vectors

Figure 5.7: Image **a** and Image **b** are two consecutive Images selected from a 30-s video. We have marked the bees that have moved with green circles in Image **b**. Image **c** and Image **d** are generated using the algorithm described in Chapter 4. The vector field in (**e**) is generated from the images in (**c**,**d**) using dpiv.



(a) Image 5 (i5) (b) Image 6 (i6) (c) Localized bees i5 (d) Localized bees i6



(e) Vectors

Figure 5.8: Image **a** and Image **b** are two consecutive Images selected from a 30-s video. We have marked the bees that have moved with green circles in Image **b**. Image **c** and Image **d** are generated using the algorithm described in Chapter 4. The vector field in (**e**) is generated from the images in (**c**,**d**) using dpiv.

In Figures 5.7a and 5.7b, we can see that three bees have moved, two upwards and one downwards. But in Figure 5.7e, we can see that an additional vector or bee movement has been detected. This is an example where the localization as described in Chapter 4 might not have pointed towards the exact body part of a bee, wherein one figure, one of the wings of the bee has been localized and marked and in the next figure the thorax of the bee has been localized. But dpiv sees the above scenario as two different pixel locations, and hence we see that additional vector.

In Figures 5.8a and 5.8b, we can see that two bees have moved upwards between successive frames. In Figure 5.8e we see the same result, where only 2 vectors are shown as a result of applying dpiv. If we observe Figure 5.8d, we can see that 5 different bee movements had been localized. We need to keep in mind that these bee movements are those bees that have moved over 1/2 second before and after the frame (Figure 5.8b) in context and are not necessarily the bees that have moved between successive frames. Hence, even though in Figure 5.8d, 5 bee movements have been detected, but dpiv reduces those 5 bee movements to only 2 by comparing the successive frames in Figures 5.8c and 5.8d respectively. Thus we can see by applying dpiv to the results of the localization algorithm presented in Chapter 4, we are able to get the actual count of the bee movements between successive image frames in a video.

5.5.2 Omnidirectional Bee Motions

In this section, we will compare the performance of the combined dpiv-based bee motion estimation algorithm with our previous research in [1], where we used a two-tier method to count omnidirectional bee motions using motion detection and motion region classification. As we stated in the previous section, we had in total 32 videos to analyze and compare the performance. For each video, full bee motions were counted frame by frame. The number of bee motions in the first frame of each video (Frame 1) was taken to be 0. In each subsequent frame, the number of bee motions were counted depending upon their positional change in comparison to the previous frame. In addition to counting bee motions between consecutive frames F_t and F_{t+1}, we also counted the bees that appeared in the current frame F_{t+1} but

were not present in the immediate predecessor frame F_t (for example, when a bee flew into the camera's field of view when frame F_{t+1} was captured). For each video V, we calculate the omnidirectional vector count $T_v(V)$ by the adding the three directional counts, as also shown in Equation (5.4).

We compared the performance of the proposed dpiv based method (**DPIV_B**) against the four best configurations of the two-tier method for omnidirectional bee counting presented in [1]: MOG2/VGG16, MOG2/ResNet32, MOG2/ConvNetGS3, and MOG2/ConvNetGS4. The first element in each configuration (e.g., MOG2 [97]) is a motion detection algorithm; the second element (e.g., VGG16 [80]) is a classifier that classifies each motion region detected by the algorithm specified in the first element. The performance was evaluated against the actual bee motion counts which were counted frame by frame from the 32 videos. In this section we used the omnidirectional vector count $T_v(V)$ for our analysis.

Video	Human Count	DPIV_B	GS3	GS4	ResNet32	VGG16
Vid1	5693	4923	16569	15109	13362	16647
Vid2	343	319	57	43	25	47
Vid3	2887	2106	597	316	145	1245
Vid4	73	56	57	127	25	47
Vid5	75	64	28	27	19	26
Vid6	239	285	97	81	45	95
Vid7	74	92	126	116	102	94
Vid8	361	712	175	274	90	329
Vid9	478	1393	314	319	130	229
Vid10	357	221	500	362	420	166
Vid11	373	512	525	708	403	333
Vid12	348	778	368	422	301	235
Vid13	176	221	157	181	158	148
Vid14	208	290	82	84	60	80
Vid15	456	416	463	466	391	492
Vid16	270	261	862	860	848	867
Vid17	154	296	180	185	183	195
Vid18	294	298	1518	1490	1485	1532
Vid19	17	22	61	59	59	62
Vid20	60	97	133	138	133	138
Vid21	432	685	1299	1279	1226	1279
Vid22	101	610	3501	3673	3052	3614
Vid23	247	279	1030	1011	1049	1021
Vid24	168	250	744	735	716	825
Vid25	90	106	0	1	0	0
Vid26	74	29	13	16	14	7
Vid27	1943	2481	2053	2193	1827	2085
Vid28	6580	8635	26611	28165	26279	27366
Vid29	186	165	492	155	52	102
Vid30	289	498	586	551	639	493
Vid31	643	815	2767	2828	2724	2271
Vid32	1401	1738	577	643	792	430

Table 5.4: Performance of **DPIV_B** on the 32 evaluation videos, the size of each video frame was (80×60): the second column gives the human bee motion counts for each video; the third column represents the results of bee motion counts ($T_v(V)$) by using the dpiv based method presented in this chapter; the remaining columns VGG16, ResNet32, GS3 and GS4 give the bee motion counts returned by the configurations MOG2/VGG16, MOG2/ResNet32, MOG2/ConvNetGS3, and MOG2/ConvNetGS4, respectively; the first 4 videos were the same videos used for evaluation in our previous research [1] and [2]. For visual representation of the absolute error for each video refer to Figure A.1.

In Table 5.4, we can see the performance of the dpiv based bee motion counts method (**DPIV_B**) presented in this chapter. Our goal in this section of the investigation is to see if dpiv can better approximate bee motion counts as compared to the omnidirectional bee counting approach proposed in [1]. From Table 5.4 we can see for the 32 evaluation videos, the dpiv based approach was able return bee motion counts which was on par and in some cases even better than the omnidirectional bee counting approach in terms of its closeness to the actual bee counts. We can see in certain cases where the actual bee motion counts were high (for example in Vid1 and Vid28), the dpiv based method was able to be generate bee counts which were much closer to the actual counts as compared to the omnidirectional approach. Similarly, if we look at other cases (for example, Vid31, Vid23, Vid16 etc), we can see that the omnidirectional approach has over estimated the bee counts when there was medium bee traffic. The above cases could be an issue if we were to build a bee hive monitoring system based upon bee motion counts, since the over estimating might lead to faulty analysis where the monitoring system would be modeled to account for more bee motions and in effect could lead to discrepancies.

The dpiv based method on the other hand gave results which were very close to the actual counts. We can see in certain cases (for example, Vid4, Vid5, Vid19, Vid25 etc) where the actual bee motion counts were low, our dpiv based method was able to generate lower bee counts as well. On the other hand in case of high bee traffic (for example, Vid28, Vid1, Vid3 etc), we can see that the dpiv based approach resulted in higher bee motion counts and also closer to the actual counts. Except for Vid8 and Vid9 where the dpiv method overestimated the counts, we can see dpiv based approach was able to better approximate the actual bee motion counts as compared to the omnidirectional bee counting approach proposed in [1]. We think the overestimation could be a result of the color variation present in the videos, which in turn led to a higher threshold value and thus the algorithm chose a much larger overlap percentage between the interrogation windows that were used for the two videos. The above findings and the results presented in Table 5.4 tells us that the dpiv based bee motion counting approach combined with bee motion localization (which was

presented in Chapter 4) is more accurate and a valid alternative to the costly convolutional neural network techniques used in the omnidirectional bee counting methods in [1].

5.5.3 Results On Raspberry Pi

One of the primary objectives of this research is to design an automated beehive monitoring system that would be able to make informed decisions regarding the health of a bee hive. Hence it is essential to be able to do that sort of analysis in a timely manner on the BeePi monitors itself. One of the building blocks for the above analysis is to get a near accurate estimate of bee motion counts from the video recordings in a timely manner. By timely manner we mean that the bee motion counts should be obtained within 15 mins from recording a video, because each BeePi monitor records a 30-s video every 15 mins from 8:00 to 21:00 every day. Thus it is important to have the bee motion counts from a recorded video before the next one starts recording. In this way we will be able to have almost 'real' time bee motion counts and thus would be able to detect any discrepancies in the bee traffic levels faster.

By evaluating 32 videos of varying traffic from different times of the beekeeping season, we saw that our proposed dpiv based bee motion counts were closer to the actual bee motion counts as seen in Table 5.4. But from [2], we know dpiv is computationally expensive on the raspberry pi hardware even when cross correlation is implemented with FFT (see Equation (5.2)). We saw in [2], that when dpiv was performed on video frames of size 640×480 (the interrogation window size was 90, and the overlap was 60%), it took on average $\approx$2.5 h for the algorithm to process a single 30-s video. In that above study, we then introduced multi-threading and were able to reduce the average video processing time from 2.5 h to 1 h 12 min on a single raspberry pi 3 model B v1.2 with four cores. In order to reduce the processing time even further, in [2] we introduced parallelism to our system by distributing the video processing over 6 raspberry pis. This helped us reduce the time to $\approx$19.49 min per 30-sec video. We later argued that it was still possible to finish analyzing videos from an entire day before the next day's video recording started at 8am by using 6 raspberry pis. But this meant that the 6 raspberry pis would be processing only videos

during majority of its uptime and hence we will not be able to analyze data from any other sensors apart from the camera. Thus it was essential for us to come up with a solution in this current study that would reduce the video processing time to less than 15 minutes on a single raspberry pi.

Towards that end, the first step we took was to reduce the size of the video frames from (640×480) to (80×60). We saw in Table 5.4 that this reduction in frame size did not affect the performance of our dpiv based method and the results were close to the actual bee motion counts. This reduction in video frame size acted as a principle factor in lowering the execution time of the current dpiv based method. Using the above adjustment we were able to achieve an improved processing time with an average of 2.15 minutes per video when we tested on our 32 evaluation videos.

Video	Human Count	dpiv Count	TIME (seconds)	TIME (minutes)
Vid1	5693	4923	243.606929063797	4.06011548439662
Vid2	343	319	114.346977233887	1.90578295389812
Vid3	2887	2106	115.406357288361	1.92343928813935
Vid4	73	56	190.043186903	3.16738644838333
Vid5	75	64	93.5045878887177	1.5584097981453
Vid6	239	285	85.085688829422	1.4180948138237
Vid7	74	92	88.9132161140442	1.48188693523407
Vid8	361	712	84.1732139587402	1.40288689931234
Vid9	478	1393	88.4134678840637	1.47355779806773
Vid10	357	221	90.2447443008423	1.50407907168071
Vid11	373	512	88.8227555751801	1.48037925958634
Vid12	348	778	88.4346191883087	1.47391031980515
Vid13	176	221	113.800625562668	1.89667709271113
Vid14	208	290	95.7730674743652	1.59621779123942
Vid15	456	416	118.601377487183	1.97668962478638
Vid16	270	261	126.284482717514	2.10474137862523
Vid17	154	296	121.399843931198	2.02333073218663
Vid18	294	298	113.230691432953	1.88717819054922
Vid19	17	22	180.32129240036	3.00535487333933
Vid20	60	97	238.475056171417	3.97458426952362
Vid21	432	685	229.064736604691	3.81774561007818
Vid22	101	610	85.9913864135742	1.43318977355957
Vid23	247	279	85.4251458644867	1.42375243107478
Vid24	168	250	112.930651903152	1.8821775317192
Vid25	90	106	89.5929608345032	1.49321601390839
Vid26	74	29	93.2399213314056	1.55399868885676
Vid27	1943	2481	111.709685564041	1.86182809273402
Vid28	6580	8635	192.920397996902	3.21533996661503
Vid29	186	165	120.029965639114	2.00049942731857
Vid30	289	498	84.8380711078644	1.41396785179774
Vid31	643	815	87.9198305606842	1.46533050934474
Vid32	1401	1738	352.485249757767	5.87475416262945

Table 5.5: Processing times for the combined methods of bee motion localization along with dpiv on each of the 32 evaluation videos; the size of each video frame was (80×60)

Table 5.5 shows the time taken to process each video on the BeePi monitor using raspberry pi 3 model B v1.2. From Table 5.5 we can see that the 32 videos individually had different processing times but the highest time was taken by Vid32 (5.87475416262945 minutes) which was still less than our desired 15 minutes. We found the processing times to be dependent on the interrogation window size and the overlap used. For examples, Vid32 had a smaller window size and a larger overlap (13, 40%) and thus the processing time was higher. On the other hand Vid1 had a larger window and overlap (25, 70%) between two interrogation windows and thus the processing time was higher. Thus we can most certainly say that the proposed dpiv based bee motion counts method along with bee

motion localization would be able to process the videos in 'real' time with a much better accuracy, before the next video starts recording on our BeePi monitors.

5.5.4 Directional Bee Traffic Analysis

In our previous investigation [1] we had proposed a two tier method of bee motion counting which is only able to count omnidirectional bee traffic and hence cannot be used to perform any analysis on directional (upward, downward and lateral) bee traffic. In our following research [2], we were able to use dpiv to estimate the levels of incoming, outgoing, or lateral bee traffic rather than actual counts. But in this chapter, we have seen how our combined approach of bee movement localization and then applying dpiv can be used to get actual counts for directional bee motions. Thus in this section, we will investigate whether the newly proposed approach can be used to analyze the incoming and outgoing traffic in healthy bee hives.

In a healthy bee hive, the levels of incoming and outgoing bee traffic should match closely because all forager traffic should come back to the hive at the end of the day. If we consider the counts of incoming and outgoing traffic as time series data, then local mismatches are bound to happen, because some foragers who leave the hive at the same time may not return to the hive at the exact same time as they might have pursued flight to different foraging sources. However in a healthy hive the two curves should show similar matching tendency over the time period of our investigation. If a larger number of forager traffic traffic leaves the hive and does not come back then that could indicate a swarming event. And if the number of incoming forager traffic greatly increases as compared to the outgoing traffic then that can indicate a hive robbing event. Hence, continuous discrepancies between the incoming and outgoing forager traffic can be treated as a deviation from normal behavior and may prompt the beekeeper for manual inspection.

We have used the foraging data from the bee keeping season of 2018 for our analysis in this section. The BeePi system records video data every 15 minutes from 8:00 am to 9:00 pm. Thus we have 4 recordings every hour and in total we have 52 recordings for each day of the investigating time period. We will be focusing on two separate hives (R_4_5 and

R_4_7) to analyze the directional traffic. Both the hives are in the same apiary in Logan, UT, but are at least 30 feet apart. Although we had 4 hives in that apiary, we chose the above two because the BeePi sensors in hives R_4_5 and R_4_7 had the least number of hardware failures over the 2018 beekeeping season. This was also to make sure that we had good quality data for our analysis. R_4_5 was in sunlight mostly during the morning hours till noon. R_4_7 remained in shade most of the day and received sunlight during the early evening to sunset hours. Both the hives were located in an apiary where there are lots of trees and small bushes.

Next, we computed the values of I_v and O_v (Equation (5.3)) for each video to study how closely the counts of outgoing and incoming bee traffic match. In order to match the trend in the incoming and outgoing counts, we calculated the Pearson product-moment correlation coefficients between them using 'numpy' package in Python3.5. A higher correlation value would indicate that both the time series have identical trends, i.e. when one increases the other one increases as well. We also used Dynamic Time Warping (DTW) [98, 99] to calculate the similarity between the two time series of incoming and outgoing bee motion counts. DTW is a numerical similarity measure of how two time series can be optimally aligned (i.e., *warped*) in a way such that the accumulated alignment cost is minimized. It is based on dynamic programming and it finds all possible alignment paths and selects a path with a minimum cost. We randomly chose 2 days from each month from the beginning to the middle of the bee keeping season of 2018 (May–August) from the hives R_4_5 and R_4_7, and showed how the time series plots of incoming and outgoing bee motion counts matched up. We chose the above months since during that time period we saw both hives grew stronger, which had also prompted us to add more supers to our Langstroth bee hives. Thus in total we investigated 832 (4*13*8*2) videos from both the hives. Figure 5.9 shows the time series plots for incoming and outgoing bee motion counts for hive R_4_5 and Figure 5.10 shows the time series plots for incoming and outgoing bee motion counts for hive R_4_7. From both the graphs we can see that the incoming time series curve matches closely to the outgoing time series plots. We can also see that when there is a spike (small or

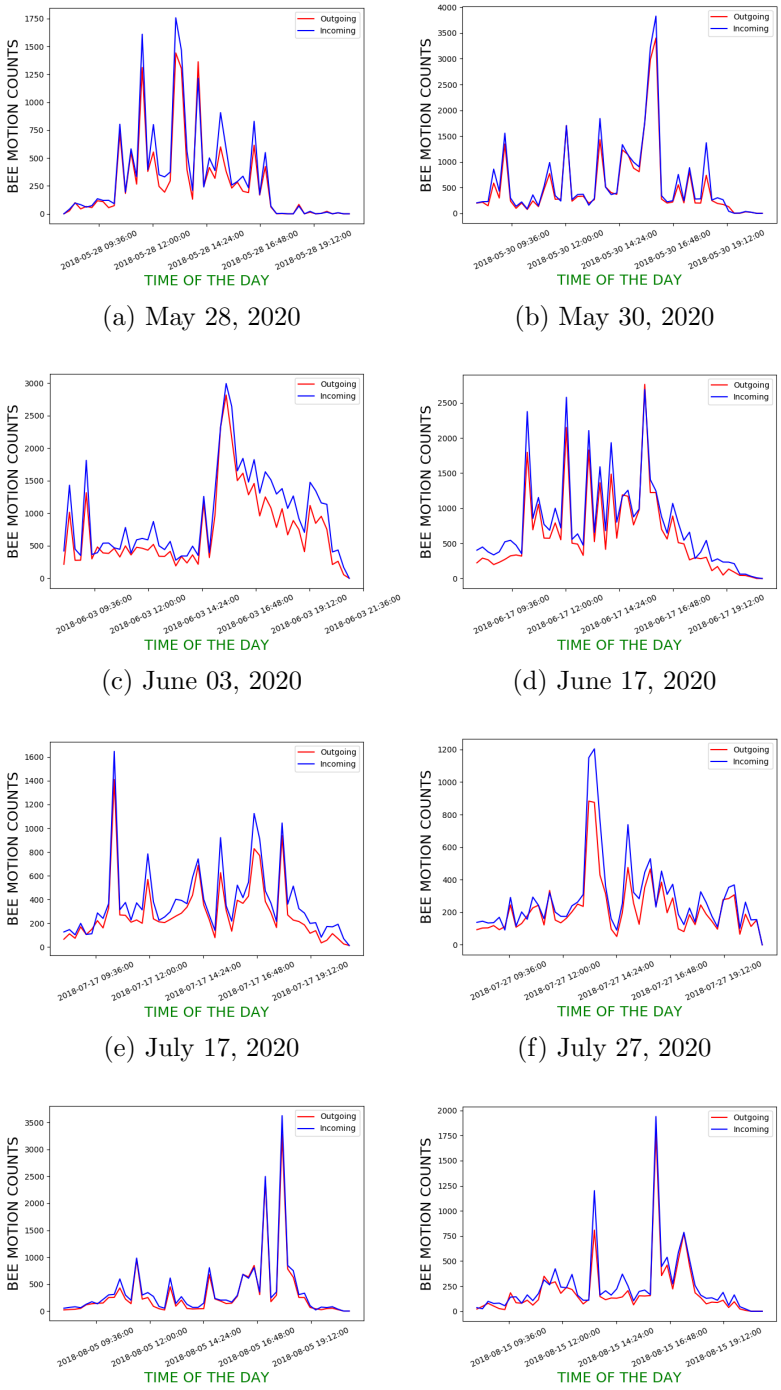
large) in the outgoing curves, it is also reflected in the incoming curves and vice versa. The above similarity is also reflected in the DTW and correlation coefficient values in Tables 5.6 and 5.7 respectively. A low value of DTW suggests both the time series are closely aligned. From our past research [2], we have seen a DTW value of less than 10 is usually good and we can see that all the DTW values are below 10 in Table 5.6. On the other hand a high correlation value suggests that as one variable increases in its values, the other variable also increases in its values via an exact linear rule. We can see that all the correlation values are very high and close to +1 in Table 5.7. This tells us that there is a strong positive correlation between the incoming and outgoing traffic during the above days we investigated.

	May 28	May 30	June 03	June 17	July 17	July 27	Aug 05	Aug 15
R_4_5	5.42	4.41	7.5	6.52	7.38	8.5	3.01	5.89
R_4_7	4.17	3.53	5.72	2.89	7.6	8.25	6.45	4.59

Table 5.6: DTW similarity scores between outgoing and incoming traffic curves for the time series plots in Figures 5.9 and 5.10. A lower DTW value is desirable.

	May 28	May 30	June 03	June 17	July 17	July 27	Aug 05	Aug 15
R_4_5	0.9835	0.9899	0.9774	0.9821	0.9805	0.9717	0.9963	0.9812
R_4_7	0.9902	0.9954	0.9887	0.9971	0.9835	0.9762	0.9866	0.9918

Table 5.7: Correlation coefficient between outgoing and incoming traffic curves for the time series plots in Figures 5.9 and 5.10. A higher correlation coefficient value is desirable.



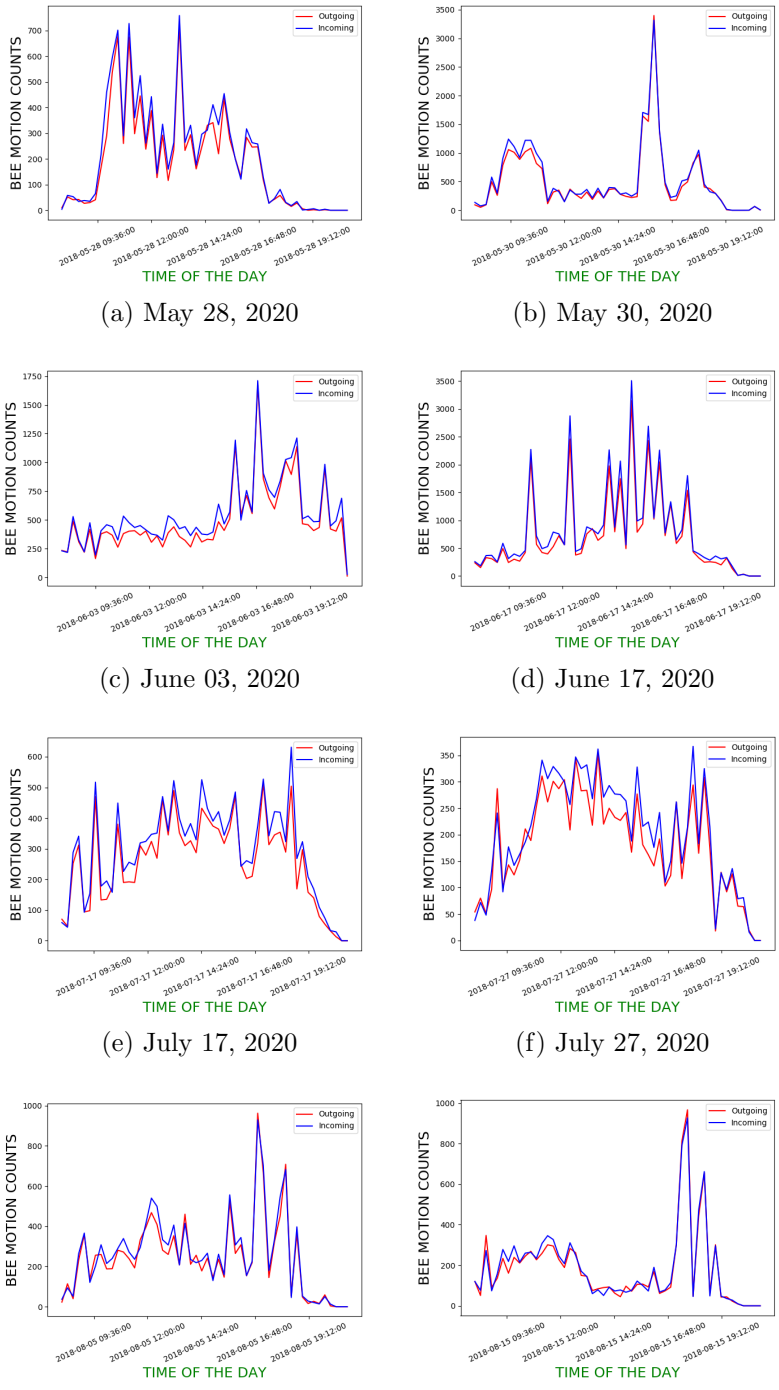
(a) May 28, 2020 (b) May 30, 2020

(c) June 03, 2020 (d) June 17, 2020

(e) July 17, 2020 (f) July 27, 2020

(g) Aug 05, 2020 (h) Aug 15, 2020

Figure 5.9: The time series plots for incoming and outgoing bee traffic for Hive R_4_5 during the certain days in May, June, July and August 2018.



(a) May 28, 2020 (b) May 30, 2020

(c) June 03, 2020 (d) June 17, 2020

(e) July 17, 2020 (f) July 27, 2020

(g) Aug 05, 2020 (h) Aug 15, 2020

Figure 5.10: The time series plots for incoming and outgoing bee traffic for Hive R_4_7 during the certain days in May, June, July and August 2018.

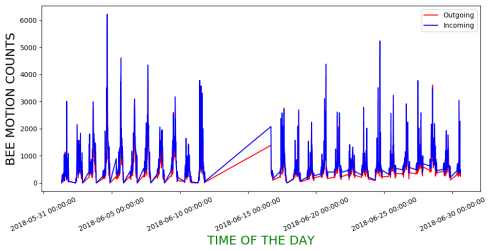
We also investigated the time series plots for the incoming and outgoing bee traffic for the months of June–August during the days video recording data was available. We evaluated the curves by calculating the average DTW score per day and the correlation coefficient.

	June	July	August
R_4_5	4.89	7.34	4.66
R_4_7	2.99	3.38	3.33

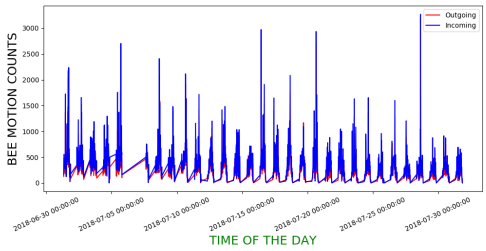
Table 5.8: Average DTW similarity scores between outgoing and incoming traffic curves for the time series plots in Figures 5.11 and 5.12 for the months of June, July and August.

	June	July	August
R_4_5	0.9843	0.9727	0.9761
R_4_7	0.9932	0.9953	0.9884

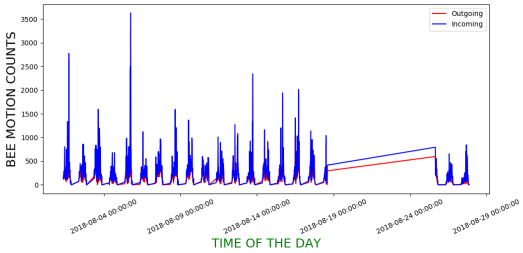
Table 5.9: Correlation coefficient between outgoing and incoming traffic curves for the time series plots in Figures 5.11 and 5.12 for the months of June, July and August.



(a) June 2020

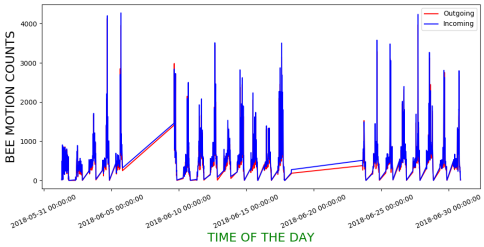


(b) July 2020

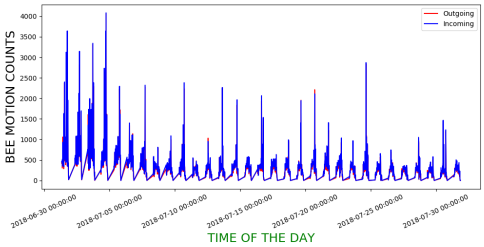


(c) August 2020

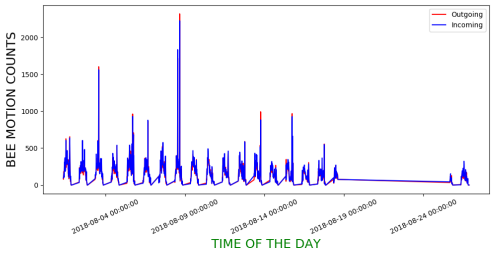
Figure 5.11: The time series plots for incoming and outgoing bee traffic for hive R_4_5 during the months of June, July and August 2018. The flat lines on the curve is during the time frame when there were hardware failures.



(a) June 2020



(b) July 2020



(c) August 2020

Figure 5.12: The time series plots for incoming and outgoing bee traffic for hive R_4_7 during the months of June, July and August 2018. The flat lines on the curve is during the time frame when there were hardware failures.

In Table 5.8, we can see that the average DTW score between the incoming and outgoing bee motion counts for the months of June, July and August for hives R_4_5 and R_4_5 were all below 10. We can also see the high correlation values in Table 5.9 for the above months for both the hives. Both of the above events of low DTW scores and high correlation coefficients tell us that the incoming and outgoing curves match closely with each other. We can see the match being reflected in the graphs in Figures 5.11 and 5.12 as well. If we observe Table 5.8, we see that the DTW score corresponding to hive R_4_5 was comparatively higher than the one for R_4_7. If we compare the corresponding graphs on Figure 5.11b and 5.12b, we can see that certain spikes in the incoming traffic have not been replicated by a similar spike in outgoing traffic. This could be the case where in certain video recordings, we observed that the bees were flying out too fast from the beehive for dpiv to be able to detect any bee movement in the videos captured by the raspberry pi camera. Hence we can assume the above event must have caused the mismatch between the incoming and the outgoing bee motion counts. Overall we can see that the dpiv based bee motion counts

method presented in this chapter can be used to measure not only omnidirectional but also directional honeybee traffic.

5.6 Seasonal Bee Motion Counts

In this section, we will plot the total bee motion counts ($T_v(V)$) for the entire beekeeping season of 2018 from May to November for hives R_4_5 and R_4_7, and then try to compare and investigate them in order to draw a conclusion regarding hive development. To perform the above comparison, we first prepared the data accordingly. Rather than using bee motion counts for every hour of a day, we calculated the mean of the bee motion counts for the entire day. So rather than having 15 bee motion count values for the day (6am to 9pm), we just have 1 value for the entire day. After preparing the data, we plotted the average daily bee motion counts for hives R_4_5 and R_4_7 from May to November. Figure 5.13 shows the plot for the average daily bee motion counts over the entire beekeeping season of 2018 for both hives R_4_5 and R_4_7. If we observe the Figure 5.13 we can see that the hives R_4_5 was performing on par with hive R_4_7 during certain parts of the 2018 beekeeping season, specifically at the beginning and during the middle (mid July–mid August) time period.

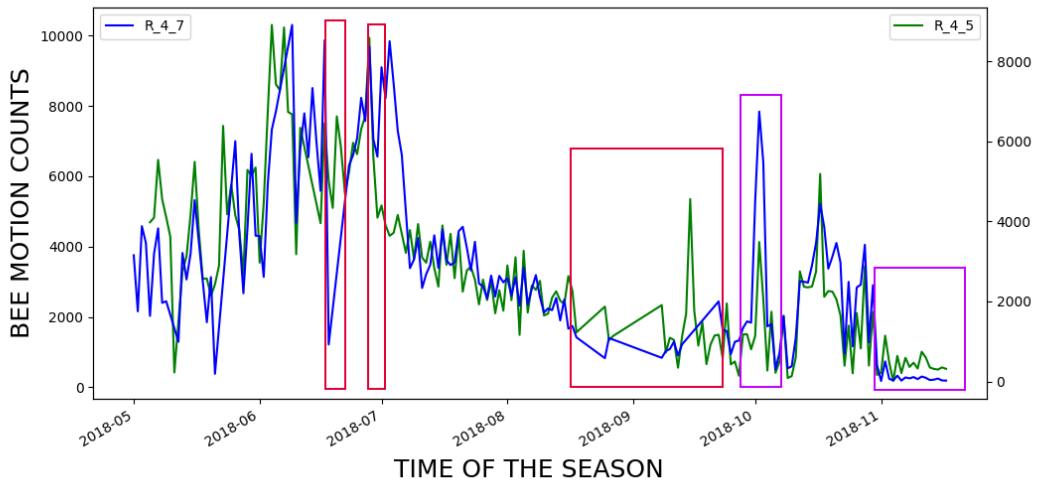


Figure 5.13: Bee motion counts for the entire season of 2018. Red boxes signify time period where there were hardware failures and no data was recorded.

But if we look closely we see that both the hives showed a downward trend based upon the bee motion counts. This could very well serve as an early detection alarm to understand the development of hives. In a large-scale apiary with many hives equipped with BeePi sensors, those hives that show a downward trend in performance during the middle of the season could be manually inspected and necessary steps for hive treatment could be performed. In Figure 5.13, three time periods have been marked with red boxes. During those time periods our BeePi monitors experienced hardware failures for either or both of the hives and thus no data was available for comparison. The magenta boxes are marked for time periods which are important to our analysis. In the first magenta box, we see that there is a sudden increase in bee traffic for hive R_4_7. We think this event is significant since up until that time period the two hives showed similar bee traffic. This event could signify colony robbing activity or it could also signify stress encountered by hive R_4_7 due to some other events. The final magenta box is during the timeline of end October–end November, which are months in the back end of the beekeeping season. The temperature falls below freezing occasionally during this time of the year. This also suggests that there will be less bee movements and then less bee motions would be detected. We can see that effect for both the hives R_4_5 and R_4_7. But the amount of change or variability in the graph for R_4_5 suggests that there were still some bee movements during that time, which could suggest that the hive was preparing for the winter. On the other hand, the graph for R_4_7 showed very little variability which also means there were a lot fewer bee movements. It is hard to explain such events, but we think it could mean the hive R_4_7 was struggling. It is also evident from our beekeeping journal, from which we know that hive R_4_7 was not able to survive the harsh winter weather in Logan, UT.

5.7 Difference Between Incoming And Outgoing Bee Motion Counts

In this section, we will plot the absolute difference between the incoming and outgoing bee motion counts ($D_v(V) = |I_v(V) - O_v(V)|$) for the entire beekeeping season of 2018 from May to November for hives R_4_5 and R_4_7, and investigate them to draw a conclusion regarding hive development and health of the hive. In healthy beehives, ideally the levels

of incoming bee traffic should approximately match levels of outgoing bee traffic, because all foragers that leave the hive should eventually come back. There might be occasional mismatches on certain days because not all foragers return back to the hive at the same time due to flying to different foraging sources. But the overall matching tendency should be visually apparent if the incoming and outgoing motion counts are treated as time series. But consistent deviations in the difference between the incoming and outgoing bee counts may indicate that the hive may not be healthy and may be in stress.

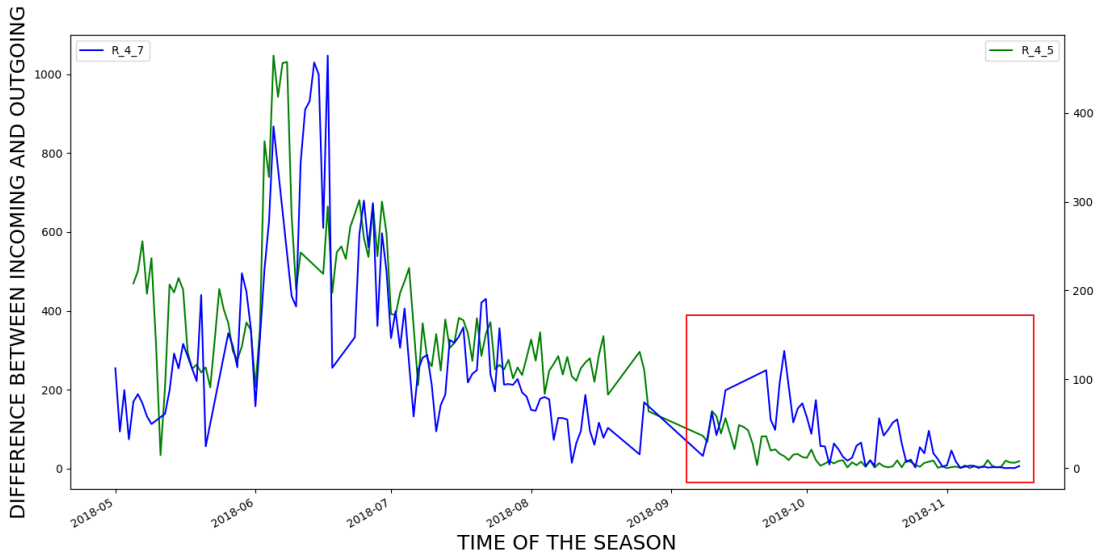


Figure 5.14: Absolute difference between the incoming and outgoing bee motion counts for the entire season of 2018. The red box signifies the time period of interest.

In Figure 5.14, we can see the difference between incoming and outgoing bee motion counts during different times of the season. During majority of the season we can see that there is similar difference between the corresponding counts for both the hives R_4_5 and R_4_7, which tells us that both the hives showed similar development. We see that the difference $D_v(V)$ is higher during the middle of the season when there is lot of bee movements and as the season progresses the difference goes down. One reason for the high difference between the middle of the season could be attributed to the bee motion count algorithm missing some bees motions in high traffic videos. From the Figure 5.14 we can

see that the max difference is close to 1000 bees which means that during the middle of the season when there is high bee traffic, the incoming counts differs from the outgoing counts by approximately 1000 bees. Since the bee traffic level is really high during the the above timeline, we believe that a mean difference of 1000 bees per day is an acceptable amount.

In the later part of the season during the timeline of September–end November indicated by the red box Figure 5.14, we can see that the difference $D_v(V)$ varies a lot for hive R_4_7 whereas there is a constant difference in counts for hive R_4_5. The variable difference along with the sudden increases in the difference tell us that hive R_4_7 was struggling and was in stress towards the end of the season. The difference could be attributed to either lots of bees leaving the hive and not coming back or even due to loss of bees. This matches with our observations in our beekeeping journal where we recorded a lot more bee loss during this timeline for hive R_4_7 as compared to hive R_4_5. Collectively the above observations suggest that while hive R_4_5 might have been preparing for the upcoming winter, hive R_4_7 was struggling during the later end of the bee keeping season. The above is also evident from our beekeeping journal, from which we know that hive R_4_7 was not able to survive the harsh winter weather in Logan, UT. The above analysis tells us that a carefully designed metric that could track this continuous discrepancies over time would help a beekeeper identify abnormal behaviour such as bees abandoning their hives, swarming or hive dying due to parasitic infestation or other hive threatening events.

5.8 Discussions

Our experiments in this chapter indicate that the combined dpiv based bee motion count algorithm along with bee movement localization performed better than the two-tier bee motion counting algorithm proposed in our previous research [1] in terms of omnidirectional bee motions based upon the results on 32 evaluation videos [96]. This current research also improved upon our previous study [2], where in we were able to get actual counts of bee motions rather than a general estimate. For the 32 evaluation videos we saw that the current combined dpiv method was able to generate results which were closer to the actual bee counts, which had been manually counted frame by frame from each video.

Thus through our results we have shown that dpiv can not only be used to quantify particle velocities but can also be used for analyzing the flow structure. We have also seen that our improved dpiv based proposed method is able to run on a single raspberry pi computer and is able to process each video on an average in 2.15 minutes. This is a significant improvement from our findings in our last research [2] where we needed 6 raspberry pis to process each video in 19 minutes. Thus using our new approach we are able to process each recorded video almost in 'real' time before the BeePi monitor records the next video in 15 minutes. This frees up a lot of time for the BeePi monitor to process and analyze data from other sensors as well. As the hardware in raspberry pis improves, we believe that the processing time of 2.15 minutes would go down even further. The above results are significant for citizen scientists in that the hardware base of the BeePi monitor can be easily replicated and the newly proposed dpiv based algorithm can operate just as fast on a single raspberry pi computer, thus requiring no significant capital and maintenance costs.

CHAPTER 6

ANALYSIS OF BEEHIVE AUDIO SAMPLES

6.1 Goal

Audio beehive monitoring helps us in the understanding of how bee buzzing could potentially determine the condition of the beehives and the environment around them. It can also be viewed as an emerging branch of ecoacoustics [100] that investigates the relationship between naturally occurring anthropogenic sounds and the corresponding environment. In this chapter we will discuss the design and compare different deep learning models towards classifying audio samples recorded by microphones on beehives. The methods presented in this chapter do not depend on the position of the microphones. The microphones can also be placed inside the hives provided proper precaution against microphone propolization is taken. We introduce two different datasets of audio samples to aid us in our investigation. In the first dataset BUZZ1 [23], the training and testing samples were separated from the validation samples by beehive and location. In the second dataset, BUZZ2 [23], the training and testing samples were separated from the validation samples by beehive, location, time and bee race.

6.2 Audio Data

The audio data used for experiments in this chapter have been collected from bee hive monitors from two different bee keeping season across two different apiaries. Figure 6.1 shows the relative position of the microphones above the landing pad on the beehive. Each monitor recorded 30 seconds of audio samples with an interval of 15 minutes. Each audio of 30 seconds was further segmented into 2-second samples with a 1 second overlap. Thus a 30 seconds audio resulted in 28 2-second audio samples. For our discussion we will work with audio data collected across two different bee keeping seasons. The first set of audio

(a) Bee hive in an apiary (b) Closeup image of microphones

Figure 6.1: Bee hive with microphones attached approximately 10cm above the landing pad. The microphones are not affected by rain or snow.

data was collected from 2 bee hives in an apiary (A1) deployed during May–July 2017 in Logan, UT. The beehives (1.1 and 1.2) in the first apiary (A1) in Logan, UT were close to a large parking lot. Thus some audio samples had sounds of car horns and engines. One of the beehive (1.1) was close to a garage with a power generator. Thus in the audio sample recorded from that beehive (1.1), the generator's sound was prominent sometimes. The second set of audio data was collected from beehives (1.3 and 1.4) with Carniolan bees deployed in the same apiary (A1) in Logan during the bee keeping season of 2018, during the months of May–July. Some recorded samples from beehives 1.1, 1.2, 1.3 and 1.4 also contained sounds of fire engine and ambulance sirens.

The second apiary (A2) was placed in a rural environment which was located in North Logan, UT. The two apiaries (A1 and A2) were approximately 3 miles apart. We collected data from two bee hives (2.1 and 2.2) in apiary A2. Beehive 2.1 was close to a lawn which was regularly mowed and watered by the property owner. Thus audio samples from beehive 2.1 had sounds of lawn mower, sprinklers, human conversation and children playing on the lawn. On the other hand bee hive 2.2 was located on the other end of the apiary where ambient noises were less audible.

Based on the audio samples collected, we labeled them into 3 different categories: bee buzz (`B`), cricket chirp (`C`) and ambient noises (`N`). The ambient noises category mostly contained sounds of thunder, wind, rain, vehicles, human conversation, sprinklers, relative

silence, i.e., absence of any sounds discernible to a human ear and also static noises from microphones. Audio samples in where we were not able to clearly listen to either bee buzz or cricket chirping were also included in the ambient noises (`N`) category. The class `B` is essential to determine different acoustic patterns in honey bee colonies, which could be used to identify various stressors affecting the bee colony. Class `C`, which represents cricket chirping is present during 11pm and 6am. Category `N` helps to identify and remove noisy samples which could hamper our analysis if included with the bee buzzing samples.

Our first dataset BUZZ1 [23], contained 10,260 audio samples. Out of which 9110 samples were used for training and testing. These samples were collected from beehives 1.1 and 2.1. The samples were divided as follows: 3000 samples of class `B`, 3000 samples of class `C` and 3000 samples of class `N`. The remaining 1150 audio samples in BUZZ1 were used as a validation dataset. These samples were obtained from beehives 1.2 and 2.2 during May–June, 2017. Thus the audio samples used for training and testing differed from the validation dataset by beehive and location. Although the training and testing data have a mixture of audio samples from beehives 1.1 and 2.1, it is worth noting that audio samples from different beehives differ from each other as their surroundings are different even if they are located in the same apiary.

Towards that end we created another dataset BUZZ2 [23] with 12,914 samples. In BUZZ2, audio samples in the training and testing dataset were completely separated with respect to the beehive and the location. The training set contained 7582 samples from beehive 1.1. It contained 2402 `B` audio samples, 3000 `C` audio samples, and 2180 `N` audio samples. On the other hand, the testing set contained 2332 audio samples from beehive 2.1. It contained 898 `B` audio samples, 500 `C` audio samples, and 934 `N` audio samples. To select the deep learning model that gives the best classification result, we set aside a validation dataset created from audio samples from beehives 1.3 and 1.4 deployed in apiary 1 during during the months of May–July 2018. The validation dataset included 1000 `B` audio samples, 1000 `C` audio samples, and 1000 `N` audio samples. As stated earlier, the beehives 1.3 and 1.4 contained fresh installation of Carniolan honey bees for the 2018 bee keeping season.

Thus in BUZZ2, the audio samples in the training and testing dataset were separated from each other by beehive and location; while the validation dataset differed from the training and testing dataset by beehive, time of collection (2017 vs 2018) and bee race, i.e. Italian honey bees vs Carniolan honey bees.

6.3 Deep Learning

Representation learning is a branch of AI that investigates automatic acquisition of representations for detection and classification from raw signals [77]. Standard ML techniques are limited in their ability to acquire representations from raw data, because they require considerable feature engineering to develop robust feature extractors that convert raw signals into feature vectors used in classification methods. Unlike conventional ML techniques, DL methods are designed to acquire multi-layer representations from raw data automatically. Starting from raw input, each layer is a transformation function that transforms its input to the one acceptable for the next layer. Many features of these layers are learned automatically by a general purpose procedure known as *backpropagation* [101]. Compositions of these transformations emulate complex classification functions. DL methods have been successfully applied to image classification [78–80], speech recognition and audio processing [81, 82], music classification and analysis [83–85],environmental sound classification [86] and bioinformatics [87, 88].

Convolutional Neural Networks (ConvNets) are a type of deep feedforward neural networks that have been shown in practice to better train and generalize than artificial neural networks (ANNs) on large quantities of digital data [102]. In ConvNets, filters of various sizes are convolved with a raw input signal to obtain a stack of filtered signals. This stack is referred to as the *convolution layer*. The size of the convolution layer is equal to the number of the convolution filters. The convolved signals are normalized. A standard choice for normalization is the rectified linear units (ReLUs) that convert all negative values to 0's [103, 104]. This layer is called the *normalization layer*. The size of each signal from the convolution layer or the normalization layer can be reduced in a *maxpooling layer*, where a window and a stride value are used to shift the window in stride steps across the input

signal retaining the maximum of the values from each window. The fourth type of layer in a ConvNet is the *fully connected layer* (FC), where the stack of processed signals is construed as a 1D vector where each value is connected via a *synapse* (a weighted connection that transmits a value from one unit to another) to each node in the next layer. FC layers can be stacked on top of each other before the top layer in the stack is connected to the nodes of the output layer. In addition to standard layers, ConvNets can have custom layers that are essentially arbitrary transformation functions that process the input from the previous layer before it is fed into the next layer. Standard and custom layers can be stacked arbitrarily deep in various permutations. The architectural features of a ConvNet that cannot be dynamically changed through backpropagation are called *hyperparameters* and include the number and size of filters in convolution layers, the window size and stride in maxpooling layers, and the number of neurons in FC layers. These permutations and hyperparameters distinguish one ConvNet architecture from another.

6.4 Convolutional Neural Network for Audio Spectrogram Classification

In audio processing, ConvNets are used by to classify raw audio or spectral features extracted from raw audio. For example, Piczak [86] used mel frequency cepstral coefficients (MFCCs) as one of the feature extraction methods for environmental sound classification. Aytar et al. [89] used raw audio samples to classify events from unlabeled videos. In [90], van den Oord et al. showed that generative networks were trained to predict the next sample in an audio sequence. The proposed network consisted of 60 layers and sampled raw audio at a rate of 16kHz to 48kHz.

In signal processing, a spectrogram represents an audio spectrum where time is represented along the x-axis and frequency of corresponding signal points along the y-axis and the strength or the amplitude of each frequency is represented by variable brightness or color. A spectrogram displays the changes in the frequencies in a signal over time. A spectrogram can also be represented as a RGB image as shown in Figure 6.2. The spectrogram in Figures 6.2, 6.3 and 6.4 was computed using the `specgram` function available from the matplotlib package in Python. Using the above function, the audio data is split into

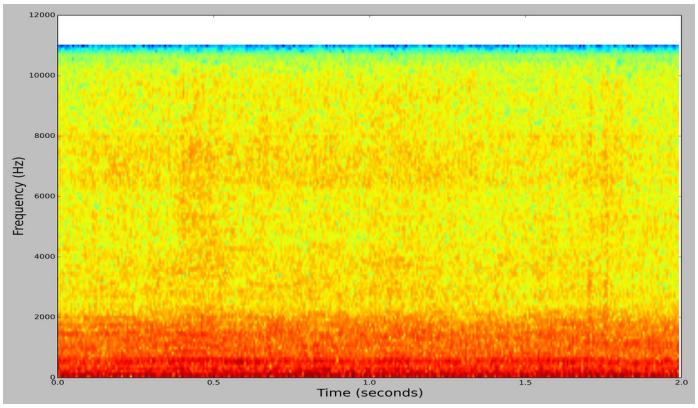


Figure 6.2: Audio Spectrogram of 2-second Audio Sample of Bee Buzzing

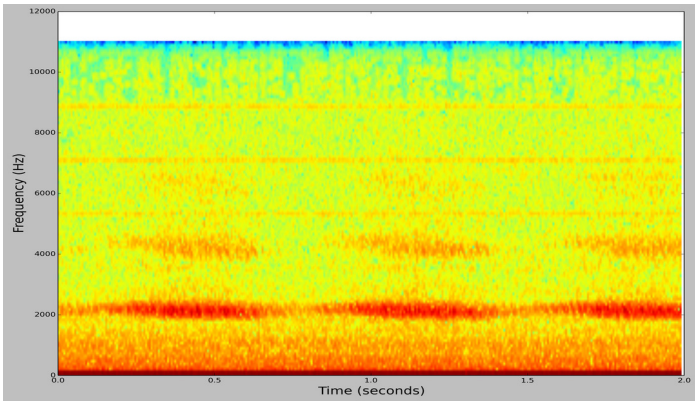


Figure 6.3: Audio Spectrogram of 2-second Audio Sample of Cricket Chirping

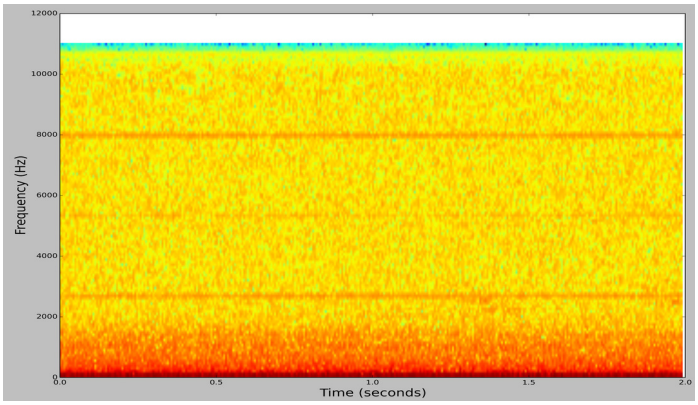


Figure 6.4: Audio Spectrogram of 2-second Audio Sample of Ambient Noise

NFF=512 length segments and the spectrum of each section is computed. Next the hanning windowing function is applied to each segment, and the amount of overlap between each segment is set to 384. The window size was set to 2048 along with a hop size of 512 (i.e. 2048/4 = 512) and the sampling frequency of 2. In order to treat the time and frequency domain equally and for faster processing, the spectrograms were resized to a 100x100 image. This resulted in each pixel value approximating a frequency value at a particular point along the time axis.

In this section, we will discuss a ConvNet architecture (SpectConvNet) that we have designed that takes as input RGB images of audio spectrograms. SpectConvNet is used to classify RGB images of audio spectrograms into 3 categories: bee buzz (`B`), cricket chirp (`C`) and ambient noises (`N`). Figures 6.2, 6.3 and 6.4 represents the spectrogram of audio samples of bee buzz (`B`), cricket chirp (`C`) and ambient noises (`N`) respectively represented as RGB images.

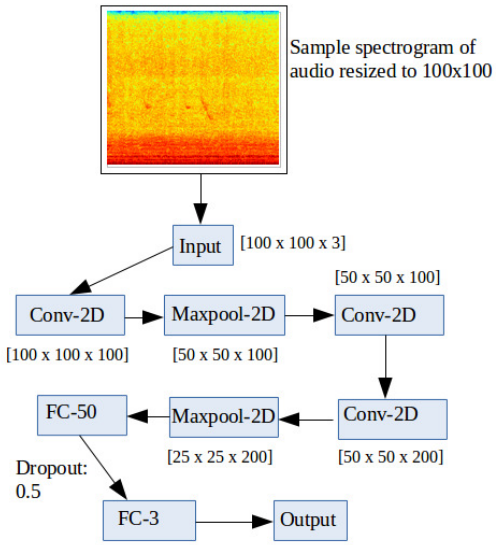


Figure 6.5: SpectConvNet Architecture. The numbers below each box, except for the input, denote the dimensions of the resultant feature map following the corresponding operation in the box.

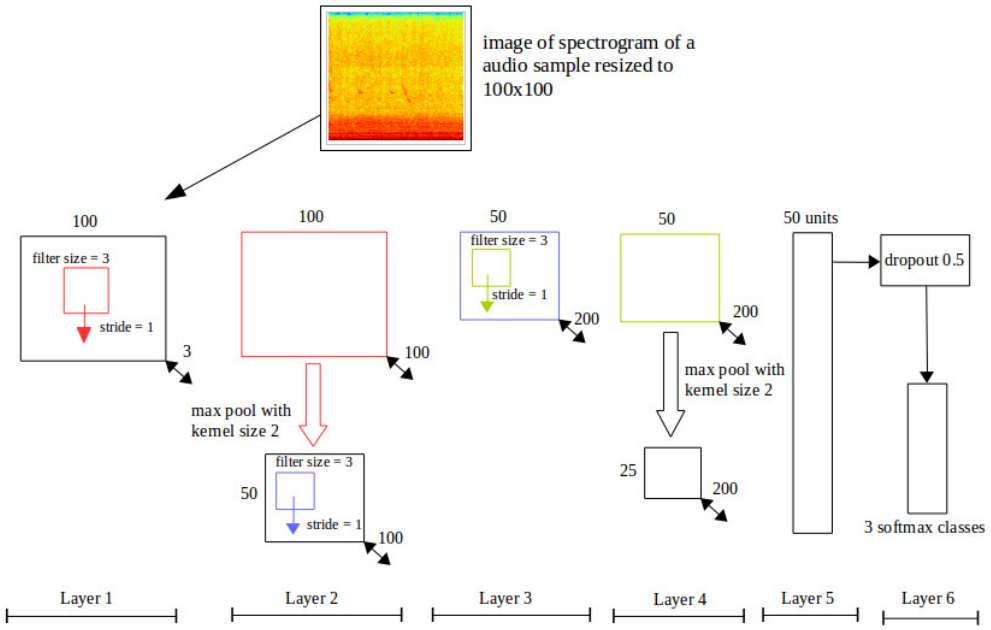


Figure 6.6: SpectConvNet Control Flow. Layer by layer control flow of SpectConvNet; the numbers at the corners of each rectangle represent the corresponding dimensions.

The architectural design of the SpectConvNet is shown in Figure 6.5 and the corresponding layer by layer arrangements is shown in Figure 6.6. SpectConvNet is trained using RGB images of dimensions 100x100 as in Figures 6.2, 6.3 and 6.4. SpectConvNet comprises of 3 convolutional layers and at each layer rectified linear units (ReLU) [104, 105] is used as the activation function. In layer 1, the input is convolved using 100 filters (filters shown in red in Figure 6.6), each of size 3×3, with a stride of 1 in both horizontal and vertical direction. This resulted in a set of feature maps of size $100 \times 100 \times 100$ shown in layer 2. Maxpooling is then applied to the feature maps with a kernel size of 2 and that results in a set of feature maps of size $50 \times 50 \times 100$. The resultant feature map is convolved using 200 filters (shown in blue in layer 2), each of size 3×3, with a stride of 1 in both horizontal and vertical direction. The output is a feature map of size $50 \times 50 \times 200$ (shown in blue in Layer 3). In Layer 3 another round of convolution is performed using 200 filters (shown in green), each of size 3×3, with a stride of 1. Maxpooling with a kernel size of 2 is then applied to generate a feature map of size $25 \times 25 \times 200$, as shown in layer 4. In layer 5, the

output is passed through a FC layer with 50 units. The FC layer (in layer 5 and also FC-50 in Figure 6.5) uses the ReLU activation function. Next in layer 6, a dropout layer [106] with a keep probability of 0.5 is added to avoid any overfitting. The addition of a dropout layer also reduces any complex co-adaptations of neurons [78]. The final FC layer (FC-3 in Figure 6.5) uses the softmax activation function [104, 105].

The softmax function is used to represent a probability distribution over k different possible outcomes by mapping a k-dimensional vector of arbitrary real values into another k-dimensional vector whose values are in the range (0, 1) that add up to 1. The softmax function emphasizes the largest values and de-emphasizes the smallest ones. Formally, if $\mathtt{X} = [x_1, ..., x_k]$ is a k-dimensional vector, then the softmax function maps each element x_i of $\mathtt{X}$ to the real value specified in Equation 6.1. The softmax function is used to transform a k-dimensional vector $\mathtt{X}$ to a k-dimensional vector $\mathtt{Z}$, as shown in Equation 6.2.

$$\mathrm{S}(\mathrm{x_i}) = \frac{e^{\mathrm{x_i}}}{\sum_{j=1}^{k} e^{\mathrm{x_j}}}, \text{ where } 1 \leq i \leq k. \tag{6.1}$$

$$\mathrm{Z} = [S(x_i), ..., S(x_k)]. \tag{6.2}$$

In multiclass classification, given a sample vector $\mathtt{X}$ and its score vector $\mathtt{Z}$, the probability of $\mathtt{X}$ belonging to class j is given in Equation 6.3, where $\mathtt{Z}_i$ is the i-th element of $\mathtt{Z}$.

$$P(\mathtt{X} = j|\mathtt{Z}) = \frac{e^{\mathrm{Z_j}}}{\sum_{j=1}^{k} e^{\mathrm{Z_j}}}. \tag{6.3}$$

Thus in layer 6, when the result is passed through a 3-way softmax function the SpectConvNet classifies the audio signal as bee buzz (`B`), cricket chirp (`C`), or ambient noise (`N`). We trained a several iterations of SpectConvNet with different choices of parameter values.

The detailed description of individual layers of the best performing SpectConvNet model is shown in Table 6.1.

Layers	**Operation**	**Specification**
Layer 1	Conv-2D	filters = 100, filterSize = 3, strides = 1, activation = `relu`, bias = `True`, biasInit = `zeros`, weightsInit = uniform scaling, regularizer = `none`, weightDecay = 0.001
Layer 2	Maxpool-2D	kernelSize = 2, strides = `none`
	Conv-2D	filters = 200, filterSize = 3, strides = 1, activation = `relu`, bias = `True`, biasInit = `zeros`, weightsInit = uniform scaling, regularizer = `none`, weightDecay = 0.001
Layer 3	Conv-2D	filters = 200, filterSize = 3, strides = 1, activation = `relu`, bias = `True`, biasInit = `zeros`, weightsInit = uniform scaling, regularizer = `none`, weightDecay = 0.001
Layer 4	Maxpool-2D	kernelSize = 2, strides = `none`
Layer 5	FC-50	number of units = 50, activation = `relu`
Layer 6	Dropout	keep probability = 0.5
	FC-3	number of units = 3, activation = `softmax`

Table 6.1: Layer Specification Of The Best Performing SpectConvNet Model

6.5 Convolutional Neural Network for Raw Audio Signal Classification

The feature engineering approach was the dominant approach till recently when deep learning techniques started demonstrating recognition performance better than the carefully crafted feature detectors. Deep learning shifts the burden of feature design also to the underlying learning system along with classification learning typical of earlier multiple layer neural network learning. From this perspective, a deep learning system is a fully trainable system beginning from raw input, for example image pixels, to the final output of recognized objects. The biggest advantage of Deep Learning is that we do not need to manually

extract features from the image. The network learns to extract features while training using its different convolution kernels. Towards that end we designed a second ConvNet architecture called RawConvNet which takes as input the waveforms of raw audio signal. The architecture of RawConvNet is shown in Figure 6.7 and the layer by layer arrangements of different operations is shown in Figure 6.8. The proposed RawConvNet can be generalized across any type of audio signal.

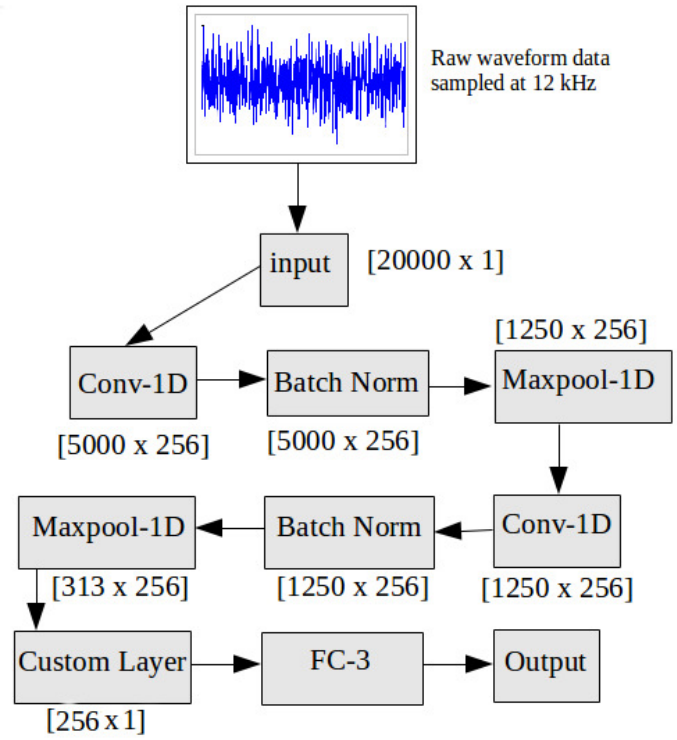


Figure 6.7: RawConvNet Architecture. The numbers below each box, except for the input, are the dimensions of the resultant feature map following the corresponding operation in the box.

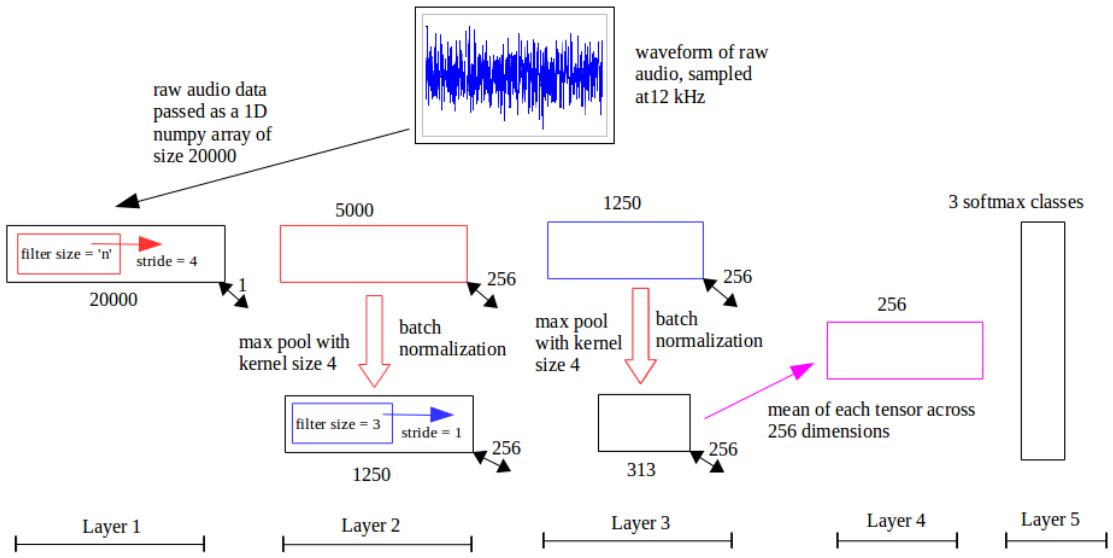


Figure 6.8: Different layers in RawConvNet trained to classify raw audio samples; the numbers below each box at the corners of each rectangle represent the corresponding dimensions; all filters and feature maps are rectangular in shape with a breadth of one unit.

The architecture of RawConvNet is comprised of two hidden layers both of which have 256 filters. In both the layers ReLu is used as the activation function. The input to the RawConvNet is a waveform of the raw audio signal. The audio signal is downsampled to 12KHz and is normalized to have a mean of 0 and variance of 1 before it passed on to the RawConvNet. The weights of the convolution or hidden layers are initialized using the methods described by Xavier et al. in [107]. The random initialization of the weights helps to keep the scale of the gradients roughly the same in both the layers. Finally the waveform of the normalized and downsampled audio signal is flattened to 20,000 $\times$ 1 and passed as an array tensor to the first layer of RawConvNet (refer to Figure 6.8). In Layer 1, the input is convolved with 256 filters (shown in red in Layer 1), with a stride of 4 in the horizontal direction each of size $n \times 1$, where $n \in \{3, 10, 30, 80, 100\}$. The effect of different sizes of receptive fields for the first convolution layer is explained in the later section while discussing the various experiments. The stride is chosen big in the first layer, to keep the number of trainable parameters less along with keeping the computation cost low. The corresponding 5000 $\times$ 256 feature map is shown in Layer 2 in red. Next a batch

normalization [108] operation is added to reduce the internal covariate shift.

Batch normalization is applied to the output of each convolution layer to reduce the computation costs and accelerate the learning process. Batch normalization forces the activations in feature maps to have a standard deviation of 1 and a mean of 0. The output of each convolution layer is a tensor of rank 4, i.e., $[B, H, W, D]$, where B is a batch size, $H \times W$ is a feature map size, and D is the number of channels or dimensions. If (x, y) is a spatial location in the feature map so that $0 \leq x < H$ and $0 \leq y < W$, then the basic batch normalization procedure computes $H \times W \times D$ means and $H \times W \times D$ standard deviations across B elements. The output of the batch normalization layer is controlled by two parameters that are learned to best represent the activation of the feature maps.

Following the normalization, maxpooling with kernel size of 4 is applied to the the resultant tensor and that results in a 1250×256 feature map. Convolution is then applied to the feature maps using 256 filters (shown in blue in Layer 2), each of size 3×1, with a stride of 1 in the horizontal direction. The above operation results in a 1250×256 feature map (shown in blue in Layer 3). Batch normalization is applied again to the resultant feature maps and then maxpooling with a kernel size of 4 is performed which results in a 313×256 feature map. In Layer 4, we introduce a custom layer that calculates the mean of the feature maps and returns a single value for each of the 256 dimensions present in the feature maps. This reduces each feature map tensor to a single real number. It results in a 256×1 tensor as shown in magenta color in layer 4. Finally, in Layer 5 the output is passed through a 3-way softmax function that classifies it as `B` (bee buzzing), `C` (cricket chirping), or `N` (noise). The detailed description of individual layers of the RawConvNet model is shown in Table 6.2.

Layers	Operation	Specification
Layer 1	Conv-1D	filters = 256, filterSize = $n \in \{3, 10, 30, 80, 100\}$, strides = 4, activation = `relu`, bias = `True`, weightsInit = `xavier`, biasInit = `zeros`, regularizer = `L2`, weightDecay = 0.0001
Layer 2	Batch Normalization	gamma = 1.0, epsilon = $1e-05$, decay = 0.9, stddev = 0.002
	Maxpool-1D	kernelSize = 4, strides = `none`
	Conv-1D	filters = 256, filterSize = 3, strides = 1, activation = `relu`, bias = `True`, weightsInit = `xavier`, biasInit = `zeros`, regularizer = `L2`, weightDecay = 0.0001
Layer 3	Batch Normalization	gamma = 1.0, epsilon = $1e-05$, decay = 0.9, stddev = 0.002
	Maxpool-1D	kernelSize = 4, strides = `none`
Layer 4	Custom Layer	calculates the individual mean of each feature map
Layer 5	FC-3	number of units = 3, activation = `softmax`

Table 6.2: Description Of Different Layers In RawConvNet

6.6 Experiments

We tested both of our designed ConvNets, i.e. SpectConvNet and RawConvNet on the two datasets BUZZ1 and BUZZ2, introduced and described in previous sections. We used Python 2.7 with tflearn [109] to train our ConvNet models on an Intel Core $i7$-4770@3.40GHz $\times$ 8 processor with 15.5 GiB of RAM and 64-bit Ubuntu 14.04 LTS. We used the Adam optimizer [110] during the training process. To analyze the performance of the models, we categorical crossentropy as the cost function. It measures the error probability in classification tasks where the classes are mutually exclusive. The best learning rate for SpectConvNet was $\eta = 0.001$ and for RawCovNet was $\eta = 0.0001$. Along with that each ConvNet model was trained for 100 epochs with a batch size of 128. Through our various it-

erations of the experiments we found the batch size of 128 to be the optimal value. Table 6.3 shows the number of trainable parameters used by the best performing SpectConvNet model and the RawConvNet with various receptive field sizes. Parameters are the weights that are learnt during the training process of the model. They are basically weight matrices that contribute to the model's predictive power, changed and updated during back-propagation process. The training algorithm we choose, particularly the optimization strategy makes them change their values.

Model Name	**Parameters (in Millions)**
SpectConvNet	6.79285
RawConvNet ($n = 3$)	0.198144
RawConvNet ($n = 10$)	0.199936
RawConvNet ($n = 30$)	0.205056
RawConvNet ($n = 80$)	0.217856
RawConvNet ($n = 100$)	0.222976

Table 6.3: Number of Learnable Parameters In The ConvNets.

6.6.1 Results on BUZZ1

We can recall from Section 6.2 that BUZZ1 comprised of 10,260 samples. From those samples we separated out our train/test dataset and our validation set for model validation. The train/test dataset consisted of 9,110 labelled audio samples from beehives 1.1 and 2.1. We used a 70-30 train/test split of the 9,110 audio samples, which resulted in 6,377 audio samples used for training (70% of 9,110 audio samples) and 2,733 audio samples used for testing (30% of 9,110 audio samples). The validation dataset consisted of 1,150 audio samples from 1.2 and 2.2. The results after training SpectConvNet and RawConvNet on BUZZ1 is shown in Table 6.4.

Number of Training Samples: 6,377**; Number of Testing Samples:** 2,733					
Model Name	Training Loss	Training Acc	Testing Loss	Testing Acc	Runtime per Epoch
SpectConvNet	0.00619	99.04%	0.00702	99.13%	690 secs
RawConvNet (n=3)	0.02759	99.24%	0.03046	98.87%	460 secs
RawConvNet (n=10)	0.01369	99.74%	0.01429	99.60%	462 secs
RawConvNet (n=30)	0.00827	99.91%	0.00679	99.71%	465 secs
RawConvNet (n=80)	0.00692	99.85%	0.00432	99.93%	462 secs
RawConvNet (n=100)	0.00456	99.97%	0.00785	99.74%	505 secs

Table 6.4: Performance Summary of ConvNet Architectures. The parameter n denotes the size of the receptive field in the first layer of RawConvNet.

From Table 6.4 we can see that RawConvNet with a receptive field size of 80 can classify the raw audio samples of bees, cricket, and noise with an accuracy above 99% along with a testing loss around 0.004. Table 6.4 indicates that the performance of RawConvNet increases as the size of the receptive field increases. This increase in accuracy can be attributed to the fact that as the size of the receptive field was increased in the first layer, more information about the audio signal was learned by our RawConvNet model which had a positive impact on the performance. RawConvNets with receptive fields sizes of 10 and above were able to outperform the SpectConvNet both in terms of testing accuracy (99.93% vs. 99.13%) and testing loss (0.00432 vs. 0.00702). Along with that, if we observe Table 6.4, we can see that it took less time per epoch to train the RawConvNet models compared to the SpectConvNet.

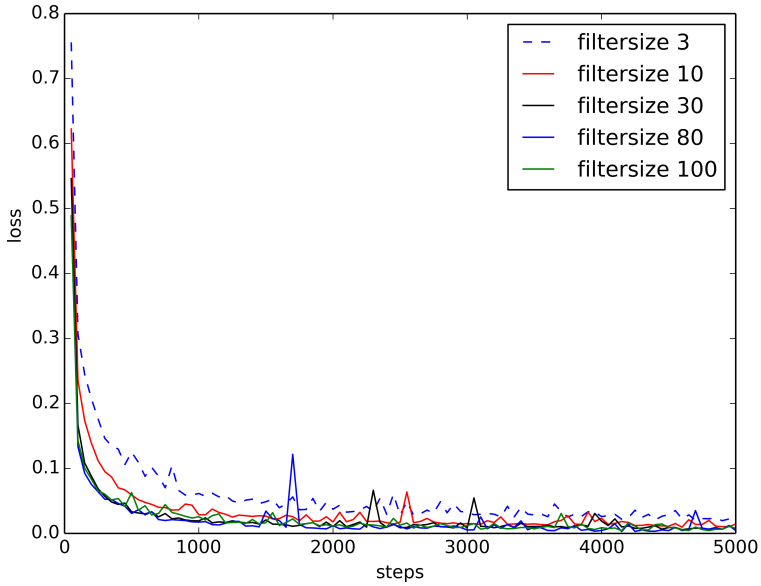


Figure 6.9: Loss Curves of RawConvNet on BUZZ1 Test Dataset. As the size of the receptive field increases, loss curves become smoother and decrease faster.

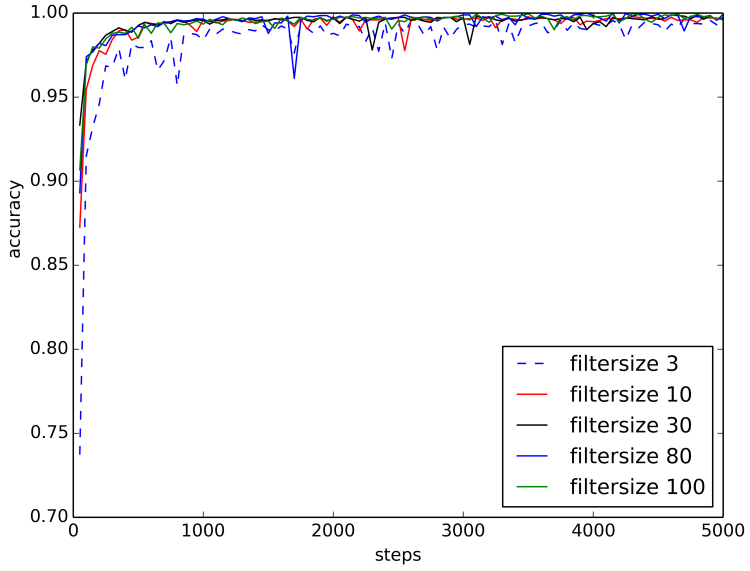


Figure 6.10: Accuracy Curves of RawConvNet on BUZZ1 Test Dataset. As the size of the receptive field increases, accuracy curves become smoother and increase faster.

The graphs in Figures 6.9 and 6.10 show the testing loss and testing accuracy graphs,

respectively, of RawConvNet with receptive field sizes $n \in \{3, 10, 30, 80, 100\}$ on the BUZZ1 testing dataset. The number of steps on the x-axis is a function of the number of samples in the testing dataset and the batch size (6,377/128 $\approx$ 50 steps per epoch). Since we train our models for 100 epochs, thus we have 5000 steps on the x-axes of the graphs in Figures 6.9 and 6.10.

6.6.2 Use of Custom Layer

In order to evaluate the contribution of the custom layer in the RawConvNet model (Figure 6.8), three additional deeper models (ConvNet1, ConvNet2, ConvNet3) were designed to classify raw audio samples where the custom layer was replaced with various combinations of FC and convolution layers. From Table 6.4, we see that the highest test accuracy was achieved when the receptive field size in the first layer was 80. Thus in ConvNet1, ConvNet2 and ConvNet3, the receptive field size in the first layer was set to 80.

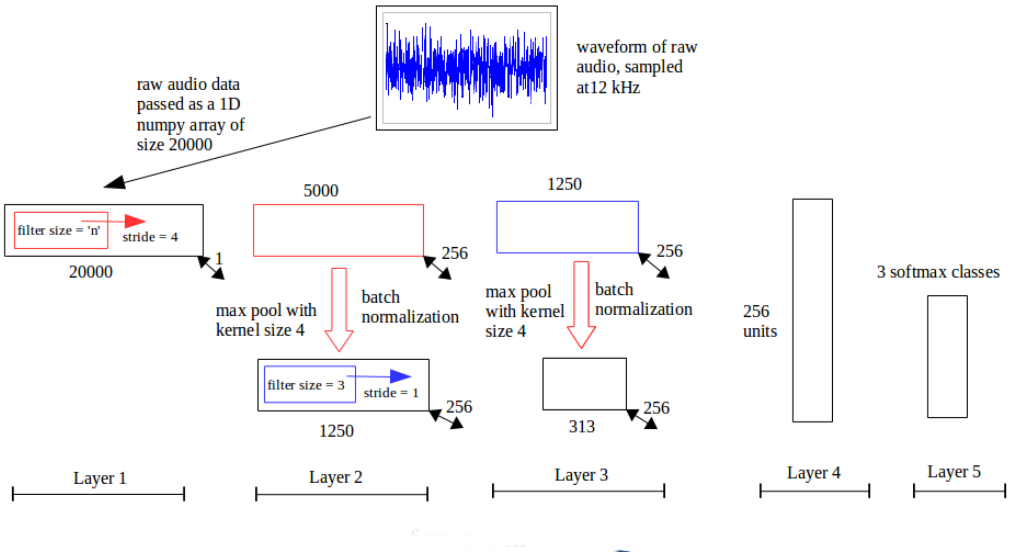


Figure 6.11: ConvNet1. This ConvNet is similar to RawConvNet, but the custom layer (layer 4) of RawConvNet is replaced with an FC layer with 256 neurons.

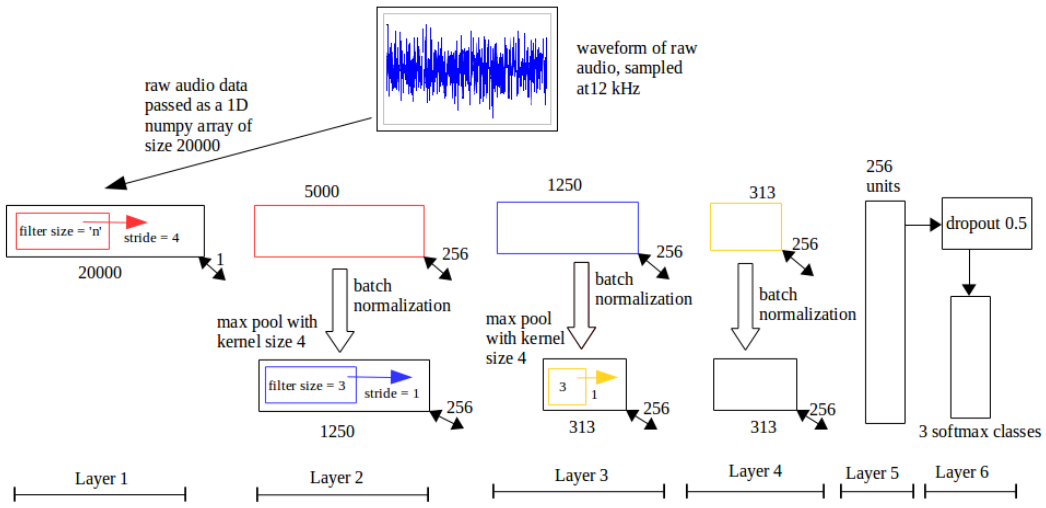


Figure 6.12: ConvNet2. Layers 1 and 2 are identical to ConvNet1; in layer 3, maxpooling output is convolved with 256 filters and batch normalization is used.

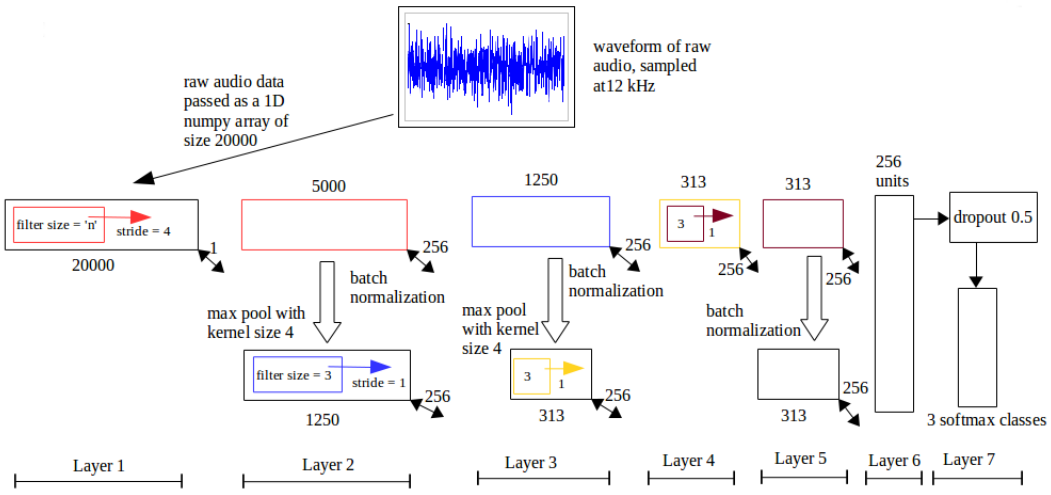


Figure 6.13: ConvNet3. Layers 1, 2, and 3 are identical to the same layers in ConvNet2; in layer 4, the output is convolved with 256 filters and batch normalization is performed in layer 5.

In ConvNet1, shown in Figure 6.11, the first three layers and the fifth layer are identical to the RawConvNet model (Figure 6.8) whereas the fourth layer is replaced with an FC layer with 256 neurons. In ConvNet2, shown in Figure 6.12, the first two layers are identical to the first two layers of the RawConvNet but in the third layer of Figure 6.8, a convolution

operation is added to the output of the maxpooling operation. The convolution layer is comprised of 256 filters, with 3 being the size of filter along with a stride of 1. The resultant size of the feature maps is $313 \times 1 \times 256$. Next batch normalization operation is applied in layer 4 and the resultant feature map is passed to an FC layer with 256. A dropout of 0.5 is added in layer 6 and the resulting tensor is passed to a FC-3 softmax layer. In ConvNet 3, shown in Figure 6.13, the first three layers are identical to the three layers of ConvNet2 (Figure 6.12). In layer 4, an additional convolution layer with 256 filters is added. Each filter size is 3 and the stride is 1. In layer 5, batch normalization is performed on the output of layer 4 before passing it to layer 6 that consists of an FC layer with 256 neurons. Layer 7 is the same as in ConvNets 1 and 2 comprising of a softmax layer.

Number of Training Samples: $6,377$**; Number of Testing Samples:** $2,733$					
Model Name	Training Loss	Training Accuracy	Testing Loss	Testing Accuracy	Runtime per Epoch
RawConvNet	0.00692	99.85%	0.00432	99.93%	462 s
ConvNet 1	0.55387	99.77%	0.57427	97.59%	545 s
ConvNet 2	0.67686	68.07%	0.59022	98.21%	532 s
ConvNet 3	0.68694	66.81%	0.59429	98.02%	610 s

Table 6.5: Contribution of Custom Layer. RawConvNet is compared with ConvNets1, ConvNets2 and ConvNets3 after 100 epochs of training on BUZZ1 train/test dataset. The receptive field size n in the first layer is set to 80 for all ConvNets.

Table 6.5 shows the usefulness and contribution of the custom layer in RawConvNet (Figure 6.8) by comparing the testing loss and accuracy results with ConvNet1 (see Figure 6.11), ConvNet2 (see Figure 6.12), and ConvNet3 (see Figure 6.13). From the results we can see that although the testing accuracies of ConvNets1, ConvNets2 and ConvNets3 are slightly lower than the testing accuracy of RawConvNet, but ConvNets1, ConvNets2 and ConvNets3 show comparatively much higher losses. Thus we can say that the addition of custom layer in RawConvNet (Figure 6.8) improved the performance of our proposed model.

It is important to note the gap between the training and testing accuracies for ConvNets2 (68.07% vs. 98.21%) and ConvNets3 (66.81% vs. 98.02%). This gap can be caused by underfitting which could be improved by increasing the number of epochs. It might also be the case for gradients being stuck at local optima position. The following section helps us understand more clearly the better performance of RawConvNet (Figure 6.8) by analyzing the gradient distributions for ConvNet1, ConvNet2, ConvNet3 and RawConvNet.

6.6.3 Effect of Custom Layer On Gradient Distribution

We analyzed the contribution of the custom layer in RawConvNet (Figure 6.8) by generating the gradient weight distributions plots for the final FC layers of RawConvNet, ConvNet1 (Figure 6.11), ConvNet2 (Figure 6.12), and ConvNet3 (Figure 6.13). We used *TensorBoard* [111] to generate the necessary plots. The plots are given in Figures 6.14, 6.15, 6.16 and 6.17. These plots consist of the time stamped histograms of gradients for ConvNet1, ConvNet2, ConvNet3 and RawConvNet, respectively. Each figure shows temporal slices of data over different steps during training with each slice being a gradient weight histogram in the FC softmax layer of an appropriate ConvNet with the oldest time slice in the back and the most recent one in the front. In all figures, the y-axis represents the step count during training and the x-axis represents the histogram bins. The step count on the y-axis starts from the back and moves to the front with the training process.

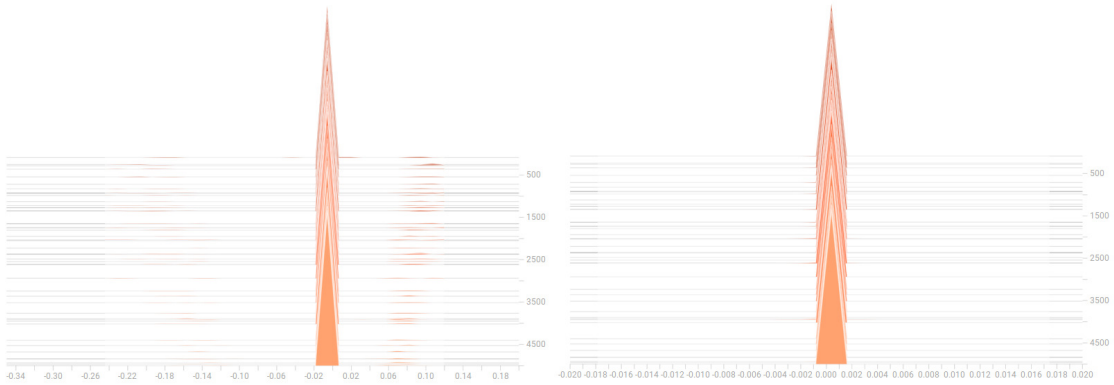


Figure 6.14: Histograms of the gradients for layers 4 and 5 in ConvNet1. The histograms on the left show the gradient distribution in the FC softmax layer in layer 4; the histograms on the right show the gradient distribution for the FC softmax layer in layer 5.

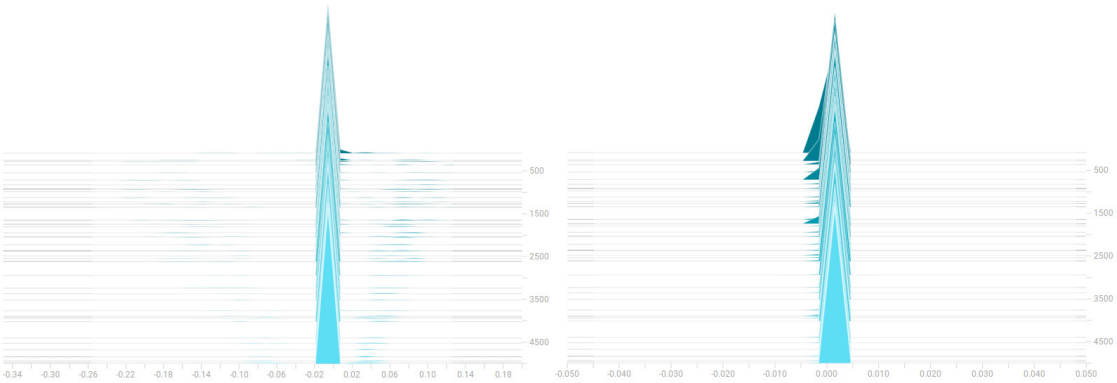


Figure 6.15: Histograms of the gradients for layers 5 and 6 in ConvNet2. The histograms on the left show the distribution of gradients for the FC softmax layer in layer 5; the histograms on the right show the gradient distribution for the FC softmax layer in layer 6.

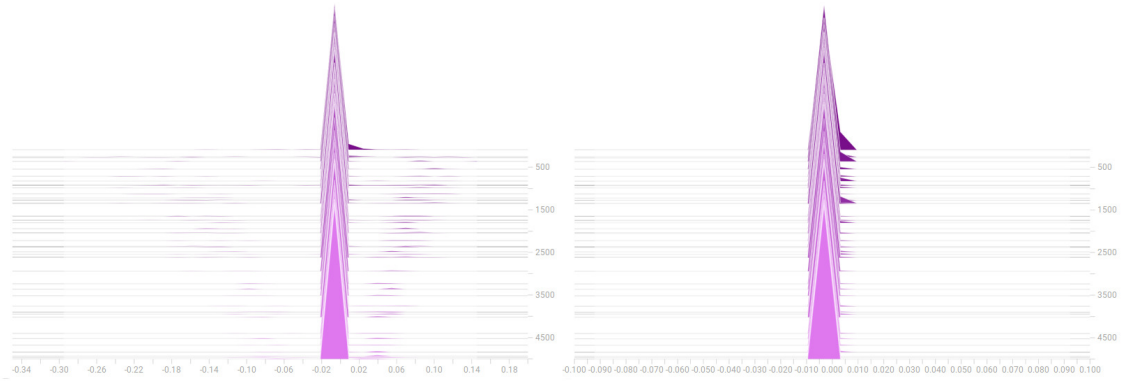


Figure 6.16: Histograms of the gradients for layers 6 and 7 in ConvNet3. The histograms on the left show the distribution of the gradients for the FC softmax layer in layer 6; the histograms on the right show the gradient distribution for the FC softmax layer in layer 7.

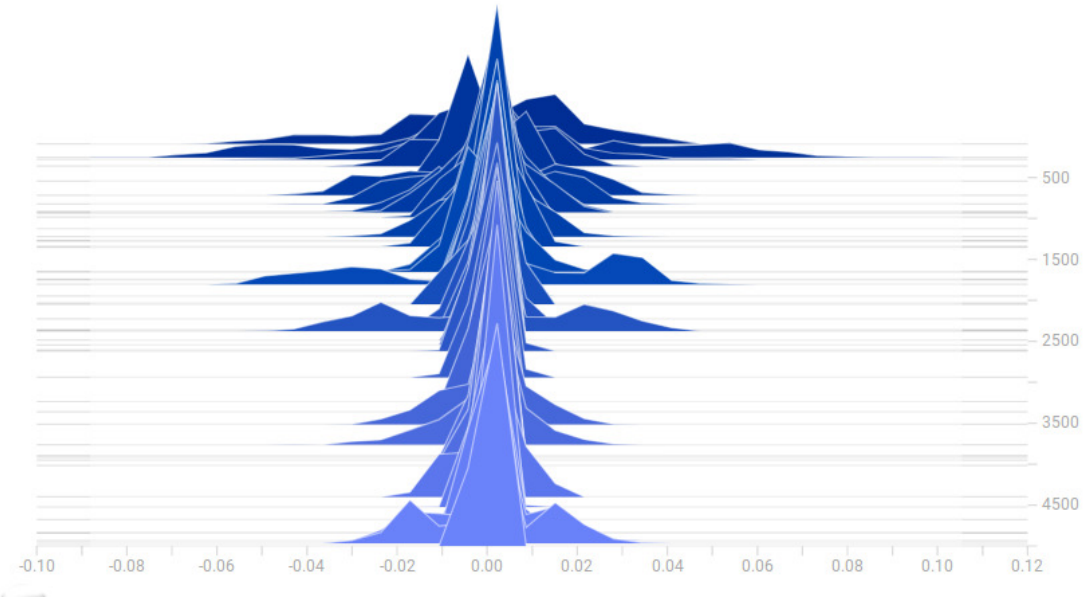


Figure 6.17: Histograms of the gradients in the FC softmax layer in layer 5 in RawConvNet.

As can be seen in Figures 6.14, 6.15, and 6.16, ConvNets1, ConvNets2 and ConvNets3 did not learn much in their FC layers as the shapes of the curves remain almost identical between the consecutive histograms, which suggests that the gradients in the FC softmax layers of these ConvNets were changing rather slowly. In ConvNets, such gradients are used

to update the weights in a way that minimizes the cost function during backpropagation. Mathematically, the weight update is modeled by Equation 6.4, where w_i is the weight matrix between the layers $i-1$ and $i+1$, and w_i^* is the updated weight matrix. The term $\partial loss/\partial w_i$ is the gradient of the corresponding weight matrix w_i.

$$w_i^* = w_i - \eta \cdot \frac{\partial loss}{\partial w_i}, \text{ where } \eta \text{ is the learning rate} \tag{6.4}$$

From Equation 6.4 and the figures one can observe that since the spread of the weight gradients is small, it suggests that the networks for the ConvNet models ConvNets1, ConvNets2 and ConvNets3 were learning slowly as weight updates w_i^* are rather small. Consequently, these networks made fewer correct predictions for the testing data. Specifically, Table 6.5 shows that the testing losses of ConvNets1, ConvNets2 and ConvNets3 were higher than the testing losses of RawConvNet although the accuracies of all four ConvNets were comparatively high. The graph in Figure 6.17 shows that the histogram of the weight gradient in the FC softmax layer of RawConvNet is more distributed over the entire training timeline. In RawConvNet, the gradients changed over different steps in the learning process and were much more spread than the gradients of ConvNets1, ConvNets2 and ConvNets3. Thus we can say that the RawConvNet model was learning continuously during the training process and in effect made more correct predictions for the testing data.

6.6.4 Results On BUZZ1 Validation Dataset

We can recall from Section 6.2 that BUZZ1 validation dataset comprised of 1150 samples out of which 300 were from class `B`, 350 from class `N` and 500 from class `C` respectively. The validation dataset consisted of data from a beehives 1.2 and 2.2 and thus differed from the training and testing dataset by both beehive and location. We tested our models ConvNet1, ConvNet2, ConvNet3 and RawConvNet on the above validation dataset.

True	***Predicted***				
	Bee	**Noise**	**Cricket**	**Total**	**Validation Accuracy**
Bee	300	0	0	300	100%
Noise	30	320	0	350	91.42%
Cricket	1	55	444	500	88.88%
Total Accuracy					92.52%

Table 6.6: Confusion matrix for RawConvNet with receptive field size n=3 on BUZZ1 validation dataset

True	***Predicted***				
	Bee	**Noise**	**Cricket**	**Total**	**Validation Accuracy**
Bee	300	0	0	300	100%
Noise	4	346	0	350	98.85%
Cricket	0	101	399	500	79.80%
Total Accuracy					90.86%

Table 6.7: Confusion matrix for RawConvNet with receptive field size n=10 on BUZZ1 validation dataset

True	***Predicted***				
	Bee	**Noise**	**Cricket**	**Total**	**Validation Accuracy**
Bee	300	0	0	300	100%
Noise	9	341	0	350	97.42%
Cricket	0	59	441	500	88.20%
Total Accuracy					94.08%

Table 6.8: Confusion matrix for RawConvNet with receptive field size n=30 on BUZZ1 validation dataset

True	***Predicted***				
	Bee	**Noise**	**Cricket**	**Total**	**Validation Accuracy**
Bee	300	0	0	300	100%
Noise	7	343	0	350	97.71%
Cricket	0	48	452	500	90.04%
Total Accuracy					95.21%

Table 6.9: Confusion matrix for RawConvNet with receptive field size n=80 on BUZZ1 validation dataset

True	***Predicted***				
	Bee	**Noise**	**Cricket**	**Total**	**Validation Accuracy**
Bee	300	0	0	300	100%
Noise	7	343	0	350	97.71%
Cricket	0	66	434	500	86.80%
Total Accuracy					93.65%

Table 6.10: Confusion matrix for RawConvNet with receptive field size n=100 on BUZZ1 validation dataset

True	***Predicted***				
	Bee	**Noise**	**Cricket**	**Total**	**Validation Accuracy**
Bee	297	3	0	300	99.00%
Noise	250	100	0	350	28.57%
Cricket	29	17	454	500	90.80%
Total Accuracy					74.00%

Table 6.11: Confusion matrix for ConvNet1 on BUZZ1 validation dataset.

True	***Predicted***				
	Bee	**Noise**	**Cricket**	**Total**	**Validation Accuracy**
Bee	297	3	0	300	99.00%
Noise	183	162	1	350	46.28%
Cricket	28	33	439	500	87.80%
Total Accuracy					78.08%

Table 6.12: Confusion matrix for ConvNet2 on BUZZ1 validation dataset.

True	***Predicted***				
	Bee	**Noise**	**Cricket**	**Total**	**Validation Accuracy**
Bee	300	0	0	300	100%
Noise	221	123	2	350	35.14%
Cricket	35	25	440	500	88.00%
Total Accuracy					75.04%

Table 6.13: Confusion matrix for ConvNet3 on BUZZ1 validation dataset.

In order to evaluate our deep learning models, we used the following standard machine learning models: Logistic regression [112]; k-nearest neighbors (KNN) [113]; support vector machine with a linear kernel one vs. rest (SVM OVR) classification [114]; and random forests [115] . All of the above methods falls into the category of supervised learning which are trained on previously known data labels and classifies new observations into one of the predefined data labels. We trained all four models on the same feature vectors extracted from the raw audio files. A detailed description about the feature extraction and feature engineering is available in [21]. In Table 6.14, the performance of the ConvNets and the four machine learning models on BUZZZ1 validate set is presented. From the results we can say that on BUZZ1 validation dataset, where the data was separated from the train/test dataset by beehive and location, RawConvNet performed the best. Although the accuracy in case of

the machine learning models were lower than the RawConvNet, but they performed better than the three deeper ConvNets (ConvNet1, ConvNet2 and ConvNet3).

Model	**Validation Accuracy**
RawConvNet ($n = 80$)	95.21%
ConvNet 1	74.00%
ConvNet 2	78.08%
ConvNet 3	75.04%
Logistic regression	94.60%
Random forests	93.21%
KNN ($n = 5$)	85.47%
SVM OVR	83.91%

Table 6.14: Performance summary for ConvNet models on BUZZ1 validation dataset.

6.6.5 Results on BUZZ2

We can recall from Section 6.2 that there are 12,914 audio samples in BUZZ2. The training and testing dataset in BUZZ2 are completely separated with respect to beehive and location. This means there is no overlap in the data as was the case in BUZZ1. The training set consisted of 7582 samples from beehive 1.1 and the testing set contained 2332 audio samples from beehive 2.1. The ConvNets trained on raw audio waveform; RawConvNet, ConvNet1, ConvNet2 and ConvNet3 were trained on BUZZ2 training/testing dataset using the same hyperparameters as discussed in Section 6.6.1. Table 6.15 shows the comparison of the performance of RawConvNet with ConvNets1, ConvNet2 and ConvNet3 on the BUZZ2 training/testing dataset.

Number of Training Samples: 7582; Number of Testing Samples: 2332				
Model Name	Training Loss	Training Accuracy	Testing Loss	Testing Accuracy
RawConvNet (n=80)	0.00348	99.98%	0.14259	95.67%
ConvNet 1	0.47997	99.99%	0.55976	94.85%
ConvNet 2	0.64610	69.11%	0.57461	95.50%
ConvNet 3	0.65013	69.62%	0.58836	94.64%

Table 6.15: RawConvNet is compared with ConvNets1, ConvNets2 and ConvNets3 after 100 epochs of training on BUZZ2 train/test dataset. The receptive field size n in the first layer is set to 80 for ConvNet1, ConvNet2 and ConvNet3.

From the Table 6.15 we can see that the Testing accuracy for all the models are approximately in the similar range. However the testing loss is considerably higher for all 3 of the deeper ConvNets; ConvNet1, ConvNet2 and ConvNet3. Thus we can say that our proposed RawConvNet with a custom layer and a shallower architecture, performs much better than the three deeper ConvNets without custom layer.

6.6.6 Results On BUZZ2 Validation Dataset

We can recall from Section 6.2 that BUZZ2 validation dataset comprised of 3000 samples out of which 1000 were from class `B`, 1000 from class `N` and 1000 from class `C` respectively. The validation dataset consisted of data from a beehives 1.3 and 1.4 which were deployed in 2018. Thus the validation data differed from the training and testing dataset by beehive, time of collection and bee race. We tested our models ConvNet1, ConvNet2, ConvNet3 and RawConvNet on the above validation dataset.

True	***Predicted***				
	Bee	**Noise**	**Cricket**	**Total**	**Validation Accuracy**
Bee	969	31	0	1000	96.9%
Noise	6	994	0	1000	99.4%
Cricket	0	67	933	1000	94.8%
Total Accuracy					96.53%

Table 6.16: Confusion matrix for RawConvNet with receptive field size n=80 on BUZZ2 validation dataset

True	***Predicted***				
	Bee	**Noise**	**Cricket**	**Total**	**Validation Accuracy**
Bee	865	134	1	1000	86.5%
Noise	244	667	89	1000	66.7%
Cricket	0	43	957	1000	95.7%
Total Accuracy					82.96%

Table 6.17: Confusion matrix for ConvNet1 on BUZZ2 validation dataset.

True	***Predicted***				
	Bee	**Noise**	**Cricket**	**Total**	**Validation Accuracy**
Bee	766	178	56	1000	76.6%
Noise	152	776	72	1000	77.6%
Cricket	0	36	964	1000	96.4%
Total Accuracy					83.53%

Table 6.18: Confusion matrix for ConvNet2 on BUZZ2 validation dataset.

True	***Predicted***				
	Bee	**Noise**	**Cricket**	**Total**	**Validation Accuracy**
Bee	864	135	1	1000	86.4%
Noise	248	674	78	1000	67.4%
Cricket	0	36	964	1000	96.4%
Total Accuracy					83.40%

Table 6.19: Confusion matrix for ConvNet3 on BUZZ2 validation dataset.

We also calculated the performance of SpectConvNet (refer to Figure 6.6 and Table 6.1) by training it in BUZZ2 train/test dataset and evaluated the model performance on BUZZ2 validation dataset. The corresponding confusion matrix is given in Table 6.20.

True	***Predicted***				
	Bee	**Noise**	**Cricket**	**Total**	**Validation Accuracy**
Bee	544	445	1	1000	55.40%
Noise	182	818	0	1000	81.80%
Cricket	0	42	958	1000	95.80%
Total Accuracy					77.33%

Table 6.20: Confusion matrix for SpectConvNet on BUZZ2 validation dataset.

In order to evaluate our deep learning models, we followed the same procedure as in the previous section and compared the performance of our deep learning models against the standard machine learning models.

Model	**Validation Accuracy**
RawConvNet ($n = 80$)	96.53%
ConvNet 1 ($n = 80$)	82.96%
ConvNet 2 ($n = 80$)	83.53%
ConvNet 3 ($n = 80$)	83.40%
SpectConvNet	77.33%
Logistic regression	68.53%
Random forests	65.80%
KNN ($n = 5$)	37.42%
SVM OVR	56.60%

Table 6.21: Performance summary of raw audio ConvNets, SpectConvNet and ML models on BUZZ2 validation dataset.

In Table 6.21, the performance of the ConvNets and the four machine learning models on BUZZZ2 validation dataset is presented. From the results we can say that on BUZZ2 validation dataset, where the data was separated from the train/test dataset by beehive, time of collection and race of bees, ConvNets designed to process raw audio waveform performed the best. But from the validation accuracies reported in Table 6.21, we can see that our proposed RawConvNet model performed the best in comparison to the other ConvNets on both BUZZ2 validation dataset. We can also see SpectConvNet showing a better performance as compared to the machine learning models but the validation accuracy is far less than the RawConvNet. This suggests that the proposed RawConvNet along with the three deeper ConvNets generalized better than the standard machine learning models. This suggests that RawConvNet generalizes the best overall.

6.7 Running The Trained RawConvNet Model On Raspberry Pi

The RawConvNet model with the custom layer was persisted along with the weights and variables of the different layers on the Linux computer where it was trained and then

saved on the sdcard of a raspberry pi 3 model B v1.2. Two experiments were performed on the raspberry pi to estimate how feasible it is to do in situ audio classification with RawConvNet on the raspberry pi. We used two hundred audio samples to perform our experiments. Each audio sample was 30 seconds long. We used a fully charged Anker Astro E7 26800 mAh portable battery to power our BeePi system during the experiments.

The BeePi system records a 30 seconds audio file every 15 minutes. In our first experiment we emulated a similar scenario. We set up a cronjob, running a script in Python 2.7, that would pick a random audio audio sample (out of the 200 test samples) every 15 minutes and split it into overlapping 2 second segments. The RawConvNet was then loaded into memory to classify those 2 second samples into the 3 classes: `B`, `N` and `C`. The fully charged battery supported audio classification for 40 hours during which 162 30-second audio samples were processed. Thus, it took the algorithm, on average, 13.66 seconds to process one 30-second audio sample.

In the second experiment, the python script was modified to pick 4 random audio samples every 60 minutes. This experiment was to test whether it is efficient to load the RawConvNet model every 15 minutes and perform the classification task on a single 30-second audio sample or load the RawConvNet model every 60 minutes and perform the classification task on 4 30-second audio samples. Thus in other words, we tried to understand whether batch approach was more efficient and faster. The fully charged battery supported audio classification for 43 hours during which 172 30-second audio samples were processed. Thus, it took the algorithm, on average, 37.68 seconds to process 4 30-second audio samples, which comes down to 9.42 s per one 30-s audio sample. Thus we can see applying a batch approach to classify recorded audio samples is much more efficient. But more importantly we see that a convolution neural network trained to classify waveforms of raw audio signals was able to operate on the raspberry pi and thus can be successfully added to the to the repertoire of in-situ audio classification algorithms for audio beehive monitoring.

CHAPTER 7
ANALYZING BEE BUZZING AND COMPARING IT TO HIVE DEVELOPMENT AND BEE MOTIONS

7.1 Goal

In this chapter, we will be discussing methods to filter out bee buzzing from audio recordings and then using the filtered audios to find the intensity of bee buzzing. We will then use the intensity or the power of bee buzzing and analyze it for the entire beekeeping season of 2018. In this chapter, we will be using the term power of bee buzzing and intensity of bee buzzing interchangeably. We will then investigate whether it is possible to make inferences about hive development by analyzing the buzzing intensities. Towards that end, we will discuss how bee buzzing can be compared with bee motions by analyzing the audio and video samples recorded using our BeePi system. We calculated the bee motions using the digital particle image velocimetry (dpiv) based method described in Chapter 5.

7.2 Audio Data For Analysis

In Chapter 6, we discussed the design of a neural network model called RawConvNet which processed raw audio samples without any feature engineering. Our designed model achieved an accuracy of 96.53% on a dataset where the training and testing samples were separated from the validation samples by beehive, location, time, and bee race. For this chapter, we selected audio data from May–November of our beekeeping season in 2018. The goal is to find the intensities of bee buzzing from the audio samples. As we have discussed in Chapter 3, audio recordings from the BeePi contain ambient noises which consists of sounds of thunder, wind, rain, vehicles, human conversation, sprinklers, relative silence, i.e., absence of any sounds discernible to a human ear and also static noises from microphones.

Our first step is to separate out the audio samples that represent the bee buzzing

from each audio recording. To achieve the above, we begin by splitting each 30 second of audio recording into overlapping 2 second segments. Then we use our described model RawConvNet, to classify those 2 second samples into the 3 classes: `B` (bee buzzing), `N` (noise) and `C` (cricket chirping). Next for our analysis, we selected only those 30 second audio in which a minimum of 20 2-second bee buzzing samples were detected. Using the above criteria for selection, we were able to select approximately 32000 30-second audio recordings.

7.2.1 Need For Audio Filtering

The RawConvNet model classifies a 2-second audio sample as bee buzzing even when there are background noises mixed in the audio signal. These background noises although not prominent can effect our goal of calculating the buzzing intensities. Examples of background noises can be bird chirping, very low intensity of human conversation, sudden spike of audio pulse or a constant noise produced due to hardware issues in audio recording arrangement. Thus the next step is to filter out the relevant frequencies that are associated with bee buzzing.

7.2.2 Different Frequencies of Bee Buzzing

Honey bees communicate through various sound signals that signify different behavior [116]. These sound signals are created from the vibrations of their wing muscles and occur within certain frequency ranges [117]. The majority of these audio signals have a fundamental frequency range between 200-600 Hz, although certain events can sometimes trigger audio signals at much higher frequency. Studies have also shown that the health of a honey bee colony can be determined by analysing the acoustic characteristics of the corresponding hive [3].

Piping Signal

Piping signal is mainly produced by young queen bees. The audio signature of a piping signal varies depending upon the queen being confined or not. The piping signal emitted by

a newly emerged virgin queen is refered to as *tooting*. The audio signature of a tooting signal possesses a fundamental frequency of 400–500 Hz. When the young queen first emerges the frequency is closer to 400 Hz but after several days of emergence, the fundamental frequency increases and becomes closer to 500 Hz [118].

Piping signal produced by queens that are still confined in their cell is called *quaking*. This signal is used by the queens to indicate their presence as a response to a tooting queen. In [116], it is stated that this signal probably acts as a call to the worker bees to protect the confined queen from the tooting queen. The fundamental frequency of this type of signal is around 350 Hz [118].

It was believed previously that the worker bees produced a similar piping signal only when the hive had no queen or when they sensed any hive disturbances. But a new study in [119], found the piping signal in colonies that contained a queen and were also calm and undisturbed. They reported that fundamental frequency of the audio signal was between 330–430 Hz, when it was produced by foragers from undisturbed hives. On the other hand worker piping produced frequencies ranging from 150–500 Hz or higher prior to or during swarm [120].

Hissing Signal

Hissing signal is used to identify different colony events depending upon the presence or absence of a preceding piping signal. It mainly occurs when the worker bees sense some distress or during swarms. It possesses a very broad band of fundamental frequency between 300 and 3600 Hz and is noisy [3]. This type of signal is usually audible to human ears and is typically accompanied by forager dancing and hive departures [121]. The authors in [121] also suggests that a coordinated and sequential piping and hissing signal may indicate presence of potential predators close to the hive. According to [3] a hissing signal that does not have any preceding piping can be an indication of a swarming event and could occur several days in advance from the actual hive departure. The authors in [3] summarized the different honey bee signals in Table 7.1.

Signal	**Frequency Range** (Hz)	**Signal Pattern**	**Producer**	**Significance**
Recruit	$200 - 350$	Pulse Sequence	Forager	Existence of a quality food source
Tooting	$300 - 500$	Pulse Sequence	Queen	Prevents hatching of further queens
Quacking	$300 - 350$	Pulse Sequence	Queen	Indicates viability of confined, mature queen
Worker Piping	$300 - 550+$	Single Pulse	Scout	Triggers colony hissing to prepare to swarm
Hissing	$300 - 3600$	Single Pulse	Colony	General warning/defense signal. Occurs during swarming, hive attacks, and other adverse events.

Table 7.1: Honey Bee Signals And Their Significance. Table Adapted From [3]

The audio data used for investigation in this chapter is from the beekeeping season of 2018, from May to November. During the season, we did not record any swarming activity in our beekeeping journal. Thus, referring to Table 7.1, the frequency range of 200–3000 Hz was chosen to be our primary focus. The reason we chose the upper frequency bound to be 3000 Hz was to include any hissing sounds if there were any. We did not include the frequency range 3000–3600 Hz in our analysis because we found that certain background noises generated as a result of hardware failure of the microphones were getting mixed in this frequency region. Since we were not able to perform manual inspections of the hives on a daily basis, we do not know if there were any other significant activities occurring within the hive. Thus we believe the above range would be able to capture most of the bee buzzing frequencies that could be present in the audio signals. Another important point that should be highlighted is that in all of the previous studies involving analysis of bee

buzzing, the microphones were placed inside the hives, but in our case the microphones were placed outside the hive just above the bee entrance. Thus we might not be able to capture all the events as in Table 7.1, but still we went forward with the above frequency range for completeness so that our analysis holds relevance even if the microphones are placed inside the hives.

7.3 Filtering Audio Signal

The audio recorded by our BeePi system is composed of several different frequency components. Thus to separate out the frequency range of 200–3000 Hz from the audio recording we need to use some sort of filtering that only allows frequency components within that range to pass through. To achieve the above a bandpass filter was used for our analysis in this chapter. A band-pass filter passes frequencies within a certain range and rejects (attenuates) frequencies outside that range. We will work through a simple example and see how a bandpass filter works. For this example, we will generate a signal by combining different frequencies. For the first step, we will select the different frequency components that will be present in our example signal. The sampling frequency (fs) is 500 Hz (500 samples / second). Let the total length of samples (N) in the signal be 400. Next the different frequency components for the signal are generated as specified in Equation (7.1):

$$Freq\_Comps = sin(x * n) * (1 - n), \;\; where\; n \in [0.9, 0.75, 0.5, 0.25, 0.12, 0.03, 0.025] \\ and\; x \;\in [0 \ldots N-1] \tag{7.1}$$

The above equation will generate 400 values for every n. Altogether we will have a 7×400 array. Next we will be mixing all of the frequency components together, i.e. we will add each of the 7 arrays column wise. This will generate a 1D array of length 400 and we will now refer to this new array as our example signal 'Y'. Compare this example signal 'Y' to be similar in structure to the audio recorded by the BeePi monitor. Both the signal 'Y' and BeePi audio are made up of different frequencies. Figure 7.1 shows the signal ('Y')

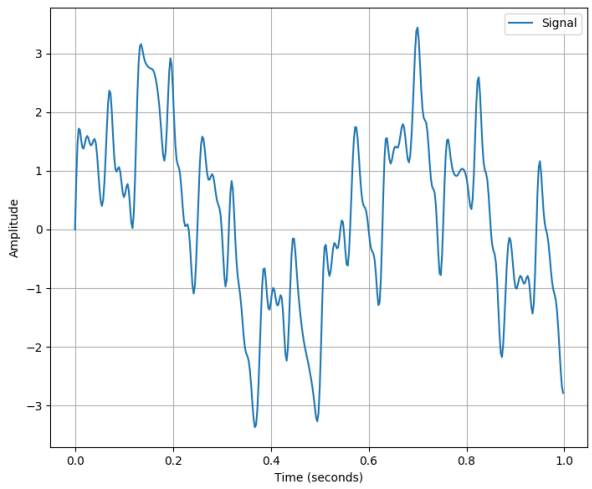


Figure 7.1: Signal Y Generated By Combining Different Frequencies

generated using Equation (7.1).

Next, we will design a bandpass filter that will only allow signals between a frequency range to go through the filter. Our goal for this example is to have the filter allow only frequencies in between 5–30 Hz, which means if we pass our signal 'Y' through the above filter, the higher frequency components will be blocked or suppressed and the graph in Figure 7.1 will look a lot smoother. Thus in the bandpass filter, the lower cutoff frequency will be 5 Hz and the higher cutoff frequency will be 30 Hz. We create the bandpass filter using inverse fft and apply the filter to the signal ('Y'). Algorithm 7.1 describes the steps involved in creating a bandpass filter.

In Algorithm 7.1, the input of Low cutoff Frequency (Low_cutoff) is the frequency below which all frequency components will be suppressed and not pass through the filter. In our examples it is 5 Hz. Similarly the other input of High cutoff Frequency ($High\_cutoff$) is the frequency above which all frequency components will not pass the filter. In our example it is 30 Hz. The green box region in Figures 7.2a and 7.2b shows the frequencies that were allowed by the bandpass filter.

The result after applying the above bandpass filter to signal 'Y' is shown in Figure 7.3a. We can see that in the filtered signal the high frequencies are gone and in effect the filtered

Algorithm 7.1 Bandpass Filtering On a Signal Using Inverse FFT

Input:

1-D array of floats, the real time domain signal (time series) to be filtered (X),
Low cutoff Frequency(Hz) (physical frequency in unit of Hz) (Low_cutoff),
High cutoff Frequency(Hz) (physical frequency in unit of Hz) ($High\_cutoff$),
The sampling frequency of the signal (Hz) (F_sample)

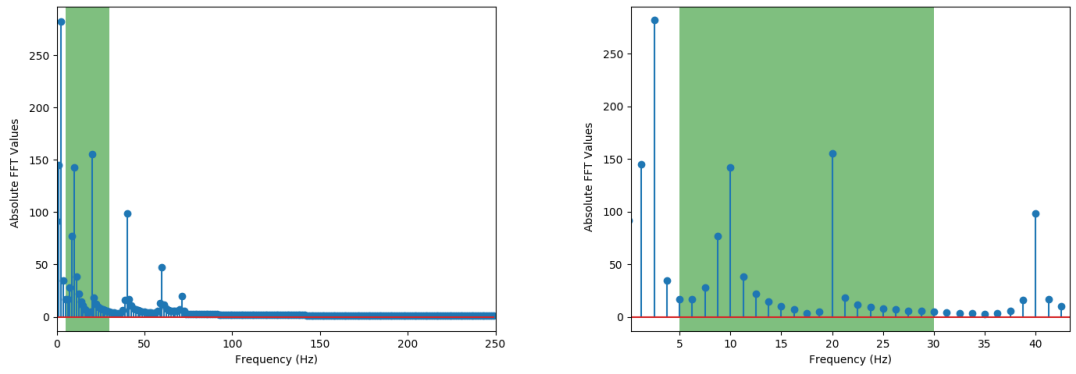
Output:

Filtered signal with the desired frequencies ($Filtered\_signal$)

Begin

Pts = length of signal X
$Freq\_Spec$ = Perform fast fourier transform (FFT) on X
$Low\_point, High\_point$ = Convert the cut off frequencies into points on $Freq\_Spec$
$Filtered\_Spec = Freq\_Spec$[i] if $i \geq Low\_point$ and $i \leq High\_point$ else 0.0,
where $i \in 0, \ldots, Pts$ /*Filtering Step*/
$Filtered\_signal$ = perform inverse fast fourier transform (iFFT) on
$Filtered\_Spec$. /*Construct filtered signal*/

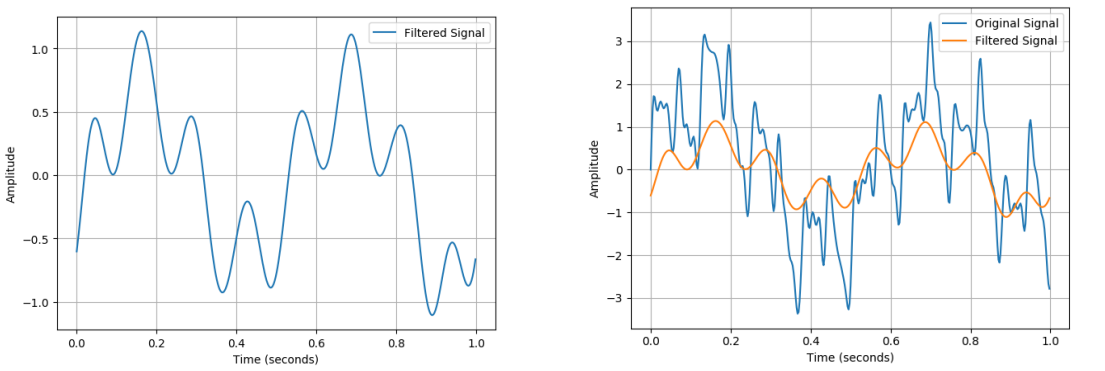
End



(a) The Green Box Shows Frequencies That Will Be Kept By The Filter

(b) Zoomed Version Of Figure 7.2a Shows The Exact Location Of The Green Box

Figure 7.2: Frequencies Allowed By The Band Pass Filter For Signal 'Y' In Equation (7.1)



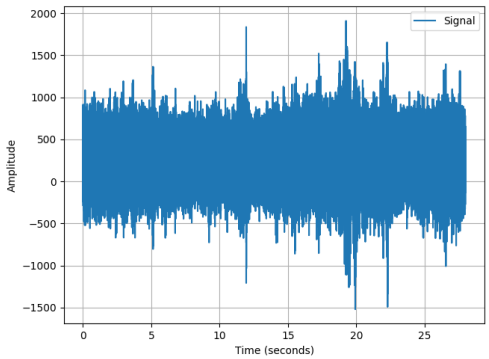
(a) Filtered Signal (b) Filtered Signal With The Original Signal

Figure 7.3: Filtered Signal Along With The Original Signal In The Same Graph

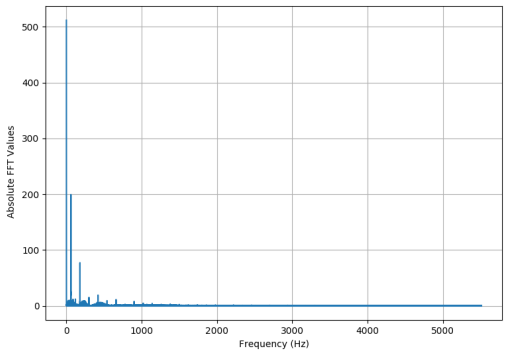
signal looks smoother than the original signal 'Y'. Figures 7.3b shows the filtered signal with our desired frequencies plotted on top of the actual signal in the same graph.

7.4 Filtering Audio Recordings From BeePi

In this section we will use the Algorithm 7.1 and the concept of bandpass filter and apply them to the 30-second audio recordings from the BeePi system. In some of the audio recordings, we found that during the last 2 seconds in the recording, there is a huge spike of static noise. We do not know the exact reason, but we can speculate that happening due to some issues in signal sampling in our recording software. So as an overall strategy we decided to remove the last 2 seconds from every audio file before processing. Thus every audio file is now 28 seconds long. The sampling rate of each audio recording is 44.1 KHz. Thus for each 28-seconds audio file, we will be dealing with $(28 \times 44.1 \times 1000)$ 1234800 samples. Since our goal is to move the entire audio analysis on the RaspberryPi itself, we decided to resample each 28-seconds audio to 11 KHz for faster processing. Hence, the total number of samples come down to $(28 \times 11 \times 1000)$ 308000. Next, from Table 7.1, we identify the frequency range of 200–3000 Hz to be of importance in our analysis. We will use the same type of analysis as we performed on our example signal in the previous section. Figure 7.4 shows a sample 28-seconds bee audio signal represented in time (Figure 7.4a) and frequency domain (Figure 7.4b) respectively.

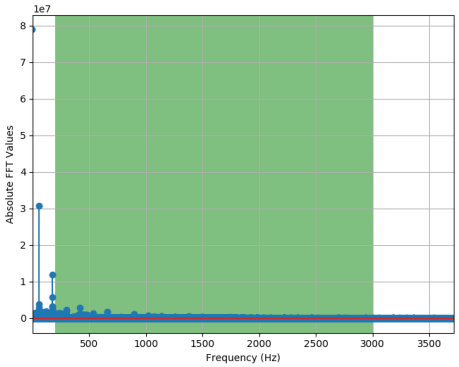


(a) Audio Signal Plotted In Time Domain

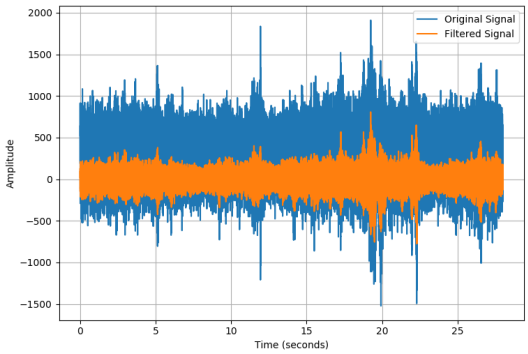


(b) Audio Signal Plotted In Frequency Domain

Figure 7.4: A bee audio recorded during 2018 beekeeping season sampled at 11 KHz. The above sample was recorded on May 28 at 11:30 Am.



(a) Frequencies Kept By The Filter



(b) Filtered Signal With The Original Signal

Figure 7.5: The green box in Figure 7.5a shows frequencies that will be kept by the filter for the bee audio signal in Figure 7.4 and in Figure 7.5b filtered signal is showed along with the original signal.

If we observe the frequency domain plot in Figure 7.4b, we can see that majority of the information spread is between 0–3000 Hz. We see a huge spike at 0 Hz and other smaller spikes at lower frequencies. We estimate this being mostly due to static noises from microphone. Table 7.1, tells us that the frequency range of 200–3000 Hz should be our primary focus to separate out bee buzzing. So next we will apply Algorithm 7.1 to the bee audio in Figure 7.4 and extract only the frequencies between 200–3000 Hz. The green box region in Figures 7.5a shows the frequencies that were allowed by the bandpass filter.

The result after applying the above bandpass filter to the original audio signal in Figure 7.4 is shown in Figure 7.5a. The filtered signal in orange is shown in the same graph as the original signal in blue for visual comparison in Figure 7.5b. From the figure we can see that most of the higher frequency values have been removed. The next step is to use the filtered signal, as in the above example and then calculate the power or intensity present in each of them.

7.5 Power In An Audio Signal

In this section we will be using the '*audioop*' module [122] in Python to find the power of an audio signal. The power can be calculated using the '*rms*' function in '*audioop*' module. It evaluates the root-mean-square of each fragment or frame inside the audio. The audioop module operates on audio fragments which are composed of signed integer samples 8, 16, 24 or 32 bits wide. In our BeePi system, the audio recording is done by setting alsa capture card on the Raspberry Pi to use '$S16\_LE$' i.e. signed 16 bits Little Endian. Audioop module requires the sample width of the sound fragment to be provided while calculating the rms. Sample width of a sound fragment is either 1, 2, 3 or 4 if the audio fragments consists of signed integer samples 8, 16, 24 or 32 bits respectively. The overall sample width also depends upon the number of channels present in the audio signal. A channel is the passage way through which a signal or data is transported. In our case audio recordings have a single channel. Algorithm 7.2 describes the steps involved calculating the power of an audio signal using the '*audioop*' module.

Algorithm 7.2 Calculate Power Of An Audio Signal

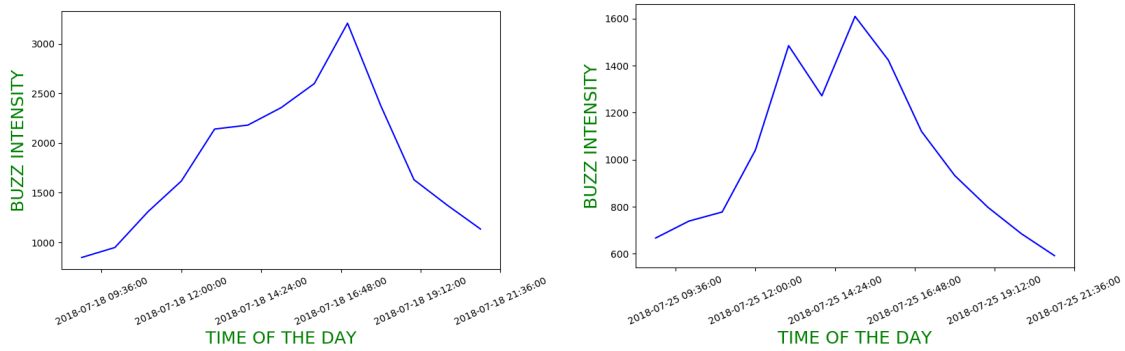
Input:
Audio file in WAVE format (*wave_audio*),
Total number of frames or samples in *wave_audio* ($nframes$),
The sampling frequency of the signal (Hz) (*rate*)

Output:
Power of the audio signal (*power*)

Begin
$sample\_width = 2$
$number\_of\_channels = 1$
$overall\_sample\_width = sample\_width \times number\_of\_channels$
while ($i < nframes$)
Begin
$frame$ = read frame or fragment at position i from *wave_audio*
$total\_amp = total\_amp$ + audioop.rms($frame$, $overall\_sample\_width$)
End
$power = total\_amp$ / $nframes$
End

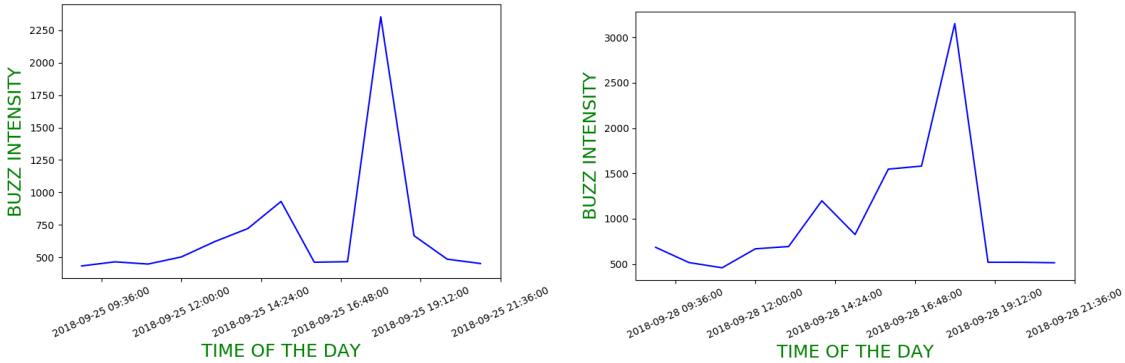
7.6 Analyzing Power Of Audio Samples

Now that we have an understanding of how to filter bee buzzing from audio samples and then calculate the power from the filtered audio, we would proceed towards analyzing the power or intensity of the bee buzzing over the entire beekeeping season of 2018 from May to November. For this analysis we will be focusing on two separate hives (R_4_5 and R_4_7) and eventually work towards finding out how these two hives have progressed through the season. Both the hives are in the same apiary in Logan, UT, but are approximately 30 feet apart. Although we had 4 hives in that apiary, but chose the above two because the BeePi sensors in hives R_4_5 and R_4_7 had the least number of hardware failures over the 2018 beekeeping season. This was also to make sure that we had good quality data for our analysis. Hives R_4_5 and R_4_7 also had slightly different surrounding as well. R_4_5 was close to a parking lot and thus had lots of background car noise and human chatter. It was in sunlight mostly during the morning hours till noon. R_4_7 on the other hand was close to a garden, where sometimes human chatter was recorded. R_4_7 remained in shade most of the day and received sunlight during the early evening to sunset hours. Both the



(a) July 18^{th} 2018, Hive R_4_5 (b) July 25^{th} 2018, R_4_5

Figure 7.6: Power Of Bee Buzzing During July 18^{th} and 25^{th} 2018 For Hive R_4_5



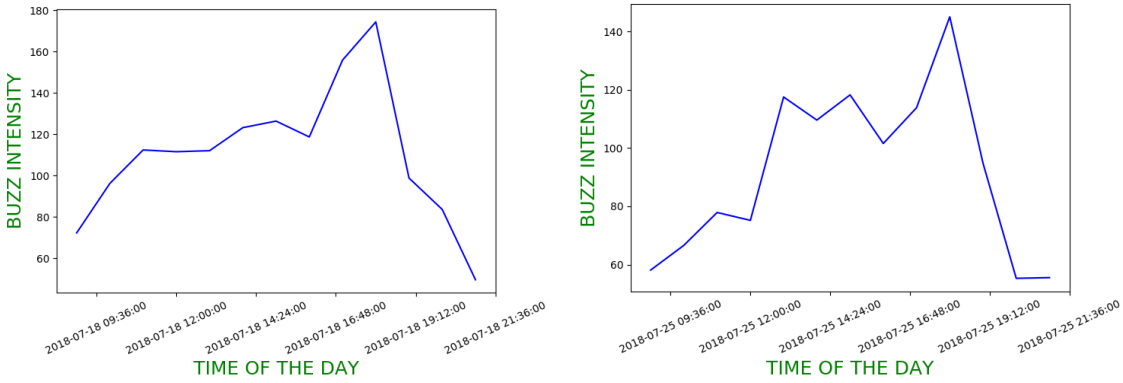
(a) September 25^{th} 2018, Hive R_4_5 (b) September 28^{th} 2018, R_4_5

Figure 7.7: Power Of Bee Buzzing During September 25^{th} and 28^{th} 2018 For Hive R_4_5

hives were located in an apiary where there are lots of trees and bushes. Thus in the audio recordings we hear a lot of bird calls and chirping. Hence the need for filtering the audio samples.

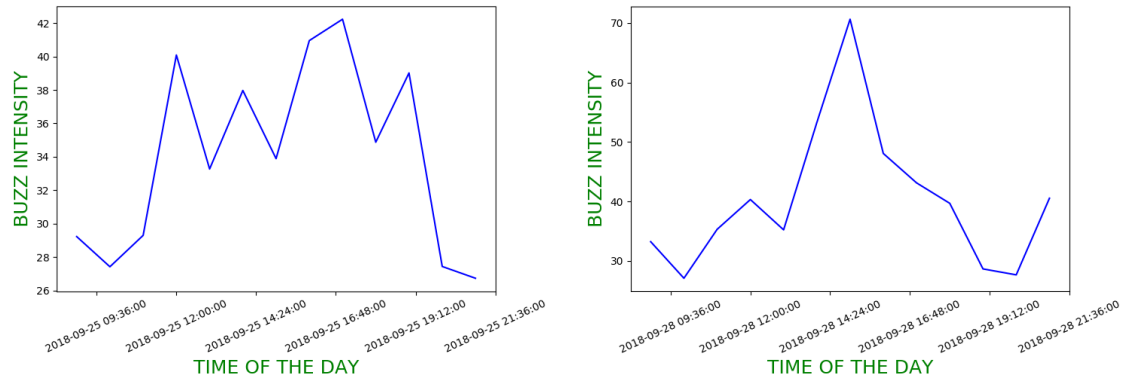
Figures 7.6a, 7.6b, 7.7a and 7.7b shows the change in buzzing power as the day progresses for Hive R_4_5 on 18^{th} July, 25^{th} July, 25^{th} September and 28^{th} September 2018 respectively. We can see from the above figures that the power of bee buzzing differs from day to day. This could be related to weather or other external factors. We will discuss about their potential relationship in the next chapter.

Figures 7.8a, 7.8b, 7.9a and 7.9b shows the change in buzzing power as the day progresses for Hive R_4_7 on 18^{th} July, 25^{th} July, 25^{th} September and 28^{th} September 2018



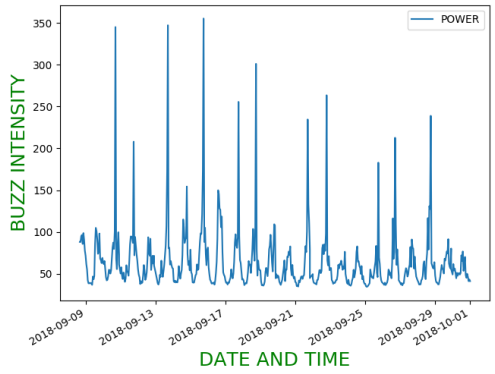
(a) July 18^{th} 2018, Hive R_4_7 (b) July 25^{th} 2018, R_4_7

Figure 7.8: Power Of Bee Buzzing During July 18^{th} and 25^{th} 2018 For Hive R_4_7

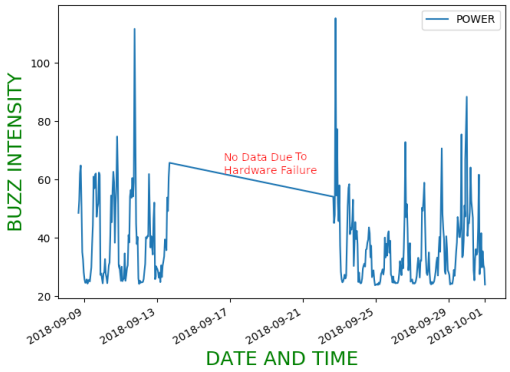


(a) September 25^{th} 2018, Hive R_4_7 (b) September 28^{th} 2018, R_4_7

Figure 7.9: Power Of Bee Buzzing During September 25^{th} and 28^{th} 2018 For Hive R_4_7

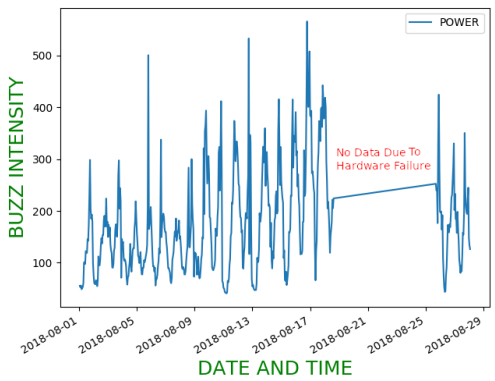


(a) September 2018, Hive R_4_5

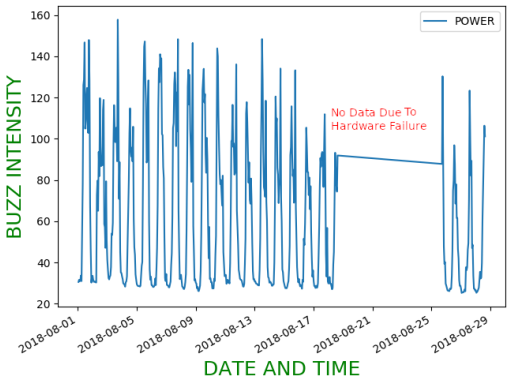


(b) September 2018, R_4_7

Figure 7.10: Power Of Bee Buzzing During September 2018 For Hives R_4_5 and R_4_7



(a) August 2018, Hive R_4_5



(b) August 2018, R_4_7

Figure 7.11: Power Of Bee Buzzing During August 2018 For Hives R_4_5 and R_4_7

respectively.

When we observe Figures 7.7 and 7.9, we see that the level of the power or intensity of bee buzzing is quite different for both the hives. The buzzing on the same days in late September has a higher intensity for the hive R_4_5 compared to R_4_7. This tells us that there was a lot less bee activity on those days in hive R_4_7. Let us see if those two days were just outliers or if the low activity of the bees continued throughout the month of August and September in hive R_4_7 as compared to hive R_4_5.

From Figure 7.10a and 7.10b, we can see that the bee activity in hive R_4_5 was

consistently higher compared to hive R_4_7 over the month of September. Thus the two days of September 25 and 28 where the buzz intensity was low, was not an outlier. The low activity of the bees continued throughout the month of September for hive R_4_7. Similarly if we observe Figure 7.11a and 7.11b we that there was consistently high bee activity in hive R_4_5 as compared to R_4_7 for the month of August. Both of these instances tell us that hive R_4_7 was not performing well in comparison to hive R_4_5. Thus if we plot the buzzing intensity graph for the entire beekeeping season of 2018 from May to November for hive R_4_5 and R_4_7, we should be able to compare them and draw a conclusion regarding hive development. To perform the above comparison, we first prepared the data accordingly. Rather than using buzzing intensities for every hour of a day, we calculated the mean of the intensities for the entire day. So rather than having 15 intensity values for the day (6am to 9pm), we just have 1 value for the entire day. After preparing the data, we plotted the average daily intensities for hive R_4_5 and R_4_7 from May to November. Figure 7.12 shows the plot for the average daily intensities over the entire beekeeping season of 2018 for both hives R_4_5 and R_4_7. If we observe the Figure 7.12 we can see that the hives R_4_5 was a better performing hive over the 2018 beekeeping season. Our observation and claim is also verified by the fact that out of all of the hives in the apiary only hive R_4_5 survived through the 2018 season into the 2019 season.

In Figure 7.13, the areas of interest for various timelines during the beekeeping season are marked. In box 'A', which is marked by a green box, we can see that the buzzing intensities grew almost similarly for both the hives R_4_5 and R_4_7. This was during the beginning of the season, which tells us that both the hives were expanding and performing well. In box 'B', which is also marked by green, we can see a very sharp increase followed by a decrease in the intensities from both the hives. The box is marked as green since during this timeline (mid June–mid July) both the hives showed almost similar performance.

The next box 'C' is drawn for analyzing the timeline end of July–end of August. In this timeline we can clearly see the difference in buzzing intensities between the two hives. hence, this tells us that during the timeline the hive R_4_5 was performing a lot better than

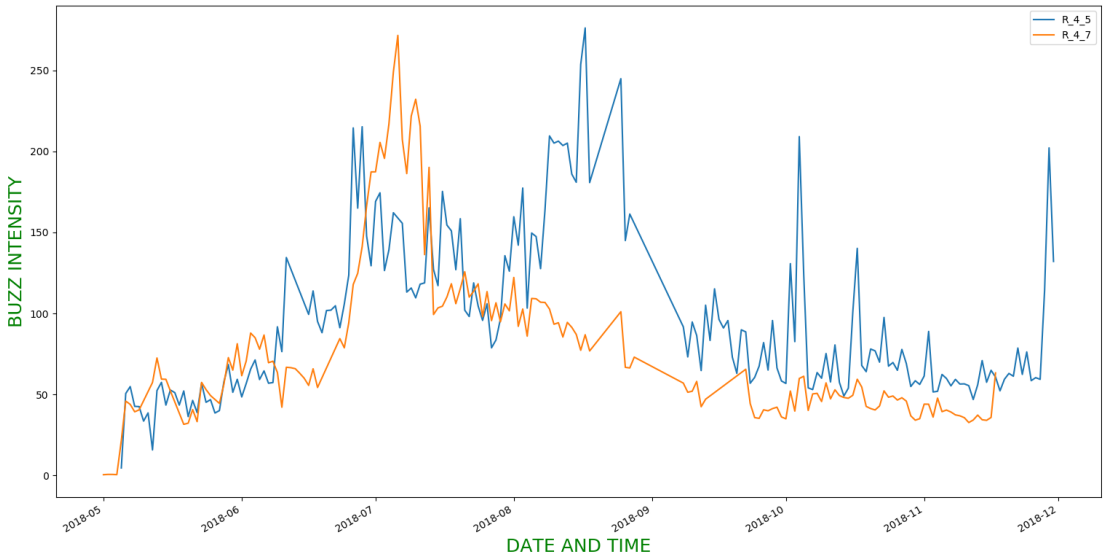


Figure 7.12: Buzzing Intensities Across Different Months In The 2018 Bee Keeping Season For Hives R_4_5 and R_4_7. The Graph Also Gives Us An Intuition About The Development Of The Hives.

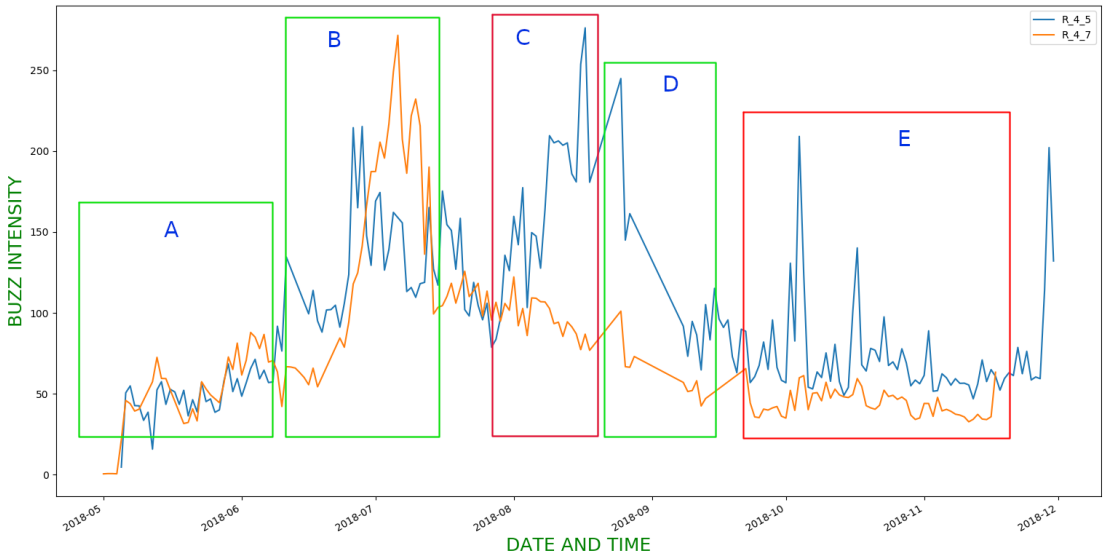


Figure 7.13: Areas Of Interests And Importance Marked For Figure 7.12. Green Boxes Signify The Timeline Where The Pattern of Buzzing Intensities Matched For Both The Curves. Red Boxes Signify The Timeline Where One Hive Was Performing Better Than The Other.

R_4_7. In other words the hive R_4_5 was developing steadily during this timeline, for R_4_7 the graph showed a downward trend. For a beekeeper, this particular timeline could be very significant, in that it can indicate the hive is under performing. This could very well serve as an early detection alarm to understand the development of hives. In a large-scale apiary with many hives equipped with BeePi sensors, those hives that show a downward trend in performance during the middle of the season could be manually inspected and necessary steps for hive treatment could be performed.

The timeline for the next box 'D' in Figure 7.13 is beginning of September–beginning of October. The box is marked as green since the performance of both the hives decreased simultaneously showing almost a similar downward trend. There might be various reasons for this downward trend, but one reason could be attributed to the local weather conditions. During this time of the year, the temperature starts to cool down as fall arrives in Logan, UT. The final box 'E' is during the timeline of mid October–end November, which are months in the back end of the beekeeping season. The temperature falls below freezing occasionally during this time of the year. This also suggests that there will be less bee movements and in turn we would be seeing lower buzzing intensities. This effect can also be seen in the graphs in box 'E'. But the amount of change or variability in the graph for R_4_5 suggests that there were still some bee movements during that time, which could suggest that the hive was preparing for the winter. On the other hand, the graph for R_4_7 showed very little variability which also means there were a lot fewer bee movements. It is hard to explain such events, but we think it could mean the hive R_4_7 was struggling. It is also evident from our beekeeping journal, from which we know that hive R_4_7 was not able to survive the harsh winter weather in Logan, UT.

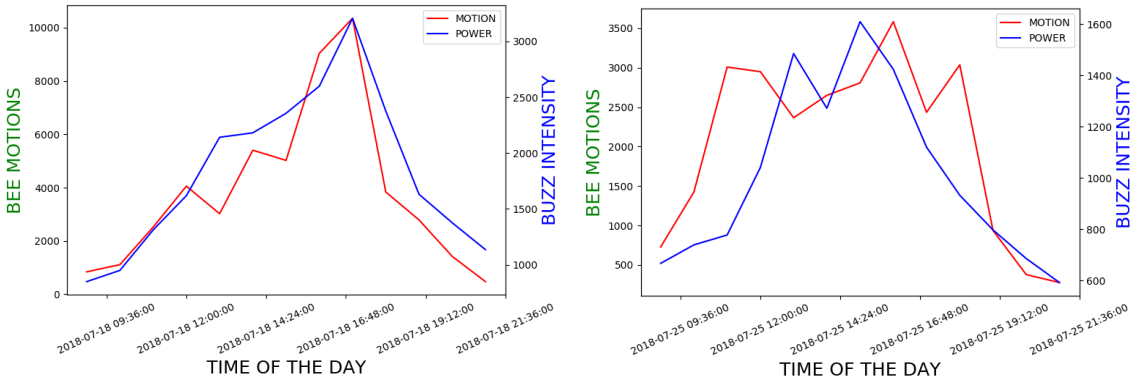
7.7 Comparing Buzz Intensities To Bee Motions

In the above sections we have used the terms bee motions and intensities of bee buzzing interchangeably. We have assumed that when there is a lot of bee motions captured by the camera in the BeePi system, likewise it will also be reflected in the power or the intensity of the bee buzzing in the audios captured by the microphones in the BeePi system. So in this

section we will investigate the relation between the bee motions and the buzz intensities. Before starting we need to have an overview of our recording process. The BeePi monitor records videos of bee traffic every 15 minutes and the audio recording script records audios every 15 minutes as well. But the video and the audio recording scripts are not set up to record at the exact same timestamp. There is sometimes a lag in the data collection between the two scripts.

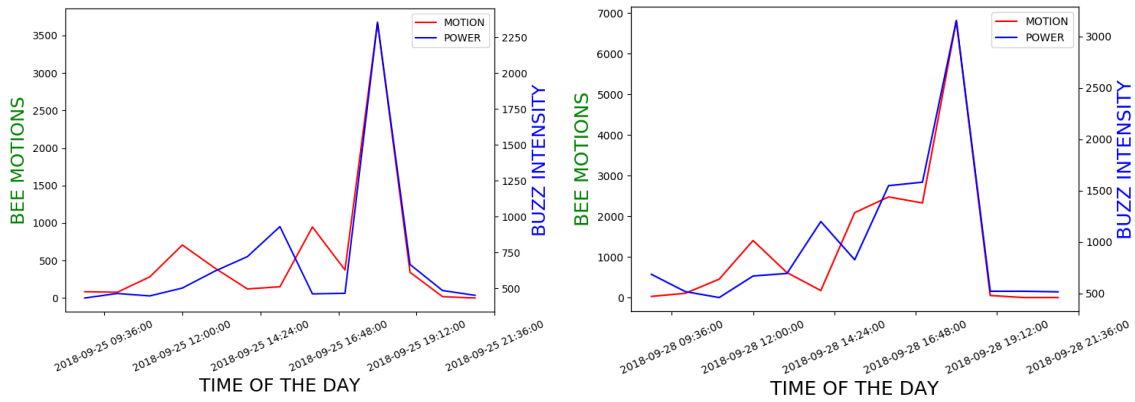
The reason behind this lag is related to how data was recorded in the previous version of BeePi monitor in 2018. In the 2018 version of BeePi, a bash script was written to perform the audio recordings. A cron job was setup which ran the audio recording script every 15 minutes. On the other hand, the video recording script was written in python and was invoked through a bash script similarly every 15 minutes. Both of the recording scripts saved the output in an external hard drive connected to the Raspberry Pi. The external hard drive was programmed to mount automatically once the Raspberry Pi turns on or restarts. But sometimes the mounting of the external hard drive would fail. Without the external hard drive both the recording scripts would fail. To solve the issue, we had to manually mount the external hard drive through the command line and then individually start both the recording script one at a time. We had to wait for one script to finish before we could start another. The difference in time could range anywhere from 1 minute to 3 minutes. Thus if the camera records a sudden large spike in number of bee motions, it might not always be reflected instantly in the audio samples. But if the general trend of the bee traffic is higher over a time period it should be reflected in the audio recordings as well. For this analysis we will be using the data from the same two hives as in the previous sections, R_4_5 and R_4_7. For easier reference, we will be using the same dates of July 18^{th}, July 25^{th}, September 25^{th} and September 28^{th}, 2018.

Figures 7.14a, 7.14b, 7.15a and 7.15b show the comparison between bee motions and buzz intensities as the day progresses for Hive R_4_5 on 18^{th} July, 25^{th} July, 25^{th} September and 28^{th} September 2018 respectively. We can see from the figures that for the above days the graph of bee motions matches closely to the graph of buzz intensities. When there



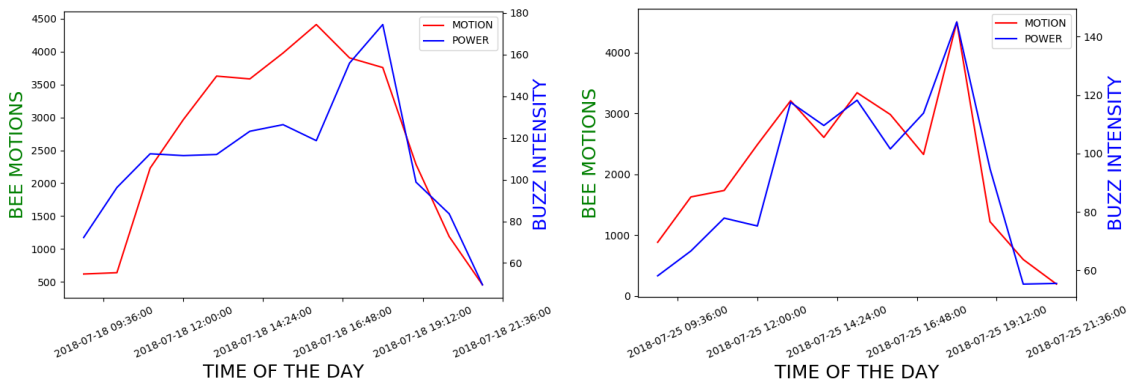
(a) July 18^{th} 2018, Hive R_4_5 (b) July 25^{th} 2018, R_4_5

Figure 7.14: Comparison Of Bee Motions And Power Of Bee Buzzing During July 18^{th} and 25^{th} 2018 For Hive R_4_5.



(a) September 25^{th} 2018, Hive R_4_5 (b) September 28^{th} 2018, R_4_5

Figure 7.15: Comparison Of Bee Motions And Power Of Bee Buzzing During September 25^{th} and 28^{th} 2018 For Hive R_4_5.

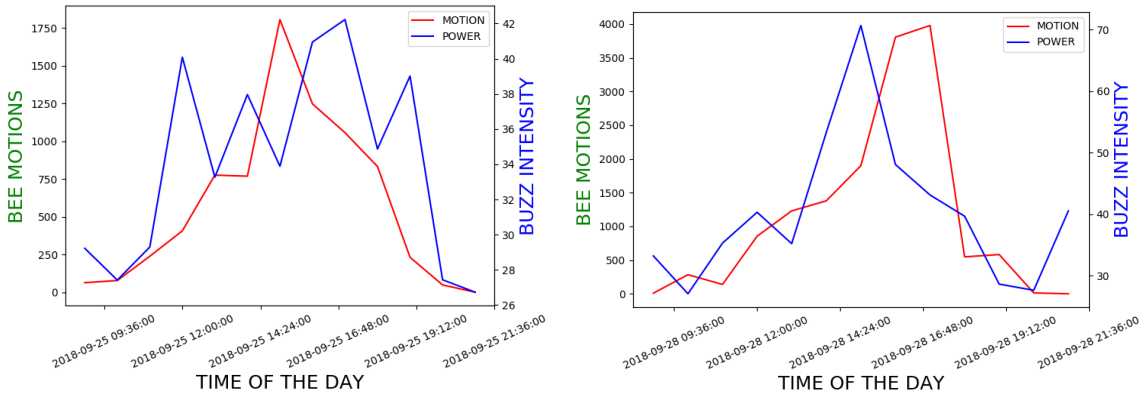


(a) July 18^{th} 2018, Hive R_4_7 (b) July 25^{th} 2018, R_4_7

Figure 7.16: Comparison Of Bee Motions And Power Of Bee Buzzing During July 18^{th} and 25^{th} 2018 For Hive R_4_7.

is a spike in the graph representing bee motions at certain times of the day, the same phenomenon is also reflected in the buzz intensity graph. Similarly when the bee motions start to reduce at the end of the day, we can see the similar reduction in the intensity of the bee buzz.

Figures 7.16a, 7.16b, 7.17a and 7.17b shows the comparison between bee motions and buzz intensities as the day progresses for Hive R_4_7 on 18^{th} July, 25^{th} July, 25^{th} September and 28^{th} September 2018 respectively. For the graphs in Figures 7.16a and 7.16b we see that the pattern is similar for both bee motions and buzz intensity. When there is a spike in the bee motion correspondingly there is a spike in the buzz intensity as well. In Figure 7.17b, we can see that the spike in the buzz intensity has been detected earlier than the spike present in the graph representing bee motion. Similarly in Figures 7.17a, we see that some spikes in the audio graphs have not been reflected in the motion graph. One possible reason for this mismatch could be due to the quality of the recorded video. Since in both the above case, the mismatch happened during the late afternoon, the videos might have been illuminated by sunlight. Thus causing the bee motions algorithm to under count. Although the spikes have been missed, but in both the cases we can see that when the bee motion count was high, it was reflected by the general upward trend in the buzz intensities as well. When the motion count reduced, the same trend can be seen in the graph representing buzz



(a) September 25^{th} 2018, Hive R_4_7 (b) September 28^{th} 2018, R_4_7

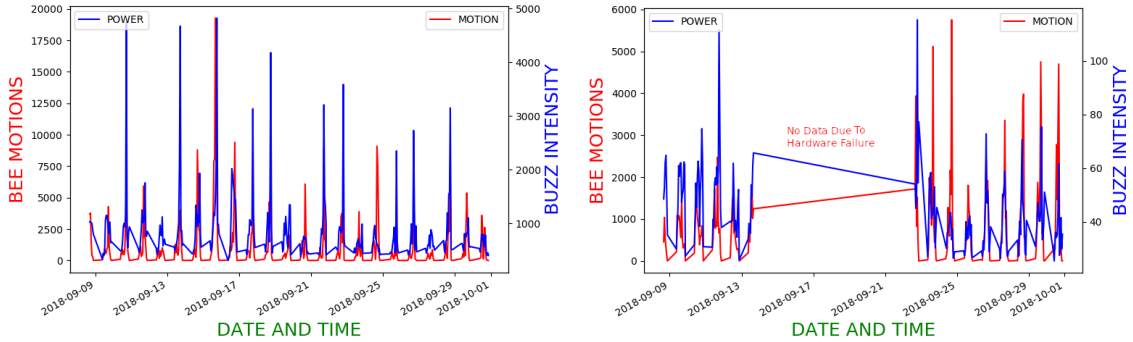
Figure 7.17: Comparison Of Bee Motions And Power Of Bee Buzzing During September 25^{th} and 28^{th} 2018 For Hive R_4_7.

intensities as well. Next, we will perform the same analysis over all the days in the months of August and September, 2018 for hives R_4_7 and R_4_5.

Figure 7.18a and 7.18b shows the comparison between bee buzz intensity and bee motions over the entire month of September. We can see that both the graphs match closely. When there is a spike in one graph is well mirrored in the other graph as well. Figure 7.19a and 7.19b shows the comparison between bee buzz intensity and bee motions over the entire month of July, 2018. We can see that both the graphs match the trends present in each other closely. Next we will see how the graphs compare against each other when plotted for the rest of the months in the beekeeping season of 2018.

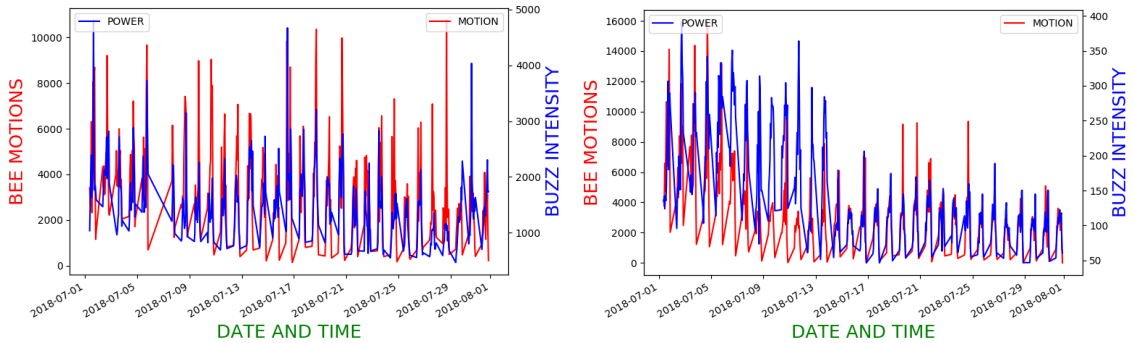
From the graphs in Figure 7.20, we can see that the curves of bee motions and buzzing intensities align closely for each month. Most of the peaks in the bee motion graphs have been mirrored in the buzzing intensity curves as well. This tells us that the hive R_4_5 was performing well and the buzzing intensity and the bee motions were closely aligned. Thus it could be possible to use either the motion counts or the buzzing intensity or a combination of both as a factor in determining the health of the bee hive.

From the graphs in Figure 7.21, we can see that the curves do not align for some of the months. In May 2018 for hive R_4_7, reason behind the misalignment in Figure 7.21a,



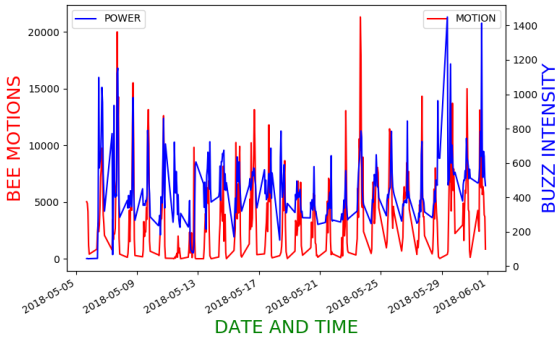
(a) September 2018, Hive R_4_5 (b) September 2018, R_4_7

Figure 7.18: Comparison Between Bee Motion And Power Of Bee Buzzing During September 2018 For Hives R_4_5 and R_4_7

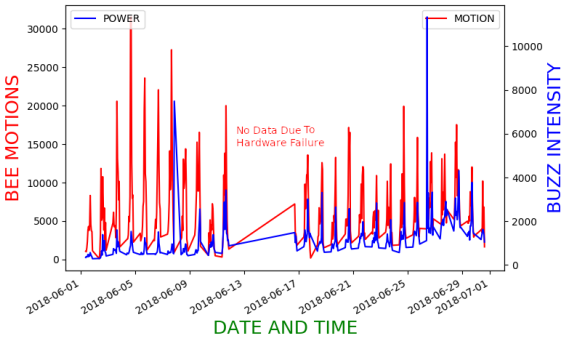


(a) July 2018, Hive R_4_5 (b) July 2018, R_4_7

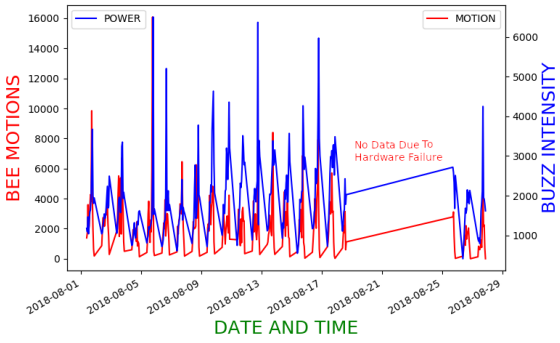
Figure 7.19: Comparison Between Bee Motion And Power Of Bee Buzzing During July 2018 For Hives R_4_5 and R_4_7



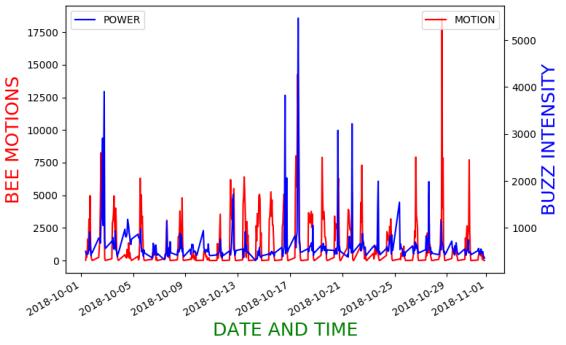
(a) May 2018, Hive R_4_5



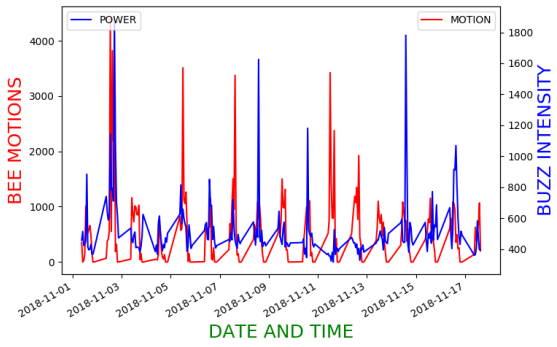
(b) June 2018, R_4_5



(c) August 2018, Hive R_4_5

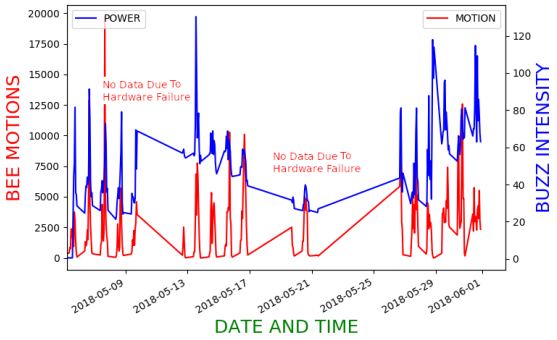


(d) October 2018, R_4_5

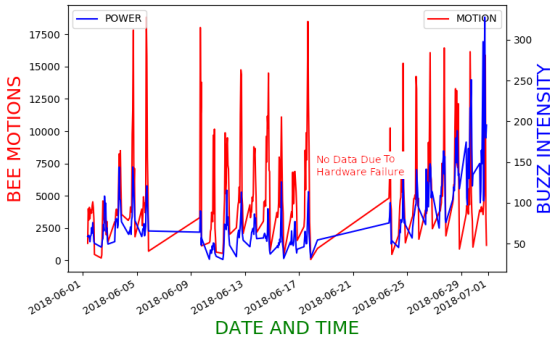


(e) November 2018, R_4_5

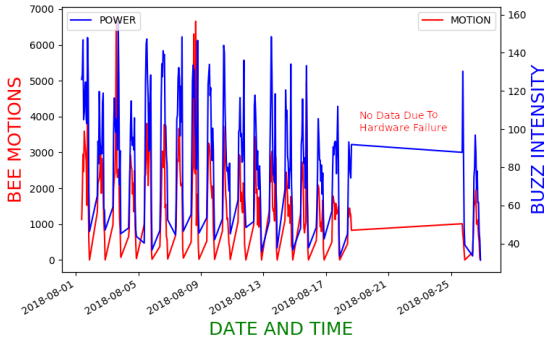
Figure 7.20: Comparison Between Bee Motion And Power Of Bee Buzzing During The Months Of May, June, August, October And November 2018 For Hive R_4_5



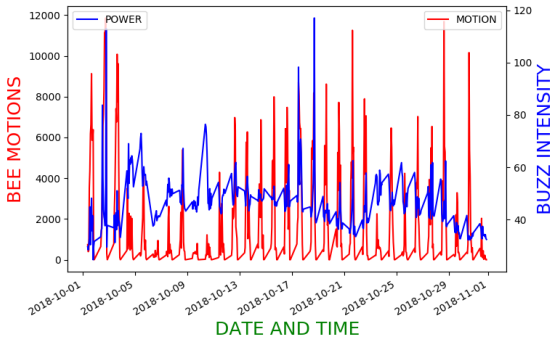
(a) May 2018, Hive R_4_7



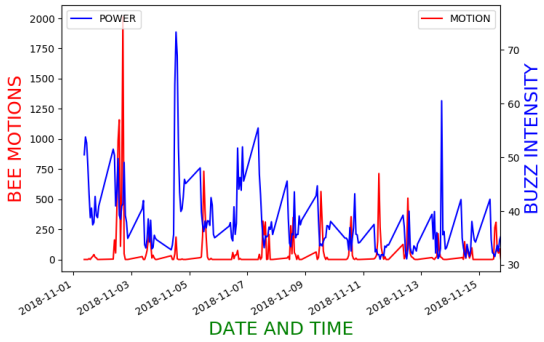
(b) June 2018, R_4_7



(c) August 2018, Hive R_4_7



(d) October 2018, R_4_7



(e) November 2018, R_4_7

Figure 7.21: Comparison Between Bee Motion And Power Of Bee Buzzing During The Months Of May, June, August, October And November 2018 For Hive R_4_7

is due to the presence of excessive noise in the audio recordings. Most of these noises were static noise at various frequency levels, which were still left behind even after filtering. Thus in Figure 7.21a even though we can see that the location of the peaks on the curves are synchronized along the timeline, but the noise remaining in the audio signals pushed the values of the buzzing intensity higher and that is what is reflected in the curves. For the Figures 7.21d and 7.21e, we can see that the curves did not align as well. It is hard to specify a particular reason, but we can speculate the fact that since we already know that hive R_4_7 could not survive the harsh winters of Logan, UT, maybe the difference in the curves could be a side effect of such a behavior. More investigation is obviously needed to understand such an effect but we can compare the Figures 7.20e and 7.21e and understand the difference in hive development for both the hives. Another possibility is the effect of weather or external conditions. It is challenging to analyze external factors that affect hive development but in the next chapter we will see how weather conditions can effect forager traffic.

CHAPTER 8
RELATING FORAGING ACTIVITY OF HONEY BEES TO LOCAL WEATHER CONDITIONS

8.1 Goal

In this chapter, we will be discussing the relationship between hive activity and local weather conditions in Logan, UT. We selected audio and video data from two of our hives in an apiary in Logan, UT during the period of May–November of our beekeeping season in 2018. The audio and video data collected by our BeePi system were then used to find the correlation between hive activity and different weather variables. The goal for this chapter is to show that without the use of additional costly intrusive hardware to get an estimate of bee counts, we can use our bee motion counting algorithm (presented in Chapter 5) to count the bee motions and then use the counts to investigate the relationship or correlation between foraging activity and local weather. Towards that end, we used the above correlation information and fitted a ordinary least squares (OLS) linear regression model to the data from June–July in 2018 and used the weather data in 2019 to predict the foraging activity for the months of May, June and July in 2019. We also investigated the performance of our model by adding bee buzzing intensity as one of the predictors alongside the weather variables to predict foraging activity, when given novel data that was not used during the model fitting.

8.2 Relation Between Weather And Foraging Activity

There have been different studies, where it has been reported that various weather conditions play a role in the foraging activity across different bee species. The authors in [92] have reported the effect of rainfall on honey bee foraging activity. They observed a higher level of foraging activity prior to heavier rainfall. They suggested that honey bees are

able to sense the likelihood of upcoming rainfall. In their experiments they used 3 honeybee colonies each having 6000 worker bees and 1 laying queen. They attached an RFID tag to newly emerged workers (n=300) of each experimental colony and monitored them for 34 days. RFID detectors and tag systems can be rather expensive, and utmost care must be taken so that the tags or the glues does not affect the bee behavior or their survivorship. Our BeePi monitors on the other hand does not require the tagging of individual bees and the sensors that we use are non-obtrusive. The dpiv based algorithm described in Chapter 5 is able to count the bee motions which is on par with human counts, thus eliminating the need for RFID based tagging of bees in our electronic beehive monitoring system.

A novel study by the authors in [93] found out the relationship between weather variables, such as temperature and solar radiation to the foraging activity of bee colonies. They gathered foraging data by using a photoelectric beecounter ('Apicard') placed at the hive entrance, for 30-min periods (data was recorded every 15s and was pooled) over 23 consecutive days from a single colony of bees and then analyzed those data with respect to ambient weather conditions. They reported that a positive correlation was found between the foraging activity and both temperature and solar radiation. It was also reported that the positive correlation between the bee activity and solar radiation was only till a certain threshold, after which the bee activity went down as the solar radiation increased.

There has also been research such as in [94], which showed the effects of temperature on the pollination activity of two separate bee species in an apple orchard in Girona (northeast Spain). It was a controlled study, that was performed during the years 1993, 1994 and 1995 in an apple orchard by monitoring 20 honey bee hives. They installed a weather station during 1995 and studied the effects of ambient temperature, relative humidity, solar radiation, and wind speed by recording data every 10 min. The weather station was located 2m above the ground and 800m away from the orchard on a flat terrain. They studied how weather conditions affect pollination and along with that they investigated the design of pollination strategies to optimize fruit yield when the weather conditions were suboptimal. Since the above investigation was performed in a controlled environment, it is challenging

to replicate the findings under natural conditions.

Following that, various research, such as in [95] and [72] have been conducted on studying the correlations between foraging behavior of honeybees and climatic conditions such as temperature, humidity and wind velocity. In [95], the investigation was performed during the 2002 growing season at Kneevi Vinogradi location, Baranja County (northeast Croatia). They performed the study on honey bees visiting sunflower inflorescences at 100, 200 and 300 meters, by recording and counting the individual bees present four times a day (9am, 11am, 1pm and 5pm). They studied the effect of temperature, humidity, precipitation and wind speed and reported the correlation between weather and foraging activity by using Spearman correlation coefficient. They reported a positive correlation between temperature and foraging activity and negative correlation between foraging activity and humidity, precipitation and wind speed. However the authors did not report the number of hives that were used for the experiments nor the data was made public.

In [72], the authors performed their analysis on data from two different bee hive colonies during the foraging seasons of July–September 2013 and June–September 2014. They presented their analysis based on 42 days of data in 2013 and 74 days of data in 2014, with data recorded with a time resolution of 1samples/min. They used a multichannel electro-optical beecounter to measure the bee egress rate and placed a weather station near the hives to record the meteorological data related to rain, solar radiation, temperature, humidity, wind speed and wind direction. A multichannel electro-optical beecounter requires hardware modifications if used on a standard hive and thus can effect the natural surroundings for the bees. They presented the effect of different meteorological variables on foraging activity and their correlation was in line with the results from previous studies in the literature. They also explored the utility of predictive modeling by fitting a generalized linear model to their data using meteorological data as the predictors and then used it to predict bee egress rate. They reported promising results and also suggested the use of other meteorological variables and even bee audio to further understand their combined effect on foraging activity. We were not able to replicate or use their models as their data was not made public.

Although a clear understanding of the different factors that might effect foraging activity is difficult, the above studies suggest that local weather conditions do play a significant role. Since weather conditions differ from place to place, we believe there is indeed scope for further analyzing the effect of different meteorological variables on bee foraging effort.

8.3 Video And Audio Data

We have used the foraging data from the bee keeping season of during June–July, October–November of 2018 and data from May–July of 2019 for our analysis in this chapter. The BeePi system records audio and video data every 15 minutes from 8:00 am to 9:00 pm. Thus we have 4 recordings every hour and in total we have 52 recordings for each day of the investigating time period. We will be focusing on two separate hives (R_4_5 and R_4_7) to understand the effect of weather conditions on foraging activity. Both the hives are in the same apiary in Logan, UT, but are at least 30 feet apart. Although we had 4 hives in that apiary, we chose the above two because the BeePi sensors in hives R_4_5 and R_4_7 had the least number of hardware failures over the 2018 beekeeping season. This was also to make sure that we had good quality data for our analysis. Hives R_4_5 and R_4_7 also had slightly different surrounding as well. R_4_5 was close to a parking lot and thus had lots of background car noise and human chatter. It was in sunlight mostly during the morning hours till noon. R_4_7 on the other hand was close to a garden, where sometimes human chatter was recorded. R_4_7 remained in shade most of the day and received sunlight during the early evening to sunset hours. Both the hives were located in an apiary where there are lots of trees and small bushes.

8.3.1 Local Weather Data

In this investigation we plan to explore the effect of several other meteorological variables on bee foraging activity. In order to use different meteorological variables, we relied on the *Utah Climate Center* [123] to gather the relevant weather data for the time period of our investigation. The weather and climate data provided by the *Utah Climate Center* are courtesy of the National Oceanic and Atmospheric Administration (NOAA), the Federal

Aviation Administration (FAA) and other federal, state and local authorities.

No.	**Variable Name**	**Units**	**Representation**
1	relative humidity	%	rh
2	grass reference evapotranspiration	mm	eto
3	clear sky solar radiation	MJ	rso
4	average vapor pressure	kPa	ea_avg
5	precipitation	mm	$precip$
6	dew point temperature	^{o}C	td_avg
7	atmospheric CO_2 concentration	ppm	$co2\_avg$
8	photosynthetically active radiation	μmol $m^{-2}s^{-1}$	ppf_avg
9	short wave solar radiation	MJ/m^2	$solarmj$
10	air temperature	^{o}C	$airt\_avg$
11	incoming longwave radiation	W/m^2	$lwdn\_avg$
12	reflected longwave radiation	W/m^2	$lwup\_avg$
13	atmospheric pressure	kPa	$pressure$
14	incoming shortwave radiation	W/m^2	$swdn\_avg$
15	reflected shortwave radiation	W/m^2	$swup\_avg$
16	wind speed	m/s	$winds\_avg$
17	net radiation	W/m^2	$netrad\_avg$
18	canopy temperature replicate 1	^{o}C	$surfacet1\_avg$
19	canopy temperature replicate 2	^{o}C	$surfacet2\_avg$
20	mean atmospheric visibility	km	$visibilitykm\_avg$
21	atmospheric pressure corrected to sea level	kPa	$pressurekpasealevel$

Table 8.1: Weather variables used in our investigation. Although we have 21 meteorological variables, but later in our analysis we leave out *precip* (No. 5) from our list of predictor variables since there was very little precipitation recorded during our observation period.

The *Utah Climate Center* has various weather stations located across the state of Utah.

Among all the weather stations, we were interested in getting relevant data from the weather station located on Utah State University campus in Logan, Utah and it is also the closest to the location of our hives. The above weather station is approximately 2 miles from the apiary where the hives are located. It provides continuous weather data every hour and is an educational resource for students and research purposes. The station measures 43 different variables related to weather and climate conditions [124]. The sensors used in the weather station have been provided by Campbell Scientific and Apogee Instruments. Out of the 43 variables that the weather station provides, we were interested in studying 21 of them. The remaining variables were different climate related timestamps, information about wind direction and information about the station, which would not help us in our analysis. The final set of 21 variables are shown in Table 8.1. We gathered the weather data from March 2018–July 2019 for our investigation in order to exactly align with the time period of the video and audio data recordings used for the analysis in this chapter.

8.4 Predicting Foraging Activity

One of the goals in this chapter is to fit a linear regression model to the 2018 data during the months of June–July, using the calculated bee motion counts as the response variable with the meteorological measurements and intensity of bee buzzing providing the predictor variables. Towards that end, we fitted an OLS linear regression model to the data from hive R_4_5 during the months of June and July 2018, using weather data as the predictor variables and bee motion counts as the dependent variable. The reason behind the above selection is that during the months of June and July there were almost no hardware failures and also the hive started to become mature during that time. The reason behind the selection of hive R_4_5 is from our knowledge that the hive survived the harsh winters of Logan, UT. Initially the training set had $((30+31)\times 13\times 4)$ 3172 data points. The weather data is timestamped every hour. Thus in order to align the weather data to our bee motion counts, we had to reduce our bee motion counts to 1 record per hour. To achieve the above, we calculated the mean bee motion counts for the hour. Hence the final training set had (62×13) 793 data points. We then tested the model by using weather data and predicting

bee motion counts for the months of May–July in 2019 for both hives R_4_5 and R_4_7. Thus the predictive power of the model is assessed by analyzing the prediction results of the model and comparing it against actual bee motion counts calculated by our algorithm which was presented in Chapter 5.

8.5 Correlation Between Forager Activity And Meteorological Variables

The meteorological variables in Table 8.1 are a combination of measured and derived variables. It means some of them are directly measured using a sensor, whereas others are additionally calculated or derived from other variables. In this section we will be discussing how some of the measured meteorological variable in Table 8.1 is related to the foraging activity of the bees. We define forager activity as the total count of omnidirectional bee motions. The weather data available from *Utah Climate Center* is timestamped every hour. That means the data for the previous hour is available at the beginning of the current hour. Our BeePi system records data every 15 minutes. Hence the first step in our analysis was to align the timestamp of the weather data with that of the data from the BeePi sensors. To do so, we calculated the average bee motion counts for the hour and similarly calculated the average buzzing intensity for the hour and aligned them with the timestamp of the weather data. Once the BeePi data was aligned to the weather data we were able to proceed with our analysis.

The first step in our analysis was to understand how the foraging activity is affected by individual meteorological variables. We discussed in the previous section the intuition behind selecting data from hive R_4_5 during the months of June and July 2018 as our training data. Hence we will use the above data from the months of June and July 2018 to study the effect of weather conditions on foraging activity for hive R_4_5. We also extended our analysis to study the effect of weather conditions on hive R_4_7, by using the data from hive R_4_7 during the months of June and July 2018.

From the months of June and July, we had 793 (61 days * 13 records each day) data points for hive R_4_5 and similarly 793 data points for hive R_4_7. To help us in our investigation we first calculated the Spearman's rank correlation score between bee motion

counts and individual meteorological variables for each day. Thus the set of 793 data points was brought down to 61 data points (one correlation value for each day of the analysis). Next, we constructed box plots to visualize the distribution of the correlation values for individual variables across the timeline of our analysis. A boxplot is a standardized way of displaying the distribution of data based on a five number summary (minimum, first quartile (Q1), median, third quartile (Q3), and maximum) along with the outliers. Figure 8.1, shows the distribution of the correlation values for each weather variable and buzzing intensity for hive R_4_5 during the months of June and July 2018.

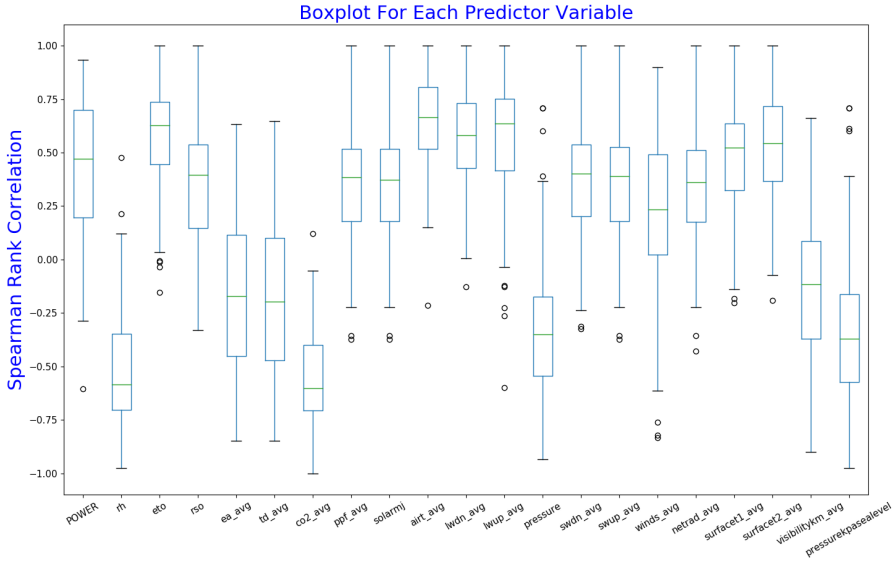


Figure 8.1: Box plot representation of the distribution of correlation values between foraging activity and meteorological variables in Table 8.1 and buzzing intensity for hive R_4_5 during June and July 2018. Each box plot also shows the corresponding median value for the distribution.

In Figure 8.1, we can visualize the distribution of correlation values across the observation period for each variable. From the median values of the box plots we can see that certain variables are correlated better to the bee motion counts, than others. For example we see that the median correlation value in case of relative humidity (rh) is less than -0.50 and only for a small number of cases the correlation value is positive. This tells us that

relative humidity is negatively correlated to foraging activity. We see a similar scenario in case of atmospheric CO_2 concentration ($c02\_avg$), where majority of the values are negative and the median value is below -0.50. This tells us that atmospheric CO_2 concentration is negatively correlated to foraging activity. There are also certain variables like average vapor pressure (ea_avg) and dew point temperature (td_avg) which does not clearly show the relationship with foraging activity. We also have variables like evapotranspiration (eto) and air temperature ($airt\_avg$) where the median for the distribution of correlation values is above 0.60 in both the cases, thereby showing high positive correlation with foraging activity. We see almost an identical trend in the distribution of correlation values for the weather variables for hive R_4_7 during the same time period as seen in Figure 8.2. This suggests that the weather conditions affected the forager traffic similarly in hives R_4_5 and R_4_7. The only difference between the Figures 8.1 and 8.2 is involving the distribution of the correlation values between buzzing intensities and forager traffic. We see in case of hive R_4_7 (Figure 8.2), the buzzing intensity is better correlated to forager activity, with the median of the distribution being closer to 0.75. On the other hand, the median of the distribution in case of hive R_4_5 (Figure 8.1), is closer to 0.50, but has less outliers.

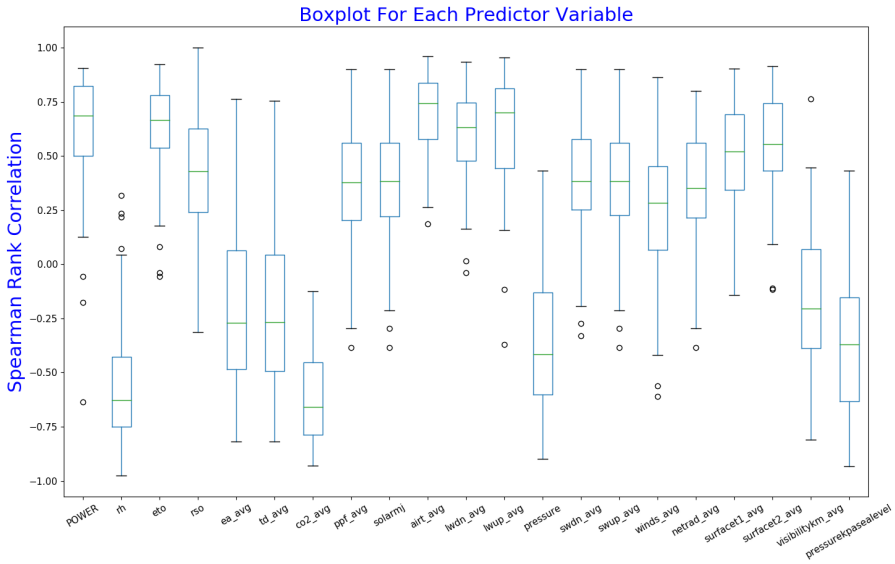


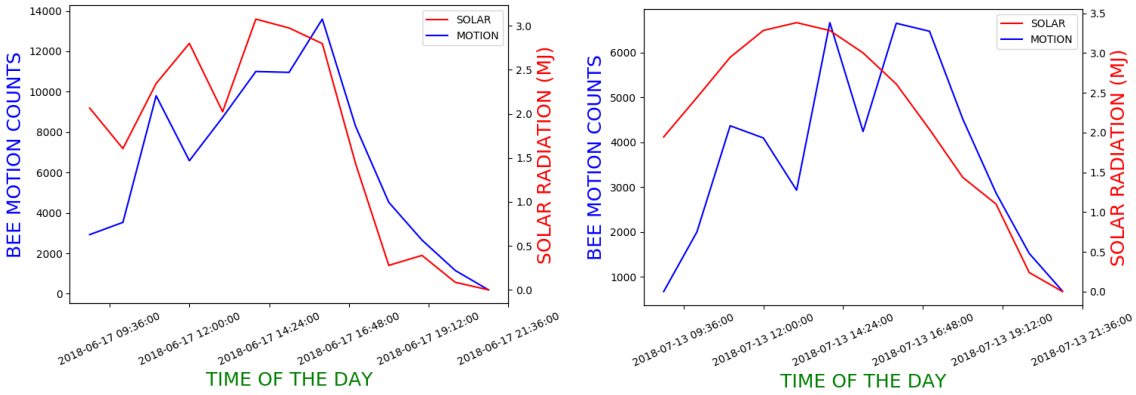
Figure 8.2: Box plot representation of correlation values between foraging activity and meteorological variables in Table 8.1 and buzzing intensity for hive R_4_7 during June and July 2018. Each box plot also shows the corresponding median value for the distribution.

In the following sections we will see the relationship between forager activity and some of the weather variables on a certain day of June 17^{th}, 2018. The purpose of this is to visualize and understand how forager activity is affected by the weather variables during different times in a day. The weather variables chosen for this per day analysis is based upon the distribution of the corresponding correlation values in Figures 8.1 and 8.2. Variables with median correlation value close to 0.50 or -0.50 were chosen. Another reason for the selection of the above date for our analysis is due to the fact that some precipitation was recorded on that particular day. We saw in Figures 8.1 and 8.2 that precipitation (*precip*) was not included in the variable list. This is because there was very little data for precipitation, since it rained very rarely. Thus in the following sections we would also perform an analysis on how forager traffic is affected by precipitation on June 17^{th}, 2018.

8.5.1 Forager Activity With Solar Radiation

We calculated the correlation between solar radiation and forager activity on June

17^{th}, 2018. Figure 8.3a shows how forager activity is affected by solar radiation. From Figure 8.3a, we can see that as the solar radiation increases between 9–12 am, the forager activity increases as well. During 12noon to 1 pm, there was a slight reduction in solar radiation due to cloud cover and we could see the same effect in forager activity. Again later in the afternoon when the solar radiation started to increase, the forager activity increased as well. But during evening when the solar radiation went down, the forager activity went down as well. This shows that for this particular day, the forager traffic is highly correlated to solar radiation. This is also reflected in the calculation of Spearman's rank correlation between the two variables. The correlation between solar radiation and forager traffic came out to be 0.81. The above value of Spearman's rank correlation tells us that the two variables are highly correlated and there is a positive correlation between them. But not all days had very high correlation between the two variables. One example is given in Figure 8.3b, where the correlation value was 0.42. We can see that as the solar radiation increases in the morning, the forager activity increases as well. But just before noon and during late afternoon around 2pm there was a slight drop in the forager traffic. Although it is hard to explain this event, we can relate this to the findings in [93], that as the solar radiation went higher above a threshold, during 11–12 noon, the forager activity went down. Along with that there could also have been other weather or external conditions that might have caused that effect.



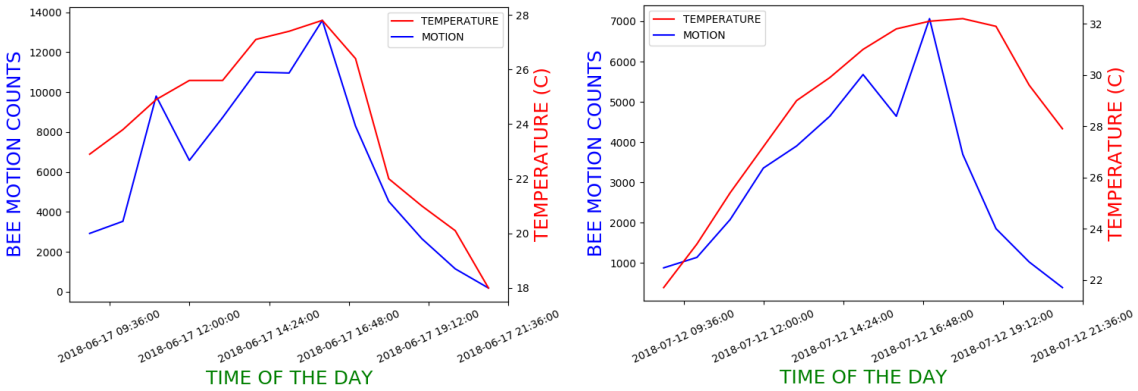
(a) June 17^{th}, 2018, Hive R_4_5 (b) July $13th$, 2018, R_4_5

Figure 8.3: Effect of solar radiation on forager traffic on June 17^{th} and July $13th$, 2018 for Hive R_4_5

Continuing with our analysis, for the months of June and July, we can see in Figure 8.1 that majority of the correlation values were close to 0.50. Although the above trend across different months tells us that it is more likely that there will be a positive correlation between solar radiation and forager activity, it also gives us the opportunity to investigate other weather variables which when combined with solar radiation could be used to predict future forager activity.

8.5.2 Forager Activity With Air Temperature

For the same day of June 17^{th} we investigated the relation between air temperature and forager activity for hive R_4_5. Figure 8.4 shows how forager activity is affected by air temperature on June 17^{th} and July $12th$, 2018.



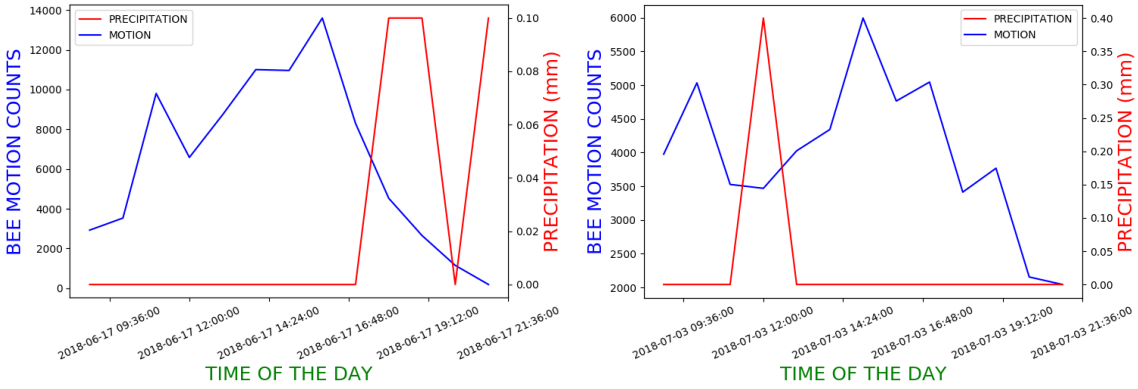
(a) June 17^{th}, 2018, Hive R_4_5 (b) July $12th$, 2018, R_4_5

Figure 8.4: Effect of air temperature on forager traffic on June 17^{th} and July $12th$, 2018 for Hive R_4_5

If we observe both Figures 8.4a and 8.4b, we can see the positive correlation between air temperature and forager activity. As a general trend we can see as the temperature goes high the forager activity increases as well and vice versa. This is reflected in the Spearman's correlation value that we calculated for both the graphs. In Figure 8.4a the correlation between temperature and forager traffic was 0.93. In Figure 8.4a the correlation between temperature and forager traffic was 0.56. From the values we can say that although the graphs of air temperature and bee motions did not exactly follow the same pattern, however they both had a positive correlation of above 0.50. We see a similar trend of high positive correlation when the correlation is calculated for the months of June and July as we can see in Figure 8.1. The median for the distribution of correlation values for $airt\_avg$ is close to 0.75. This tells us that temperature has a high positive correlation with foraging activity.

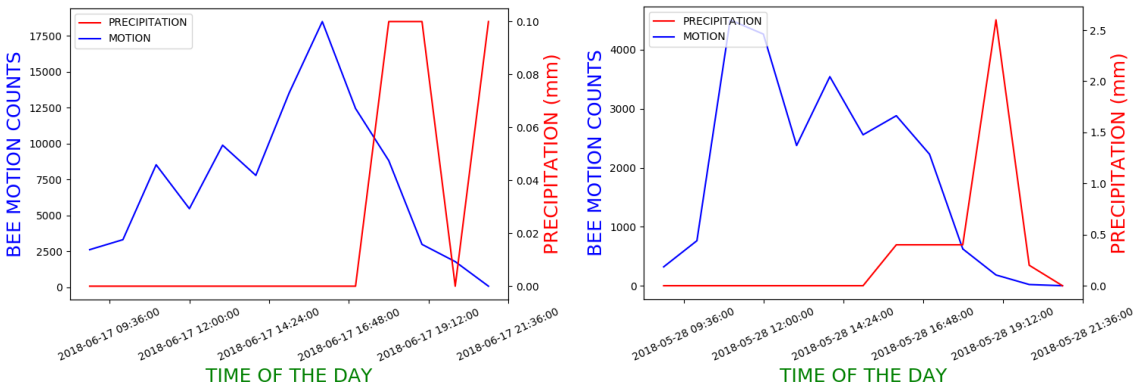
8.5.3 Forager Activity With Precipitation

During the months of June and July 2018, rainfall or precipitation was recorded only on June 17^{th} and July 3^{rd} respectively. There was also some rainfall recorded on May 28^{th}. Thus for those days we investigated how precipitation affected forager traffic for hives R_4_5 and R_4_7.



(a) June 17^{th}, 2018, Hive R_4_5 (b) July $3rd$, 2018, R_4_5

Figure 8.5: Effect of precipitation on forager traffic on June 17^{th}, July $3rd$, 2018 for Hive R_4_5



(a) June 17^{th}, 2018, Hive R_4_7 (b) May 28^{th}, 2018, R_4_7

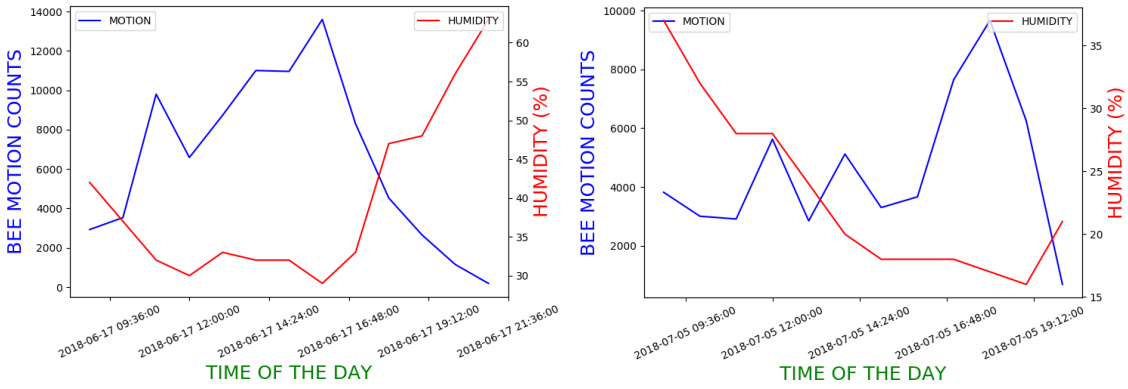
Figure 8.6: Effect of precipitation on forager traffic on June 17^{th} and May 28^{th}, 2018 for Hive R_4_7

In all four graphs in Figures 8.5 and 8.6, we can see that the forager activity goes down as soon as the rainfall starts. In Figures 8.5a, 8.6a and 8.6b, the rainfall was recorded during early evening and we see that as the rainfall amount increases, the bee activity simultaneously decreases as well. This is also reflected in the correlation value where for all three graphs we get a negative correlation value of more than 0.50. In Figure 8.6b, we see

rainfall was recorded during the late morning and noon. Due to the effect of the rainfall we see that the forager activity went down and later went up as the rainfall stopped. In this case the correlation value was -0.42. These above instances tell us that forager traffic is negatively correlated to precipitation. But is to be noted that since rainfall only occurred during a small time frame, the correlation value is on the lower side. This is because the correlation value is calculated across the entire day and for most of the day there was no effect of rainfall on the forager traffic. This tells us that although we can see that precipitation has a negative effect on forager traffic, it will not be a good predictor for the same, since it's effect is only for those specific hours when it rains.

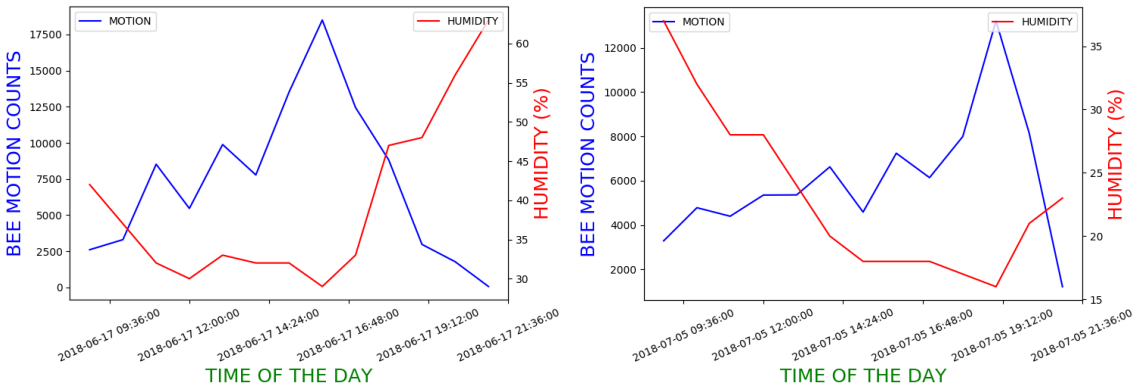
8.5.4 Forager Activity With Humidity

We investigated the effect of humidity on forager traffic for both the hives R_4_5 and R_4_7 on June 17^{th} and July 5^{th}, 2018.



(a) June 17^{th}, 2018, Hive R_4_5 (b) July $5th$, 2018, R_4_5

Figure 8.7: Effect of humidity on forager traffic on June 17^{th} and July $3rd$, 2018 for R_4_5



(a) June 17^{th}, 2018, Hive R_4_7 (b) July 5^{th}, 2018, R_4_7

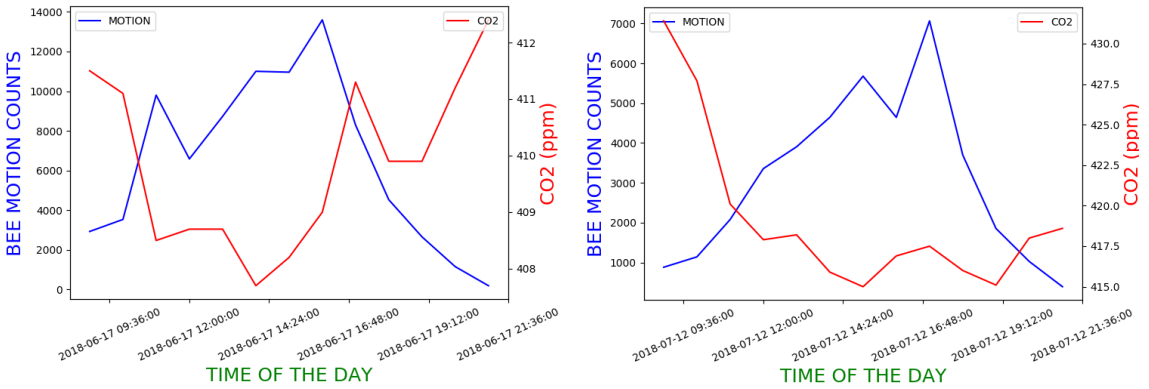
Figure 8.8: Effect of humidity on forager traffic on June 17^{th} and July 5^{th}, 2018 for R_4_7

From Figures 8.7 and 8.8, we can see that humidity has a negative effect on forager traffic. It is also evident from the Spearman's correlation value obtained for all four graphs. In each of the 4 graphs, Figures 8.7a, 8.7b, 8.8a and 8.8b the correlation value was below -0.65. If we observe the graphs closely we can see that as the humidity goes down during the day, the bee traffic goes up and when he humidity starts to go up during the later part of the day, the foraging activity goes down as well. One interesting event we can observe in each of the four graphs is that, during the timeline when the humidity is the lowest for that day, the foraging activity is also the highest during the same timeline. We also calculated the correlation values for the entire months of June and July 2018 and found for majority of the days in both months, the effect of humidity on foraging activity is high, i.e. the correlation value was below -0.60. We can see the median value close to -0.60 for humidity in both Figures 8.1 and 8.2. The above instances suggest that there is a high negative correlation between humidity and forager traffic activity.

8.5.5 Forager Activity With Carbon Dioxide (CO_2) Gas Concentration

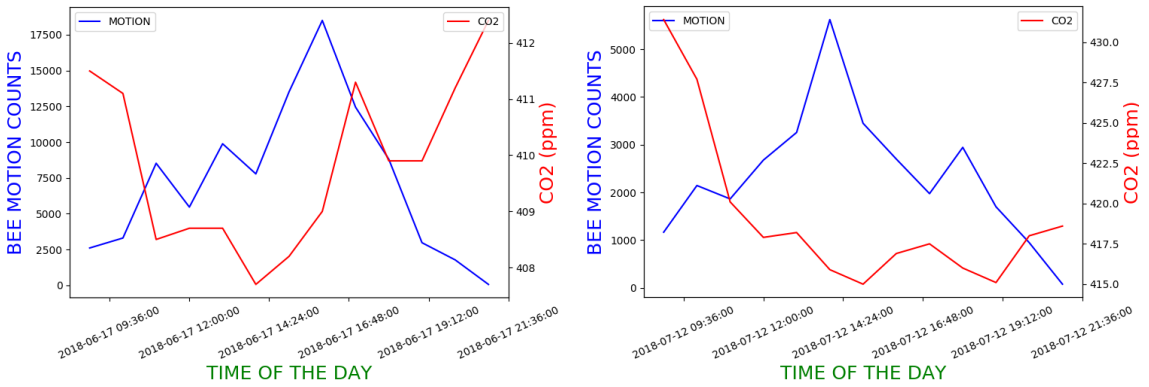
We investigated the effect of Carbon Dioxide (CO_2) gas concentration on forager traffic for both the hives R_4_5 and R_4_7 on June 17^{th} and July $12th$, 2018. The selection of days were intentionally different from the days used in the previous sections, just to show the

effect of weather across various dates.



(a) June 17^{th}, 2018, Hive R_4_5 (b) July $12th$, 2018, R_4_5

Figure 8.9: Effect of CO_2 on forager traffic on June 17^{th} and July $12th$, 2018 for R_4_5



(a) June 17^{th}, 2018, Hive R_4_7 (b) July 12^{th}, 2018, R_4_7

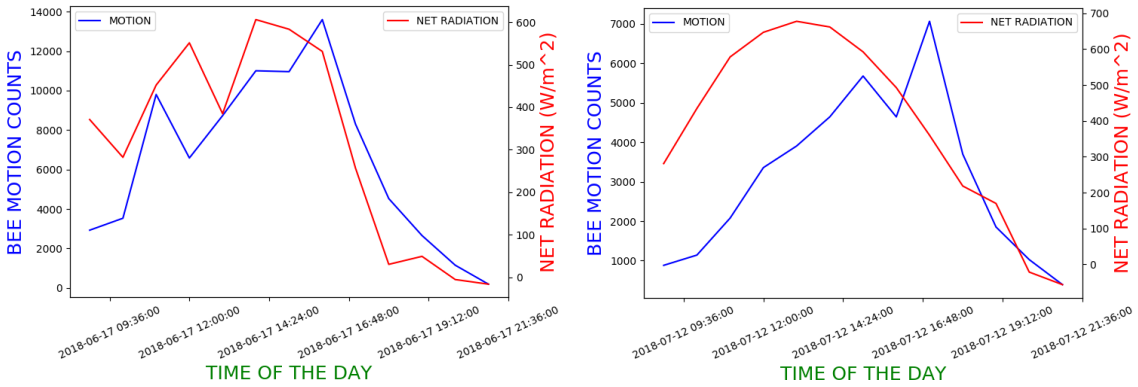
Figure 8.10: Effect of CO_2 on forager traffic on June 17^{th} and July 12^{th}, 2018 for R_4_7

From Figures 8.9 and 8.10 we can see the effect of CO_2 gas concentration on foraging activity. In each of the four graphs we can see that as CO_2 gas concentration goes down, the bees start to move more. We can also see that a small change or variation in CO_2 gas concentration does not effect the foraging activity that much. But when there is a big change in CO_2 gas concentration level, the foraging activity goes down considerably.

This can be seen in Figures 8.9a and 8.10a during the later part of the afternoon when the CO_2 level went up significantly which caused the foraging activity to come down as well. This is also evident from the correlation value, which is less than -0.60 for all the graphs in Figures 8.9a, 8.9b, 8.10a and 8.10b. We also calculated the correlation value for the entire months of June and July 2018 and found for majority of the days in both the months, the effect of CO_2 gas concentration on foraging activity is high. We can see the median of the distribution of correlation values for $co2\_avg$ was close to -0.60 in both Figures 8.1 and 8.2. The above instances suggest that there is a negative correlation between CO_2 gas concentration and forager traffic activity.

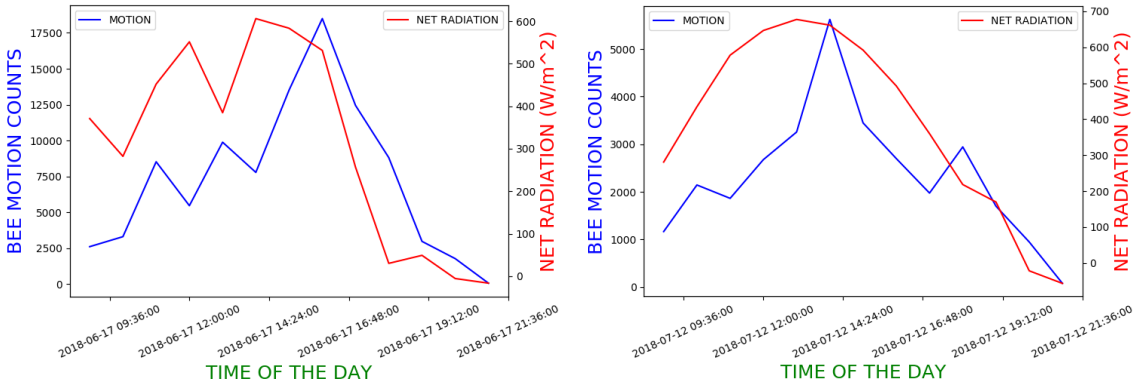
8.5.6 Forager Activity With Net Radiation

Net radiation, also sometimes referred to as net flux, is the balance between the incoming solar radiation that is absorbed by the Earth's surface and the radiation that is reflected back [125,126]. In other words, it is the total energy that is available at the Earths surface and is influential towards the climate. Some places absorb more energy than reflect, while other places on Earth reflect more energy than absorb. Earth's average temperature will rise, if the global net radiation is not close to zero. Thus we can see that net radiation acts as an important factor in our climate and weather conditions. Hence in this section we investigate the relation between net radiation and forager traffic for hives R_4_5 and R_4_7.



(a) June 17^{th}, 2018, Hive R_4_5 (b) July 12^{th}, 2018, R_4_5

Figure 8.11: Effect of net radiation on forager traffic on June 17^{th} and July $12rd$, 2018 for R_4_5



(a) June 17^{th}, 2018, Hive R_4_7 (b) July 12^{th}, 2018, R_4_7

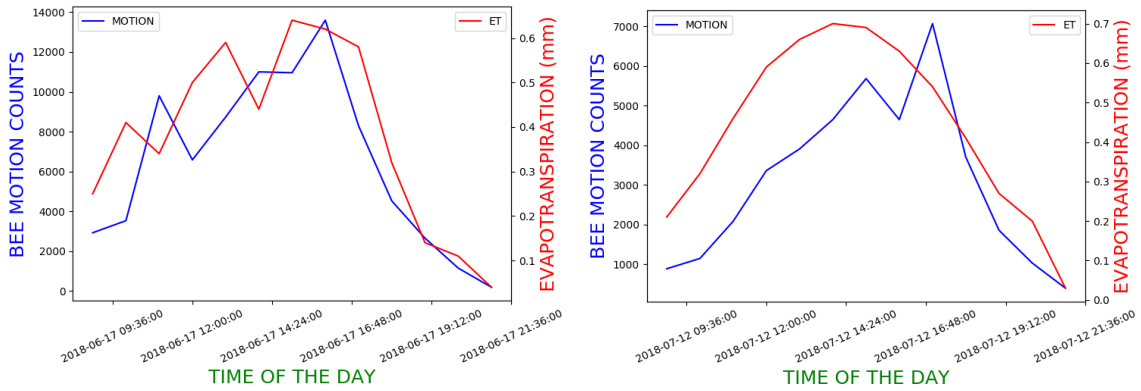
Figure 8.12: Effect of net radiation on forager traffic on June 17^{th} and July 12^{th}, 2018 for R_4_7

By visual inspection of Figures 8.11 and 8.12 we can see that there is a positive correlation between net radiation and forager activity in each of the four time series graphs. In terms of correlation value, it was above 0.60 for all the four graphs in Figures 8.11a, 8.11b, 8.12a and 8.12b. But when we observe the distribution of correlation values for the entire months of June and July in Figures 8.1 and 8.2, we can see that the values are quite spread apart,

with the median value being close to 0.40. Although this tells us that net radiation has a positive effect on forager activity, but there is not a strong relation or association between them.

8.5.7 Forager Activity With Evapotranspiration

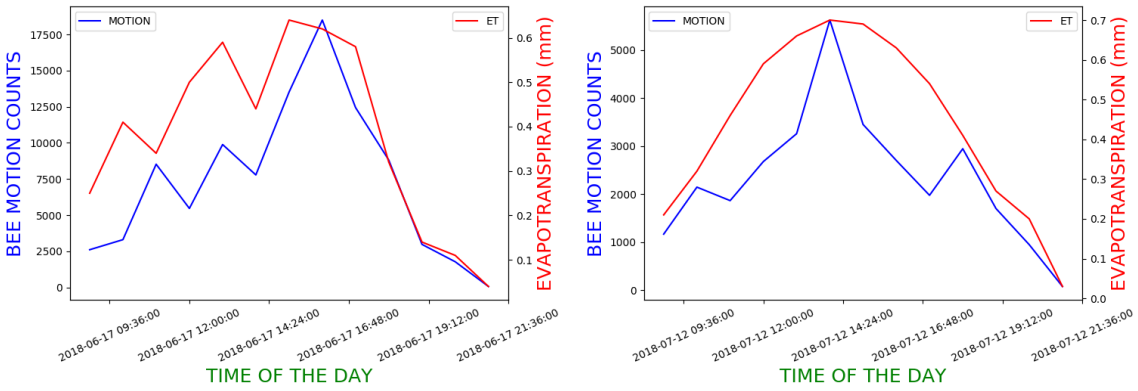
Evapotranspiration (ET), is the process by which moisture is transferred from the Earth into the atmosphere. It is the combined effect of evaporation and transpiration [127]. Evaporation happens when water vapor leaves from the surface of a plant or from the soil. Transpiration happens when water passes through a plant from it's roots through it's vascular system. The process of ET accounts for majority of the water lost from the soil during the growing period of a crop. Water continuously moves between land, sky and ocean; and this cycle is essential for the availability of water on the planet thus also affecting life on Earth. ET is the key element within that water cycle and it accounts for 15% of the water vapor in the atmosphere. Hence without ET there would not be any rainfall as clouds would not form. Thus we believe it is important to investigate the effect of ET on forager activity for hives R_4_5 and R_4_7.



(a) June 17^{th}, 2018, Hive R_4_5

(b) July 12^{th}, 2018, R_4_5

Figure 8.13: Effect of ET on forager traffic on June 17^{th} and July 12^{th}, 2018 for R_4_5



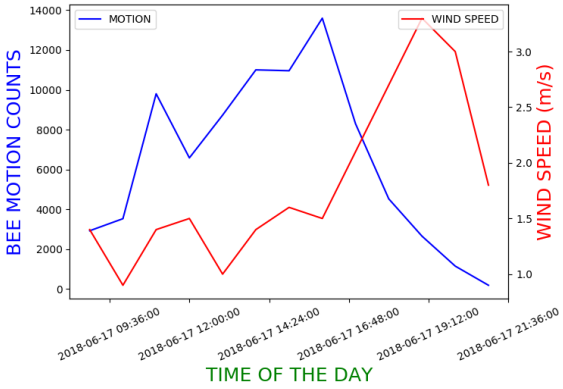
(a) June 17^{th}, 2018, Hive R_4_7 (b) July 12^{th}, 2018, R_4_7

Figure 8.14: Effect of ET on forager traffic on June 17^{th} and July 12^{th}, 2018 for R_4_7

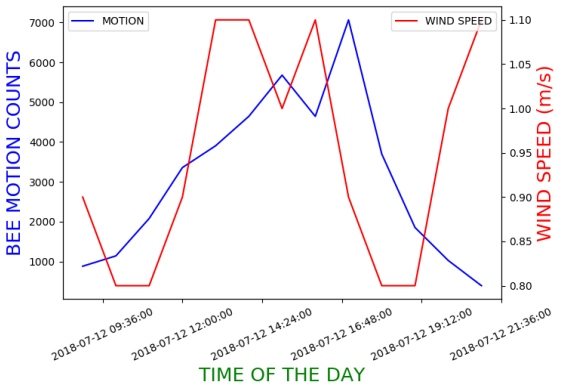
In Figures 8.13 and 8.14 we can clearly see that the foraging activity is affected by the change in ET. With the increase of ET the activity goes up and similarly the foraging activity goes down when the ET value goes down. This suggests that there is a positive correlation between foraging activity and evapotranspiration. This is also evident from the correlation value of above 0.86 for all four graphs in Figures 8.13a, 8.13b, 8.14a and 8.14b. This trend holds across the entire months of June and July, in where the median correlation value was above 0.70 as we saw in Figures 8.1 and 8.2. This tells us that there is a high positive correlation between forager activity and evapotranspiration.

8.5.8 Forager Activity With Wind Speed

In this part, we will investigate how wind speed affects forager activity. Before starting the intuition would be that when the speed of wind increases, it should result in lesser bee traffic as it would be harder to fly. Thus keeping that in mind, we should be able to see some effect of wind on forager activity for hives R_4_5 and R_4_7.

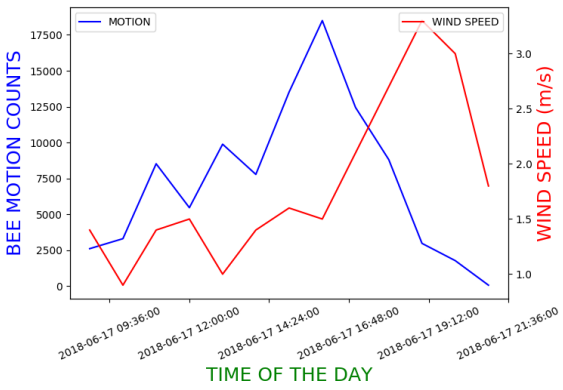


(a) June 17^{th}, 2018, Hive R_4_5

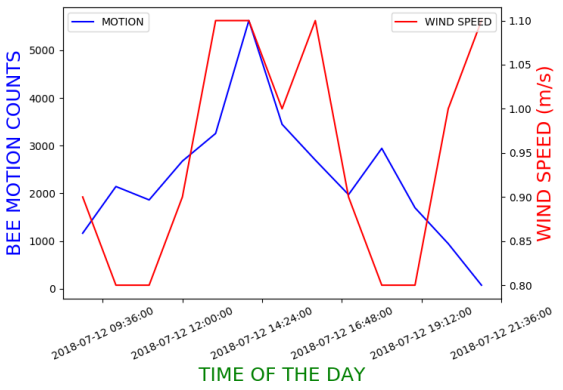


(b) July 12^{th}, 2018, R_4_5

Figure 8.15: Effect of Wind Speed on forager traffic on June 17^{th} and July 12^{th}, 2018 for R_4_5



(a) June 17^{th}, 2018, Hive R_4_7



(b) July 12^{th}, 2018, R_4_7

Figure 8.16: Effect of Wind Speed on forager traffic on June 17^{th} and July 12^{th}, 2018 for R_4_7

If we observe the Figures 8.15 and 8.16, we can observe a small trend in the graphs where the foraging activity goes down as the speed of wind increases. But the effect is not as large as the other weather variables and it is evident in the correlation value of around 0.20 which is low in all the cases. When we calculated the correlation value for the entire months of June and July, we saw in Figures 8.1 and 8.2 that the distribution of correlation

values have a wide spread from -0.50 to 0.80. This suggests that we cannot clearly say whether wind speed has a positive or a negative effect on foraging activity. One reason the above low correlation value can be attributed to is that the weather station is located 2 miles away from the hives and hence the data might not always represent the local weather conditions near the hive.

8.6 Relation Between Multiple Meteorological Variables

We have analyzed the effect of different weather or meteorological variables on foraging activity for the hives R_4_5 and R_4_7. We saw a visual representation of how change in one variable effects another. We particularly kept the day of June 17^{th} the same for each comparison. On visual observation of Figures 8.3a, 8.4a, 8.5a, 8.7a, 8.9a, 8.11a, 8.13a and 8.15a we can see that the foraging activity is not just dependent on a single weather variable, rather every variable had some effect on the bee motions. This tells us that all of these variables could act as predictors to forecast or predict future bee motions. Next we will see how individual variables are correlated with each other. Since we have presented a visual representation for June 17^{th} and July 12^{th} in the previous sections, we calculated the correlation between individual variables for the same days. The variables we are interested in are presented in Table 8.1. Figure 8.17 shows the correlation values between multiple meteorological variables for June 17^{th}, 2018 and Figure 8.18 similarly shows the correlation values between multiple meteorological variables for July 12^{th}, 2018. The correlation was calculated using Spearman's rank correlation. In both the figures we can observe some interesting events where in some cases the sign of correlation between a variable tuple is not the same on both the days.

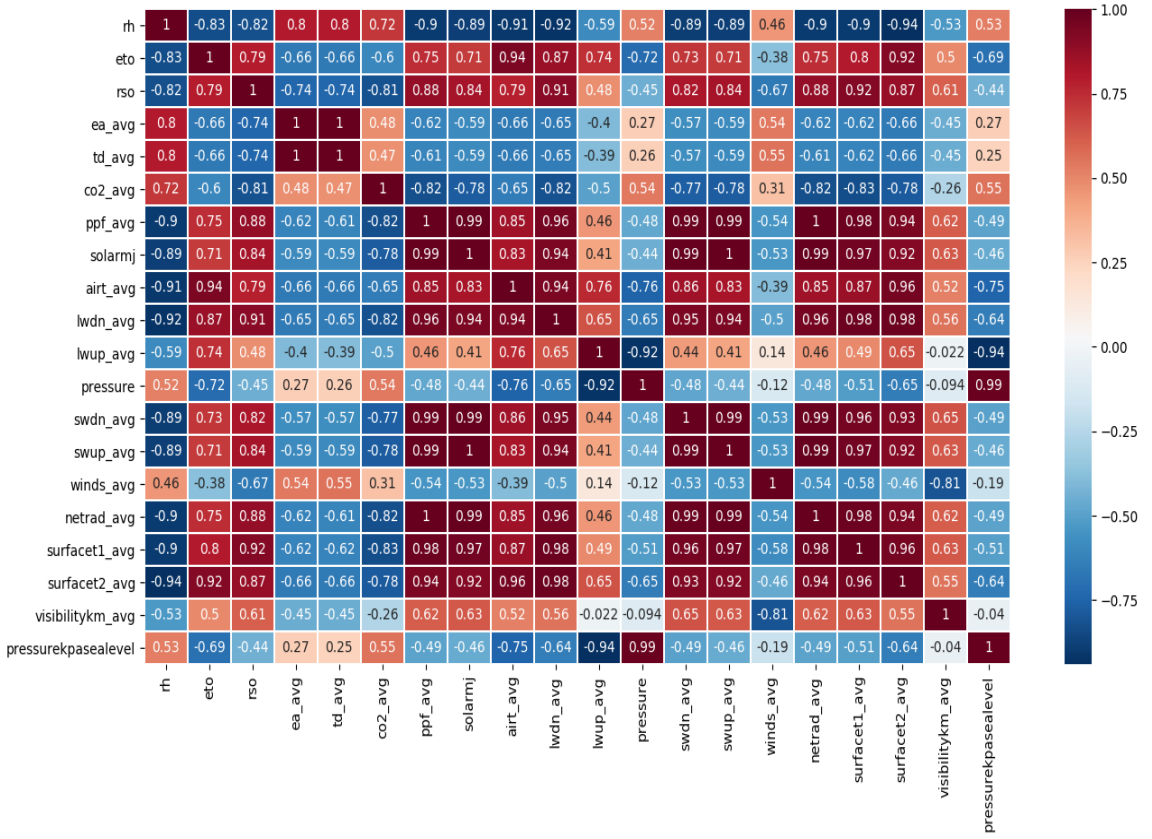


Figure 8.17: Correlation between different meteorological variables in Table 8.1 on June 17^{th}, 2018

In Figure 8.17 we see solar radiation (solarmj) is negatively correlated to relative humidity (rh) and in Figure 8.18 we see the reverse where solarmj and rh are positively correlated. We can also see that *netrad_avg* is highly correlated to solarmj and there are other highly correlated tuples as well. Since we will be using the weather variables to fit a linear regression model to the data, it is essential to calculate the colinearity between different predictor variables as it could lead to uncertainty during the calculation of regression coefficients. The reason is due to the fact that if the predictor variables are mutually covariant then there will be uncertainty in the weights of the coefficients assigned to each predictor by the regression model. Thus leading to multicollinearity, which can be a problem in a regression model because we would not be able to distinguish between the individual effects of the

independent variables on the dependent variable.

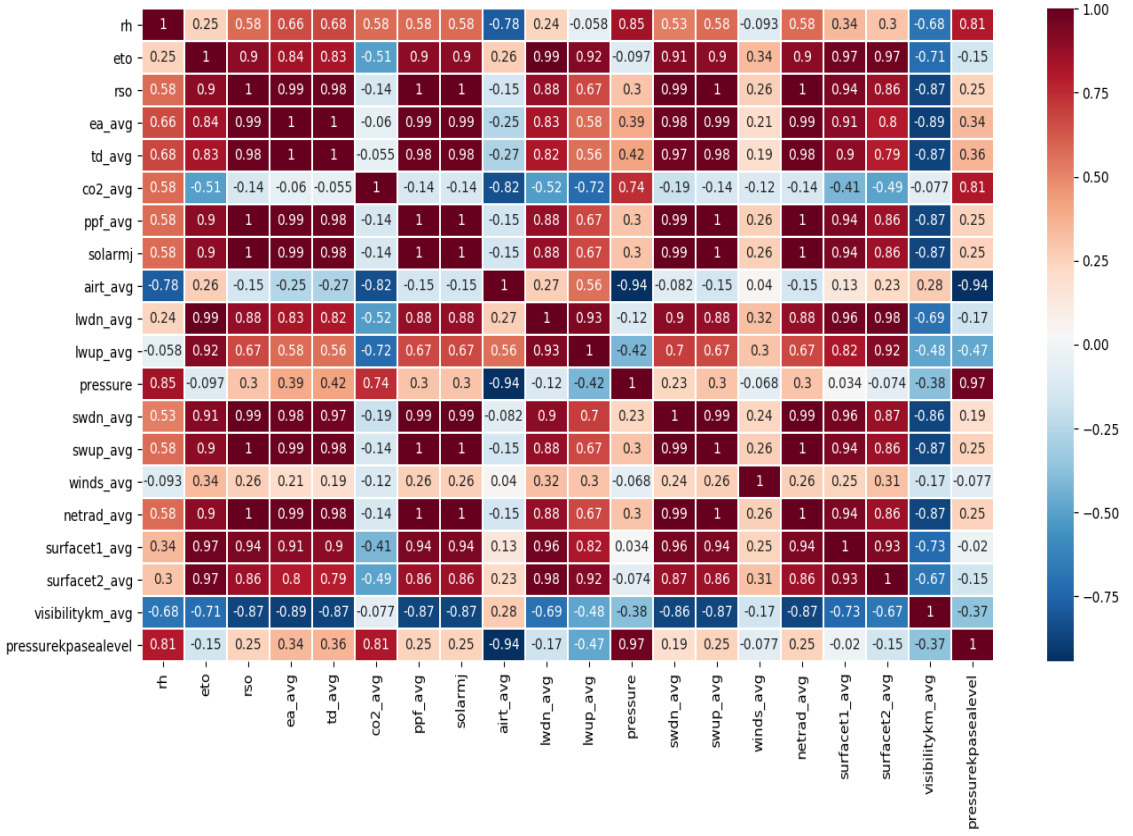


Figure 8.18: Correlation between different meteorological variables in Table 8.1 on July 12^{th}, 2018

In order to measure the effect of colinearity between our possible predictor variables we use variance inflation factor (VIF). The VIF represents the increase in variance in the value of the model coefficients as a result of their colinearity, so a value of 2 means the variance of the estimated coefficient is twice as high as it would be if the predictor variables were perfectly independent. On the other hand, a value of 1 tells us that the variables are completely independent. A VIF of more than 5 indicates potentially severe correlation between a given explanatory variable and other explanatory variables in the model, thus in those cases it is suggested to generate a derived variable from the correlated ones. The lower the value of VIF the better the effect of the predictor variables. A value between 1 and 5

indicates moderate correlation between a given explanatory variable and other explanatory variables in the model, but this is often not severe enough to require attention. It is usually a good practice to have a VIF less than 2 [128]. VIF for the variables is calculated over the entire training dataset assuming all the meteorological variables are used together as predictors.

VIF Value	**Variable**
22.66	rh
12.89	eto
20.01	rso
57.29	ea_avg
61.74	td_avg
2.01	$co2\_avg$
508.19	ppf_avg
648.94	$solarmj$
38.29	$airt\_avg$
37146.15	$lwdn\_avg$
27633.57	$lwup\_avg$
112.37	$pressure$
176296.11	$swdn\_avg$
3781048.22	$swup\_avg$
2.05	$winds\_avg$
2039656.39	$netrad\_avg$
90.68	$surfacet1\_avg$
100.42	$surfacet2\_avg$
1.63	$visibilitykm\_avg$
111.99	$pressurekpasealevel$

Table 8.2: VIF for weather variables when they are used together as predictors for building a regression model using data from June and July 2018. The dependent variable is the bee motion counts.

In Table 8.2, we can see when all the weather variables are together used as a predictor, the corresponding VIF values are really large. This tells us that there is high multicollinearity between several variables. In one such example we can see the VIF corresponding to $solarmj$ and $netrad\_avg$ is high. This makes sense, as $netrad\_avg$ is computed using $solarmj$. Hence those two variables are highly correlated and that is reflected in the

VIF values. Hence in our next step we will try to find out which subsets of weather variables result in corresponding low VIF values and higher model performance.

8.7 Regression Model Design

Our training set contained 793 (61×13) samples from the months of June and July 2018 from hive R_4_5. We fitted an ordinary least square linear regression model to the data using meteorological variables as the predictors to predict foraging activity. For simplicity we assumed there is linear relationship between the bee counts and the weather variables. To test the performance of our model, we used the data from the months of May–July 2019 as our validation set and the predicted bee motion counts or the foraging activity is compared against the actual bee motion counts.

Regression is a powerful analysis that can analyze multiple variables simultaneously by drawing a random sample from a population and then using it to estimate the properties of that population. Linear regression is used as a predictive model that assumes a linear relationship between the dependent variable (which is the variable we are trying to predict/estimate, in our case it is the bee motion counts) and the independent variable/s (input variable/s used in the prediction, in our case they are the weather variables). Since in our case the relationship exists between the dependent variable and multiple independent variables, we can call such regression as multiple linear regression. It has the following structure.

$$Y = C + M_1 * X_1 + M_3 * X_2 + M_3 * X_3 + \ldots + \epsilon \tag{8.1}$$

In the above equation (8.1), 'Y' is the dependent variable, 'C' is the constant or the Y-intercept. X_1, X_2, X_3 are the independent variables. M_1, M_2, M_3 are the corresponding coefficients on the independent variables X_1, X_2, X_3. They represent the change in the dependent variable 'Y' due to a change of one unit in the independent variables. Finally 'ϵ' is the random error. The error term accounts for the variation in the dependent variable that the independent variables cannot explain. Ordinary Least Squares (OLS) is the most

common estimation method for linear models. Using the ols function from the statsmodels package in Python3.7, we construct our linear regression models. The process of selecting the predictors for the model is done starting from a constant model (Y=C) to creating separate models by adding terms one by one and calculating the corresponding VIF values. At each iteration we check, if all the VIF values corresponding to the predictor variables are less than 2 then we keep that set of predictor variables and then use them to fit the OLS model on the data. We then check the p-value corresponding to each predictor variable for significance. The p-value for each independent variable tests the null hypothesis that the variable has no correlation with the dependent variable. If there is no correlation, there is no association between the changes in the independent variable and the shifts in the dependent variable. This tells us that each variable can be considered to be statistically significant. Thus our predictor list goes through two rounds of checking, one where the VIF is checked before fitting the model, second after the model fit the corresponding p-values are checked. If a set of predictor variables passes through both the checks, we consider that set as relevant and save it for further analysis.

From Table 8.1, we can see that there are 21 different weather variables. But since there was no precipitation recorded during majority of the observation period, we decided to leave precipitation out from our predictor list. Hence if we consider every possible combinations of the 20 weather variables we will have $2^{20} = 1048576$ different combinations. We could have used PCA to reduce the dimensions of the data or perform feature importance using Random Forest and reduce the number of features from 20, but for completeness we performed an exhaustive analysis involving all the variables. So the next step is to design an algorithm for selecting only the relevant set of predictor variables out of the 1048576 possible combinations that satisfies the above VIF and p-value criteria. The steps are explained in Algorithm 8.1. In order to optimize our processing steps and use less memory, we use the 'chain' function from the 'itertools' module in Python. So rather than generating all the 1048576 possible combinations and saving them to memory, the 'chain' functionality helps us to create an chain object that takes a series of iterables and returns one iterable. It groups all the

iterables together and produces a single iterable as output. We can then call that object and get access to the individual set of variables (*combo*).

Algorithm 8.1 Algorithm to choose a set of weather variables as predictors based on VIF and p-value.

Input:
A set of weather variables (*combo*),
Output:
Accept=True, if meets criteria. False, otherwise
Begin
Calculate VIF between the variables in *combo*
vif_bool = True, if any VIF corresponding to the variables is greater than 2.0
False, otherwise
if (not vif_bool**)**
Begin
Fit an OLS model to the data using the variables in *combo*
Calculate the p-values for the predictor variables
p_bool = True, if any p-value is greater than 0.05
False, otherwise
if (not *p_bool***)**
Begin
Accept = True, and save *combo* as a possible predictor set.
End
End
End

We applied our selection criteria using Algorithm 8.1 and were able to reduce 1048576 possible combinations of weather variables to 1872 different sets. Table 8.3 shows the R-squared (R^2) and the AIC values when individual weather variable is used as predictor in the OLS model. R-squared (R^2) value reflects the fit of a model. It values range from 0 to 1, where a higher value generally indicates a better fit. We also calculated the Akaike InformationCriterion (AIC) [129] score each time. AIC is based on a technique that performs in-sample fit and estimates the likelihood of a model to predict future values. Generally a good model will have the minimum AIC among all other models since AIC is low for models with high log-likelihoods.

R-squared	AIC	Variable
0.42	12050.32	rh
0.66	11674.81	eto
0.56	11848.03	rso
0.48	11968.46	ea_avg
0.29	12194.89	td_avg
0.58	11824.33	$co2\_avg$
0.56	11861.98	ppf_avg
0.57	11844.82	$solarmj$
0.59	11798.34	$airt\_avg$
0.6	11782.36	$lwdn\_avg$
0.59	11808.35	$lwup\_avg$
0.58	11813.62	$pressure$
0.59	11806.8	$swdn\_avg$
0.56	11856.31	$swup\_avg$
0.62	11758.35	$winds\_avg$
0.52	11916.47	$netrad\_avg$
0.61	11777.99	$surfacet1\_avg$
0.63	11741.08	$surfacet2\_avg$
0.59	11802.22	$visibilitykm\_avg$
0.58	11813.63	$pressurekpasealevel$

Table 8.3: R-squared and AIC score when individual weather variable is used as predictor for building a regression model using data from June and July 2018. The dependent variable is the bee motion counts.

We can see in Table 8.3 that when individual weather variable is used as a predictor, the highest R-squared (R^2) value is 0.66 in case of *eto*. The above value of 0.66 means that 66% of the observed variation in the forager activity can be explained by the variation in *eto*. Thus we believe that if we combine other weather variables, we should be able to explain the observed variation in the forager activity better, which would in effect result in higher R^2 value. Hence in the next step, we sorted the 1872 sets of predictor variables depending upon the R^2 value of the corresponding model.

Table 8.4 shows the 10 best performing models in terms of R^2 value. From the table we can see that *eto* is present as a predictor in 9 of the 10 best performing model. Thus we can see that *eto* has the highest effect on forager activity which we also saw in Table 8.3. The above makes sense because *eto* is a derived variable and is calculated using other meteorological variables and is a key element in the water cycle on the planet and accounts for 15% of the water vapor in the atmosphere. Hence we believe that bees have a way to sense the evapotranspiration present in the atmosphere which in turn has an effect on their foraging activity. We can see in Table 8.4, that the model with highest R^2 value, has *eto*, *ea_avg*, *airt_avg*, *winds_avg*, *visibilitykm_avg*, *pressurekpasealevel* as the independent or predictor variables. We know from previous studies [71, 72, 95] regarding the effect of temperature on foraging activity, and we also see *airt_avg* as one of the predictor variables in the best performing model. Another interesting observation we see is the inclusion of visibility as one of the predictor variables. This is particularly significant because during the summer heat, big wildfires are common in the western parts of USA, which results in low air quality and low visibility in Logan, UT as well. We did not have a way to measure the air quality index, but it was interesting to see *visibilitykm_avg* playing a part in the decision making process of the bees regarding leaving or returning to their hives. We also see *ea_avg* as one of the predictor variables. *ea_avg* or vapor pressure is another way of measuring the humidity of the air. It measures how much water vapor the air would contain if saturated. At a given temperature, an increase of water vapour in the air corresponds to an increase in the humidity of the air. If we refer to Table 8.1, we can see *rh* or relative humidity as one of the weather variables being tested. Relative humidity compares the the absolute humidity (measures how much water vapor the air actually contains) and the vapor pressure. We can see that *rh* is present in 5 of the models in Table 8.4. Hence in a way we can say that humidity has an effect on foraging activity as was reported in [72].

Predictor List	**R-squared (R^2)**	**AIC**
eto ea_avg airt_avg winds_avg visibilitykm_avg pressurekpasealevel	0.7171	11546.36
eto ea_avg airt_avg pressure winds_avg visibilitykm_avg	0.7170	11546.41
eto td_avg airt_avg winds_avg visibilitykm_avg pressurekpasealevel	0.7169	11546.72
eto td_avg airt_avg pressure winds_avg visibilitykm_avg	0.7169	11546.77
rh eto ea_avg winds_avg visibilitykm_avg pressurekpasealevel	0.7167	11547.35
rh eto ea_avg pressure winds_avg visibilitykm_avg	0.7166	11547.38
rh eto td_avg winds_avg visibilitykm_avg pressurekpasealevel	0.7166	11547.61
rh eto td_avg pressure winds_avg visibilitykm_avg	0.7165	11547.64
rh eto ea_avg co2_avg winds_avg visibilitykm_avg	0.7156	11549.88
eto ea_avg co2_avg airt_avg winds_avg visibilitykm_avg	0.7155	11550.17

Table 8.4: The predictor list, R-squared values and the AIC for the top 10 best performing models in terms of R-squared value.

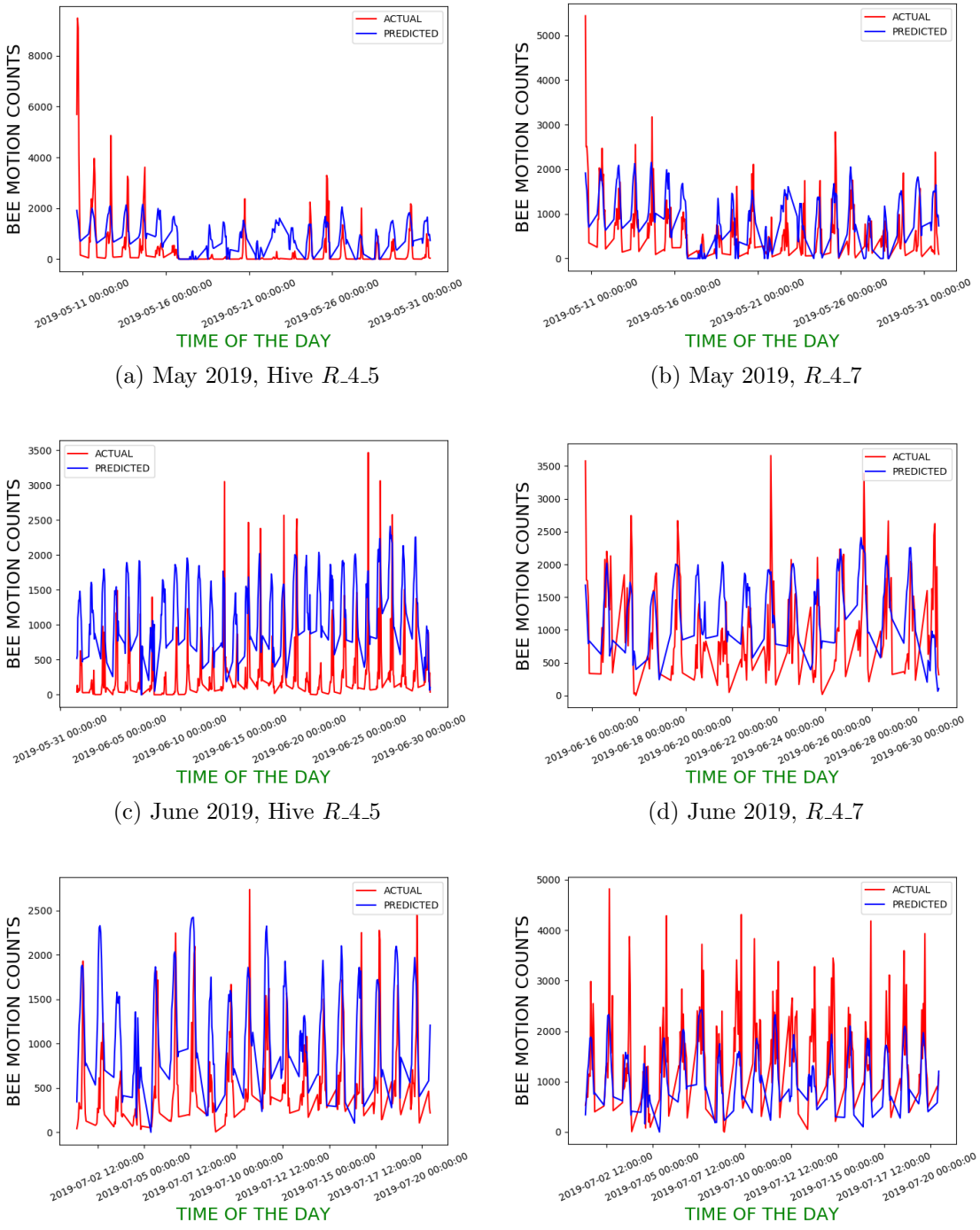
VIF	Variable
1.41	*eto*
1.21	*ea_avg*
1.41	*airt_avg*
1.39	*winds_avg*
1.12	*visibilitykm_avg*
1.34	*pressurekpasealevel*

Table 8.5: VIF values for the best performing predictor set from Table 8.4.

In the next few steps we will use the best performing predictor set and build our first linear regression model ($LR1$). In other words we will fit an OLS linear regression model ($LR1$) to the data, using *eto*, *ea_avg*, *airt_avg*, *winds_avg*, *visibilitykm_avg*, *pressurekpasealevel* to predict bee motion counts or the foraging activity. If we compare Table 8.4 and Table 8.3, we can see that when we combined multiple variables the model was a better fit as reflected in the R-squared value and AIC value. In Table 8.5, we see the VIF values corresponding to the predictor set of the best performing model.

We tested our first model $LR1$ on data from the bee keeping season of 2019, specifically we tested for the months of May, June and July for hives R_4_5 and R_4_7. We evaluated the performance of the model by comparing the predicted bee counts against the actual bee motion counts and then calculating the mean absolute error per day along with the overall root mean squared error (RMSE).

In Figures 8.19a and 8.19b for the month of May 2019, we see a big spike in bee motion counts at the beginning of the month. The reason behind the huge spike is due to the fact that a fresh hive was installed in both the cases. As the days progressed in that month, we see the bees settling down and the bee motion counts becoming smaller. We can see that our predicted model was able to capture the effect if we refer to the predicted bee motion count curves. The mean error for both predictions in Figures 8.19a and 8.19b is reported in the 'Error' column in Table 8.6. We see that the values are less than 250 in both the cases



(a) May 2019, Hive R_4_5

(b) May 2019, R_4_7

(c) June 2019, Hive R_4_5

(d) June 2019, R_4_7

(e) July 2018, R_4_5

(f) July 2018, R_4_7

Figure 8.19: A linear regression model $LR1$ was fitted to the data from the months of June and July 2018 using *eto*, *ea_avg*, *airt_avg*, *winds_avg*, *visibilitykm_avg*, *pressurekpasealevel* as predictors with bee motion counts as the response or dependent variable. The prediction was performed on data from the months of May, June and July 2019. Actual/ Measured (red) bee motion counts are presented in the same graph as the predicted (blue) bee motion counts across the months of May, June and July during the bee keeping season of 2019 for hives R_4_5 and R_4_7.

for the month of May. This means the error in the bee counts predicted by the model $LR1$ was within 250 bees of the original bee motion counts per day. In Figures 8.19c and 8.19d for the month of June 2019, we see that the prediction curve is closer to the original bee count curve. It is especially closer in Figure 8.19d, where we see that apart for some sudden spikes, most of the days in the month the prediction was closer to the actual count. It is reflected in Table 8.6 as well, where the mean error was 218.27 for the month of June in case of hive R_4_7.

Hive	**Month**	**RMSE**	**Error**
R_4_5	May	1073.47	161.35
R_4_5	June	608.54	371.07
R_4_5	July	518.99	248.64

Hive	**Month**	**RMSE**	**Error**
R_4_7	May	684.97	247.43
R_4_7	June	748.47	218.27
R_4_7	July	905.07	394.51

Table 8.6: The tables show the RMSE and the mean error for the respective graphs in Figure 8.19 for hives R_4_5 and R_4_7. The error in the fourth column is the mean absolute error per day. In other words, it is the mean of the absolute difference per day between the actual bee motion counts and the number of bees predicted by our model $LR1$.

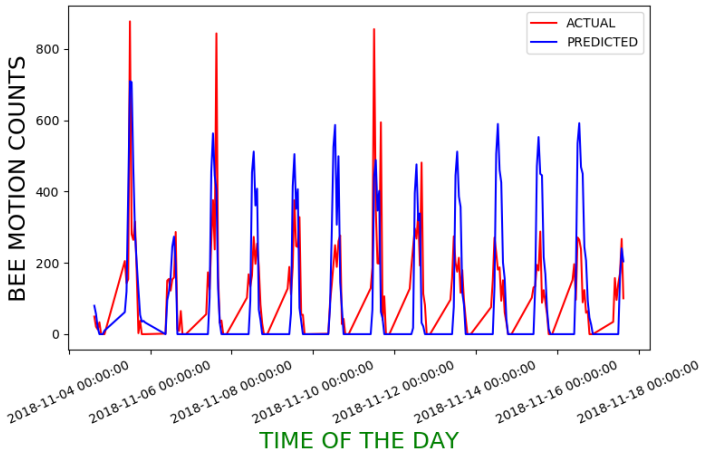
In Figure 8.19e, we see the shape of prediction curve matches closely to the actual bee count curve over the entire month of July 2018 for hive R_4_5. In Figure 8.19f, we see that during certain days in July when the bee count was high, the model $LR1$ was not able to predict those higher counts effectively. This resulted in higher RMSE of 905.07, which can be seen in Table 8.6. Thus we can see the power of a simple model wherein by using a simple linear regression model $LR1$ we were able to match the original bee motion counts within a certain error percentage. A more complex model with quadratic predictors might lower the error percentage and generate better results. But in this study, our objective was to investigate whether it is possible to use different meteorological variables (in addition to the ones previously studied in the literature) to explain observed bee foraging activity and we have seen for our hives evapotranspiration, visibility play an important role too alongside temperature and humidity.

8.8 Using Regression Model To Study Hive Health

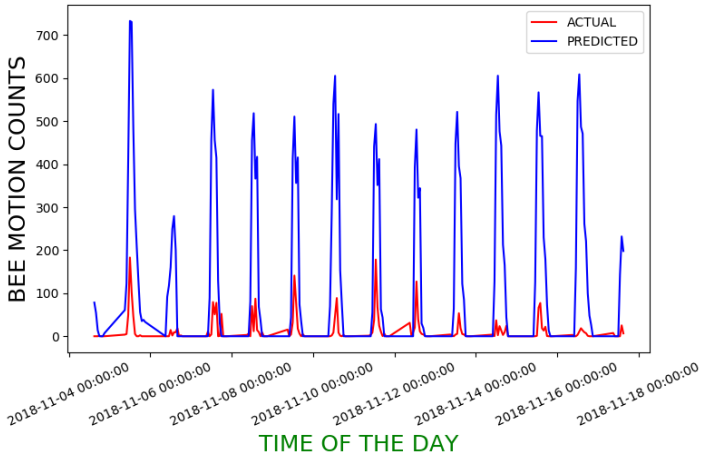
One of the important aspect for a beekeeper is to monitor the health of the hives during a bee keeping season. There could be several factors behind a hive not developing properly in comparison to others. We saw in the previous chapter that during the timeline of end of September–beginning of October the performance of both of our hives R_4_5 and R_4_7 decreased simultaneously showing almost a similar downward trend. There could be various reasons for this downward trend, but we believe one such reason could be attributed to the local weather conditions. During this time of the year, the temperature starts to go down as fall arrives in Logan, UT. During mid October–end November, which are months in the back end of the beekeeping season, the temperature falls below freezing occasionally. So it is expected to see less forager traffic during this time as the bees prepare their hives for the winter. But if a hive is struggling it would be reflected in the forager activity too as there will be less to no movement with significant drop in foraging activity during these times, hence telling us that the hive is close to dying. Between hives R_4_5 and R_4_7, we know that hive R_4_7 was not able to survive the winter. Thus in this section we will see if it is possible to identify the hive's struggle beforehand through a prediction model using only meteorological variables.

As we discussed earlier regarding the decrease in forager activity during the timeline of end of September–beginning of October, it would be wise to design a new linear regression model that would be trained on data during that particular period. Towards that end we design a new regression model $LR2$, using eto, ea_avg, $airt\_avg$, $winds\_avg$, $visibilitykm\_avg$, $pressurekpasealevel$ as predictors, and bee motion counts as the response or dependent variable. We train the model on data from the hive R_4_5 during the month of October 2018. The reason behind selecting hive R_4_5 is that the hive survived the winter and hence we have chosen the foraging activity observed in that hive to be the reference point. The training data initially had 1612 $(31 \times 13 \times 4)$ data points for the whole month of October 2018 from hive R_4_5. Since the weather data is timestamped every hour, we had to reduce our dataset of bee motion counts to 1 value per hour. We did this by calculating

the mean of the bee motion counts every hour. Hence the final training set had (31×13) 403 data points. We tested the model on data from both the hives R_4_5 and R_4_7 during the month of November 2018.



(a) November 2018, Hive R_4_5

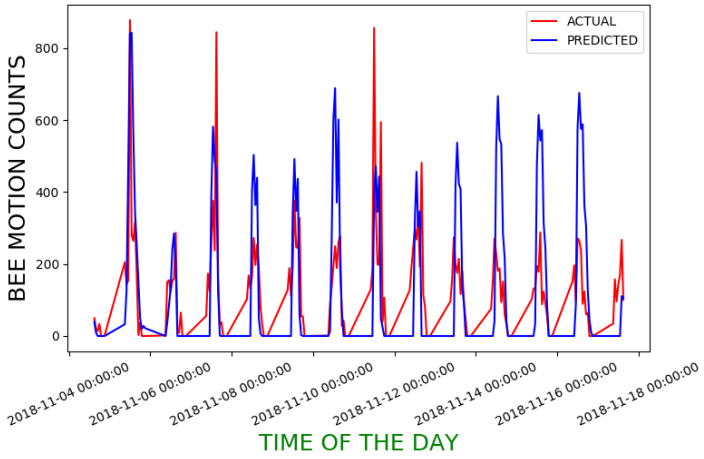


(b) November 2018, R_4_7

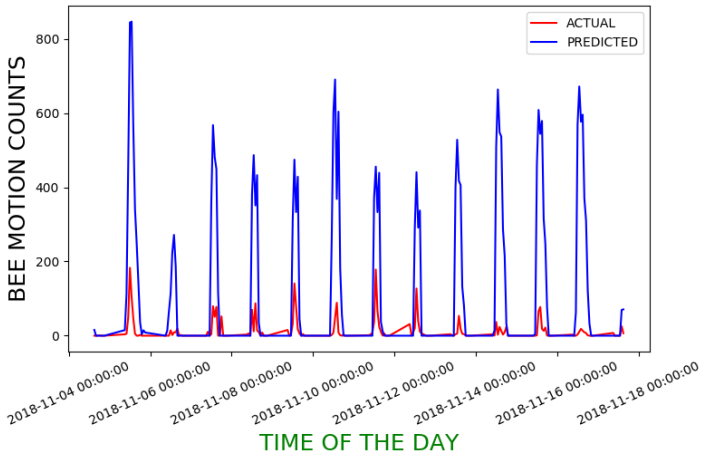
Figure 8.20: A linear regression model $LR2$ was fitted to the data from hive R_4_5 during the month of October 2018 using *eto*, *ea_avg*, *airt_avg*, *winds_avg*, *visibilitykm_avg* and *pressurekpasealevel* as the predictors, and bee motion counts as the response or dependent variable. The prediction was performed on data from both hives during the month of November 2018. Actual/ Measured (red) bee motion counts are presented in the same graph as the predicted (blue) bee motion counts across the months of November during the bee keeping season of 2018 for hives R_4_5 and R_4_7.

In the graphs in Figure 8.20, we can clearly see the difference in the forager activity between the hives R_4_5 and R_4_7. In Figure 8.20a we see that the predicted bee motion counts and the actual bee motion counts match very closely for hive R_4_5. This means the hive was performing normal as expected towards the end of the bee keeping season. But on the other hand, we see in Figure 8.20b, the predicted bee counts are a lot larger than the actual counts for hive R_4_7. This tells us that the hive R_4_7 was not performing as it should have towards the end of the beekeeping season. This event was also mirrored by our observation recorded in the beekeeping journal, wherein we noted that hive R_4_7 was not able to survive the winter and hence died. Thus we can see that using meteorological variables we were able to describe the performance of a hive by predicting the forager activity and comparing against actual bee motion counts. To remove any bias in our results, we did the same analysis as above by training a different model $LR3$ on data from hive R_4_7 during the month of October 2018 and tested the model on data from November 2018. The results can be interpreted similarly as above and we can clearly see in Figure 8.21b, that hive R_4_7 was struggling.

A way to analyze this difference is to use a metric which will compare the model predictions with the actual bee motion counts generated by our dpiv based algorithm. Since the bee population changes depending upon the time or the stage during the bee keeping season, the metric needs to be calibrated accordingly. For example the allowable difference between the predicted counts and the actual counts could be higher during June or July or even August, since there is a lot of foraging activity or bee movement during this time. But the acceptable difference needs to be small during November, since we already know forager activity goes down a lot during the final stages of the beekeeping season. Hence we believe that a carefully designed metric that could track continuous discrepancies over time would help a beekeeper identify abnormal behaviour such as bees abandoning their hives, swarming or hive dying due to parasitic infestation or other hive threatening events.



(a) November 2018, Hive R_4_5



(b) November 2018, R_4_7

Figure 8.21: A linear regression model $LR3$ was fitted to the data from hive R_4_7 during the month of October 2018 using *eto*, *ea_avg*, *airt_avg*, *winds_avg*, *visibilitykm_avg* and *pressurekpasealevel* as the predictors, and bee motion counts as the response or dependent variable. The prediction was performed on data from both hives during the month of November 2018. Actual/ Measured (red) bee motion counts are presented in the same graph as the predicted (blue) bee motion counts across the months of November during the bee keeping season of 2018 for hives R_4_5 and R_4_7.

8.9 Using Bee Buzzing Intensity

In the previous chapter we learned about calculating the buzzing intensity from audio recordings and how we can comment on the health of a hive by analyzing those buzzing

intensities over time. From the Figures 8.1 and 8.2 we can tell that the buzzing intensities and the foraging activity are positively correlated and the correlation value was above 0.50 for majority of the days for both hives R_4_5 and R_4_7 during our investigation period. In this section we will investigate whether we can use the buzzing intensity alongside the weather variables to predict future foraging traffic. Towards that end, we fitted a linear regression model $LR4$ on the data from hive R_4_5 during the month of June and July 2018 (same data as in the previous sections) using buzzing intensity alongside the weather variables, to predict bee motion counts. We saw in Table 8.4, that the predictor set of *eto*, *ea_avg*, *airt_avg*, *winds_avg*, *visibilitykm_avg* and *pressurekpasealevel* resulted in the best performing model with the highest R^2 value. The first step is to calculate the VIF after adding buzzing intensity to the above predictor set. We can see in Table 8.7, that all the VIF corresponding to each variable is less than 2. But when we fit model $LR4$ to our training data, the p-values corresponding to $POWER$ and *airt_avg* are greater than 0.05. This tells us that both the variables are not statistically significant in the model. Hence simply adding buzzing intensity to the best performing predictor list was not a good idea.

VIF	**Variable**
1.09	$POWER$
1.41	*eto*
1.22	*ea_avg*
1.51	*airt_avg*
1.39	*winds_avg*
1.12	*visibilitykm_avg*
1.34	*pressurekpasealevel*

Table 8.7: VIF values for after adding buzzing intensity to the best performing predictor set from Table 8.4.

From Table 8.1, we can see that there are 20 different weather variables and we are now

adding buzzing intensity to the list. Hence if we consider every possible combinations of the variables we will have $2^{21} = 2097152$ different combinations. We could have used PCA to reduce the dimensions of the data or perform feature importance using Random Forest and reduce the number of features from 21, but for completeness we did an exhaustive analysis involving all the variables. So we followed Algorithm 8.1 for selecting only the relevant set of predictor variables out of the 2097152 possible combinations that satisfied the VIF and p-value criteria. As a result we were able to reduce 2097152 possible combinations of weather variables to 3179 different sets.

Table 8.8 lists the 10 best performing predictor sets which had $POWER$ as one of the predictors. We can see in Table 8.8 that *eto* is present in all of the sets which tells us about the importance of *eto* in predicting forager traffic. We fit a linear regression model $LR5$ to the data using the top predictor set in Table 8.8, but we can see that adding $POWER$ or the buzzing intensity to the list of predictor variable did not improve the R^2 value of the model, rather the value is slightly lower than the R^2 value of the best performing model in Table 8.4. It is important to know that R^2 value does not always indicate how well the model fits to the data. Even if a model has a low R^2 value, we can still draw important conclusions about the relationship between the variables as long as the independent variables are statistically significant (satisfied in our study following Algorithm 8.1). High R^2 values are important when we need to generate almost precise predictions. But in our case our goal is not to generate precise predictions rather a general estimate of the forager traffic.

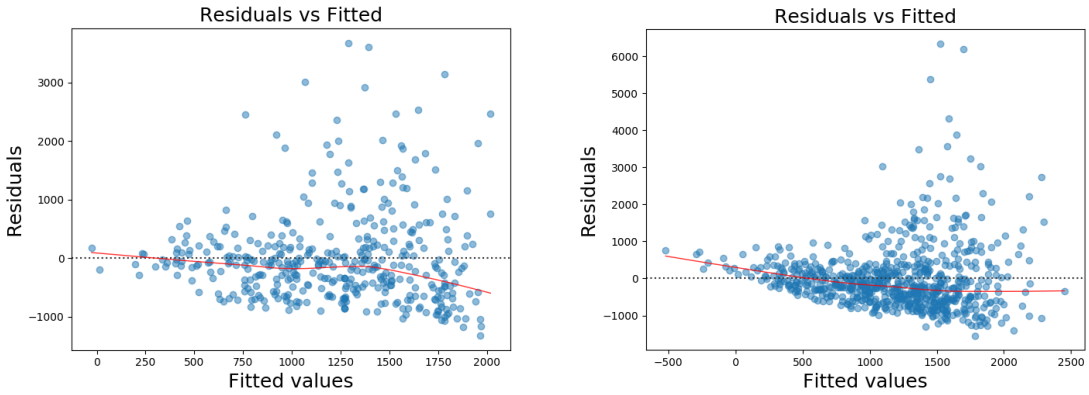
Next, in order to compare the performance of the models $LR1$ and $LR5$, we plot the corresponding diagnostic plots to visually determine how our models fit the data and if any of the basic assumptions of an OLS model are being violated. Each diagnostics plot will focus on the residuals (or errors) generated by a model, which is basically the difference between the actual value and the predicted value. The basic assumptions of the OLS model are: the data can be fit by a line, any errors during model fitting are normally distributed with mean zero and all errors have constant variance.

Predictor List	**R-squared (R^2)**	**AIC**
POWER eto td_avg winds_avg visibilitykm_avg pressurekpasealevel	0.7151	11551.27
POWER eto td_avg pressure winds_avg visibilitykm_avg	0.7151	11551.3
POWER eto ea_avg winds_avg visibilitykm_avg pressurekpasealevel	0.7150	11551.42
POWER eto ea_avg pressure winds_avg visibilitykm_avg	0.7150	11551.45
POWER eto ea_avg co2_avg winds_avg visibilitykm_avg	0.7141	11553.62
POWER eto td_avg co2_avg winds_avg visibilitykm_avg	0.7139	11554.22
POWER eto td_avg airt_avg visibilitykm_avg pressurekpasealevel	0.7130	11556.49
POWER eto td_avg airt_avg pressure visibilitykm_avg	0.7130	11556.53
POWER eto ea_avg airt_avg visibilitykm_avg pressurekpasealevel	0.7125	11557.79
POWER eto ea_avg airt_avg pressure visibilitykm_avg	0.7125	11557.82

Table 8.8: The predictor list, R-squared values and the AIC for the top 10 best performing models in terms of R-squared value, when buzzing intensity was added along with the weather variables

First we analyzed the 'Residuals vs Fitted' plots for both the models. This type of graph shows if there are any nonlinear patterns in the residuals, and thus in the data as well. One of the mathematical assumptions in building an OLS model is that the data can

be fit by a line. If this assumption holds and our data can be fit by a linear model, then we should see a relatively flat line when looking at the residuals vs fitted. An example of this failing would be trying to fit the function $f(x) = x^2$ with a linear regression $y = C + M_1 * X_1 + M_3 * X_2 + M_3 * X_3 + \ldots + \epsilon$. Clearly, the relationship is nonlinear and thus the residuals will have non-random patterns.



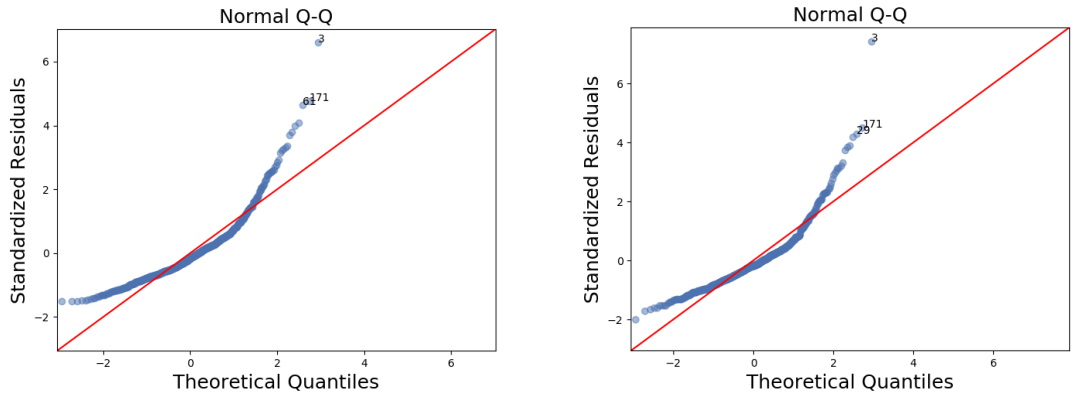
(a) Model $LR1$ (b) Model $LR5$

Figure 8.22: Residual vs Fitted plots for model $LR1$ and $LR5$.

In an ideal 'Residuals vs Fitted' plot, the red line in Figure 8.22 would be horizontal. This is due to the fact that, for a perfect fit, there should not be any pattern or clustering in a 'Residuals vs Fitted' plot, rather it should be spread more like randomly around the zero x-axes(in an uniform way). In Figures 8.22a and 8.22b, neither of the red lines are horizontal. If we observe the plot in Figure 8.22b, the bow shaped red line indicates that our simple linear model has failed to capture some of the non-linear features that could be present and hence the model has resulted in underfitting. One way to address that is to transform the predictor variables in a way such that the variance in the data could be better captured. The transformation could be performed by squaring (or some other non-linear transformation) of one or more of the predictor variables.

Next, in order to check if all the errors or the residuals are normally distributed, we plot the corresponding Q-Q plot. An ideal Q-Q plot will have all of the residuals lying in

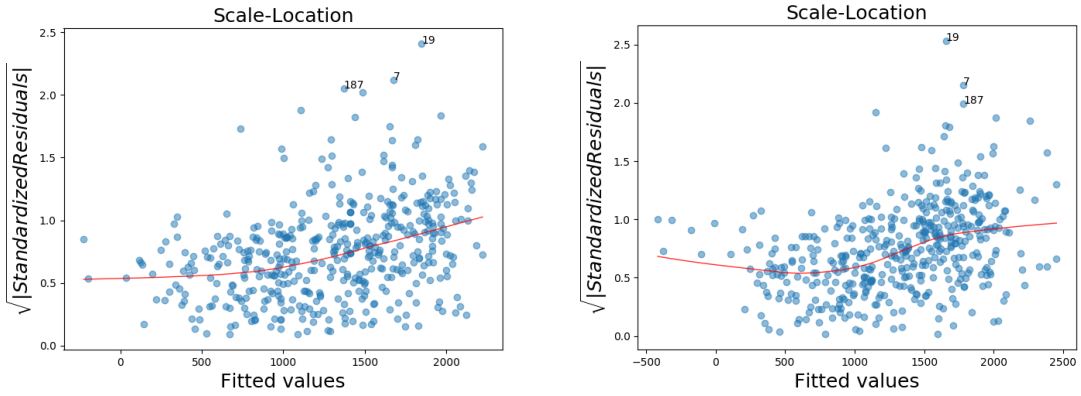
or close to the red line. If we look at the plots in Figures 8.23a and 8.23b, we see that there are number of points that fall further away from the red line. Although in Figure 8.23b the points are comparatively closer to the red line, but still we can see in both the cases the errors not being normally distributed.



(a) Model $LR1$ (b) Model $LR5$

Figure 8.23: Q-Q plots for model $LR1$ and $LR5$.

When we plotted the the residuals vs the fitted response values (as per the model), we observed that the variance of the residuals increase with response variable magnitude. So the next step is to visually check if the residuals suffer from non-constant variance or heteroscedasticity. For that we will plot our fitted values against residuals values which are standardized (i.e. mean=0 and variance=1). The more horizontal the red line is, the more likely the data is homoscedastic i.e. the errors will have constant variance. We can see in Figure 8.24, that neither red line is horizontal. This might be caused due to the fact that the models were not able to capture the non-linearities and hence this tells us that further tweaking of both the models is necessary.



(a) Model $LR1$ (b) Model $LR5$

Figure 8.24: Plots to check heteroscedasticity for models $LR1$ and $LR5$.

On visual observation of the diagnostic plots we can see that the residuals for $LR5$ were more normally distributed in comparison to $LR1$. This tells us that the model $LR5$ was a better fit. We also learned from the plots that both models were not able to capture the non-linearity in the data and thus further tweaking of the model with different non linear transformation of the predictor variables are needed to achieve better model fit.

8.10 Discussion

One important aspect we need to keep in mind regarding our study is that the weather data is obtained from a weather station that is almost 2 miles away from the apiary. Thus it might not always reflect the local conditions near the hive which might affect the forager activity differently, thus resulting in over prediction or under prediction by our model. The goal of this study was not to design a perfect model, but was to investigate by designing a simple linear model, whether local weather conditions have an effect on foraging activity. In particular this study was also in part to validate our bee motion counting algorithm presented in Chapter 5. Previous researchers have used additional costly intrusive hardware to get an estimate of bee counts which were then used for the weather analysis. But here we have used our designed bee motion counting algorithm which counts the bee motions insitu. Next we have seen, weather conditions have an effect on foraging activity and the results

suggest that weather conditions particularly temperature, humidity, evapotranspiration and surrounding visibility are continuously monitored by the bees in making decisions regarding taking a flight outside of the hive or not. On the other hand, we can also assume a similar decision making process is constantly happening outside the hives as well, where the bees decide whether to return back to their hives depending upon the weather conditions. We also designed a different linear regression model by including the buzzing intensity captured by the microphone of our BeePi sensor as one of the predictors and did not find any significant improvement in the performance on our data. We have seen in our analysis in Section 8.5 that bees can sense instantaneous weather changes and thus behave accordingly rather than holding information about previous weather conditions. This tells us one important aspect, in which we can hypothesize that no complicated neural network with millions of parameters is needed to implement a system that can replicate how bees think. Rather just a simpler pre–trained model would be sufficient. Thus such low storage and processing requirements would help us do these analysis on a low powered device such as a Raspberry Pi. We believe our analysis is just a starting point, and it would require further investigation to understand the response of bee colonies to changing climatic conditions and how it effects pollination.

We designed and validated our models using data from only 2 hives. Although we validated our model on data from a different year with a completely different bee race, still a large scale study is necessary with data from lots of hives. Data recordings in our new BeePi system have been seamless starting with the bee keeping season of 2020 and thus we plan to obtain good quality data from each of our hives in the apiary. This would help us mix data from several hives in order to efficiently train our models and in effect help the models generalize better. This study used data from weather station located 2 miles away. Hence in the next step we plan to incorporate data from our own weather stations located in close proximity to the hives in the same apiary. This would help us to improve our evaluation of truly local weather condition on foraging activity.

CHAPTER 9

CONCLUSION

Honeybee forager traffic, which is the number of bees that move in close proximity to a beehive, acts an important factor to indicate how honey bee colonies react to external conditions like foraging opportunities, weather events, pests and diseases. Many commercial and amateur beekeepers observe the bee traffic near their respective hives to identify the state of the corresponding bee colonies, since it is believed that honeybee traffic carries information on colony behavior and phenology. But commercial beekeepers have to manage several hundreds of beehives. Therefore, it is not always possible for them to be physically present to observe the beehives in time to administer appropriate treatment. According to the report published by the United States Department of Agriculture [130], it has been estimated that the cost to the commercial beekeepers to replace lost colonies was around $2 billion between the years 2006–2012. We believe that if the labor cost of monitoring the hives can be reduced by automating the process, along with early detection of stressors (for example, failing queens, predatory mites, airborne toxicants etc), then the complete loss of honeybee colonies can be averted, which in turn may have a significant economic impact on the industry.

In this dissertation we have seen how electronic beehive monitoring (EBM) can be used to extract useful information on bee colony behavior and hive health without invasive beehive inspections and transportation costs. In our research we have conducted various tests and have proposed diagnostic models to address different aspects of electronic beehive monitoring. We have seen that it is possible to computationally estimate bee traffic patterns and correlate them to beehive development through an integration of audio and video analysis. This estimation and correlation were achieved by collecting and analyzing audio and video data in the vicinity of each monitored beehive. We also investigated the connection between foraging activity of honey bees and local weather conditions and dis-

cussed its potential application in continuous and remote monitoring of honey bee colonies. Our designed approach has several advantages. It helps in the reduction of invasive hive inspections that disrupt the colony's life cycle and put stress on the bees. It helps reduce the transportation costs for the beehive owners who travel long distances to their beehives. One of the primary goals of this dissertation was to design an EBM system that required no structural modifications to a standard beehive (such as the Langstroth beehive [14] or the Dadant beehive [15]), thereby preserving the sacredness of the bee space without disturbing the natural beehive cycles. In this dissertation we have been able to build a reliable and low–cost EBM system based on raspberry pi that does not require any structural modifications to be made and can just be placed on top of the beehive to be monitored. Through our designed system, we were also able to collect good data, which has been used in building diagnostic models to investigate the health of a hive.

In Chapter 3, we presented an algorithm to count bees on the landing pad of a Langstroth beehive [14] from static images recorded by a raspberry pi camera. The camera was placed just above the landing pad and it recorded static images of forager traffic leaving and entering the hive. The above static images were first passed through a pad localization module to localize the landing pad and were then processed by a skew detection module where the pad's skew angle was determined and the images were rotated accordingly. The above localized and rotated images of landing pads with bees were given to our designed bee counting module which returned a non-negative real number approximating the number of bees on the pad. We evaluated the performance of our presented method on 793 images of landing pads with bees and for 90.29% of the images we were within an error margin of 15 when compared to the ground truth or actual counts.

In our continuing research in [1], we presented video bee traffic analysis where we analyzed 30 seconds of videos captured by the raspberry pi camera. We contributed to the body of research on object recognition in videos by proposing a two-tier system that couples motion detection with image classification. Our system used motion detection (KNN [131], MOG [132], MOG2 [133]) to extract possible object regions and then classified each detected

region using a class-specific classifier (e.g., a Convolutional Neural Network or a Support Vector Machine) or an ensemble of classifiers such as a random forest (RF) [115]. In [1], we also presented a strategy to automate the design of Convolutional Neural Network (CNN) architectures to classify each detected motion region. We designed a multi-generational greedy grid search (GS) method to automate the finding of the best performing CNN architectures with 1, 2, 3, 4 and 5 hidden layers respectively by constraining the number of convolutional and maxpooling layer pairs (i.e., hidden layers), filter sizes, and the number of nodes in each hidden layer. We trained our models on 19083 Bee Images and 19057 No-Bee Images. For model accuracy, we tested our models on 6362 Bee Images and 6362 No-Bee Images. For model selection, we tested our models against a validation data (the held-out testing dataset) set of 1810 Bee images and 1718 No-Bee images. We also introduced three image datasets (BEE1 [24], $BEE2\_1S$ [25], $BEE2\_2S$ [26]) for our analysis. We found out that the best performing models generated from the automated design of CNN architecture performed on par with the popular image classification CNN architectures of ResNet32 [134] and VGG16 [80] on BEE1 and $BEE2\_1S$ and even better in case of $BEE2\_2S$. Through our above two studies, we experimentally demonstrated that computer vision and deep learning, which had so far not been explored in EBM systems ([10, 73, 75, 76, 135]), could be put to productive use in estimating forager bee traffic levels through by estimating bee motion counts. In that same study we also made a preliminary evaluation of the proposed two-tier method on four 30-seconds videos with different levels of bee traffic. We manually labeled each frame from the video and counted the bees that moved between successive frames. We then evaluated our two-tier system on the four videos to count the number of bee motions using MOG2 [132] for motion detection. We obtained good result on a video that had low traffic, but overestimated in case of mid and high traffic videos mainly due to detected overlapping motion regions.

The two-tier bee counting method in [1] was only able to count omnidirectional bee motions in that it did not distinguish between incoming and outgoing bee traffic. So in our following research [2] we addressed the above by introducing a bee motion estimation

algorithm using digital particle image velocimetry (dpiv) to measure levels of directional traffic along with omnidirectional bee motions. The dpiv based approach did not rely on image classification and hence could be used to measure traffic of any other insects such as bumble bees and ants. Since dpiv is purely analytical, it does not require the use of any machine learning or large curated datsets. On the same four evaluation videos as above, the dpiv based bee motion estimation algorithm [2] performed on par with the two-tier bee motion counting algorithm [1] in measuring omnidirectional honeybee traffic. Since our goal is to have the BeePi system process the recorded videos on the raspberry pi itself, the advantage of the dpiv based approach was apparent when the running time was compared to the two-tier method. We distributed the dpiv processing over 6 different raspberry pis and found that it took ≈19.49 min to process a 30-sec video of resolution 640×480 at 25fps. On the other hand, we were not able to distribute the two-tier method and hence it took more than 2.5hrs on a single raspberry pi.

We continued our research [22] by improving the design of the dpiv based algorithm to count bee motions in Chapter 4 and Chapter 5. In order to improve the dpiv based bee counting method, the first step we took was to reduce the size of the video frames from (640×480) to (80×60). In Chapter 4, we saw how we could take frames with bees and convert them into a frames with white background having only the positions of interest marked, i.e. identifying the bees that have moved between successive frames and then separating the corresponding positions on a different frame with a white background. In Chapter 5, we combined the above bee movement localization technique with the dpiv based algorithm to count bee motions between successive frames. We showed how we could use dpiv to analytically classify and count different bee motions into incoming, outgoing or lateral and then use those individual counts as measurements of directional traffic levels. We added 28 more videos to our video evaluation dataset, which were taken from different times during the beekeeping season with varying levels of bee traffic across different backgrounds. To generate the ground truth data, each of the frames in the 32 videos (744*32=23808 frames) were individually evaluated before hand and the number of bees that moved in successive

frames were counted from them. The data is publicly available at [96]. In Chapter 5, we saw the combined dpiv based bee motion count algorithm along with bee movement localization performed better than the two-tier bee motion counting algorithm proposed in our previous research [1] in terms of omnidirectional bee motions based upon the results on 32 evaluation videos. It also improved upon our previous study [2], where in we were able to get actual counts of bee motions rather than a general estimate. For the 32 evaluation videos we saw that the combined dpiv method presented in Chapter 5 was able to generate results which were closer to the actual bee counts. We also saw that our improved dpiv based method was able to run on a single raspberry pi computer and was able to process each video in ≈2.15 minutes. Since the BeePi system records video every 15 minutes, we envision that in the future BeePi monitors will use the remaining time (15-2.14=12.86 minutes) to process and analyze data from other sensors as well. In Chapter 5, we also investigated whether the newly proposed approach could be used to analyze the incoming and outgoing traffic levels in healthy beehives. We also compared the time series data for the bee motion counts during the entire beekeeping season of 2018 from May to November for two of our hives R_4_5 and R_4_7 and observed some interesting events concerning the health of the hives.

In Chapter 6, we focused our attention towards analyzing audio samples recorded by the BeePi monitors. We discussed the design of a CNN architecture that processed raw audio waveforms and compared the performance of different deep learning models towards classifying audio samples recorded by microphones on beehives. We introduced two different datasets of manually labeled audio samples to aid us in our investigation and evaluation. In the first dataset BUZZ1 [23] with 10,260 audio samples, the training and testing samples were separated from the validation samples by beehive and location. In the second dataset, BUZZ2 [23] with 12,914 audio samples, the training and testing samples were separated from the validation samples by beehive, location, time and bee race. We saw in Chapter 6 that the CNN model we designed to classify raw audio waveforms performed better than the four machine learning methods [21] and a CNN model trained to classify spectrogram images of audio samples. We also experimentally showed that our CNN model designed to

process raw audio waveforms is able to operate in situ on a raspberry pi computer.

Continuing with our audio analysis in Chapter 7, we used band-pass filtering to remove any background noises and filter out the relevant frequencies that are associated with bee buzzing from the audio samples. Examples of background noises can be bird chirping, very low intensity of human conversation, sudden spike of audio pulse or a constant noise produced due to hardware issues in audio recording arrangement. We found in our literature review that the frequency range of 200–3000Hz was able to capture most of the honey bee signals. We filtered out the above frequencies which are associated with bee buzzing from audio recordings and then used those filtered audios to find the power/intensity of bee buzzing. We then analyzed the power/intensity of the bee buzzing over the entire beekeeping season of 2018 from May to November and investigated how two of our hives(R_4_5 and R_4_7) had progressed through the 2018 beekeeping season. We analyzed the time series data of power/intensity of the bee buzzing and observed interesting events during certain times of the season. Specifically we saw that one hive showed better performance in comparison to the other during the middle of the bee keeping season. We envision that in the future a metric would be developed and a dissemination system would be designed which would address such discrepancies and inform the beekeepers regarding the health of the hives, in process acting as an early detection alarm for beekeepers. In a large-scale apiary with many hives equipped with BeePi sensors, those hives that show a downward trend in performance during the middle of the season could be manually inspected and necessary steps for hive treatment could be performed. We also saw the effect of the cold temperature in Logan, UT during the timeline of mid October–end November, on the buzzing intensities in both the hives. Both the hives showed lower buzzing intensities, but the variability in the time series graph told us that one hive was struggling to stay alive which was also confirmed from our beekeeping journal as well.

Following our findings in Chapter 7, we investigated in Chapter 8 the effect of local weather conditions on the forager traffic. We studied the effect of 21 different meteorological variables on the foraging activity during the period of May–November of our beekeeping

season in 2018 and during May–July 2019. Through our study we were able to show that without the use of additional costly intrusive hardware to count the bees, we can use our bee motion counting algorithm to count the bee motions and then use the counts to investigate the relationship or correlation between foraging activity and local weather. We also fitted several simple linear regression models to the data using different combinations of weather variables to predict the bee motion counts and saw that for our hives, evapotranspiration and visibility play an important role alongside temperature and humidity. We found that 71% of the observed variation in the forager activity can be explained by the variation in evapotranspiration, visibility, temperature, pressure and humidity. We also designed separate linear regression models by including the buzzing intensity captured by the microphones as one of the predictors and did not find any significant improvement in the performance. The above results were interesting, because in Figures 8.1 and 8.2 we saw that the median correlation values between forager traffic and buzzing intensity were high positive above 0.65 during the months of June–July 2018. We can also see from Figures 7.20 and 7.21, that the shape of the time series plots of buzzing intensity and forager traffic counts followed each other closely during certain months in the beekeeping season of 2018. Thus the above observations tell us that while buzzing power is not contributing much as a predictor variable, it is definitely highly correlated to forager traffic. Towards that end, we believe that a future study involving the use of weather variables to predict the power/intensity of bee buzzing could be fruitful. Hence we can think of the audio as a redundant sensor variable which could be useful during certain scenarios when the camera fails and we could still be able to get information regarding the status of the hive.

We acknowledge that there are certain open ended challenges associated with our analysis and addressing them as part of future work could help our goal of building an autonomous beehive monitoring system capable of making informed decisions. In Chapter 8, we learned through our investigation that the linear regression models were not able to capture the non-linearity in the data and thus further tweaking of the model with different non linear transformation of the predictor variables are needed to be done as part of future work to

achieve a better model fit. In order to better generalize the predictive models and truly understand the effect of local weather on foraging activity, a large scale study is necessary with data from lots of hives across a wide range of weather conditions. The weather data that we used was from a weather station 2 miles away. Hence in the future we plan to incorporate data from our own weather stations which will be located in close proximity to the hives in the same apiary. We plan to incorporate in our future study, a detailed and manual colony health assessment by an experience beekeeper which could help shine light to certain hive specific events as an effect from different external conditions. The addition of other sensors such as brood temperature, hive weights etc may reveal predictor variables that were not considered in this dissertation. Along with that since our final goal is to build an autonomous monitoring system, we believe that there is a need to develop a metric that would compare the model predictions with the observed bee motion counts, which in turn could lead to better understanding of hive stressors. Differences between the predicted counts and the observed counts could be tracked and appropriate warnings could be sent to the user depending upon the designed metric and the length of the time period over which the discrepancies occurred. Hence a large scale study is important for us to build the confidence in our predictive modeling, which in turn could save labor costs by reducing periodic manual inspections of all hives by a beekeeper in a large apiary.

We also see a scope for improvement in the bee motion localization algorithm proposed in Chapter 4. The localization method is dependent upon the color variations between a moving object and the background. Hence it is susceptible to non-bee movements as well. An additional step could be added to it wherein after the movements have been localized, the objects at those positions could be passed through a pre–trained image classification model. This will help us detect the movements and classify not only honeybees but also other flying or crawling insects or animals near the landing pad of the hive as long as we have a trained model to classify them. The dpiv based method presented in Chapter 5 could then be applied on top of that to get the directional counts. Along with that we are also in the process of increasing our evaluation dataset of 32 videos.

In our study we were not able to record in our beekeeping journal any specific events related to a hive such as queen loss, swarming, etc, which could have been reflected in the audio recordings. Although we were able to analyze the bee buzzing within a certain frequency range, but we were not able to analyze individual honey bee signals such as recruit, tooting, quacking, hissing etc. Thus, we were not able to correlate different honey bee signals to the forager activity for a much broader analysis. We believe as part of the future work, an investigation into the different honey bee signals as indicators of hive performance could give us valuable insights.

REFERENCES

[1] V. Kulyukin and S. Mukherjee, "On video analysis of omnidirectional bee traffic: counting bee motions with motion detection and image classification," *Applied Sciences*, vol. 9, no. 18, p. 3743, Sep 2019. [Online]. Available: http://dx.doi.org/10.3390/app9183743

[2] S. Mukherjee and V. Kulyukin, "Application of digital particle image velocimetry to insect motion: Measurement of incoming, outgoing, and lateral honeybee traffic," *Applied Sciences*, vol. 10, no. 6, p. 2042, March 2020. [Online]. Available: https://doi.org/10.3390/app10062042

[3] A. Qandour, I. Ahmad, D. Habibi, and M. Leppard, "Remote beehive monitoring using acoustic signals," *Acoustics Australia / Australian Acoustical Society*, vol. 42, pp. 204–209, 12 2014.

[4] D. Goulson, E. Nicholls, C. Botías, and E. L. Rotheray, "Bee declines driven by combined stress from parasites, pesticides, and lack of flowers," *Science*, vol. 347, no. 6229, 2015. [Online]. Available: https://science.sciencemag.org/content/347/6229/1255957

[5] Food and A. O. of the United Nations. 2018., "Why bees matter: The importance of bees and other pollinators for food andagriculture," http://www.fao.org/3/i9527en/i9527en.pdf (accessed Oct. 16, 2020).

[6] Food and A. O. of the United Nations. 2016., "The power of pollinators: why more bees means better food," http://www.fao.org/zhc/detail-events/en/c/428504/ (accessed Oct. 16, 2020).

[7] G. Patrcio-Roberto and M. Campos, "Aspects of landscape and pollinatorswhat is important to bee conservation?" *Diversity*, vol. 6, no. 1, p. 158175, Mar 2014. [Online]. Available: http://dx.doi.org/10.3390/d6010158

[8] L. A. Garibaldi, I. Steffan-Dewenter, R. Winfree, M. A. Aizen, R. Bommarco, S. A. Cunningham, C. Kremen, L. G. Carvalheiro, L. D. Harder, O. Afik, I. Bartomeus, F. Benjamin, V. Boreux, D. Cariveau, N. P. Chacoff, J. H. Dudenhöffer, B. M. Freitas, J. Ghazoul, S. Greenleaf, J. Hipólito, A. Holzschuh, B. Howlett, R. Isaacs, S. K. Javorek, C. M. Kennedy, K. M. Krewenka, S. Krishnan, Y. Mandelik, M. M. Mayfield, I. Motzke, T. Munyuli, B. A. Nault, M. Otieno, J. Petersen, G. Pisanty, S. G. Potts, R. Rader, T. H. Ricketts, M. Rundlöf, C. L. Seymour, C. Schüepp, H. Szentgyörgyi, H. Taki, T. Tscharntke, C. H. Vergara, B. F. Viana, T. C. Wanger, C. Westphal, N. Williams, and A. M. Klein, "Wild pollinators enhance fruit set of crops regardless of honey bee abundance," *Science*, vol. 339, no. 6127, pp. 1608–1611, 2013. [Online]. Available: https://science.sciencemag.org/content/339/6127/1608

[9] B. Walsh, "A World without Bees," *Time*, vol. August 19, pp. 26–31, 2013.

[10] W. G. Meikle and N. Holst, "Application of continuous monitoring of honeybee colonies," *Apidologie*, vol. 46, no. 1, pp. 10–22, 2015.

[11] M. E. A. McNeil, "Electronic Bee Monitoring," *American Bee Journal*, vol. August, pp. 875–879, 2015.

[12] A. Zacepins, V. Brusbardis, J. Meitalovs, and E. Stalidzans, "Challenges in the development of precision beekeeping," *Biosystems Engineering*, vol. 130, pp. 60–71, 02 2015.

[13] M. T. Sanford, "Second International Workshop on Hive and Bee Monitoring," *American Bee Journal*, vol. December, pp. 1351–1353, 2014.

[14] "Langstroth beehive," available online: https://en.wikipedia.org/wiki/Langstroth_hive (accessed on 15 Oct 2020).

[15] "Dadant beehive," available online: https://en.wikipedia.org/wiki/Charles_Dadant (accessed on 15 Oct 2020).

[16] V. Kulyukin, M. Putnam, and S. Reka, "Digitizing buzzing signals into a440 piano note sequences and estimating forager traffic levels from images in solar-powered, electronic beehive monitoring," in *Lecture Notes in Engineering and Computer Science: Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 1, 03 2016, pp. 82–87.

[17] V. Kulyukin and S. Reka, "Toward sustainable electronic beehive monitoring: algorithms for omnidirectional bee counting from images and harmonic analysis of buzzing signals," *Engineering Letters*, vol. 24, no. 3, pp. 317–327, 2016.

[18] V. Kulyukin and S. Mukherjee, "Computer vision in electronic beehive monitoring: in situ vision-based bee counting on langstroth hive landing pads," *Graphics, Vision and Image Processing GVIP*, vol. 17, no. 1, pp. 25–37, 5 2017.

[19] V. Kulyukin, "Beepi: a multisensor electronic beehive monitor," https://www.kickstarter.com/projects/970162847/beepi-a-multisensor-electronic-beehive-monitor (accessed Jan. 2, 2020).

[20] V. Kulyukin and S. Mukherjee, "Beepi: Honeybees meet ai: stage 2," https://www.kickstarter.com/projects/beepihoneybeesmeetai/beepi-honeybees-meet-ai-stage-2 (accessed Oct 20, 2020).

[21] V. Kulyukin, S. Mukherjee, and P. Amlathe, "Toward audio beehive monitoring: Deep learning vs. standard machine learning in classifying beehive audio samples," *Applied Sciences*, vol. 8, p. 1573, Sep 2018. [Online]. Available: https://doi.org/10.3390/app8091573

[22] B. N. Metz, J. Wu-Smart, and M. Simone-Finstrom, "Proceedings of the 2020 american bee research conference," *Insects*, vol. 11, no. 6, p. 362, Jun 2020. [Online]. Available: http://dx.doi.org/10.3390/insects11060362

[23] S. Mukherjee and P. Amlathe, "Buzz1: A dataset of bee buzzing, cricket chirping, and ambient noise audio samples," available online: https://usu.app.box.com/v/BeePiAudioData (accessed on 15 Oct 2020).

[24] A. Tiwari and V. Kulyukin, "Bee1: A dataset of 54,383 labeled images of honeybees obtained from beepi, a multi-sensor electronic beehive monitor." available online: https://usu.box.com/s/p7y8v95ot9u9jvjbbayci61no3lzbgx3 (accessed on 15 Oct 2020).

[25] P. Vats and V. Kulyukin, "Bee2_1s: A dataset of 54,201 labeled images of honeybees obtained from beepi, a multi-sensor electronic beehive monitor, mounted on top of the first super of a langstroth beehive." available online: https://usu.box.com/s/p7y8v95ot9u9jvjbbayci61no3lzbgx3 (accessed on 15 Oct 2020).

[26] P. Vats, V. Kulyukin, and S. Mukherjee, "Bee2_2s: A dataset of 54,678 labeled images of honeybees obtained from beepi, a multi-sensor electronic beehive monitor, mounted on top of the second super of a langstroth beehive." available online: https://usu.box.com/s/3ccizd5b1qzcqcs4t0ivawmrxbgva7ym (accessed on 15 Oct 2020).

[27] R. J. Adrian, "Particle-imaging techniques for experimental fluid mechanics," *Annual Review of Fluid Mechanics*, vol. 23, no. 1, pp. 261–304, 1991.

[28] R. Adrian, "Twenty years of particle image velocimetry," *Experiments in Fluids*, no. 39, pp. 159–169, 2005.

[29] J. Nieh, "Bumblebees: Behaviour, ecology, and conservation. second edition." *The Quarterly Review of Biology*, vol. 85, no. 3, pp. 373–373, 2010. [Online]. Available: https://doi.org/10.1086/655083

[30] T. D. Seeley, *Honeybee Democracy*. Princeton University Press., 2010.

[31] B. Gates, "The temperature of the bee colony," *Bulletin. U.S.D.A.*, vol. 96, pp. 1–29, 01 1914.

[32] W. E. Dunham, "Hive temperatures for each hour of a day," 1931.

[33] E. Woods, "Means for detecting and indicating the activities of bees and conditions in beehives." u.S. Patent 2,806,082, 10 September 1957. [Online]. Available: https://patents.google.com/patent/US2806082A/en

[34] M. Bencsik, J. Bencsik, M. Baxter, A. Lucian, J. Romieu, and M. Millet, "Identification of the Honey Bee Swarming Process by Analyzing the Time Course of Hive Vibrations," *Computers and Electronics in Agriculture*, vol. 76, pp. 44–50, 2011.

[35] W. G. Meikle, N. Holst, G. Mercadier, F. Derouan, and R. R. James, "Using balances linked to dataloggers to monitor honey bee colonies," *Journal of Apicultural Research*, vol. 45, no. 1, pp. 39–41, 2006. [Online]. Available: https://doi.org/10.1080/00218839.2006.11101311

[36] J.-F. Odoux, P. Aupinel, S. Gateff, F. Requier, M. Henry, and V. Bretagnolle, "Ecobee: a tool for long-term honey bee colony monitoring at the landscape scale in west european intensive agroecosystems," *Journal of Apicultural Research*, vol. 53, no. 1, pp. 57–66, 2014. [Online]. Available: https://doi.org/10.3896/IBRA.1.53.1.05

[37] A. Kviesis, A. Zacepins, and G. Riders, "Honey bee colony monitoring with implemented decision support system," vol. 14, 05 2015.

[38] S. E. Barlow and M. A. ONeill, "Technological advances in field studies of pollinator ecology and the future of e-ecology," *Current Opinion in Insect Science*, vol. 38, pp. 15 – 25, 2020, ecology Parasites/Parasitoids/Biological control. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2214574520300225

[39] V. Snchez, S. Gil, J. M. Flores, F. J. Quiles, M. A. Ortiz, and J. J. Luna, "Implementation of an electronic system to monitor the thermoregulatory capacity of honeybee colonies in hives with open-screened bottom boards," *Computers and Electronics in Agriculture*, vol. 119, pp. 209 – 216, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169915003312

[40] W. Blomstedt, "Technology V: Understanding the Buzz with Arnia," *American Bee Journal*, vol. October, pp. 1101–1104, 2014.

[41] E. Woods, "Hivemind, remote beehive monitoring," online, Accessed 20 October 2020. [Online]. Available: https://hivemind.nz/

[42] A. Rafael Braga, D. G. Gomes, R. Rogers, E. E. Hassler, B. M. Freitas, and J. A. Cazier, "A method for mining combined data from in-hive sensors, weather and apiary inspections to forecast the health status of honey bee colonies," *Computers and Electronics in Agriculture*, vol. 169, p. 105161, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169919317661

[43] R. B. Wright, "Lundie's flight activities of the honey bee.," *Bee World*, vol. 9, no. 2, pp. 21–23, 1928. [Online]. Available: https://doi.org/10.1080/0005772X.1928.11096153

[44] A. C. Fabergé, "Apparatus for recording the number of bees leaving and entering a hive," *Journal of Scientific Instruments*, vol. 20, no. 2, pp. 28–31, Feb 1943. [Online]. Available: https://doi.org/10.1088%2F0950-7671%2F20%2F2%2F302

[45] E. H. Erickson, H. H. Miller, and D. J. Sikkema, "A method of separating and monitoring honeybee flight activity at the hive entrance," *Journal of Apicultural Research*, vol. 14, no. 3-4, pp. 119–125, 1975. [Online]. Available: https://doi.org/10.1080/00218839.1975.11099814

[46] C. Liu, J. J. Leonard, and J. J. Feddes, "Automated monitoring of flight activity at a beehive entrance using infrared light sensors," *Journal of Apicultural Research*, vol. 29, no. 1, pp. 20–27, 1990. [Online]. Available: https://doi.org/10.1080/00218839.1990.11101193

[47] R. G. Danka and N. E. Gary, "Estimating Foraging Populations of Honey Bees (Hymenoptera: Apidae) from Individual Colonies," *Journal of Economic*

Entomology, vol. 80, no. 2, pp. 544–547, 04 1987. [Online]. Available: https://doi.org/10.1093/jee/80.2.544

[48] Struye, M. H., Mortier, H. J., Arnold, G., Miniggio, C., and Borneck, R., "Microprocessor-controlled monitoring of honeybee flight activity at the hive entrance," *Apidologie*, vol. 25, no. 4, pp. 384–395, 1994. [Online]. Available: https://doi.org/10.1051/apido:19940405

[49] J. Bromenshenk, C. Henderson, R. Seccomb, P. Welch, S. Debnam, and D. Firth, "Bees as biosensors: Chemosensory ability, honey bee monitoring systems, and emergent sensor technologies derived from the pollinator syndrome," *Biosensors*, vol. 5, pp. 678–711, 10 2015.

[50] C. Chen, E.-C. Yang, J.-A. Jiang, and T.-T. Lin, "An imaging system for monitoring the in-and-out activity of honey bees," *Computers and Electronics in Agriculture*, vol. 89, p. 100109, 11 2012.

[51] C. Schneider, J. Tautz, B. Grnewald, and S. Fuchs, "Rfid tracking of sublethal effects of two neonicotinoid insecticides on the foraging behavior of apis mellifera," *PloS one*, vol. 7, p. e30023, 01 2012.

[52] I. Heidinger, M. Meixner, S. Berg, and B. Ralph, "Observation of the mating behavior of honey bee (apis mellifera l.) queens using radio-frequency identification (rfid): Factors influencing the duration and frequency of nuptial flights," *Insects*, vol. 5, pp. 513–527, 07 2014.

[53] Cheng Yang and J. Collins, "A model for honey bee tracking on 2d video," in *2015 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, 2015, pp. 1–6.

[54] C. Yang and J. Collins, "Improvement of honey bee tracking on 2d video with hough transform and kalman filter," *Journal of Signal Processing Systems*, 11 2017.

[55] C. Yang, J. Collins, and M. Beckerleg, "A model for pollen measurement using video monitoring of honey bees," *Sensing and Imaging*, vol. 19, 12 2018.

[56] T. Kimura, M. Ohashi, K. Crailsheim, T. Schmickl, R. Odaka, and H. Ikeno, "Tracking of multiple honey bees on a flat surface," 11 2012, pp. 36–39.

[57] T. N. Ngo, K.-C. Wu, E.-C. Yang, and T.-T. Lin, "A real-time imaging system for multiple honey bee tracking and activity monitoring," *Computers and Electronics in Agriculture*, vol. 163, p. 104841, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169919301498

[58] G. R. Spedding, A. Hedenström, and M. Rosén, "Quantitative studies of the wakes of freely flying birds in a low-turbulence wind tunnel," *Experiments in Fluids*, vol. 34, no. 2, p. 291, Feb 2003. [Online]. Available: https://doi.org/10.1007/s00348-002-0559-8

[59] C. E. Willert and M. Gharib, "Digital particle image velocimetry," *Experiments in Fluids*, vol. 10, no. 4, pp. 181–193, Jan 1991. [Online]. Available: https://doi.org/10.1007/BF00190388

[60] P. Henningsson, G. R. Spedding, and A. Hedenström, "Vortex wake and flight kinematics of a swift in cruising flight in a wind tunnel," *Journal of Experimental Biology*, vol. 211, no. 5, pp. 717–730, 2008.

[61] A. Hedenström, L. C. Johansson, M. Wolf, R. von Busse, Y. Winter, and G. R. Spedding, "Bat flight generates complex aerodynamic tracks," *Science*, vol. 316, no. 5826, pp. 894–897, 2007.

[62] G. R. Spedding, M. Rosén, and A. Hedenström, "A family of vortex wakes generated by a thrush nightingale in free flight in a wind tunnel over its entire natural range of flight speeds," *Journal of Experimental Biology*, vol. 206, no. 14, pp. 2313–2344, 2003. [Online]. Available: https://jeb.biologists.org/content/206/14/2313

[63] F. T. Muijres, L. C. Johansson, R. Barfield, M. Wolf, G. R. Spedding, and A. Hedenström, "Leading-edge vortex improves lift in slow-flying bats," *Science*, vol. 319, pp. 1250 – 1253, Feb 2008.

[64] T. Y. Hubel, D. K. Riskin, S. M. Swartz, and K. S. Breuer, "Wake structure and wing kinematics: the flight of the lesser dog-faced fruit bat, cynopterus brachyotis," *Journal of Experimental Biology*, vol. 213, no. 20, pp. 3427–3440, 2010. [Online]. Available: https://jeb.biologists.org/content/213/20/3427

[65] M. Dickinson, F. Lehmann, and S. Sane, "Wing rotation and the aerodynamic basis of insect flight," *Science*, vol. 284, pp. 1954 – 1960, 1999.

[66] R. J. Bomphrey, N. J. Lawson, G. K. Taylor, and A. L. R. Thomas, "Application of digital particle image velocimetry to insect aerodynamics: measurement of the leading-edge vortex and near wake of a hawkmoth," *Experiments in Fluids*, vol. 40, pp. 546 – 554, Jan 2006.

[67] A. Michelsen, *How do honey bees obtain information about direction by following dances?* Springer, 2012, pp. 65–76.

[68] I. F. Rodriguez, R. Megret, R. Egnor, K. Branson, J. L. Agosto, T. Giray, and E. Acuna, "Multiple insect and animal tracking in video using part affinity fields," *Workshop Visual observation and analysis of Vertebrate And Insect Behavior (VAIB) at International Conference on Pattern Recognition (ICPR)*, Aug 2018. [Online]. Available: http://par.nsf.gov/biblio/10095768

[69] Z. Babic, R. Pilipovic, V. Risojevic, and G. Mirjanic, "Pollen bearing honey bee detection in hive entrance video recorded by remote embedded system for pollination monitoring," in *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.*, 2016, pp. 51–57. [Online]. Available: https://doi.org/10.5194/isprs-annals-III-7-51-2016

[70] D. Reynolds and J. Riley, "Remote-sensing, telemetric and computer-based technologies for investigating insect movement: a survey of existing and potential

techniques," *Computers and Electronics in Agriculture*, vol. 35, no. 2, pp. 271 – 307, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169902000236

[71] J. Devillers, J. Dor, M. Tisseur, S. Cluzeau, and G. Maurin, "Modelling the flight activity of apis mellifera at the hive entrance," *Computers and Electronics in Agriculture*, vol. 42, no. 2, pp. 87 – 109, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169903001029

[72] D. Clarke and D. Robert, "Predictive modelling of honey bee foraging activity using local weather conditions," *Apidologie*, vol. 49, 02 2018.

[73] D. A. Mezquida and J. L. Martnez, "Platform for bee-hives monitoring based on sound analysis: a perpetual warehouse for swarm's daily activity," *Spanish Journal of Agricultural Research*, vol. 7, no. 4, pp. 824–828, 2009.

[74] J. Rangel and T. Seeley, "The signals initiating the mass exodus of a honeybee swarm from its nest," *Animal Behaviour - ANIM BEHAV*, vol. 76, pp. 1943–1952, 12 2008.

[75] S. Ferrari, M. Silva, M. Guarino, and D. Berckmans, "Monitoring of swarming sounds in bee hives for early detection of the swarming period," *Computers and Electronics in Agriculture*, vol. 64, no. 1, pp. 72–77, nov 2008.

[76] M. Ramsey, M. Bencsik, and M. I. Newton, "Long-term trends in the honeybee whooping signal revealed by automated detection," *PLOS ONE*, vol. 12, no. 2, pp. 1–22, Feb 2017.

[77] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, aug 2013.

[78] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12. Curran Associates Inc., 2012, pp. 1097–1105.

[79] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1915–1929, aug 2013.

[80] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[81] G. Hinton, L. Deng, D. Yu, G. Dahl, A. rahman Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *Signal Processing Magazine*, pp. 82–97, Nov 2012.

[82] T. N. Sainath, A. Rahman Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr." in *ICASSP*. IEEE, 2013, pp. 8614–8618.

[83] T. N. Sainath, R. J. Weiss, A. W. Senior, K. W. Wilson, and O. Vinyals, "Learning the speech front-end with raw waveform CLDNNs." in *INTERSPEECH*. ISCA, 2015, pp. 1–5. [Online]. Available: http://dblp.uni-trier.de/db/conf/interspeech/interspeech2015.html#SainathWSWV15

[84] A. Roberts, C. Resnick, D. Ardila, and D. Eck, "Audio deepdream: optimizing raw audio with convolutional networks," 2016.

[85] A. v. d. Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'13. USA: Curran Associates Inc., 2013, pp. 2643–2651.

[86] K. Piczak, "Environmental sound classification with convolutional neural networks," in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, Sept 2015, pp. 1–6.

[87] M. K. K. Leung, H. Y. Xiong, L. J. Lee, and B. J. Frey, "Deep learning of the tissue-regulated splicing code," *Bioinformatics*, vol. 30, no. 12, pp. i121–i129, Jun 2014.

[88] H. Y. Xiong, B. Alipanahi, L. J. Lee, H. Bretschneider, D. Merico, R. K. C. Yuen, Y. Hua, S. Gueroussov, H. S. Najafabadi, T. R. Hughes, Q. Morris, Y. Barash, A. R. Krainer, N. Jojic, S. W. Scherer, B. J. Blencowe, and B. J. Frey, "The human splicing code reveals new insights into the genetic determinants of disease," *Science*, vol. 347, no. 6218, 2015.

[89] Y. Aytar, C. Vondrick, and A. Torralba, "Soundnet: learning sound representations from unlabeled video," in *Advances in Neural Information Processing Systems 29*, 2016, pp. 892–900.

[90] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *Arxiv*, 2016. [Online]. Available: https://arxiv.org/abs/1609.03499

[91] E. Humphrey and J. Bello, "Rethinking automatic chord recognition with convolutional neural networks," in *11th International Conference on Machine Learning and Applications (ICMLA)*, 2012.

[92] X.-J. He, L.-Q. Tian, X.-B. Wu, and Z.-J. Zeng, "Rfid monitoring indicates honeybees work harder before a rainy day," *Insect Science*, vol. 23, no. 1, pp. 157–159, 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/1744-7917.12298

[93] BURRILL, Robert M. and DIETZ, Alfred, "The response of honey bees to variations in solar radiation and temperature," *Apidologie*, vol. 12, no. 4, pp. 319–328, 1981. [Online]. Available: https://doi.org/10.1051/apido:19810402

[94] N. Vicens and J. Bosch, "Weather-Dependent Pollinator Activity in an Apple Orchard, with Special Reference to Osmia cornuta and Apis mellifera (Hymenoptera: Megachilidae and Apidae)," *Environmental Entomology*, vol. 29, no. 3, pp. 413–420, 06 2000. [Online]. Available: https://doi.org/10.1603/0046-225X-29.3.413

[95] Z. Pukadija, E. Stefanic, A. Miji, Z. Zduni, N. Paraikovi, F. Tihomir, and A. Opaak, "Influence of weather conditions on honey bee visits (apis mellifera carnica) during sunflower (helianthus annuus l.) blooming period," *AGRICULTURE: Scientific and Professional Review (ured@pfos.hr); Vol.13 No.1*, vol. 13, 06 2007.

[96] S. Mukherjee and V. Kulyukin, "Human curated dataset of frame by frame count of bee motions," available online: https://usu.box.com/s/azl62oog4quds2jhegz5hw1f7j8v5441 (accessed on 23 Oct 2020).

[97] Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognit. Lett.*, vol. 27, pp. 773–780, 2006.

[98] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, ser. AAAIWS'94. AAAI Press, 1994, pp. 359–370. [Online]. Available: http://dl.acm.org/citation.cfm?id=3000850.3000887

[99] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intell. Data Anal.*, vol. 11, no. 5, pp. 561–580, oct 2007.

[100] S. Farina A, Gage, *Ecoacoustics: the ecological role of sounds*. Wiley, 2017.

[101] T. Mitchell, *Machine Learning*. McGraw-Hill, 1997.

[102] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, 1998, pp. 2278–2324.

[103] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10. Omnipress, 2010, pp. 807–814.

[104] M. D. Zeiler, M. Ranzato, R. Monga, M. Z. Mao, K. Yang, Q. V. Le, P. Nguyen, A. W. Senior, V. Vanhoucke, J. Dean, and G. E. Hinton, "On rectified linear units for speech processing." in *ICASSP*. IEEE, 2013, pp. 3517–3521.

[105] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[106] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of machine learning research*, vol. 15, no. 1, p. 19291958, 2014. [Online]. Available: http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf

[107] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks." in *AISTATS*, ser. JMLR Proceedings, vol. 9. JMLR.org, 2010, pp. 249–256.

[108] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *ICML*, ser. JMLR Workshop and Conference Proceedings, vol. 37. JMLR.org, 2015, pp. 448–456.

[109] "Tflearn," available online: https://github.com/tflearn/tflearn (accessed on 15 Oct 2020).

[110] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[111] "Tensorboard," available online: https://www.tensorflow.org/programmers_guide/summaries_and_tensorboard (accessed on 15 Oct 2020).

[112] D. Freedman, *Statistical models : theory and practice*. Cambridge University Press, August 2005.

[113] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[114] J.-H. Hong and S.-B. Cho, "A probabilistic multi-class strategy of one-vs.-rest support vector machines for cancer classification," *Neurocomputing*, vol. 71, no. 16–18, pp. 3275–3281, Oct 2008.

[115] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct 2001.

[116] Kirchner, W. H., "Acoustical communication in honeybees," *Apidologie*, vol. 24, no. 3, pp. 297–307, 1993. [Online]. Available: https://doi.org/10.1051/apido:19930309

[117] S. Pratt, S. Khnholz, T. Seeley, and A. Weidenmller, "Worker piping associated with foraging in undisturbed queenright colonies of honey bees," *Apidologie*, vol. 27, pp. 13–20, 01 1996. [Online]. Available: http://doi.org/10.1051/apido:19960102

[118] A. Michelsen, W. Kirchner, B. B. Andersen, and M. Lindauer, "The tooting and quacking vibration signals of honeybee queens: a quantitative analysis," *Journal of Comparative Physiology A*, vol. 158, no. 5, pp. 605–611, 2004.

[119] C. Thom, D. Gilley, and J. Tautz, "Worker piping in honey bees (apis mellifera): The behavior of piping nectar foragers," *Behavioral Ecology and Sociobiology*, vol. 53, pp. 199–205, 01 2003.

[120] T. Seeley and J. Tautz, "Worker piping in honey bee swarms and its role in preparing for liftoff," *Journal of Comparative Physiology*, vol. 187, pp. 667–676, 10 2001.

[121] M. Sarma, S. Fuchs, C. Werber, and J. Tautz, "Worker piping triggers hissing for coordinated colony defence in the dwarf honeybee apis florea," *Zoology (Jena, Germany)*, vol. 105, pp. 215–23, 02 2002.

[122] "Audioop–manipulate raw audio data," last Accessed: 09-07-2020. [Online]. Available: https://docs.python.org/3/library/audioop.html

[123] "Utah climate center," last Accessed: 09-11-2020. [Online]. Available: https://climate.usu.edu/

[124] "Usu weather station," last Accessed: 09-11-2020. [Online]. Available: https://climate.usu.edu/mchd/dashboard/overview/USUwx.php

[125] J. Seemann, *Net Radiation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1979, pp. 22–25. [Online]. Available: https://doi.org/10.1007/978-3-642-67288-0_4

[126] "Net radiation," last Accessed: 09-11-2020. [Online]. Available: https://earthobservatory.nasa.gov/global-maps/CERES_NETFLUX_M

[127] "Evapotranspiration," last Accessed: 09-11-2020. [Online]. Available: http://www.fao.org/3/x0490e/x0490e04.htm#evapotranspiration%20(et)

[128] R. Johnston, K. Jones, and D. Manley, "Confounding and collinearity in regression analysis: a cautionary tale and an alternative procedure, illustrated by studies of British voting behaviour," *Quality & Quantity: International Journal of Methodology*, vol. 52, no. 4, pp. 1957–1976, July 2018. [Online]. Available: https://ideas.repec.org/a/spr/qualqt/v52y2018i4d10.1007_s11135-017-0584-6.html

[129] H. Akaike, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 716–723, 1974. [Online]. Available: https://doi.org/10.1109/TAC.1974.1100705

[130] N. Committee, "Report on the national stakeholders conference on honey bee health," pp. 9–68, 01 2013. [Online]. Available: https://www.usda.gov/sites/default/files/documents/ReportHoneyBeeHealth.pdf7

[131] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, vol. 2, 2004, pp. 28–31 Vol.2.

[132] P. Kaewtrakulpong and R. Bowden, "An improved adaptive background mixture model for realtime tracking with shadow detection," *Proceedings of 2nd European Workshop on Advanced Video-Based Surveillance Systems; September 4, 2001; London, U.K*, 05 2002.

[133] Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognition Letters*, vol. 27, no. 7, pp. 773 – 780, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865505003521

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

[135] J. Bromenschenk, C. Henderson, R. Seccomb, S. Rice, and R. Etter, "Honey bee acoustic recording and analysis system for monitoring hive health," Patent US7549907B2, Sep 2007.

APPENDICES

APPENDIX A

Interrogation Window and Overlap

We chose 32 videos across different times during the bee keeping season of 2018 and 2019 [96] such that they had varying levels of bee traffic across different backgrounds. We first found in Table A.1, for each video which values of interrogation window and overlap worked the best, i.e. had the least error when compared to the ground truth data.

Video	**Window**	**Overlap**	**Dpiv**	**Human Count**	**Difference**
Vid1	14	6	4893	5693	800
Vid2	17	2	189	343	154
Vid3	17	2	2656	2887	231
Vid4	17	2	59	73	14
Vid5	17	2	75	75	0
Vid6	17	4	231	239	8
Vid7	12	3	71	74	3
Vid8	10	4	360	361	1
Vid9	17	2	478	478	0
Vid10	12	3	368	357	11
Vid11	12	1	374	373	1
Vid12	17	2	339	348	9
Vid13	8	2	174	176	2
Vid14	12	3	208	208	0
Vid15	10	4	456	456	0
Vid16	10	4	263	270	7
Vid17	12	2	157	154	3
Vid18	12	1	294	294	0
Vid19	17	4	14	17	3
Vid20	12	1	65	60	5
Vid21	8	2	419	432	13
Vid22	10	2	101	101	0
Vid23	12	3	249	247	2
Vid24	12	3	168	168	0
Vid25	17	2	90	90	0
Vid26	16	4	74	74	0
Vid27	16	4	1901	1943	42
Vid28	12	3	6582	6580	2
Vid29	17	4	186	186	0
Vid30	17	4	279	289	10
Vid31	10	2	652	643	9
Vid32	14	3	1023	1401	378

Table A.1

A.1 A Simple Strategy To Select Interrogation Window And Overlap

We can design a strategy where in case of low traffic we could use 10% overlap, in case of medium traffic we could use 20% overlap and in case of high traffic we could use 30% overlap. Next we can select the size of the interrogation window based upon the average size of an individual bee on an image frame. Since, the average size of an individual bee is between 8-10 pixels, we can choose the size of the interrogation window to be approximately twice the size of an individual bee (17 pixels).

Normalized Threshold ($thresholdNorm$)	Interrogation Window	Overlap
$9 \leq thresholdNorm < 18$	17	2
$18 \leq thresholdNorm < 36$	17	3
$36 \leq thresholdNorm < 65$	17	5

Table A.2: Values for interrogation window and overlap used by the bee motion counting algorithm **DPIV_A**.

Table A.2 shows the choice of interrogation window and overlap based on $thresholdNorm$, which is another measure of the level of bee traffic (Chapter 4). We tested our strategy on 32 evaluation videos and the results are given in Figure A.1. The strategy described in Table A.2 was used to generate the counts for the **DPIV_A** bee counting algorithm. Although we can see that for some videos the error in the bee count was low for **DPIV_A** but there were also certain cases where absolute error was higher. Hence, we can see that Table A.2 was not the optimal choice as we found out through our experiments with 32 evaluation videos. One can design other strategies too to determine the interrogation window and overlap but since the videos were chosen in a way such that they had varying levels of bee traffic across different backgrounds, we believe that fixing the interrogation window size for the entirety was not an ideal choice.

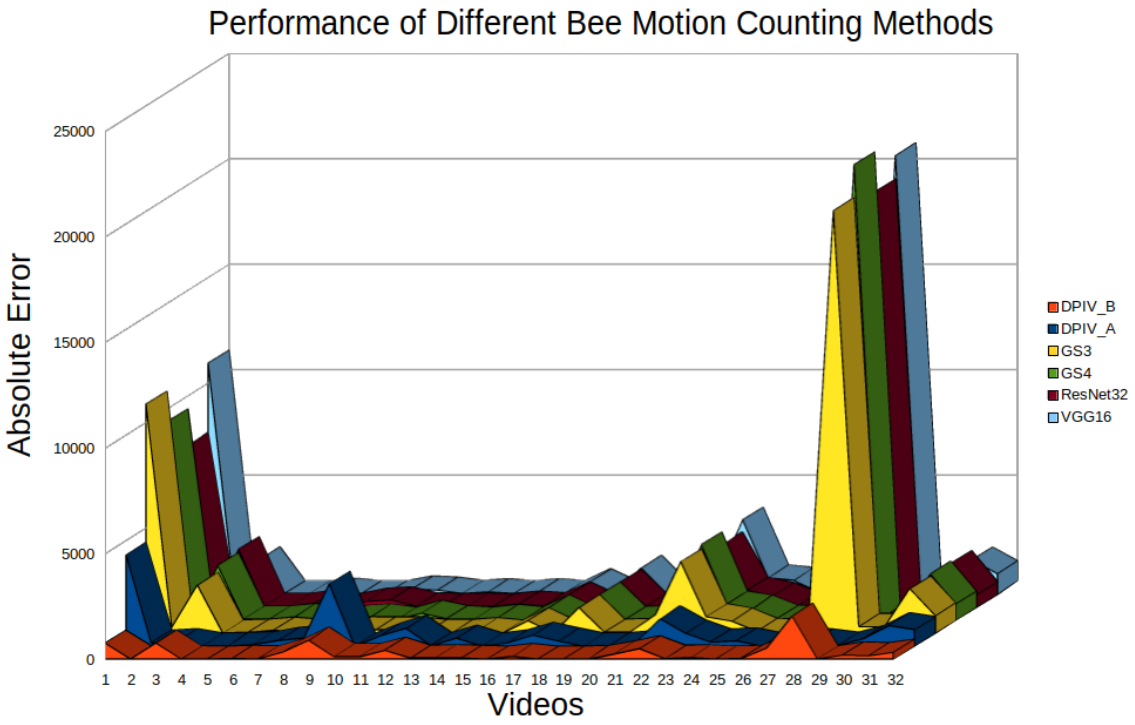


Figure A.1: Absolute difference in bee motion counts between the results of the five bee counting algorithm and the ground truth data for 32 evaluation videos.

CURRICULUM VITAE

Sarbajit Mukherjee

Education

- Ph.D., Computer Science, Utah State University, Logan, Utah, US, Adviser: Dr. Vladimir Kulyukin, November 2020.
- M.E., Computer Science And Engineering, Indian Institute of Engineering Science and Technology, Shibpur, West Bengal, India, 2014.
- B.Tech., Computer Science And Engineering, West Bengal University of Technology, Kolkata, India, 2012.

Research Interests

- Machine Learning, Deep Learning, Data Science, Sensor Fusion, Computer Vision, Statistical learning.

Published Journal Articles

- S. Mukherjee and V. Kulyukin, "Application of digital particle image velocimetry to insect motion: Measurement of incoming, outgoing, and lateral honeybee traffic", *Applied Sciences*, vol. 10, no. 6, p. 2042, March 2020.
- A. Minichiello, D. Armijo, S. Mukherjee, et al. "Developing a mobile application based particle image velocimetry tool for enhanced teaching and learning in fluid mechanics: A design based research approach", *Comput Appl Eng Educ.*, p. 1–21, June 2020;

- V. Kulyukin and S. Mukherjee, "On video analysis of omnidirectional bee traffic: counting bee motions with motion detection and image classification", *Applied Sciences*, vol. 9, no.18, p. 3743, Sep 2019.

- V. Kulyukin, S. Mukherjee, and P. Amlathe, "Toward audio beehive monitoring: Deep learning vs. standard machine learning in classifying beehive audio samples", *Applied Sciences*, vol. 8, p. 1573, Sep 2018.

- V. Kulyukin and S. Mukherjee, "Computer vision in electronic beehive monitoring: in situ vision-based bee counting on langstroth hive landing pads", *Graphics, Vision and Image Processing GVIP*, vol. 17, no. 1, pp. 25–37, May 2017.

Published Conference Papers

- V. Kulyukin and S. Mukherjee, "A Two-Tier Method for Counting Omnidirectional Bee Motions in Videos", *2020 American Bee Research Conference (ABRC)*, Schaumburg, Illinois, US, January 2020,

- S. Mukherjee and V. Kulyukin, "Work in Progress: Using Particle Image Velocimetry (PIV) to Estimate Incoming, Outgoing, and Lateral Bee Traffic Flows in Videos", *2020 American Bee Research Conference (ABRC)*, Schaumburg, Illinois, US, January 2020.

- Lori. M. Caldwell, D. Armijo, S. Mukherjee, et al., "Work in Progress: Mobile Instructional Particle Image Velocimetry for STEM Outreach and Undergraduate Fluids Mechanics Education". *2019 ASEE Annual Conference & Exposition*, Tampa, Florida, US, June 2019.

- A. Echols, S. Mukherjee, M. Mickelsen and Z. Pantic, "Communication Infrastructure for Dynamic Wireless Charging of Electric Vehicles", *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, San Francisco, CA, US, March 2017.

Classes Taught

- *CS 3200: Mobile Application Development for Android*, Fall 2017.
- *CS 3200: Mobile Application Development for Android*, Fall 2016.