IMPLEMENTATION OF AN AUTOMATED IMAGE PROCESSING SYSTEM FOR OBSERVING THE ACTIVITIES OF HONEY BEES

A Thesis
by
AHMAD GHADIRI

Submitted to the Graduate School
at Appalachian State University
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

August 2013
Department of Computer Science

IMPLEMENTATION OF AN AUTOMATED IMAGE PROCESSING SYSTEM FOR OBSERVING THE ACTIVITIES OF HONEY BEES

A Thesis
by
AHMAD GHADIRI
August 2013

APPROVED BY:

Abstract

IMPLEMENTATION OF AN AUTOMATED IMAGE PROCESSING SYSTEM FOR OBSERVING THE ACTIVITIES OF HONEY BEES

Ahmad Ghadiri
B.Sc., Isfahan University of Technology
M.S., Appalachian State University

Chairperson: Rahman Tashakkori

Honey bees are responsible for more than half of the pollination across the world, hence they have a very important role in agriculture. In recent years, the number of honey bees have declined due to the Colony Collapse Disorder (CCD), which may be due to possible causes that include climate change and pollution. A large number of bee researchers believe that climate change can have a significant effect on honey bees [BeesFree Inc. 2012].

Honey bees fly around to pollinate the flowers, fruits, and plants and to collect pollen and nectar for their own use. For a pound of honey, honey bees fly over 55000 miles [York County Beekeepers' Association 2011]. Since honey bees start from their hives to go for pollination, their activities around the hive can be used as a good indicator of their health. The beehive's health status and the activities of honey bees can be related to the environment in which they live. As such, local weather data need to be gathered and correlated to the level of bee activities in front of their hives. In this thesis, an automated system is designed and implemented using computer science techniques for data acquisition. The system utilizes image processing techniques to extract data from the videos taken in front or at the top of the hive’s entrance. Several web-based applications are used to obtain temperature and humidity data from the National Weather Service to supplement the data that are collected at the hive locally. All the weather data and those extracted from the images are stored in a MySQL database for analysis and accessed by an iPhone App that is designed as part of this research.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my adviser, Dr. Rahman Tashakkori, for his continuous support of my M.S. study and research. I am thankful for his patience, motivation, enthusiasm, and immense knowledge. His guidance has been very valuable to me throughout my research and writing of this thesis. I cannot imagine having a better advisor and mentor for my Master's study.

I would also like to thank the rest of my thesis committee: Dr. James Wilkes and Professor Frank Barry, for their encouragement, insightful comments, and valuable advice.

I would also like to thank all the faculty and staff of the department of computer science, who throughout my Master's degree have supported and encouraged me to believe in my abilities. They have directed me through numerous challenges allowing me to accomplish my goals.

I would like to thank my family for never ceasing to believe in me. Their continued encouragement has helped me to pursue my dreams and goals. Without my family's love and presence success would have been meaningless.

Finally, I would like to thank my peers and friends for helping me through my Master's study and for allowing me to have fun with them in the last two years.

Table of Contents

CHAPTER 1 INTRODUCTION

1.1 Background

Studying the behavior of honey bees is not new [Von Frisch 1993], and several studies have been conducted using computer science techniques [Campbell et al. 2005; Apiservices 2007; Lundie 1925; Pham-Delègue et al. 2002]. Most of the recent honey bee research studies have been conducted to determine the cause of the decline in the number of honey bees around the world [Ellis et al. 2010]. Several research projects attempted to automate the process of studying the honey bees [Campbell et al. 2008; Campbell et al. 2005; Estivill-Castro et al. 2003]. Although some of these studies have made some progress in automating the process, the research is still ongoing and requires significant improvement. Some of these research studies will be described in more detail in Chapter 2.

1.2 Image Processing

In computer science, image processing refers to any form of signal processing for which the input is an image.

An image may be defined as a two-dimensional function $f(x, y)$, where x and y are spatial (plane) coordinates, and the amplitude of f at any pair of coordinates (x, y) is the intensity or gray level of the image at that point. When x, y, and the amplitude values of f are all finite and discrete quantities, the image is called a digital image. The field of digital image processing refers to processing digital images using a computer. Image processing usually refers to digital image processing, but optical and analog image processing are also possible [Gonzalez and Woods 2001].

The most important requirement for image processing is that the image be available in a digitized form, i.e., containing a finite number of binary words. For digitization, the given image is sampled on a discrete grid and each sample or pixel is quantized using a finite number of bits. The digitized image then can be processed by a computer. To display a digital image, it is first converted back into an analog signal, and then scanned onto a display.

Image processing is closely related to computer vision. Computer vision is often considered a high-level form of image processing when a computer attempts to decipher the physical contents of an image or a sequence of images (e.g., videos or 3D full-body magnetic resonance scans).

Digital processing techniques help in the manipulation of the digital images using computers. The raw data from imaging sensors may contain deficiencies, to correct these flaws and obtain accurate data, the image must undergo several phases of processing. The three general phases that all types of data must undergo during digital processing are: pre-processing, enhancement and display, and information extraction.

1.3 Overview of the Research

This thesis provides details on the design and implementation of a fully automated system for observing the activities of honey bees. To monitor and observe the activities of honey bees, a camera is placed in front of the beehive or above it to record their movements during the day. The video is transferred to a server and converted into image frames. The goal is to track the activities of bees in front of a beehive. To achieve this goal, this research finds an estimation of number of bees in front of the beehive on each image frame at any given time.

This thesis uses the illumination-invariant change detection algorithm developed by Durucan and Ebrahimi [2000] to estimate the number of bees in an image. This algorithm decreases the effect of changes in illumination caused by the natural environment. The main source of such changes is the sun and its location in the sky, which produces various changes in illumination of the scene. Also, the light in the scene makes shadows of the honey bees that can be mistaken with the real bees in an image. Other sources of changes in the scene include rain, clouds, and wind.

The automated system developed as part of this thesis also collects data related to the local weather, which can be compared with the movements of bees. The local temperature and humidity data are obtained hourly from the National Weather Service website [NOAA 2013]. An iPhone application is also designed and implemented to make accessing the data from the server more convenient.

1.4 Thesis Organization

This research implements an automated system for observing and monitoring the activities of honey bees. This system extracts different data from images of beehives using image processing techniques and combines the data with the local weather data fetched from the National Weather Service. The system can be used to observe the honey bees' activities for a long period of time. Other parameters can be embedded into the system to provide additional information.

The related work that have been conducted to observe the behavior of honey bees are described briefly in Chapter 2. Chapter 3 explains some basic concepts in image processing as well as different techniques that have been used in this research. Chapters 4, 5, and 6 cover the methodology and implementation of the system designed for this thesis. Chapter 4 explains methods used to extract the time and date information from images. Chapter 5 covers the details of image processing techniques that are used to estimate the number of honey bees in an image. Chapter 6 explains the implementation of various parts of system for automating the process and storing data. Chapter 7 summarizes results of this research. Chapter 8 contains a discussion on the results of the research and the effects of different parameters on the results.

CHAPTER 2 RELATED WORK

Apiculture has been in decline across the world over recent decades, as is shown by the decreasing numbers of managed honey bee colonies [Ellis et al. 2010; Neumann and Carreck 2010; Potts et al. 2010]. The decline is mainly attributed to various factors such as bacteria and viruses, therefore it is crucial to observe, study, and understand honey bees' behavior as a way to determine their health. Several research studies that applied different methods have been conducted to study bees [Campbell et al. 2005; Apiservices 2007; Lundie 1925; Pham-Delègue et al. 2002]. Among these research studies are several that have used computer science techniques to study bees and to automate the analysis process. One of the main computer science techniques used for studying bees is image processing in which videos or images of beehives are used in the analyses. These studies usually use a camera to capture videos of bees inside or outside the beehives. The videos are then processed using image processing techniques to observe the activities of bees. This chapter provides a summary of such work.

To study the effect of honey bees on the pollination process, the research by Estivill-Castro et al. [2003] places a camera in a macadamia orchard to capture the movement of bees around a macadamia tree. Although this method can track the bees, it intensifies the effect of wind and shaking of the scene, which leads to detecting movements of surrounding flowers as bees. To resolve this problem, a simple solution is to place the camera in front of the beehive, which has two advantages over the method used by Estivill-Castro et al. [2003]; the first advantage is that the movement of all the honey bees is captured by the camera, and the second advantage is that the effect of shaking of flowers in the scene due to wind is significantly reduced.

The small size of the bees in images and their quick movements, makes it challenging to use pattern recognition techniques to detect them; hence a change detection technique is used to track the honey bees. Estivill-Castro et al. [2003] use a simplified difference method to produce the changes between two images. As a result of the change detection technique, a mask is created that is used as a base for color projection to find the bees. Although this method works for a clear background, when the background consists of flowers and is not clear, its accuracy drops significantly. Also, this method does not address the changes in illumination.

Unlike Estivill-Castro et al. [2003], Campbell et al. [2008] place the camera at the top of the beehive. This method reduces wind related object movement in the scene as the camera installed above the entrance of the beehive does not shake much. This camera captures the bees while they enter or leave the beehive. Since the camera is placed at a distance from the entrance of the beehive, it does not affect the bees' normal living conditions; however, the distance between the camera and the beehive entrance makes the detection of bees in the image more challenging.

To make the tracking process easier, Campbell et al. [2008] divide the motion of the bees in four different types: *loitering*, *crawling*, *flying out*, and *flying in*. This classification makes the tracking process easier, since the system can predict the next movement of a bee according to the most recent motion type. For example, if a bee is detected as a *flying in* bee, the system can predict that in the next frame this bee must be closer to the beehive and in the same state as before.

To detect bees, the technique also uses a change detection method where an adaptive background subtraction is applied to the images to separate bees from the background. The background is obtained by taking the average of 300 recent video frames. Although this is a good approximation of the background, it makes the process very slow. As a possibility to make the process faster, a smaller number of frames could be used to make the background.

To detect the bees, Campbell et al. [2008] use a change detection method. The change detection method is accurate in detecting the bees; however the color of beehive box changes over time and makes the tracking the bees less accurate. Similar to the method presented by Estivill-Castro et al. [2003], this method does not account for any illumination changes in the environment which reduces its accuracy.

The research presented by Campbell et al. [2008] tracks the bees using maximum weighted bipartite graph matching. Each bee that is detected in a frame is modeled as a node in a graph. The weight is modeled as the likelihood that a bee with a position in an arbitrary frame i goes to frame $i+1$. This method works 79% of the time on the annotated dataset.

The research presented by Knauer et al. [2005] uses image processing techniques to study the honey bees in a different way. Unlike the previous research studies that use a camera outside the beehive, this research uses a camera inside the hive and uses image processing techniques to detect the bees' hygiene based on the number of mites developing on the bees' brood. Since the camera is installed inside the beehive, the changes in illumination and weather have very little effects on the video quality and illumination.

To find the bees' hygiene, first the uncapped cells in the honey comb should be detected. In the next step a backward process is applied to find the bee(s) that started the process of uncapping the cell. The challenge for this method arises from the number of bees that block the cells, making it difficult to see inside the cells. To resolve this issue, a static background model is used. Since the camera is installed inside the beehive where illumination does not change as often, using one or more static backgrounds for different hours of the day produces reasonable results.

The method that is used to detect uncapped cells by Knauer et al. [2005] is based on the gray level of the image, which is used to make a binary image by applying a threshold. Two algorithms are applied to the image: first the Canny Edge Detection [Canny 1986] algorithm is applied to find all the edges in the binary image and then the Teh-Chin Chain Approximation Algorithm [Teh and Chin 1989] is used to find the circular shape contours. The algorithm first finds the center of the contour by finding the average of all the points in the contour and then calculating the probability of the contour being an uncapped cell. This method is efficient and provides reasonable accuracy. The drawback of this method is that the static background model cannot be used in an outdoor environment considering the changes in illumination.

Another research that has been conducted in Japan, Kimura et al. [2006] is similar to that of Knauer et al. [2005] and emphasizes the detection and tracking of honey bees. In this study, a camera is placed in front of a beehive similar to the method used by Knauer et al. [2005] to track the bees. A vector quantization method is used to separate the background and objects in the scene. Following this step, a classification algorithm is applied to the objects to find the objects candidate for bees' bodies. This step is followed by an identification of bees and tracking them in the images.

The method explained above can detect 73% of individual bees and track them in 400 images. Although this is a reasonable result, it cannot be used to track the bees outside the beehive. The reason is that the movement of bees inside and near the hive is slower than outside and the camera can capture the bees in the consecutive frames. Outside the beehive the movement of the bees is faster, thus regular cameras cannot capture the bees in the consecutive frames. This makes it infeasible to track bees in the video frames.

In the final research discussed in this review [Arbuckle et al. 2001], image processing is used to automate the identification process of different species of bees. An automated system was designed to identify the bees' specimen. Since there are more than 40,000 different species of bees, with less than one hundred experts around the world who can correctly identify their species, it seems vital to design such a system. To identify bees, Arbuckle et al. [2001] uses the venation or vein pattern on the wing of bees. This identification system is available via Internet, so one can submit an image from a bee's wing to find the species of the bee.

As explained in this chapter, various research have been conducted on bees for different purposes. However very little, if any, research has been conducted to correlate the activities of honey bees and the local weather parameters such as temperature and humidity. This research aims to design an automated system for observing and monitoring the activities of bees and the effects of local weather data changes on their behavior.

CHAPTER 3 IMAGE PROCESSING

3.1 Introduction

One of the goals of this research is to find the relationship between local weather data and activities of honey bees. For this purpose, the activity of honey bees is monitored continuously over a period of time. To monitor the activities of honey bees, a camera records videos from beehive. The videos received from the camera are broken into the individual image frames. Each image frame contains a timestamp at which the frame is captured. The time stamp is used to distinguish the images of the beehive at different times. In our research, a challenge was to extract the time and date from the images and to store them in the database along with the corresponding images. The steps involved in processing the individual images will be described in this chapter.

3.2 Image Enhancement

In every digital image processing analysis, the first step is very likely applying filters and enhancing images. Image enhancement adjusts the digital images such that they are suitable for further processing. For example, the enhancement process may make an image brighter or darker depending on the quality of the image and the desirable outcome. Most of the images taken by digital cameras contain some unwanted information that is referred to as noise. The goal of image enhancement is to reduce or eliminate the noise level inside the image.

Most image enhancement techniques are applied on the gray-scale images. Gray-scale images in 8-bit format have pixels with intensity values between 0 and 255 for each pixel. In such a gray-scale image, intensity value 0 represents black and 255 represents white. To enhance color images in RGB format, the target images are often converted to gray-scale first and then image enhancement techniques are applied.

The enhancement methods fall into two broad categories:

- Spatial Domain Methods
- Frequency Domain Methods

Spatial domain methods process the image directly using the intensity value of pixels in the gray level image. On the other hand, the frequency domain methods first transfer the image into frequency domain by applying the Fourier Transform on the image. After completion of analyses in the frequency domain, the image is transferred to image space by applying the Inverse Fourier Transform. Some of the branches of these two main categories of image enhancement will be explained in the following sections.

3.2.1 Image Negatives

The negative of a gray-scale image is obtained by subtracting the intensity values of each pixel from the maximum possible intensity value on that image. For example, in an 8-bit image the maximum possible value for a pixel is 255, which represents the white color on the image [Gonzalez and Woods 2001]. In a more general form, when the intensity values are in the $[0, L-1]$ range, the negative transformation is obtained as in 3.1.

$$s = L - 1 - r, \tag{3.1}$$

where r represents the original intensity value and s is the inverted intensity, i.e., the transformed value. Image negative technique is suitable for enhancing details embedded in the dark regions of an image [Gonzalez and Woods 2001].

3.2.2 Log Transformation

Another method of image enhancement is the Log Transformation. The general form of the Log Transformation is shown in 3.2 [Jain et al. 1995].

$$s = c \times \log(r + 1), \tag{3.2}$$

where c is a constant and $r \geq 0$ represents the intensity value of the original image. The Log Transformation maps a narrow range of low input gray values into a wider range of output values. The inverse of the Log Transformation does the exact opposite. This transformation is useful in cases where the image has a large dynamic range of values such as those seen in a Fourier spectrum.

3.2.3 Power-law Transformations

The Power-law Transformation can have the basic form as in 3.3 [Gonzalez and Woods 2001].

$$\mathrm{s} = \mathrm{c} \times r^{\gamma}, \tag{3.3}$$

where c and γ are positive constants. Sometimes this equation is written as presented in 3.4 to account for an offset, ε [Gonzalez and Woods 2001].

$$s = c \ \times (r + \varepsilon)^{\gamma}. \tag{3.4}$$

By convention, the exponent part is represented by gamma, hence in some literature this transformation is also referred to as the *gamma correction*. In addition to correcting the gamma factor, the Power-law Transformation is useful for general purpose contrast manipulation. By applying the Power-law Transformation, the visibility of details can be changed in an image. For example, in an image of a city taken from an airplane, some details may not be visible. However, by applying a gamma correction with a gamma value greater than 1, significant details can be revealed in the image, as shown in Figure 3.1.

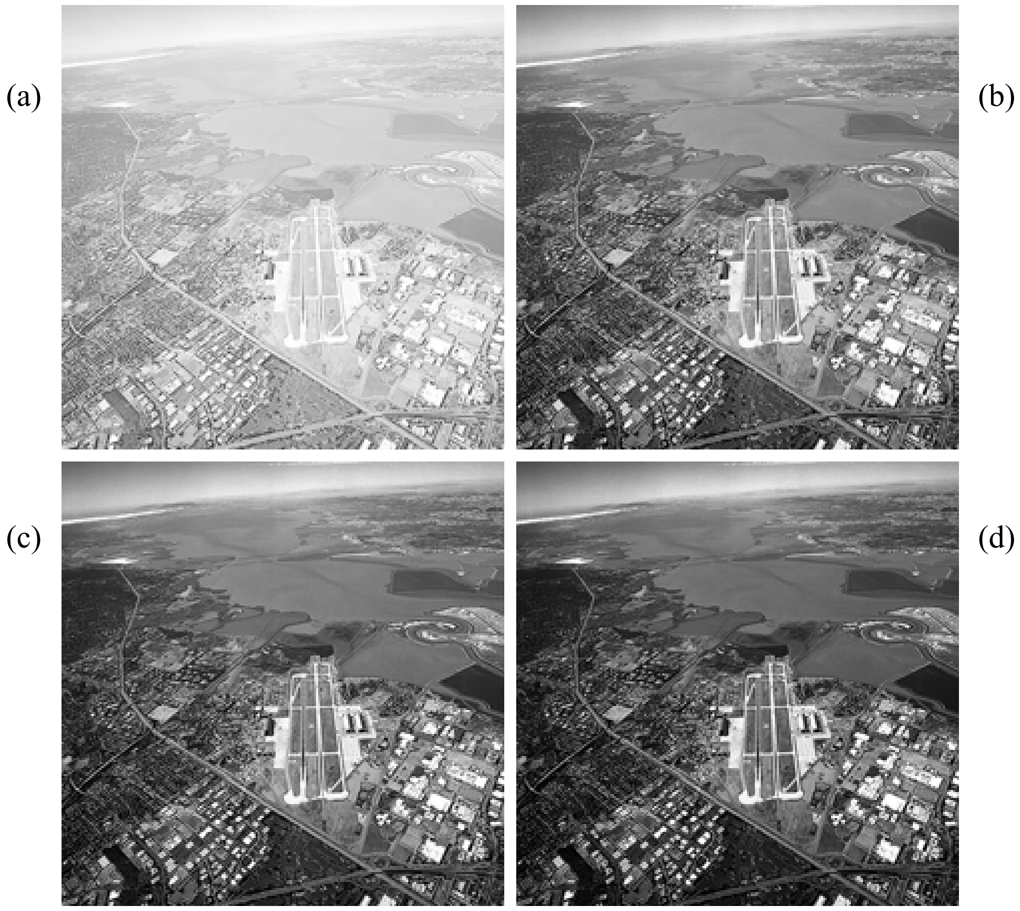


Figure 3.1: (a) Aerial image. (b)- (d) Results of applying power law transformation [Gonzalez and Woods 2001]

3.2.4 Piecewise-linear Transformation

Piecewise-linear Transformation is another method for image enhancement. The advantage of this method is that the form of piecewise function can be arbitrarily complex. A drawback of this method is that it requires a large number of inputs from the user to create the piecewise function.

There are different types of piecewise functions. Some of the most common ones are:

- Contrast stretching
- Gray level slicing
- Bit-plane slicing

One of the simplest functions of the Piecewise-linear Transformations is contrast stretching. Contrast Stretching Transformations increase the contrast between the darks and the lights in an image. In more general fields of data processing, such as digital signal processing, this transformation is referred to as dynamic range expansion. The purpose of dynamic range expansion in various applications is usually to bring the image or other type of signal into a range that is more familiar or normal to the senses. For example, a newspaper will strive to make all of the images in an issue share a similar range of grayscale [Gonzalez and Woods 2001].

For cases in which there is a need to highlight a specific range of gray level in an image, gray level slicing is being used. Applications of gray level slicing include enhancing features such as masses of water in satellite imagery and enhancing flaws in X-ray images. There are two basic themes of doing gray slicing. One approach is to display a high value for the range of interest and a low value for other pixels. The second approach is to display a high value for the range of interest, but preserve the other pixels [Gonzalez and Woods 2001]. Depending on the application, one of the approaches or the combination of these approaches can be used.

Given an image with X-bit gray level pixel value, slicing an image at different bit planes play an important role in image processing. An application of bit plane slicing is in the data compression [Jordan and Lotufo 1997]. If the pixels of an image are represented by 8-bit values, the image can be sliced in 8 different bit planes. The planes range from 0 to 7, where the 0 plane represents the least significant bit and the 7 plane represents the most significant bit [Gonzalez and Woods 2001].

3.3 Image Histogram Processing

Given an image with pixel values in the range of $[0, L-1]$, the histogram of the image is a discrete function defined as in equation 3.5.

$$h(r_k) = n_k, \tag{3.5}$$

where r_k is the k_{th} gray level and n_k is the number of pixels in the image with gray level r_k. A normalized histogram is given by $p(r_k) = {}^{n_k}/_{n}$, where n is the number of pixels in the image [Gonzalez and Woods 2001]. Histogram of an image is the base for many spatial domain processing and enhancement methods.

Two common categories of image histogram processing are:

- Histogram Equalization
- Histogram Matching (Specification)

Histogram equalization is a contrast adjustment technique that uses the image's histogram. This method usually increases the global contrast of an image, especially when the usable data of the image are represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. This allows for areas of lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values [Laughlin 1981].

Histogram matching is a process where a time series, image, or higher dimension scalar data are modified such that its histogram matches that of another (reference) dataset. A common application is to match the images from two sensors with slightly different responses or from a sensor whose response changes over time [Laughlin 1981].

Other enhancement methods that are widely used in image processing use arithmetic or logic operations. In the following sections some of these methods will be briefly introduced.

3.3.1 Image Subtraction

This method computes the difference between the corresponding pixels on two similar size images as shown in 3.6.

$$g(x, y) = f(x, y) - h(x, y), \tag{3.6}$$

where $f(x, y)$, and $h(x, y)$ are corresponding pixel values in the matrices f and h that represent the two images. The subtracted image holds the difference between all pairs of corresponding pixels. The key benefit of image subtraction is the enhancement of difference between the two target images [Gonzalez and Woods 2001].

3.3.2 Image Averaging

One of the best methods to decrease the effect of noise in an image is image averaging. Consider a noisy image $g(x, y)$ formed by adding noise $n(x, y)$ to the original image $f(x, y)$ as shown in 3.7.

$$g(x, y) = f(x, y) + n(x, y). \tag{3.7}$$

The general idea of image averaging is that for every pair of coordinates (x, y) on the image where the noise is uncorrelated, the amount of noise can be reduced by adding a set of noisy images to the original image [Gonzalez and Woods 2001] as shown in Figure 3.2.

3.4 Image Filters

Most of the enhancement methods introduced in the previous sections use filters to apply that enhancement to the images. Some of the more common spatial filters will be introduced in the following sections.

3.4.1 Smoothing Spatial Filters

Smoothing spatial filters are used to blur the images and reduce the noise. They are widely used in software tools, typically to reduce image noise and details. There are different filters for smoothing images. Blurring is used in pre-processing steps, such as removing small details from an image prior to extracting large objects or bridging small gaps between lines or curves on the image. Noise reduction can be accomplished by blurring with either a linear or non-linear filter. There are different methods to smooth an image, but among the most popular ones are the linear filter and the order-statistic filter [Gonzalez and Woods 2001].

One of the most commonly used smoothing filters is the Gaussian filter. The Gaussian filter is a type of image-blurring filter that uses a Gaussian function (which also expresses the normal distribution in statistics) for calculating the transformation to apply to each pixel in the image. The Gaussian function in one dimension is expressed as shown in 3.8.

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}, \tag{3.8}$$

where σ is the standard deviation.

For data in two dimensions such as images, it is the product of two such Gaussians, one in each dimension [Stockman and Shapiro 2001; Nixon and Aguado 2008], as shown in 3.9.

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{3.9}$$

where x is the distance from the origin in the horizontal axis, y is the distance from the origin in the vertical axis, and σ is the standard deviation of the Gaussian distribution [Figure 3.2].

3.4.2 Sharpening Spatial Filters

Sharpening filters are used to sharpen images. Sharpening an image creates an image that is less blurry than the original image. The principal objective is to highlight fine details in an image. Image sharpening is used for various purposes and applications ranging from electronic printing, medical imaging, industrial inspection, and autonomous guidance in military systems [Figure 3.2].

(a) (b) (c) (d)

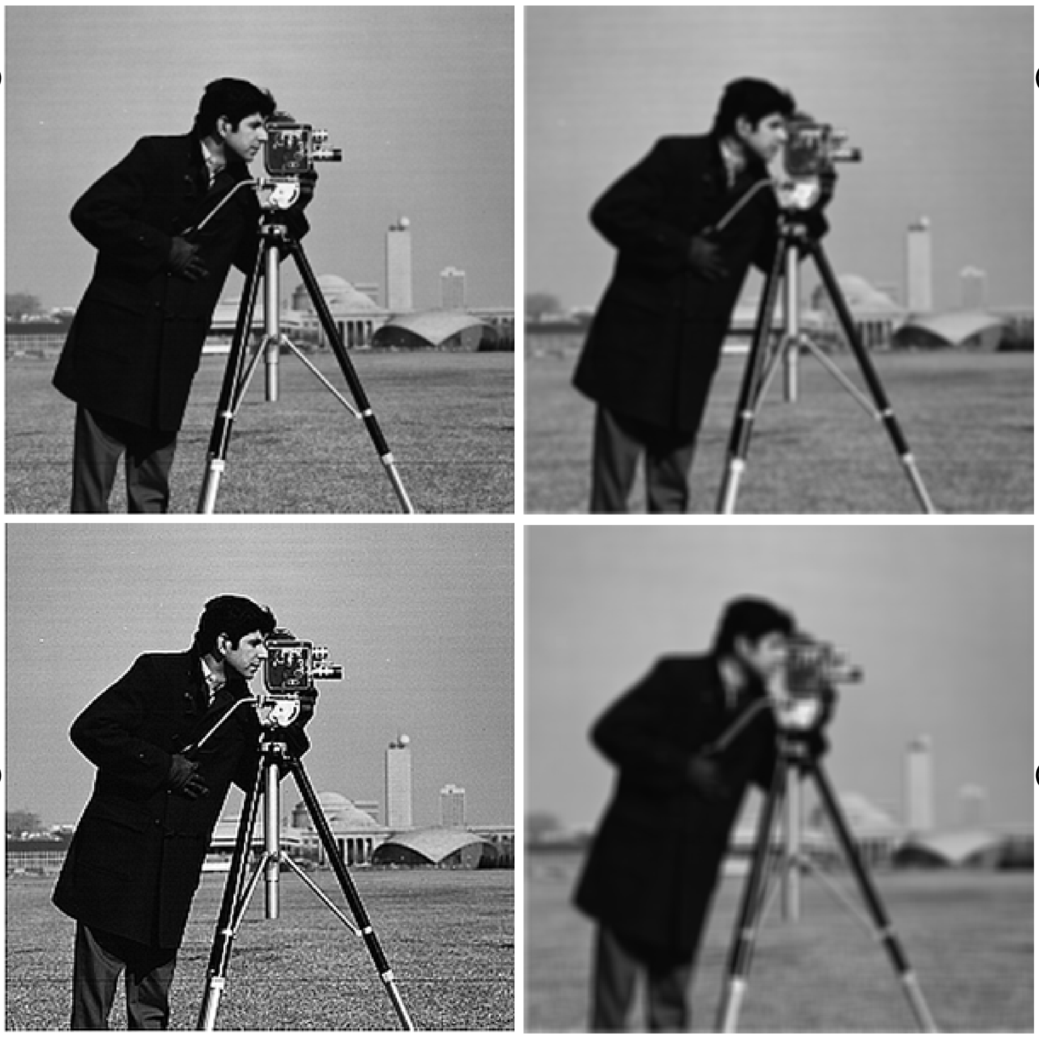


Figure 3.2: Applying different filters to an image (a) Original image [MathWorks Inc. 2011] (b) Smoothing filter (c) Sharpening filter (d) Gaussian

3.5 Frequency Domain Enhancements

Another type of enhancement that utilizes the Fourier transform of the target image is called the frequency domain enhancement. As described in the preceding sections, these enhancement methods process an image in the frequency domain. Some of these methods will be introduced briefly in the following sections.

3.5.1 Notch Filter

Suppose that there is a need to force the average value of an image to zero. To make this happen, the value of the Fourier's principal frequency $F(0,0)$, which is equivalent to the average of the image, must be set to zero. This operation can be done by multiplying all values of $F(u,v)$ by the filter function as shown in 3.10.

$$H(u,v) = \begin{cases} 0 & if\ (u,v) = (M/2\,,N/2) \\ 1 & otherwise \end{cases}, \qquad (3.10)$$

where u and v are Fourier transform variables, and M and N are width and height of the image respectively. Following this step, an inverse Fourier transform is applied to $H(u,v)$ which will result in an image with zero average.

Notch filters are exceptionally useful in identifying spatial image effects caused by specific localized frequency domain components [Gonzalez and Woods 2001].

3.5.2 Low-pass and High-pass Filters

In the frequency domain of an image, the low frequency components are responsible for the general gray level appearance of an image over smooth areas, while the high frequency components are responsible for details, such as edges and noise. As a result, a filter that attenuates high frequencies while "passing" low frequencies is called a low-pass filter. Such a filter is used to smooth the image. On the other hand, a filter that has the opposite characteristics is appropriately called a high-pass filter which sharpens the image.

3.6 Image Segmentation

Image segmentation is the process of partitioning a digital image into multiple segments, also known as sets of pixels or super-pixels. The goal of segmentation is to simplify or change the representation of an image into something that is more meaningful and easier to analyze [Stockman and Shapiro 2001]. Image segmentation is typically used to locate objects and boundaries, such as lines and curves in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics.

The result of image segmentation is a set of segments that collectively cover the entire image or a set of contours extracted from the image. All of the pixels in a region are similar with respect to some characteristics or computed properties such as color, intensity, or texture. Adjacent regions are significantly different with respect to the same characteristic(s) [Stockman and Shapiro 2001].

Several general-purpose algorithms and techniques have been developed for image segmentation. In the following sections, some of these methods will be briefly introduced.

3.6.1 Thresholding

The simplest method of image segmentation is called the thresholding method. This method is based on a clip-level (or a threshold value) to turn a gray scale image into a binary image. The key of this method is to select the threshold value (or values when multiple levels are desired). The algorithm then classifies the pixels into different segments according to the threshold value(s). Several popular thresholding methods are used in industry including the maximum entropy method, Otsu's method (maximum variance), and k-means clustering.

In some recent research projects, new methods have been developed for thresholding Computed Tomography (CT) images. The key idea is that unlike other methods, the thresholds are derived from the radiographs instead of the reconstructed image [Batenburg and Sijbers 2009a; Batenburg and Sijbers 2009b].

3.6.2 Edge Detection

Edge detection is a well-developed field of image processing. Since there is often a sharp adjustment in intensity at the region boundaries, region boundaries and edges are closely related. Edge detection techniques have therefore been used as the base of another segmentation technique. Edge detection is by far the most common approach for detecting meaningful discontinuities in gray level.

The edges identified by edge detection algorithms are often disconnected. To segment an object from an image, however, one needs asset of closed region boundaries. The desired edges are the boundaries between such objects.

Segmentation methods can also be applied to edges obtained from edge detectors. Lindeberg and Li [1995] developed an integrated method that segments edges into straight and curved edge segments for parts-based object recognition. This method is based on a minimum description length (MDL) criterion that is optimized by a split-and-merge-like method. This method can be used to obtain more likely points at which to consider partitions into different segments.

3.6.3 Region-growing Methods

As its name implies, region growing is a procedure that groups pixels or subregions into larger regions based on predefined criteria. The basic approach is to start with some seed points, and from these points grow regions. The region for a seed grows by appending to each seed those neighboring pixels that have properties similar to the seed (such as a specific range of gray level or color). In that case, it is called seeded-region-growing method.

Seeded-region-growing methods require seeds as additional inputs. The segmentation results are dependent on the choice of seeds. Noise in the image can cause the seeds to be poorly placed. Unseeded-region-growing is a modified algorithm that does not require explicit seeds, starting off with a single region A_1. The pixel selected for processing does not significantly influence the final segmentation. At each iteration, the algorithm considers the neighboring pixels in the same way as seeded-region-growing. This method differs from the seeded-region-growing method in that if the minimum δ is less than a predefined threshold T then it is added to the respective region A_j. If not, then the pixel is considered significantly different from all current regions A_i and a new region A_{n+1} is created with this pixel.

One variant of this technique, proposed by Haralick and Shapiro [1992], is based on pixel intensities. The mean and scatter of the region and the intensity of the candidate pixel is used to compute a test statistic. If the test statistic is sufficiently small, the pixel is added to the region and the region's mean and scatter are recomputed. Otherwise, the pixel is rejected and is used to form a new region.

3.6.4 Split-and-merge Method

Split-and-merge segmentation, sometimes called quad-tree segmentation, is based on a quad-tree partition of an image. This method begins at the root of the tree that represents the whole image to determine if the image is non-uniform (not homogeneous). If so, the image is split into four son-squares (the splitting process), recursively. Conversely, if four son-squares are homogeneous, they can be merged as several connected components (the merging process). The node in the tree is a segmented node. This process continues recursively until no further splits or merges are possible [Horowitz and Pavlidis 1974; Horowitz and Pavlidis 1976]. The optimal time complexity of this method is $O(nlogn)$. When a special data structure is involved in the implementation of the method, the optimal time complexity can be achieved [Chen 1991].

3.7 Optical Character Recognition

Optical Character Recognition (OCR) is a method for detection and recognition of characters and numbers and converting them into digital format. It deals with the problem of recognizing optically processed characters. Optical recognition is performed off-line after the writing or printing has been completed, as opposed to on-line recognition where the computer recognizes the characters as they are drawn. Both hand written and printed characters may be recognized, but the performance is directly dependent upon the quality of the input documents. The more constrained the input is, the better the performance of the OCR system will be. However, when it comes to totally unconstrained handwriting, OCR machines are still a long way from reading as well as humans [Eikvil 1993].

Character recognition techniques associate a symbolic identity with the image of the target character. The modern version of OCR appeared in the middle of the 1940s with the development of the digital computers. OCR machines have been commercially available since the middle of the 1950s. Nowadays, OCR systems are available both as hardware devices and software packages, and a few thousand systems are sold every week.

A typical OCR system consists of several components. The first step in the process is to digitize the analog image/document using an optical scanner. When the regions containing text are located, each symbol is extracted through a segmentation process. The extracted symbols may then be preprocessed to facilitate the extraction of features in the next step. This step also reduces the noise in the scanned image. The identity of each symbol is found by comparing the extracted features with descriptions of the symbol classes obtained through a previous learning phase. Finally, contextual information is used to reconstruct the words and numbers of the original text [Eikvil 1993].

One of the simplest techniques for implementing OCR is to have a stored pattern for every character and then compare the similarity of the unknown character with all the stored patterns. An unknown character is then recognized as the one with the highest similarity. Although this method is not very accurate in most cases, it is often one of the first options, as it is not computationally intense.

For the purpose of this research, since the only patterns that need to be recognized are the numbers from 0 to 9 that are used for expressing time and date on the image, a simple and fast OCR method is implemented to recognize the digits from the timestamp in an image.

3.8 Identifying and Counting Bees

A goal of this research is to successfully identify and estimate the number of honey bees in front of the beehive. As such, several different approaches are used to accomplish this. In the following sections some of the methods will be discussed.

3.8.1 Change Detection

Change detection algorithms apply image processing techniques on two or more separate images to find the difference between the images of the object at different stages. These techniques are usually utilized for detecting motion in a sequence of images. An example is detection of moving objects in the videos obtained by a surveillance system. These methods are becoming more popular due to their vast applications in remote sensing, surveillance, medical diagnosis and treatment, civil infrastructure, and underwater sensing.

Change detection attempts to identify pixels that are significantly different in a given set of images that are taken from the same scene at different times. The significant change here means that the differences between pixel values are resulted from the moving objects. In other words, the difference in color of an object that may be due to illumination changes should not be distinguished as a significant change. The pixels that show a significant difference are represented as the change mask.

The definition of the change mask depends on the application for which it is used. For example, for the purpose of this research the change in the color of the beehive is not observed as a significant change, thus it should not be included in the change mask. This is very different in a remote sensing application where it might be useful to account the color changes in the change mask.

The goal of the change detection algorithms is to detect significant changes. To eliminate insignificant changes, some pre-processing algorithms may be applied to the target image. Two of these pre-processing algorithm are Geometric Adjustment, and Radiometric/Intensity Adjustment [Radke et al. 2005].

A. Geometric Adjustment

Some of the changes in the video recordings are due to the unwanted movements of the camera. In most cases, these changes are undesired and must be eliminated before the change detection methods are applied to the image. The adjustment of the image due to the motion of the camera is referred to as Image Registration. In most applications, such as the one on this research or that on a surveillance system, the camera is mounted and the motion is small. In such cases, the image registration can be performed using a low-dimensional spatial transformation, such as similarity, affine, or projective transformation. One of the application of image registration is in commercial digital cameras. When an image is taken, image registration is applied to the image to eliminate the changes resulted from hand shaking. The problem of geometric adjustment has been well studied and several software applications are available for image registration [Radke et al. 2005].

B. Radiometric/Intensity Adjustment

Some of the changes in image sequences taken by a video camera are resulted from the changes in the strength or position of the light source in the image. In many applications, these changes are classified as “unimportant” changes. Radiometric/Intensity adjustment methods are applied to eliminate these unimportant changes. There are several techniques to achieve this with reasonable success. In the following sections, several techniques for radiometric or intensity adjustment will be introduced.

1) Intensity Normalization

One of the simplest methods for illumination-invariant change detection is to normalize the intensity level of the target image such that its mean and variance are the same [Lillestrand 1972; Ulstad 1973; Dai and Khorram 1998]. This calculation is shown in 3.11.

$$\tilde{I}_2(x) = \frac{\sigma_1}{\sigma_2}\{I_2(x) - \mu_2\} + \mu_1, \tag{3.11}$$

where $\tilde{I}_2(x)$ is the normalized second image and μ_i and σ_i are the mean and variance of intensity values of I_i, respectively.

Instead of applying the normalization 3.11 to each pixel using the global statistics μ_i and σ_i, the images can be divided into several corresponding disjoint blocks, and the normalization is performed independently using the local statistics of each block. This can achieve better local performance at the expense of introducing blocking artifacts.

2) Homomorphic Filtering

For images of scenes containing Lambertian surfaces, the observed image intensity of a pixel x can be modeled as the product of two components: the illumination $I_l(x)$ from the light source(s) in the scene and the reflectance $I_o(x)$ of the object surface to which x belongs, as shown in 3.12.

$$I(x) = I_l(x)I_o(x). \tag{3.12}$$

This is called the Shading Model [Phong 1975]. Only the reflectance component $I_o(x)$ contains information about the objects in the scene. Due to this fact, a homomorphic filter can be used to separate the two components of the intensity signal. To obtain this, logarithms are taken on both sides of 3.12 to obtain 3.13.

$$\ln I(x) = \ln I_l(x) + \ln I_o(x). \tag{3.13}$$

Since the lower frequency component is now additive, it can be separated using a high-pass filter. The reflectance component can thus be estimated as 3.14 [Radke et al. 2005].

$$\tilde{I}_o(x) = exp\{F(\ln I(x))\}, \tag{3.14}$$

where $F(.)$ represents a high-pass filter. The reflectance component can be provided as the input to the decision rule step of a change detection process [Toth et al. 2000; Aach et al. 2001].

3) Linear Transformations of Intensity

Another method for reducing illumination changes is to use eigenvalue of the image to determine the changed parts of the image. The parts of the image corresponding to the large eigenvalues are estimated to reflect the parts with changes, and those parts corresponding to the smaller eigenvalues are determined as the unchanged parts. The difficult part in this method is to determine the principal components of the changes without the need of visual inspection [Radke et al. 2005].

4) Sudden Changes in Illumination

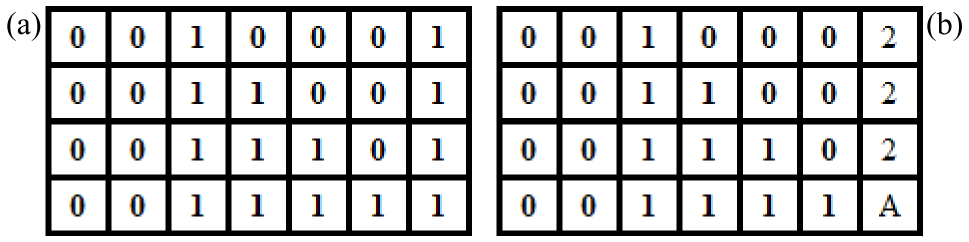
Xie et al. [2004] observed that under the Phong shading model with slowly spatially varying illumination, the sign of the difference between corresponding pixel measurements is invariant to sudden changes in illumination. This result can be used to create a change detection algorithm that is able to discriminate between “uninteresting” changes caused by a sudden change in illumination (e.g., turning on a light switch) and “interesting” changes caused by object motion.

As explained in the preceding sections, there are different methods to detect changes. For the purpose of this research, an illumination invariant algorithm is used [Durucan and Ebrahimi 2000]. This algorithm uses the linear independency between parallel vectors to separate the changes caused by illumination from the changes caused by movement of objects in the scene. This method is explained in more detail in Chapter 5.

3.9 Counting Methods

After applying change detection methods on a target image, the result is a change mask that has the same size as the reference image. This mask illustrates the changes between the reference image and another image. The number of changes in the image is a good approximation of objects that have moved in the scene at the time the image is taken, assuming that only the significant changes are represented in the change mask. If the changes in the image are represented by 1 and the background is represented by 0 in the change mask, then by counting the number of separated white parts (pixel value of 1) of the change mask, the result would be a good approximation of the number of moving objects. One of the methods most commonly used to count the number of regions in an image and labeling them is called connected components labeling [Haralick and Shapiro 1992]. This operation performs one unit change from a pixel to a region or from a pixel to a segment. All pixels that have value 1 and are connected to each other by a path of pixels with value 1 are given the same identifying label. This label is a unique identifier for each region. Since the building blocks of a digital image are its pixels, this counting algorithm is applied to pixels. In this process, the connected component labeling is a conversion from pixels to regions. The number of labels determines the number of regions detected as separate changes in the scene.

There are several algorithms to implement the connected component labeling. All of these algorithms process the image row-by-row and then label the connected pixels. For example, consider Figure 3.3.a, and assume a 4-adjacency with left-to-right, top-to-bottom scan order. In the first row, two 1 pixels separated by three 0 pixels are encountered which will be assigned label 1 and label 2, respectively. In row two the first 1 pixel is assigned label 1 because it is a 4-neighbor of the already-labeled pixel above it. The second 1-pixel on row 2 is also assigned label 1 because it is in the 4-neighbor of the already-labeled pixel on its left. This process continues until the pixel marked 'A' in row 4 is encountered [Figure 3.3.b]. Pixel 'A' has a pixel labeled 2 above it, and it connects regions 1 and 2. Thus all the pixels labeled 2 and all the pixels labeled 1 belong to the same component. There are different approaches to solve this problem. Two of these approaches will be explained briefly in this section.



(a)

0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	0	1	1	1	0	1
0	0	1	1	1	1	1

(b)

0	0	1	0	0	0	2
0	0	1	1	0	0	2
0	0	1	1	1	0	2
0	0	1	1	1	1	A

Figure 3.3: Propagation process. Label 1 has been propagated from the left to reach pixel A. Label 2 has been propagated down to reach pixel A. The connected components algorithm must assign a label to A and make labels 1 and 2 are equivalent [Haralick and Shapiro 1992].

Iterative algorithm: This algorithm uses an initialization step to label all the pixels with value 1. This step is followed by a sequence of top-down label propagation followed by the bottom-up label propagation iterated until no label changes occur. This algorithm is useful on computers on which the memory size in not a problem and also in SIMD hardware [Haralick and Shapiro 1992].

Classical algorithm: Compared to the iterative algorithm, the classical method only passes through the image twice, which makes the algorithm faster. The disadvantage of this method is that it requires a large global table for storing the equivalences. The first pass finds the difference labels in a similar approach as explained in the previous section. With this approach, whenever there is a pixel that can have both labels, it will be labeled with the smaller value. After the first pass, the equivalence classes are found by taking the transitive closure of the set of equivalence recorded in the Equivalence Table. This step is called the “Resolve” step which is a standard algorithm discussed in many algorithm books, such as Aho et al. [1983]. Each equivalence class is assigned a unique label, usually the minimum (or oldest) label in the class. Finally, a second pass through the image performs a translation which is basically assigning to each pixel the label of the equivalence class of its pass-1 label.

Although these two algorithms work reasonably well in assigning a label to each object in the image, they both use a significant amount of memory. Because of this drawback, other algorithms have been developed that label images faster and require less memory. One of those methods is used for the purpose of this research and it is explained in Chapter 5 in more detail.

3.10 SNR and Entropy

Signal-to-noise ratio (SNR) is a measure of a desired signal to the background noise. SNR is an indication of the amount of noise added to the image data. A high SNR value is often seen as an indicator of the image quality. The SNR can be applied to digital images to calculate the level of the noise and also to measure the effectiveness of different filters. Different information can be retrieved from the SNR of an image which makes it an excellent candidate for image analysis. For example, if the background of an image taken from a beehive is defined as the desired signal, then the changes due to illumination or moving object can be interpreted as the noise. The SNR of an image can be calculated using 3.15 [Brislawn 1995].

$$SNR = 10 \times \log_{10}\left(\frac{\sum_{i,j}(orig_{i,j})^2}{\sum_{i,j}(enc_{i,j} - orig_{i,j})^2}\right), \tag{3.15}$$

where $orig_{i,j}$ refers to the pixel value at location i, j in the original image and $enc_{i,j}$ represents the pixel value at the same location in the noisy image.

Entropy is a statistical measure of randomness that can be used to characterize the texture of a target image [Gonzalez and Woods 2001]. When applied to an image, entropy is a measure of the amount of details in the image. The entropy value is calculated using 3.16.

$$Entropy = \sum_j \Pr(a_j) \log_2 \Pr(a_j), \tag{3.16}$$

where a_j represents the j_{th} unique intensity value and $\Pr(a_j)$ is the probability of that pixel value occurring within the image. Usually, entropy is calculated for grayscale images; however it can be defined for RGB images by treating them as a multidimensional grayscale image [Gonzalez et al. 2003]. In information theory, entropy represents the level of uncertainty in a random variable. In the context of image processing, the term usually refers to the Shannon Entropy, which quantifies the expected value of the information contained in a message [Gonzalez and Woods 2001]. In physics, entropy is a mathematically defined function that accounts for the flow of the heat through the Carnot Cycle.

CHAPTER 4 EXTRACTION OF DATA FROM IMAGES

4.1 Introduction

To observe the behavior of honey bees and to find a relationship between their activities and local weather data such as temperature and humidity, an automatic system is desirable. As part of this research, a system is designed and implemented to monitor the entrance of beehives using video cameras. The flowchart of the system is shown in Figure 4.1. This system is divided into two main parts: image processing and data acquisition and analysis system. Implementation and design of image processing part is described in this and the next chapter. The other parts of the system that contain gathering images, weather data, and iPhone application are described in Chapter 6.

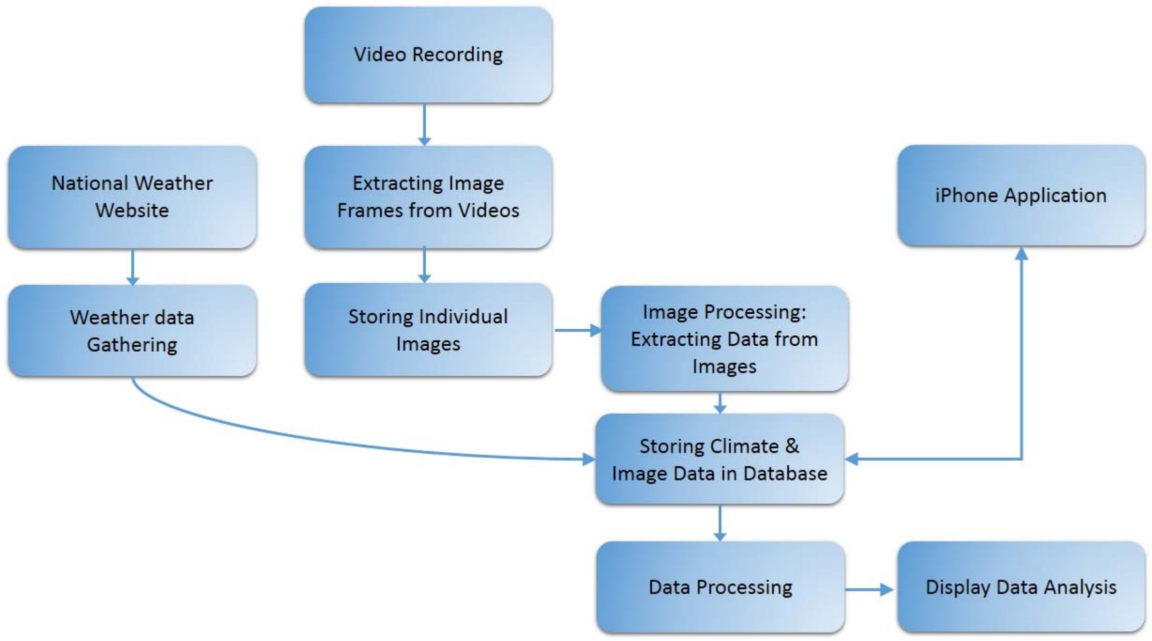

Video Recording
National Weather Website
Extracting Image Frames from Videos
iPhone Application
Weather data Gathering
Storing Individual Images
Image Processing: Extracting Data from Images
Storing Climate & Image Data in Database
Data Processing
Display Data Analysis

Figure 4.1: Flowchart of the system designed for studying the activity of honey bees

The processing of images that are taken from a beehive is conducted in two independent steps: one step is to extract the date and time from images and the other step is to count the number of bees. Time and date are embedded in the videos taken from a beehive. In this chapter, the methods used for extracting time and date are explained.

4.2 Timestamp Extraction

To have an easy access to images that are taken from a beehive, they need to be stored in a database. In this research, a relational database is designed using MySQL to store the data. To retrieve and also have easy access to the data, a primary key is needed to distinguish images from each other. For the purpose of this research, time and date are selected as the primary key for each tuple in the database. The cameras that are used in this research embed time and date in the videos. This timestamp is placed at the bottom right corner of the images (Figure 4.2).

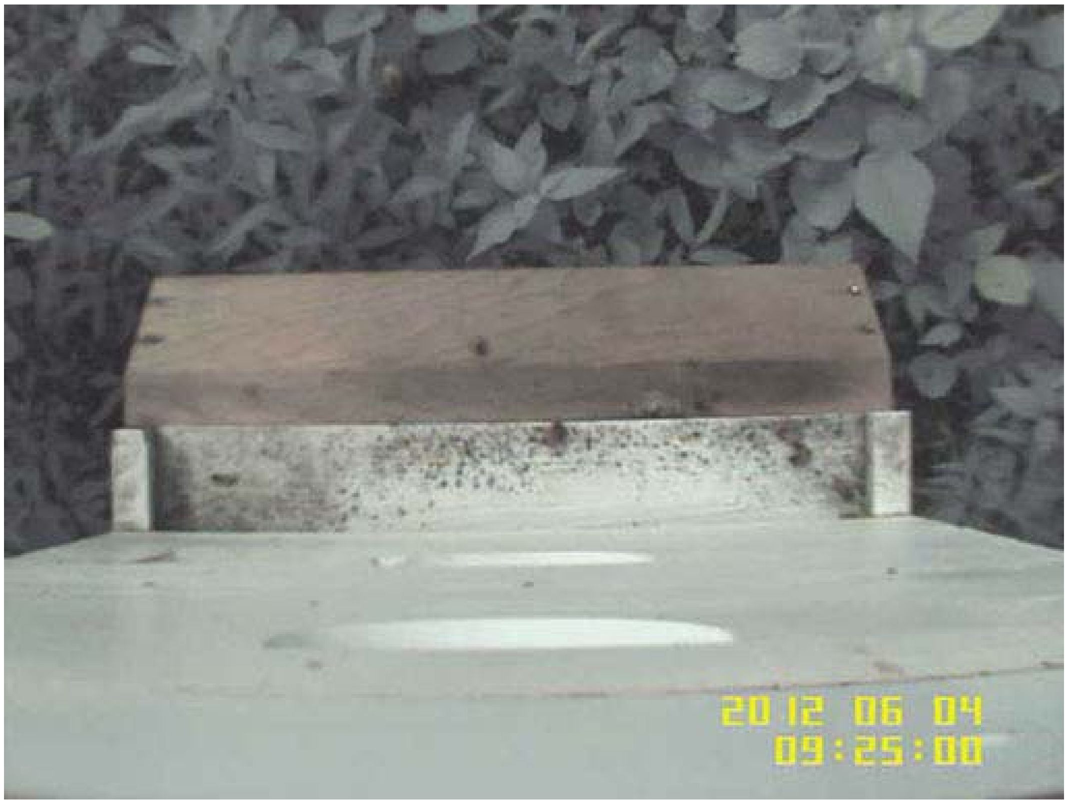


Figure 4.2: Top view of the beehive entrance with timestamp

As shown in Figure 4.2, the timestamp has a distinct color from the whole image. This feature can be used to extract the timestamp from the images. Also, the position of the timestamp is the same in all images. The location and the color of the timestamp can be used to separate it from the background of the image in an optimized way.

The implemented program for separating the timestamp from the background goes through each image pixel-by-pixel to find the timestamp, using the color value of the pixels in the timestamp. This process is time consuming since the algorithm needs to check each pixel on the image. As mentioned above, the position of the timestamp in the images never changes. This fact can be used to optimize the time aspect of the algorithm. As a result, the searching algorithm is only applied to the portion of image that the timestamp is located. The size of the images taken by the camera is 640 × 480 pixels, and only a small portion of the image (96 × 147 pixels) used for the timestamp is processed by the algorithm. This portion has 20 times fewer pixels than the original image which makes the process of finding the timestamp 20 time faster (Figure 4.3).

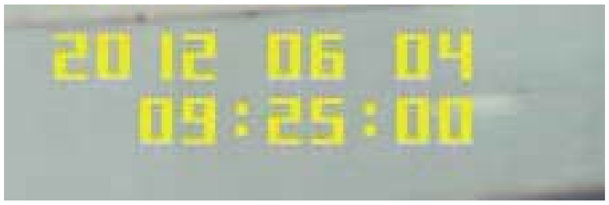


Figure 4.3: The timestamp image section

4.3 Segmentation of the Timestamp from Images

The next step in extracting time and date from the image is to separate the timestamp from the other part of the image (i.e., the background). Since the color of the timestamp never changes, and is distinctive from the background, it can be used in a searching algorithm. To use a search algorithm based on the color in an image, the Cylindrical Coordinate format is used. This format is the closest model to the human interpretation from colors and is abbreviated as HSV (Hue-Saturation-Value). To find the timestamp in an image using the HSV format, the algorithm only needs to process the Value component of the format. This is another reason for using HSV over other color models. For example, to do the same process in the Red-Green-Blue (RGB) format, all the three components of the color space should be processed to find the desired color. The three components of the timestamp image in the HSV format are depicted in Figure 4.4.

As shown in Figure 4.4, the value part of the image can be used to extract the timestamp from an image.

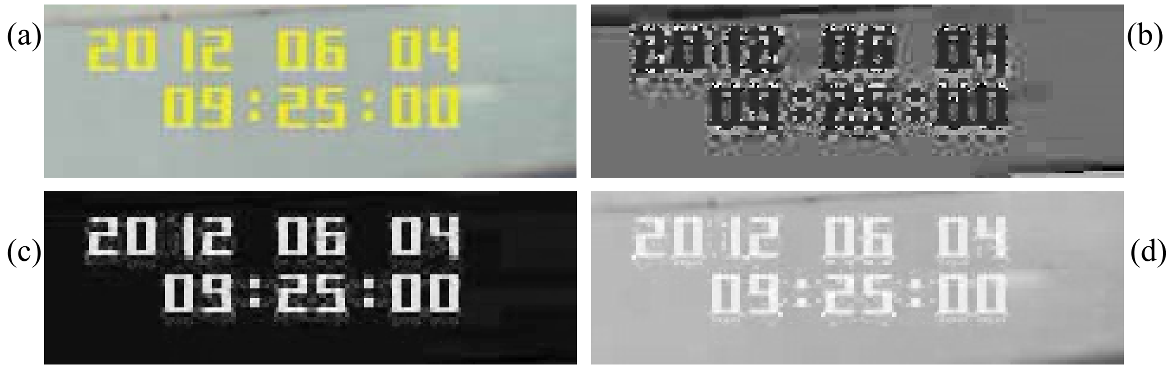


Figure 4.4: (a) Original image, (b) Hue component (c) Saturation component (d) Value component

To produce the binary image, the HSV format of the image is computed and the value component is used to find the timestamp. The implemented search algorithm goes through the value component of this format and checks the value of each pixel. If the value of the pixel is greater than a threshold value, it is estimated as a part of the timestamp, otherwise it is estimated as part of the background, as shown in 4.1.

$$B(x,y) = \begin{cases} 0 & if \;\; V(x,y) < threshold \\ 1 & otherwise \end{cases}, \tag{4.1}$$

where $B(x,y)$ is the binary image with x and y coordinates and $V(x,y)$ is the value part of the timestamp image with the same coordinates.

In the resulting binary image, the pixels corresponding to the timestamp get value 1 or maximum possible value and the background pixels get 0, as shown in Figure 4.5.

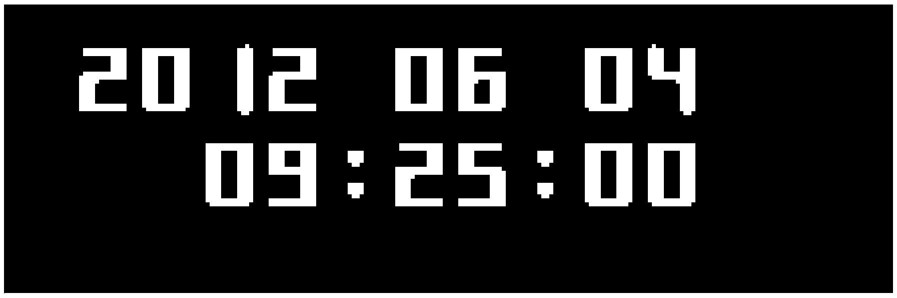


Figure 4.5: Enlarged timestamp binary image after applying threshold

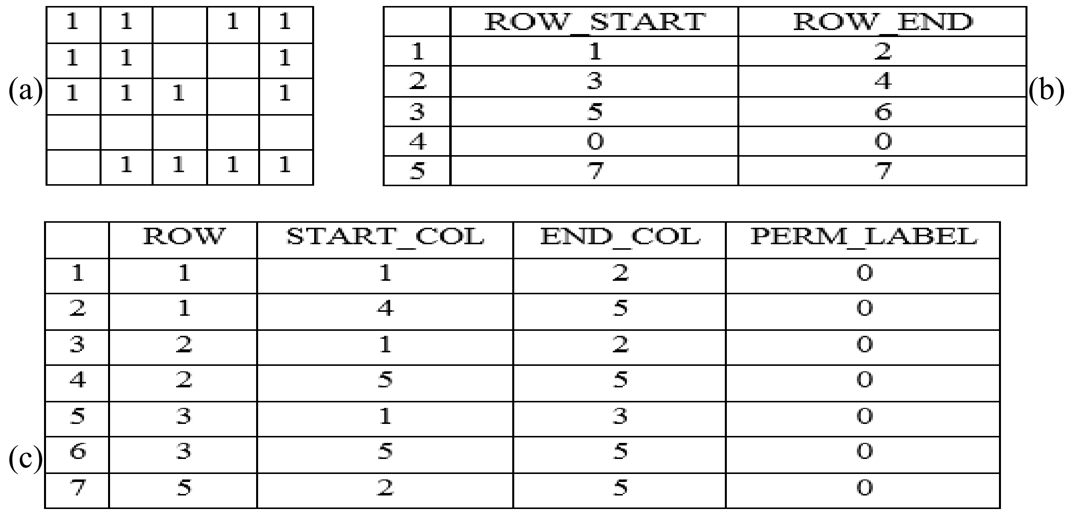
The resulting image from the last step is a binary image consisting of the timestamp and black background. This image can be used in the next step to extract time and date where the image is processed and the digits in the image are recognized. Before this step, each digit in the image should be separated from other digits in the timestamp.

4.4 Separating the Digits

The size of images used in this research is 640 × 480 pixels. To separate the digits from each other as different segments of the image, either the iterative or classical method explained in Chapter 3 can be used. Using one of these algorithms, depending on the size of images, processing an image to separate the digits would take more than ten seconds. To reduce this time for the purpose of this research and to achieve a faster algorithm, the separation process is implemented using an efficient run-length branch of the local method.

This process starts by making a run-length encoding of a binary image. A run-length encoding of a binary image is a list of contiguous, typically horizontal runs of 1-pixel. For each run, the location of the starting pixel of the run and either its length or the location of its ending pixel must be recorded. Figure 4.6 shows the run-length data structure of this run-length encoding. Each run in the image is encoded by its starting and ending pixel locations. The location of the starting pixel is (ROW, START_COL) and the location of the ending pixel is (ROW, END_COL). The label of the connected component to which this run belongs will be stored in PERM_LABEL. This is initialized to zero and assigned temporary values in pass 1 of the algorithm. At the end of pass 2, the PERM_LABEL field contains the final and permanent label of the run. This structure can then be used to output the labels back to the corresponding pixels of the output image.

Consider a run P of 1-pixel. During pass 1, when the run has not yet been fully processed, PERM_LABEL (P) will be zero. After run P has been processed and determined to be adjacent to some other run Q on the previous row, it will be assigned to the current label of Q, PERM_LABEL (Q). If it is determined to be adjacent to other runs $(Q_1, Q_2, \ldots, Q_k)$ also on the previous row, then the equivalence of PERM_LABEL(Q), PERM_LABEL(Q_1), PERM_LABEL(Q_2), …, PERM_LABEL(Q_k) must also be recorded. The data structures used for recording the equivalences are shown in Figure 4.7.


(a)

1	1		1	1
1	1			1
1	1	1		1
	1	1	1	1

(b)

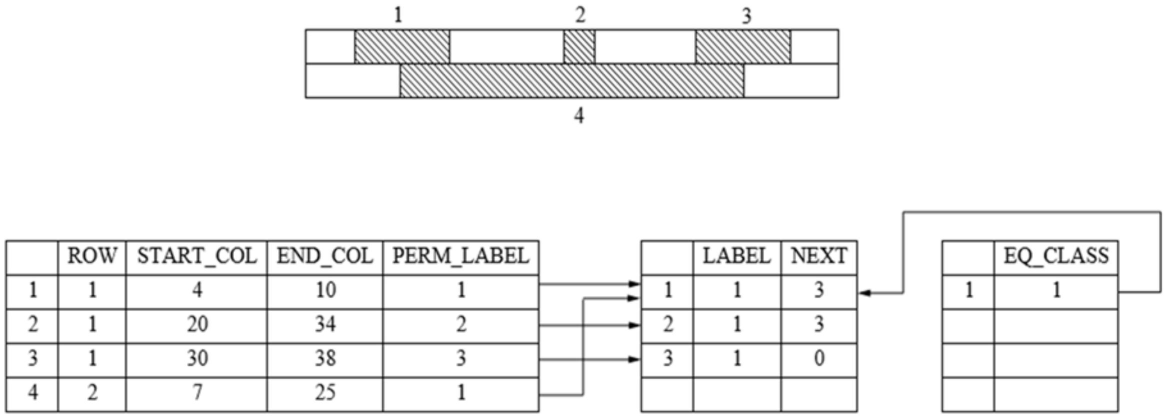
	ROW_START	ROW_END
1	1	2
2	3	4
3	5	6
4	0	0
5	7	7

(c)

	ROW	START_COL	END_COL	PERM_LABEL
1	1	1	2	0
2	1	4	5	0
3	2	1	2	0
4	2	5	5	0
5	3	1	3	0
6	3	5	5	0
7	5	2	5	0

Figure 4.6: The process of making run-length table. (a) Binary image (b) and (c) Run-length table

For a given run P, PERM_LABEL(P) may be zero or nonzero; If it is nonzero, then LABEL(PERM_LABEL(P)) may be zero or nonzero, and if it is zero, then PERM_LABEL(P) is the current label of the run and there is no equivalence class. If it is nonzero, then there is an equivalence class and the value of LABEL(PERM_LABEL(P)) is the label assigned to that class. All the labels that have been merged to form this class will have the same class label; that is, if run P and run P' are in the same class, LABEL(PERM_LABEL(P)) = LABEL(PERM_LABEL(P')). When such an equivalence is determined, if each run was already a member of a class and the two classes were different, then the two classes are merged. This is accomplished by linking together each prior label belonging to a single class in a linked list pointed to by EQ_CLASS(L) for class label L and linked together using the NEXT field of the LABEL/NEXT structure. To merge two classes, the last cell of one is made to point to the first cell of the second and the LABEL field of each cell of the second is changed to reflect the new label of the class [Haralick and Shapiro 1992].



	ROW	START_COL	END_COL	PERM_LABEL
1	1	4	10	1
2	1	20	34	2
3	1	30	38	3
4	2	7	25	1

	LABEL	NEXT
1	1	3
2	1	3
3	1	0

	EQ_CLASS
1	1

Figure 4.7: Data structure used for keeping track of equivalence classes

This algorithm runs through the image one time to make the run-length table. In the second step, this table is used to find the connected segments and label them the same. Since the run-length table is much smaller than an image, and also this algorithm only runs through the whole image once, this process is much faster than the methods explained in Chapter 3. After running this algorithm, each digit in the timestamp of the image is labeled differently. To complete the recognition of time and date, each segment should be recognized as a number. Prior to the recognition step, the segments with a smaller size than a minimum threshold are removed. This is done by keeping in mind that the number of pixels in a segment that is representing a digit in the timestamp should be greater than a minimum value, otherwise that segment is a result of the noise in the image. As shown in Figure 4.8, Number 1 has the least number of white pixels, so the minimum value for a segment is 0.7 percent of the number of pixels representing Number 1. This process also removes the colons between hour, minute, and second in the timestamp.

4.5 Recognition of Digits

The last step in extracting time and date is the recognition part. In this step, each segment from the last step is recognized as a digit from 0 to 9. All the segments from the last step should be recognized as a digit, since the colons in the time part of the timestamp is removed in the last step. Because the size of the timestamp is small and the color is distinguishable from other parts of the image, a simple character recognition algorithm is used to detect the digits.

The algorithm designed and implemented in this research for recognizing the digits is simple and fast. Since the font of the digits in the timestamp is the same, first each white segment resulted from the last step is placed on a black background with a predefined size which is called badges on this thesis. The unknown badge can be recognized as either a specified number or an unknown number between 0 and 9. The algorithm compares each unknown badge with every number in the pattern. The patterns for the numbers are made in

Photoshop with the same predefined size as the badges, and each badge is compared to all of the patterns, as shown in Figure 4.8.

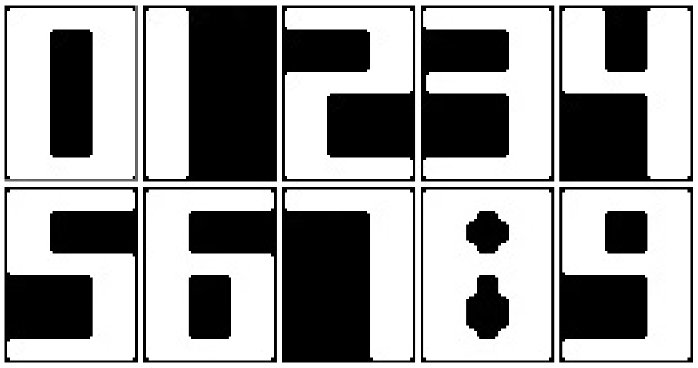


Figure 4.8: All the numbers with the same pattern as the timestamp

The similarity is simply measured from the number of white pixels in each number and the unknown segment. Also, some of the unique features of each number is considered in the similarity analysis process. For example, Number 1 has no white pixel on the right side of its pattern; in this case, if the program finds white pixels on the right side of the separated segment (unknown badge) and the number of those white pixels happens to be more than a threshold, the unknown badge is not recognized as Number 1, even if the badge has the highest similarity with Number 1 as compared to the other numbers. Furthermore, if the similarity between an unknown badge and all the patterns is lower than a threshold, then the number is recognized as unknown.

In some cases, due to the noise introduced in the videos, the algorithm might not be able to recognize the numbers, correctly. For example, Number 5 might be detected as Number 6 or the number of segments resulted from the last step could be less than 14, which is the number of digits in the timestamp. To reduce these errors the script that stores the data in the database compares the date and time with the last image stored in the database. To find out the last image stored in the database, an auto increment number (id) is used to store the data, so the query looks for the tuple in the database with the highest id. Although date and time could be used to find the latest image, it would be more time consuming to find the latest date and latest time in a long query. After finding the latest data, the script compares the date and time to check whether the difference is negative, the difference is more than one year, or the date and time are not completely detected. In these cases, the algorithm uses the time and date of the latest image, and adds 10 seconds to it, then stores those values as the time and date for the new image in the database. The process of extracting time and date from images and storing them in the database is shown in Figure 4.9.

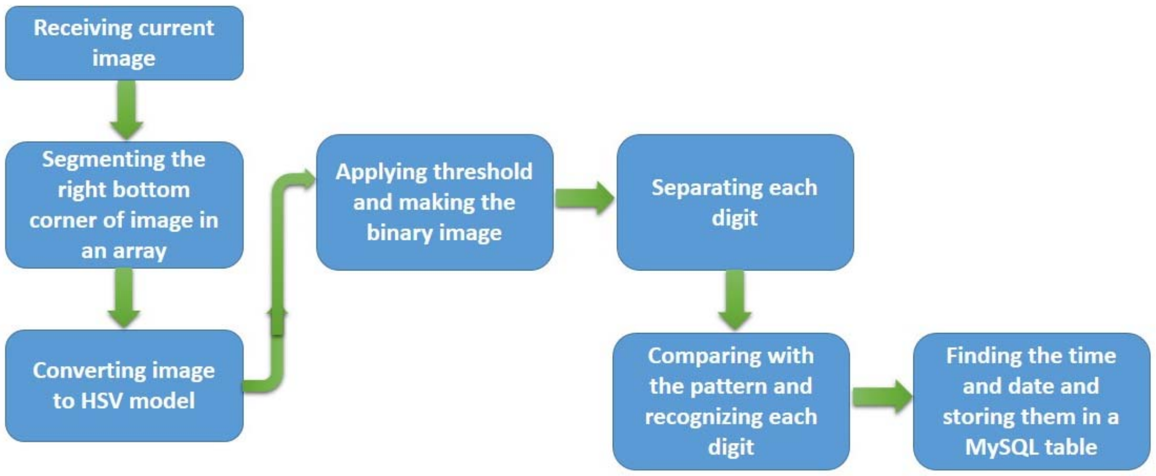

Receiving current image
Segmenting the right bottom corner of image in an array
Converting image to HSV model
Applying threshold and making the binary image
Separating each digit
Comparing with the pattern and recognizing each digit
Finding the time and date and storing them in a MySQL table

Figure 4.9: The flowchart of the process of extracting time from images

CHAPTER 5 IDENTIFYING AND COUNTING BEES

5.1 Introduction

To study the activities of honey bees, the best way is to monitor them over a long period of time. This task can be done using image processing techniques. Using image processing, bees are detected in each image and their number in front of the beehive is estimated. The number of bees in front of the beehive can be a good indication of the beehive's health status.

This chapter explains the process of identification of bees in an image, as shown in Figure 5.1. This process is done in parallel with time and date extraction. The results of both processes are stored in a MySQL database. These data can be stored for a long period of time and then can be used to find the relationship between the activities of bees and changes in the local temperature and humidity.

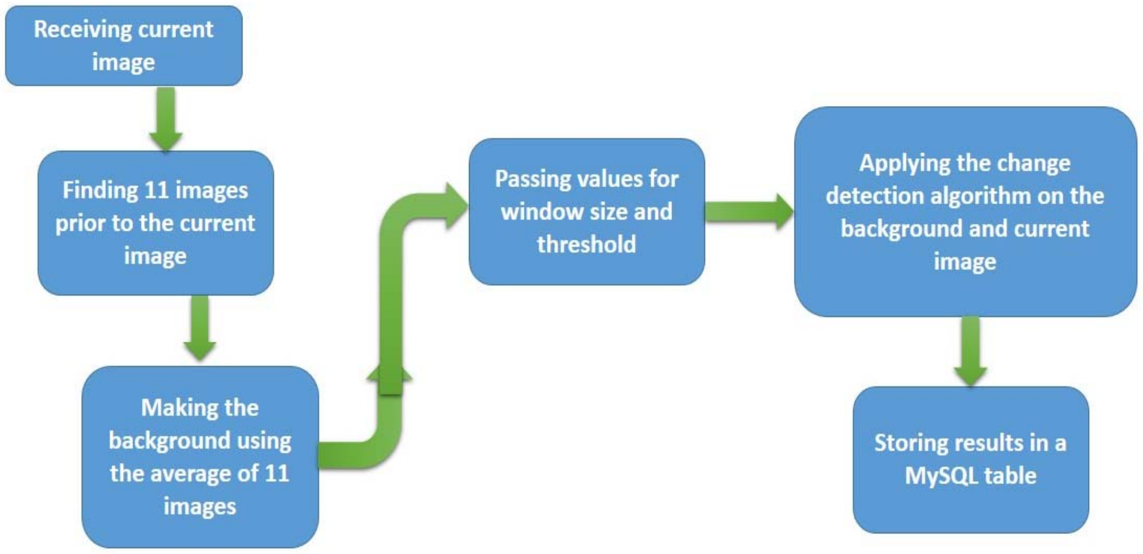

Receiving current image
Finding 11 images prior to the current image
Making the background using the average of 11 images
Passing values for window size and threshold
Applying the change detection algorithm on the background and current image
Storing results in a MySQL table

Figure 5.1: The process of finding the number of bees in an image

5.2 Detecting Bees

To record the videos from a beehive, surveillance cameras are used in this research. These cameras are easy to set up, provide wireless communication, and can work in various weather conditions. Despite these advantages, they do not provide high quality videos. The highest quality provided by these cameras is 640 × 480 pixels.

To identify and detect bees in an image, a variety of methods can be used mainly based on the bees' features. The size of bees, their color, and also the structure of their body can be used as distinguishable features. In this research, due to the quality of videos, identification of bees using their features and finding their direction in an image were not feasible. Also, due to the same problem and the distance between the camera and the hive, it is extremely hard for human eyes to detect the bees in an image as shown in Figure 5.2. It is worth nothing that the cameras used in this research do not provide true color data. The color of bees in the image is almost black, with the yellow color of bees not clearly visible as shown in Figure 5.2. Due to this fact, the color data cannot be used for detecting the bees.

To study bees better, their movement and their flying pattern in front of the beehive can be monitored. By tracking the bees' movements, their speed in front of the beehive can be estimated. To track the bees, there is a need for fast cameras with sufficient frame-per-second rate to capture the movements closely. The cameras used in this research have an average frame rate of 16 fps. The average speed of honey bees in front of a beehive is 12 mph [Dolan 2000] which means a bee flies 5.5 meters every second, therefore, in two consecutive frames of a video that is recorded with a 16 fps camera, a bee can fly almost the whole length of an image (50 cm). Consequently, it is not feasible to track a bee in the videos. This limitation prompted the use of another method to identify and detect bees in an image.

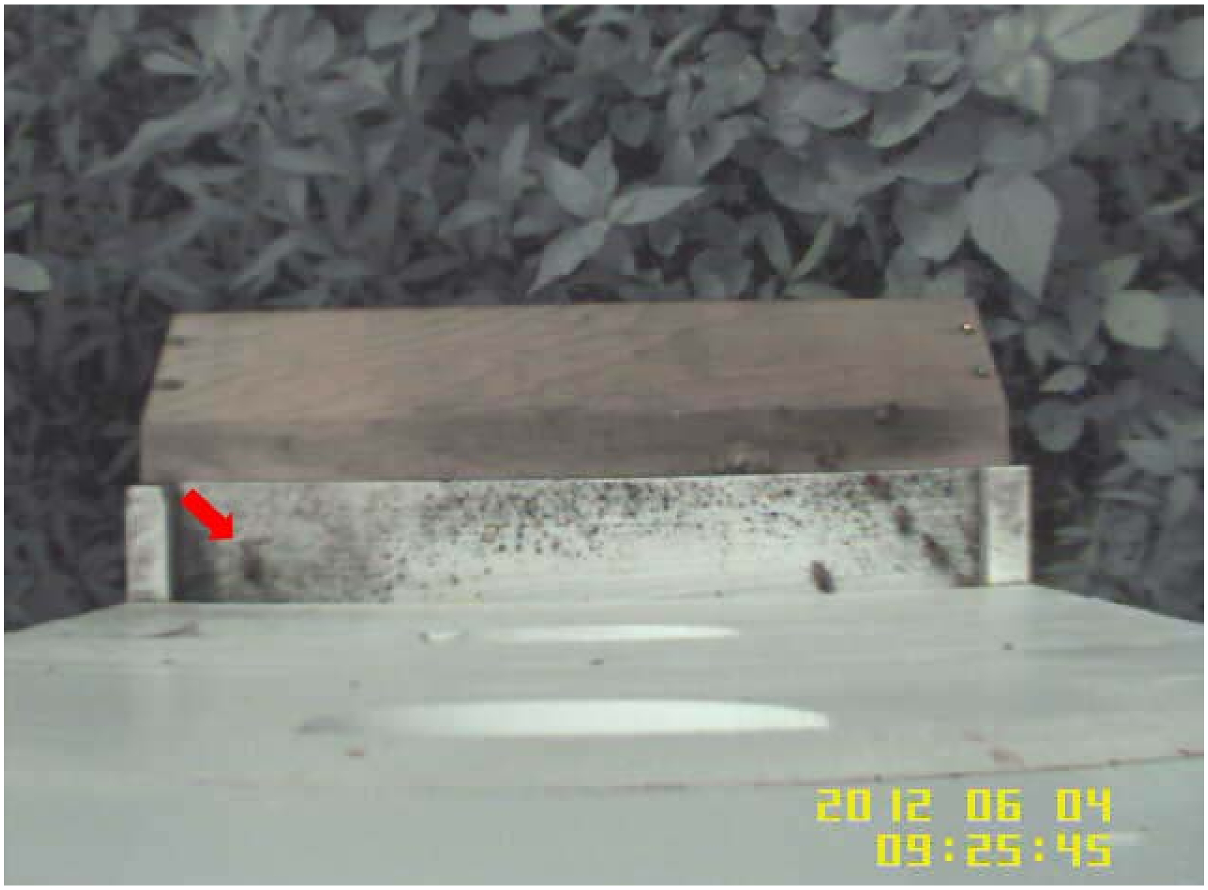


Figure 5.2: Honey bees’ movement in front of the entrance of beehive

Since the bees are the only moving objects in front of the beehive, the fastest and easiest method for finding the bees in an image is the change detection method. A change detection algorithm, as the name implies, compares two images and returns the differences between them as a binary mask, showing the changes as 1’s and the unchanged parts as background. This mask is called change mask, and this method can be used to observe movement of objects or humans in a surveillance video.

To use a change detection method to detect bees, the first step is to find the background image. This background should not have any changing part in it, i.e., it should not contain any bees. Finding a static background that can be used as a reference for every image is not feasible. Changes in illumination and weather conditions make it impossible to have a constant background for reference. For example, if a background is made from the images taken in the morning and it is used as a reference for an image taken in the afternoon, the algorithm might detect all the illumination changes as moving objects, and making it harder to detect the actual moving objects. Also, weather changes can lead to error in the results of the algorithm. Rain drops can be detected as moving objects when using a background reference that is made from the images taken in a sunny day.

A simple solution to the background problem is to compute the background for every image. To find a background for each image that has a similar illumination to the current image and is taken in the same weather conditions, the background can be prepared by computing the average of 11 recent images taken from the beehive prior to the current image. This task makes the whole process more time consuming, but it makes the process of change detection more accurate and reliable as shown in Figure 5.3.

The next step in detecting the bees in an image is to find the differences between each image and the background. To accomplish this, different methods of change detection can be used to find the change mask. Since the cameras are placed outdoors and all the recordings are done outside, changes in illumination are the main problem to be eliminated in the final mask. The movement of cameras and the leaves of the plants close to the hive can also cause some changes in the scene. Different change detection methods are applied and their results are compared to each other and the method with the best results is selected for detecting changes in this research.

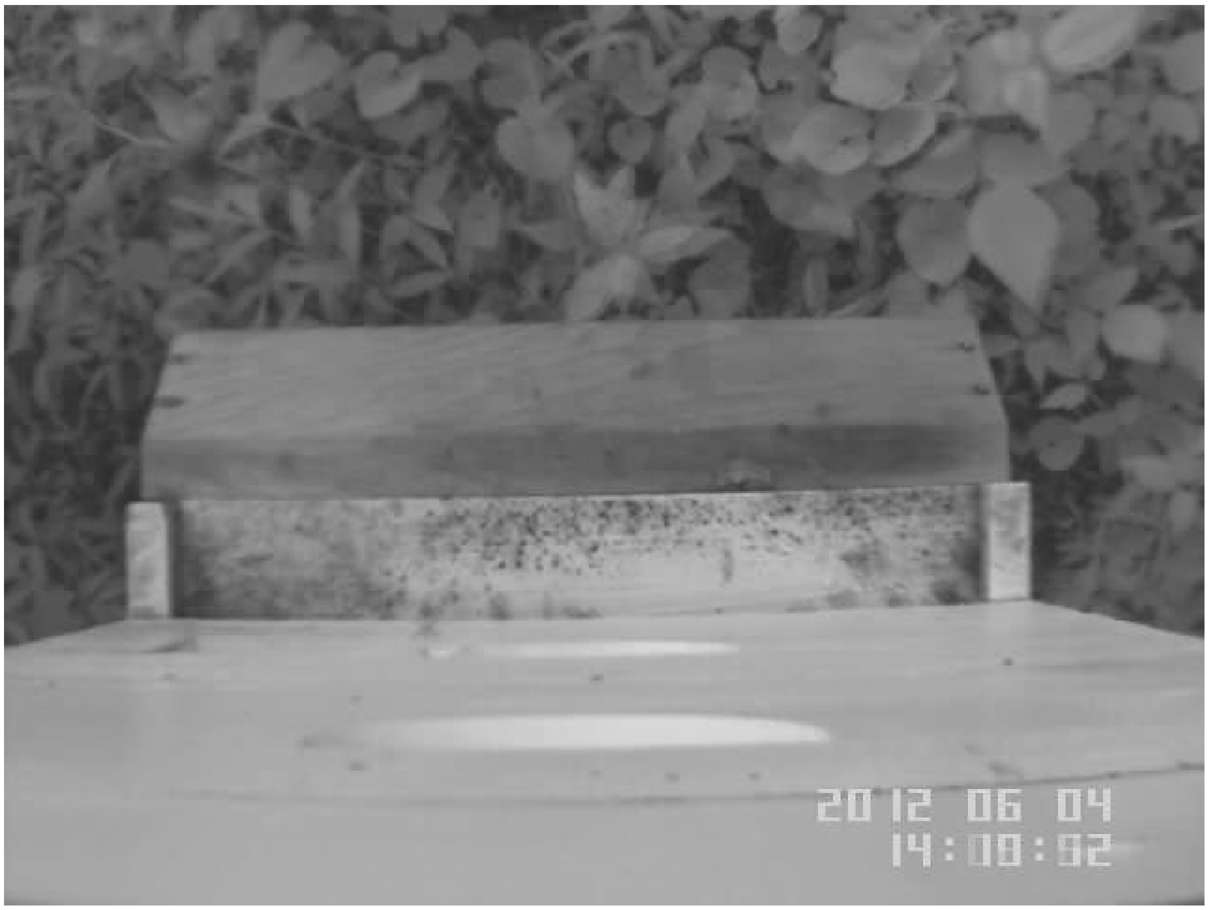


Figure 5.3: Background image made by averaging

The change detection method that produced the best result for use in this research is robust and illumination invariant change detection and is based on linear dependence [Durucan and Ebrahimi 2000]. The advantage of this method is that it has all the illumination invariant part embedded in the processing over most other methods, hence it gives better results for the purposes of this research.

5.2.1 Illumination Invariant Change Detection

For the sake of simplicity, in this research the initial state of the image or background is called the reference image and the final state or the image for detecting bees is called the current image, shown as I_r and I_c, respectively. To find the differences between the reference image I_r and the current image I_c, the easiest way is to subtract one image from another and find the pixels with value greater than a specified threshold. Although this method works for most cases, due to illumination changes in the images used in this research, many false changes are detected. A dynamic background eliminates many of these changes.

To make sure that illumination will not introduce more changes, the illumination invariant method can be used to make the change mask. This method uses the concept of linear dependency to find the differences between two images. Linear dependency states that a finite subset $\{\vec{a}_1, \dots, \vec{a}_n\}$ of a vector space V is called linearly dependent if the zero vector $\vec{0}$ can be written as a nun-null linear combination of those vectors, as stated in 5.1.

$$\vec{0} = k_1\vec{a}_1 + \cdots + k_n\vec{a}_n \; with \; \textstyle\sum_{i=1}^{n}|k_i| \neq 0. \tag{5.1}$$

If there are only two vectors (n=2) shown as with $\vec{a}_r$ and $\vec{a}_c$ and $k_i \in R$, 5.1 can be rewritten using $\vec{0} = \vec{a}_r - k_i.\vec{a}_c$, as 5.2.

$$\frac{a_{r_1}}{a_{c_1}} = \frac{a_{r_2}}{a_{c_2}} = \frac{a_{r_3}}{a_{c_3}} = k, \tag{5.2}$$

where a_{r_i} and a_{c_i}, with $a_{c_i} \neq 0 \; \forall i$, denote the components of each vector. Since the two vectors are parallel and all the ratios are the same, their variance should be zero, as shown in 5.3.

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{a_{r_i}}{a_{c_i}} - \mu\right)^2, \; n = \dim(V). \tag{5.3}$$

The mean is calculated using 5.4.

$$\mu = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{a_{r_i}}{a_{c_i}}\right). \tag{5.4}$$

The illumination invariant change detection method models the images as vectors. A vector can be modeled as the values of pixels in a small window rolling over the target images and after finding the two vectors, 5.3 can be applied to every pixel in the window. After calculating σ^2 for each pixel over a window with size $\mathrm{n} \times \mathrm{n}$ for each vector, two scenarios can happen: if the value of σ^2 is greater than a threshold, the pixel is accounted as a change in the image; otherwise, it is detected as a background pixel. To summarize, an invariance operator σ^2 that is based on the linear dependence definition is derived. This operator is invariant to linear transformation, which is mostly caused by illumination changes. Thus, by using this operator as a factor for detecting changes, this method is independent of illumination changes.

5.2.2 Implementation of Illumination Invariant Change Detection

To implement the illumination invariant method, the first step is to convert the image to a gray-scale. This is done because the algorithm needs to find the changes in the image, therefore, the gray-scale format has the advantage over the others. For example, the RGB format has the color information, and to detect moving objects, there is no need to measure the changes in the color content since the color of an object would change as illumination changes. After converting an image to its gray-scale, the algorithm can be implemented to process it.

To implement the algorithm, the size of the window and the threshold for invariance should be optimized. To do this, these parameters are modeled as inputs for a function that implements the algorithm. This way the arguments can be changed and the values with the best results can be selected. The code on the server is written in C and the algorithm is implemented in a MATLAB function. Thus, the MATLAB program can be called with all the arguments from the C program. The arguments for the function include the window size to make the vector V, a threshold factor to be compared with σ^2, and the value of μ for each window.

The first two values are experimental; hence, the best values for them can be found by trying different values and comparing the results with each other. The value of μ is not constant for the whole image, since it changes for each window, i.e., it must be calculated for each window (vector) using 5.4. The implemented program goes through an image and calculates the value of μ for each window, comparing each pixel in the reference image with the same pixel in the current image, and computes σ^2 using 5.3. Each pixel with a σ^2 greater than the threshold value is marked as a change and is assigned with the value of 255 (white color), otherwise, is assigned with 0 (black color).

The result of this step is a change mask which might contain some of the changes that are caused by the movement of anything in the scene other than the bees. To make the change detection more accurate, only the changes that are detected in front of the hive are taken into account. It should be noted that the algorithm can miss some of the changes that are actually caused by the bees as they are flying around the hive outside the camera's field of view. The illumination invariant method makes the results more consistent and also eliminates most of the detected changes that are unwanted, such as the movements of leaves.

The final step in change detection and finding bees is to refine the change mask and calculate the size of each segment in the mask. To calculate the size of the segments, the algorithm uses the same method used in Chapter 4 for labeling the digits in the timestamp to count the number of pixels in each label. If the size of a segment is less than a threshold value, which is equal to the average size of bee in the images used in this research, the change is eliminated; otherwise, it is detected as a bee and taken into account for the change mask. This is done because there is a possibility that the detected change is not caused by the movements of a bee. The result of this step is the final change mask that can be used to count the number of bees in the image.

5.3 Counting Honey Bees

The final step in the image processing part is to count the number of bees in the change mask. If the camera used for such a research could provide true color information, this step could use that data to produce more accurate results.

To find the number of bees in the image, one method is to count the number of segments in the change mask that is obtained from the change detection algorithm. Since bees are moving close together in front of the hive, it is possible that some of them are detected as one segment in the change mask, leading to inaccurate results. Counting that segment as one bee makes the results inaccurate. In such a case, there will be some differences between the actual number of bees in the image and the number of segments in the change mask as shown in Figure 5.4.

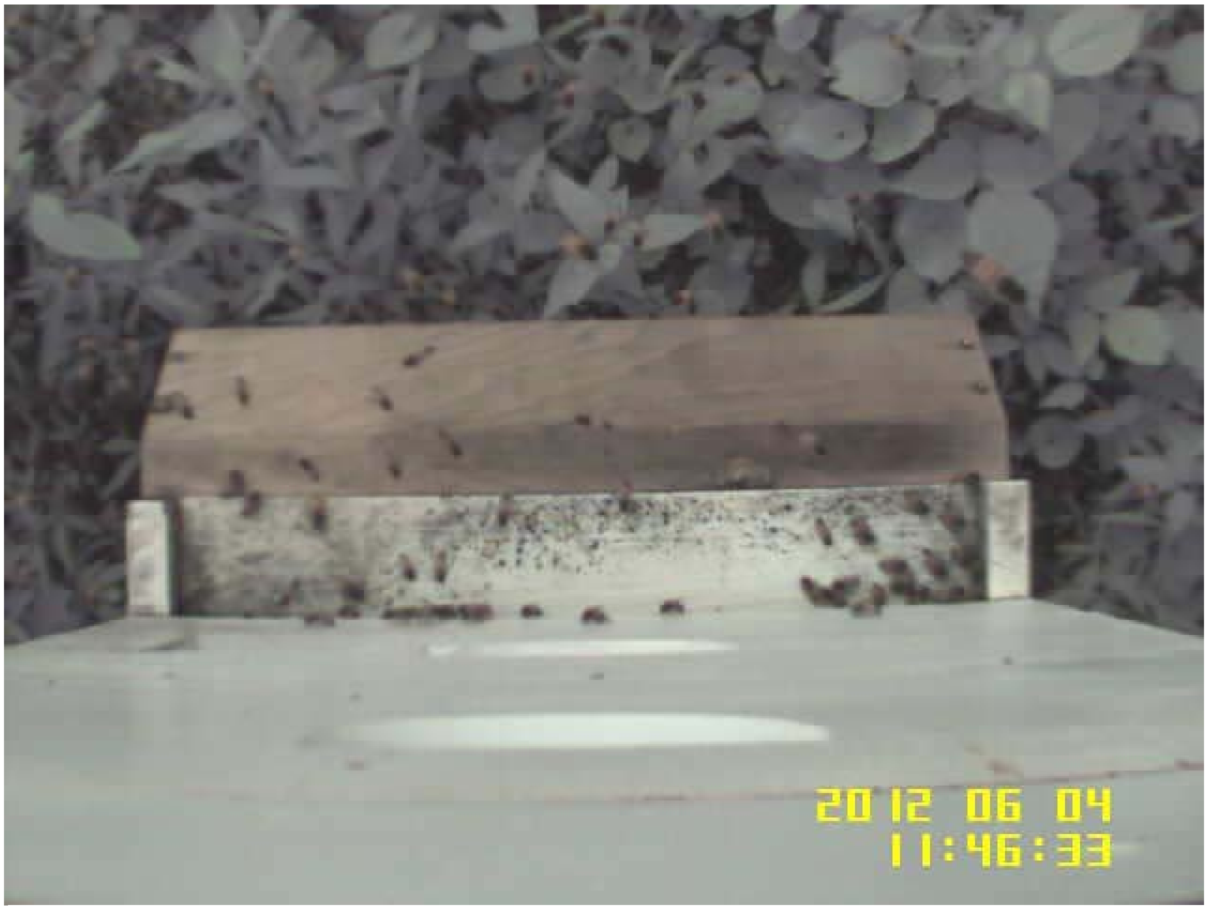


Figure 5.4: A crowded top view of the beehive

To address the above problem when two or more bees are detected as one segment, the size of that segment in pixels is larger than the average size of a bee in the image. Therefore, the size of each segment is compared with the average size of a bee, which is obtained using several images. After finding all the segments, the size of each segment is calculated and then is divided by the average size of a bee. The result is used as the number of the bees detected in that segment. The resulting numbers are added together to estimate the total number of bees in the image.

CHAPTER 6 THE IMAGE ACQUISITION AND DATA MANAGEMENT SYSTEM

6.1 Introduction

An automated system is designed and implemented in this thesis to capture the data from a beehive and stores the data in a MySQL database. In the preceding chapters, the image processing step of this system was explained. This automated system consists of four main parts. All the image processing techniques, explained in Chapter 4 and Chapter 5, are implemented in the image processing part of the system. Other than image processing part, the system includes three more sections.

Receiving data from the cameras, converting them into image frames, storing them on the server, passing them to the image processing part for processing, and storing the results and the images in the database are completed in one part of the system. The second part of the system fetches the temperature and humidity data from the National Weather Service website [NOAA 2013] and stores them in the database. The third part of the system consists of an iPhone application and the backend programs. This part makes it possible for the users of the system to view the stored data on their iOS-based devices.

6.2 Video Recording

To capture the data from a beehive, a 4-channel wireless quad surveillance system with digital video recorder and an indoor/outdoor motion camera are used. This camera provides the videos with two different qualities (the highest quality is 640×480 pixels) and can send data wirelessly to its receiver up to a maximum distance of 150 feet. Sending data wirelessly makes the process easier for two reasons: first, there is no need to wire the camera to the receiver and the processor, and second, the camera can be installed anywhere on or near the hive. For the purpose of this research, cameras are either installed in front of the hive or at the top as shown in Figure 6.1.

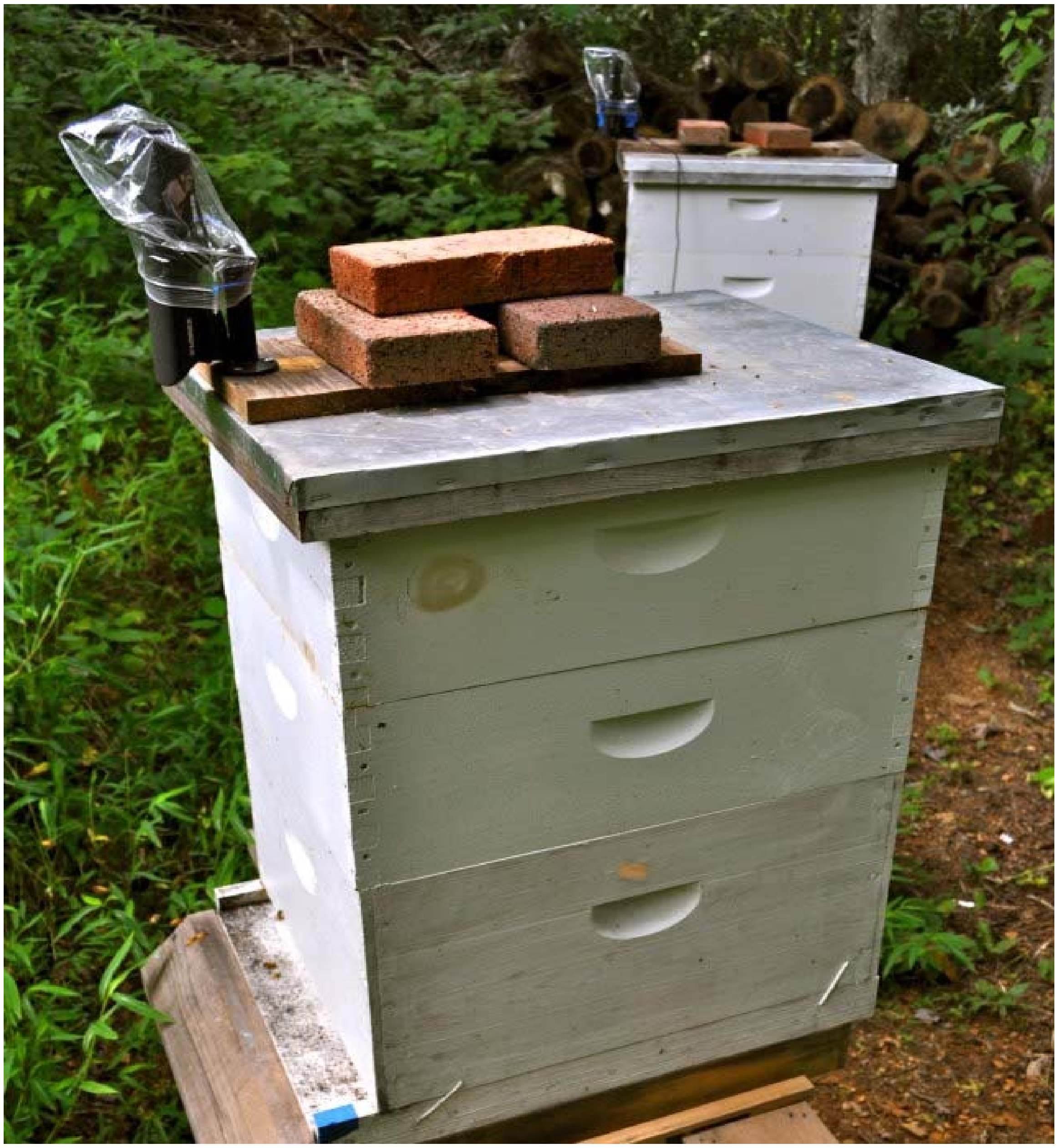


Figure 6.1: Camera installed on top of the beehive

Although there is no need to wire the cameras to a network, they still need to be powered. There are two options for providing power to the system: first option is to use chargeable batteries as a power source at the site, and the second option is to wire the cameras to a power outlet. The first option is portable, but harder to maintain as the batteries need to be recharged regularly. In this research, since the portability of the system is not much of a concern, the second option is used. The two cameras are connected to the power outlet to record the activities of the bees without any interruption.

The qualities provided by the cameras are 640 × 480 and 320 × 240 pixels. Each of these qualities have its own advantages for use in this research. Processing low quality videos is four times faster than that of the high quality, but these videos do not provide significant details and may not contain important features. Since the details are important for the purpose of this research, the cameras are set to record the videos with the highest quality.

6.3 Extracting Image Frames from Videos

The camera used in this research can send data to the wireless receiver that comes with it. The data from the receiver can be stored on a SD card, or can be sent to a monitor or a TV to be viewed using a component video cable. The videos then can be copied from the SD card onto the server for processing. This requires a person to check the SD card for whether it is full or not, and copy the videos onto the server. The data are stored on the SD card as videos and takes significant disk space. For example, a 16GB SD card, which is the maximum capacity that the receiver can support, is filled by the highest quality videos recorded from a beehive within 72 hours. Furthermore, the image processing part is designed to process images, so the videos are not ready for immediate use, hence these videos must be broken into their frame images. For the purpose of this research, a method that will be explained in the next section is used to extract the image frames from the videos as recording takes place. Then the resulting images are stored on the server for further processing.

6.4 Storing Individual Images

The receiver that comes with the cameras provides an AV connection that can be used to connect the receiver to a TV or a monitor for the videos to be viewed live. This connection can be used to store the image frames directly to the server using a digital video converter. The converter receives the video stream from an analog or digital device and sends it to a computer. To use this converter, its driver must be installed on the server or the computer that is receiving the data. This allows the data to be received and be stored directly on the server. Since the videos are received by the computer, they can be processed in real time. A program can process the videos and save them as images as the videos are coming in.

The raw data that are received from the digital video gets converted to individual images at a desired time interval between two consecutive images. For the purpose of this research, the images are saved for processing every 10 seconds. This requires a disk storage that is 160 times smaller than that needed by video recordings. To accomplish this, a script using MJPG-streamer is implemented.

The MJPG-streamer is a command line application that copies JPG-frame from a single input plugin to multiple output plugins. It can be used to stream JPEG files over an IP-based network from the webcam to a viewer like Firefox or even to a Windows Mobile device running the core pocket media player (TCPMP). The application is written for embedded devices with very limited RAM and CPU capabilities and can be used on a Unix-based system [Google Inc. 2011].

In our research, the MJPG-streamer is installed on a Linux machine that is connected to the converter. A Bash script is written that uses the MJPG-streamer to manage the files. This script creates a new folder with a name that contains the date information. Also, the script kills all the running instances of the MJPG-streamer and runs a new instance of MJPG-streamer to capture the images from the input device (converter) every 10 seconds and to store them in the new folder. For data to be more organized and easily accessible later, this script is automatically run by the system at midnight every day; hence, the folder name contains the date when the recording is made. This way, all the images taken in the same day are stored in one folder to make their processing and accessing easier. The folder for each day contains almost 9000 images.

To run the script every night a Cron job is used. Cron is the time-based job scheduler in Unix-like operating systems. Cron enables users to schedule jobs (commands or shell scripts) to run periodically at certain times or dates. It is commonly used to automate system maintenance or administration, though its general-purpose nature means that it can be used for other purposes such as connecting to the Internet and downloading emails [Cognition 1999]. For the purpose of this research, a Cron job is written to run the image extracting script every night at 12:00 am. The stored images are then processed on the server.

6.5 Image Processing on the Server

After storing the data on the server, they need to be processed, organized, and stored in a database. All the image processing modules explained in Chapter 4 and Chapter 5 are written using MATLAB. MATLAB provides almost all the required functions for working with images. Since the goal is to make the system autonomous, the image processing part needs to run on the server. To run MATLAB on the server for executing the image processing scripts, the environment must change to the MATLAB environment, which is not possible since the images should be processed automatically and changing environment on the server cannot be done by a program.

To run the image processing algorithms on the server, MATLAB engine library is used. The MATLAB engine library contains routines that allow users to call the MATLAB software from another program, thereby employing MATLAB as a computing engine. To use this library MATLAB needs to be installed on the machine. Engine programs are standalone C/C++ or FORTRAN programs that communicate with a separate MATLAB process via pipes on UNIX systems and through a Microsoft Component Object Model (COM) interface on Microsoft Windows systems. MATLAB provides a library of functions that allows users to start and end the MATLAB process, send data to and from MATLAB, and send commands to be processed in MATLAB [MathWorks Inc. 2010].

Using MATLAB engine library, a program is written in C. This program calls the engine, opens it, and executes the functions that are passed to it. Using this method, the MATLAB script cannot be interactive, so all the scripts written in MATLAB must be changed to MATLAB functions. Therefore, the input image and other parameters can be passed to them as input arguments.

Following this step, the C program is then placed on the server. This program can be called by any other program to run the scripts and return the results. Since the goal of this research is to store the data in a database and be able to access the data on the web, a program is written in PHP to handle this task. This program can interact with users via the web, call the C program, and interact with it. This way, the images can be processed from every device using the web interface and the results can be sent back to the program and shown to the user.

To automate the procedure for processing the images and storing them in a database, a script is written in Bash that calls the C program for every image. This program calls the C program and passes the images. After processing the images, the C program returns the results to the script, which stores the processed data in the database. The image cannot be stored in a MySQL database directly, instead its path on the server is stored in the database. The data that are stored in the database includes the path to the image on the server, estimated number of bees in an image, the SNR and entropy of the image, and also the time and date that the image was taken. The time and date were extracted from individual images using the image processing techniques that was discussed in Chapter 5.

The Image acquisition and image processing parts work together to extract and store the data from the hive images. Figure 6.2 provides a summary of all the steps taken to acquire an image and process it.

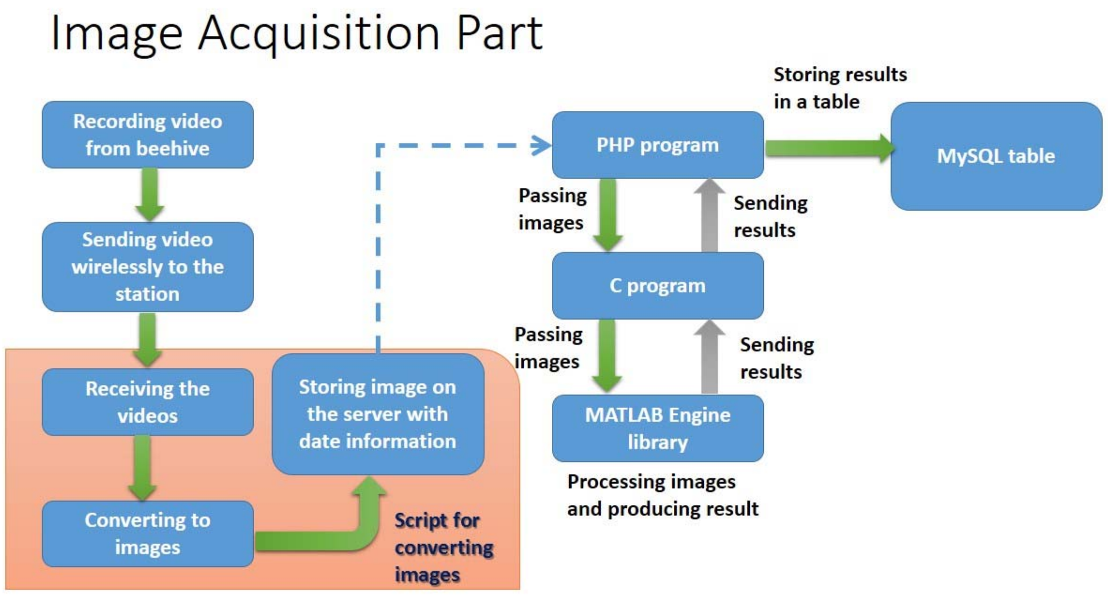


Image Acquisition Part
Recording video from beehive
Sending video wirelessly to the station
Receiving the videos
Converting to images
Script for converting images
Storing image on the server with date information
PHP program
Storing results in a table
MySQL table
Passing images
Sending results
C program
Passing images
Sending results
MATLAB Engine library
Processing images and producing result

Figure 6.2: The flowchart of image acquisition and image processing part

6.6 Weather Data Gathering

To observe the activities of honey bees, the number of bees in an images should be estimated at different temperature and humidity values. The local weather data must to be collected from a reliable source on a daily basis. In this research, the data are gathered from the National Weather Service (NWS) website [NOAA 2013]. This website provides a data feed of its data, therefore, the data can be fetched using a PHP program.

The data from the National Weather Service are in extensible markup language (XML) format. Extensible markup language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable [Bray et al. 2013]. It is a textual data format with strong support via Unicode for the languages of the world. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example, in web services.

The PHP program reads the data formatted in XML from the NWS website. The PHP script connects to the website, parses the XML data from the website, and stores the results in a MySQL database table. To be able to easily retrieve data later, the program reads the date and time from the server and stores them for every image in the table. This program only reads the data once and stores them in the database. For the purpose of this research, humidity and temperature need to be read continuously from the NWS web page. Since the temperature and humidity data do not change dramatically within one hour, the PHP program is set to run every hour to fetch these data items. To run the program every hour, a Cron job is created and placed on the server. As a result, the database table contains climate data for every hour.

This table can be used for finding the relationship between the number of bees around the beehive and the climate data. The climate data are stored in a table for every hour, but the image data are acquired every 10 seconds. In cases where the data is not available at the desired time an interpolation schema is used. To find the weather W_T (it can be temperature or humidity) for a specific time T that is between specific hours h_1 and h_2 with weather values W_{h_1} and W_{h_2}, respectively, the interpolation schema in (6.1) is used.

$$W_T = W_{h_1} + \frac{T - h_1(in\ seconds)}{3600} \times (W_{h_2} - W_{h_1}). \tag{6.1}$$

Using 6.1, the temperature and humidity for any given time can be estimated from the existing weather data in the database table. The formula in 6.1 assumes that temperature and humidity change linearly with the time within one hour. This assumption would make the value resulted from 6.1 slightly different from the actual value for a specific time.

6.7 Storing the Weather and Image Data in the Database

All the data extracted from the image processing part and the data retrieved from the NWS website need to be organized and stored on the server. The data resulting from the image processing include a beehive image, date and time that the image is taken, and other data that are computed for the image, such as SNR and entropy. To store and provide easy access to data, a MySQL relational database management system is used, in this research. The system runs as a server providing multi-user access to a number of databases [Oracle Corporation 2013].

A MySQL table is used to store the image data from individual images. This table needs a primary key that is a combination of one or more attributes in the database and must be unique for every tuple in a table. For the purpose of this research, an auto increment attribute (id) is used for all the tables as the primary key. The data from a MySQL table can be accessed by queries. The most common query for this table is retrieving an image and its corresponding data at the specific date and time. The schema of the table used for image processing data is shown in Table 6.1.

Table 6.1: Schema of MySQL table used for storing humidity and temperature data

Id	Date	Time	Temperature	Humidity

For weather data that are gathered from the NWS website, another table is used. The data from image processing and weather data cannot be stored in one table, as the weather data are read from the NWS website every hour and the data from image processing unit are processed for each image every 10 seconds. As a result, another table is created for the weather data. Similar to Table 6.1, an incremental id is used as the primary key for this table. This table stores the time and date, which are read from the server and the temperature and humidity for that specific date and time. The schema of the table used for storing the climate data is shown in Table 6.2.

Table 6.2: Schema of MySQL table used for storing images and image processing results

Id	Date	Time	Image URL	SNR	Entropy	Estimated No. of bees

6.8 The iPhone Application

One of the goals of this project is to make a tool for the beekeepers to easily keep track of the activities of bees. For the purpose of this research, an application is designed and implemented for iOS-based devices. This application must be able to connect to the database and access the data. It also must be able to display the data to the user in a user-friendly way, hence making it easier for the beekeepers to view the beehive data at any time and from anywhere.

There are three main operating systems already running on most of the smart phones. Android, iOS, and RIM Blackberry for various mobile devices, such as Samsung [Velazco 2011], Apple, and Blackberry mobile devices. Since iPhone is one of the most commonly used smart phones in the world [Telegraph Media Group 2013], the application is written for this device. Also, writing an application for iPhone is easier, since there are fewer devices on which the application deploy.

To develop applications for iPhone, the only integrated development environment (IDE) that can be used is Xcode. Xcode is an IDE containing a suite of software development tools developed by Apple for developing software for OS X and iOS. The latest version is available for Mac OS X Lion and OS X Mountain Lion users [Apple Inc. 2013].

Objective-C is used as the programming language to develop the application for iPhone. Objective-C is a general-purpose, high-level, object-oriented programming language that adds Smalltalk-style messaging to the C programming language. It is the main programming language used by Apple for the OS X and iOS operating systems and their respective APIs, Cocoa, and Cocoa Touch [University of Texas at Arlington 2013].

The iPhone application provides a simple-to-use and user-friendly interface for the users to access the data on their iPhones. To connect to the database, the application receives the time and date as inputs from the user. The interface passes those parameters to a PHP program using GET method. The PHP program uses the set of arguments and makes a query from the database table. After getting data from the MySQL database, the PHP program returns the values to the application in JavaScript Object Notation (JSON) format. JSON is a text-based open standard designed for human-readable data interchange. After receiving the data from the database, the application displays the data on the screen, as shown in Figure 6.3.

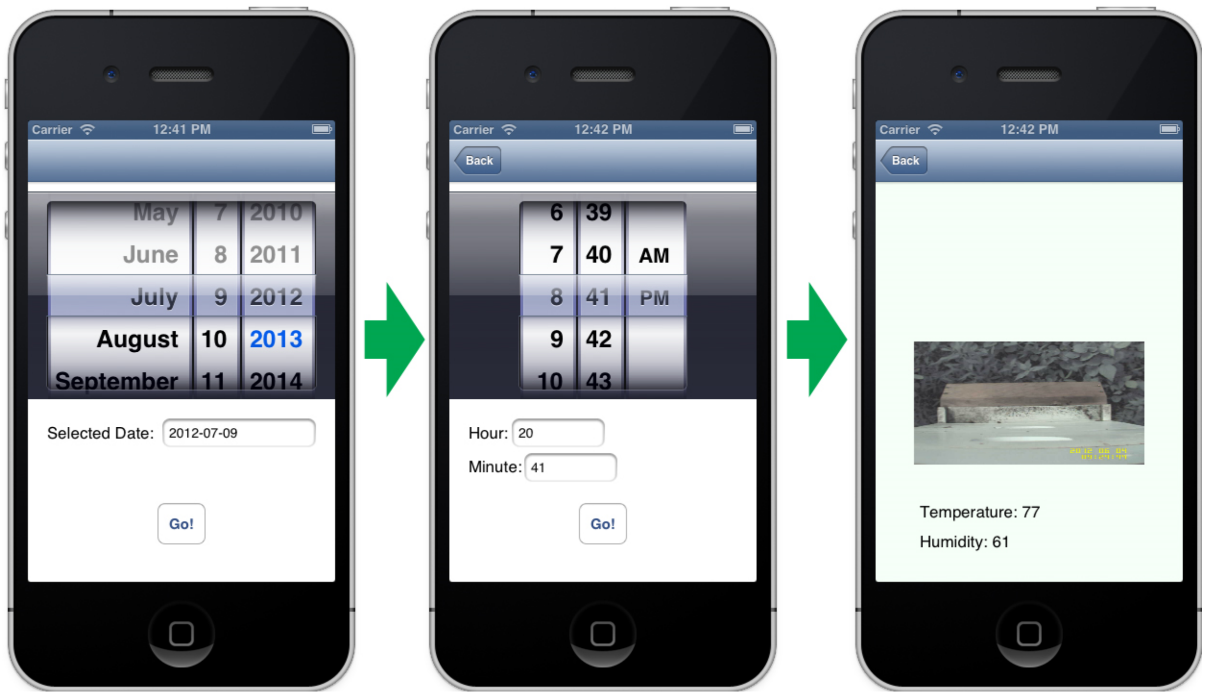


Figure 6.3: Screenshots from the iPhone Application

The flowchart and design of the iPhone application is show in detail in Figure 6.4. As it was noted earlier, this program enables the beekeepers to access the beehive data on their iOS-based devices.

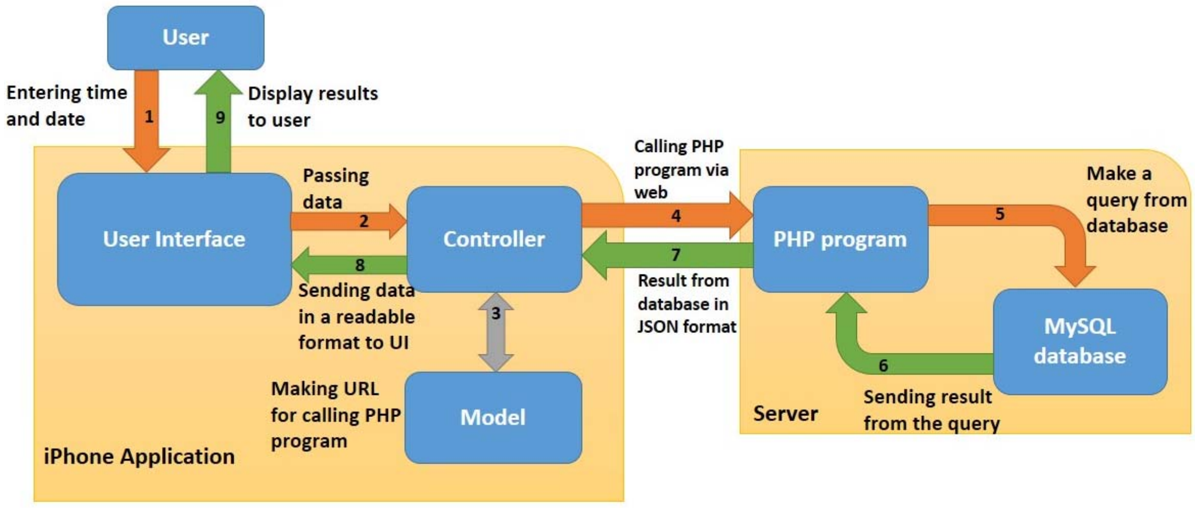

User
Entering time and date
1
9
Display results to user
User Interface
Passing data
2
Controller
8
Sending data in a readable format to UI
3
Making URL for calling PHP program
Model
iPhone Application
Calling PHP program via web
4
7
Result from database in JSON format
PHP program
5
Make a query from database
MySQL database
6
Sending result from the query
Server

Figure 6.4: Flowchart and design of the iPhone application

CHAPTER 7 RESULTS

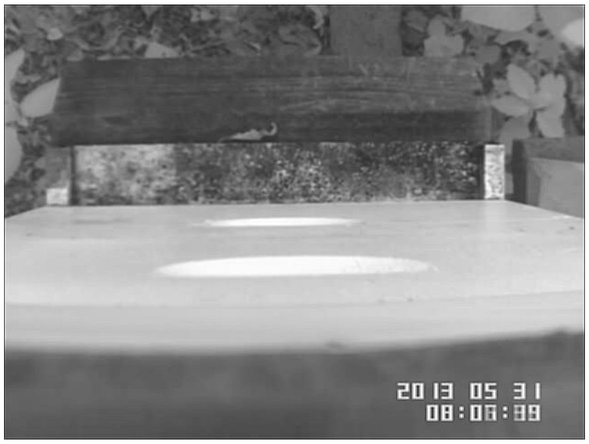
7.1 Introduction

To identify and count the number of honey bees in an image, different image processing techniques are applied in this research. The result is an estimate from the number of bees in the image. In this chapter, the results of the image processing methods used in this research will be provided.

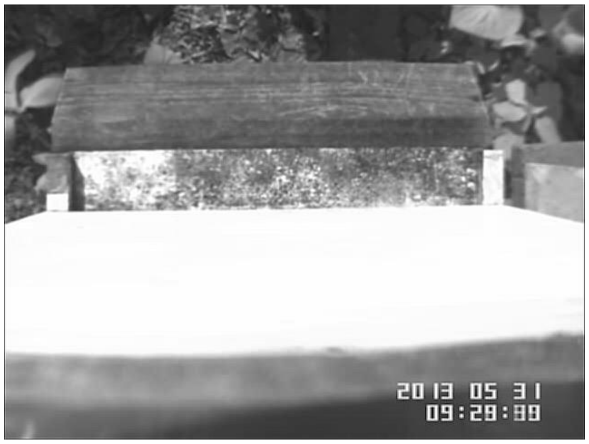
7.2 Background Images

As explained in Chapter 4, the background for an image is obtained by computing the average of 11 consecutive images prior to that image. Using this method, the background can be easily obtained for every image. In this section, the background images are shown for different hours of the day.

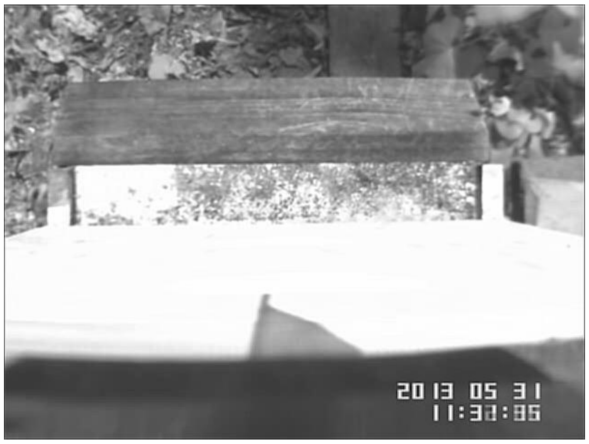
Figure 7.1 and Figure 7.2 show the background image that is computed from the average of consecutive images. Figure 7.1 shows the background of the images that are taken from the top of the hive. The background image in the morning is shown in Figure 7.1 (a & b). Figure 7.1 (c & d) show the same view during mid-day. Finally, the last two images, Figure 7.1 (e & f), show the background image in the evening. Figure 7.2 displays the background images at the same time for the front of the hive. As shown in Figure 7.1 and Figure 7.2, the background illumination changes drastically during the day. These changes prove that it is not feasible to use a static background for all the images taken at different times during that day.



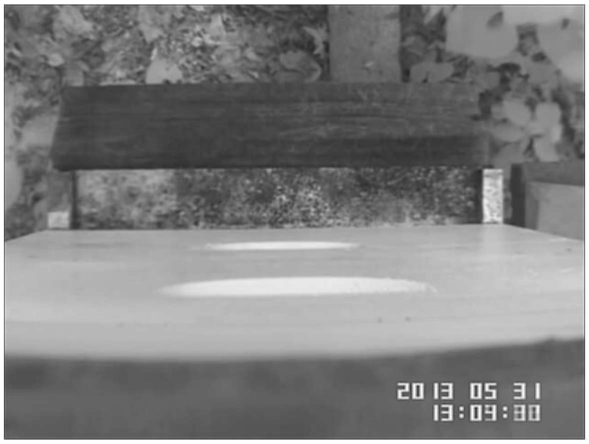
(a)



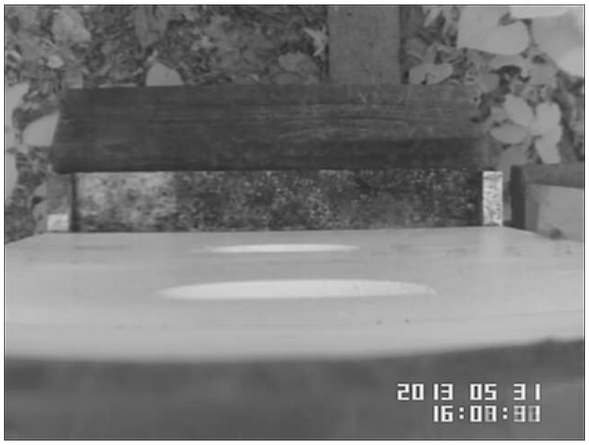
(b)



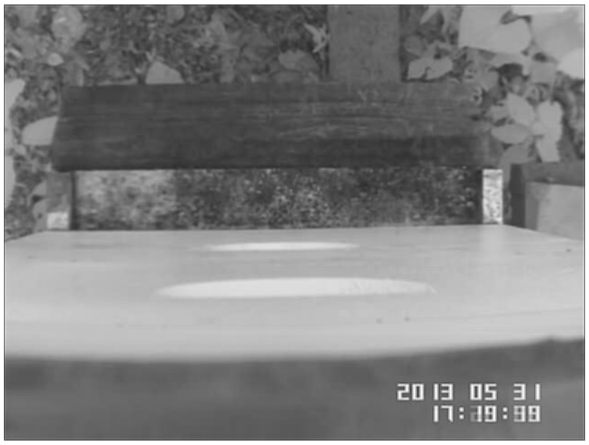
(c)



(d)

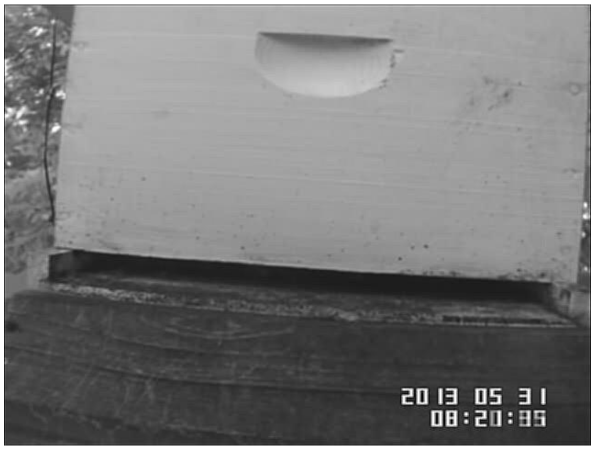


(e)

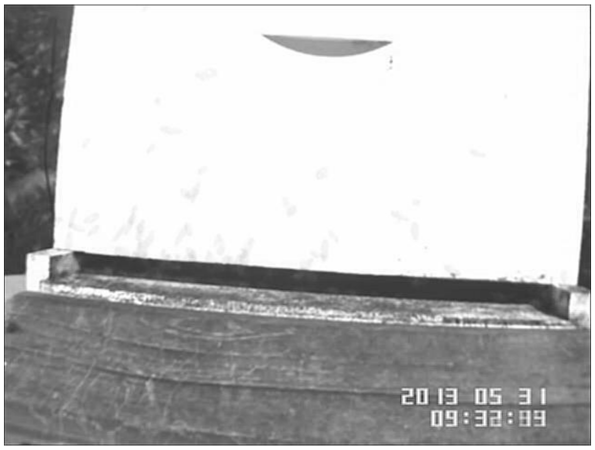


(f)

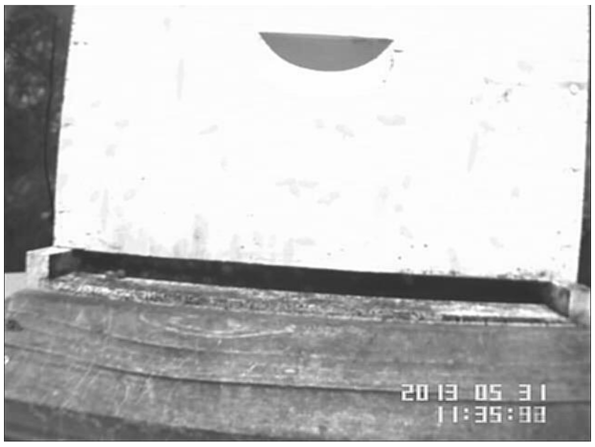
Figure 7.1: Top view background image for different hours of the day



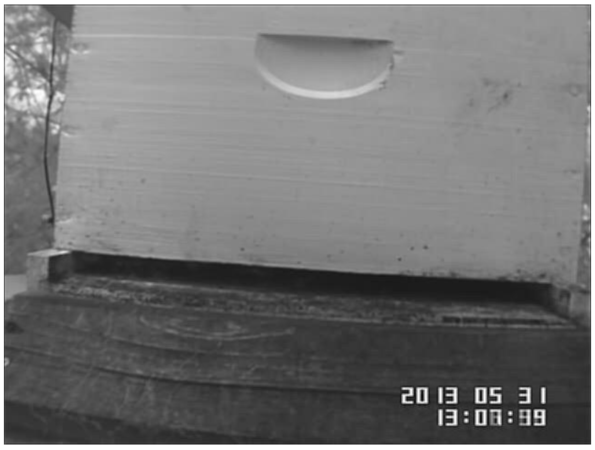
(a)



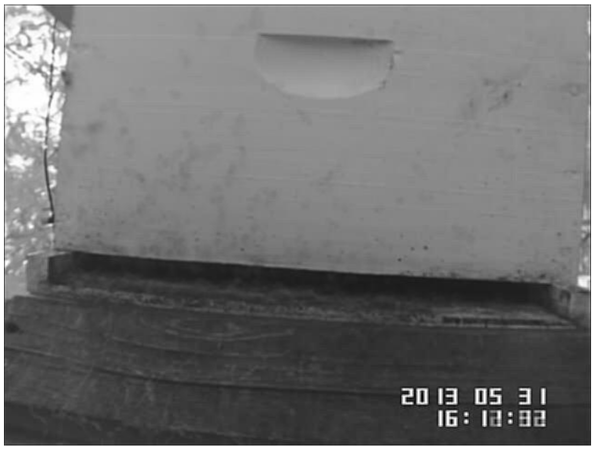
(b)



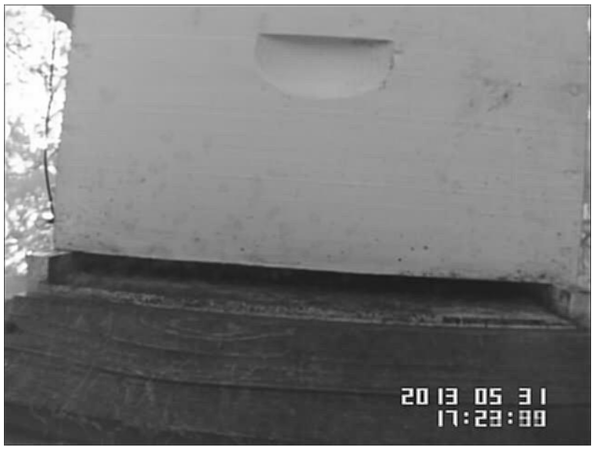
(c)



(d)



(e)



(f)

Figure 7.2: Front view background image for different hours of the day

7.3 Time Extraction

The results of the time extraction process are shown in Figure 7.3 and Figure 7.4. The time and date extracted from the timestamp are depicted in the caption of each image.

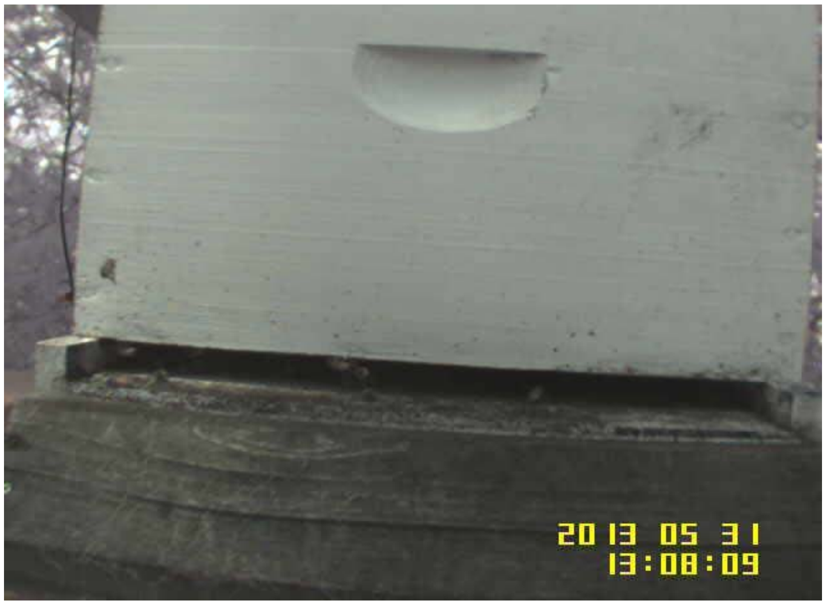


Figure 7.3: Extracted image frame at 13:08:09 on 05/31/2013

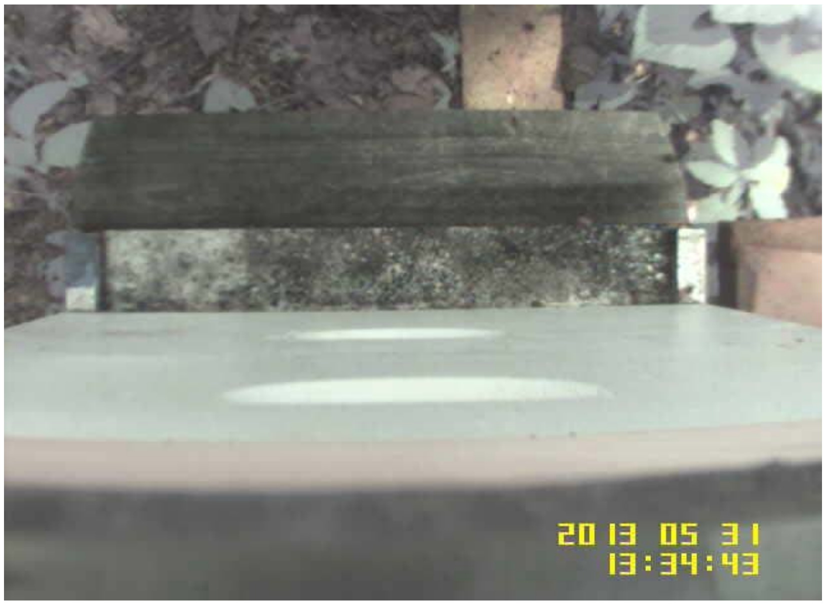


Figure 7.4: Extracted image frame at 13:34:43 on 05/31/2013

7.4 Temperature, Humidity, and Number of Bees

In this section, the climate data fetched from the National Weather Service website [NOAA 2013] and the data extracted from the images taken from the beehive are represented for one day. Figure 7.5 shows the SNR and the number of bees during the day. These data are gathered from images taken from the beehive at different hours.

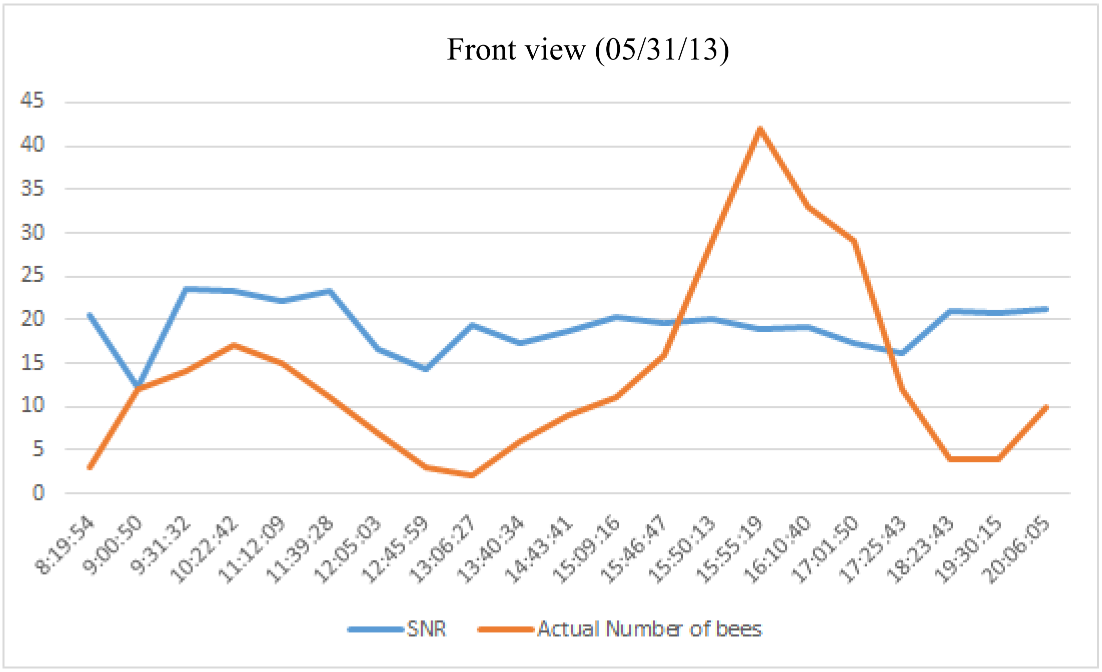


Figure 7.5: SNR and number of bees during the day

Figure 7.6 and Figure 7.7 display the data for the same time and the same beehive. Figure 7.6 shows the SNR in a comparison with the estimated number of bees resulted from the change detection method. Also, Figure 7.7 compares the estimated number of bees calculated from the change detection method with the actual number of bees. For the top view, since it is almost impossible to count the number of bees visually in the images, only the estimated number of bees and the SNR values for each image are shown in Figure 7.8. Figure 7.9 displays the climate data gathered from the National Weather Service website [NOAA 2013], as well as the number of bees for the same day.

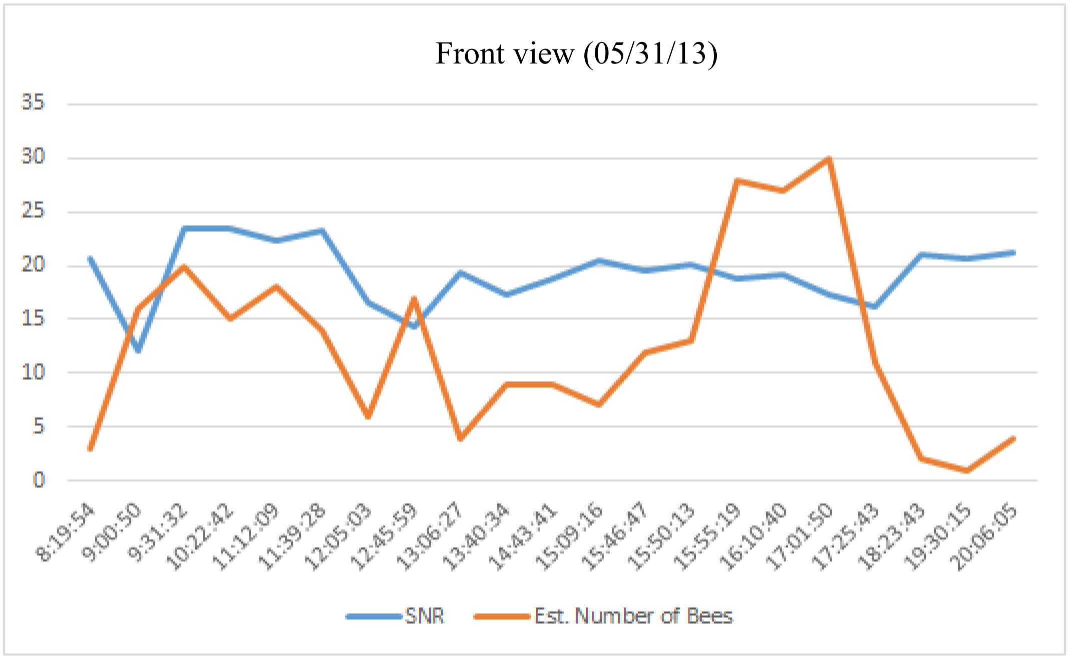


Figure 7.6: Comparison of the SNR and estimated number of bees during the day

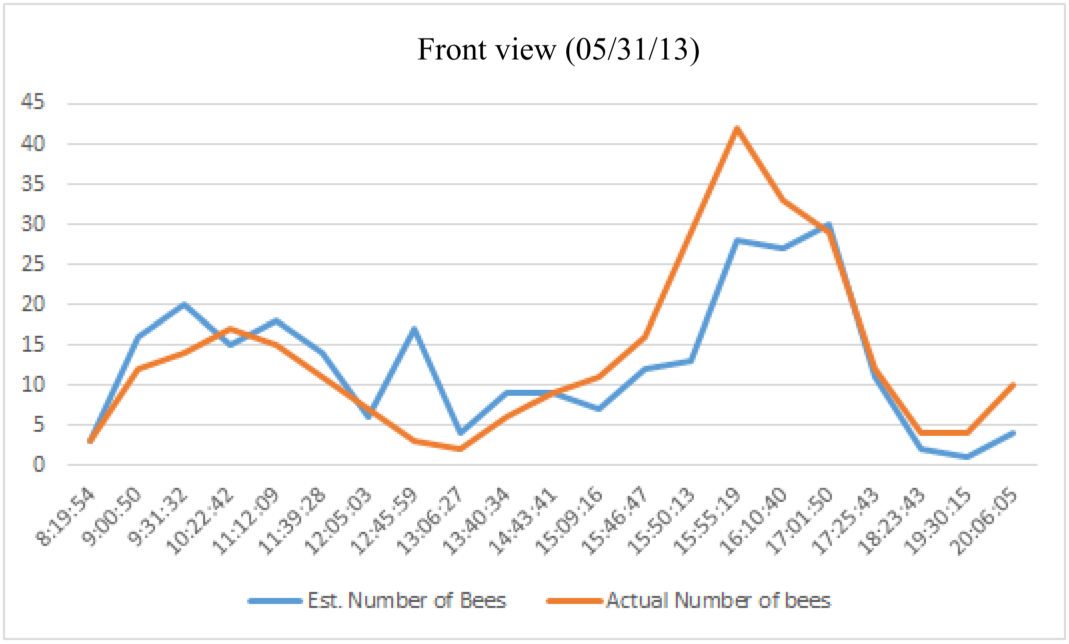


Figure 7.7: Comparison of estimated and actual number of bees during the day

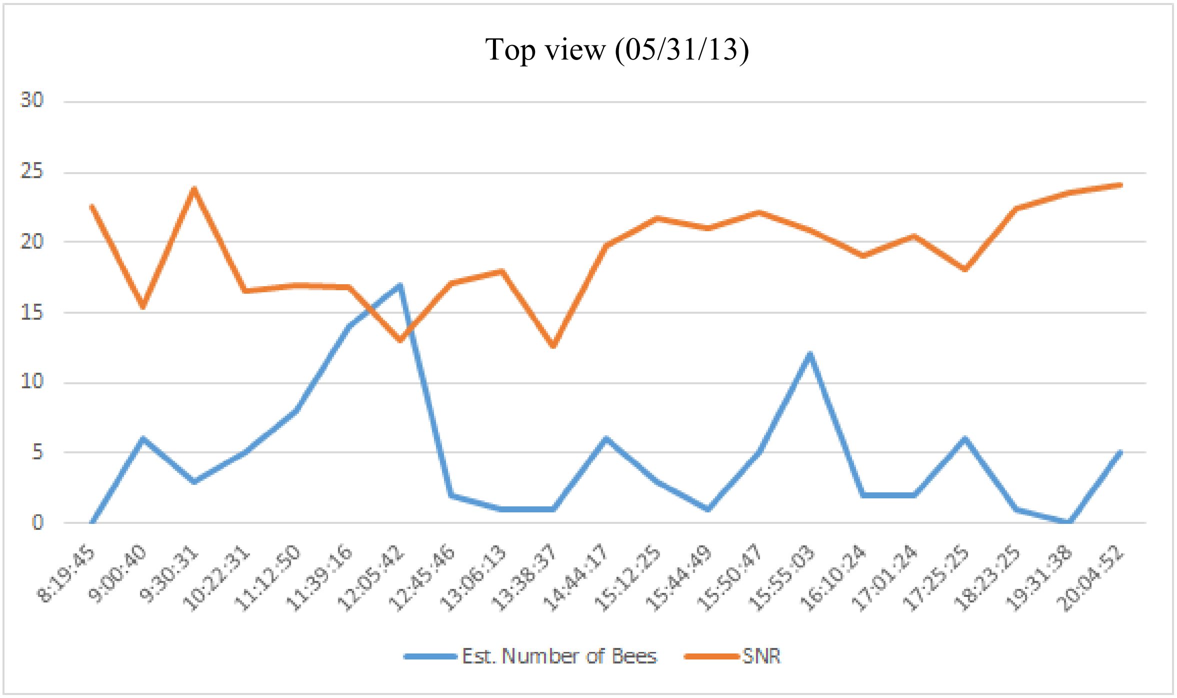


Figure 7.8: Comparison of estimated number of bees and SNR

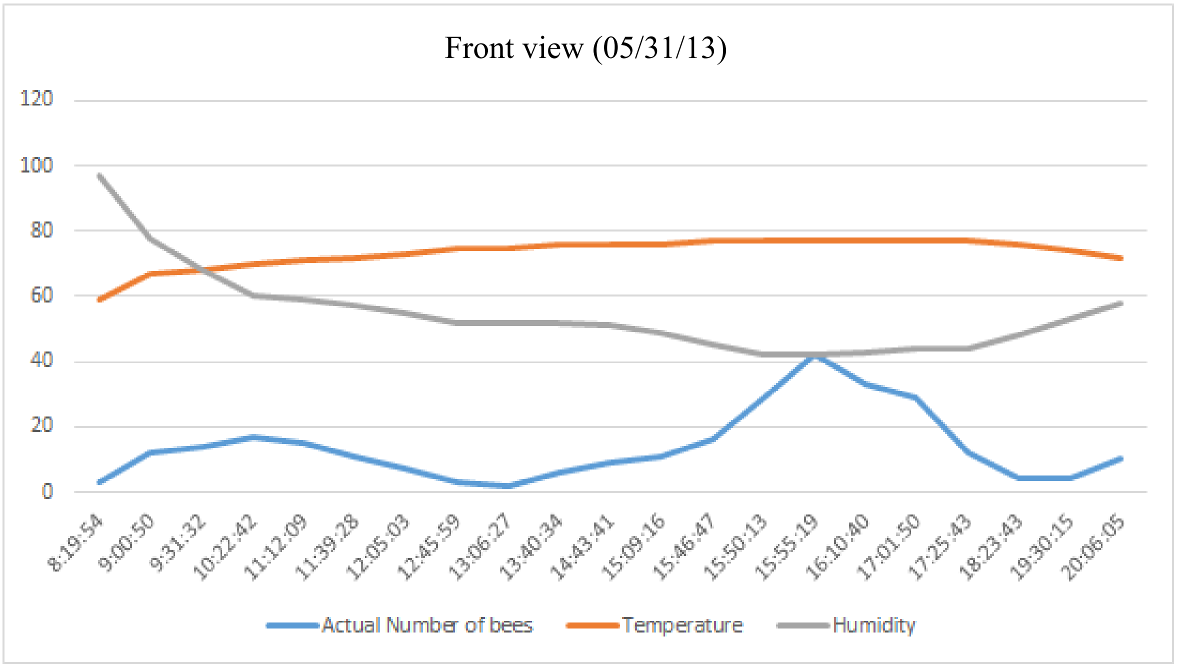


Figure 7.9: Actual number of honey bees by humidity and temperature

7.5 Estimated Number of Bees, SNR, and Entropy

When the number of bees increases in an image, if the background image is modeled as the actual signal and the bees in the image are modeled as the noise, then the value of the SNR can be used to provide an estimate to the number of bees in the image.

This section provides the results of the change detection method besides the values of the SNR and entropy is represented. Figure 7.10 and Figure 7.11 show the values of the SNR and entropy for the images. Figure 7.10 shows the SNR and entropy in images taken from the top view, while the number of the bees are increasing from top to bottom in the images. Figure 7.11 shows the same information for images taken from the front view of the beehive.

Figure 7.12 and Figure 7.13 show the results of applying the change detection method to the images from the beehive. Figure 7.12 shows the results for images taken from the beehive in the front view and Figure 7.13 shows the same results for the top view of the beehive.

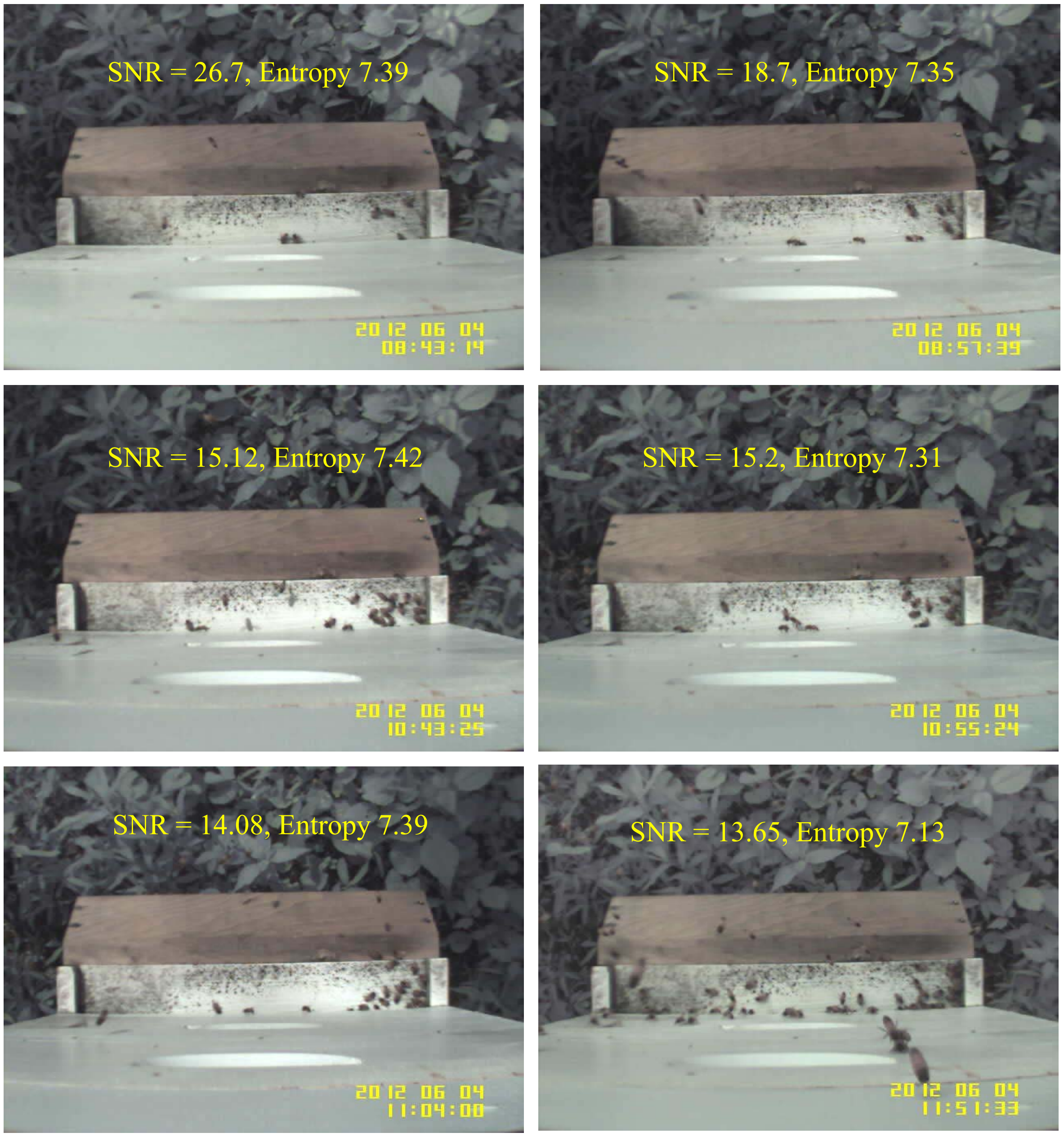


Figure 7.10: SNR and entropy values for top view

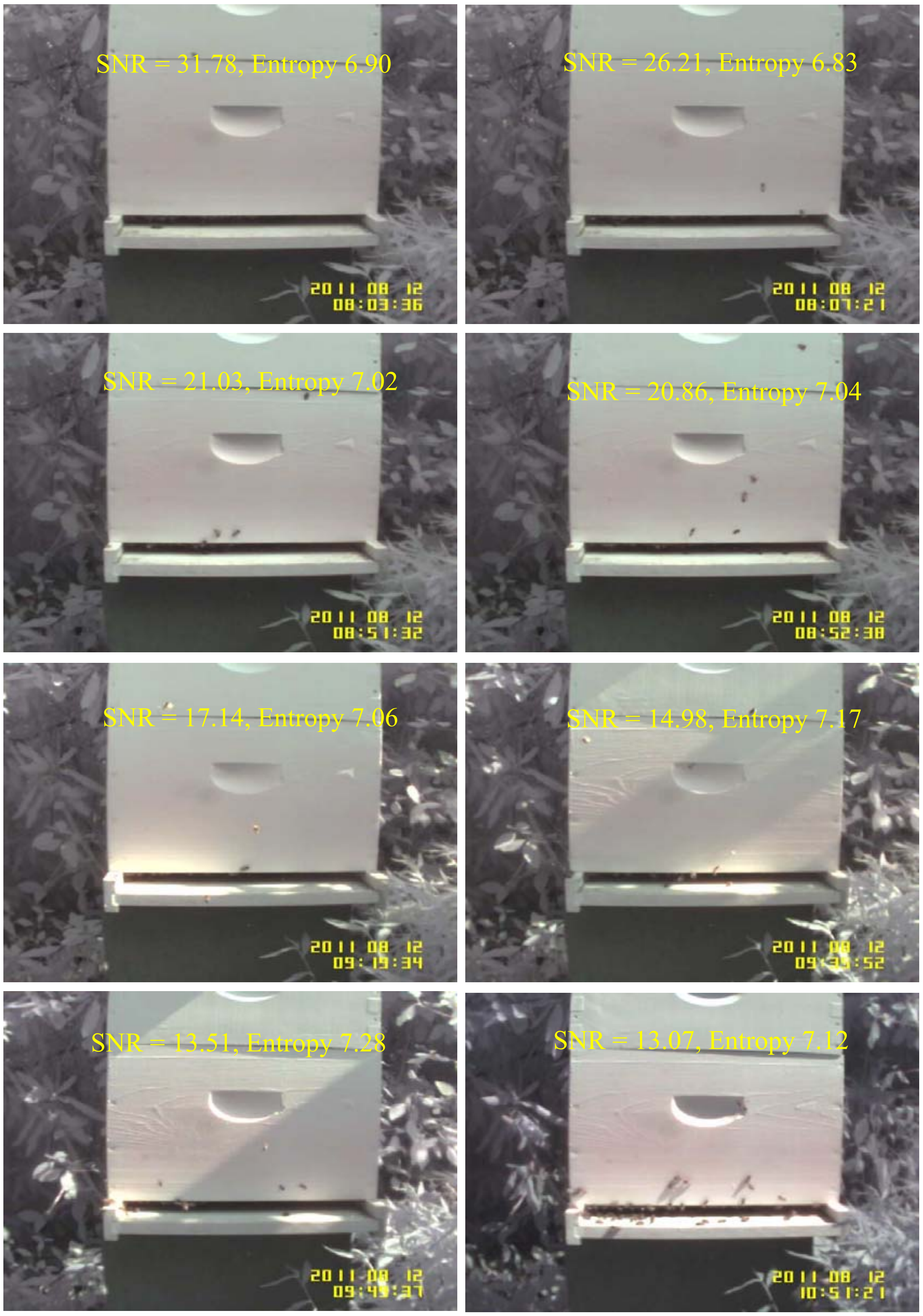
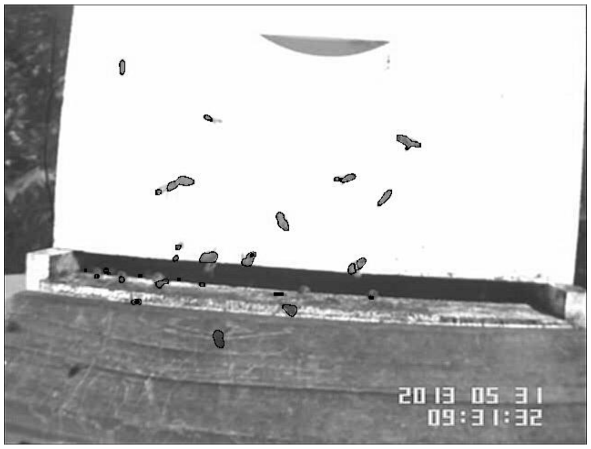
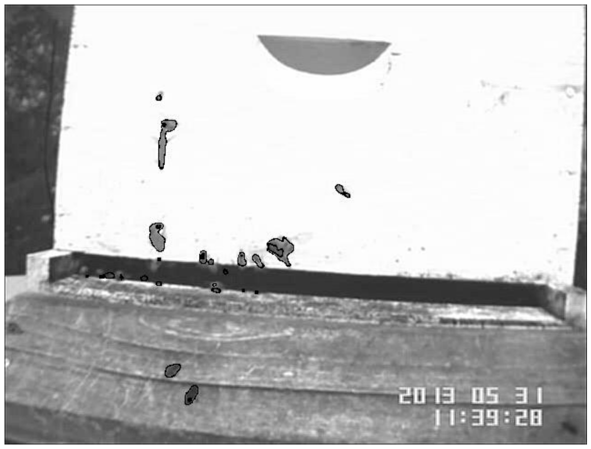

SNR = 31.78, Entropy 6.90
2011 08 12
08:03:36
SNR = 26.21, Entropy 6.83
2011 08 12
08:07:21
SNR = 21.03, Entropy 7.02
2011 08 12
08:51:32
SNR = 20.86, Entropy 7.04
2011 08 12
08:52:38
SNR = 17.14, Entropy 7.06
2011 08 12
09:19:34
SNR = 14.98, Entropy 7.17
2011 08 12
09:39:52
SNR = 13.51, Entropy 7.28
2011 08 12
09:49:37
SNR = 13.07, Entropy 7.12
2011 08 12
10:51:21

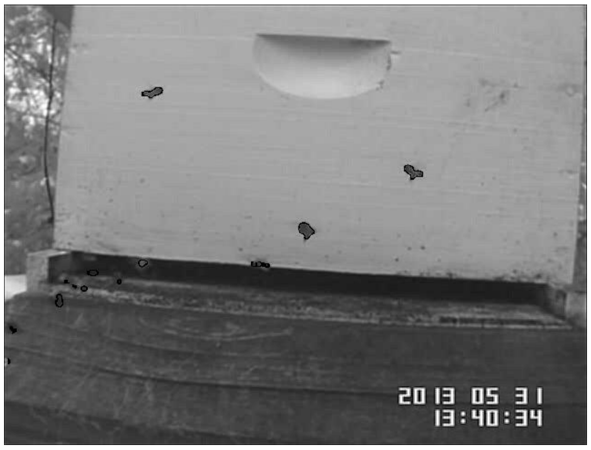
Figure 7.11: SNR and entropy values for front view



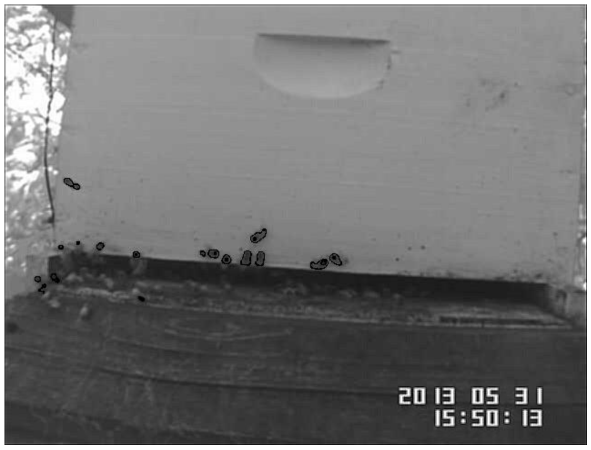
(a)



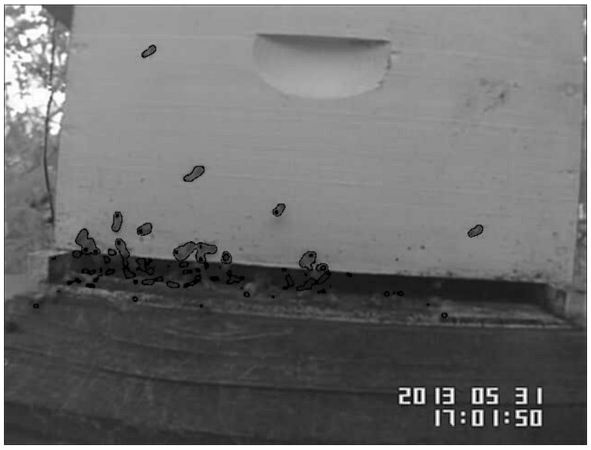
(b)



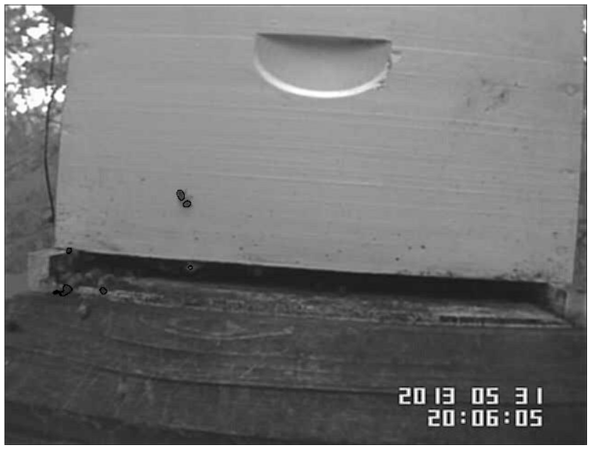
(c)



(d)

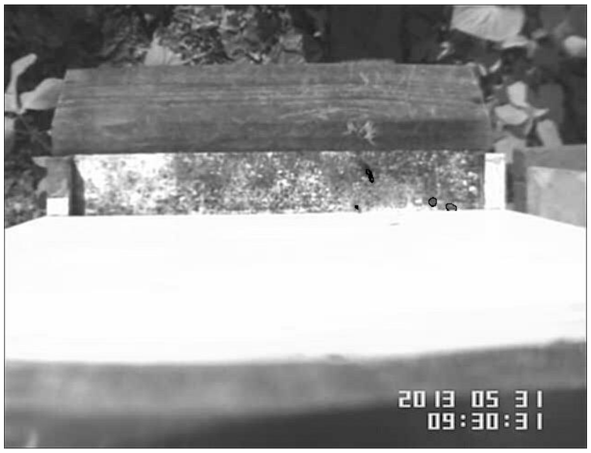


(e)

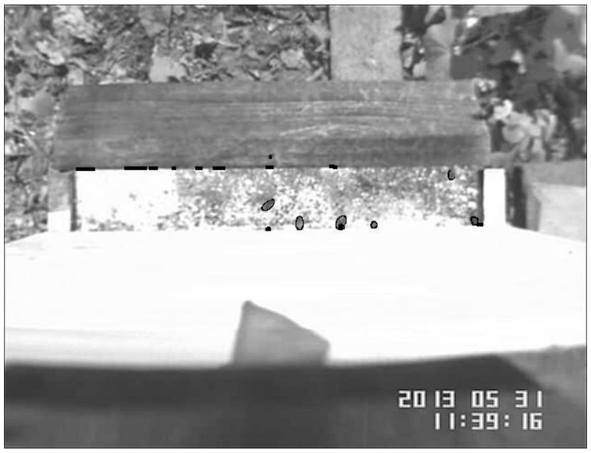


(f)

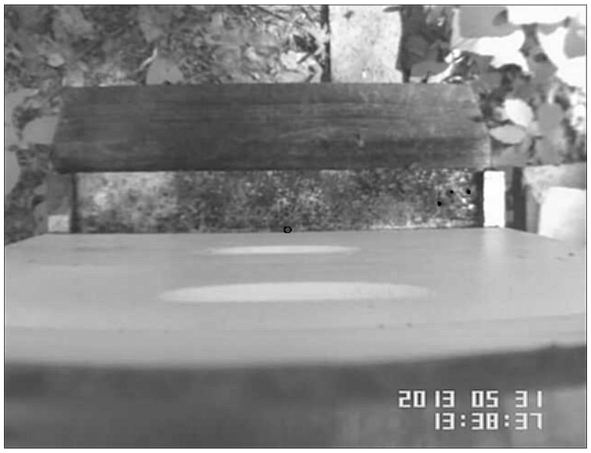
Figure 7.12: Number of bees during a day (front view)



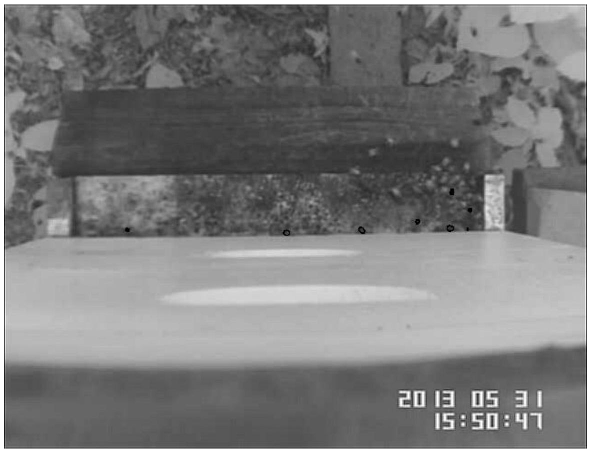
(a)



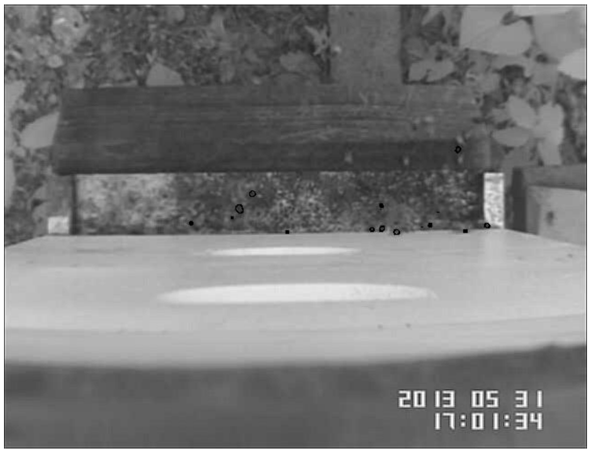
(b)



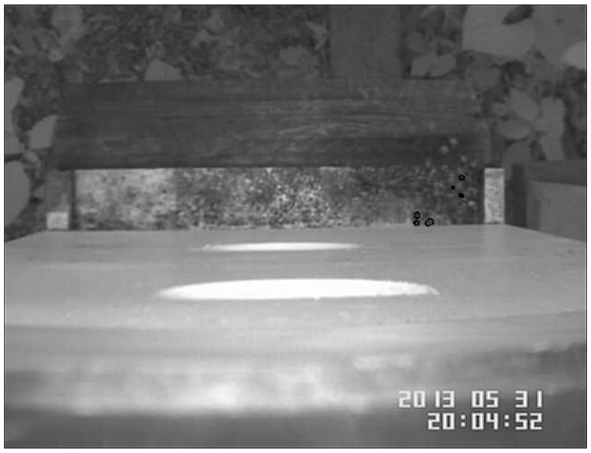
(c)



(d)



(e)



(f)

Figure 7.13: Number of bees during a day (top view)

7.6 Change Detection Algorithm Results

As it is mentioned in Chapter 5, to control the change detection algorithm, two parameters can be changed. The size of the vector that is used in 5.1 to find μ, σ, and the threshold factor. The size can vary from a 3 × 3 window to a bigger size such as 11 × 11. The bigger the size of the vector is, the less changes can be found by the algorithm. To find the changes caused by the bees, the best value should be found by testing different window sizes. The threshold factor determines whether a pixel is classified as a change or as the background. To find the best values for these parameters, different values were tested. The results of those values are shown in Figure 7.14 through Figure 7.18.

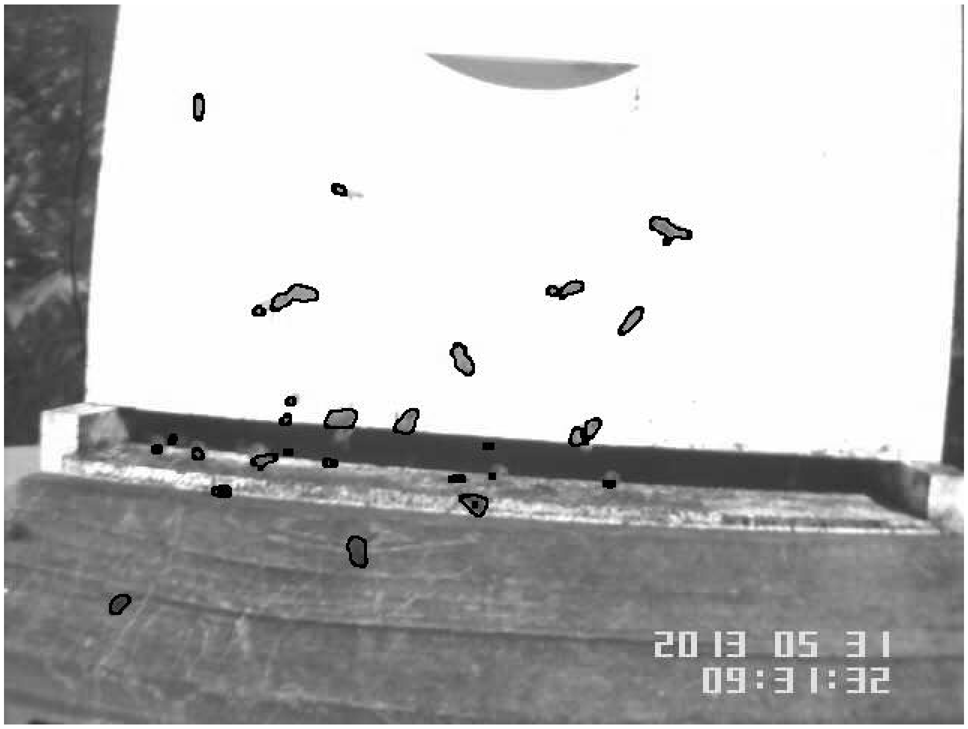


Figure 7.14: Window size=25 and threshold=0.05

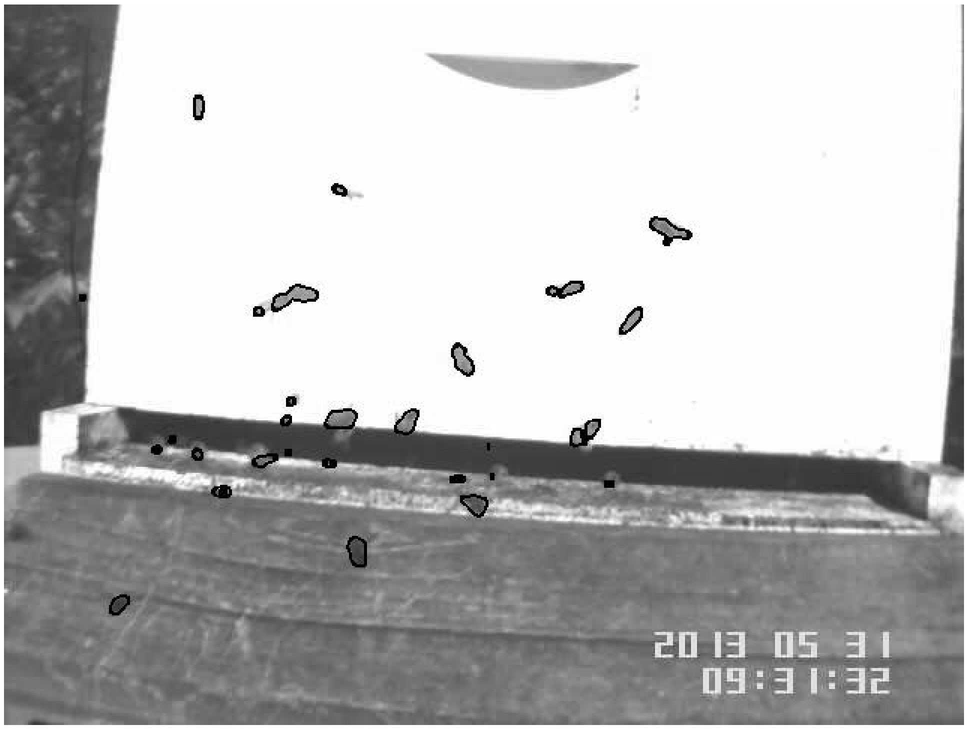


Figure 7.15: Window size=25 and threshold=0.1

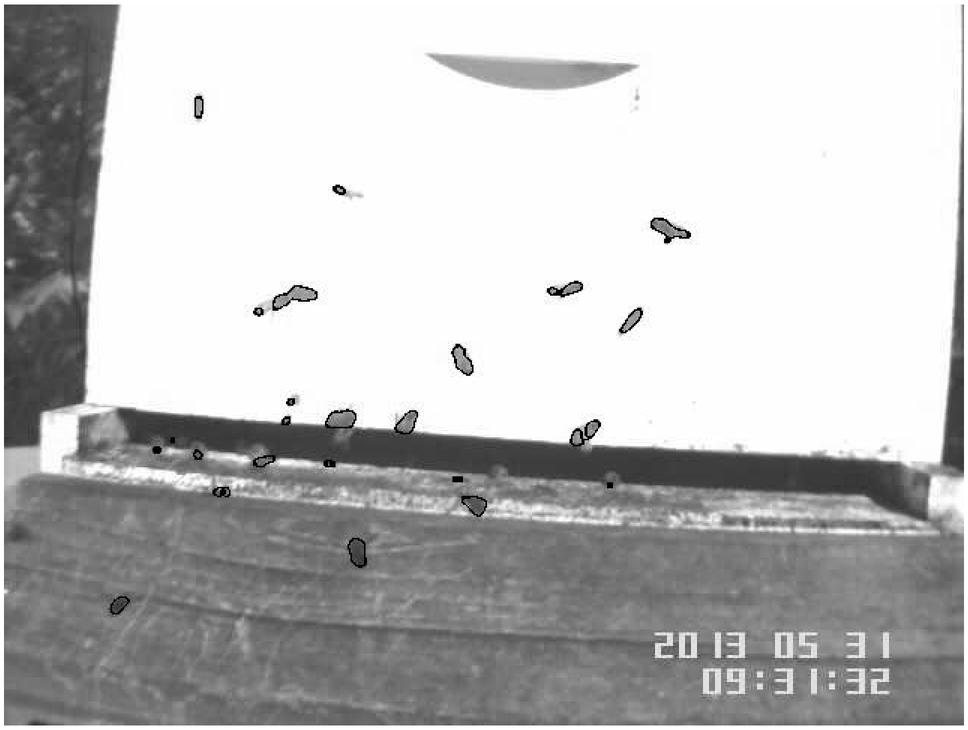


Figure 7.16: Window size=25 and threshold=0.25

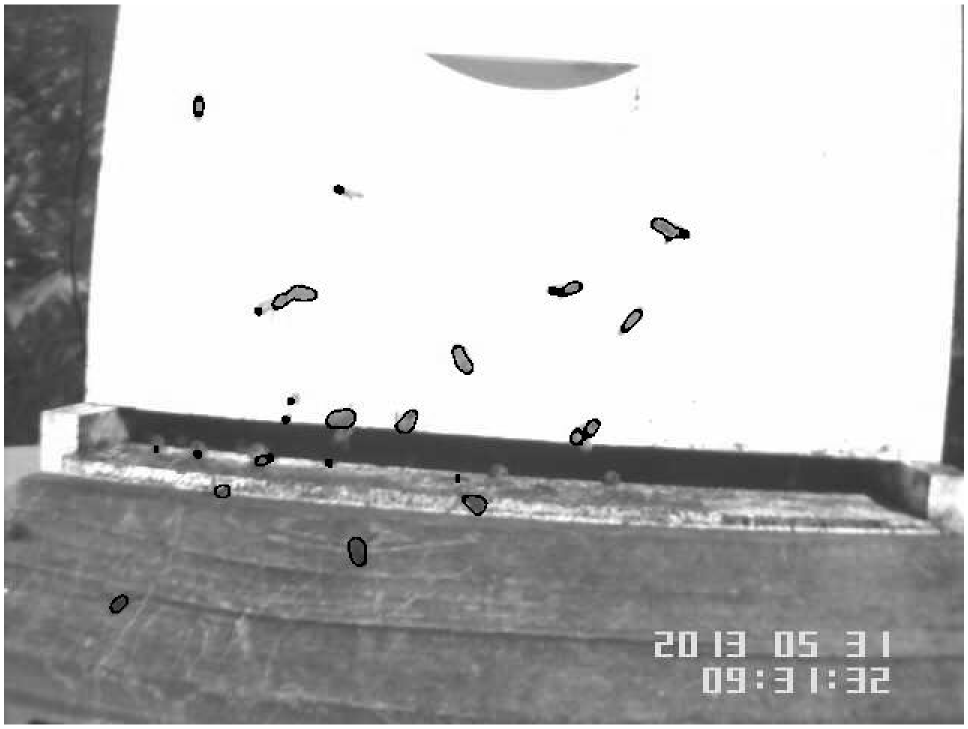


Figure 7.17: Window size=49 and threshold=0.25

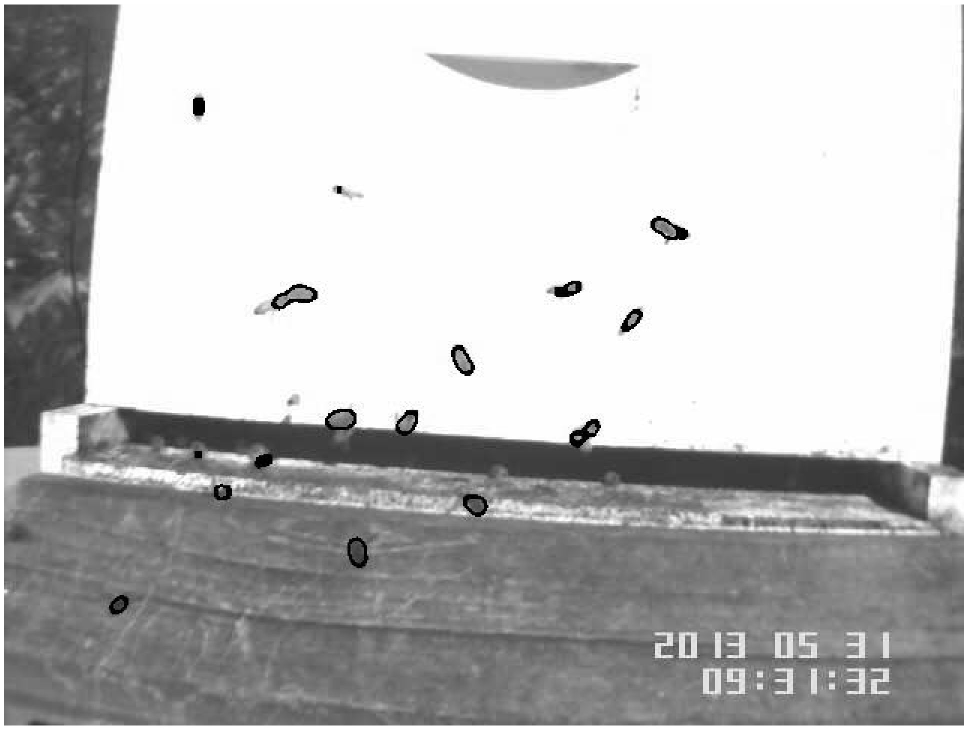


Figure 7.18: Window size=81 and threshold=0.25

CHAPTER 8 CONCLUSION AND FUTURE WORK

This chapter provides the conclusions made for the research and analyses conducted on this thesis.

8.1 Background

For the purpose of this research, and due to the constant changes in illumination in the scene, it was not feasible to make a static background for the scene. Due to this fact, the background image is made from consecutive images prior to an image that is processed by the change detection method. As shown in Figure 7.1 (a) and Figure 7.2 (a), when the illumination does not change much, the background image constructed from multiple images is almost the same as the actual background of an image without bees. Also, as it can be seen in both Figure 7.1 and Figure 7.2, finding the background from averaging eliminates all the bees in the scene. When the illumination changes quickly, for example, during mid-day, it can affect the background resulted from averaging. As shown in Figure 7.2 (b & c), some segments, such as shadows would be added to the background that make detecting the bees challenging. A simple solution to address this problem is to capture the snapshots with less time interval between them. In this research, the interval between two consecutive images is 10 seconds. Using a faster system the time interval could be reduced to 1 second to produce a more accurate image of the background.

8.2 Time Extraction

The Hue-Saturation-Value (HSV) model is used in this research to extract the time and date. Since the color of the timestamp is brighter than the other colors in the image, the value part of this model is used to distinguish this part. By using the value part of the model and using a threshold value equal to 0.7, the time and date were extracted and recognized. As it is shown in Figure 7.3 and Figure 7.4, the time and date were extracted in all the images with 100% accuracy. This method can be used to further the research where the local weather data can be embedded into the image frames of the video as well.

8.3 Temperature, Humidity, and Number of Bees

For the data shown in Figure 7.9, temperature does not change dramatically, but the humidity varies between 95% in the morning to 40% in the evening. Based on the data for our research the number of bees during the day suggests that bee activities are negatively correlated to the humidity. This can be studied further to improve the assumption.

The other parameters that are measured for each image in this research are the SNR and entropy. As stated in Chapter 4, if the image including the bees is modeled as noise and the background image is modeled as the actual signal, calculating the SNR gives a measurement of the noise in the image, hence provides an estimate to the number of bees. A higher SNR shows that there is less noise, hence fewer bees in the image as compared to that of a lower SNR. Another advantage of using the SNR to estimate the number of bees is the simplicity and speed of its calculation compared to the change detection algorithm used in this research to find the number of bees. Therefore, the SNR can be used to find a quick estimate of bee activities from the images. Also, to make the SNR more accurate, it can be calculated using only the part of the image in front of the hive framework, eliminating the movement of plants around the hive.

As shown in Figure 7.7, the estimated number of bees obtained from the change detection algorithm follows the actual number of bees, except during mid-day, when there is a significant difference between the actual number and estimated number of bees. Although this method works for the images that were obtained from the camera installed in front of the hive, it does not work well for the images from the top. As shown in Figure 7.1, changes in the hive color make the background darker, hence, making the distinction of the bees flying in front of the hive more challenging. For example, in Figure 7.13 (d), it is almost impossible to distinguish the bees and count them visually. The change detect algorithm and the SNR value face the same problem.

To find the best value for the parameters used in the change detection algorithm, different window sizes and different threshold values were applied for this algorithm. As shown in Figure 7.14 through Figure 7.18, different values produce different results. Also, depending on the time of the day, some values work better. The best results obtained with the window size of 5 and a threshold value of 0.45. These two values are used for the estimated number of bees shown in Figure 7.7.

8.4 Change Detection Results

The Illumination invariant change detection algorithm is used as the core algorithm for detecting the movements of the bees in front of the hive. The results of this algorithm are shown in Figure 7.14 through Figure 7.18. As shown in Figure 7.14 through Figure 7.16, with the smaller threshold value of 0.05, less changes are detected by the algorithm, thus some bees are not be detected as changes in the image. On the other hand, a larger threshold for the algorithm makes the algorithm more sensitive to the changes, and more segments are detected as changes. To find an optimal value for the algorithm, different values were applied for the threshold value and 0.25 was used as the threshold value in this research.

Figure 7.16 through Figure 7.18 show the effects of changing the window size. As it can be seen from these Figures, window size is negatively correlated to the number of bees detected by the change detection algorithm. Furthermore, increasing window size increases the time for producing the change mask, hence 5×5 was used as the window size to conduct this research.

8.5 Future Work

In this research, an automated monitoring system was designed and implemented. This system, takes images from a beehive and stores them on the server. Then the images are processed on the server using several programs and tools. The collected data include the time and date that the image was taken, an estimation of the number of bees in the image, the SNR, and the entropy of the image which are stored in a MySQL database. Besides these data, weather data are fetched from the National Weather Service website and stored in another table. To make accessing data easier for the users, an iPhone application is designed and implemented. This application makes it possible for the users to access data from everywhere on their iOS-based devices.

For future work, the first improvement is to use higher quality cameras that can transfer color data wirelessly and provide videos with faster frame rate. Therefore, making it possible for using other image processing methods besides the change detection method for detecting the bees.

To decrease the run-time of the image processing algorithms, these algorithms can be implemented in C using open computer vision (OpenCV) libraries. Compared to MATLAB, these algorithms can be run faster using OpenCV.

The local weather data can be obtained locally inside and outside the hive and be embedded into the videos. Since the data are embedded in the videos, synchronization of weather data with the image processing data will be easier. Also, since the weather data are collected locally, they are probably more accurate as compared to the data fetched from the National Weather Service website at that location.

To improve the recognition rate of the bees, a learning algorithm can be implemented to detect the bees using their features. Also, using video processing methods on a faster camera, it is possible to track the bees individually and gather more information about their activities and the effect of climate data on their health.

Bibliography

Til Aach, Lutz Dümbgen, Rudolf Mester, and Daniel Toth. 2001. Bayesian Illumination-Invariant Motion Detection. In *Proceedings IEEE International Conference on Image Processing (ICIP)*, ISBN 0-7803-6725-1 (Ed.), Vol. 3. IEEE, Thessaloniki, 640–643.

Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. 1983. *Data Structures and algorithms*. Addison-Wesley, Boston, MA.

Apiservices. 2007. DATA2 creates the world's smallest barcode. (2007). Retrieved March 10, 2013 from http://www.beekeeping.com/articles/us/barcode.htm

Apple Inc. 2013. Mac App Store - Xcode. (2013). Retrieved August 3, 2013 from https://itunes.apple.com/us/app/xcode/id497799835

Tom Arbuckle, Stefan Schröder, Volker Steinhage, and Dieter Wittmann. 2001. Biodiversity Informatics in Action: Identification and Monitoring of Bee Species using ABIS. In *Proceedings 15th Int. Symp. Informatics for Environmental Protection. Volume*. 425–430.

Kees J. Batenburg and Jan Sijbers. 2009a. Adaptive thresholding of tomograms by projection distance minimization. *Pattern Recognition* 42, 10 (2009), 2297 – 2305. Selected papers from the 14th {IAPR} International Conference on Discrete Geometry for Computer Imagery 2008.

Kees J. Batenburg and Jan Sijbers. 2009b. Optimal Threshold Selection for Tomogram Segmentation by Projection Distance Minimization. *IEEE Transactions on Medical Imaging* 28, 5 (2009), 676–686.

BeesFree Inc. 2012. Bee Colony Collapse Disorder. (2012). Retrieved January 2, 2013 from http://www.beesfree.biz/Ccd

Paul Bourke. 2011. Histogram Matching. (2011). Retrieved April 4, 2013 from http://paulbourke.net/texture_colour/equalisation/

Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and Francois Yergeau. 2013. Extensible Markup Language. (2013). Retrieved August 3, 2013 from http://www.w3.org/TR/REC-xml/

Christopher M. Brislawn. 1995. Fingerprint Go Digital. *Notices of American Mathematical Society* 42, 11 (November 1995), 1278–1283.

Jason Campbell, Lily Mummert, and Rahul Sukthankar. 2008. Video Monitoring of Honey Bee Colonies at the Hive Entrance. In *ICPR Workshop on Visual Observation and Analysis of Animal and Insect Behavior*, ICPR, 1-4.

Jennifer M. Campbell, Douglas C. Dahn, and Daniel A. J Ryan. 2005. Capacitance-based sensor for monitoring bees passing through a tunnel. *Measurement Science and Technology* 16, 12 (2005), 2503. http://stacks.iop.org/0957-0233/16/i=12/a=015

John F. Canny. 1986. A Computational Approach to Edge Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* 8, 6 (June 1986), 679–698.

Li Chen. 1991. The lambda-connected segmentation and the optimal algorithm for split-and-merge segmentation. *Chinese J. Computers* (1991). Cognition. 1999. Newbie: Intro to Cron. (1999). Retrieved March 3, 2013 from http://www.unixgeeks.org/security/newbie/unix/cron-1.html

Cognition. 1999. Newbie: Intro to Cron. (1999). Retrieved March 3, 2013 from http://www.unixgeeks.org/security/newbie/unix/cron-1.html

Xiaolong Dai and Siamak Khorram. 1998. The effects of image misregistration on the accuracy of remotely sensed change detection. *IEEE Transactions on Geoscience and Remote Sensing*, 36, 5 (1998), 1566–1577.

Maureen Dolan. 2000. Tales from hive. (2000). Retrieved April 4, 2013 from http://www.pbs.org/wgbh/nova/bees/buzz.html

Emrullah Durucan and Touradj Ebrahimi. 2000. Robust and Illumination Invariant Change Detection Based on Linaer Dependance. In *EUSIPCO (Cerebrovascular Diseases)*. SPIE, 1041–1044.

Line Eikvil. 1993. *OCR - Optical Character Recognition*. Norsk Regnesentral, P.B., Oslo, Norway.

James D. Ellis, Jay D. Evans, and Jeff Pettis. 2010. Colony losses, managed colony population decline, and Colony Collapse Disorder in the United States. *Journal of Apicultural Research* 49, 1 (2010), 134–136.

Vladimir Estivill-Castro, Darren Christopher Lattin, Francis Suraweera, and V. Vithanage. 2003. Tracking bees - a 3D, outdoor small object environment. In *2003. ICIP 2003. Proceedings. 2003 International Conference on Image Processing*, Vol. 3.III–1021–4 vol.2.

Rafael C. Gonzalez and Richard E. Woods. 2001. *Digital Image Processing* (2nd ed.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Rafael C. Gonzalez, Richard E. Woods, and Steven L. Eddins. 2003. *Digital Image Processing Using MATLAB*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Google Inc. 2011. mjpg-streamer. (2011). Retrieved May 10, 2012 from https://code.google.com/p/mjpg-streamer/

Robert M. Haralick and Linda G. Shapiro. 1992. *Computer and Robot Vision* (1st ed.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Steven L. Horowitz and Theodosios Pavlidis. 1974. Picture Segmentation by a directed split-and-merge procedure. *Proceedings of the 2nd International Joint Conference on Pattern Recognition, Copenhagen, Denmark* (1974), ICPR, 424–433.

Steven L. Horowitz and Theodosios Pavlidis. 1976. Picture Segmentation by a Tree Traversal Algorithm. *J. ACM* 23, 2 (April 1976), 368–388.

Ramesh C. Jain, Rangachar Kasturi, and Brian G. Schunck. 1995. *Machine vision*. McGraw-Hill. I–XX, 1–549 pages.

Ramiro Jordan and Roberto Lotufo. 1997. Bit Plane Slicing. (1997). Retrieved February 18, 2013 from http://www.inf.ufsc.br/~visao/khoros/html-dip/c4/s12/front-page.html

Toshifumi Kimura, Hidetoshi Ikeno, Ryuichi Okada, and Etsuro Ito. 2006. A Study for Identification and Behavioral Tracking of Honeybees in the Observation Hive Using Vector Quantization Method. (2006). Retrieved March 2, 2013 from http://www.noldus.com/mb2008/individual_papers/FPS_animal_behavior_in_the_field/FPS_animal_behavior_in_the_field_Kimura.pdf

Uwe Knauer, Michael Himmelsbach, Frank Winkler, Fred Zautke, Kaspar Bienefeld, and Beate Meffert. 2005. Application of an Adaptive Background Model for Monitoring Honeybees. In *Proceedings of 5th IASTED International Conference on Visualization, Imaging & Image Processing*. Benidorm, Spain, 46–50. http://sites.google.com/site/michaelhimmelsbach/knauer05a.pdf

S. B. Laughlin. 1981. A simple coding procedure enhances a neuron’s information capacity. *Naturforsch* (1981), 910-912.

R.L. Lillestrand. 1972. Techniques for Change Detection. *IEEE Transactions on Computers,* C-21, 7 (1972), 654–659.

Tony Lindeberg and Meng-Xiang Li. 1995. Segmentation and classification of edges using minimum description length approximation and complementary junction cues. In *Selected papers from the 9th Scandinavian conference on Image analysis: theory and applications of image analysis II: theory and applications of image analysis II*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 39–52. http://dl.acm.org/citation.cfm?id=242838.242842

Arnold E. Lundie. 1925. *The flight activities of the honeybee*. U.S. Dept. of Agriculture. http://books.google.com/books?id=w8Q9AAAAYAAJ

MathWorks Inc. 2010. Using MATLAB Engine. (2010). Retrieved October 10, 2012 from http://www.mathworks.com/help/matlab/matlab_external/using-matlab-engine.html

MathWorks Inc. 2011. Create predefined 2-D filter. (2011). Retrieved October 10, 2012 from http://www.mathworks.com/help/images/ref/fspecial.html

Peter Neumann and Norman L. Carreck. 2010. Honey bee colony losses. *Journal of Apicultural Research* 49, 1 (2010), 1–6.

Mark S. Nixon and Alberto S. Aguado. 2008. *Feature Extraction & Image Processing*. Academic Press, Waltham, MA.

National Oceanic and Atmospheric administration. 2013. NOAA National Weather Service. (2013). Retrieved July 4, 2013 from http://www.weather.gov/

Oracle Corporation. 2013. MySQL 5.0 Reference Manual. (2013). Retrieved June 10, 2013 from http://dev.mysql.com/doc/refman/5.0/en/what-is-mysql.html

Minh-Hà Pham-Delègue, Axel Decourtye, Laure Kaiser, and James Devillers. 2002. Behavioral methods to assess the effects of pesticides on honey bees. *Apidologie* 33, 5 (2002), 425–432.

Bui Tuong Phong. 1975. Illumination for computer generated pictures. *Commun. ACM* 18, 6 (June 1975), 311–317.

Simon G. Potts, Stuart P. M. Roberts, Robin Dean, Gay Marris, Mike A. Brown, Richard Jones, Peter Neumann, and Josef Settele. 2010. Declines of managed honey bees and beekeepers in Europe. *Journal of Apicultural Research* 49, 1 (2010), 15–22.

Richard J. Radke, Srinivas Andra, Omar Al-Kofahi, and Badrinath Roysam. 2005. Image change detection algorithms: a systematic survey. *Image Processing, IEEE Transactions on* 14, 3 (2005), 294–307.

George Stockman and Linda G. Shapiro. 2001. *Computer Vision* (1st ed.). Prentice Hall PTR, Upper Saddle River, NJ, USA.

C.-H. Teh and R.T. Chin. 1989. On the detection of dominant points on digital curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11, 8 (1989), 859–872.

The Telegraph Media Group. 2013. The 20 bestselling mobile phones of all time. (2013). Retrieved April 4, 2013 from http://www.telegraph.co.uk/technology/picture-galleries/9818080/The-20-bestselling-mobile-phones-of-all-time.html?frame=2458961

Daniel Toth, Til Aach, and Volker Metzler. 2000. Illumination-invariant change detection. In *Image Analysis and Interpretation, 2000. Proceedings. 4th IEEE Southwest Symposium*. IEEE, 3–7.

M.S. Ulstad. 1973. An algorithm for estimating small scale differences between two digital images. *Pattern Recognition* 5, 4 (1973), 323 – 333.

University of Texas at Arlington. 2013. Division for enterprise development. (2013). Retrieved April 7, 2013 from https://web-ded.uta.edu/wconnect/CourseStatus.awp?~~13PG9081M001

Chris Velazco. 2011. Android Still Most Popular Smartphone OS, IOS Holds Steady in Second Place. (2011). Retrieved June 10, 2013 from http://techcrunch.com/2011/11/03/android-still-most-popular-os-ios-holds-steady-in-second-place/

Karl von Frisch. 1993. *The Dance Language and Orientation of Bees*. Harvard University Press, Boston, MA. http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0674190513

York County Beekeepers' Association. 2011. Honeybee facts and trivia. (2011). Retrieved May 4, 2013 from http://www.ycbk.org/Honeybee\%20Facts\%20and\%20Trivia.htm

Binglong Xie, Visvanathan Ramesh, and Terrance Boult. 2004. Sudden illumination change detection using order consistency. *Image and Vision Computing* 22, 2 (2004), 117 – 125. Statistical Methods in Video Processing.

Appendix

All the programs developed for this thesis are included on the enclosed CD. Following is the list of all the programs:

1. finalparser.php: This program parses the xml file containing weather data from NOAA [2013] and stores the data in a MySQL table.
2. iPhoneProgProj.php: This program reads the data from the MySQL table, changes the format to JSON, and sends them to iPhone application.
3. matlabphp.c: This program executes a MATLAB function and displays the results to the output.
4. phpmatlab.php: This program runs the matlabphp.c program on the web, displays the results to the user, and stores them in a MySQL table.
5. time_extraction.m: This MATLAB program extracts the time and date from a single hive image and displays the results
6. indepent_change_detection_with_n_images.m: This program makes the background from 11 recent images, applies the change detection to the current image and background, and displays the results.
7. beeproject.sh: Bash script that converts the video from analog input to image frames at a desirable time interval between consecutive image frames.
8. counting.m: This programs counts the number of white segments in a binary image.
9. isPointInsidePolygon.m: This program checks whether a given point in the image is in a polygon. The polygon is specified by the coordinates of its vertices.
10. snrCalculator: This program computes the value of SNR and entropy for a given image and its corresponding background.

Vita

Mr. Ahmad Ghadiri was born in the historical city of Esfahan, Iran. After finishing high school in his hometown, he pursued his Bachelor’s degree in the field of Electrical Engineering. After graduating from Isfahan University of Technology with a degree in Electrical Engineering in 2010, he was hired as a product manager in a Value Added Service (VAS) Company and worked there for one year. In Spring 2011, he started his graduate studies at Appalachian State University, Boone, NC, USA. During the course of his graduate studies at Appalachian State University he was involved with the Summer Ventures Program as an instructor of the Visual Image Processing (VIP) research course for two years. He received his Master of Science Degree in August 2013.