AUTOMATED BEEHIVE SURVEILLANCE
USING COMPUTER VISION

A Thesis
by
DAVID JOEL KALE

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

August 2015
Department of Computer Science

AUTOMATED BEEHIVE SURVEILLANCE
USING COMPUTER VISION

A Thesis
by
DAVID JOEL KALE
August 2015

APPROVED BY:

_____

Rahman Tashakkori
Chairperson, Thesis Committee

_____

R. Mitchell Parry
Member, Thesis Committee

_____

Cindy A. Norris
Member, Thesis Committee

_____

James T. Wilkes
Chairperson, Department of Computer Science

_____

Max C. Poole, Ph.D.
Dean, Cratis D. Williams School of Graduate Studies

Abstract

AUTOMATED BEEHIVE SURVEILLANCE
USING COMPUTER VISION

David Joel Kale
B.S., Appalachian State University
M.S., Appalachian State University

Chairperson: Rahman Tashakkori

The number of honey bees entering and leaving a hive throughout the day is an important metric for beekeepers monitoring the health of their hives. Activity at the hive entrance depends on many factors such as weather, colony health, and presence of an adequate food supply. Some commercial systems that utilize infrared sensor gates at the hive's entrance exist for accurately measuring honeybee traffic. In this thesis, a solution that is based on visual information obtained through videos taken in front of the hives is explored. This thesis provides details on the design and implementation of a beehive surveillance system, describing the algorithms and techniques used. The system operates in a realistic outdoor apiary environment, providing constant surveillance in real time. Results obtained using different algorithms within the system are compared and methods for evaluating performance are discussed.

Acknowledgments

Firstly, I would like to express my sincerest gratitude and appreciation to my mentor and friend Dr. Rahman Tashakkori. The guidance and insight he has provided along the way has been invaluable to my success. This project also could not exist without him, for I would not have had the resources to undertake this research if it were not for him and his beehives.

I would also like to thank my committee members, Dr. Parry and Dr. Norris, for their time and effort. They have both been great resources and their contributions are greatly appreciated. Additionally, their courses have equipped me with the skills required to complete this thesis, and I am grateful for that as well.

To my family who has supported me throughout the years- thank you. Without your support I never could have come this far. The same goes to the friends I have made during my time here at Appalachian State- you have helped push me to new limits and shown me incredible times. I sincerely hope our paths cross again.

Last but not least, I would like to thank my girlfriend Julie Ragsdale. Your endless compassion and support has never failed to motivate me to succeed. Without you to keep me on track I may have given up long ago. Wherever I go next, I look forward to being there with you.

Table of Contents

Chapter 1 - Introduction

1.1 Visual Surveillance Systems

Visual surveillance systems are an active area of research in computer vision. These systems generally attempt to detect, identify, or track objects in video sequences. The goal of such systems is to be able to automatically understand and describe the behavior of these objects. Moreover, these systems can effectively replace traditional video surveillance systems that require active monitoring by a human worker. A computer automated system can tirelessly monitor any number of camera feeds without suffering lapses in attention or other human error. Visual surveillance has a wide range of applications, including security for important locations, traffic monitoring in cities, detection of military targets, and more [1].

1.2 Motivation

Honeybees play an important role in agriculture, being responsible for a large portion of crop pollination around the world. Beekeepers can also harvest honey, beeswax, propolis, pollen, and royal jelly from their hives. With honeybees playing such an important role and producing so many useful products, it is of utmost importance for beekeepers to be able to manage and monitor the health of their hives.

This research explores the use of visual surveillance for monitoring honeybee hives. Visual information from the entrance of the hive is valuable because bees rarely die inside the hive. A balanced flow of bees entering and departing from the hive is an indication of a

healthy hive, as bees leave the hive to gather resources and return when they are done. This information could be gathered by a sensor based system, utilizing infrared gates to detect objects passing through the hive entrance [2]. However, a vision-based system has some key advantages over one that is based on sensors. The vision-based system could count the number of bees loitering at the entrance of the hive, which may indicate swarm behavior, and could be extended to detect the presence of non-bee intruders, such as beetles. Additionally, a visual surveillance system provides an archive of easily understood data that the beekeeper could manually inspect later should an anomalous activity occur. Common tasks of surveillance systems such as object detection and tracking can be applied to solve these problems.

1.3   Research Overview

Object detection and tracking are two major areas of computer vision that are used heavily in nearly all visual surveillance systems. This research applies traditional object detection and tracking methods to beehive surveillance, i.e., monitoring the bees around the entrance of the hive using a video feed. As part of this research, a system is implemented for surveillance, and its performance using different algorithms is evaluated in this specific domain. Two goals of the system developed in this research are to detect bees in an image in order to count them and to track them over a short period of time to determine whether they are entering or leaving the hive. Constraints on the system are as follows:

- It should be capable of handling videos taken outdoors without requiring artificial lighting or modifications to the beehive.
- The system must be able to process videos in real-time to allow for constant surveillance.

1.4 Thesis Organization

The rest of this thesis is organized as follows: Chapter 2 examines recent works in visual surveillance systems, object detection, and tracking. Similar works that study honeybees using both visual and non-visual techniques are also reviewed. Chapters 3, 4, and 5 provide the theoretical background for background segmentation, object detection, and motion tracking, respectively. Chapter 6 discusses the methodology of this research. Sections 6.2 through 6.5 discuss the implementation of data collection, background subtraction, object detection, and motion tracking stages of the surveillance system, respectively. Chapter 7 presents the results of the experiments that were carried out using the previously mentioned methods. The performance at each stage of the surveillance is discussed and the different algorithms used at each stage are compared. Chapter 8 concludes the thesis, summarizing the findings and providing a brief overview of the system that was developed. Some possible directions for future work are also introduced.

Chapter 2 - Literature Review

Visual surveillance systems are by no means a new or understudied research topic in computer vision. There have been numerous systems developed utilizing different algorithms. Regardless of the system's purpose, whether it is for security or for studying some behavior, nearly all of these systems share commonalities in the steps they take to go from a raw video feed to more useful information. Nearly all surveillance systems begin with a motion detection step. This process usually involves modelling the environment and classifying pixels that build the model as either corresponding to an object in motion or to the stationary background. Next, these systems attempt to classify the moving objects into categories, using shape, color characteristics, motion, or some other model of appearance. Objects that are deemed important by classification are frequently tracked over time, and finally, all this information is fused in order to understand or describe their behavior [1]. Historically, a large portion of visual surveillance research has been focused on systems for monitoring humans and vehicles, as these systems tend to have the most practical applications. Regardless, many of the principles and patterns of these systems can be applied to the surveillance of honey bee hives.

Constructing and updating a model of the environment is an important step in surveillance. A model of the environment can either be 2D or 3D, but they are often 2D due to the complexity of modelling in 3D. For a single, fixed camera system, 2D is all that is possible but it is often sufficient. In such a fixed camera system, the problem lies in recovering and updating the background from a sequence of images. Illumination changes

and small motions from things like grass or leaves blowing in the wind add difficulty to the problem. There have been several studies introducing methods to address these concerns. Using a temporal average of recent frames was shown to be effective for handling gradual changes in the environment [3]. Modelling each pixel as a mixture of Gaussian distributions is an even more robust technique [4,5], protecting against sudden changes in illumination by maintaining multiple distributions corresponding to different possible backgrounds for each pixel. A simple but less robust method for determining the foreground of an image is to use a single reference image of the background, and detect moving regions in a pixel-by-pixel fashion by finding the difference between the corresponding pair of pixels in the current image and reference image. Similar to this method, temporal differencing uses the pixel-wise differences from consecutive frames to find moving regions. While this is more adaptive to dynamic environments, there are often holes inside of large moving objects when its color is relatively uniform. Additionally, areas of the background revealed by motion will be erroneously classified as foreground [6].

Moving objects in a scene often correspond to unique objects, or entirely different classes of objects, such as scene containing multiple people and vehicles. To effectively track and analyze the behavior of these objects, a surveillance system needs to be able to correctly classify these moving objects. The predominant method for this task is based on examining the shape of the object. Useful features for this classification include blob dispersedness, blob area, and aspect ratio. These features can be used with a neural network to differentiate people from vehicles as illustrated by [6]. A more recent and highly successful system for classifying moving objects as human or not human used a histogram of oriented gradients to describe image regions and classified them using a support vector machine (SVM) [7].

5

Once objects are detected and classified, they are generally tracked. Tracking over time involves matching detected objects in consecutive frames by reasoning about which matches are the most likely. Aircraft have been tracked by computing scale-invariant feature transform (SIFT) keypoints in each motion region, and matching these keypoints between consecutive frames [8]. Another study matches detections between frames by optimizing the matches such that they provide the overall shortest path for all of the objects in the scene using the k-shortest paths algorithm [9]. Related to this thesis, Black et al. use Kalman filters to track multiple human targets [10]. They use the Kalman filters to make predictions about the future state of the detected objects, so when objects are occluded they can maintain an approximation of its location.

This thesis deals with the detection and tracking of honeybees in front of their hive. The behavior of honeybees at the hive has been studied for nearly a century. Work by Lundie [11] represents a very early attempt that uses sensitive pressure plates to connect an electrical circuit when a bee walks across it, registering either an arrival or departure. The gates containing these triggers can only be open on one end at a time, meaning if a bee enters through the gate, the next bee passing through must be exiting and vice versa.

Another non-visual system that can very accurately count how many bees have entered and left the hive is described in [2]. This study describes a counter utilizing infrared sensors at the entrance. The system uses 32 bi-directional pathways wide enough for a single bee to pass through. Movement in these pathways is detected using two infrared beams. The order in which the beams are interrupted indicates whether a bee has entered or left the hive.

More recently, computer systems using image processing techniques for surveillance have been explored. Kimura et al. [12,13], conducted a study of honeybee behavior inside of

the hive by placing a populated beehive frame behind glass, and capturing video footage through the glass. They used vector quantization to separate a large number of honeybees from the background, resulting in areas that they classify as either a single honeybee region (SHR), or a plural honeybee region (PHR). They extracted all SHRs and then attempt to break the PHRs into SHRs. In their study, they initialized the tracking of each bee based on the center points of the SHRs, and they examined the tracks of the honeybees to study behavior within the hive.

Chen et al. [14] monitored the in-and-out flow of honeybees from a hive using video sequences. They used circular tags with a unique code attached to honeybees to identify each bee individually. As honeybees entered and left the hive, an imaging system captured video frames and the bees were located using a Hough transform to find the circular tag. Character recognition using an SVM was employed to read the tag and identify the bee so that it could be associated between frames. This also allowed them to record the times at which a specific honeybee departed or returned to the hive. This research uniquely tagged and observed 100 honeybees. Using the tags made detecting and tracking the bees much easier, but it is not feasible for a real hive with thousands of bees.

Work similar to this thesis is that of Salas and Vera [15]. They used an indoor hive with two circular entrances, with a mechanism to force them to only be accessible from a single direction, one for entering and one for exiting the hive. They used a front facing camera to detect the presence of bees in these entranceways using an SVM. They detected bees that were not in the entranceway using a layered background model to detect foreground regions corresponding to bees. Their research tracked bees over time using a Bayesian inference framework, and determined arrivals and departures by examining these tracks in

relation to the detections at the entrance and exit holes. For example, if a track began inside of the entrance hole and ended outside of the entrance hole then a departure was registered, and vice versa.

Another related work is by Campbell et al. [16]. They describe a visual system for monitoring beehives with the goal of detecting bees in the frame and also tracking them. For detection, background subtraction is performed using the average of the most recent 300 frames as a reference image, and then elliptical template was matched across the background-subtracted frame. To track bees, they solved frame-to-frame matching using maximum weighted bipartite graph matching. They expressed each edge in this graph as the likelihood that a bee with a certain location a could move to location b in the time span of a single frame. They examined the properties of tracks that did not get matched to a bee in order to determine arrivals and departures. For detection, they reported a precision and recall of 0.94 and 0.79 respectively, although they do not discuss by what metric they consider a detection to be a hit or miss. They showed promising results for tracking, reporting only 2% overcounting for departures, and 7% undercounting for arrivals.

Prior work by Ghadiri [17] explores combining data obtained from image processing with weather data in order to monitor and study honeybee activity. The number of bees in front of a hive was estimated using signal-to-noise ratio, entropy, and a background subtraction method. These results were compared to a ground truth. The data were then correlated with the temperature and humidity for the day. This research used surveillance cameras that capture frames at 16 frames/sec, which is a prohibitively low sample rate for tracking fast moving objects such as bees.

Previous works that are similar to this thesis have all performed their experiments on disparate data sets which were not made publically available. Due to this, it is challenging to compare the results of these different studies. This thesis compares the performance of multiple different algorithms which have not been utilized in previous work, allowing conclusions to be drawn about the effectiveness of certain methods in the context of beehive surveillance. Compared to previous work, this thesis presents a novel way of using tracking information to count the arrivals and departures of honeybees. While previous works have focused on examining the properties of lost tracks, this thesis explores a solution based on examining the tracks before they are lost.

Chapter 3 - Background Subtraction

3.1   Introduction

In visual surveillance systems, a common first step is background subtraction. In general, background subtraction or background segmentation refers to the classification of all pixels in a frame as either foreground or background. In a system with a single, static camera, the foreground will consist of all moving objects and all regions of the frame that undergo changes in illumination due to environmental factors such as the sun and shadows. The background will then consist of all of the non-moving objects in the frame. Almost always though, foreground pixels corresponding to shadows, illumination changes, and moving but uninteresting objects such as grass waving back and forth in the wind should be classified as background. Thus, background subtraction aims to identify meaningful motion in a video sequence. This is a critical first step in a surveillance system, as the subsequent steps such as detection, classification, or tracking depend on accurate results from this step. In a system where the camera remains fixed such as the one used in this research, background subtraction methods have the advantage that the only motion will be by the foreground objects. However, in an outdoor environment such as an apiary, shadows cast by clouds or leaves, and non-foreground motion such as grass blowing in the wind can cause pixel intensities to vary from the background model, leading to false positives if not handled correctly.

3.2   Gaussian Mixture Model

In a single camera system, an applicable assumption is that the scene without any objects in it will exhibit regular behavior that can be described by a model. Objects that come into the scene and move around can then be found by identifying which parts of the image do not fit the model. Background subtraction tasks are largely done in a pixel-by-pixel fashion, and a common approach is to maintain a probability density function for every pixel in the scene. A pixel can then be classified as foreground or background if it is well described by the density function for either foreground or background pixels. Background pixels often have complex distributions, for example a background pixel that may or may not be under light, or a pixel that is affected by periodic motion such as a swaying leaf should still be considered in the background. To handle these cases, the Gaussian mixture model (GMM) is introduced in [18]. A GMM models each pixel of the scene with multiple Gaussian components, each corresponding to either a foreground or a background pixel distribution. When new pixels are observed, they are classified based on which of these components it best fits. This method allows for a high degree of adaptability, because there can be components for different modes of the background, such as being under light or periodically switching colors due to motion from things like swaying leaves. This early method used a fixed number of components for each pixel, even though certain pixels may require more or less, depending on how complex its distribution of values is. This issue is rectified in [19], which describes an algorithm for adapting not only the parameters of the GMM, but also the number of components in the mixture for each pixel. The background subtraction algorithm used in this thesis is the one described in [19].

The value of a pixel at time t in RGB or any other color space is represented by $\vec{x}^{(t)}$. The background subtraction decision, R, of whether the pixel belongs to the background class (BG) or foreground class (FG) is given by 3.1.

$$R = \frac{P(BG|\vec{x}^{(t)})}{P(FG|\vec{x}^{(t)})} = \frac{P(\vec{x}^{(t)}|BG)P(BG)}{P(\vec{x}^{(t)}|FG)P(FG)} \qquad 3.1$$

In general, nothing is known about the foreground objects, how often they will occur, or how large they are. Thus, $P(FG) = P(BG) = 0.5$. $P(BG|\vec{x}^{(t)})$ is referred to as the background model. This model is estimated from a training set X, and the estimated model is denoted by $P(\vec{x}^{(t)}|X, BG)$. In order to adapt to new changes in the scene, the training set is updated over time, adding new samples and removing old ones. For a time period T, the training set at time t is given by $X_T = \{\vec{x}^{(t)}, \dots, \vec{x}^{(t-T)}\}$. Whenever a new sample arrives, the training set is updated and the estimated model $P(\vec{x}^{(t)}|X, BG)$ is reestimated. However, because some samples from the recent history could be from foreground objects, the estimated model is actually $P(\vec{x}^{(t)}|X, BG \vee FG)$. In a GMM with M components, the model is given by 3.2.

$$P(\vec{x}^{(t)}|X, BG + FG) = \sum_{m=1}^{M} \hat{\pi}_m N (\vec{x}; \widehat{\vec{\mu}}_m, \hat{\sigma}_m^2 I) \qquad 3.2$$

where $\widehat{\vec{\mu}}_1, \dots, \widehat{\vec{\mu}}_m$ are estimates of the means and $\hat{\sigma}_1^2, \dots, \hat{\sigma}_m^2$ are estimates of the variances of the Gaussian components. The mixing weights, $\hat{\pi}_1, \dots, \hat{\pi}_M$ are non-negative values that sum to one. Given a new sample $\vec{x}^{(t)}$ at time t, the equations to update the Gaussian components are shown in 3.3, 3.4, and 3.5.

$$\hat{\pi}_m \leftarrow \hat{\pi}_m + \alpha(o_m^{(t)} - \hat{\pi}_m) \qquad 3.3$$

$$\widehat{\vec{\mu}}_m \leftarrow \widehat{\vec{\mu}}_m + o_m^{(t)} \left(\frac{\alpha}{\hat{\pi}_m}\right) \vec{\delta}_m \qquad 3.4$$

$$\hat{\sigma}_m^2 \leftarrow \hat{\sigma}_m^2 + o_m^{(t)} (\frac{\alpha}{\hat{\pi}_m})(\vec{\delta}_m^T \vec{\delta}_m - \hat{\sigma}_m^2) \qquad 3.5$$

where $\vec{\delta}_m = \vec{x}^{(t)} - \widehat{\vec{\mu}}_m$. The learning rate α is used to increase the influence of recent data, and is set to $\alpha = \frac{1}{T}$. For new samples, the ownership $o_m^{(t)}$ is set to 1 for a close component, which has the largest $\hat{\pi}_m$, and is set to 0 for all other components. A sample is considered close to a Gaussian component if the Mahalanobis distance to the component is less than some threshold. If there are no close components, a new one is created with: $\hat{\pi}_{M+1} = \alpha$, $\widehat{\vec{\mu}}_{M+1} = \vec{x}^{(t)}$, and $\hat{\sigma}_{M+1} = \sigma_0$, where σ_0 is some fixed initial variance. If the maximum number of components is already reached, the component with the lowest $\hat{\pi}_m$ is removed. Usually, foreground objects are represented by multiple components with small mixture weights. Thus, the background model is approximated with the largest B components, and shown in 3.6.

$$P\left(\vec{x}^{(t)}\big|X, BG\right) \sim \sum_{m=1}^{B} \hat{\pi}_m N\left(\vec{x}; \widehat{\vec{\mu}}_m, \hat{\sigma}_m^2 I\right) \qquad 3.6$$

The components are sorted by mixture weight in descending to obtain 3.7.

$$B = \arg\min_b \left(\sum_{m=1}^{b} \hat{\pi}_m > \left(1 - c_f\right)\right) \qquad 3.7$$

where c_f is the maximum portion of the data which can be considered foreground without influencing the background model. For example, using 3.3, if $c_f = 0.1$ and $\alpha = 0.001$, the object must remain still for 105 frames before it is considered background.

Zivkovic [19] continues to build on this method, and introduces a way to adaptively select the number of components M for a mixture on a pixel-by-pixel basis. Since the mixture weights π_m describe how much of the data belongs to component m, it can be thought of as the probability that a new sample will come from component m. Thus, the mixture weights define a multinomial distribution. If there are t data samples, each of which belongs to a

component of the GMM, and the number of samples that belong to component m is $n_m = \sum_{i=0}^{T} o_m^{(i)}$, then the multinomial distribution for n_m gives the likelihood function 3.8.

$$L = \prod_{m=1}^{M} \pi_m^{n_m} \qquad 3.8$$

Since the mixing weights must sum to one, the Lagrange multiplier λ is introduced, and the maximum-likelihood estimate is obtained as 3.9.

$$\frac{\partial}{\partial \hat{\pi}_m} \left(\log L + \lambda \left(\left(\sum_{m=1}^{M} \hat{\pi}_m \right) - 1 \right) \right) = 0 \qquad 3.9$$

After eliminating lambda this can be rewritten as 3.10.

$$\hat{\pi}_m^{(t)} = \frac{n_m}{T} = \frac{1}{T} \sum_{i=1}^{T} o_m^{(i)} \qquad 3.10$$

The estimate from T samples, $\hat{\pi}_m^{(T)}$, can be rewritten recursively as a function of the previous estimate, as shown in 3.11.

$$\hat{\pi}_m^{(t)} = \hat{\pi}_m^{(t-1)} + \frac{1}{t} (o_m^{(t)} - \hat{\pi}_m^{(t-1)}) \qquad 3.11$$

The influence of new samples, $\frac{1}{t}$, can be fixed to $\alpha = \frac{1}{T}$, yielding 3.3, the equation for updating $\hat{\pi}_m$. Using $\alpha = \frac{1}{T}$ means that old samples are exponentially downweighted, contributing less as they age. Prior knowledge for the multinomial distribution is introduced using the Dirichlet prior $P = \prod_{m=1}^{M} \pi_m^{c_m}$. The coefficients c_m represent the prior evidence for class m, which is the number of samples that belong to the class a priori. Negative prior evidence $c_m = -c$ is used to ensure that a class m only exists if there is sufficient evidence from the data. The maximum a priori (MAP) solution that follows from 3.9, using the Dirichlet prior is obtained as 3.12.

$$\hat{\pi}_m^{(t)} = \frac{1}{K}\left(\sum_{i=1}^{t} o_m^{(i)} - c\right) \qquad\qquad 3.12$$

where $K = \sum_{m=1}^{M}\left(\sum_{i=1}^{t} o_m^{(t)} - c\right) = t - Mc$. 3.12 can then be rewritten as 3.13.

$$\hat{\pi}_m^{(t)} = \frac{\Pi_m - \frac{c}{t}}{1 - \frac{Mc}{t}} \qquad\qquad 3.13$$

where $\Pi_m = \frac{1}{T}\sum_{i=1}^{T} o_m^{(t)}$ is the Maximum Likelihood estimate from 3.10. After each new

sample, the mixture weights π_m are normalized to sum to one. The first component of a new

GMM is centered on the first sample, and new samples are added as shown above. Using the

Dirichlet prior with negative weights as discussed, components that are not supported by data

will be discarded when their mixture weight becomes negative. The adding of new

components and removal of components that do not model a mode of the background is what

makes this method robust to changes in a dynamic scene.

Chapter 4 - Object Detection

4.1   Introduction

Two methods for object detection are examined in 4.2 and 4.3. The first method is based on examining the blobs of foreground pixels output by the background subtraction step and the second method is learning based, attempting to learn and model the appearance of a honeybee using many examples. Both methods operate on a frame-by-frame basis, producing a list of center points in 2D, each corresponding to a single detection. While the method in 4.3 uses a sliding window to classify each subwindow of the frame as either containing the object or not, it is trivial to go from the coordinates of the four corners to the center point of these corners.

4.2   Blob Analysis

In section 3.2 the background subtraction model used in this thesis was introduced. For each frame, a binary image is generated with 1's corresponding to foreground pixels and 0's corresponding to background pixels. Clusters or blobs of foreground pixels correspond to regions occupied by moving objects. The next step is to convert this information into meaningful object detections. To do so, a simple border following algorithm is used [20]. The binary image is scanned until a pixel (i, j) is found that satisfies the conditions of a border following starting point for either an outer border, 4.1, or an inner border, 4.2.

$$(i, j-1) = 0 \; and \; (i, j) = 1 \qquad\qquad 4.1$$

$$(i, j) >= 1 \; and \; (i, j+1) = 0 \qquad\qquad 4.2$$

16

A unique identifying number N is assigned to any newly found border following starting

point. During scanning the last border encountered, LN, is memorized. LN will either be the

parent border of N, i.e. N will be in a region completely surrounded by LN, or, it will share a

common parent border with N [20]. The border is then followed using the algorithm in [21],

marking each border pixel along the way. This algorithm defines directions 0, 1, 2, and 3,

corresponding to right, up, left, and down, where:

$$right: (i, j + 1)$$

$$up: (i - 1, j)$$

$$left: (i, j - 1)$$

$$down: (i + 1, j)$$

The algorithm then proceeds as follows:

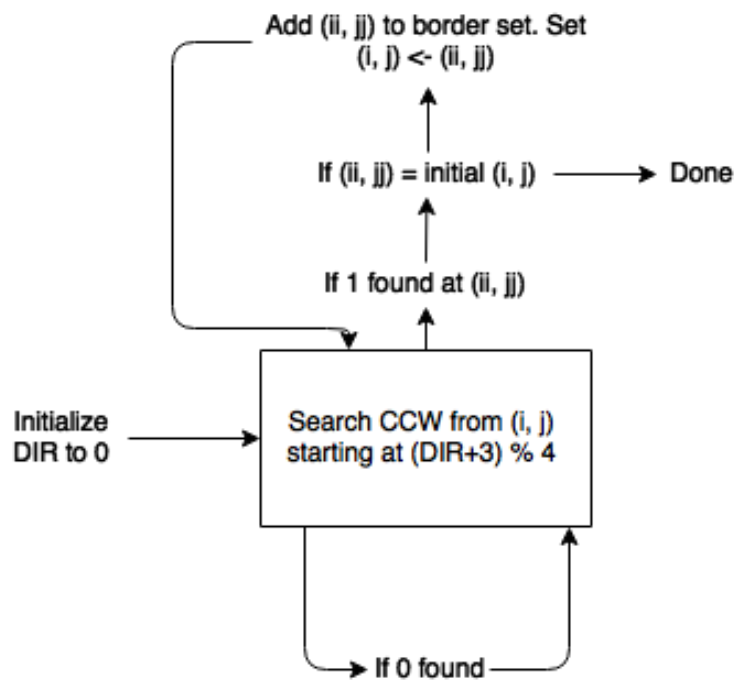

Figure 4.1: Border following algorithm

Each set of border pixels corresponds to a unique blob.

4.3 Cascade Classification

Another popular and powerful object detection algorithm is cascade classification, introduced in [22] for the task of detecting faces. Cascade classification is often an effective algorithm, and one that does not rely on knowledge of the background, meaning the object can be detected no matter what environment is in, unlike detection using blobs. It is a machine learning algorithm that can process video frames at a high rate while remaining accurate. It selects a small number of highly discriminant features from a training set using a variant of the AdaBoost [23] algorithm, and combines increasingly more complex classifiers into a cascade, allowing certain regions of the image to be quickly ruled out, while focusing more time and attention on areas more likely to contain the object of interest. The object detection framework introduced in [22] utilizes three main steps. First, they described an image representation called an *integral image* that allows features to be computed extremely quickly. Second, they show how to construct a classifier by selecting the most important features using the AdaBoost algorithm. Finally, they show how to combine multiple classifiers into a cascade, creating a significantly more complex classifier.

Object detectors, in general, classify images based on the value of features. Detection occurs by examining all subwindows of a larger image, and attempting to classify each of them. The features used are Haar-like features, shown in Figure 4.2. The sum of the pixels under the white rectangles is subtracted from the sum of the pixels under the grey rectangles. The four different types of features are shown in A, B, C, and D.

These rectangular features can be computed extremely quickly using an intermediate image representation that is called the integral image [22].
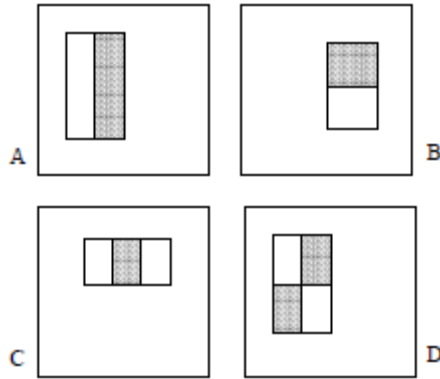
Figure 4.2. Rectangle features relative to detection window

The integral image is a 2D image, which at location (x, y) contains the sum of all pixels above and to the left of (x, y), inclusively, as given by 4.3, where *i* is the original image, and *ii* is the integral image.

$$ii(x, y) = \sum_{x' \leq x, \ y' \leq y} i(x', y') \qquad 4.3$$

Using the integral image, any rectangular region can be summed in four array references, as shown using Figure 4.3.
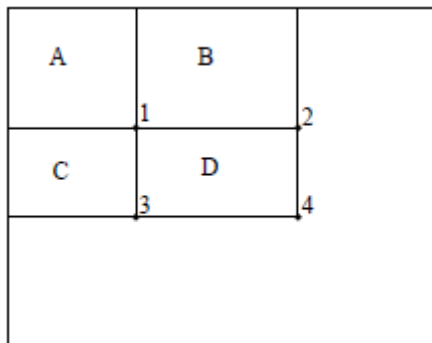

Figure 4.3: Summing pixels in an integral image

The value at location 1 of the integral image, $ii(1)$, is the sum of all pixels within rectangle A. The value at location 2 is the sum of A, plus the sum of B, and location 3 is A + C. The sum of D can be computed quickly as shown in 4.4.

$$sum(D) = ii(4) + ii(1) - (ii(2) + ii(3)) \qquad 4.4$$

19

While these features are simple, the ability to compute a large number of them quickly allows them to remain powerful.

Given a set of positive and negative training examples, and a set of features, any classifier could theoretically be used. The framework described in [22] makes use of the AdaBoost algorithm introduced in [23]. The AdaBoost, or adaptive boosting, algorithm is used to improve the classification performance of a weak learning algorithm. The weak learning algorithm's goal is simply to find the single rectangular feature that best discriminates between positive and negative examples. The AdaBoost algorithm for learning the optimal classifier is summarized as follows:

- Given example images $(x_1, y_1), ..., (x_n, y_n)$ where y is either 1 or 0, for positive and negative examples.

- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 1, 0$, where m and l are the number of negative examples and positive examples.

- For $t = 1, ..., T$:

  1. Normalize the weights:

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$

  2. For every feature, j, train a classifier h_j which uses only the single feature.

     The error of the classifier is given by:

$$\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|.$$

  3. Select the classifier h_t with the smallest error ϵ_t

  4. Update the weights:

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$

where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ if it is classified

correctly, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$

- The final classifier is:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2}\sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where $a_t = \log\frac{1}{\beta_t}$

Given the previous algorithm for creating strong but simple classifiers, the next step

in the object detection framework is to combine these classifiers into a cascade which can

achieve significantly improved detection performance. The main idea is to combine

classifiers such that the simple classifiers can reject as many of the negative subwindows as

possible. The form of the detection process resembles a decision tree, which is called a

cascade. A positive response from the first classifier allows the second classifier to run, and

so on. A negative response from any classifier causes the subwindow to be discarded. In the

process of training multiple classifiers to be part of the cascade, subsequent classifiers are

trained using only the examples that pass the previous stage. This generally means that early

stages reject examples that are clearly negative, while later stages classify examples which

are more likely to be positive.

Chapter 5 - Motion Tracking

5.1 Introduction

Sections 5.2 and 5.3 introduce the two methods that are used for object tracking. Both of these methods attempt to match detections in one frame to detections in the subsequent frame. However, they differ in how long the tracks are maintained for each detection. In 5.2, detections are simply matched from frame to frame, but no history beyond that is maintained. The method introduced in 5.3 differs in that all previous locations for a particular detection are stored until that detection cannot be found.

5.2 Optical Flow

The problem of optical flow is to calculate the motion between two images at time t and $t + \Delta t$, for all of the pixels in the image. This method can also be applied to a local neighborhood of pixels rather than the whole image. Lucas-Kanade optical flow estimation [24] finds a two-dimensional vector (d_x, d_y), that describes the vertical and horizontal motion of the neighborhood of the pixel being considered. This method assumes that the motion of the pixels is constant across the neighborhood, i.e., the image flow vector (d_x, d_y) must apply to all of the pixels in the neighborhood. The Lucas-Kanade method finds an approximate solution for (d_x, d_y) using a least squares fit criterion.

The original algorithm by Lucas and Kanade does not handle large pixel motions well. In the case of tracking bees, which can move large distances in the time between two frames, it is necessary to be able to handle these pixel motions. An extension to the algorithm

that searches the image at multiple scales is presented in [25], where they define the optical flow, or image velocity, $\boldsymbol{d}$, as the vector that minimizes the error in the function as shown in 5.1.

$$\epsilon(\boldsymbol{d}) = \sum_{x=u_x-w_x}^{u_x+w_x} \sum_{y=u_y-w_y}^{u_y+w_y} \left(I(x,y) - J(x + d_x, y + d_y) \right)^2 \qquad \text{5.1}$$

where I and J are greyscale images at time t and Δt, respectively, and w_x, w_y is the size of the image neighborhood, or integration window, and x, y are the pixel coordinates to calculate optical flow on. To be able to handle pixel motions larger than the integration window, a pyramidal image representation is presented. This representation builds multiple images at different scales recursively, starting with the original resolution. The subsequent pyramid levels are simply scaled down by some factor at each level. A practical value for the number of levels is four, so that when starting with an image of 640x480 resolution, the image pyramid levels have resolutions 320x240, 160x120, and 80x60 if scaled down by two in each dimension per level. The pyramidal optical flow algorithm operates as follows: optical flow is first computed at the lowest resolution pyramid level using the standard Lucas-Kanade method. The result obtained from this step is used as an initial guess for computing optical flow at the next pyramid level, where the initial guess is refined. This process is repeated for every level of the pyramid, resulting in a more accurate estimation in the original image.

5.3   Kalman Filter

The second approach for tracking multiple objects is using the Kalman filter [26]. While the Kalman filter has many uses in control, navigation, and other applications with time series data, it can also be applied to the task of object tracking. The Kalman filter

operates on a series of data points in order to estimate the state of the system. In terms of

object tracking, it is operating on multiple observations of an object's location, and

producing an estimate of its state, i.e., its location at the next time step. Incoming

measurements are used to update or correct the predictions. This approach allows the

prediction of an objects future location and it facilitates the association of multiple tracked

objects to their correct tracks. A track is defined as a series of previous locations, and is

associated with a single Kalman filter. It is also associated with either none or a single

tracked object at any given time. The object that is associated with a track may change over

time, but the Kalman filter will not. A track also has an age, which represents the number of

frames for which it has existed, and the count of how many frames it has not been matched

with an object.

Frames in a video sequence occur at discrete points in time, thus, when applying the

Kalman filter to object tracking, measurements occur and state is estimated at these discrete

points. The state of the object tracker, x, developed in this thesis is four dimensional,

representing the x location and velocity, and y location and velocity. In general, the problem

addressed by the Kalman filter is to estimate the state $x \in \Re^n$ of a process that is governed

by the linear stochastic difference equation shown in 5.2.

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \qquad\qquad 5.2$$

with a measurement $z \in \Re^n$ as shown in 5.3.

$$z_k = Hx_k + v_k \qquad\qquad 5.3$$

Random variables w_k and v_k represent the process and measurement noise. The $n \times n$ matrix

A in 5.2 describes the relation of the state at the previous time step $k - 1$ to the current state

in the absence of a driving function. The matrix B describes the optional control input to the

state. The matrix H in 5.3 relates the measurement to the state. The matrices A and H are assumed to be constant [27].

The Kalman filter estimates a process by predicting the process state and then obtaining feedback from an actual measurement of the state. Thus, the equations can be categorized as either *time update* or *measurement update* equations. The time update equations make predictions about the future state given the current state. The measurement update equations combine the *a priori* estimate with the current estimate to obtain an improved *a posteriori* estimate of the state. The time update formulas are given by 5.4 and 5.5.

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \qquad\qquad 5.4$$
$$P_k^- = AP_{k-1}A^T + Q \qquad\qquad 5.5$$

where P_k^- is the *a priori* estimate error covariance, and P_k is the *a posteriori* estimate. Similarly, $\hat{x}_k^-$ denotes the *a priori* estimate of state and Q is the process noise covariance. The measurement update equations are given by 5.6, 5.7, and 5.8.

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \qquad\qquad 5.6$$
$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \qquad\qquad 5.7$$
$$P_k = (I - K_k H)P_k^- \qquad\qquad 5.8$$

where z_k is a measurement at time k, K is the gain, or blending factor that minimizes the a posteriori error covariance equation, and R is the measurement noise covariance.

Chapter 6 - Methodology

6.1   Introduction

Processing a video happens in a step-by-step fashion with each stage passing its output to the next stage. There are four general stages: data acquisition, background subtraction, object detection, and tracking. The object detection and tracking stages are both implemented in two separate ways, but the two methods both take the same input and produce the same output. A general overview of the system in given in Figure 6.1.
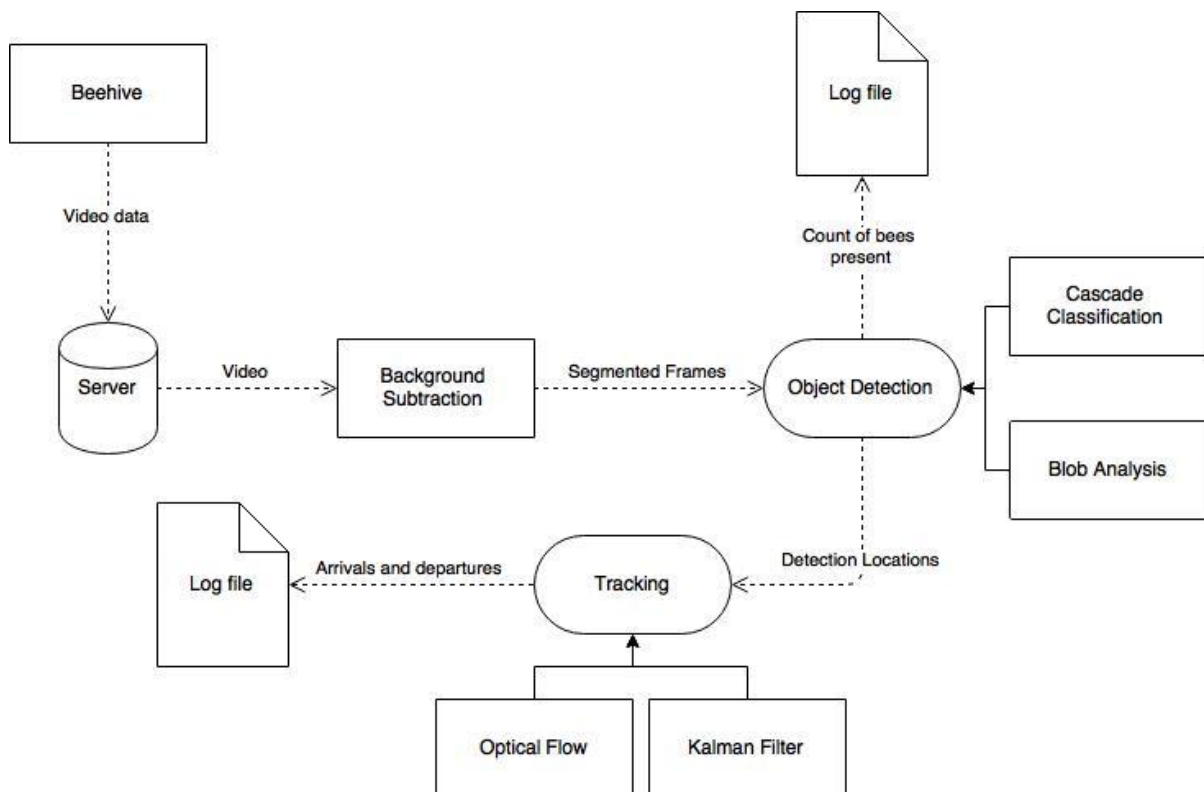

Figure 6.1: Video surveillance system overview

26

6.2   Data Acquisition

Capturing video with appropriate quality is the essential first step in any surveillance system. This section describes the sensing platform that is responsible for recording and managing the videos used in this research. Figure 6.2 shows how the devices are installed. The data acquisition system is placed on the front of the hive above the entrance and records one video channel and two audio channels. A camera module, in the red plastic case, is installed on the device. The camera faces downward toward the entrance of the hive.

Video is captured by the camera in one minute segments at 30 frames/sec and in 640x480 resolution. The minute long videos are saved and encoded as H264. After being compressed, videos are uploaded to a server via Ethernet cable using FTP.

The videos are processed in two different modes. A single video can be downloaded via FTP and stored locally, allowing analyses to be conducted on the same test video. Alternatively, the most recent video stored on the server can be automatically obtained and processed, moving on to the next, more recent video if it is available. Otherwise, the system waits for a new video to be uploaded from the data acquisition system to the server before downloading it and beginning to process it.

Figure 6.2: Data acquisition system installed at hive

6.3 Background Subtraction

The background subtraction method used in this thesis is the Gaussian mixture model (GMM) introduced in 3.2. Background subtraction is an important first step in processing videos, as subsequent detection and tracking steps rely on the background being properly segmented. At each frame of the video, the frame is converted to grayscale and background subtraction is applied. This finds the binary image describing the segmentation of foreground and background, and updates model parameters based on the new information obtained from the frame.

For the experiments carried out, the videos used are minute long clips. If the background model was initialized with the first frame of the video and then continuously updated throughout the video, the first few seconds would contain considerable noise as the model has not fully adapted to the properties of the background yet. In order to simulate the behavior that the system would exhibit if it were running on much longer sequences or live footage, thus making the first few seconds of initialization less meaningful, the background model is initialized with the first 100 frames from a video sequence. That is, the first 100 frames are processed, updating the background model along the way, and then the video is reset to the first frame without reinitializing the background model. This provides the system with ample statistics about how the background pixel values in the scene are distributed.

The GMM is an adaptive model that updates over time. This aspect makes it ideal for an outdoor environment where the lighting changes throughout the day. This model is also robust against periodic changes such as swaying leaves or blades of grass. While being adaptive to dynamic scenes is important, it can also hinder performance in certain scenarios. On occasion, honeybees densely crowd the entrance to the hive. When this happens, the

background becomes occluded for extended periods of time and the model adapts to what it thinks is a new background. Honeybees that cross through this area fail to get detected as foreground objects because they match the characteristics of the learned background. One solution is to use a single reference image of the background when it does not contain bees. This reference image can be used for background subtraction, allowing bees to be detected even if they have been stationary for extended periods of time. However, performing background subtraction in this way forgoes the benefits of an adaptive model- that is robustness to changes in illumination and periodic motion. When choosing a background subtraction model, tradeoffs must be made. The GMM is selected because it provides the best performance under normal operating circumstances.

6.4　Detecting Bees

Object detection is the process of not only determining whether an image contains an object of interest, but also showing where in the image the object is. For this step, the segmentation information from the background subtraction step can be utilized to help improve the accuracy of detection. In this research, the objects to be detected are honeybees. Being able to detect honeybees is of interest for two reasons: it allows them to be counted, and detecting an object is a prerequisite for tracking that object. Counting how many bees are in front of the hive over time gives valuable information about how active the hive is on a given day.

6.4.1　Blob Analysis

Applying background subtraction at every frame provides a binary image. This image is then used to detect bees using the algorithm introduced in 4.2. By following the borders of white regions in the binary image, the regions in the image corresponding to bees are found.

Each region is defined by a set of border pixels at coordinates (i, j). A bounding rectangle for each bee is found by taking the maximum and minimum (i, j) from each set. Using these four points, a center point and area can be calculated for each bee. This process is repeated for each frame in a video sequence, and the center points of the bees are considered the detections for that frame. These detections are passed on to the next stage, which is motion tracking.

An important parameter that impacts how pixels get classified, and thus how effective our blob analysis can be is the threshold that decides if a pixel is well described by a component of the GMM. By examining attributes of the blobs, an informed decision can be made about this threshold. Figure 6.3 and Figure 6.4 characterize how the choice of threshold affects foreground segmentation. These figures show how many detections occurred and the size of the detected areas for different thresholds. In general, it is ideal to minimize noisy detections and maximize legitimate ones. Detections that have a very small or very large area are likely noise. Only detections that are roughly the average size of a bee should be kept. From this information, a threshold of 64 is chosen. This threshold minimizes small detections without throwing out an excessive number of possibly legitimate detections. Additionally, from this information, a minimum and maximum detection size can be defined. A minimum of 200 and maximum of 1500 is selected. As discussed in 6.3, significant noise gets introduced as the hive entrance becomes more crowded. Filtering detections by size helps to reduce the amount of erroneous detections caused by this noise.
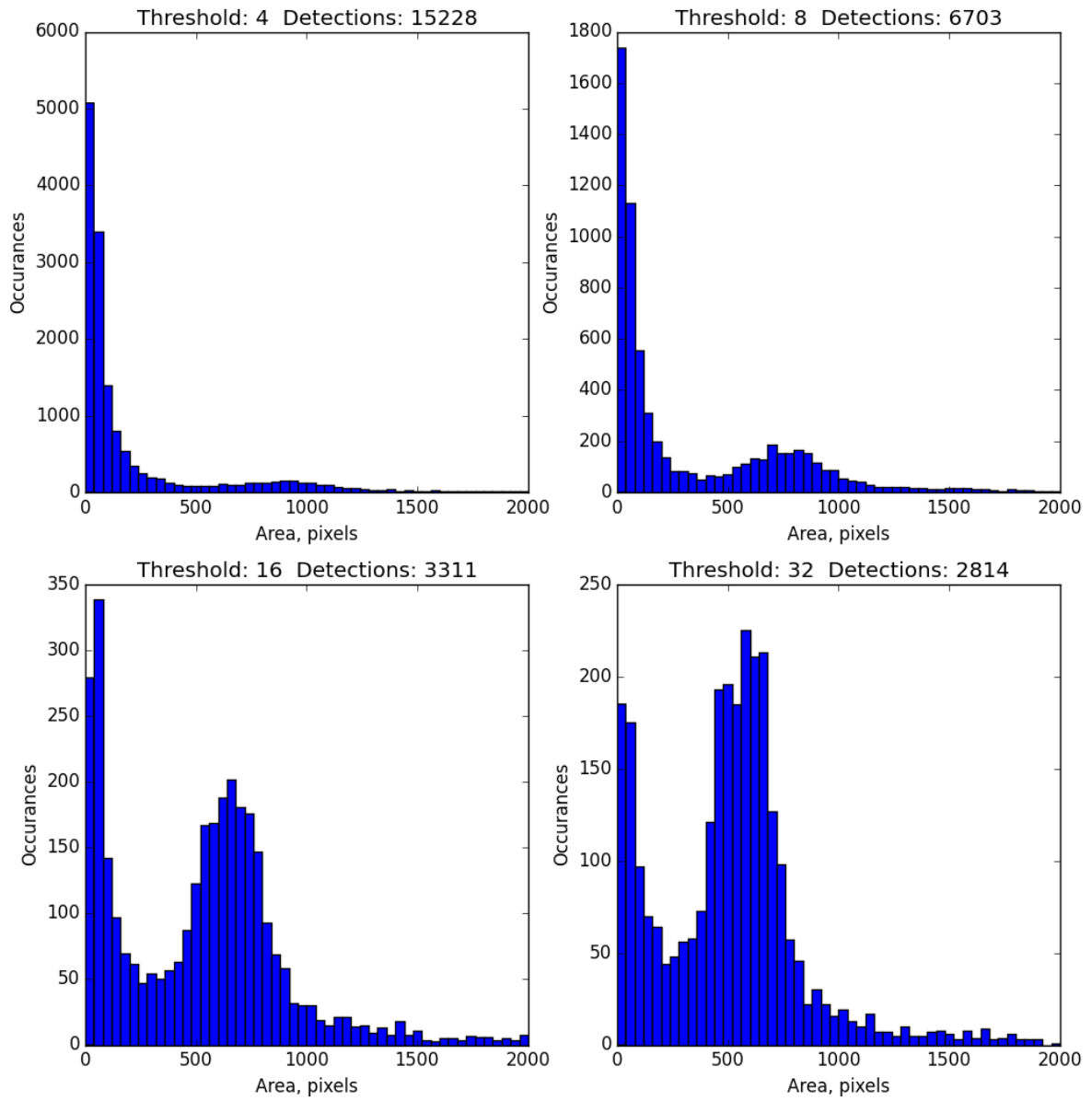
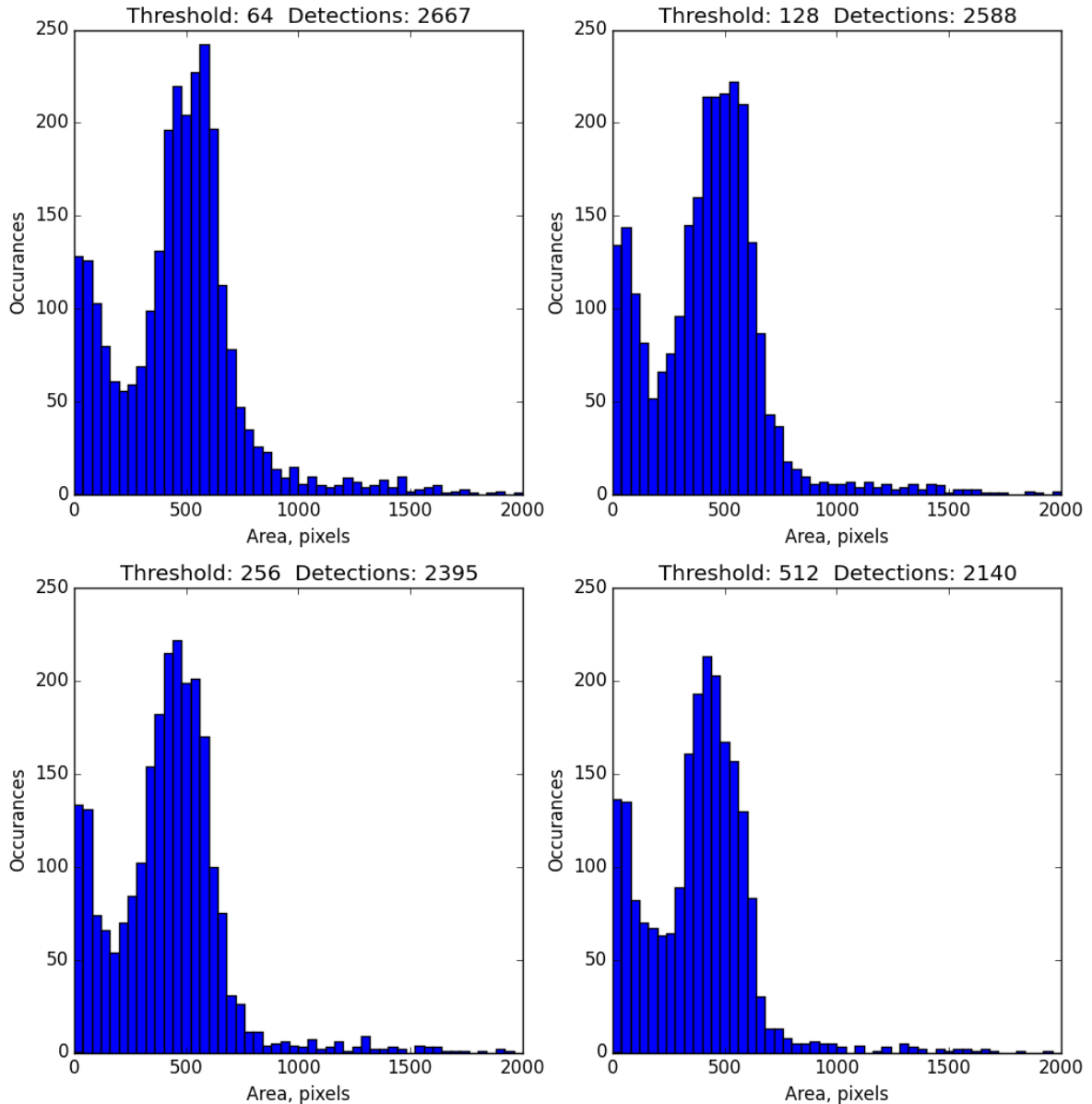Figure 6.3: Areas of foreground regions for thresholds 4, 8, 16, and 32

Figure 6.4: Areas of foreground regions for thresholds 64, 128, 265, and 512

6.4.2 Cascade Classification

The detector used in this thesis is trained using synthetic examples. [28] shows that synthetic, computer generated training examples can be as or more effective than samples prepared by hand. 1000 positive examples are generated by using a single positive sample, and randomly transforming it by rotating it along all of its axes, randomly adjusting its overall pixel intensity, and placing it on top of a subwindow randomly sampled from a set of

images containing no bees. The negative backgrounds are generated by averaging 100 frames of footage for each separate hive, in order to represent all of the backgrounds that are likely to occur. The negative samples for training are obtained in the same way, which is they are randomly sampled subwindows from the set of negative background. Figure 6.6 shows an example image that would be used for sampling negative examples. 2000 negative samples are used for training. Figure 6.5 shows four examples of positive examples used for training. These four examples were all synthetically generated beginning with an image of bee in a vertical position.

With a classifier trained, it is next applied for the task of detection in video frames. Detection is performed using a sliding window algorithm. All rectangular subwindows of the frame are classified as either containing the object or not. The subwindows that generate a positive response from the classifier are considered detections.

The detector described can only detect objects in the pose that it was trained to detect, so bees that are more horizontal, or on the front of the hive and thus perpendicular to plane they are expected to be in will fail to get detected. This can be remedied by training multiple detectors, one for each pose, and applying each of them to the frame. In doing so however, the time it takes to perform detections on a frame increases proportionally to the amount of poses trained, and it becomes prohibitively expensive if real-time performance is desired. Additionally, having multiple detectors increases the false positive rate, as there are more opportunities for a false detection to occur.

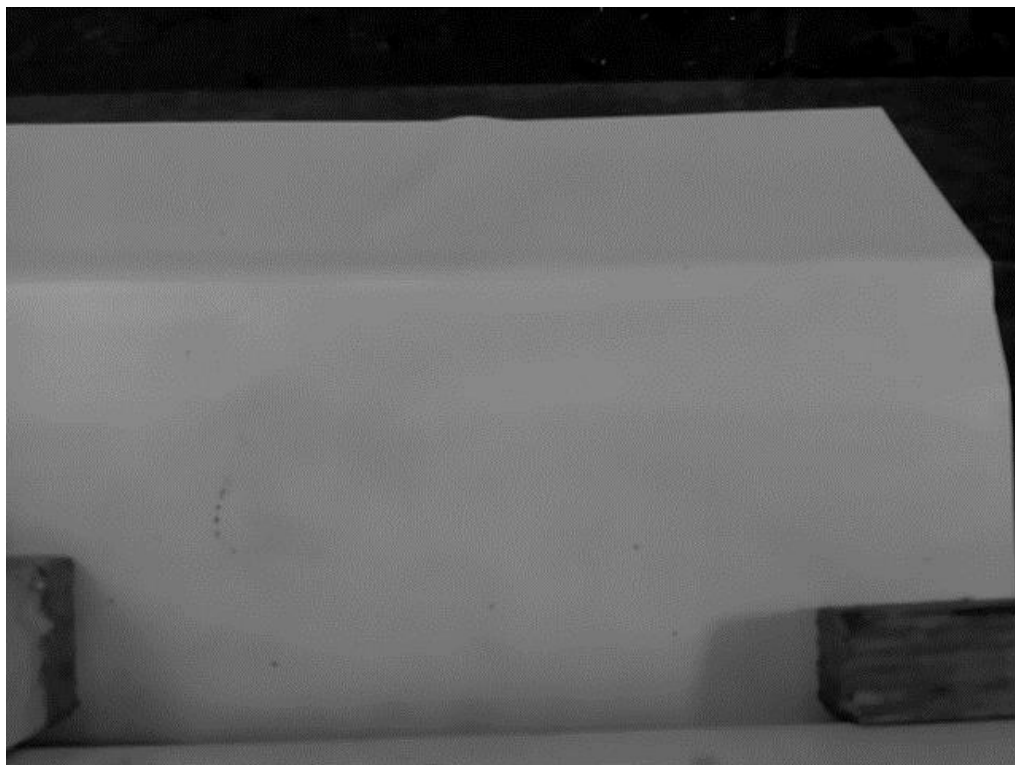Figure 6.5: Positive training examples

Figure 6.6: Negative background

6.5   Tracking Bees

Motion tracking is applied to count the number of bees that enter and leave the hive. In general, a motion tracking algorithm is able to produce a set of motion vectors for a single object over time. In this thesis, the motion is in 2D, and thus this motion vector will describe the horizontal and vertical displacement of an object between two frames. If this motion crosses over a rectangular boundary surrounding the hive entrance it is considered an entry or exit, depending on the direction in which the track crosses the boundary. Figure 6.7 shows the shows the location of the rectangular boundary, drawn in green. In the experiments carried out in this thesis, it was noticed that the size and exact location of this boundary do not largely impact results. The only requirement is that it fully surrounds the entrance of the hive with a small margin between each side of the rectangle and the entrance, where this margin is at least the size of a bee. This allows a bee to be fully detected when it is within the boundary, before it disappears into the hive. In this framework, motion tracking can easily be understood as the problem of associating detections; that is, for every detection in a frame, figure out which previous detection it corresponds to. If there is no correspondence, the object spawned, e.g., left the hive or flew out of the frame.

Figure 6.7: Rectangular boundary surrounding hive entrance

6.5.3 Optical Flow

Object detection using blob analysis is applied to each frame in the video sequence and the center points of the detected objects are obtained. These center points become the pixels under consideration of the optical flow estimation relative to the next frame. That is, detections are stored, and searched for in the next frame in the sequence. The next frame will also have background subtraction applied, resulting in a foreground mask. This foreground mask is a binary image where 1 corresponds to foreground and 0 corresponds to background. The foreground mask is used to modify the frame by changing all background regions to white. Modifying the frame in this way allows the optical flow to be computed more accurately because it excludes regions of the image that do not contain bees, leaving a much smaller set of pixels to search.

A downside of tracking using optical flow is that if a bee crosses the boundary in the span of two frames, it not only must have been detected in the first frame, but it must be correctly matched in the second frame. If it is only detected in one of the two frames, i.e. the frame before it crossed the boundary or the frame after it crossed the boundary, then the fact that it crossed cannot be registered.

6.5.4 Kalman Filter

A Kalman filter is maintained for each tracked object in the video. At each frame, the Kalman filters go through the following steps:

1. Predict the next location of each object using the time update equations.

2. Match each predicted location from 1 with a detected object by minimizing the overall distance between predictions and the locations of detections. If the distance

between any match is greater than a threshold, the detection or prediction is left

unmatched.

3. Update matched tracks by using the measurement update equations. The

   measurement is the 2D location of the object that was matched to the track.

4. Update unmatched tracks by incrementing the time they have been unmatched.

5. Delete tracks that have been unmatched for longer than some threshold.

6. Create new tracks for every object that was not matched to a track. These correspond

   to bees entering the frame.

7. Check if any tracks have crossed the rectangular boundary and increment arrivals and

   departures accordingly. Read the next video frame and repeat from step 1.

Step 2 is the main step in using the Kalman filter to track multiple objects. Predictions that

are made are based on previous locations, so any prediction made should be closer to the

objects' actual location than any other prediction, which would be made based the

trajectories of other objects.

The tracking system using Kalman filters implemented in this thesis maintains tracks

as a list of 2D coordinates. To determine when a bee arrives or departs from the hive, the

current tracks are examined at each frame. The most recent two locations in the tracks history

are checked against the rectangular boundary, and if the two points are not both either inside

or outside the boundary, an arrival or departure is registered, depending on which direction

the points cross the boundary.

One downside to this method arises from the fact that tracks are maintained from

frame to frame. If a bee is being tracked, and it leaves the frame at the same time that another

bee enters the frame, the new bee could be incorrectly associated with the track of the leaving

bee, if it is sufficiently close to the track. This causes a problem when the mismatch occurs between two points on either side of the rectangular boundary, because when a track crosses this boundary an arrival or departure is registered.

Chapter 7 - Results

7.1   Background Subtraction

Background subtraction was applied as the first step in the surveillance system. For a given frame, the system determines the foreground mask, a binary image that describes which pixels correspond to the foreground. Figure 7.1 and Figure 7.2 show two example frames taken from the same camera at different times of day. Figure 7.1 shows the output of the background subtraction step when the scene has a mostly constant illumination and is not crowded with bees. It can be seen in the foreground mask that the four visible bees are properly classified as foreground object. Figure 7.2 shows the same scene at a different time of day, when moving shadows are present. Moving shadows can cause false positives when using a less robust background subtraction algorithm, but as shown, the regions affected by the shadow are properly classified as background pixels.

Figure 7.1: Sample frame and foreground mask

Figure 7.2: Sample frame and foreground mask

A more challenging situation arises when the bees heavily crowd the entrance to the hive. When the intensity of a pixel is affected either by a stationary bee, or multiple bees repeatedly covering the pixel, the adaptive background model will eventually learn new background modes corresponding to bees. During these scenarios, a pixel may be covered by a foreground object longer than it is a background pixel. When the background is finally uncovered, it may be considered foreground. Figure 7.3 shows an example of this occurring. There is significantly more noise around the entrance to the hive than in Figure 7.1 or Figure 7.2, but because the background model operates on a pixel by pixel basis, areas where the bees remain more separated, and move about more quickly, can still be effectively segmented.

Figure 7.3: Sample frame and foreground mask

7.2   Bee Detection

The second step of the surveillance system is to detect bees in the image. Two

methods for detection were introduced in 4.2 and 4.3, and the results of those algorithms are

presented in this section. The performance of both detectors is characterized by the precision-

recall curve. Precision or positive predictive value (PPV) is given by 7.1. Recall or true

positive rate (TPR) is given by7.2.

$$Precision = PPV = \frac{TP}{TP + FP} \tag{7.1}$$

$$Recall = TPR = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{7.2}$$

Where TP is the number of true positives, FP is the number of false positives, TN is the true

negatives, and FN is false negatives. P is the total number of positive examples present in a

test set. Precision and recall are both area based, which considers each pixel within positive

regions of the ground truth as positive examples, and vice versa [29]. It is shown in [30] that

precision-recall is a better indicator of performance than the receiver operator curve (ROC) for

tasks with largely skewed class distributions. Because bees are small and occupy a small

percent of the image, the distribution of classes considered for detection is in fact skewed in

favor of the background.

7.2.5 Blob Analysis

`   Blob analysis is used to detect bees in a segmented binary image. Figure 7.4 shows

the result of this operation. The blobs produced by the background subtraction are analyzed

by following their borders, which are drawn in green. From this set of points, a bounding

rectangle is found and is drawn in red. The detector performance for an annotated test

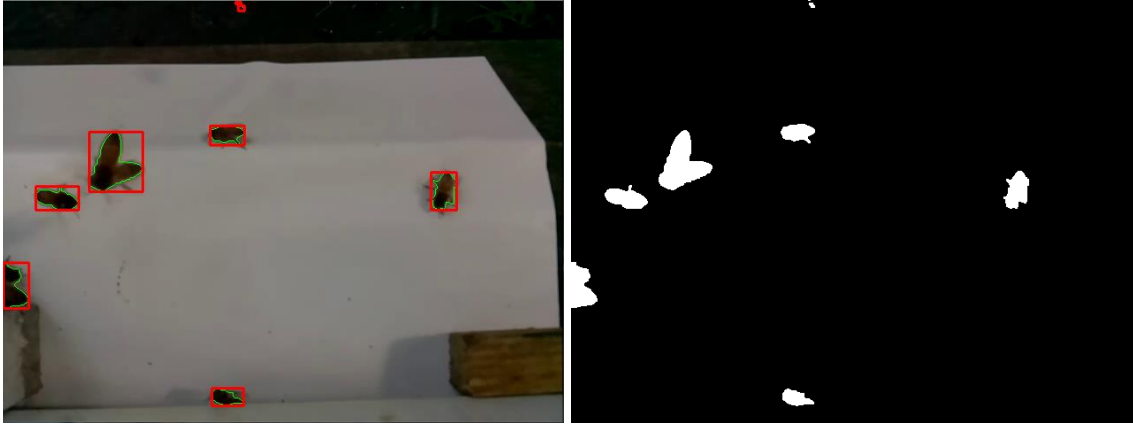sequence is characterized by Figure 7.5.

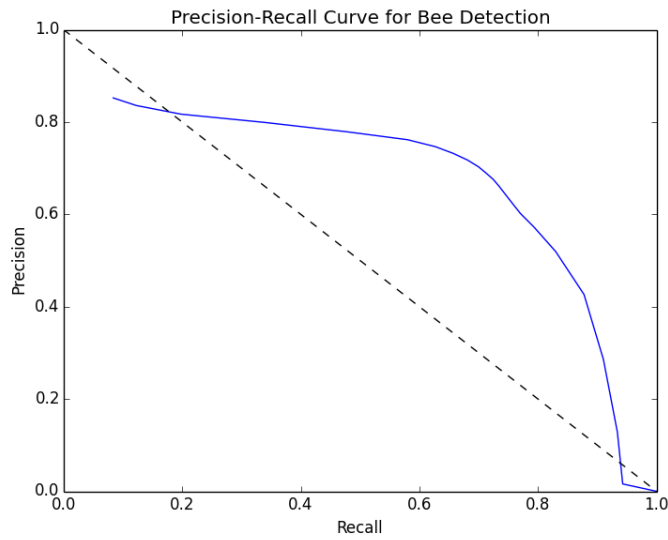Figure 7.4: Sample frame and blob analysis

Figure 7.5: Blob detection performance

7.2.6 Cascade Classification

Cascade classification is the second method that is used to detect bees. Figure 7.6 shows an example frame with the detector output. It can be seen that the detector tends to produce large detection regions that do not tightly bound the bee. It also fails to detect several of the bees in the image, and produces a false positive. Figure 7.7 characterizes the performance across an annotated test sequence.
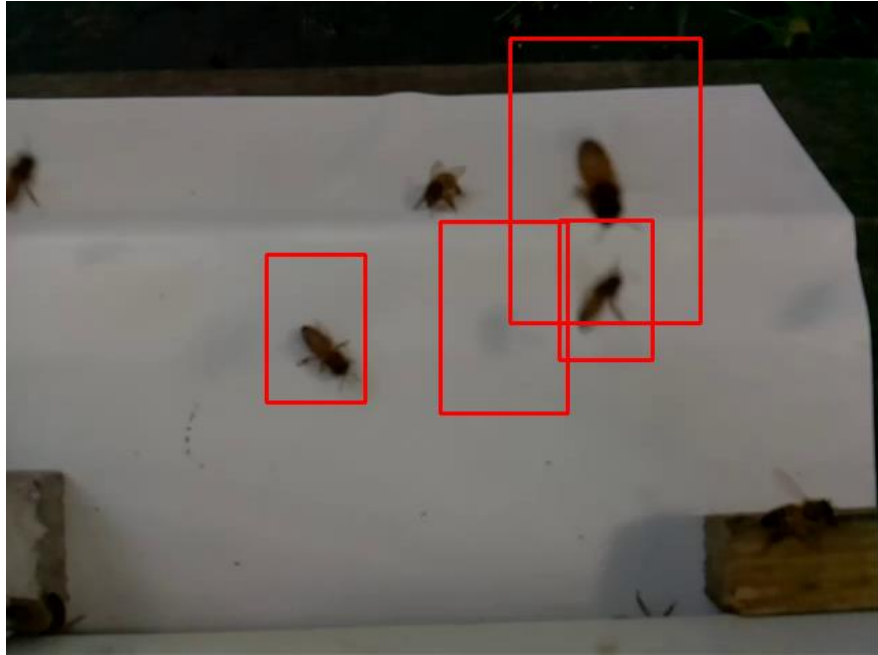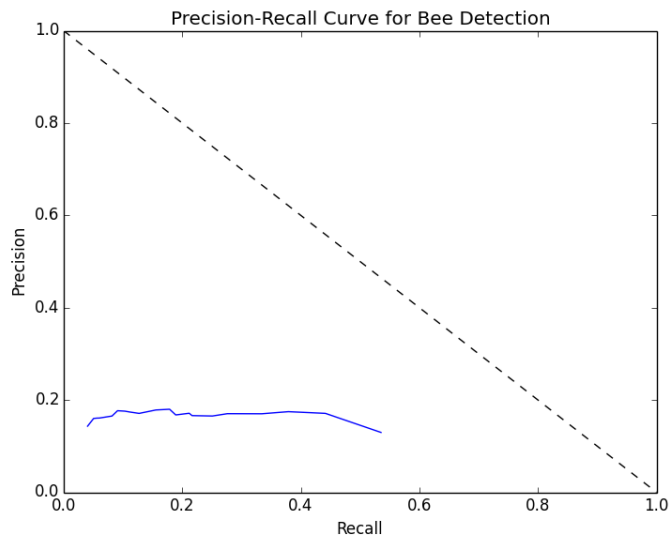
Figure 7.6: Detection using cascade classification

Figure 7.7: Cascade classification performance

7.3 Motion Tracking

In this section, results from the tracking systems described in 5.2 and 5.3 are

presented. Both methods are tested on the same two test sequences, *noshadow3pm* and

shadow2pm. The first sequence is considered easier, as it has a constant illumination

throughout the entire sequence. The second sequence contains multiple changes in illumination caused by shadows.

7.3.7 Optical Flow

Optical Flow is the first method used for object tracking. For each frame, optical flow estimation is used to match detections from the previous frame to detections in the current frame. A sample result is shown in Figure 7.8. Red dots are drawn to indicate the locations of detections from the previous frame, and these are the points that are searched for in the current frame. If a match is found and it crosses over the boundary, a red line is drawn and an arrival or departure is registered.

Figure 7.8: Tracking result using optical flow

Optical flow is used to determine how many bees are arriving and departing from the hive over time. To analyze the performance in this regard, two sample videos are manually annotated with the actual number of arrivals and departures, and compared with the system's estimate.

Table 7.1 and Table 7.2 show tracking results on two test sequences when using the optical flow based tracking algorithm.

Table 7.1: Optical flow tracking results from noshadow3pm sequence

	Ground Truth	System Estimate	Percent Error
Arrivals	75	55	26.7%
Departures	63	63	0.0%

Table 7.2: Optical flow tracking results from shadow2pm sequence

	Ground Truth	System Estimate	Percent Error
Arrivals	50	25	50.0%
Departures	64	55	17.2%

7.3.8 Kalman Filter

Tracking based on Kalman filtering is the second way in which bees are tracked over time. Figure 7.9, Figure 7.10, and Figure 7.11 show a graphical view of tracking results in a real world environment. In these video frames, the current frame is shown with system output drawn on top. A detected bee is outlined with a blue ellipse. In the case of Kalman filter based tracking, a single bee is associated with a single track, and thus can be uniquely identified. These bees are individually identified by a green number. The detected bees have their tracks drawn in green and red behind them. The green track is the trajectory that was predicted by the Kalman filter, and the red track corresponds to the actual locations the bee was detected at. The green rectangle around the entrance of the hive is again the region the bee must enter or leave to be considered entering or leaving the hive.

Figure 7.9 shows system output when two bees are present. There is a slow moving bee, 63, and a faster moving one, 67. It can be seen that the slow moving bee is tracked over a long period of time, and with high accuracy. The faster moving bee moves much more chaotically, and moves further per frame, which is why the predicted trajectory has a harder time accurately predicting where it will be next. However, it is still correctly associated with

its track over multiple frames. Figure 7.10 shows a similar scenario, except both bees are moving quickly, and it is also under different lighting conditions. Figure 7.11 shows a challenging example where six bees are being tracked, two of which have overlapping trajectories. In Figure 7.11, the green predicted trajectories are turned off to make the image less cluttered.
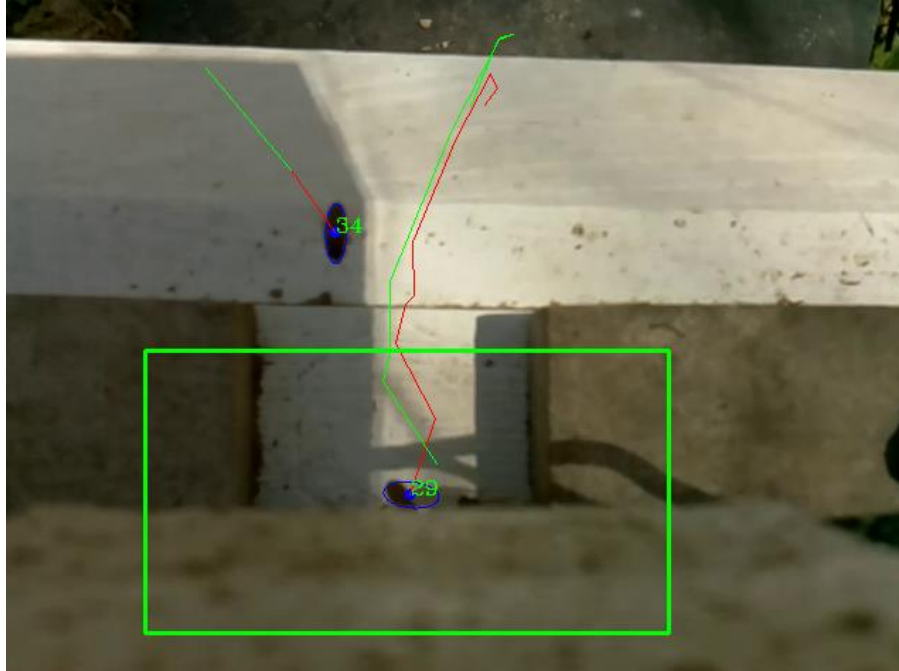

Figure 7.9: Tracking two bees

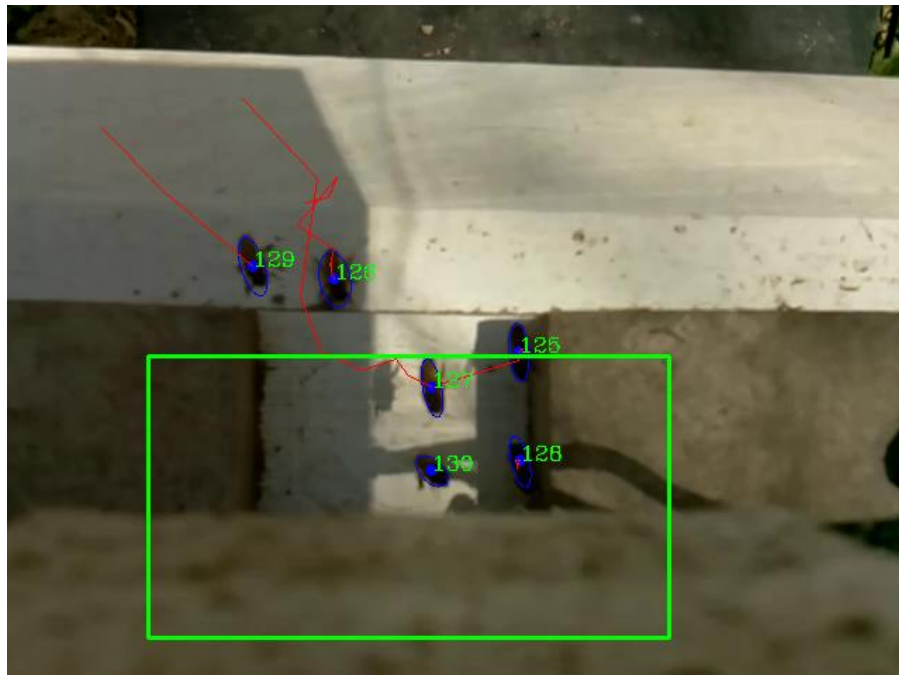Figure 7.10: Tracking two bees in varying illumination environment

Figure 7.11: Tracking multiple bees in varying illumination environment

Table 7.3 and Table 7.4 describes the results on the two test sequences using the Kalman filter tracking algorithm. For this experiment the blob detection algorithm was used, and the rectangular boundary is the same one that was used in 7.3.7.

Table 7.3: Kalman filter tracking results from noshadow3pm sequence

	Ground Truth	System Estimate	Percent Error
Arrivals	75	78	4.0%
Departures	63	60	4.8%

Table 7.4: Kalman filter tracking results from shadow2pm sequence

	Ground Truth	System Estimate	Percent Error
Arrivals	50	60	20.0%
Departures	64	52	18.8%

Chapter 8 - Conclusion

8.1   Background Subtraction

Background subtraction is the first stage of processing in the surveillance system. This stage is important for filtering out irrelevant information and making subsequent stages more effective. It was shown in 7.1 that under normal circumstances the bees are sparsely distributed in the video frame and do not remain stationary for very long. Due to this, the image can be properly segmented. However, as the hive entrance becomes more crowded, background subtraction begins to fail.

8.2   Object Detection

Object detection was implemented using two different algorithms. Detection performance for both was characterized using the precision-recall curve in Figure 7.5 and Figure 7.7. It can be seen that detection using blob analysis vastly outperformed cascade classification on the test sequences. In the case of cascade classification, adding more classifiers trained on different poses increases the recall by allowing the system to detect more positive examples and does not significantly affect the precision.

8.3   Motion Tracking

Two different methods for motion tracking were implemented for the purpose of counting arrivals and departures from the hive. Table 7.1 and Table 7.2 presented the results using optical flow. For both test sequences, tracking with optical flow significantly undercounted arrivals. It demonstrated inconsistent performance when counting departures,

49

estimating the correct amount for one test sequence and greatly undercounting departures on the other.

Table 7.3 and Table 7.4 summarized the tracking performance when using the Kalman filter algorithm. For the first test sequence, the system slightly undercounted both arrivals and departures. For both arrivals and departures, the percent error from the actual value was similar, at 4.0% and 4.8%. In the second test sequence, the system erred 20.0% and 18.8% on estimating arrivals and departures. When comparing the two methods, it is useful to note that tracking using the Kalman filter tended to produce more consistent results. That is, its most accurate estimate erred by 4.0%, and its least accurate estimate erred by 20.0%. In contrast to this, tracking using optical flow produce a minimum error of 0.0% percent, but a maximum error of 50.0%

8.4 Future Work

The visual surveillance system presented in this thesis is easy to install, non-intrusive, and shows promising results in real-world scenarios. However, it does suffer from some drawbacks. When bees crowd the entrance to the hive, background subtraction can fail. One way to solve this is to use a static reference image of the background containing no bees. Future work could examine combining this reference image with the GMM to improve background subtraction results in situations where the hive entrance becomes densely crowded.

Bee detection using blob analysis effectively identifies regions of the image containing bees. This method can fail when multiple bees are close enough to one another to be considered a single blob, however. Future research could explore methods for splitting large detections into multiple smaller detections. Additionally, a detector that does not rely

50

on knowing background information could be used in this circumstance. The cascade classifier can handle this situation, but does not perform well enough to be useful.

The speed that honeybees move at makes tracking a difficult problem. Tracking using both the Kalman filter and optical flow provide promising results, but there is room for improvement. The main cause for error in tracking is track aliasing, which is when a track becomes associated with a different object. This can occur when a bee leaves the frame at the same time a bee enters the frame. In this scenario, the track may become associated with the new bee. If the videos could be recorded at a higher frame rate, the bees would not be able to move as far in a single frame and thus the tracking algorithm would not have to search as far for an association. Searching a small radius for track associations would reduce the amount of track aliasing that occurs.

Bibliography

[1] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man, and Cybernetics*, 34:334–352, 2004.

[2] M. H. Struye, H. J. Mortier, G. Arnold, C. Miniggio, and R. Borneck. Microprocessor-controlled monitoring of honeybee flight activity at the hive entrance. *Apidologie*, 25(4):384–395, 1994.

[3] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, and S. Russell. Towards robust automatic traffic scene analysis in real-time. In *Proceedings of the 33rd IEEE Conference on Decision and Control,* 1994.

[4] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *IEEE Computer Science Society Conference on Computer Vision and Pattern Recognition,* 1999.

[5] C. Stauffer and W. E. L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, 2000.

[6] A. J. Lipton, H. Fujiyoshi, and R. S. Patil. Moving target classification and tracking from real-time video. In *Proceedings of Applications of Computer Vision,* 1998.

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Computer Science Society Conference on Computer Vision and Pattern Recognition,* 2005.

[8] S. W. Ha and Y. H. Moon. Multiple object tracking using SIFT features and location matching. *Int. J. Smart Home*, 5(4):17–26, 2011.

[9] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua. Multiple object tracking using k-shortest paths optimization. In *IEEE Transactions On Pattern Anal. Mach. Intell*, 2011.

[10] J. Black, T. Ellis, and P. Rosin. Multi view image surveillance and tracking. In *Workshop on Motion and Video Computing,* 2002.

[11] A. E. Lundie. *The Flight Activities of the Honeybee.* United States Department of Agriculture, 1925.

[12] T. Kimura, H. Ikeno, R. Okada, and E. Ito. A study of identification and behavioral tracking of honeybees in the observation hive using vector quantization method. In *Conference on methods and techniques in behavioral research*, 2008.

[13] T. Kimura, M. Ohashi, R. Okada, and H. Ikeno. A new approach for the simultaneous tracking of multiple honeybees for analysis of hive behavior. *Apidologie*, 42(5):607-617, 2011.

[14] C. Chen, E.-C. Yang, J.-A. Jiang, and T.-T. Lin. An imaging system for monitoring the in-and-out activity of honey bees. *Comput. Electron. Agric.*, 89:100–109, 2012.

[15] J. Salas and P. Vera. Counting the Bumblebees Entering and Leaving a Beehive.

[16] J. Campbell, L. Mummert, and R. Sukthankar. Video monitoring of honey bee colonies at the hive entrance. *Vis. Obs. Anal. Anim. Insect Behav. ICPR*, 8:1–4, 2008.

[17] A. Ghadiri. Implementation of an Automated Image Processing System for Observing the Activites of Honey Bees. Appalachian State University, 2013.

[18] N. Friedman and S. Russell. Image Segmentation in Video Sequences: A Probabilistic Approach. In *Proceedings of The Thirteenth Conf. on Uncertainty in Artificial Intelligence*, 1997.

[19] Z. Zivkovic.Improved adaptive Gaussian mixture model for background subtraction. In *Proceedings of the 17th International Conference on Pattern Recognition*, 2004.

[20] S. Suzuki. Topological Strucutal Analysis of Digitized Binary Images by Border Following. *Comput. Vis. Graph. Image Process.*, 30:32–46, 1985.

[21] A. Rosenfeld. Connectivity in Digital Pictures. *Journal of the ACM*, 17(1):146–160, 1970.

[22] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.

[23] Y. Freund and R. E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. *Computational learning theory*, 1995.

[24] B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence.* San Francisco, CA, USA, 1981.

[25] J.-Y. Bouguet. Pyramidal Implementation of the Lucas Kanade Feature Tracker.

[26] R. Kalman. A New Approach to Linear Filtering and Prediction Problems," *Trans. ASME*, 83:35–45, 1960.

[27] G. Welch and G. Bishop. An Introduction to the Kalman Filter. In *SIGGRAPH*, Los Angeles, CA, USA, 2001.

[28] C. Zhang, J. C. Platt, and P. A. Viola. Multiple instance boosting for object detection. In *Advances in neural information processing systems*, 2005.

[29] V. Y. Mariano, J. Min, J.-H. Park, R. Kasturi, D. Mihalcik, H. Li, D. Doermann, and T. Drayer. Performance evaluation of object detection algorithms. In *Proceedings of the 16th International Conference on Pattern Recognition*, 2002.

[30] J. Davis and M. Goadrich. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine learning*, 2006.

Appendix

The source code for the system developed in this thesis is included on the enclosed CD. Following is a list of the files that appear on this CD:

1. analysis: Python package containing code for comparing results to ground truth

    a. class_counter.py: Counts the number of foreground and background pixels in a binary image

    b. compare2truth.py: Calculates precision and recall for a given image and its ground truth image

    c. performance_eval.py: Processes a test video with varying system parameters and plots the precision-recall curve that is generated

2. classifier: Contains pre-trained cascade classifiers for use with OpenCV's CascadeClassifier class

3. samples: Contains positive and negative classifier training examples

4. training: Source code for training the cascade classifier

    a. create_samples.py: Generates synthetic training examples using a single training example

    b. generate_negative.py: Creates a negative example image from an input video

    c. split_frames.py: Splits an input video into frames at a given interval

    d. train_cascade.py: Trains the cascade classifier from positive and negative examples

5. source: Source code for the surveillance system

    a. area_histograms.py: Computes histograms describing the size of detections in a given test video

    b. background_subtractor.py: Implements the GMM based background subtractor which maintains information about the adaptive background model

    c. bee_parser.py: Tool for annotating video frames with actual bee locations

    d. bgsub_mog.py: Script that processes a test video with both detection algorithms to generate precision and recall data

    e. drawing.py: Contains utility functions for drawing tracks and other information on video frames

    f. kalman_track.py: Implements Kalman filter based tracking for an input video and outputs the number of arrivals and departures that occurred

    g. keys.py: Maps keyboard codes to symbolic names

    h. live_mode.py: Implements live processing, retrieving and processing the most recent video available on the server

    i. lk_track.py: Implements tracking using Lucas-Kanade optical flow estimation

    j. tools.py: Contains various utility functions for initialization, morphological operations, checking for boundary crosses, and computing optimal track assignment

    k. track.py: Implements a Track class which maintains information about a tracked object

    l. video_writer.py: Writes processed video frames to a new video file

6. GROUND_TRUTHS.txt: Contains arrival and departure ground truths for test videos

7. Videos: Directory of test videos

    a. whitebg.h264

    b. newhive_noshadow3pm.h264

    c. newhive_shadow2pm.h264

Vita

David Kale was born in 1991, in Greensboro, North Carolina to Judy and Joel Kale. After spending a few short years in Florida and Georgia, he was raised in Winston Salem, NC. When he finished high school, he wasn't sure of his field of study in college, but growing up tinkering with computers at home influenced him to try Computer Science and he has never looked back. He graduated from Appalachian State University with a Bachelor of Science in Computer Science and proceeded into graduate studies, also at Appalachian State University. During this time he worked as a teaching assistant, aiding in CS1440 and CS2440 labs. In August 2015, after two years of rigorous study he received the Master of Science degree in Computer Science.