The Use of Video to Detect and Measure Pollen on Bees Entering a Hive

Cheng (Robert) Yang

A thesis submitted to Auckland University of Technology
in fulfilment of the requirements for the degree of PhD

2017

School of Engineering,
Computer & Mathematical Sciences
Primary Supervisor: John Collins

Abstract

This research will measure the pollen being brought into the beehive in the bees' pollen sacs. It will process 2-D video which is obtained from in front of the entrance to the beehive and automatically count the number of full pollen sacs which bees bring back. This can help bee keepers to check forage inside beehive without opening the hive manually. The technology used in this research relates to object detection using computer vision. Many papers in the field of object detection try to detect and track different objects, such as human beings, vehicles and animals. However, very little research has focused on the flight of bees, and few has involved identifying the pollen sacs. Difficulty arises from requiring high resolution video and high speed processing because bees can fly at high speed.

The first step of the procedure for this research is to detect and track bees. The bee detection method is to combine foreground subtraction and colour thresholding. The tracking approach mainly uses the Kalman filter. After that, the Hough transform is to solve the problem of tracking bees in the occlusion situation.

The second step is the pollen detection and measurement. There are two methods to complete this mission in this research. The first method is to use image processing and statistics to analyse individual bee images. The bee's body is analysed to identify the main body (head and abdomen) and other parts (wings, legs and pollen sacs). Then the pollen sacs can be detected using the features of colour, size, blob orientation, blob ellipticity, position and blob extent. The thresholds to distinguish pollen and non-pollen blobs are estimated using the receiver operating characteristic (ROC). The amount of pollen is estimated by counting the bees with full pollen sacs. The second method is to use deep learning method. 1000 individual bee images with pollen sacs labels are collect for training a deep neural network. After the training, a network can be used to detect pollen on individual bee images automatically. Then the individual bee images can be identified as pollen or non-pollen images. This identification is combined with bee tracking on bee monitoring video to count the number of pollen carrying bees.

This thesis explains the detail of the theory of the methods used in this research. In addition, it reports the results of applying the methods in practice. The experimental results indicate the tracking of single bees is over 99% accurate. More than 80% of bees in a merged situation are tracked successfully. In pollen detection and measurement test, the difference (or error) between measured number and actual number of pollen sacs is about 7%, if the deep learning method is applied. This is a significant improvement from the image processing and statistics model, which produces 33% error.

Table of Contents

List of Figures

List of Tables

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been accepted for the award of any other degree or diploma of a university or other institution of higher learning.

Signed_____杨成 1/12/2017___

Acknowledgements

This piece of work is the result of hard work, patience, sacrifice and the unconditional support of many people. I wish to thank everyone who has supported and helped me in completing this research.

Firstly, I would like to acknowledge my supervisors: John Collins, Mark Beckerleg and Tek Tjing Lie, for their support and encouragement in this thesis. I would like to thank them for the guidance they have given me throughout this period.

I also acknowledge Darren Bainbridge for his important role providing access to beehives for this research.

I express my honest thanks to the AUT library staff for their overall support for the entire period of my study at AUT.

I would like to thank my friends and family for their prayers and encouragement, which motivated me to complete this thesis successfully.

Above all, I thank the Almighty God for helping me accomplish this venture with ease.

Ethics Approval

There was no ethics approval requirement for this thesis, because the research only observes bees without any interference to their behaviour.

Confidential Material

There is no confidential material in this thesis.

Chapter 1

Chapter 1: Introduction

This research aims to use computer vision to help bee keepers to monitor the health of bees, especially concerning foraging for food. This chapter firstly introduces the background of bee breeding and health to show the significance of this research. Then the research gap will be identified which is the computer vision for bees. Finally, the structure of this thesis is presented in this chapter.

1.1 Introduction of this research

This research applies computer vision technology to detect pollen sacs (very small object) from flying honey bees' bodies, and also measure the number of pollen sacs being brought to beehive. Computer vision has been applied to detect and track objects on a stream of video in different environments. For example, it is often used for safety and security. In recent years, robust, real-time and model-based detection and tracking techniques have been developed for rigid and non-rigid moving objects. They enable automated monitoring and event detection. Most contributions are using computer vision to deal with detection and tracking in an urban environment such as cars and pedestrians. These object detections can be carried out by either stationary or moving cameras (Aycard et al., 2006, Ikoma et al., 2014).

This research extends the application of computer vision to insect detection and tracking, particularly for bee detection and tracking. In addition, this research uses computer vision to detect very small pollen sacs on flying bees. In this research, the video is two dimensions with resolution of 1920×1080. It can display the bees' bodies clearly enough to show pollen sacs on the back legs of bees. However, the pollen sacs are still very small, which is a challenge for computer vision detecting them. Therefore, bees should be detected first, then pollen sacs can be detected from individual bee images.

Bees can be detected on the video with their special orange colour. Conversely, bees' bodies have not only orange colour, but also black colour. In addition, motion detection can also be used for bee detection, because only bees are flying on the video. With a simplified background (single colour background), the colour and motion are combined to detect bees on bee monitoring video.

After bee detection, pollen sacs can be detected on the individual bee images. The bees' images can be analysed to find the structure of bee bodies, such as main body of head and abdomen, wings, legs and pollen sacs. The abdomen of main body may have similar colour to pollen sacs.

The main body should be removed, then colours of pollen sacs (orange, yellow and white) are used for pollen detection. Because the pollen sacs are very small objects, the result of detection will be very noisy. To improve the result, the pollen sacs should be analysed to find other features, such as area (number of pixels), shape and position relating to bee body. These features are used to separate pollen sacs from image noise. This research shows how well computer vision can perform for pollen detection and measurement.

A second method for pollen detection is deep learning algorithm. Deep learning mainly uses deep neural network for object detection. In this research, the training data set includes about 1000 individual bee images with pollen sacs labelled. The neural network is trained by using the image data. After the training, a network model can be used to automatically detect pollen sacs on new individual bee images. The deep learning methods have been tested by many researches using many image data sets. However, these image data sets have very clear objects and very high resolution for the objects. In this research, the images are from bee monitoring videos. They include motion blur and the resolution of pollen sacs is small (from 15×15 to 40×40). Therefore, this is a challenge for deep learning detecting pollen sacs. This research will show how well the deep learning can perform for pollen detection and measurement.

For the measurement of pollen sacs being brought to beehive, the tracking of bees is required to avoid double counting. If bees can be tracked on bee monitoring video, the measurement of pollen sacs is to count the number of pollen carrying bees appearing on the video. Bees can be tracked by simply predicting their position on the next video frame. The prediction can be calculated from the positions of previous frames. In addition, the predicted position can be used by some algorithms, such as Kalman filter and particle filter. This research also uses Kalman filter to track bees on bee monitoring video. However, if bees are flying across each other (bees' occlusion) on the video, the bee detection cannot detect each bee's position. Finding bee positions in this occlusion situation is a challenge. This research introduces a special application of Hough transform to solve this problem. Hough transform uses bee shapes just one frame before the occlusion to find bee positions in the occlusion situation. This research will show how well this tracking technique performs in the experimental test.

In summary, this research creates a model for the 2-D video to automatically detect pollen sacs on honey bees, and estimate the number of pollen sacs being taken to beehive. It uses computer vision techniques to complete the mission.

1.2 significance of this research

The number of pollen sacs relates to bees' forage. Forage is important for the bees' health and honey production. Bee health is very important in the world, because they are the main insects for crop pollination. Reduction of the number of bees may reduce the pollination of crops, and then food production may be reduced. More detail of the bees' important role is introduced in section 1.5. Honey production is also important. It is part of the income of bee keepers, and honey is used in different food production. Especially, New Zealand produces a special manukau honey. Therefore, bee forage is very important in New Zealand.

Normally, bee keepers check the forage of bees by manually opening the beehive. This is laborious job for a farm which has hundreds or thousands of beehives. This research aims to build a model which can automatically count pollen sacs being brought to beehive, so that bee keepers can know the forage situation of bees without opening beehives. If this research is successful, it will significantly reduce the human effort in bee keeping. The cultivation of bees will become more convenient.

1.3 Research gaps

Computer vision is not commonly used for bee detection and tracking. Most research for monitoring bees uses a camera inside the beehive. These researches have their advantages and disadvantages. One example was to utilize a vector quantization method to detect bees (Kimura et al., 2011). A limitation of this method is that the bee needs to fill the whole image. Another example is to apply region homogeneity to detect bees (Knauer and Meffert, 2010). However, this method is not precise enough for use in uncontrolled (natural) conditions outside the beehive. Last but not least, A background model of Gaussian functions was also used to monitor bees (Knauer et al., 2005). This method was applied inside the beehive. Then the setting of the camera was complicated, and it might affect bee behaviour

Some papers focus on the flight of bees outside the beehive. Kimura et al tracked bees on a flat surface (Kimura et al., 2012). However, they just tracked small bees' bodies as black points. Campbell et al reported motion detection for tracking flying bees (Campbell et al., 2008). Conversely, they had a problem because shadows caused confusion. Chiron et al reported the combination of motion detection and colour segmentation to detect flying bees on 3-D video (Chiron et al., 2013a). This combination is able to detect bees, but the 3D video cost time and resources.

The difficulty is that bees fly fast and bees all have very similar features, so this can affect bee tracking. However, the information of the bees' positions over several image frames can be used. When tracking bees, the predicted position can be calculated for each bee and compared to the actual position. A Kalman filter with position prediction could be used to track bees on 3-D video (Chiron et al., 2013a). However, it is unclear whether this method can be used successfully with 2D video. Other techniques, such as blob analysis, may also be suitable for bee tracking (Lu et al., 2005), but they have not been reported.

Pollen detection and measurement is an original feature of this research. Colour segmentation may be useful for pollen detection because, the bee's body is close to the pollen, the legs and head of bee are a black colour. Although many papers have used colour segmentation, there is quite few researches concerning the detection of pollen sacs on flying bees. One example was to analyse bee bodies colours to identify pollen carrying bees (Babic et al., 2016). They used the Mixture of Gaussians method to detect bees. Then, the colour variance and eccentricity were analysed for distinguishing pollen bearing bees. However, they did not detect pollen directly, but just find bees which were carrying pollen on each frame of a video. They also did not design a bee tracking model to count the number of pollen sacs. Another possibility was to use the circular shape of the pollen sac. Liew et al detected bee cells in a beehive using the Hough Transform

(Liew et al., 2010). However, the pollen sacs on a flying bee may be too blurry to be detected with this method.

The main aim of this research is to analyse the 2-D video to detect pollen sacs on honey bees and estimate the amount of pollen being taken into the hive. The following questions need to be addressed:

- How can technology be used to detect and track bees entering the hive?
- What computer vision techniques can be used or need to be developed to detect and measure the pollen sacs?
- How can these methods be improved and what are the limitations of these methods?

1.4 Outline of this thesis

Because bee detection and tracking and pollen detection is not common in computer vision, this thesis will review research concerning object detection and tracking in chapter 2. In addition, it also includes a review of research in the area of bee and pollen detection. In chapter 3, the theories of methods which relate to this research will be explained. This includes the Gaussian Mixture Model for motion detection, colour thresholding, Hough transform, morphological calculations, Kalman filter, Hungarian assignment method, image moments and receiver operating characteristic and deep convolutional neural networks.

The methodology of bee detection and tracking is introduced in chapter 4. The motion (foreground) detection and colour thresholding will be combined for bee detection. Bees are then tracked using a Kalman filter with Hough transform. Chapter 5 presents two methods of pollen sacs detection and measurement. The first method is to use conventional image processing and statistical analysis. It includes three steps. At beginning, the main body of the bee is detected and removed. In the next step, colour thresholding is used to detect possible pollen blobs and other blobs of the same colour. After that, pollen sacs are filtered out using various features. The number of pollen blobs is estimated by using these features. Finally, the pollen detection result is combined with bee tracking model to count the number of pollen carrying bees on a honey bee monitoring video. The second method is to use deep learning algorithm to detect and measure pollen sacs. At the first step, the deep convolutional neural network is trained and applied to detect pollen sacs on individual bee images. The next step is to identify whether an individual bee image includes pollen sacs or not. Finally, the identification is combined with the bee tracking model to count the number of bees which are carrying pollen on a video.

The Chapter 6 will show the results of the bee detection, tracking and pollen detection and measurement. It also compares the results between the conventional image processing method and the deep learning algorithm for pollen sacs measurement on different bee monitoring videos. Finally, conclusions and future work will be presented in chapter 7.

1.5 The important of bees and contribution of this research

This section will introduce the important of bees in the world to show more significant of this research's outcome. Bees have two important roles which are pollination of crops and honey production. Therefore, bees' health is observed by bee keepers all the time. The main point is to

check the forage in beehives. This research can help bee keepers to know the forage by showing pollen measurement.

1.5.1 The important role of bees

Honey bees have two special roles in the apiary industry, which are crop pollination and honey production. The most important contribution of honey bees is pollination services for agriculture. More than 45% of the most commonly produced global food commodities rely on honey bee pollination. The honey bee is the most efficient pollinator of most crops, especially for monoculture crops (vanEngelsdorp and Meixner, 2010). Worldwide, bees and other insects contribute crop production value for human food of about US$190 billion each year. In the United States alone, around US$16.4 billion of agricultural production came from insect pollination in 2010, and the value from honey bees was US$12.4 billion, about three-quarters of the total value (Johnson and Corn, 2014). About one-third or more of food consumed in America relies on bee pollination, including high-value fruits, vegetables, tree nuts, livestock forage, some field crops, and other specialty crops. In Europe, the value of insect pollination was estimated at €14.2 billion (Gallai et al., 2009). 84% of crop species cultivated in Europe need insect pollinators, especially bees (Williams, 1994). In the United Kingdom, about £120 million to £200 million of crops, including forage crops, is dependent on honey bee pollination every year.

Honey which is produced by bees, is an important commodity. The global production of honey was estimated at 1.07 million metric ton in 2007, which was estimated to be worth US$1.25 billion. Production has increased 58% since 1961. In 2006, the US average price for honey was US$1168 metric ton (vanEngelsdorp and Meixner, 2010). In New Zealand, the value of honey exports is around $81 million. $5.1 billion of NZ's economy comes from pollination by honey bees, domestic honey sales and exports, beeswax and exported honey bees (Zealand, 2013). In addition, approximately 9000 to 12000 tonnes of honey were produced every year until 2012 (Industries, 2013). In 2013, the production of honey was increased to around 18000 tonnes, almost one-third to one-half is exported (Industries, 2013). Manuka honey is very famous and exclusive to New Zealand. The price of manuka honey increased 10 to 20 percent from 2012 to 2013, reflecting strong export demand (Industries, 2013).

The global population of managed honey bees has generally been increasing. The number of managed honey bee colonies was estimated at 72.6 million in 2007, an increase of 64% since 1961. Over the last five decades, the global stocks of honey bees have increased, but not in all parts of the world. The largest increases were recorded in Asia (426%), Africa (130%), South America (86%), and Oceania (39%). However, the colonies of managed honey bees have been reducing in Europe (26.5%) and North America (49.5%) (vanEngelsdorp and Meixner, 2010). In the United States, according to the annual "winter loss" survey from USDA, bee colony winter losses averaged 17%-20% per year between the 1990s and mid-2000s. By comparison, bee colony winter losses were about 30% per year from 2006 to 2013 (Johnson and Corn, 2014). Winter loss is a result of bees dying in cold weather. These bees are replaced in warmer weather, but the increased loss indicates the bees are less healthy in some way. Especially, from 2006, commercial beekeepers began to report sharp declines of their honey bee colonies. This phenomenon has been named colony collapse disorder (CCD), because these colonies decline

in unusual circumstances (Johnson and Corn, 2014). The reasons for this phenomenon are suburban urbanization, insecticide use, insect pests and decreasing numbers of bee farms. In Europe, the number of colonies decreased from 21 million in 1970 to 15.5 million in 2007, but this reduction was slow before 1990. Since 1961, colonies in Finland and Spain have increased more than 50%, whereas Austria and Germany have seen a decline (although Germany increased before 2000), and Sweden has dropped approximately 75% (vanEngelsdorp and Meixner, 2010).

The reduction of bees can affect not only the economy but also agricultural production. Honey bee pollination is very important to crops. Globally, 52 of the 115 leading crop species rely on or at least benefit from animal pollination (Klatt et al., 2014, vanEngelsdorp and Meixner, 2010). 5 of the honey bee-dependant commodities would reduce yield (including fruit size, quality, or quantity) more than 90% without honey bees, 16 commodities would reduce yield 40-90%, 19 commodities would reduce 10-40%, and 13 commodities would reduce slightly less than 10% (Klein et al., 2007, and vanEngelsdorp and Meixner, 2010). Agricultural production would reduce by about 8% without pollination (vanEngelsdorp and Meixner, 2010). Many crops are not 100% reliant on insect pollination, so some reduced production can be compensated for by increasing cultivated acreages. An increase in the cultivated land for pollinator-dependent crops of 15% in the developed world and 42% in the developing world would be required to offset the effects of reduced pollination (Aizen and Harder, 2009, vanEngelsdorp and Meixner, 2010).

In the United Kingdom, the decreasing honey bee population has caused crops and forages to lose this method of pollination. Biesmeijer et al reported there is a relationship between the reduction of bee numbers and the declining of pollinated plants (Biesmeijer et al., 2006). If this situation is maintained for a long time, livestock will suffer because of the lack of forage, crop yields will be cut, and there will be massive food shortages.

The reduction of pollination also affects human health. Most of the pollination-dependent crops provide essential nutrients for human health such as vitamins, antioxidants and fibre (Seeram, 2008, Eilers et al., 2011). Without these crops, such as fruits, human beings will suffer more malnutrition.

1.5.2 Health of bees

The health of bees has been studied for a long time. The factors which affect bee health include disease, nutrition, bee management issues, environment changes, pesticides and other agricultural practices (Staveley et al., 2014). Bee disease is the most important factor, but bee diet and nutrition factors also cause reductions in honey bee numbers. Individual bee health and colony longevity are both impacted by nutrition, and a poor diet can affect a bee’s immune system, making it susceptible to harm from disease and parasites. Bees need good forage and different types of plants to support colony health (Johnson and Corn, 2014).

Well-nourished bees, which produce progeny and resist multiple stress, can guarantee the health of a bee colony (Brodschneider and Crailsheim, 2010). Nutrition deficiency relates to the quality of food and the lack of availability of food. Honey bees' diverse diet needs include protein, carbohydrates, lipids, vitamins, minerals, and water for good health (Huang, 2010). Pollen is a source of protein, minerals, lipids and vitamins (Gaboratory, 1978). Different plant species and

regions have pollens with different nutritional values. (Brodschneider and Crailsheim, 2010). Because of the variable nutrients in pollen and nectar, honey bee colonies can get deficient in one or more nutrients if they are only visiting one type of plant, such as blueberries (Staveley et al., 2014). In addition, some farmers pollinate their crops by breeding bees. However, since farmers often cultivate a single species of tree, honey bees may only get a single type of pollen. In this situation, honey bees suffer malnutrition and immune deficiency. Naug reported that nutrition deficiency can lead to a reduction of adult bee survival, such as a shorter lifespan for worker bees, compromised brood development, and subsequent depopulation of the colony (Naug, 2009).

In addition, nutrient deficiency affects adult bee survival indirectly. Adult bees require sufficient food to feed larvae and rear the larvae to adulthood. The next generation of adults and the rearing of subsequent broods can be affected by an inadequate store of pollen. Larvae which are reared during the nutrient-deficient periods will have a reduced lifespan as adults and reduced foraging abilities (Brodschneider and Crailsheim, 2010).

A lack of nutrients in a honey bee's diet also affects the immunity of the bee. Well-nourished honey bees are less susceptible to the Nosema Cerane parasite than weakly nourished honey bees. In addition, bees which feed on a multi-floral diet of mixed pollen have improved immune functions, such as glucose oxidase activity which leads to better in-hive antiseptic protection. Alaux et al reported that glucose oxidase activity can help bees to disinfect the colony and brood food (Alaux et al., 2010). Therefore, there is a relationship between nutrition and the health of a bee's immunity system.

1.5.3 Contribution of this research for bee health observation

Pollen measurement is a useful method to monitor the health of a bee colony. If inadequate pollen goes into the beehive, lack of forage leads to a reduction of adult bees. If too much pollen is brought back to the beehive, there may be some unhealth things going into beehive, which can also cause health problem. If the pollen being collected can be identified and measured, it is easier for beekeepers to supervise the health of bees.

1.6 Publications

Y. Cheng and J. Collins, "A model for honey bee tracking on 2D video," in *2015 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, 2015, pp. 1-6, doi: 10.1109/IVCNZ.2015.7761542.

C. Yang, J. Collins, and M. Berckerleg, "Receiver Operating Characteristic for Pollen Sacs Measurement from Honey Bees' Monitoring Video," *International Journal of Imaging and Robotics,* vol. 17, pp. 1-14, 2017.

C. Yang and J. Collins, "Improvement of Honey Bee Tracking on 2D Video with Hough Transform and Kalman Filter," *Journal of Signal Processing Systems,* vol. 90, Iss. 12, pp. 1639-1650, 2017, doi: https://doi.org/10.1007/s11265-017-1307-x.

C. Yang, J. Collins, and M. Beckerleg, "A Model for Pollen Measurement Using Video Monitoring of Honey Bees," *Sensing and Imaging*, vol.1, Iss. 2, 2018, doi: https://doi.org/10.1007/s11220-017-0185-4.

C. Yang and J. Collins, " Use of deep learning algorithm for pollen sacs detection and measurement on honey bee monitoring video," *IET Computer Vision,* 2018. (submitted)

Chapter 2

Chapter 2: Literature review

There has been a lot of research in the wider field of object detection and tracking. Object detection includes two situations: single frame and multiple frames. Single frame detection aims to detect an object in one image. In this situation, features of an object such as colour, texture contour, etc. are used to detect object pixels. In the multiple frame situation, detection is based on consecutive frames of a video. In addition, the deep learning has become popular and powerful method for object detection recently. Object tracking also includes several different approaches. One approach is to use features to track a single object. Another approach for object tracking uses filtering. Normally, the position of an object is calculated after object detection. Then a combination of the predicted position and detected position is used to calculate a candidate position for the object. This review will introduce detection and tracking methods.

2.1 Object detection in a single frame

There are several methods of object detection in image processing technology for single image detection. They have limited areas of application, so there are different methods for detecting particular objects. The common methods use detection of features, such as colour, contour and texture. The features come from pixel values, and pixels with similar features are clustered into regions. If this pixel-based segmentation is not enough, then region-based segmentation can combine regions at a higher level for object detection.

2.1.1 Colour feature for image segmentation

Colour based segmentation uses colour features to detect objects. It is a pixel-based segmentation. Compared to other features such as texture, intensity and similar statistics, object detection using colour is simple and obvious. Therefore, colour is widely used in different ways to detect object regions.

Normally, the colour space for video is Red-Green-Blue (RGB). Each channel represents the intensity for each colour. However, there are many different types of colour spaces such as LAB, XYZ, HSV, YCbCr, YIQ and DHT that have been used for different situations and methods (Gritzman et al., 2014, Kumar, 2014). Human skin is an example, where Kumar tried to find a suitable colour space for face detection (Kumar, 2014). He compared the RGB, HSV and YCbCr colour spaces. The experiment indicated that the YCbCr colour space detected face skin more accurately. Another instance was the human lip, where Gritzman et al compared seven colour space models for lip segmentation (Gritzman et al., 2014). They evaluated 33 different colour

transforms in 21 channels from the colour spaces. They also added 12 new transforms, 8 of which were designed for lip segmentation. The 33 colour transforms were compared by using histogram intersection and Otsu's discriminant. The results showed that Hue-based transforms using the Hue component of HSV were the best for lip-skin segmentation; the a^* component of LAB achieved was best for lip-oral cavity segmentation; and the pseudo hue and the LUX transform performed reasonably well for both lip-skin and lip-oral cavity segmentation. For a special satellite image, Ganesan and Rajini utilized the YIQ colour space to separate objects (Ganesan and Rajini, 2014). This colour space is defined by the National Television System Committee (NTSC). The paper reported that the advantage of the YIQ colour space was that the human eye is very sensitive to intensity changes rather than changes of hue or saturation. This colour space separated the luminance in the Y component from the chrominance in the I and Q components. In the satellite image segmentation, they only performed histogram equalization on the Y channel. Compared to the RGB colour space which needs to be performed on three channels, the YIQ colour space was more efficient. Another example was to use the LAB colour space to perform segmentation for satellite image segmentation (Chitade and Katiyar, 2010). The satellite image in the paper included the three colours white, blue and pink. The LAB colour space separates luminosity in layer 'L', with all of the colour information in the 'a' and 'b' layers. This was easier for colour based segmentation. In natural image segmentation, Tse-Wei et al used the HSV colour space for segmentation of different objects, such as animals and houses (Tse-Wei et al., 2008). They produced a special colour histogram and a grey histogram using the HSV colour space. The 'H', 'S' and 'V' channels were used together to collect pixels in the colour histogram. This method helped them to cluster pixels using K-means clustering. If one space is not enough, more than one colour space can be useful to perform detection. An example was to use a colour segmentation algorithm to detect traffic signs (Shadeed et al., 2003). They converted an RGB image to the HSV and YUV colour spaces. In the HSV space, the H value determines the colour. In the YUV space, U is positive if blue is greater than a certain percentage of green and red, and V is positive if the red is greater than a certain percentage of green and blue. Normally, the traffic signs had red frames with black and white colours inside. The threshold in V determined the red colour to be considered as the red part of a sign, and the threshold in H was selected to remove the colours which could not be removed by the V threshold, such as yellow and orange colours. In summary, the different colour spaces are used for different types of images and algorithms.

Once pixels are segmented by colour, the pixels need to be arranged into different region groups. There are many methods for doing this.

A common method is to perform histogram thresholding on the colour image. Histogram thresholding has been used for monochrome image segmentation. For example, Otsu performed histogram thresholding after grey-level image segmentation (Otsu, 1975). The aim of him was to find an optimal threshold which could separate object regions automatically. He analysed the grey-level histogram, and then the optimal threshold is selected by a discriminant criterion which was discussed. In summary, the thresholding method is based on the histogram to find the thresholds which can be used to separate different regions of the image. A colour image has multiple colour dimensions (normally 3 channels), so the colour histograms are created for each

colour channel. The three channels of colour image were organised together to determine thresholds. A summary which described recursive thresholding was in year 1980 (Ohta et al., 1980). Initially, a picture stack was created for storing region masks. Then the whole image was made to be a mask which is placed at the bottom of the stack. After that, histograms of colour features were analysed to choose the thresholds. Then the thresholds were used to detect a new region. Finally, a region mask was created for this region. This new region mask was put onto the stack for new mask creation. Cheng et al stated that selecting thresholds based on the 3-dimensional colour channels is computationally expensive (Cheng et al., 2001). One solution is to develop an efficient method for storing 3-D colour information. An efficient method was described in year 2003 (Shadeed et al., 2003). They used multiple thresholds to detect traffic signs. They used the H channel of the HSV colour space with the U and V channels of the YUV colour space. They found the thresholds for traffic signs were $-20 < \mathrm{H} < 20$ degrees in H channel, and $\mathrm{V} > 28$, $\mathrm{U} < 0$ in the V and U channels respectively. Then they applied these thresholds, and combined the results from the two colour spaces using the 'and' operation, for traffic sign colour segmentation. Another method was to project the 3-D channels onto a lower dimensional 2-D or 1-D space (Cheng et al., 2001). Ohta et al presented a $I_1 I_2 I_3$ colour feature from the RGB colour components for special histograms (Ohta et al., 1980). Compared to other colour spaces (such as RGB, XYZ, and YIQ, etc.), their features were more effective. For instance, the method could segment an image using only two features rather than using three features. Another example was to estimate the 3-D space cluster distribution using only 1-D histograms (Celenk, 1990). Then the clusters were projected onto a line determined by the Fisher discriminant method for 1-D thresholding. The Fisher discriminant could minimize the clustering error rate. This permitted the simultaneous utilization of all colour information. In addition, Siang Tan and Mat Isa introduced another special method to improve the thresholding efficiency (Siang Tan and Mat Isa, 2011). They applied histogram thresholding to detect possible uniform regions in the colour image. There were three steps: peak identification, region initialization and a merging process. The peak identification was achieved by thresholding to identify the dominant peaks in the histogram. Then all the peaks in the three colour channels were used to obtain uniform regions corresponding to the intensity level of dominant peaks in their respective histograms. After that, a merging process was applied to merge these regions together to detect the object region. In summary, histogram thresholding for colour segmentation is more complex than for the gray-level image. However, simple colour objects, such as a pear, mouse or cup, do not require a complicated calculation.

Another method for colour base segmentation is clustering. The common algorithms are K-means (KM) clustering (Chitade and Katiyar, 2010) and Fuzzy C-means (FCM) clustering (Ganesan and Rajini, 2014). KM method could be used to segment a satellite image using colour information in the LAB colour space (Chitade and Katiyar, 2010). According to their paper, KM was used to determine the natural spectral groupings present on an image. This could determine the number of clusters located on the image. Each pixel in the image was then assigned to the cluster whose mean vector was closest. The procedure continued until there was no significant change in the location of class mean vectors between iterations of the algorithm. The method in this paper could

be used for mapping the changes in land use from land cover images taken over a time period. Tse-Wei et al reported using KM for colour segmentation (Tse-Wei et al., 2008). They used Maximin initialization to find the number of clusters K and to choose the centroid positions automatically. In their paper, the segmentation results were close to the perception of human vision. The number of clusters was suitable for colour representation in their test images. KM was also used to segment an image into regions with colour information (Nirgude and Jain, 2014). They employed the colour histogram and hill climbing technique to detect the pixels which are peak points. The number of identified peaks was the initial number of clusters. Then KM was performed to segment the image into regions. However, in the paper, KM was not enough to detect an object. They merged regions and detected the object at the region level.

FCM is another cluster method. Ganesan and Rajini reported that the image pixel could belong to more than one cluster (Ganesan and Rajini, 2014). They performed FCM on the YIQ colour space to segment satellite images. They reported that FCM relied on the spatial information and that it was affected by the noise pixels incorporated with the image pixels. They modified the FCM by adding a spatial function which was defined as a weighted sum of the membership function in the neighbourhood of each pixel under consideration. The experimental result showed the efficiency of the method. Kumar compared different statistical methods (including Fuzzy and Gaussian) and chose the fuzzy classification with the RGB colour space to identify human face skin (Kumar, 2014). He finally showed that face detection using fuzzy logic was more robust and was the optimum technique for improving accuracy in skin colour detection. The clustering algorithm was improved to be faster and more robust, but the selection of the initial clusters and the number of clusters can affect the results.

Another approach for colour image segmentation is the neural network. This algorithm needs initial training of an object to calculate features from the colour information. These features are used to detect the object in other image situations, such as with different backgrounds. An example of using a neural network is in 1997 (Littmann and Ritter, 1997). The neural networks were used for colour segmentation. Their approach was based on local linear maps (LLM). They used a human hand image as an example and trained the network using 160 images of same hand, but from different viewpoints. In their experiment, they compared the LLM and standard statistical methods. The LLM network was more robust when the image condition was more difficult. Hassanat et al trained artificial neural networks (ANN) using the colours of the pixel to perform the segmentation (Hassanat et al., 2016). They applied the method to the human lip, face, hand and finger, and to a tree leaf. The colour information was from each pixel and its surrounding eight neighbours which were used to create feature vectors. Then the feature vectors were classified using the ANN. The experiments showed their special colour information was effective and fast. The ANN was also faster than other machine learning algorithms, such as K-nearest-neighbour and its variants.

Colour information can also be used in some uncommon approaches. Some research has combined other features with colour or in cooperation with different approaches to improve the accuracy of detection. One example is a shadow-highlight invariance method for detecting of traffic signs (Fleyeh, 2006). This method was based on the HSV colour space. He reported the

colour information was very sensitive to the variations of the light conditions such as shadows, clouds, and sunlight. The solution was that the hue and saturation were invariant to the effects of illumination variations which was used to develop a shadow-invariant algorithm. The hue was also invariant to the effects of highlighting which was used to develop highlight-invariant colour segmentation algorithms. He tested the method on hundreds of images. The results showed that the method was robust, and successful in more than 95% of cases. Peng combined colour and shape information to perform object detection (Peng, 2015). He used an automatic seeded region growing method for image segmentation. The use of colour was complementary to shape. After that, he presented a colour template which was similar to dominant orientation templates as the next step. Then the matching scores from the two templates were combined to detect objects. Finally, he analysed the complexity of this method to propose a speed-up strategy. Another example was to segment an image with colour and texture features (Bhattacharya et al., 2014). They employed Haar wavelets to decompose the red, green and blue of an input image. This produced approximate coefficients and vertical, horizontal and diagonal detail coefficients. Then these coefficients were used to cluster pixels using KM and FCM. The result of this wavelet-base clustering indicated that this method not only improved the detection result, but also reduced the execution time for each image, compared with the single clustering technique. (Ananth et al., 2014) introduced an intermediate feature of maximum overlap wavelet transform (IMOWT) as a pre-processing step for histogram grouping segmentation. IMOWT is an efficient transform which uses a set of wavelet features of the same size and various levels of resolution and different local window sizes for different levels. This method was fast, flexible and invariant to translation. After the image input, the coefficients derived from IMOWT were subjected to 2D histogram grouping to obtain different colour regions. The experiment showed that this method produced better results when compared to the direct application of 2D histogram grouping.

Comparing different types of colour segmentation, histogram thresholding is the simplest and fastest method, although it is limited in complex colour detection. It is also easy to combine with other approaches for object detection. Apart from thresholding, the other colour based segmentation methods are complicated. If they are combined with other approaches, the computational cost is likely to be very high.

2.1.2 Other features for object detection

Apart from colour, there are other features that can be used for image segmentation and object detection. These include intensity, texture and contour. These features can be combined to detect complicated objects. The features are used in different algorithms to perform object detection. The algorithms include clustering, thresholding, neural networks and machine learning. Some of these methods were discussed in the previous section.

The intensity value of a pixel is the value of the pixel in a grey image. The intensify feature includes the information of brightness and illumination. Human faces could be detected by using the features of illumination and reflection on grayscale images (Mohapatra et al., 2014). They reported the change in lighting conditions caused the problem of apparent changes in facial appearance. They eliminated this problem by enhancing the reflectance while reducing the contribution of illumination. This illuminance-reflection model was used to normalize the images.

Then the images were used to train support vector machines (SVM) to get the classifier. The classifier stores the features of the faces from training images, and then filters out the features in the following images to detect the face.

The contour feature shows the shape of an object as an edge against the background. These edges separate an object from different regions, so edge detection can detect the shape of an object. Normally, the contour is used with region-base detection. Arbelaez et al developed a gPb contour detector for object detection (Arbelaez et al., 2011). They combined multiscale local brightness, colour, and texture cues to a powerful globalization framework using spectral clustering. They then linked this contour detector with a generic grouping algorithm. Firstly, an Oriented Watershed Transform constructed a set of initial regions from an oriented contour signal. Secondly, they used an agglomerative clustering procedure to form these regions into a hierarchy which could be represented by an Ultrametric Contour Map. Finally, their hierarchical region trees served as a natural starting point for interactive segmentation. A user could correct errors in the automatic segmentation with minimal annotation. Another method was to integrated edge and region information to segment lymph nodes of the thoracic region on real CT images (Yu and Poh, 2015). The edge-based snake model is based on the edge- or gradient-driven energy landscape for the snake to evolve. The region-based snake model separates the image into two different regions (object and background) with maximum intensity separation. These two models were integrated with the region-based snake with edge constraint to overcome their respective limitations. The contour was evolved by using a region-based snake formulation on grey level CT images while the partial edge map of the lymph node was derived using the Canny edge detector. The contour grew freely until any segment encountered the partial edges of lymph nodes. The result indicated that this integrating improved the segmentation significantly compared to generic snake methods.

Texture describes visual information which is related to local variations of the image. The texture feature cannot be described by each pixel, so it is extracted by different filters on a group of pixels. Qi et al extracted the texture feature by using Laws' filter for image segmentation (Qi et al., 2015). Initially, the Laws' texture energy measures were computed by applying small convolution kernels to an image. Then the measures performed a nonlinear windowing operation. The two-dimensional convolution kernels were used for texture discrimination. In the paper, they selected 5 vectors as convolution kernels, so filters of 5×5 matrices were used to compute the energy of texture. After the filtering, they calculated the histogram of two-channel grey value images of texture energy. The histogram was used to design a global threshold and select pixels from the image whose grey values fulfilled the conditions. Finally, they segmented the image by means of a two-dimensional pixel classification and displayed the texture region which was the desired object region. Another special texture descriptor was a texture image segmentation with factorization method (Yuan et al., 2015). The feature which they used was a form of texture descriptor based on local spectral histograms. The spectral histograms captured local spatial patterns via filtering and global impression through histograms. When the filters were selected properly, the spectral histogram could represent an arbitrary texture appearance. This method represented an image by an $M \times N$ feature matrix, which contains *M*-dimensional feature vectors

computed from *N* pixels. They regarded the feature at each pixel as a linear combination of representative features, which encoded a natural criterion to identify boundaries. Therefore, the feature matrix was expressed by representative features and their combination weights per pixel. The combination weights indicated segment ownership for each pixel. Finally, they used singular value decomposition and nonnegative matrix factorization to factor the feature matrix, which generated accurate segmentation.

Features are also combined for image segmentation and object detection. The integrating method can be better than using a single feature. One example was to segmented natural images based on colour and texture features (Ilea and Whelan, 2005). The colour features were extracted by using a multispace adaptive clustering algorithm. They extracted the dominant colours from the filtered image on RGB and YIQ colour spaces and calculated the optimal number of clusters using an unsupervised classification procedure base on the self-organizing maps (SOM). Then the RGB-YIQ data was clustered with a 6-D K-Means algorithm to finalise the colour segmentation. The texture features were calculated by a multichannel texture decomposition scheme which was achieved by filtering the input texture image with a 2-D Gabor filter bank. Finally, they used the clustering strategy of adaptive spatial K-Means clustering (ASKM) to integrate the colour and texture features. The ASKM could minimize the errors in the assignment of the data-points into clusters by adaptively sampling the local texture continuity and the local colour smoothness in the image. The experimental results indicated that this method produced accurate segmentation, even if it was applied to images with low resolution and low contrast. Another research built an intensity-texture model for image segmentation (Min et al., 2015). In their intensity term, they employed a global division algorithm to construct a regional based term as the intensity term, which detected an image object with a large intensity difference or noise in the same region. In the texture term, the algorithm of adaptive scale local variation degree (ASLVD) extracted the amplitude and frequency components of the local intensity variation, which was used to reflect the texture feature effectively. Finally, they combined the intensity term with the texture term which made the evolution contour smooth by minimizing an energy function. This intensity-texture model was mainly used to segment complicated natural images.

2.1.3 Region based segmentation

When pixels are grouped into different regions, each region can be identified from its boundary or its constituent pixels. Region based segmentation builds homogenous regions, which satisfy a given homogeneity criterion (Hanbury, 2007). The criterion is based on image features such as intensity, colour, texture, shape, etc. The regions can be constituted either by grouping pixels from regions (region growing algorithm) or by starting with a single region and successively subdividing it (splitting and merging algorithm).

The region growing approach grows a region of a partition starting from a seed (one or more pixels). It adds adjacent pixels to the region if they satisfy a similarity criterion (Hanbury, 2007). The growing stops when no adjacent pixels satisfy the similarity criterion. One of the region growing approaches was introduced in year of 1994 (Adams and Bischof, 1994). They introduced a seeded region growing (SRG) method. SRG was controlled by choosing a small number of pixels which are known as seeds. The seeded regions were chosen to be as homogeneous as

possible. Then the seeds were expanded to include all homogeneous neighbours. This procedure was repeated until all pixels in the whole image were allocated. The problem was the selection of initial seed regions. In their example, the SRG was applied to grey-scale images, but it could be extended to colour or multispectral images straightforwardly. They concluded that SRG was a rapid, robust, easy-to-use image segmentation procedure requiring neither tuning parameters nor training sets. The region growing method was used for edge detection. In 1990, the region growing method was used to split the image to an over-segmented result (Pavlidis and Liow, 1990). Then they eliminated the region boundaries using a uniformity criteria. After that, contour modification replaced boundary modification for edge detection. The conclusion was that this combination of region growing and edge detection generated better results than those of a single method.

Region splitting begins with the whole image considered as a single region. If the homogeneity criterion is not satisfied for this region, then it is split into four sub-regions. The homogeneity criterion is then tested on each sub-region, and those that do not satisfy the criterion are split into four sub-regions. This process continues until all regions satisfy the homogeneity criterion (Hanbury, 2007). An example of the region splitting method is presented in 1978 (Ohlander et al., 1978). The whole image is the initial seed region. Histograms of feature values were used in this region to determine a threshold for one feature that was used to split the region into sub-regions, which became new seed regions. This process is repeated until all sub-regions were homogeneous. This method could be applied to many different types of images, such as natural scenes, satellite images, aerial photographs, and radar images. They concluded the method could be used in a larger system such as an image understanding system, or image matching system. The disadvantage was that the resulting image segments were rectangular.

Region merging is a hierarchical approach which can be combined with region growing and region splitting to merge similar regions for constructing homogeneous regions as large as possible. If the union of a pair of adjacent regions satisfies the homogeneity criterion, then these regions are merged (Hanbury, 2007). The region growing and merging method could be used to segment an image (Tremeau and Borel, 1997). The region growing was based on criteria that took into account both colour similarity and spatial proximity. After that, the regions were merged following a criterion that took colour similarity into account. The approach combined both local parameters and global parameters to generate a non-partitioned segmentation of the image being processed in spatially disconnected regions of similar colour. They concluded this algorithm could be used not only in the RGB colour space, but also the LAB and LUV uniform colour spaces. Another approach was to use a splitting and merging method to propose a video-rate image segmentation (Aneja et al., 2009). An optimal split method was used to split the image into several rectangular regions. The initial splitting created homogenous regions. Each homogenous region was assigned a unique label. A binary tree data structure was used to maintain a list of the regions, so that all the regions could be accessed by scanning the binary tree. In the merging step, every region was merged with its surrounding regions to produce larger regions which were still homogenous enough. Then the binary tree was renewed to perform the next merging. This merging process was repeated until no more regions could be merged. This method was suitable

for real-time segmentation and it could be used for fast pre-segmentation purposes for localization or tracking.

Region merging has also been combined with some pixel-based segmentation methods. The pixel-based algorithms split the image into regions, and then region merging is performed at a higher level to merge regions for object detection. The region merging could be used after K-means clustering to detect objects in an image (Nirgude and Jain, 2014). The segmentation method used the KM with colour feature. Their merging algorithm included three stages. The first stage was the consistency property test (SPRT) which checked whether the neighbouring regions were homogenous or not. The second stage was the nearest neighbour graph (NNG) that grouped the colour clusters. Similar pairs of regions were connected by analysing the region edge. The third stage was a dynamic region merging (DRM) algorithm which was the final step to produce merged data. In DRM, a region predicate was compared with the description of adjacent regions. If they were same, they were merged, otherwise they were identified as different regions. They concluded this technique was useful for medical and security purposes.

Region based segmentation has some disadvantages. Region growing has difficulties selecting seeds and the growing range. An improvement is to try to find an effective algorithm for selecting seed regions. Zhen et al utilized an Improved Visual Attention Model (IVAM) to select better seed regions for region growing (Zhen et al., 2010). They used grey values and edge feature information rather than the colour, intensity and orientations used in VAM. In addition, these features were extracted using the Gauss-Laplace operator and Gabor filters which was different from the traditional VAM that obtained features using a Gaussian pyramid and Gabor filters. This improved VAM not only quickly extracted visual features, but also provided the growing range for region growing. After that, grey feature maps and edge feature maps were obtained and they were combined to build the saliency maps which were used to select saliency regions using dynamic neural network methods. Finally, the region growing seed was selected from the salient region as the starting point of growth. Pixels with the same or similar characteristics to the seed were combined into regions until there were no pixels similar to the seed combination in the salient area. However, this paper only segments images with a simple background and foreground.

Another problem of region based segmentation is the clustering of regions. One solution was to take the features of colour, texture and location into a mean-shift clustering algorithm to identify the regions (Yong-Mei et al., 2008). They first extracted colour, texture and location features from each pixel to form a feature vector from the LAB colour space. Then these vectors were clustered using a mean-shift clustering algorithm. In addition, a window parameter was chosen by selecting an optimal clustering amount, so the numbers and the centres of clusters were also selected, and each pixel was grouped and labelled. Then the regions with the same labels were segmented again and relabelled using neighbour connection theory to describe the image regions. This method was more precise, but it took 2 to 6 seconds to identify the seed region.

To make the region based segmentation more accurate, Gould et al combined multi-class image segmentation with object detection (Gould et al., 2009). They proposed a hierarchical model that included pixels, regions and objects in the image. At the region level, pixels were labelled as

belonging to one of a number of background classes or a single foreground class. A modular energy function was built, which included the location of the horizon, region label preferences, region boundary quality, object labels and contextual relationships between objects and regions. Then the foreground class was classified, at the object level, into one of the object classes. This method could identify the sophisticated shape and appearance features computed over candidate object locations with precise boundaries, but it took almost 5 minutes to complete the detection. In addition, this method requires complicated calculations. Actually, many region-based methods have similar time consuming computational problems.

2.1.4 The Hough transform for object shape detection

The Hough transform has been used to recognise the shape of the edge on an image. The Hough transform uses shape parameters and edges to detect regular shapes, such as the straight line, triangle, rectangle, circle, ellipse, etc. The Hough transform was first reported in1962 (Hough, 1962). The classical example is straight line detection which comes from (Duda and Hart, 1972). The Hough transform uses a parameterization which transforms the parameters of the straight line formula to a 2-D parameter space, so that the line is represented as a single point. In addition, the parameters of straight lines through a point in the image space correspond to a sinusoidal curve in the parameter space. Therefore, points which come from the same straight line in the image space correspond to curves through a common point in the parameter space. Points which were on the same curve in the parameter space correspond to lines through the same point in the picture space. With this method, they transformed all of the points from the image space into their corresponding sinusoidal curves in the parameter space. Then they created an accumulator matrix (from the parameter space) in which each element (point) counted the number of curves going through that point. Thus, any high counts correspond to the parameters for straight lines in the image space. They also extended this method to circle detection, but here the parameter space became 3-D. Therefore, this method has computational problems for complex curve detection. The Hough transform was generalized for detection of non-analytic curves in grey images (Ballard, 1981). This method constructed a mapping from edge space to accumulator space. This mapping was described as a table of edge pixel orientations relative to a reference point. This was termed an R-table. The R-table contained the information to recognise an arbitrary non-analytic shape.

The Hough transform is commonly utilized for detection of analytic curves, such as lines, circles, parabolas and ellipses. Research has either reduced the computation or improved the accuracy of the Hough transform. Chiu and Liaw proposed an efficient voting method for circle detection which reduced the computation and the storage requirements of the Standard Hough Transform (SHT) (Chiu and Liaw, 2005). The main idea was to select two edge points to check the third point rather than randomly selecting all three points, which improved the efficiency of the Randomized Hough Transform (RHT). This method caused every edge point in the image to vote only for the most probable candidate in the parameter space. From their experiment results, this method was more efficient than SHT and RHT. In addition, ellipses could be detected by using a method based on the Straight Line Hough Transform (SLHT) (Nair and Saunders, 1996). The SLHT was used to calculate the loci of lines possibly passing through ellipse centres. Then a new variant of the

Hough transform built maps for the loci to the locations of possible ellipse centres. A simple cluster detecting method was used to estimate the initial ellipse centres. Next a more exact centre was determined from the estimation and the edge image. Then all of the parameters of the ellipse were determined and verified. This method improved the efficiency of ellipse detection. Lu and Tan improved the Iterative Randomized Hough Transform (IRHT) for detection of incomplete ellipses under strong noise conditions (Lu and Tan, 2008). The IRHT applied a randomized Hough transform to a region of interest in the image space. The region of interest is updated based on the latest estimates of parameters. The IRHT detected the target curve by iterative parameter adjustment and iterative use of the image space and parameter space. Because of the iteration process, noise pixels were gradually excluded from the region of interest and the estimation became close to the target. The IRHT was tested for ellipse detection with synthesized images. It was also applied to foetal head detection in medical ultrasound images. The results demonstrated that the IRHT was a robust and efficient ellipse detection method.

2.2 Object detection with multiple frames

Multiple frame object detection can use object motion that is observed when more than one frame is viewed. The most popular method is the Gaussian Mixture Model (GMM) which is a statistical method for subtracting the image background. There are also other background subtraction methods for motion detection. Other common methods include frame difference and optical flow. Some methods can remove the shadows of the objects.

2.2.1 Gaussian mixture model

Moving object detection is a method to remove the background and detect the moving foreground object. A common method is the Gaussian Mixture Model (GMM) or Mixture of Gaussians (MOG) which can identify the background model on a video (Bouwmans et al., 2008, Lin et al., 2011). GMM is a statistical method for background subtraction.

GMM is popular for the motion detection. This model is introduced initially in 1997 (Friedman and Russell, 1997). This model uses video sequences to remove the background or segment the foreground. Stauffer and Grimson generalized it for real-time video tracking (Stauffer and Grimson, 1999). They modelled each pixel as a mixture of Gaussians. According to the persistence and the variance of each pixel, the method can determine which pixels probably correspond to background colours. Pixel values are considered as foreground if they do not fit the background distribution. The method deals robustly with slow lighting changes caused by shadows, specularity, swaying branches, computer monitors and other troublesome features. It recovers quickly when the background reappears. The paper showed that their model could track people and remote vehicles in indoor environments, people and cars in outdoor environments, fish in a tank, and ants on a floor. After this method was introduced, different methods were used to improve the GMM. Bouwmans et al introduced different methods to improve GMM to fit different situations (Bouwmans et al., 2008). These improvements include intrinsic and extrinsic model improvements.

The intrinsic model improvements consider the MOG directly. Davis and Sharma used the GMM in the first step of their research to subtract the background in thermal imagery (Davis and

Sharma, 2004). The GMM was applied to each pixel location. The foreground identification which was the final process of the GMM was achieved using the squared Mahalanobis distance. This improved the normal GMM model. Another method was to use adaptive-K GMM to detect cars in video images (Tan et al., 2006). Their AKGMM was learned by a modified Expectation Maximum (EM) algorithm. The adaptive-K algorithm adaptively uses more Gaussian components (bigger K) if the image pixels include a complex pattern, whereas simple pattern pixels use fewer components. They found that the background estimation from GMM (the final step of GMM) could not be used to remove dynamic casting of shadows, but they used a Normalized Cross-Correlation (NCC) algorithm to solve this problem. The NCC explored the relationship between the shadow casting and background, where the intensity of the shadowed pixel was linear to the corresponding background, so the background model was also used in shadow detection. The GMM design had a conflict between model robustness to background changes and model sensitivity for detecting abnormal foreground objects (Lin et al., 2011). Efficient management of the background model could be controlled by using an adaptive learning rate for the GMM. They developed a Learning Rate Control Scheme with high-level feedback to resolve the conflict. In addition, a heuristic method based on frame difference was introduced to assist the learning rate control scheme for adaptation to fast lighting changes. The experiments showed this method improved the GMM ability to deal with regularized background adaptation, adaptation to fast lighting changes and adaptation to scene changes. The fast lighting changes were absorbed into the background within one second.

The extrinsic model improvements use other segmentation methods combined with GMM to improve object detection. MOG was reported that it may be unsuccessful in removing the dynamic background or objects with a motion pattern (Harville, 2002). Even though adding more Gaussians to the model could improve the accuracy, some desired foreground would also be selected as background, such as when a person occludes the background for a long time on a video. He used higher level primitives such as image regions, image frames or object semantics as high-level feedback to correct the background model. He reported the feedback repaired the background model within two seconds. A hierarchical method was presented to subtract the background (Javed et al., 2002). The approach included three levels of pixel, region and frame. At the pixel level, the MOG was used to model each pixel colour, and then combined with gradient based subtraction. The region level grouped foreground pixels into regions by edge and colour. The frame level could handle the global illumination changes which were a common problem in the MOG background subtraction. Zeng et al introduced a Type-2 Fuzzy set to the GMM to deal with the problem of a dynamic background or noisy data (Zeng et al., 2008). El Baf et al applied this T2F-GMM on an infrared video (El Baf et al., 2009). Their results showed that the T2F-GMM is more robust than the crisp GMM in the case of dynamic backgrounds, especially for waving vegetation. However, this paper reported that it would be better to build an adaptive version of the T2F-GMM which was able to determine dynamically the optimal number of Gaussians. Bouwmans and El Baf also reported this method has noise in the detection result when the background is complicated (Bouwmans and El Baf, 2009). In addition, GMM was found that it suffered from a low rate of background learning and identification of moving shadows (Zhu, 2011).

He combined colour clustering based post-processing and GMM to improve object detection results. The shadow was removed by comparing the background with lower brightness and similar chromaticity. In addition, he introduced a method of area confidence measurement. The experiment showed the over-segmentation of colour clustering was faster than K-means. Moreover, the whole detection method detected people on the road successfully. The conclusion reported it can be used for moving object tracking. Ma et al focused on improving GMM for the problem of complex scenes, such as swinging branches (Ma et al., 2015). They reported that GMM cannot detect slow and large vehicles accurately. When vehicles started moving from stopped with a low velocity, the shadows appeared easily. This paper improved the algorithm to detect vehicles, which combined frame difference and GMM. This improved GMM was tested in sunny and rainy conditions, and in bright and dark environments. The result showed the GMM was improved in the aspects of adaptability, accuracy and real-time operation. The moving vehicles could also be detected correctly and effectively in situations with various complicated factors. Another approach was to combine a differential operation method for each frame to the GMM to improve the target detection (Wang et al., 2015b). A candidate background was obtained from GMM. Then, they utilized a differential operation to analyse the current frame and the background and used the “and” operation with moving targets detected by GMM after thresholding process. The evaluation showed that the former GMM model to detect human movement was poor because the target blob had a black hole inside it. However, their combination method eliminated the hole. In addition, this improved method needed a shorter video sequence to perform the detection.

2.2.2 Background subtraction

Background subtraction is another common method that recognises motion through a background model. A background model is created as a reference to highlight differences on the current frame. This method had been studied since the 1990s, and it was used initially to detect people, vehicles and animals. Later, it was used for the more complex processes of intrusion detection, tracking and counting people (Sobral and Vacavant, 2014). They also introduced different types of background subtraction. GMM is the most popular method of background subtraction. However, there are other methods to calculate the background model.

A non-parametric background model was presented for estimating the probability of pixel intensity based on sampling the intensity values of individual pixels (Elgammal et al., 2000). The model adapted quickly to changes in the background process and detected moving objects with high sensitivity. This model could deal with a background that was not completely static, but included small motions like moving tree branches and bushes. This paper evaluated and compared the model to GMM. The evaluation result was that both models have similar false negative rates for small contrast values, but the non-parametric model has a much smaller false negative rate as the contrast increases. In addition, the non-parametric model was more sensitive in detecting objects with low contrast, and this model was less affected by the presence of objects. Han et al improved the non-parametric model by using an efficient density updating method (Han et al., 2004). They reported that updating the density function required a large amount of memory to maintain the model representation of the density. This paper introduced the modelling and

propagation of density modes for sequential density approximation. Once a new pixel was classified as background, the density function was updated selectively at each time step. The simulations in this paper showed this method was successfully applied to background modelling and subtraction.

A median value of the first frames could be used as the background model (Lo and Velastin, 2001). This method could handle some of the inconsistencies due to lighting changes. Rakibe and Patil also utilized the median method to initialize the background model (Rakibe and Patil, 2013). They then updated the background model by using a threshold technique. After the background image was obtained, the background could be subtracted from the current frame. In this paper, this method was used to detect a moving human body. Their conclusion reported this method could extract an accurate picture of a moving human body.

Kavitha and Tejaswini introduced a method which was similar to the GMM, but they incorporated several innovative mechanisms (Kavitha and Tejaswini, 2012). In their method, for each pixel, values were taken from the previous frame at the same location, and then compared to the current pixel value to determine whether the pixel belonged to the background or not. Then the method updated the model by choosing which pixels values to substitute in the background model, from the current static background scene. This was different from the classical idea that the oldest values should be replaced. They also employed the colour illumination to analyse the pixel value to detect shadow and shading in the background.

A Wallflower method could also be used for background subtraction on a video surveillance system (Toyama et al., 1999). This method processed images at pixel, region and frame levels. The pixel-level made probabilistic predictions which estimated expected background pixel values for the next image by using a Wiener prediction filter. The region-level considered inter-pixel relationships to develop the raw classification of the pixel-level. It avoided the foreground aperture problem. The frame-level background maintenance looked for sudden changes in large parts of the image and switched in alternate background models that explained as much of the new background as possible. The conclusion stated this method could be used to solve many of the common problems of background maintenance.

2.2.3 Frame difference

Frame difference is another common method for motion detection. The approach is to detect the moving objects from the difference between the current frame and the previous (reference) frame. Sometimes, the difference is calculated from three or more consecutive frames. This method uses pixel-based differences to detect moving objects.

This method is also named the temporal differencing method (Lipton et al., 1998). The principle was to take consecutive video frames and determine the absolute difference. They used this method to detect motion. Choi et al utilized the temporal difference to detect the moving part of a person (Choi et al., 2006). They introduced a weighted accumulation method to obtain the final temporal difference image.

Ha and Lee reported motion segmentation and tracking in visual surveillance, typically based on updating background images and detecting foreground objects (Ha and Lee, 2010). It required

many parameters. They employed an algorithm of multiple difference images which was more robust and only required one parameter. The experimental results showed this method required less computation time. The frame difference method was explained more in year 2014 (Singla, 2014). The method also used to detect object motion. He also introduced a method of transforming the absolute differential image to a grey image which could easily display the difference. The experiment indicated this method detected motion accurately inside a house, but it was affected by the air moving in an open area environment. Frame difference is also combined with other methods to improve object detection. Migliore et al detected a moving person using frame difference and background subtraction (Migliore et al., 2006). They combined the difference between the actual frame and previous frame with the difference between the actual frame and the model of the background. The evaluation of this joint difference method was better than the original frame difference method. Another approach was to build a frame difference energy image (FDEI) to detect a human gait (Chen et al., 2009). A gait cycle was divided into clusters. Then, the averaged image of each cluster was denoised to obtain a dominant energy image (DEI). Meanwhile, the frame difference was calculated by subtracting two consecutive frames. Then, the FDEI was constructed as the summation of its corresponding cluster's DEI and the positive portion of its frame difference. The positive portion of frame difference was obtained by setting the negative pixel values of the frame difference to zero. Their result showed the method was reliable and stable when an occluded or lost body portion appears in the gait sequences. Another example is to use three-frame-difference as the main method to detect fishes in water (Lan et al., 2014). To reduce the effect of water wave reflection, they also added background subtraction and a four-neighbour topology. The results showed that the method performed detection robustly during dynamic changes of environment, and was less sensitive to noise, and easy to implement.

2.2.4 Optical flow

Another method for motion detection is Optical Flow. This method calculates the image optical flow field, and then clusters pixels according to the optical flow distribution characteristics of the image. It can detect the overall movement from the background. Optical Flow was introduced in 1950 (Gibson, 1950). He also described the visual stimulus provided to animals moving through the world. Different optical flow techniques were compared in 1994 (Barron et al., 1994), including differential techniques, region based matching, energy based methods and phase based techniques. They emphasized the accuracy and density of measurements. Optical Flow (OF) was used in many different areas including motion detection and object segmentation (Beauchemin and Barron, 1995). It can also calculate an object's speed.

The optical flow method could detect motion accurately even without knowing the background (Lu et al., 2008). The temporal differencing technique was used to detect the coarse motion area for the optical flow calculation. Then, the optical flow method was applied to calculate possible movement pixels for each video frame. The Lucas-Kanade method was used to compute the derivatives of the image in three dimensions (x, y and t) to find the optical flow field. Then the companion matrix method was employed to simplify the calculation. After that, a 3D Sobel operator was used as the gradient operator to estimate the gradient for the three dimensions x, y, and t. Finally, an adaptive threshold was selected to distinguish moving pixels. The results from

optical flow for detecting movement still included some background noise, so they used double background filtering with morphological processing to improve the result. Another example was to use an absolute differential image and optical flow field for motion detection (Shuigen et al., 2009). The difference from two consecutive frames of a grey image produced the absolute differential image. Then the pixel differences were used to estimate the optical flow field. The optical flow field was used to judge whether holes in a moving object belonged to the motion area or not. The experimental results indicated that the method is suitable and effective situations where the camera is stationary while objects are moving. Wang et al detected motion from video which included camera motion (Wang et al., 2015a). They used homography to remove the effects of camera motion from optical flow. In the dense trajectory features approach, points (pixels) were tracked by median filtering in a dense optical flow. Their experiment example reported that sometimes the homography was incorrect and the corresponding optical flow was not correctly warped. They solved the problem by estimating the homography and warped optical flow for every two frames independently to avoid error propagation. Optical flow estimation can help motion detection. The optical flow field is the velocity field that represents the 3-dimensional motion of object pixels through a 2-dimensional image (Shafie et al., 2009). However, the optical flow limitation is the movement distance. According to the smoothness constraint, the corresponding points in two consecutive frames should not move more than a few pixels (Lu et al., 2008). If the object moves too fast, the pixels move a long distance between two frames, and the pixels may not be tracked successfully.

2.2.5 Shadow detection and removal

Shadows are a common problem in moving object detection and background subtraction. Shadows move with objects and have the same colour information as the background, although they are darker than the original background. Campbell et al showed their method of adaptive background subtraction can be used to detect bees in their flight and crawling, but the shadows affected the results (Campbell et al., 2008). Many researches have attempted to remove this effect. Horprasert et al identified the shadow in their background subtraction model by comparing the brightness and chromaticity components between the background image and the current image (Horprasert et al., 1999). They reported that the shadow has a similar chromaticity but lower brightness than the same pixel in the background image. In their result, the human shadow could be detected in the case of human detection. Another approach was to separate colour information from lightness information to remove the shadow from the target (Elgammal et al., 2000). The chromaticity coordinates *r*, *g* and *b* were $r = \frac{R}{R+G+B}$, $g = \frac{G}{R+G+B}$, $b = \frac{B}{R+G+B}$ where $r + g + b = 1$. In this model, the shadow pixels were less sensitive to small changes in illumination. Then they also measured lightness using $s = R + G + B$ to detect shadows on grey backgrounds such walls and roads. In the experiment, this method could remove shadows in indoor and outdoor environments. Cucchiara et al detected shadows by analysing pixels in the Hue-Saturation-Value (HSV) colour space (Cucchiara et al., 2003). This colour space separates chromaticity and luminosity. If pixels belong to shadows, the Hue component changes compared with the background, but within certain limits. In addition, the difference in Saturation must be an absolute difference, but the difference in Hue is an angular difference. They concluded the

shadows could be detected and removed from the background update function. Another approach was to use the features of the shadow to detect it (Tian et al., 2016). They analysed sRGB colour matching functions (CMFs) and spectral power distribution (SPD) of illumination to obtain four properties (as features) of shadows. Then they identified the shadow edge by using edge extraction. Thus, the shadow region could be detected. The experiment indicated this method was robust, but the average run time was 39.7s. The limitation was that the method failed in overexposed regions of an outdoor image.

The shadow can also be removed without using colour information. Jacques et al removed shadows on a grayscale image (Jacques et al., 2005). They focused on two types of shadows: one was produced by obstruction of indirect light and the other was caused by direct blocking of sunlight. The shadowed pixels were darker than the corresponding pixels in the background model. The detection method was the normalized cross-correlation (NCC) which could distinguish the darker pixels from the corresponding background pixels. Then, a refinement technique was applied to modify shadows by using local statistics of pixel ratios between the current image of the video sequence and the background model. In their experiment, there was some shadow misclassification because of the dark part of the object, but they reported this was not common. Another example was to combine contour projection analysis and shape analysis to remove the effect of shadow (Rakibe and Patil, 2013). The horizontal and vertical projections were analysed to get the height of the motion region after their background subtraction. This could eliminate the shadow effectively.

Shadow detection and removal has attracted a lot of attention as part of background subtraction. Despite this, there is no general method which is robust and fast. There are different methods corresponding to different situations. In summary, the common method is to use the colour information to remove the shadow.

2.3 Object tracking

The goal of object tracking on video is to identify the same object in different video frames. The object detection process will have already identified some object features such as colour, texture and shape. Position is also an important feature, especially when similar objects are being tracked, such as people, cars or insects. Some other features like size and orientation are also useful for tracking. The common tracking methods are the particle filter and Kalman filter. The Kalman filter is effective for tracking objects with linear movement, and the particle filter is more effective for tracking objects with non-linear movement.

2.3.1 Object blob analysis for tracking

Object detection normally generates a binary image which displays the foreground object as white (binary one), and the background as black (binary zero). The group of white pixels is the blob of the object. The object blob can be analysed to calculate blob features such as area (the number of pixels in the blob region), height (length of the major axis of the blob), width (length of the minor axis of the blob), orientation (angle between the major axis and the x-axis) and position (the coordinates of the centre point). Commonly, blob analysis is performed by calculating geometric moments of the binary image. A summary of the geometric moments was presented in 2005

(Kotoulas and Andreadis, 2005). The moments were used for aircraft identification, scene matching, shape analysis, image normalization, character recognition, accurate position detection, colour texture recognition, image retrieval, etc. An example is to utilize image moments to analyse blobs for tracking a person's hands and face (Lu et al., 2005). In this approach, a colour distribution method was used in the first step to extract the blobs of face and hands. The image moments of these blobs were calculated to obtain the shape and motion parameters. They used central, first and second order moments to characterise the blobs. The regions of the face and hands were determined by the ellipse parameters such as position, angle and aspect ratio between short and long axes. Another blob analysis method is described in 2007 (Thou-Ho et al., 2007). They tried to tracked vehicles using blob analysis. They analysed object blobs to obtain the features of perimeter, blob area, and bounding-box information such as height and width. These features were used to calculate the dispersedness ($\frac{Perimeter^2}{Area}$), aspect ratio ($\frac{Height}{Width}$) and area ratio ($\frac{Area}{Height*Width}$) which were used as temporal features for recognising the same object at different times. In addition, they calculated the centroid coordinates of each object for tracking.

Blob features such as position and orientation are useful for tracking, especially the blob position, because it can be used by the Kalman filter. Lu et al used the position and velocity to track the hands and face (Lu et al., 2005). They employed a Kalman filter to predict the location. Then, the predicted location was combined with the measured location to calculate the current location as a weighted average of the predicted and measured locations. If the hand blob existed in the current frame, the observed location was updated using the current hand blob. However, if the hand blob disappeared, the observed location was updated from the previous hand blob location. In addition, the orientation of the head blob was used to analyse the motion of a head gesture. The orientation was calculated as the angle between the Y-axis and major axis of the head blob. Another example was to apply blob analysis to track cars and pedestrians (Telagarapu et al., 2012). Detection was based on the different characteristics of the car blobs and pedestrian blobs. The blob analysis eliminated blobs that were of no interest, based on their spatial characteristics, and kept only the relevant blobs for further analysis. The cars were tracked after the blob analysis, and pedestrians could be differentiated using blob analysis concepts. Another approach was to extract bounding-boxes and centroids for tracking by using blob analysis (Thou-Ho et al., 2007). Then, objects in successive frames were identified as the same object, based on how close they were and how similar their sizes were. Euclidean distance was utilized to measure the distance between their centroids. In addition, the area of a vehicle was also used for enhanced tracking. Objects were identified as the same by searching for the minimum distance between blobs and similar sized blobs in two consecutive frames. This method could also count the vehicles and calculate their velocities.

It can be seen that analysing the object blob can produce spatial features for the object. In summary, blob analysis is an important preparation for object tracking.

2.3.2 Object tracking with a Kalman filter

The Kalman filter has been widely used in video based object tracking. It predicts the next location of the object, then combines it with the next observed location, and allows a refined location to be

calculated from the prediction and the observation. The discrete Kalman filter was introduced in 2006 (Welch and Bishop, 2006). The Kalman filter includes prediction and measurement. It provides a recursive solution of the least-squares method and allows tracking different objects using estimations of past, present and future states. The Kalman filter can also implement non-linear object tracking (Extended Kalman filter) by employing the Jacobian matrix of partial derivatives of transition matrices and noise.

In object tracking, the discrete linear Kalman filter is commonly used. The Kalman filter predicts the moving location of the object. It updates the location from the prediction and measurement. Gao et al presented a multi-Kalman filtering approach to track a single rigid object (Gao et al., 2005). Initially, they extract feature points of the object on the initial frame at its first-time appearance. Then these feature points are grouped together for Kalman-based updating of motion parameters. After the current feature extraction, the motion was updated from the feature prediction and extraction. This approach kept track of the image feature points from the time they appeared until they disappeared due to occlusion. This method provided a 3D motion estimation scheme for object tracking with emphasis on solving the occlusion problem, even when the object was rotating relative to the camera. Weng et al developed an adaptive Kalman filter for object tracking, which was a simple and efficient method (Weng et al., 2006). They used moving object segmentation and dominant colour features to detect the object. In the tracking procedure, a motion model was constructed for the system state and was applied in the prediction step. The measurement for the filter was provided by moving object detection in HSI colour space. The ratio of moving object area in current frame to the previous frame was used as the object occlusion ratio. The occlusion ratio was calculated and applied to adjust the prediction and measurement errors adaptively. This method could track an object in real-time, even in the multiple tracking situation. In addition, it successfully estimated the object's position in some real-world situations such as fast moving objects, partial occlusion, long-lasting occlusion, changing lighting, changing direction and orientation of the moving object, and sudden velocity changes. Zhang et al presented a Bayesian Kalman Filter (BKF) method for visual object tracking (Zhang et al., 2014). They employed the Gaussian Mixture (GM) algorithm to represent the state and noise densities and simplified the algorithm (SGM) to reduce complexity. The mean shift (MS) tracker was used to obtain the observation vector for measurement. It was also improved (IMS) in the paper to reduce the possibility of the MS tracker getting trapped in a local maximum point on the background or other similar objects. The result was a more accurate measurement. This BKF-SGM-IMS visual object tracker was proposed to handle complex scenarios with good performance and low arithmetic complexity. Another example of using Kalman filter was to track the motion of a segmented region (Sanchez-Garcia et al., 2014). They segmented the moving object using the optical flow method for each pair of frames. Each object was divided into polygon shaped regions. Then the Kalman filter was used to track the polygons through the sequence of images. The experiment showed that this Kalman filter was a robust model for moving object tracking.

The Kalman filter is a fast approach for real time object tracking. The state vector is the movement information, including any motion features such as location, velocity and acceleration. It is a commonly used method in computer vision.

The Kalman filter has also been used in non-linear tracking, such as movement of human eyes, hands and head, especially for poses and gestures. One way to solve the non-linear problem is linearization of the non-linear function. The Kalman filter could be used to track human eyes moving on a video (Xangdong et al., 1995). In this approach, the centre of the iris was chosen as the tracking parameter vector, because the iris provided a visual assessment of eye movements. The grey level centroid of the eye region was chosen as the measurement vector. Then the model related the measurement to the tracking parameters. The nonlinear measurement equations were linearized to reduce the computation. The Kalman filter was employed to track the eye feature. The Kalman filter was chosen in this paper because it was an efficient recursive procedure requiring a minimum amount of storage for the past samples, it embodied the information about the system and measurement noise in its model, it effectively dealt with time varying signals, the result of the previous step was used to predict the current states, and the accuracy of the estimation was assessed by monitoring the error covariance. The experimental results showed that the method could be successfully applied to smooth eye movements. This approach was suitable for real-time applications, although the initialization procedure took a long time.

Another technique is the non-linear Kalman filter, such as the Extended Kalman filter (Welch and Bishop, 2006) or the Unscented Kalman filter. The unscented Kalman filter could be used to track the human hand and estimate the pose of a 3D hand model (Stenger et al., 2001). The 3D hand model was built from quadrics which represented the anatomy of a human hand. They used a hierarchical model with 27 degrees of freedom (DOF) which was constructed from 37 truncated quadrics. Each clipped quadric of the hand model was projected individually generating a list of clipped conics. The unscented Kalman filter was a filter for nonlinear tracking which fitted the hand pose tracking. The state vector included the hand pose and hand motion (such as velocity and acceleration) and had 27n dimensions, where n-1 was the order of the dynamic model. The observation vector was obtained from edges in the neighbourhood of the projected hand model. The filter minimised the geometric error between the projection of the hand pose model and the edges detected in the image. The results demonstrated the efficiency of the method. Another example of utilizing Unscented Kalman filter was to track human hand gestures (Li et al., 2004). The measurement model of the contour of the hand was non-linear, but the unscented transform could predict the mean and covariance precisely up to the second order and did not require linearization of the nonlinear function, such as derivation of Jacobians or Hessians. During each time step, the tracking model made multiple measurements in terms of the set of appropriately chosen sample points. Therefore, it could obtain the best observation according to the measurement probability function. They also applied the method to head tracking with a nonstationary background. The results indicated that it was more accurate and robust than normal Kalman filter tracking. However, if the background contained heavy clutter, the state density will be multi-modal. The unscented Kalman filter could not work well on the multi-modal distribution.

2.3.3 Object tracking with a particle filter

The principle of the particle filter is to sample the object pixels in the first frame, and then resample pixels around the object in the next frame, and try to find similar features in the pixels. A classical application of the particle filter for object detection was presented in 2006 (Dearden et al., 2006). They tracked one soccer player on a pitch. They created a system that enabled players to be automatically tracked from a single, moving camera. The background of the pitch was a green colour. They segmented the player region using histogram thresholding in HSV space, and the pixels were grouped into regions. Then a binary image was created of non-pitch colours within the pitch region showing possible player positions. They used the sample importance resampling (SIR) particle filter to perform the tracking. The particle filter estimated and maintained the position of the tracked player. The non-linear and non-Gaussian nature of a particle filter could track players in spite of occlusions by other players. Another application of particle filter was to track multiple soccer players (Vermaak et al., 2003). They found the general particle filter was unsuccessful at consistently maintaining the multi-modality in the target distribution, because of insufficient measurements, or clutter, or measurements from multiple objects producing ambiguity in the tracking. To solve this problem, they introduced a strategy to model the target distribution as a non-parametric mixture of filtering distributions. The mixture of filtering distributions was computed recursively in two steps: a prediction step followed by an update step when the new data became available. The Monte Carlo implementation of the general recursion led to a mixture of particle filters that interacted only in the computation of the mixture weights. This mixture particle filter maintained the multi-modality inherent in multi-tracking problems.

The particle filter is widely used in object tracking systems. Much research focuses on improving the computational problem. Changjiang et al mentioned the trade-off of using the particle filter: the more samples and richer the target representation, the better chance that the tracking algorithm succeeds in cluttered and noisy environments, but the more computational load is required by the filter (Changjiang et al., 2005). Their solution was to simplify the features that they used. The features tracked were the colour histogram and edge orientation. The computation of the colour histogram was expensive if the tracked region and the number of samples were large. They employed Harr-like rectangle features which could be efficiently evaluated by several table lookup operations on an “integral” image. The “integral” image is an image in which each pixel’s value is the sum of all pixels above and to the left of the current position (Viola and Jones, 2004). To overcome the problem of illumination changes, they used an edge orientation histogram (EOH) which also discriminated against a background with confusing colours. In addition, they improved the convergence of the Monte Carlo integration in the particle filter by using a quasi-random generator to generate the sample points. They concluded that their improvements made the tracing algorithm efficient and robust against clutter, illumination changes and short period time occlusions. The particle filter could be improved by using Rao-Blackwellization (Zia et al., 2004). They used subspace representations as the measurement model of the particle filter. They employed Rao-Blackwellization to integrate out the appearance subspace coefficients of the state vector, leaving only the original target state. Fewer samples were needed since part of the posterior over the state was analytically calculated, rather than being approximated using a more

expensive sample representation. After that, the integral could be computed analytically by using probabilistic principal component analysis (PPCA). They concluded this method reduced the number of particles needed and the filter increased tracking performance and decreased the number of tracking failures. The advantage of the particle filter is that it tracks objects effectively during non-linear movement. The particle filter was more efficient and robust than the EKF in the nonlinear and non-Gaussian situation (Arulampalam et al., 2002). In addition, Iqbal et al compared three motion object tracking methods: mean-shift, Kalman filter and particle filter (Iqbal et al., 2014). Their test environment was indoors, and the tracked objects were a ball moving linearly and a woman moving non-linearly. The three methods were all effective for the linear tracking. However, the mean-shift and Kalman filter were less effective than the particle filter for non-linear tracking. The mean-shift is based on a histogram and rapid change disturbed the tracking, and the Kalman filter depends upon the previous state of the object to estimate the current state. Conversely, in the particle filter, the tracked object is being represented by a possible location denoted by a set of weighted particles.

2.3.4 Other tracking methods

The Kalman filter and particle filter are well-known tracking algorithms. There are other tracking methods which are not so commonly used, but they have different advantages.

The mean-shift algorithm used a colour based statistical model to estimate of next position of the target (Iqbal et al., 2014). The main features of the mean shift algorithm are mode seeking, clustering density estimation and tracking. The mean-shift algorithm iteratively shifts data points in its neighbourhood and locates the new position of the target using density estimation and the colour histogram.

The adaptive local movement model (ALMM) was used to track a single object (Zhang et al., 2015). The basic idea is that the distribution of the movements of the local patches were easier to model instead of the whole object motion. In this method, the local patch centres were related to the centre of gravity of the whole tracked object. They first tracked the local patches to calculate the position and the validity of each image patch. In the second step, the locations of patches were further corrected by using an outlier detection process based on GMM that pruned the patches which diverged from the current statistics. Finally, they assigned a weight to each patch, with the patch tracking, to decide whether a patch should be kept or discarded in computing the centre of gravity. The experiment results showed that this method was robust for the problems of occlusion, fast motion and texture variations.

A two dimensional hashing algorithm could be used to represent objects (Ma and Liu, 2015). The hashing method could improve the speed of similarity measurement which could be utilized to track an object. To overcome the time-consuming hash function training, they proposed a two dimensional hashing method which reduced the dimensionality of the covariance matrices, and then the speed of training the hash function was improved. During the tracking, samples and templates were hashed to binary matrices, and the Hamming distance was used to measure a confidence of candidate samples. The sample with highest confidence was chosen as object in the current frame. Then hash functions were updated by an incremental learning model to adapt

to changing situations. Through these innovations, this tracker obtained improved performance in accuracy and efficiency. The authors conclude this method could also be implemented and applied to many other applications in computer vision.

2.3.5 Multiple object tracking problem

The main tracking problem is to identify the same object in the frame flow of a video. If only one object is visible on the video, it is easy to track. However, the identification of multiple similar objects requires mathematical calculation. The simplest idea is to track using the distances between objects on successive frames, but this is unreliable if the objects move very fast. Filtered tracking produces predicted positions for the tracking, and then using the distances between predicted and detected positions improves the tracking accuracy. However, assignment of predictions to detections is still often a problem.

The Hungarian method can be used for this problem. Nandashri and Smitha used the Hungarian method for multiple tracking (Nandashri and Smitha, 2015a). Initially, they employed GMM to extract the moving object, and then the Kalman filter was utilized to predict the next detection in the following frame, and to correct the locations from these predictions and object detections. The Hungarian method was finally used to assign detections to predictions in the process of tracking the multiple objects. This process also determined which objects had disappeared from the video and which detections should begin a new tracking process. They showed the final result by tracking two people walking in opposite directions. Their model also solved the problem of occlusion that occurred between the individual objects being tracked. Tashita et al proposed a tracking method for time-lapse macrophages based on the Hungarian algorithm (Tashita et al., 2015). The method was applied to mouse brain magnetic resonance (MR) images. They utilized a temporal background subtraction method to detect macrophages. After that, they performed tracking in each frame image using the Hungarian method. They minimized the movement distances of macrophages between frames. The result showed that the method successfully tracked the macrophages. They also reported the method can be enhanced to 3-D processing by detecting macrophage candidates and tracking in 3-D space.

The Hungarian algorithm is also useful for multiple object tracking in computer vision. It can also help to identify the appearance of objects new to the video frame flow and the disappearance of objects from the frame flow. In this research, it is used with the Kalman filter in the tracking process.

2.4 Small object tracking

The aim of this research is to detect and recognise pollen sacs on the bodies of flying bees. Although high resolution video is used, the pollen sacs are still small objects on the video. The definition of small object detection is to detect a special small region on a much larger image, such as a ping pong ball, a logo on a T-shirt or part of the human body. Although the object area may be too small to capture directly, some papers have reported detecting small object regions using various methods.

2.4.1 Direct detection of small object regions

Some papers have reported the detection of small objects on large images. One example was to combine wavelet based detection and gradient based detection for small object detection (Desai et al., 2005). Both of the two methods estimated the motion. In addition, they also tracked the small targets with a multiple filter bank which included two filters of constant acceleration and Signers' manoeuvre model. They applied the method to track cricket and ping pong balls in sports videos. Furthermore, the method could also track airborne targets in an IR image sequence. Moreover, it also detected point targets. The experimental results showed that it could detect objects on low contrast and negligible texture backgrounds. Another approach to detect small objects in a low contrast video sequence was presented in 2006 (Hsieh et al., 2006). The frame differencing technique was first utilized on two consecutive image frames to generate candidate objects. However, the low signal to noise ratio (SNR) was a serious problem for the detection of small objects with low contrast image sequences. To solve this problem, they developed a noise removal algorithm by encoding every pixel and its neighbours according to the noise distribution. Then, they located the regions of interest (ROI) by thresholding the de-noised image. After that, a segmentation algorithm and region matching technique were used on the ROI to extract the object contour. A watershed based algorithm was devised on the ROI to extract the final object contour. In addition, they performed a watershed transformation with a lower value of the drowning threshold to detect low contrast objects. Baojun et al detected small maritime object on IR images (Baojun et al., 2011). They used wavelet transforms to generate local minimum patterns (LMP). The LMP estimated the background and produced a saliency map. Then the regions of candidate objects were segmented using an adaptive threshold based on the histogram of the saliency map. The regions of small objects were finally detected using a fast clustering algorithm. The experiment indicated that the method could detect multiple objects of various scales and objects in highly noisy images. An active contour model was presented to detect small objects from cluttered and textured backgrounds in 2012 (Vard et al., 2012). The objects were as small as lighting points. Image features were obtained using the normalized accumulated short-term autocorrelations (NASTA) function. The features were exploited to define an energy function of the contour model which could detect the small objects. The NASTA was calculated from the image pixels to represent region information. The values of NASTA were exploited instead of intensity as a feature to characterize the pixels in the model. They gave examples to show that the NASTA values were better than the intensity values for detecting small objects in a cluttered background. Qi et al reported small surface object detection from depth-aware analysis of image features using the atmospheric scattering model on video of maritime scenes (Qi et al., 2011). The suspicious small surface objects were located with the dark channel image. The small surface objects, only a few pixels wide, could be enhanced in the dark channel image. After that, they detected the horizon using dyadic cubic spline wavelet transforms and estimated the scaled depth map of the sea surface base on the theory of perspective projection. They identified the final small surface objects by obtaining spatial-variant thresholds using the atmospheric scattering model and the estimated depth map. This algorithm had a more than 90% positive rate. However, it could not work if objects were similar to atmospheric light, because of the invalidation of the dark channel prior. Wei detected objects of small size and fast motion in Infra-Red (IR) sequences,

referred to as targets of primary interest (TPIs) (Wei, 2013). Initially, he employed a 2D band stop filter in the spectral domain of each frame to reduce system noise. The next step was to apply the pixel process algorithm to detect the small size and fast motion of TPIs. The candidate TPIs were further refined after spatial post-processing on each frame, using size filtering and component labelling. Finally, TPIs were declared after a consistency check across adjacent IR frames by use of the adaptive Hough transform.

In all of the small object detection papers, the small object was obvious on the background. Both (Wei, 2013) and (Baojun et al., 2011) show their results using infrared images, where the object was a white point on the background before the detection. The small points in (Vard et al., 2012) were also light points which were obvious against the dark background around them. The backgrounds in (Desai et al., 2005), (Qi et al., 2011) and (Hsieh et al., 2006) were simply sky or ocean, and it was easy to detect airplanes or ships in those situations.

2.4.2 Segment the big object region in small parts

Another method of small object detection is to detect a bigger object region, then detect the small object inside the bigger region. Crandall et al detected different parts of an object using a special statistic model (Crandall et al., 2005). They used edge maps obtained using the Canny method to calculate the foreground and background probability of pixels. Then they introduced a K-fans method based on a full joint Gaussian model and a tree-structured model to analyse spatial priors. They trained for different objects such as motorbikes, airplanes and faces, and collected the spatial structures of them. The spatial structure of the image produced the final part detection and localization. Wang et al segmented semantic objects to understand the detail of the object (Wang et al., 2015c). They proposed a joint solution that tackled object and part segmentation simultaneously, in which a higher level object context was provided to guide the part segmentation. In addition, they utilized a more detailed part-level localization to refine the object segmentation. Firstly, they introduced semantic compositional parts (SCP) in which similar semantic parts were grouped and shared among different objects. A two-stream fully convolutional network (FCN) was then trained to provide the SCP and object potentials. Moreover, a compact set of segments was also obtained from the SCP predictions of the network at the same time. Finally, they constructed a fully connected conditional random field (FCRF) to jointly predict the final object and part labels. The evaluation showed that this method could mutually enhance the performance of object and part segmentation.

Some research detects parts for bigger object detection. The parts in these researches can share their region with other parts which belong to same object. Wu and Nevatia reported using part base detection to detect human body parts (Wu and Nevatia, 2007). The part detectors learned from a set of silhouette oriented features which they called edgelet features. These features were suitable for human detection as they were relatively invariant to clothing differences. The detected human body parts included head-shoulder, torso and legs. The aim of the paper was to detect and track a person walking on the road. The whole human body detection combined the various part detectors. They defined a joint image likelihood function for multiple, inter-occluded humans. The tracking method was also based on tracking parts of the human body. The advantage of part detectors was that it could track partially occluded humans. Fulkerson et al detected superpixels

on the pixel level (Fulkerson et al., 2009). The superpixels were the regions of parts of the object. They performed quick shift using a five-dimensional vector composed of the LUV colour space representation of each pixel and its location in the image. Then, they built a superpixel classifier into a conditional random field (CRF) to perform the whole object segmentation. It could recognise the neighbourhood of two objects, such as a human riding a horse, a dog and a cat together, and table and chairs together. A training method for detecting parts of an object was presented in 2016 (Kuang et al., 2016). They applied k-means clustering to generate a “seed” for training the initial detectors. The whitened histograms of oriented gradient (HOG) features were clustered for the regions of parts on the image. They sampled densely to get all possible part images and collected hundreds of thousands of parts. These were the mid-level part detectors. They finally selected the parts using the max-pooling technique to detect the whole object.

2.5 Deep learning for computer vision

Deep learning is a class of machine learning algorithm, which has become very powerful and popular for computer vision research recently. It creates computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction (LeCun et al., 2015). Deep neural network (DNN) is one of the deep learning architectures, which is an artificial neural network with multiple layers between the input and output layers (Bengio, 2009, and Schmidhuber, 2015). The DNN is trained to find the correct mathematical manipulations to turn the inputs into the outputs. After training, the network is tested with new inputs to calculate the probability of each output. Then, the user can review the results and select which probabilities the network should display (for an example: probability above a certain threshold) and return the proposed result. A complex DNN has many layers, hence the name “deep” networks.

For computer vision, the most common DNN is convolutional neural networks (CNNs) (Lecun et al., 1998). CNNs are designed to process data that come in the form of multiple arrays, for example a colour image composed of three 2D arrays containing pixel intensities in the three colour channels. There are four features behind CNNs: local connections, shared weights, pooling and the use of many layers (LeCun et al., 2015). The architecture of a CNN is structured as a series of stages. The first few stages are composed of two types of layers: convolutional layers and pooling layers. Units in a convolutional layer are organized in feature maps, within which each unit is connected to local patches in the feature maps of the previous layer through a set of weights called a filter bank. The result of this local weighted sum is then passed through a non-linearity such as a rectified linear unit (ReLU) method (Nair and Hinton, 2010). All units in a feature map share the same filter bank. Different feature maps in a layer use different filter banks. A pooling layer then merges semantically similar features into one. A typical pooling unit computes the maximum of a local patch of units in a feature map. Neighbouring pooling units take input from patches that are shifted by more than one row or column, thereby reducing the dimension of the representation and creating an invariance to small shifts and distortions. Two or three stages of convolution, non-linearity and pooling are stacked, followed by more convolutional and fully-connected layers. Backpropagation gradients (Rezende et al., 2014) through a CNN are as simple as through a regular deep network, allowing all the weights in all the filter banks to be trained.

In the 1990s, CNNs were developed for object detection in natural images, including faces and hands (Vaillant et al., 1994, Lawrence et al., 1997). Since the 2000s, CNNs have been applied with great success to the detection, segmentation and recognition of objects and regions in images. These were all tasks in which labelled data was relatively abundant, such as traffic sign recognition (Cireşan et al., 2012), the segmentation of biological images (Feng et al., 2005) and the detection of faces , text, pedestrians and human bodies (Sermanet et al., 2013b, Garcia and Delakis, 2004, Osadchy et al., 2007, Tompson et al., 2015). Despite these success, CNNs were largely forsaken by the mainstream computer-vision and machine-learning communities until the ImageNet competition in 2012 (Russakovsky et al., 2015). Since then, many trained CNNs have been created for image classification, such as Alex network, VGG, GoogLeNet and ResNet, etc. The Alex network was introduced in year 2012 (Krizhevsky et al., 2012). It includes 60 million parameters and 650000 neurons, consists of eight layers: five convolutional and three fully-connected layers. Some of the convolutional layers are followed by max-pooling layers. The output of the last fully-connected layer is fed to a 1000-way softmax method which produces a distribution over the 1000 class labels. The data augmentation and dropout method in the network are significant innovations to prevent overfitting. The network won the ImageNet large scale visual recognition challenge in 2012. The VGG network is similar to the Alex network in structure, but it includes more layers (Simonyan and Zisserman, 2014). This network increases depth using an architecture with very small (3x3) convolution filters. The depth can be built to 16-19 weight layers. The experiment result in ImageNet Challenge 2014 was much better than the Alex network. It demonstrates that the representation depth is beneficial for the classification accuracy, and better results on the ImageNet challenge dataset can be achieved using CNN architecture with substantially increased depth.

GoogLeNet was presented in 2015 (Szegedy et al., 2015). It introduced an inception architecture in the network, which allowed increasing not only depth (the number of network levels), but also width (the number of units at each level) without getting into too much computational difficulty. The network was 22 layers deep when counting layers with parameters. In the network's structure, two convolutional layers followed the input layer, and then inception modules were created and located in following layers. This could handle the image scaling. After the inception layers, the fully connected layers (in the VGG network) were replaced by an average pooling layer, which improved the accuracy. This network won the ImageNet Challenge 2014 in the experiment test, performing even better than the VGG network. ResNet includes the residual learning framework in the network, which solves the problem of deeper neural network training (He et al., 2016). Adding more layers to a suitably deep model leads to higher training error (He and Sun, 2015, Srivastava et al., 2015). ResNet was managed to overcome the optimization difficulty and demonstrated accuracy gains when the depth increases. It increased layers from 18 to 152. The 152 layers produced the best result in the experimental results. This network won the 1st place not only on the ImageNet Challenge 2015, but also on the task of COCO detection and COCO segmentation (Lin et al., 2014). In summary, when deep convolutional networks were applied to a data set of about a million images from the web that contained 1000 different classes, they

achieved spectacular results, almost halving the error rates of the best competing approaches (Krizhevsky et al., 2012).

Apart from CNNs becoming successful in image classification, more and more researches also focused on object detection and image segmentation. Particularly, multiple object detection has become a popular research topic since 2014, when the Region-Based CNN (RCNN) was created. For applying CNN in object detection, some researchers built object region proposals for the object detection network. One example was the method of Regions with CNN features (RCNN) which separated the detection problem into two stages (Girshick et al., 2014). The first stage was to utilize low-level cues such as colour and texture in order to generate object location proposals in a category-agnostic fashion. The second stage was to use a CNN classifier to identify object categories at those locations. Such two stage approaches leveraged the accuracy of bounding box segmentation with low-level cues, as well as the highly powerful classification power of state-of-the-art CNNs. The experimental results indicated that this detection algorithm achieved more than 20% mAP (the mean average precision) compared with their previous method, which was standard HOG-based DPM (Girshick et al., 2012).

The Fast RCNN was built on RCNN, but is more efficient than RCNN (Girshick, 2015). It takes an entire image and a set of object proposals as input. Then, the network processes the whole image with convolutional layers to produce a convolutional feature map. At this stage, the network trained the very deep VGG-16 network for higher detection quality. After that, for each object proposal, a region of interest (ROI) pooling layer extracts a feature vector from the feature map. Each feature vector finally produced probabilities estimating different object classes as well as a “background” class. It also outputted bounding box positions for each of the object classes. Compared to the RCNN, the Fast RCNN improved training and testing speed while also increasing detection accuracy.

Faster RCNN was the next version of Fast RCNN(Ren et al., 2015). In this network, a region proposal network (RPN) was created, which shared full-image convolutional features with a detection network (such as Faster RCNN). The RPN was a fully-convolutional network (FCN) (Long et al., 2015) which predicted object bounds and object-ness scores at each position. This network was trained end-to-end to generate high-quality region proposals, which was used by Fast RCNN for detection. With a simple alternating optimization, RPC and Fast RCNN could be trained to share convolutional features. The conclusion was that the RPN was an efficient and accurate region proposal generation network. By sharing convolutional features with the down-stream detection network, the region proposal step was nearly cost-free. The RPN improved region proposal quality, so that the overall object detection accuracy was also increased.

Mask RCNN is the latest version of the RCNN network (He et al., 2017). This network has two stage procedures, which is similar to Faster RCNN. The first stage is the RPN network. In the second stage, the approach followed the spirit of Fast RCNN that applied bounding-box classification and regression in parallel. This turned out to largely simplify the multi-stage pipeline of the previous RCNN method. The Mask RCNN also outputted a binary mask for each region of interest (ROI). The mask encoded an input object’s spatial layout. Thus, unlike class labels or box

offsets that were inevitably collapsed into short output vectors by fully-connected layers, extracting the spatial structure of masks could be addressed naturally by the pixel-to-pixel correspondence provided by convolutions. The Mask RCNN was faster than Faster RCNN and improved the object detection results.

In summary, the different types of RCNN networks include two stages: object region proposal and object detection. Faster RCNN and Mask RCNN include two networks for object detection: the RPN for generating object proposals and an object detection network (such as Fast RCNN).

Some researchers have tried to use a single deep neural network for object detection. One deep learning method was single shot multi-box detector (SSD) (Liu et al., 2016). It discretized the output space of bounding boxes into a set of default boxes with different aspect ratios, and scales per feature map location. In prediction, the network generated scores for each object category in each default box and produced adjustments to the box to better match the object shape. The network also combined predictions from multiple convolutional feature maps with different resolutions to handle objects of various sizes. The SSD eliminated proposal generation and subsequent pixel or feature resampling stages, so that all computation is in a single network. The test results showed that the SSD was outperformed the Faster RCNN in terms of accuracy as well as detection speed.

Another single deep learning network for object detection on images is YOLO (Redmon et al., 2016). YOLO (You Only Look Once) was designed as a model which separated the input image into a grid (*SxS* grid for an example). If the centre of an object was in a grid cell, that grid cell was responsible for detecting that object. Each grid cell predicted a number of bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model was that the box contained an object and also how accurate it thought the box was that it predicted. The confidence score related to the intersection over union (IOU) between the predicted box and the ground truth. If no object existed in that cell, the confidence score was zero. Each bounding box consisted of 5 predictions: x, y, w, h and confidence. The (x, y) coordinates represented the centre of the box relative to the bounds of the grid cell. The w and h represented the width and height of the bounding box respectively. This model was implemented in a convolutional neural network. The initial convolutional layers extracted features from the image, while the fully connected layers predicted the output confidence and coordinates. This network was also much faster than Faster RCNN, and it did not lose accuracy. The latest version of YOLO v3 (Redmon and Farhadi, 2018) had better results for the small object detection and is much faster than previous versions.

In summary, the single network can perform very fast object detection, because they do not have the object proposal before detection. The detection can be real time on the 30 frame rate video.

Deep learning has become very a powerful method for computer vision since year of 2012. The CNN was becoming more and more deep from 2012 to 2016, while the image classification was more and more accurate. Especially, the structure of ResNet (He et al., 2016) included more than 100 convolutional layers. Object detection on entire images was also growing fast from 2015 to current year (2018). From the combination of object proposal and object detection network (Faster

RCNN) to single network object detection (SSD and YOLO), object detection became faster and faster with accuracy increasing. There are many other networks created for object detection such as Deformable Parts Models (DPM) (Felzenszwalb et al., 2010), SPPnet (He et al., 2014), Deep Multibox (Erhan et al., 2014) and Over Feat (Sermanet et al., 2013a). Deep learning is more and more popular in computer vision field. In this thesis, the deep learning method is applied pollen sac detection and measurement. It is also compared to the traditional image processing method.

2.6 Bee and pollen sacs detection in computer vision

In this research, computer vision technology is applied to a video that is monitoring bees. There are papers reporting detection and tracking of honey bees. Some papers focus on surveillance inside the beehive. A method of vector quantization (VQ) for identifying and tracking bees in the hive was introduced in year 2011 (Kimura et al., 2011). VQ separated honeybee bodies from one another in time series photographs without any preconditions. A honeybee-code image was extracted from the whole hive image. The average body size and shape was used to separate regions of single and multiple honeybees from the honeybee-code image. After that, honeybee movement was tracked from temporal contextual information in the sequential image data. This method analysed a large number of frames quickly and correctly tracked more than 72% of the individual bees in the hive. In addition, it was able to partly resolve the problem of overlapping honeybees. Another approach was to use the region homogeneity feature to detect bees in the hive (Knauer and Meffert, 2010). The brood cells appeared as homogeneous image regions, while the regions which were covered by a crowd of bees appeared as non-homogeneous. They improved the region homogeneity algorithm efficiency and speed by applying the concept of integral image. They also segmented regions which were not covered by bees with an adaptive thresholding method. However, this method falsely detected homogeneous regions in the outdoor situation, such as a street surface. Knauer et al reported an adaptive background model to monitor bees in the hive (Knauer et al., 2005). In the first step, an adaptive background model based on the GMM was used to detect uncapped cells on the honey comb crowded with bees. Then canny edge detection was applied to detect the contour of the open cell. The contour was then analysed to identify the uncapped cells. Once the cells were known, the bees were easy to detect and track. A behaviour model was presented to track bees and to recognise specific behaviour in year 2008 (Veeraraghavan et al., 2008). This model used the factors of position, orientation and current behaviour to perform the tracking automatically. The bee body was modelled as three ellipses in a shape model. The position and orientation of each part was calculated. Then a behaviour model was created using a three-tier hierarchical motion model. The first tier built a local motion model for the vocabulary of behaviour. The second tier built a Markov model on top of the local motion vocabulary. The final tier modelled the switching between behaviours based on the Markov model. This method could be extended to the behaviours of other insects and to human activities. Hendriks et al used tags which were glued on the bees' bodies to assist tracking (Hendriks et al., 2012). The tags were designed for image processing. The tags had white dots for the detection system and included symbols showing the direction of the head of the bee. However, an inconvenience in the experiment was that the tags could only be stuck on bees manually. Knauer and Meffert compared and combined five classifiers for monitoring honeybees (Knauer and

Meffert, 2009). The five methods included: edge detection and classification of contours, rapid detection of Haar-like features using AdaBoost, detection of clusters of interest points, the Hough transform, and template matching. They evaluated the five methods using the Receiver Operating Characteristic (ROC) and Precision and Recall (PR) techniques. Contour detection produced the best results for both ROC and PR. Then they combined the methods based on the complementary information. They selected the best performing method (contour detection) as the main method. Then, they looked for high precision methods which can detect some additional objects and were also reliable in doing this. For example, the Interest Point Density Detector compensated for some errors in the contour detection. Conversely, the Hough transform was discarded because the results reduced the precision.

Several papers focus on monitoring the flight of bees outside the beehive. Motion models could be used to detect and track the flight of bees in the uncontrolled outdoor environment of an apiary (Campbell et al., 2008). An adaptive background subtraction method was utilized to detect motion regions. Then, a motion template was built for the bee's different behaviours. Crawling and flying bees were modelled by a Gaussian distribution. Loitering was detected with a small distance threshold. After this, tracking was performed by using a method of maximum weighted bipartite graph matching to assign bees from frame to frame. This approach was applied to measure the flight activity of honey bee colonies. Kimura et al reported a method to detect and track bees on a flat surface (Kimura et al., 2012). It employed the background subtraction method to detect candidate bee regions. Then they identified individual bees using a combination of two methods: the overlapping information in temporal changes of position, and the prediction of the candidate regions based on the bee's movement. Then, the locations and trajectories of the bees' movements were identified. However, because the movement was only a few pixels, if the bee moved fast, the tracking failed. 3D stereo vision systems have also been used to detect and track bees (Chiron et al., 2013a, and Chiron et al., 2013b). The 3D camera recorded video from two viewpoints. One view was an intensity video which was used detect the bees' flights with background subtraction. The other view was in colour and the colour histogram method was utilized to detect bees on this frame. Bees were detected from a combination of both views. In addition, they tracked bees using a Kalman filter.

Pollen detection and measurement has been reported in very few researches. One research has been found for detection of pollen bearing bees (Babic et al., 2016). In their research, a special hardware platform which consisted of a wooden box (sensing box) with a Raspberry Pi camera module inside, was mounted on the front side of a standard hive, above the hive entrance. The sensing box was closed on all sides except for a 2 cm high opening at the bottom of the front side, which was intended for honey bees to enter and leave the hive. This meant bees in the field of view of the camera could not fly, so this reduced the bees' movement speeds and removed the shadows of flying honey bees. This also overcame the uncontrolled environment conditions which affect the quality of the video. In addition, the paper used background subtraction based on Mixture of Gaussians (MOG) for the honey bee detection. Then, the colour variance and eccentricity features were analysed for distinguishing pollen bearing honey bees from non-pollen bees. In the experiment, the approach correctly classified 88.7% of honey bees on the test video

frames. However, the paper did not state whether the hardware setup affected the bees' life. Bees may build cells in the sensing box. The paper did not recommend any idea to avoid this. In addition, the pollen bearing honey bees could be distinguished from non-pollen bees, but the pollen sacs were not detected from bees' bodies. Furthermore, pollen bearing bees were only counted on each video frame. There was no bee tracking to help count the bees going through a sequence of video frames.

In the research of this thesis, the camera was setup on the front of the beehive in the uncontrolled environment, similar to the situation of the research (Campbell et al., 2008). This research faces the difficulties of fast bee speed, motion blur and flying bees' shadows. In addition, the pollen sacs were planned to be detected from the individual flying bees' bodies in this research. The next section will introduce the approaches for this research.

2.7 Approaches for this research

The situation of this research is that the video is recorded monitoring bees outside the beehive, so that the pollen sacs on the flight of honeybees can be detected and measured. This leads to some difficulties. Firstly, it is an uncontrolled outdoor environment. The light level changes with the time of the day and weather. Shadows are cast by the camera enclosure, moving bees and moving vegetation. Secondly, the speed of flight is fast, so flying bees can move a significant distance between frames. Moreover, when bees are flying, there is motion blur on the image. Finally, the scene is often cluttered. Bees can group or occlude each other which challenges simple segmentation and tracking methods. Considering these factors, the approach below was utilized for this work.

Motion detection is the common method for object detection with multiple frames. The background of the video is simplified by using a single coloured background board in this research. In this situation, the moving objects are bees and their shadows. As a result, background subtraction is more robust than frame difference for this research. The background model is more reliable than the difference between two continuous frames. Optical flow is based on the geometrical distance calculation. However, if bees change speed, the calculation can match the wrong pixel from the previous frame. Background subtraction creates a background model from some consecutive frames. GMM builds more than one background model for the comparison of each frame. It works for any bee speed and detects foreground moving pixels. GMM is sensitive to any movement, but bees' shadows are also detected. This problem cannot be solved using only motion detection.

The colour feature is the most obvious information on a single frame image of the video. The video includes a particular single colour background. Whether the background colour is white or green, the orange and black bee colours are different from the background. The colour histogram and thresholding algorithm is the main method for single frame bee detection. Apart from the colour feature, other features may not be suitable for bee detection. Although a bee's body includes a striped orange and black pattern, the texture information may not be clear on the video because of motion blur. Contour feature detection separates regions according to edges, but region location and reconstruction is a complicated calculation. Moreover, if the pollen colour is similar to bee body colour, it is difficult to detect the shape of a pollen sac. Therefore, this research

utilizes the colour thresholding method rather than texture and contour to detect bees and pollen sacs. In addition, combined with motion detection, it can remove the bees' shadows.

The aim of object tracking is to identify the same objects in consecutive frames of the video. A common method is to predict the movement of the object and then combine the prediction with the detection to get a better estimate of the position of the object. The particle filter samples the pixels of the object and then predicts the motion of each sample pixel. It tracks the object based on the pixels, so it can be a large computational problem. In addition, bees normally fly quickly, so the re-sampling region needs be chosen big enough to include the object region on the next frame. Bees can fly anywhere on the video, so tracking can easily fail with this method. Conversely, blob analysis produces the bees' positions after the bees' detections. Positions from previous frames can be used to predict the bees' next positions, so the Kalman filter can be employed to perform the tracking. The Kalman filter predicts the position based on the position of bees' blobs (coordinates of the blob centre) rather than the pixels of the bees. This reduces the computational requirements. The predicted and detected positions are assigned using the Hungarian method (Nandashri and Smitha, 2015b) which minimizes the sum of the distances between these positions. However, because the video is two dimensional, occlusion can be occur when bees appear to fly across each other (merge) in the video frame. In this case, the detected blob is the large merged blob rather than individual bee blobs. The position of the detection is the centre coordinates of the merged blob, which has a large error relative to the individual bees' positions. This affects the next prediction so the tracking can fail. The solution is to detect individual bees in the merged blob using the shape information from the detection of the previous frame when the bees have not merged yet. The shape information is recorded and the individual bee location is found using the Hough transform. In summary, bee tracking is performed using the Kalman filter and Hungarian algorithm with the Hough transform for the merging problem.

Pollen detection and measurement is the final aim of this research. It is impossible to detect pollen directly, because pollen sacs are too small. Therefore, the method of small object detection cannot be employed in this research. However, in the part based recognition model, each part is generally represented by a small template, and then the spatial relationships between parts are represented by statistical models between pairs of parts (Crandall et al., 2005). This method can be applied for the pollen. The approach is to detect individual bees on the whole video frame, and then detect pollen sacs on the smaller individual bee images. Initially, bee blobs are analysed to remove main body of bees. Then, pollen sacs can be detected separately from the other parts of bee (legs, wings, etc.) by using colour thresholding. After that, the detection is not perfect, because the colour detection may leave noisy blobs which are not pollen. The blob of the pollen may not be similar to the actual pollen sac's shape and area. Therefore, after the colour detection, the output includes not only pollen blobs, but also non-pollen blobs. These blobs are found four regular features which can be used for distinguishing pollen and non-pollen blobs. The features are analysed by receiver operating characteristic (ROC) and the statistical methods.

Another suitable pollen detection and measurement approach is machine learning. This method can mine the data of pollen sacs on images and find pollen sacs. The latest machine learning method is deep learning which uses deep neural networks to detect objects. The most popular

deep learning neural network for computer vision is CNN. Different CNNs approach for object detection have been built in recent years. Very popular approaches include SSD, YOLO, Faster RCNN and Mask RCNN. Faster RCNN includes an object proposal network before the detection. This is a benefit for small object detection (Liu et al., 2016). SSD and YOLO include only one network, which can increase the detection speed, but it is harder for small object detection (Liu et al., 2016, Redmon et al., 2016). The latest methods of YOLO v3 (Redmon and Farhadi, 2018) and Mask RCNN (He et al., 2017) have better performance on smaller object detection, but they are not convenient to use with the hardware limitations in this research. The Faster RCNN is chosen for pollen detection and measurement in this research, because the pollen sac is very small compared to the size of the bee body. In addition, it has been built in the MATLAB (2018a) neural network tool box, which is easy to apply for the pollen detection and measurement. Thus, the pollen detection model can be combined with the bee tracking model to measure pollen carrying bees.

Chapter 3

Chapter 3 Background of research

This research is implemented using the following areas of computer/machine vision. Image segmentation is used to detect the bees by motion detection and colour thresholding. This research also includes morphological dilation and erosion. Bee tracking is implemented using the Kalman filter and the Hungarian method. Blob analysis is used to detect the small pollen sacs on the bees' legs. This section will explore these methods to give some background to the research.

3.1 Motion detection using the Gaussian mixture model

Tracking the motion of bees flying with their pollen sacs to the entrance of the hive is a feature of this research. The video recording frame pixels can be separated into two groups: one group of pixels represents the foreground movement and the other group is the stationary background. This is based on the Gaussian mixture model (GMM) (Stauffer and Grimson, 1999).

The GMM estimates a probability density distribution (pdf) from a sequence of intensities [$I_{0,x}$, $I_{1,x}$, . . ., $I_{t,x}$] for a pixel at a position **x**. A mixture model consisting of K Gaussian distributions at time instant ***t*** can be denoted by

$$P(I_{t,x}) = \sum_{k=1}^{K} \omega_{t-1,x,k} \aleph(I_{t,x}; \mu_{t-1,x,k}, \sigma^2_{t-1,x,k}) \qquad 3\text{-}1$$

where $\aleph$ symbolizes a Gaussian probability density function (pdf):

$$\aleph(I; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{(I-\mu)^2}{2\sigma^2}) \qquad 3\text{-}2$$

In equation (3-1), the $\mu_{t-1,x,k}$ and $\sigma^2_{t-1,x,k}$ are the Gaussian mean and variance parameters of the k-th single distribution. $\omega_{t-1,x,k}$ is the mixture weight for maintaining this mixture model. K is normally 3 to 5 for the object detection on the video.

The GMM uses Gaussian pdfs to characterize the features of each pixel in an image. The mixture model is updated whenever a new image frame is obtained. Each pixel in the current image is tested as a match with the Gaussian mixture model. A match is defined as a pixel value within 2.5 standard deviations of a distribution. The matched pixel is assumed to belong to the background, otherwise it is a foreground (or moving object) pixel (Stauffer and Grimson, 1999). After that, the Gaussian mixture model is updated.

In the new image frame, if none of the K distributions match the new pixel value, the least probable single distribution is replaced by a new single distribution with the new pixel value as its mean

value, and an initially high variance, and low prior weight. The prior weights (of the k-th single distribution at time t in pixel x) are adjusted as follows:

$$\omega_{t,x,k} = (1-\gamma)\omega_{t-1,x,k} + \gamma M_{t,x,k} \quad \textit{3-3}$$

where γ is a learning rate and the $M_{t,x,k}$ is 1 for the single distribution which matched, and 0 for the remaining distributions. After this approximation, the weights are renormalized. The $1/\gamma$ defines the time constant which determines the speed at which the distribution's parameters change. The $\omega_{t,x,k}$ is a causal low-pass filtered average of the (thresholded) posterior probability that pixel values have matched model k given observations from time 1 through t. This is equivalent to the expectation of this value with an exponential window on the past values (Stauffer and Grimson, 1999).

The parameters μ and σ for unmatched distributions remain the same. The parameter of the distribution which matches the new observation are updated as below:

$$\mu_{t,x,k} = (1-\alpha)\mu_{t-1,x,k} + \alpha I_{t,x} \quad \textit{3-4}$$

$$\sigma^2_{t,x,k} = (1-\alpha)\sigma^2_{t-1,x,k} + \alpha(I_{t,x,k} - \mu_{t-1,x,k})^T(I_{t,x,k} - \mu_{t-1,x,k}) \quad \textit{3-5}$$

where α is the second learning rate:

$$\alpha = \gamma\aleph(I_{t,x}; \mu_{t-1,x,k}, \sigma^2_{t-1,x,k}) \quad \textit{3-6}$$

The α controls how fast the estimate μ converges to new observations. In updating the Gaussian parameters μ and σ^2, their values should reflect the up-to-date statistics of a scene as accurately as possible. It is preferable to set the learning rates to large values to quickly derive Gaussian distributions that fit new observations. A higher learning rate for μ and σ^2 improves model convergence and accuracy and brings few side effects in model stability (Lin et al., 2011).

One of the advantages of this method is that when an object is allowed to become part of the background, it does not destroy the existing model. The original background colour remains in the mixture until it becomes the k-th most probable and a new colour is observed. Conversely, if an object is stationary just long enough to become part of the background and then it moves, the distribution describing the previous background still exists with the same μ and σ^2, but a lower ω. It will be quickly re-incorporated into the background.

As the parameters of each pixel update, it is necessary to determine which of the Gaussians of the mixture are most likely produced by the background processes. Heuristically, it should be the Gaussian distributions which have the most supporting evidence and the least variance. This is the background model estimation.

When a new object occludes the background, it will not match one of the existing distributions, but it will either create a new distribution or increase the variance of an existing distribution. In addition, the variance for this distribution is expected to remain larger than a background pixel until the moving object stops. Considering this, it is necessary to decide what portion of the mixture model best represents background processes.

The Gaussians are ordered by the value of ω/σ. This value increases both as a distribution gains more evidence and as the variance decreases. After re-estimating the parameters of the mixture, it is sorted from the matched (less probable) distribution towards to the most probable background distribution. This orders the list of distributions, where the most likely background distributions remain on top and the less probable background distributions move to the bottom and are eventually replaced by new distributions. The first B distributions are chosen as the background model so that:

$$B = argmin_b(\sum_{k=1}^{b} \omega_{t,x,k} > T). \quad 3\text{-}7$$

T is a measure of the minimum portion of the data that should be accounted for by the background. If a small value of T is chosen, the background model is usually unimodal. In this case, only the most probable distribution is used. If T is higher, a multi-modal distribution caused by a repetitive background motion, such as leaves, a flag in the wind or river surface, can result in more than one colour being included in the background. This allows the background to accept two or more different colours.

The first B background Gaussian distributions are used to model the background. The pixels are determined as foreground points if there is no match with a background Gaussian distribution, otherwise, they are determined as background points. Actually, the foreground pixels belong to the moving object. If the background object also moves, it is hard to separate it from the foreground.

3.2 Colour segmentation

Colour segmentation means using colour to segment images. This is the natural method to start object detection. It is widely used in different ways to detect objects. In the frame flow, each frame is checked for the desired object with this method being repeated for each frame image on the video. If an object has only one colour, it can be detected. However, if the object has multiple colours, it requires a more complicated calculation.

3.2.1 Histogram thresholding with colour

In colour base segmentation, a simple method is to perform histogram thresholding on the colour image. Histogram thresholding is normally used for monochrome image segmentation (Littmann and Ritter, 1997). For example, an image $f(x, y)$ may be composed of two light objects on a dark background.

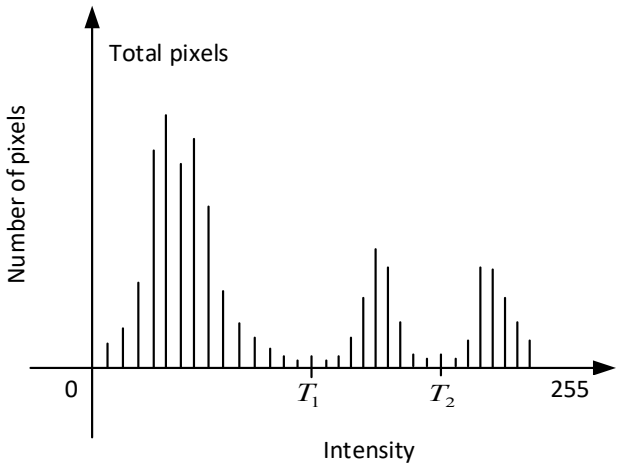


Figure 3.1 The example of histogram of an assumed image

Thus, the pixels would have intensity values grouped into three dominant modes, as indicated in Figure 3.1. The obvious way to extract the second light object from the background and the other lighter object is to select the thresholds of T_1 and T_2 for separating these modes. Then, any point (x, y) in the image at $T_1 < f(x, y) < T_2$ is an object point, otherwise the point does not belong to this object. In other words, the segmented image $\mathrm{g}(\mathrm{x}, \mathrm{y})$ is:

$$g(x,y) = \begin{cases} 1, & T_1 < f(x,y) < T_2 \\ 0, & f(x,y) < T_1, or\ f(x,y) > T_2 \end{cases} \quad \textit{3-8}$$

$g(x, y)$ is a binary image which displays the blob of the second light object. This is a simple example of multiple thresholding. It can be seen that if the object includes different intensity values, it needs more than two thresholds, resulting in the calculation being complicated. It can be seen that single intensity object detection is faster than texture and pattern detection.

In the case of a colour image, because it has multiple colour dimensions (Cheng et al., 2001), the thresholding is applied in the three dimensions of the colour channels. For example, the normal colour video has three channels of red, green and blue, which is the RGB colour space. The colour object is detected with histogram thresholding on the three image channels. There are some ways of simplifying the calculation, which are discussed in the next section

3.2.2 Colour spaces

For an RGB image, thresholding could be used for each component of the colour space (Siang Tan and Mat Isa, 2011), but this is computationally expensive. One way to solve this problem is to develop efficient methods for storing and processing the information of the 3-D colour space image (Cheng et al., 2001). Another way is to project the 3-D colour space onto a lower dimensional space (Cheng et al., 2001).

One more way for object detection is to consider other colour spaces rather than RGB. Many different types of colour space have been developed for different purposes. Traditionally there have been various colour spaces for image segmentation, such as LAB, XYZ, HSV, YCbCr, YIQ and DHT (Gritzman et al., 2014, Kumar, 2014). In the LAB colour space, L is the lightness, and A and B are measures of the colour opponents green-red and blue-yellow. The XYZ space is defined as luminance (Y) and chromaticity information (X and Z). Z is quasi-equal to blue

stimulation; and X is a mix (a linear combination) of human cone response curves chosen to be nonnegative. The YCbCr is a family of colour spaces used as a part of the colour image pipeline in video and digital photography systems. 'Y' is the luma component, and 'Cb' and 'Cr' are the blue-difference and red-difference chroma components. YIQ is used by the NTSC colour TV system. The 'Y' component represents the luma information, and is the only component used by the black-and-white television receiver. 'I' and 'Q' represent the other chrominance information differently from 'X' and 'Y'.

The HSV colour space is used for this research. It has the Hue-Saturation-Value (HSV) colour channels, as shown in Figure 3.2. In the HSV image, the Hue component contains information about the colour. Hue defines the dominant 'colour', as degrees in a circle going from $0°$ around to $360°$. Red occurs at $0°$ (and also $360°$), green is at $120°$ and blue is at $240°$. The Saturation represents the magnitude of the dominant colour (R, G or B). A higher value of Saturation gives a deeper colour. The Value presents the brightness of the pixel. In this channel, a higher pixel value relates to white and lightness. The HSV colour space and conversion between RGB and HSV were described by (Cardani, 2001). The colour segmentation was used to detect traffic signs (Fleyeh, 2006).

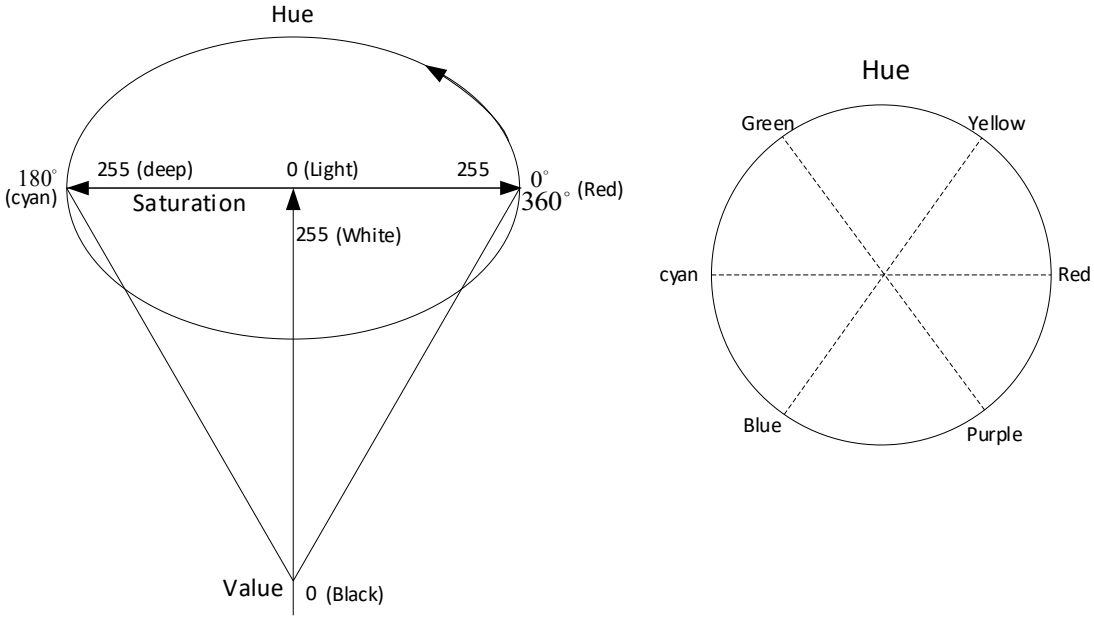


Figure 3.2 The HSV colour space

Assume that R, G and B are the values of the three channels in a pixel in an RGB image and that H, S and V are the values in HSV image. The RGB is converted to HSV as follows:

$$M = max(R, G, B) \quad \textit{3-9}$$
$$m = min(R, G, B) \quad \textit{3-10}$$
$$C = M - m \ (known\ as\ the\ chroma) \quad \textit{3-11}$$

and then:

$$H' = \begin{cases} undefined, \ if \ C = 0 \ (R = G = B) \\ \frac{G-B}{C} \quad , \ if \ M = R(R \ is \ maximum) and \ G \geq B \\ \frac{G-B}{C} + 6 \ , \ if \ M = R(R \ is \ maximum) and \ G < B \\ \frac{B-R}{C} + 2 \ , \ if \ M = G(G \ is \ maximum) \\ \frac{R-G}{C} + 4 \ , \ if \ M = B(B \ is \ maximum) \end{cases} \quad 3\text{-}12$$

Then $\mathrm{H} = 60 * H'$. When 'C' equals zero, 'H' is typically assigned to zero as well, and then:

$$V = M \quad 3\text{-}13$$
$$S = C/V \quad 3\text{-}14$$

where 'V' and 'S' are both in the range 0 to 1.

When histogram thresholding detection is applied on the RGB colour space, the colour space needs 3-D histogram thresholding to detect an object because this colour space has three colour channels. Conversely, the HSV colour space can reduce the number of dimensions to detect colours. Therefore, thresholding can be reduced from 3-D to 2-D (Hue and Saturation).

However, colour thresholding can only detect an object with a particular colour, and does not work for multiply coloured objects. All colour spaces have some disadvantages. For example, in an HSV image, an intensity that is too low or too high and saturation that is too low can affect the stability of the Hue (Vitabile et al., 2001). In addition, according to the equations for colour conversion, the white and black colours are not stable in the conversion. For example, although $\mathrm{R} = 255$, $\mathrm{G} = 254$ and $\mathrm{B} = 253$ appears as white to human vision, the conversion becomes $\mathrm{H}' = 0.5$. Therefore, if a pixel is on a too light or too dark object, it may be lost or detected as the wrong colour.

3.3 Hough transform detection

The Hough transform is a feature extraction technique used in image analysis, computer vision, and digital image processing. The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform.

The classical Hough transform was utilized to identify straight lines in an image, but it has been extended to identify the positions of arbitrary shapes such as circles and ellipses.

Sometimes, edge detection can be used as a pre-processing stage to obtain image pixels that are on the desired curve in the image space. In contrast, because of imperfections in either the image data or the edge detector, there may be missing points or pixels on the desired curves as well as spatial deviations between the ideal shapes and the noisy edge points as they are obtained from the edge detector. For these reasons, it is often necessary to group the extracted edge features to an appropriate set of lines, circles or ellipses. The aim of the Hough transform is to address this problem by making it possible to perform groupings of edge points into object candidates by performing an explicit voting procedure over a set of parameterized image objects.

3.3.1 The Hough transform theory

The simplest example of the Hough transform is detecting straight lines (Illingworth and Kittler, 1988). The line's equation is written as:

$$y = ax + b \tag{3-15}$$

where *x* and *y* are the horizontal and vertical coordinates of the line, *a* is the slope of the line, and *b* is the y-intercept. According to equation (3-15), the line is a set of parameters (*a*, *b*). The equation can be rewritten as:

$$b = y - xa \tag{3-16}$$

If it is considered that *x* and *y* are parameters, and *a* and *b* are variables, this is a new line in the (a, b) space which is parameterized by *x* and *y*. Therefore, a point in (*x*, *y*) space is a line in (*a*, *b*) space. Another point (*x*, *y*) will be another line in (*a*, *b*) space.

Figure 3.3 shows the space transform. In (*x*, *y*) space there are two points (0, 1) and (1, 2) on the same line (y = x+1). In the other (*a*, *b*) space, these two points are transformed into two lines (b = 0a + 1) and (b = a + 2) which are crossing at point (1, 1), which is the line (y = x + 1) in (*x*, *y*) space. With this method, in order to detect a line in (*x*, *y*) space, all the edge points are transformed to (*a*, *b*) space. Then it is obvious to see some lines concentrate to a single intersection point in (*a*, *b*) space. This point in (*a*, *b*) space represents a line in (*x*, *y*) space on which many of the edge points are located. However, a vertical line in (*x*, *y*) space has unlimited slope and no y-intercept, so its parameters *a* and *b* are infinite. Therefore, the vertical line cannot be detected by this means.

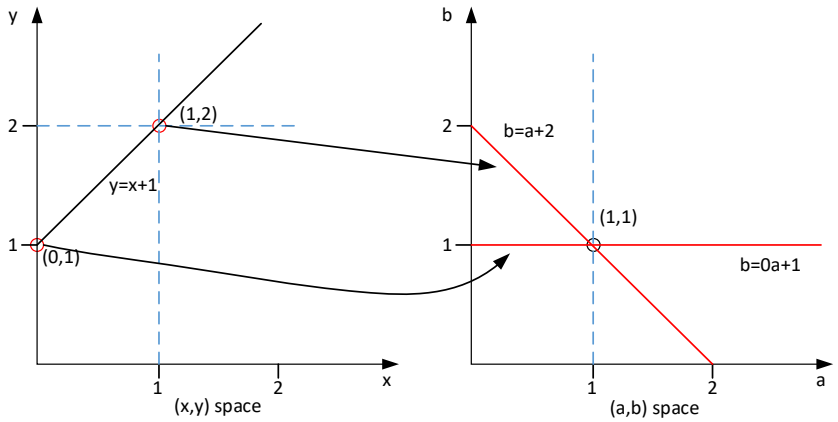


Figure 3.3 The (x, y) space and (a, b) space transformation

There is another method to detect a straight line from (Duda and Hart, 1972). In the (*x*, *y*) space, a straight line is defined as the equation:

$$r = x\cos\theta + y\sin\theta \tag{3-17}$$

where *r* is the distance from the origin to the closest point on the straight line. θ is the angle between the *x*-axis and the line linking the origin with that closest point. Each line in the (*x*, *y*) space can be transformed to a point in the (*r*, θ) space (or (*r*, θ) plane). The (*r*, θ) plane is also referred to as Hough space for the set of straight lines in two dimensions. For a single point in the (*x*, *y*) plane, all straight lines going through that point are on a sinusoidal curve in the (*r*, θ) plane,

which is unique to that point. Two or more points on a straight line in the (*x*, *y*) plane produce sinusoids which intersect at the point (*r*, θ) for the line. Therefore, the problem of detecting collinear points is converted to the problem of finding concurrent curves.

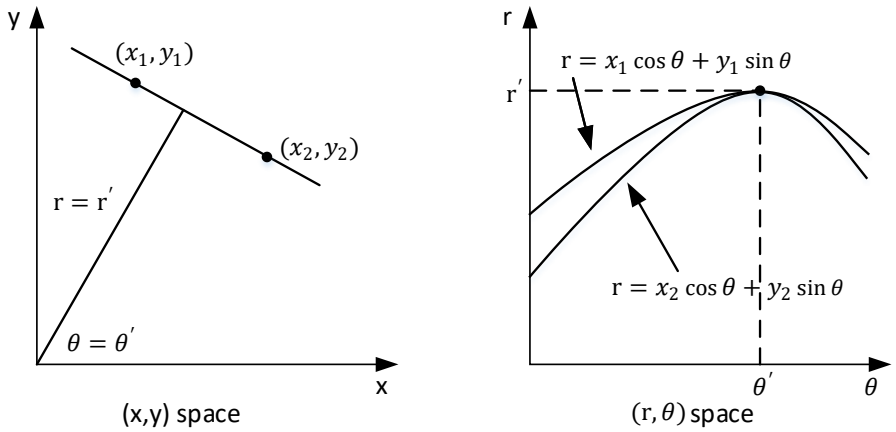


Figure 3.4 The (x, y) space and the (r, θ) *space transformation*

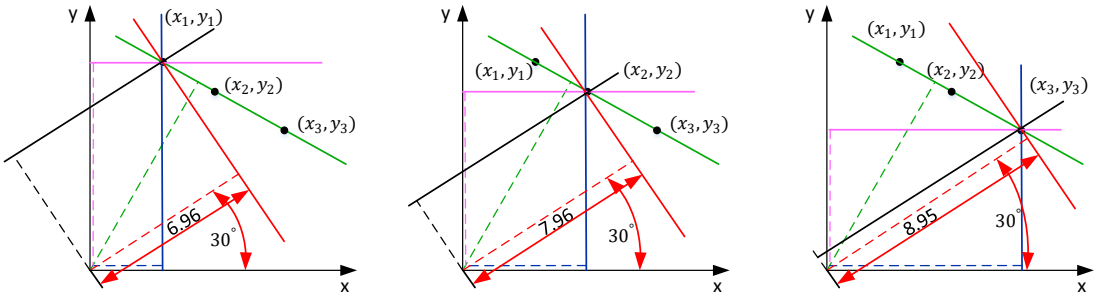
Figure 3.4 shows an example of the relationship between (x, y) space and (*r*, θ) space. The two curves in the (*r*, θ) space represent the families of lines that pass through the two particular points (x_1, y_1) and (x_2, y_2) in the (*x*, *y*) space. The intersection point (r', θ') of the two curves corresponds to the straight line that goes through the two points (x_1, y_1) and (x_2, y_2) in the (*x*, *y*) space. Using this method, a horizontal line has $\theta = 0$ and *r* equal to the intercept with the y-axis, and a vertical line has $\theta = 90$ and *r* equal to the intercept with the x-axis.

The Hough transform algorithm uses a 2-dimensional array, which is an accumulator (or voting map), to detect a line as specified in equation (3-17). The dimension of the accumulator depends on the number of unknown parameters, i.e. two dimensional for the parameters *r* and θ in the pair (*r*, θ). For each pixel at (*x*, *y*) and its neighbours, the Hough transform determines whether or not there is enough evidence of a straight line at the pixel. If so, it calculates the parameters (*r*, θ) of the line, and increments the value of the appropriate accumulator bin. The bins with the highest values, especially those at local maxima in the accumulator space, are the most likely lines to be extracted. The simplest way of finding these highest value peaks is by applying some form of threshold, but other techniques may yield better results in different circumstances – determining which lines are found as well as how many. After that, it is necessary to find which lines match which parts of the image, because the detected lines do not contain length information. Moreover, the edge detection before the Hough transform is imperfect, so there can be errors in the accumulator space, which means it may find inappropriate peaks, and thus inappropriate lines.

Finally, the result of the Hough transform is a two dimensional matrix similar to the accumulator – the (*r*, θ) axis. Each element of the matrix is the number of points (or pixels) which are located on the line represented by quantized parameters (*r*, θ). Therefore, the element with the highest value represents the most likely straight line in the input image.

3.3.2 An simple example of straight line detection

Assume there are three points corresponding to a line on an image as shown in Figure 3.5:



Angle	Distance
0	4.00
30	6.95
60	8.13
90	7.00
120	4.05

Angle	Distance
0	5.75
30	7.96
60	8.07
90	6.00
120	2.35

Angle	Distance
0	7.45
30	8.95
60	8.05
90	5.00
120	0.60

Figure 3.5 The process of the Hough transform.
Note: the green line is the one that crosses three points.

In Figure 3.5, each point has five lines going through it at different angles. These are displayed as solid lines with different colours. For each line, the shortest distance to the origin is shown as a dashed line. The distance to the origin and the angle of each dashed line are listed in the tables. These parameters can be used to create the Hough transform (r, θ) space:

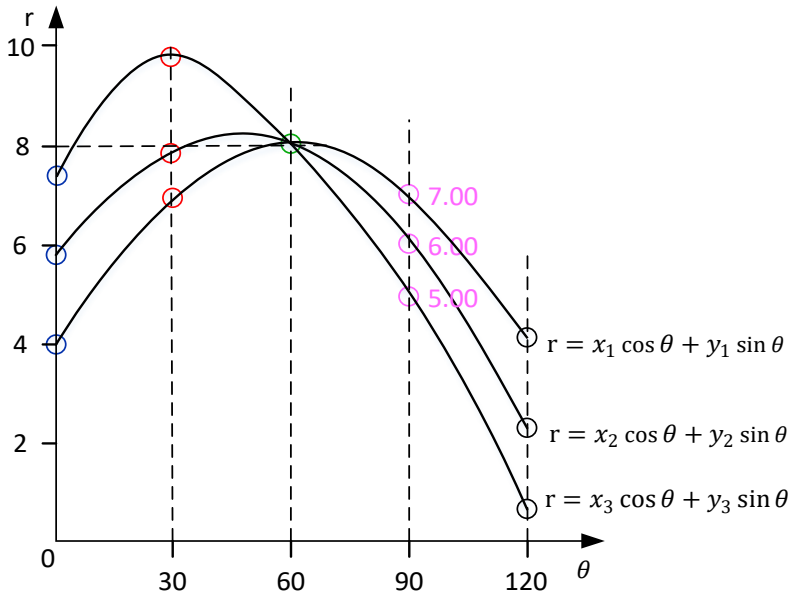


Figure 3.6 The transform result in Hough (r, θ) space.

The Hough transform (r, θ) space from the three points above is indicated by Figure 3.6. The accumulator matrix is built in the same (r, θ) space as Figure 3.6. Each point (or pixel) is the bin for counting matching curve points. From Figure 3.6, except for the green point, the point bins shown on the curves have a value of 1. The green point bin has a value of 3, because it matches three curves. All other point bins have the value 0. In other words, the point where the curves intersect most often gives the distance and angle of the most likely straight line, which intersects the points being tested. The diagram shows that the curves intersect at the green point, and this

corresponds to the solid green line in the (*x*, *y*) space above, which passes through all three points.

3.3.3 The generalized Hough transform

The transform described above applies only to finding straight lines. A similar transform can be generalized for finding any other shape that can be represented by parameters. For example, a circle can be transformed into a set of three parameters, the centre coordinates and radius, so the Hough space becomes three dimensional. Ellipses and other curves can also be detected in this way.

However, the computational complexity increases rapidly as the space dimensionality increases. In this situation, it is necessary to find some way to reduce the number of dimensions to simplify the computing. Take circle detection as an example. The equation of a circle can be written as:

$$(x - x_0)^2 + (y - y_0)^2 - R^2 = 0 \qquad 3\text{-}18$$

where (x_0, y_0) is the centre of the circle and *R* is its radius. In this case, the Hough transform is a transformation from the (*x*, *y*) space to the (x_0, y_0, R) space (this is a 3-dimensional space). The (x_0, y_0, R) space is also the accumulator space which includes bins for each pixel. If the radius of the circle is known before detection, the accumulator is reduced to 2-dimensions. As a simple case, for example, in Figure 3.7 there are five discrete points. If it is assumed that a circle with radius R = 1 intersects four of the points, then because the radius is known, the dimension of the accumulator space is reduced to two.

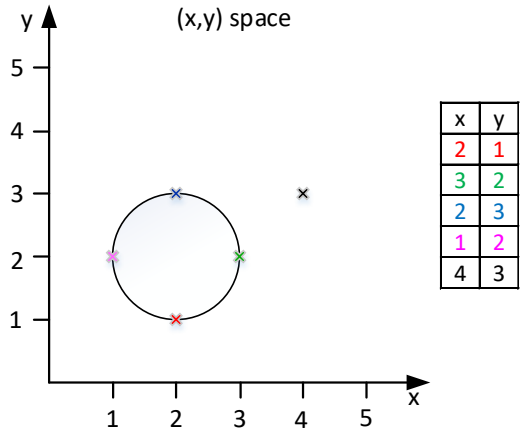


Figure 3.7 The 5 points at (x, y) space. 4 of them are in a same circle

For any fixed point (*x*, *y*), all possible circles with radius R = 1 passing through the point will contribute to the accumulator matrix. Figure 3.8 displays an example of a green point with four circles (with R = 1) passing through it.

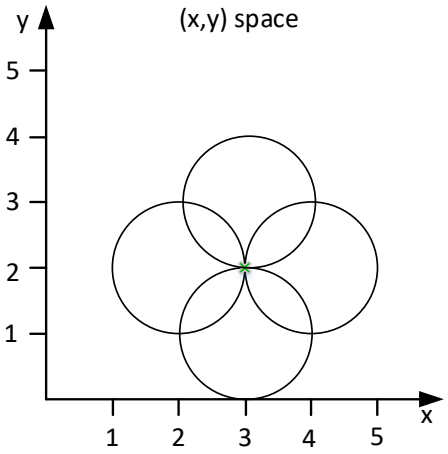


Figure 3.8 An example of circles intersecting the green point

At some distinct point $(x_0, y_0, 1)$ in the accumulator matrix, a point has the highest number of circles (in the parameter $(x_0, y_0, 1)$ space) passing through it. This is the probable candidate location of the circle.

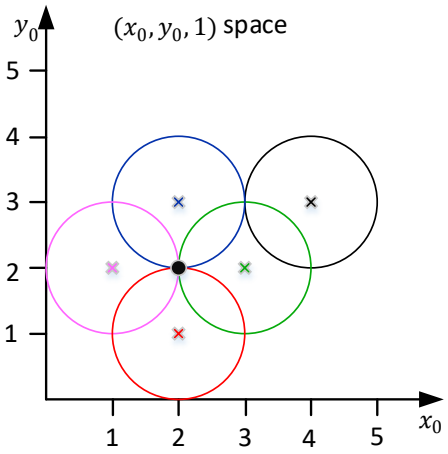


Figure 3.9 The Hough $(x_0, y_0, 1)$ space result

Figure 3.9 shows the Hough $(x_0, y_0, 1)$ space. On these axes, the five points are treated as circle centres. These circles have the same radius as the circle in (*x*, *y*) space. The black point (2, 2) matches 4 circles. This point is the centre of the circle in the (*x*, *y*) space of image.

The circle detection example above is a special situation with a known radius. If the radius of the circle is unknown, it is necessary to use a 3-dimensional accumulator space to search for circles with an arbitrary radius. This is more computationally expensive. This method can also detect circles which are partially outside the accumulator space, as long as enough of the circle's area is still present within the space.

In summary, the Hough transform algorithm uses the parameters from a shape's curves or equations. For example, straight lines, circles and ellipses have equations. The variables in the equations are the parameters which are utilized to create the Hough space. The number of parameters determines the dimension of the Hough space. The Hough space can be used as the accumulator matrix. Each element of this accumulator is a bin that counts the number of curves

drawn in the Hough space that intersect the corresponding pixel. The bin which has highest value is the candidate value of the desired shape equation. Therefore, if the shape is only partly in the image, it still can be detected, even if appears only as a broken line, half circle or incomplete ellipse.

However, the dimension of the Hough space depends on the number of parameters. Therefore, if the shape has a complicated equation or curve, it increases the computational requirement. The computational problem can be reduced by decreasing the dimension of the Hough space, by selecting values for some of the parameters beforehand, as in the circle detection example above. Moreover, another way to reduce the computing is to decrease the accumulation sample space. For instance, in Figure 3.5, each point can be given three lines instead of five lines, and it is still possible to detect the line that intersects the three points.

In addition, if a shape is non-analytical, there is no certain equation for it, so it is important to find an approach to detect the shape. Once the approach is found, the next step is the accumulator matrix creation. This research applies the Hough transform to detect bees when they fly across each other. The bees' shapes are somewhat similar in successive frames, so the shapes in the previous frame are used for the bee detection in the next frame.

3.4 Morphological dilation and erosion

Mathematical morphology in image processing is based on set theory (Gonzalez and Woods, 2011). Sets in mathematical morphology represent objects in an image. For example, the set of all white pixels in a binary image is a complete morphological description of the image. In binary images, the set members come from a 2-D integer space (Z^2), where each element of the set is a 2-D vector whose coordinates are the (x, y) coordinates of a white pixel in the image. The most commonly used morphological algorithms are dilation and erosion. These are used to smooth objects in an image. This research utilises these two operations for bee detection. This section introduces these algorithms. Before explaining the main operations, this section describes some basic concepts of set reflection and translation in morphology.

The concepts of set reflection and translation are used extensively in morphology. If **B** is the set of pixels representing an object in an image, then:

$$\hat{B} = \{\mu | \mu = -b, \ \ for\ b \in B\} \qquad \textit{3-19}$$

where $\hat{B}$ is defined as the reflection of set **B**, which is the set of points in **B** whose (x, y) coordinates have been replaced by $(-x, -y)$. Figure 3.10 (a) and (b) show a simple set and its reflection. The shaded triangle is constructed of pixels which are members of the set under consideration. In a binary image, the sets of interest are pixels corresponding to objects. These are shown in white, and all other pixels in black. The terms foreground and background are often used to denote the sets of pixels in an image defined to be objects and non-objects respectively.

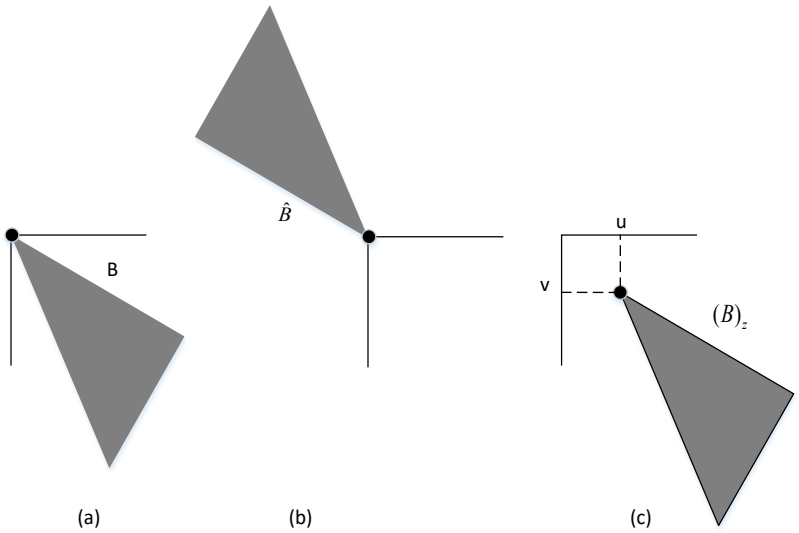


Figure 3.10 The example of reflection and translation.
Note: (a) A set B. (b) Its reflection. (c) Its translation by z(u,v).

The translation of a set B by point $z(u,v)$, denoted $(B)_z$, is defined as:

$$(B)_z = \{c | c = b + z, \ for\ b \in B\} \qquad 3\text{-}20$$

If **B** is the set of pixels representing an object in an image, then $(B)_z$ is the set of points in **B** whose (x,y) coordinates have been replaced by $(x+u, y+v)$. Figure 3.10(c) illustrates this concept from the set **B** from Figure 3.10(a).

3.4.1 Morphological erosion

With *A* and *B* as sets in Z^2, the erosion of *A* by *B*, denoted $A \ominus B$, is defined as:

$$A \ominus B = \{z | (B)_z \subseteq A\} \qquad 3\text{-}21$$

In words, this equation indicates that the erosion of *A* by *B* is the set of all points z such that *B*, translated by z, is contained in *A*. In the following discussion, set *B* is assumed to be a structuring element. Because the statement that *B* has to be contained in *A* is equivalent to *B* not sharing any common elements with the background, the erosion can be expressed in the following equivalent form:

$$A \ominus B = \{z | (B)_z \cap A^c = \emptyset\} \qquad 3\text{-}22$$

Where A^c is the complement of *A* and ∅ is the empty set.

Figure 3.11 shows examples of erosion. The elements of *A* and *B* are shown shaded and the background is white. The solid boundary in Figure 3.11(c) is the limit beyond which further displacements of the origin of *B* would lead to the structuring element to cease being completely contained in *A*. Thus, the points within and including this boundary, constitute the erosion of *A* by *B*. The erosion is shaded in Figure 3.11(c). In addition, the erosion is simply the set of values of z that satisfy equation (3-21) and (3-22). The boundary of set *A* is shown dashed in Figure 3.11 (c) and (e) only for reference, because it is not part of the erosion operation. Figure 3.11(d) shows an elongated structuring element, and Figure 3.11(e) shows the erosion of *A* by this element. The original set has been eroded to a rectangle.

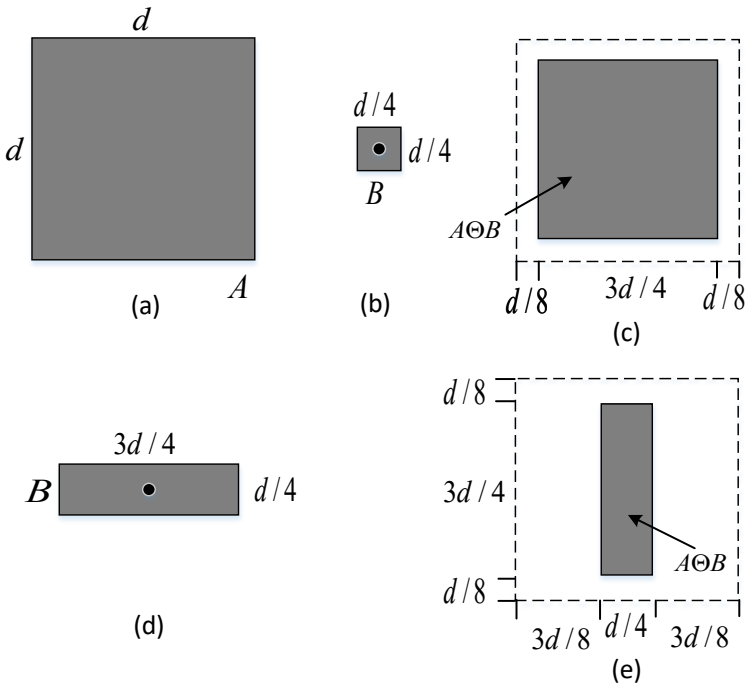


Figure 3.11 Examples of erosion.

Note: (a) The set A. (b) Square structuring element B. (c) Erosion of A by B, shown shaded. (d) Elongated structuring element. (e) Erosion of A using this element. The dotted border in (c) and (e) is the boundary of set A, shown for reference.

3.4.2 Morphological dilation

With *A* and *B* as sets in Z^2, the dilation of *A* by *B*, denoted $A \oplus B$, is defined as:

$$A \oplus B = \left\{ z \middle| (\hat{B})_z \cap A \neq \emptyset \right\} \qquad \textit{3-23}$$

This equation is based on reflecting *B* about its origin, and shifting this reflection by z (see Figure 3.10). The dilation of *A* by *B* is the set of all displacements, z, such that $\hat{B}$ and *A* overlap by at least one element.

It is assumed that *B* is the structuring element (SE) and *A* is the set (image objects) to be dilated. *B* can be various different shapes, such as a square, diamond, circle or disk. The dilation of set *A* by *B* is different for different SE shapes.

The dilation “grows” and “thickens” objects in a binary image. The specific manner and extent of this thickening is controlled by the shape of the structuring element used. Figure 3.12 shows an example of dilation. The set A is shown shaded in Figure 3.12 (a) and Figure 3.12 (b) shows a structuring element (in this case $\mathrm{B} = \hat{B}$ because the SE is symmetric about its origin). The dashed line in Figure 3.12 (c) shows the original set *A* for reference, and the solid line shows the limit beyond which any further displacements of the origin of $\hat{B}$ by z would cause the intersection of $\hat{B}$ and *A* to be empty. Therefore, all points on and inside this boundary constitute the dilation of *A* by *B*. Figure 3.12 (d) shows a structuring element designed to achieve more dilation vertically than horizontally, and Figure 3.12 (e) indicates the dilation achieved with this element.

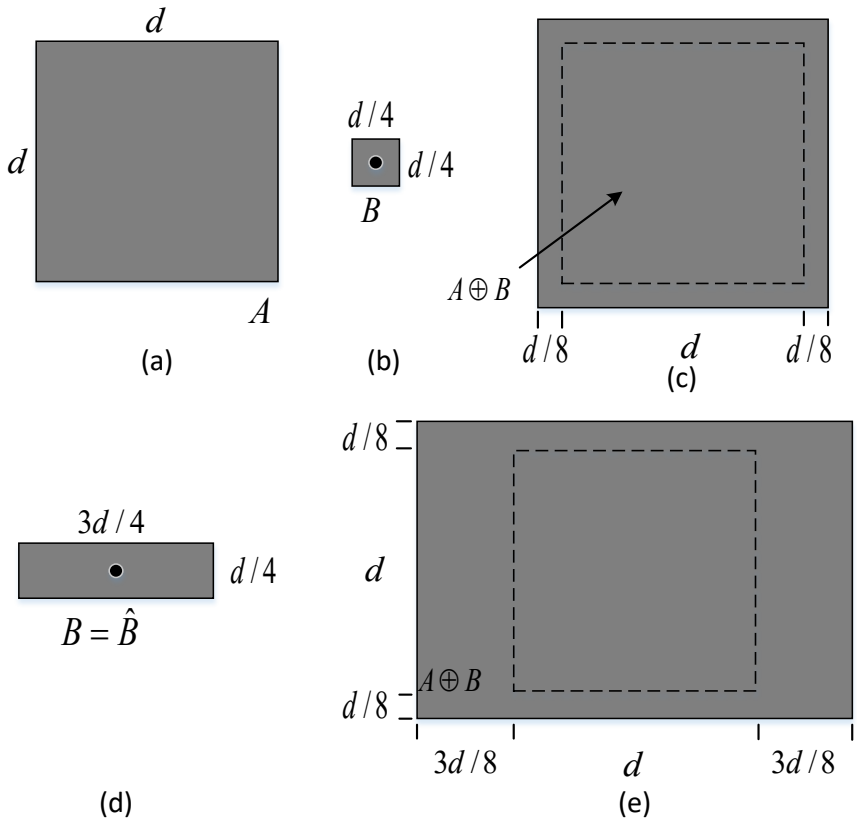


Figure 3.12 Examples of dilation.

Note: (a) The set A. (b) Square structuring element B. (c) Dilation of A by B, which is shown shaded. (d) Elongated structuring element. (e) Dilation of A by B using this element. The dotted border in (c) and (e) is the boundary of set A, shown for reference.

3.4.3 Image opening, closing and hole filling

3.4.3.1 Image opening and closing

Dilation expands the components of an image, while erosion shrinks them. Two other important morphological operations are opening and closing. Opening generally smooths the contour of an object, breaking narrow isthmuses, and eliminating thin protrusions. Closing on the other hand, while it tends to smooth sections of contours, is the opposite of opening in that it generally fuses narrow breaks and long thin gulfs, eliminating small holes, and filling gaps in the contour.

The opening of set *A* by structuring element *B*, denoted $A \circ B$, is defined as:

$$A \circ B = (A \ominus B) \oplus B \qquad \textit{3-24}$$

Thus, opening *A* by *B* is the erosion of *A* by *B*, followed by a dilation of the result by *B*.

The closing of set *A* by structuring element *B*, denoted $A \bullet B$, is defined as:

$$A \bullet B = (A \oplus B) \ominus B \qquad \textit{3-25}$$

This indicates that the closing of *A* by *B* is the dilation of *A* by *B*, followed by the erosion of the result by *B*.

3.4.3.2 Filling holes

A hole is defined as a background region surrounded by a connected border of foreground pixels. In an image, filling holes is an algorithm based on set dilation, complementation, and intersection. It is assumed *A* is a set whose elements are 8-direction-connected boundaries, each boundary enclosing a background region (a hole). Each hole pixel is a point, the aim is to fill all of the hole points with 1s.

An array (X_0) of 0s is formed, (the same size as the array containing *A*), except at the locations in X_0 corresponding to the points of holes, which are set to 1. Then, the procedure of filling all the holes with 1s is:

$$X_n = (X_{n-1} \oplus B) \cap A^c \quad n = 1, 2, 3, \ldots \qquad \textit{3-26}$$

where *B* is the symmetric structuring element in Figure 3.13 (c). The algorithm terminates at iteration step n if $X_n = X_{n-1}$. The final set X_n contains all the filled holes. The set union of X_n and *A* contains all the filled holes and their boundaries.

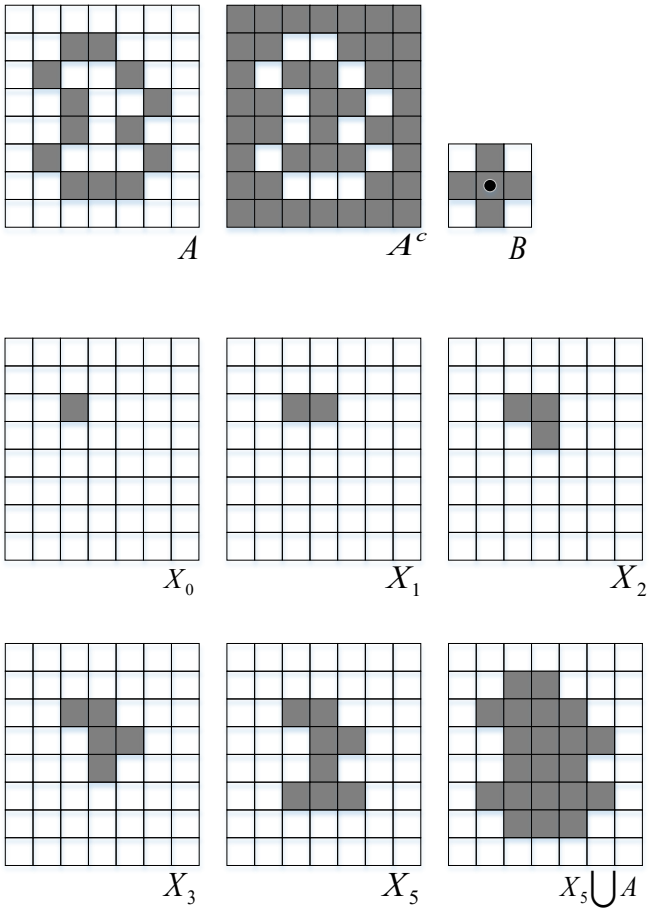


Figure 3.13 An example of hole filling.
Note: (a) Set A is shown shaded. (b) Complement of A. (c) Structuring element B. (d) Initial point inside the boundary. (e)-(h) Various steps of the algorithm. (i) Final result of union of (a) and (h).

The dilation in equation (3-26) fills the entire area if left unchecked. However, the intersection at each step with A^c limits the result to inside the region of interest. The Figure 3.13 (d)-(h) indicates further the mechanics of equation (3-26). Although this example only has one hole, the concept

clearly applies to any finite number of holes, assuming that a point is given inside each hole region.

Image opening, closing and filling holes are common operations used in image processing after image segmentation. Normally, the binary image blob that is the result of object detection is operated on using image opening to smooth the blob edge.

3.5 Kalman filter for object tracking

The Kalman filter has been used successfully in engineering for many years. It is described by a system state model and a measurement model (Weng et al., 2006):

- System state model:

$$s(t+1) = A(t)s(t) + \mu(t) \qquad 3\text{-}27$$

- Measurement model:

$$z(t) = H(t)s(t) + \omega(t) \qquad 3\text{-}28$$

$\mathrm{A}(t)$ and $\mathrm{H}(\mathrm{t})$ are the state transition matrix and measurement matrix, respectively. $\mu(t)$ and $\omega(\mathrm{t})$ are white Gaussian noise with zero mean. The state vector $\mathrm{s}(\mathrm{t})$ at the current time **t** is predicted from the previous estimate and the new measurement $\mathrm{z}(\mathrm{t})$. The procedure of the Kalman filter has two steps: prediction and correction. The prediction step is to calculate forward from the current state, to obtain an a priori estimate of the next state. The correction step combines the prediction and the new measurement to produce an optimal estimate of the state. The prediction-correction cycle is repeated for an optimal tracking result.

3.5.1 Discrete Kalman filter estimation

In digital computing, the Kalman filter has been the subject of extensive research and application, such as optimal control, assisted navigation and object tracking. This section introduces the theory of discrete Kalman filter. It is assumed that the state $s \in R^n$ is a discrete-time prediction that is governed by the linear stochastic difference equation (Welch and Bishop, 2006):

$$s_{k+1} = A_k s_k + \mu_k \qquad 3\text{-}29$$

and the measurement $\mathrm{z} \in R^n$ is:

$$z_k = H_k s_k + \omega_k \qquad 3\text{-}30$$

The random variables μ_k and ω_k represent the prediction and measurement noises respectively. They are assumed to be independent of each other, and with normal probability distributions:

$$p(\mu) \sim N(0, Q) \qquad 3\text{-}31$$

$$p(\omega) \sim N(0, R), \qquad 3\text{-}32$$

where Q is the co-variance matrix of the prediction noise (μ_k), and the R is the co-variance matrix of measurement noise (ω_k). The $n \times n$ matrix A_k in the difference equation (3-29) relates the state at time step *k* to the state at step *k*+1. The $m \times n$ matrix H_k in the measurement equation (3-30) relates the state of s_k to the measurement z_k.

$\hat{s}_k^- \in R^n$ is defined as the **a priori** state estimate at step *k* for the prediction. $\hat{s}_k \in R^n$ is defined as the **a posteriori** state estimate at step *k* for measurement z_k. The **a priori** and **a posteriori** estimate errors are defined as:

$$e_k^- \equiv s_k - \hat{s}_k^- \quad \textit{3-33}$$

$$e_k \equiv s_k - \hat{s}_k \quad \textit{3-34}$$

The **a priori** estimate error covariance is then

$$P_k^- = E[e_k^- e_k^{-T}] \quad \textit{3-35}$$

and the **a posteriori** estimate error covariance is

$$P_k = E[e_k e_k{}^T] \quad \textit{3-36}$$

In deriving the equations for the Kalman filter, an equation has to be found that computes an a posteriori state estimate $\hat{s}_k$ as a linear combination of an priori estimate $\hat{s}_k^-$ and a weighted difference between an actual measurement z_k and a measurement prediction ($H_k\hat{s}_k^-$) as shown in (3-37).

$$\hat{s}_k = \hat{s}_k^- + K_k(z_k - H_k\hat{s}_k^-) \quad \textit{3-37}$$

The difference $(z_k - H_k\hat{s}_k^-)$ in (3-37) is known as the measurement residual. The residual reflects the discrepancy between the predicted measurement $H_k\hat{s}_k^-$ and the actual measurement z_k. A residual of zero means that the two are in complete agreement.

The $m \times n$ matrix K_k in (3-37) is the Kalman gain that minimizes the a posteriori error covariance (3-36). This minimization is accomplished by first substituting (3-37) into the above definition for e_k, substituting that into (3-36), performing the indicated expectations, taking the derivative of the trace of the result with respect to K, setting that result equal to zero, and then solving for ***K***.

3.5.2 Calculation of the Kalman gain

The derivation of the Kalman gain is shown below.

From (3-36):

$$P_k = E[(s_k - \hat{s}_k)(s_k - \hat{s}_k)^T] \quad \textit{3-38}$$

Substituting from (3-37):

$$P_k = E\{[(s_k - \hat{s}_k^-) - K_k(z_k - H_k\hat{s}_k^-)][(s_k - \hat{s}_k^-) - K_k(z_k - H_k\hat{s}_k^-)]\} \quad \textit{3-39}$$

Substituting from (3-30):

$$P_k = E\{[(s_k - \hat{s}_k^-) - K_k(H_k s_k + \omega_k - H_k\hat{s}_k^-)][(s_k - \hat{s}_k^-) - K_k(H_k s_k + \omega_k - H_k\hat{s}_k^-)]\} \quad \textit{3-40}$$

Because $(s_k - \hat{s}_k^-)$ is uncorrelated with ω_k:

$$P_k = (I - K_k H_k)P_k^-(I - K_k H_k)^T + K_k R_k K_k^T \quad \textit{3-41}$$

The optimum K_k value will minimise the sum of the terms along the diagonal of P_k. From the mathematics:

$$\frac{d_{trace}(AB)}{dA} = B^T \quad \textit{3-42}$$

$$\text{and } \frac{d_{trace}(ACA^T)}{dA} = 2AC \quad \text{3-43}$$

Expanding P_k from above:

$$P_k = P_k^- - K_k H_k P_k^- - P_k^- H_k^T K_k^T + K_k(H_k P_k^- H_k^T + R_k)K_k^T \quad \text{3-44}$$

The trace of P_k needs to be minimised bacause that is the sum of the diagonal elements which should be minimised, so:

$$\frac{d_{trace}(P_k)}{dK_k} = 0 \quad \text{3-45}$$

where:

$$-2(H_k P_k^-)^T + 2K_k(H_k P_k^- H_k^T + R_k)^T = 0 \quad \text{3-46}$$

Solving for the optimum gain gives:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} = \frac{P_k^- H_k^T}{H_k P_k^- H_k^T + R_k} \quad \text{3-47}$$

which is the Kalman gain. According to the equation above, as the measurement error covariance R_{k} approaches zero, the gain $\mathrm{K_k}$ weights the residual more heavily:

$$\lim_{R \to 0} K_k = H_k^{-1} \quad \text{3-48}$$

On the other hand, as the priori estimate error covariance P_k^- approaches zero, the gain $\mathrm{K_k}$ weights the residual less heavily:

$$\lim_{P_k^- \to 0} K_k = 0 \quad \text{3-49}$$

Another way to understand the weighting of $\mathrm{K_k}$ is that as the measurement error covariance R_{k} approaches zero, the actual measurement $\mathrm{z_k}$ is “trusted” more, while the predicted measurement $H_k \hat{s}_k^-$ is trusted less. On the other hand, as the priori estimate error covariance P_k^- approaches zero, the actual measurement $\mathrm{z_k}$ is trusted less, while the predicted measurement $H_k \hat{s}_k^-$ is trusted more.

3.5.3 Correction covariance update

Substituting the Kalman gain expression (3-47) into the covariance update (3-41):

$$P_k = P_k^- - P_k H_k^T (H_k P_k^- H_k^T + R_k)^{-1} H_k P_k^- \quad \text{3-50}$$

or:

$$P_k = P_k^- - K_k (H_k P_k^- H_k^T + R_k)^{-1} K_k^T \quad \text{3-51}$$

or even:

$$P_k = (I - K_k H_k) P_k^- \quad \text{3-52}$$

3.5.4 Summary of Discrete Kalman filter algorithm

The Kalman filter estimates a process by using a loop: the filter estimates the prediction state and then obtains feedback of noisy measurements (Welch and Bishop, 2006). The equations for the Kalman filter fall into two groups: time update (or prediction) and measurement update (or correction). The prediction equations are responsible for projecting forward the current state and

error covariance estimates to obtain the a priori estimates for the next time step. The correction equations are responsible for the feedback for incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate. As a summary, the estimation algorithm resembles that of prediction-correction algorithm for solving numerical problems.

The prediction equations are as follows:

$$s_{k+1} = A_k s_k \quad \textit{3-53}$$

$$P_{k+1}^{-} = A_k P_k A_k^T + Q_k \quad \textit{3-54}$$

The prediction equations project the state and covariance estimates from time step ***k*** to step ***k+1***. $\mathrm{A_k}$ is from (3-29), while Q_k is from (3-31) which is the initial condition for the filter.

The correction equations are:

$$K_k = P_k^{-} H_k^T (H_k P_k^{-} H_k^T + R_k)^{-1} \quad \textit{3-55}$$

$$\hat{s}_k = \hat{s}_k^{-} + K(z_k - H_k \hat{s}_k^{-}) \quad \textit{3-56}$$

$$P_k = (I - K_k H_k) P_k^{-} \quad \textit{3-57}$$

The first equation in the correction computes the Kalman gain $\mathrm{K_k}$. Here, the equation given as (3-55) is the same as (3-47). The next step is to measure the process to obtain $\mathrm{z_k}$, and then to generate the a posteriori state estimate by incorporating the measurement as in (3-56) which is a restatement of (3-37) for completeness. The final equation is to obtain the a posteriori error covariance estimate via (3-57). After each time interval, the Kalman filter process is repeated with the previous a posteriori estimates used to project or predict the new a priori estimates.

3.5.5 The filter parameters and tuning

In the actual implementation of the filter, the measurement error covariance matrix R_k and the prediction noise Q_k are measured prior to the operation of the filter. In the case of the measurement error covariance R_k, it is necessary to take some off-line sample measurements in order to determine the variance of the measurement error.

Q_k is often chosen less deterministically. For example, this noise source is often used to represent the uncertainty in the predicted model (3-29). Sometimes a very poor model can be used simply by “injecting” enough uncertainty via the selection of Q_k. Certainly in this case, the measurement of the prediction is hoped to be reliable.

In this case, whether the chosen parameters optimal or not, often superior filter performance can be obtained by “tuning” the filter parameters Q_k and R_k. This tuning is usually performed off-line, frequently with the help of another distinct Kalman filter.

In particular, under the conditions in which Q_k and R_k are constant, both the estimation error covariance P_k and the Kalman gain $\mathrm{K_k}$ will stabilize quickly and then remain constant. In this case, these parameters can be pre-computed by either running the filter off-line, or solving (3-54) for the steady-state value of P_k by defining $\mathrm{P_k^-} \equiv P_k$ and solving for P_k.

It is frequently the case however that the measurement error does not remain constant. In addition, the prediction noise Q_k sometimes changes dynamically during filter operation in order

to adjust to different dynamics. For example, in the case of tracking the head of a person in a 3D virtual environment, the Q_k may be reduced if the person seems to be moving slowly, and increase if the dynamics start changing rapidly. In such a case, Q_k can be used to model not only the uncertainty in the model, but also the uncertainty of the person's intentions.

3.5.6 The Kalman filter for object tracking

When applied to movement tracking in a series of video images, the system state model is the moving model. The movement distance within a frame interval is the feature to be estimated and predicted. This distance can be calculated from the position in the previous frames. Therefore the Kalman filter can use the positions to track objects. The Kalman filter process is shown below.

The state of an object includes its position $(x_k,\ y_k)$ and velocity $(u_k,\ v_k)$ in the image, where k is the frame number on the video recording. The motion (prediction) model in each frame interval is:

$$x_k = x_{k-1} + u_{k-1} \quad \textit{3-58}$$
$$y_k = y_{k-1} + v_{k-1} \quad \textit{3-59}$$

The model assumes the velocity is constant in successive frames:

$$u_k = u_{k-1} \quad \textit{3-60}$$
$$v_k = v_{k-1} \quad \textit{3-61}$$

The state vector in frame k is:

$$s_k = [x_k, u_k, y_k, v_k]^T \quad \textit{3-62}$$

The state vector in frame $k-1$ is:

$$s_{k-1} = [x_{k-1}, u_{k-1}, y_{k-1}, v_{k-1}\ ]^T \quad \textit{3-63}$$

The state-space model of the Kalman filter is:

$$s_k = A_{k-1} s_{k-1} \quad \textit{3-64}$$

where the transition matrix is defined as:

$$A_{k-1} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textit{3-65}$$

The measurement vector in frame k is:

$$m_k = [x_k, y_k]^T \quad \textit{3-66}$$

The measurement model of the Kalman filter is:

$$m_k = H_k s_k \quad \textit{3-67}$$

where the measurement transition matrix is:

$$H_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \textit{3-68}$$

It is assumed the prediction noise Q_k and measurement error covariance R are constant in the object tracking, because normally the movement object for the Kalman filter has a slow speed compared to the video with a 30-60Hz frame rate. For example, Kalman filter was utilized to track

human movement (Weng et al., 2006). The person's movement was treated at constant velocity. However, if the velocity changed quickly, tracking was affected. Another example was that road vehicles could be tracked by using a Kalman filter (Shantaiya et al., 2015). However, they mentioned that the tracking was somewhat difficult if the object changed velocity.

Another way to describe the Kalman filter for object tracking comes from (Weng et al., 2006). The predicted position in frame k is set as:

$$d(k) = d(k-1) + (d(k-1) - d(k-2)), \qquad 3\text{-}69$$

where $d(k-1)$ and $d(k-2)$ are the actual positions in frame $(k-1)$ and $(k-2)$, respectively. The system state model of Kalman filter can be described as

$$s(k) = A(k-1)s(k-1) + \mu(k) = \begin{bmatrix} 2 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} d(k-1) \\ d(k-2) \end{bmatrix} + \begin{bmatrix} \mu(k) \\ 0 \end{bmatrix}, \qquad 3\text{-}70$$

where $s(k) = \begin{bmatrix} d(k) \\ d(k-1) \end{bmatrix}$, $s(k-1) = \begin{bmatrix} d(k-1) \\ d(k-2) \end{bmatrix}$ and $A(k) = \begin{bmatrix} 2 & -1 \\ 1 & 0 \end{bmatrix}$.

Although the transition matrix A_{k-1} is different from (3-65), the principle is the same. This research uses this technique to apply the Kalman filter.

3.6 Hungarian assignment method

The Hungarian method is a combinatorial optimization algorithm that solves the assignment problem in polynomial time (Kuhn, 2005). Consider a simple example for explanation. There are three workers: Tom, Stephen and Allan. One of them has to clean the bathroom, another sweep the floors, and the third wash the windows, but they each demand different pay for the various tasks. The problem is to find the lowest-cost way to assign the jobs. The problem can be represented in a matrix of the costs of the workers doing the jobs:

Table 3-1 The example of Hungarian method with three workers and three jobs

	Clean bathroom	Sweep floors	Wash windows
Tom	$2	$3	$3
Stephen	$3	$2	$3
Allan	$3	$3	$2

If the Hungarian method is applied to the above table, it gives the minimum cost as $6, achieved by having Tom clean the bathroom, Stephen sweep the floors, and Allan wash the windows.

This is a simple example of Hungarian method application. The 3×3 matrix has 6 assignment trials to get the minimum cost. However, if there are ten workers and ten jobs, there have to be many trials to calculate the lowest cost. In summary, there are $\mathbf{n}!$ ways of assigning **n** resources to **n** tasks.

3.6.1 The process of the Hungarian method

The process of the Hungarian method is described in (Dutta and Pal, 2015). This section will introduce the process. There is a non-negative $n \times n$ cost matrix, from which is to be found in an optimal assignment.

Assuming n = 4, the cost matrix can be:

$$\begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix}$$

where the rows are the workers and the columns are jobs respectively. The entries of the matrix are c_{ij}. The matrix is square, so each worker can perform only one task.

Step 1:

Subtract the smallest entry in each row from all the entries of its row. This will lead to at least one zero in each row. If there are two equal lowest entries in a row, the result is multiple zeros in that row. Then, each worker can be assigned only one job as shown by the zero. This is shown below:

$$\begin{bmatrix} 0 & c'_{12} & c'_{13} & c'_{14} \\ c'_{21} & c'_{22} & c'_{23} & 0 \\ c'_{31} & 0 & c'_{33} & c'_{34} \\ c'_{41} & c'_{42} & 0 & c'_{44} \end{bmatrix}$$

Step 2:

Subtract the smallest entry in each column from all the entries of its column. This is necessary because sometimes it may turn out that the matrix after step 1 cannot be used for assigning as seen in the matrix below:

$$\begin{bmatrix} 0 & c'_{12} & c'_{13} & c'_{14} \\ c'_{21} & c'_{22} & c'_{23} & 0 \\ 0 & c'_{32} & c'_{33} & c'_{34} \\ c'_{41} & c'_{42} & 0 & c'_{44} \end{bmatrix}$$

In the above case, no assignment can be made. The job 1 is done with a lowest cost by both workers 1 and 3. They cannot both be assigned the same job. In addition, no one does job 2 with the lowest cost. This step is used to solve this problem. Then it can be checked if an assignment is possible. In most situations, this will give the desired result, but if this is still not possible, then the next step should be taken.

Step 3:

Draw lines through appropriate rows and columns so that all the zero entries of the cost matrix are covered and the minimum number of such lines is used. this is described as one way below:

$$\begin{bmatrix} 0 & c'_{12} & c'_{13} & c'_{14} \\ c'_{21} & c'_{22} & c'_{23} & 0 \\ 0 & c'_{32} & c'_{33} & c'_{34} \\ c'_{41} & 0 & 0 & c'_{44} \end{bmatrix}$$

Take the matrix above as an example. Firstly, assign as many jobs as possible.

- Row 1: has one zero, assigned for worker 1 and job 1. The 0 in row 3 is crossed out because it is in the same column.
- Row 2: has one zero, for worker 2 and job 4.
- Row 3: the only zero has been crossed out, so nothing is assigned.

- Row 4: has two uncrossed zeros. Either one can be assigned (both jobs 2 and 3 are optimal for worker 4), and the other zero would be crossed out. By the way, the 0 in row 3 may be assigned for worker 3 and job 1, in which case the 0 in row 1 is crossed out.

$$\begin{bmatrix} \times & & & & \\ 0 & c'_{12} & c'_{13} & c'_{14} & \times \\ c'_{21} & c'_{22} & c'_{23} & 0 & \\ 0 & c'_{32} & c'_{33} & c'_{34} & \times \\ c'_{41} & 0 & 0 & c'_{44} & \end{bmatrix}$$

Second, mark the matrix and draw lines.

- Mark all rows having no assignments (row 3).
- Mark all (unmarked) columns having zeros in newly marked row (column 1).
- Mark all rows having assignments in newly marked columns (row 1).
- Repeat for all non-assigned rows.
- Draw lines through all marked columns and unmarked rows.

$$\begin{bmatrix} 0 & c'_{12} & c'_{13} & c'_{14} \\ c'_{21} & c'_{22} & c'_{23} & 0 \\ 0 & c'_{32} & c'_{33} & c'_{34} \\ c'_{41} & 0 & 0 & c'_{44} \end{bmatrix}$$

The details above provide just one way of drawing the minimum number of lines needed to eliminate all the 0s. Other methods also work as well.

Step 4:

Test for optimality: if the minimum number of lines is **n**, an optimal assignment of zeros is possible and the assignment can be finished. If the minimum number of lines is less than **n**, it is not enough to find an optimal assignment.

Step 5:

Determine the smallest entry not crossed out by any line. Subtract this entry from each row that is not crossed out, and then add it to each crossed out column. Then, return to the Step3.

3.6.2 The example of the method

Example 1:

Assuming there are three people and three jobs, the hourly paid is shown below:

Table 3-2 The example 1: three people and three jobs hourly paid

	Job 1 ($)	Job 2 ($)	Job 3 ($)
Person 1	25	40	35
Person 2	40	60	35
Person 3	20	40	25

$$\begin{bmatrix} 25 & 40 & 35 \\ 40 & 60 & 35 \\ 20 & 40 & 25 \end{bmatrix} \sim \begin{bmatrix} 0 & 15 & 10 \\ 5 & 25 & 0 \\ 0 & 20 & 5 \end{bmatrix}$$

Step 1: Subtract 25 from row 1, 35 from row 2, and 20 from row3.

Step 2: subtract 0 from column 1, 15 from column 2, and 0 from column 3.

$$\begin{bmatrix} 0 & 15 & 10 \\ 5 & 25 & 0 \\ 0 & 20 & 5 \end{bmatrix} \sim \begin{bmatrix} 0 & 0 & 10 \\ 5 & 10 & 0 \\ 0 & 5 & 5 \end{bmatrix}$$

Step 3: draw all the zeros of the matrix with the minimum number of horizontal or vertical lines.

$$\begin{bmatrix} 0 & 0 & 10 \\ 5 & 10 & 0 \\ 0 & 5 & 5 \end{bmatrix}$$

Step 4: since the minimal number of lines is 3, an optimal assignment of zeros is possible and the task is completed.

Because the total cost for this assignment is 0, it must be an optimal assignment:

$$\begin{bmatrix} 0 & 0 & 10 \\ 5 & 10 & 0 \\ 0 & 5 & 5 \end{bmatrix}$$

Here is the same assignment using the original cost matrix.

$$\begin{bmatrix} 25 & 40 & 35 \\ 40 & 60 & 35 \\ 20 & 40 & 25 \end{bmatrix}$$

Example 2:

A taxi company organises four customers to be picked up by four taxis. The distance in kilometres between the customers and the taxi cars are given below.

Table 3-3 The distance between customers and taxi cars.

Customers \ Cars	A	B	C	D
1	8.8	7.5	7.5	8
2	3.5	8.2	5.5	6.5
3	12.5	9.5	9	10.5
4	4.5	10.8	9.5	11.2

The company needs to organise the taxis picking up the customers in order to minimize the total distance travelled.

Step 1: subtract 7.5 from row 1, 3.5 from row 2, 9 from row3, and 4.5 from row 4.

$$\begin{bmatrix} 8.8 & 7.5 & 7.5 & 8 \\ 3.5 & 8.2 & 5.5 & 6.5 \\ 12.5 & 9.5 & 9 & 10.5 \\ 4.5 & 10.8 & 9.5 & 11.2 \end{bmatrix} \sim \begin{bmatrix} 1.3 & 0 & 0 & 0.5 \\ 0 & 4.7 & 2 & 3 \\ 3.5 & 0.5 & 0 & 1.5 \\ 0 & 6.3 & 5 & 6.7 \end{bmatrix}$$

Step 2: subtract 0 from column 1, 0 from column 2, 0 from column 3, and 0.5 column 4.

$$\begin{bmatrix} 1.3 & 0 & 0 & 0.5 \\ 0 & 4.7 & 2 & 3 \\ 3.5 & 0.5 & 0 & 1.5 \\ 0 & 6.3 & 5 & 6.7 \end{bmatrix} \sim \begin{bmatrix} 1.3 & 0 & 0 & 0 \\ 0 & 4.7 & 2 & 2.5 \\ 3.5 & 0.5 & 0 & 1 \\ 0 & 6.3 & 5 & 6.2 \end{bmatrix}$$

$$\begin{bmatrix} 1.3 & 0 & 0 & 0 \\ 0 & 4.7 & 2 & 2.5 \\ 3.5 & 0.5 & 0 & 1 \\ 0 & 6.3 & 5 & 6.2 \end{bmatrix}$$

Step 3: draw all the zeros of the matrix with the minimum number of horizontal or vertical lines.

Step 4: since the minimal number of lines is less than 4, it is necessary to proceed to step 5.

Step 5: 0.5 is the smallest entry which is not covered by any line. Subtract 0.5 from each row not covered by a line.

$$\begin{bmatrix} 1.3 & 0 & 0 & 0 \\ 0 & 4.7 & 2 & 2.5 \\ 3.5 & 0.5 & 0 & 1 \\ 0 & 6.3 & 5 & 6.2 \end{bmatrix} \sim \begin{bmatrix} 1.3 & 0 & 0 & 0 \\ -0.5 & 4.2 & 1.5 & 2 \\ 3 & 0 & -0.5 & 0.5 \\ -0.5 & 5.8 & 4.5 & 5.7 \end{bmatrix}$$

Then add 0.5 to each covered column.

$$\begin{bmatrix} 1.3 & 0 & 0 & 0 \\ -0.5 & 4.2 & 1.5 & 2 \\ 3 & 0 & -0.5 & 0.5 \\ -0.5 & 5.8 & 4.5 & 5.7 \end{bmatrix} \sim \begin{bmatrix} 1.8 & 0 & 0.5 & 0 \\ 0 & 4.2 & 2 & 2 \\ 3.5 & 0 & 0 & 0.5 \\ 0 & 5.8 & 5 & 5.7 \end{bmatrix}$$

Then return to step 3. Draw all the zeros of the matrix with the minimum number of horizontal or vertical lines.

$$\begin{bmatrix} 1.8 & 0 & 0.5 & 0 \\ 0 & 4.2 & 2 & 2 \\ 3.5 & 0 & 0 & 0.5 \\ 0 & 5.8 & 5 & 5.7 \end{bmatrix}$$

Step 4: again, since the minimal number of lines is less than 4, go to step 5.

Step 5: 2 is the smallest entry which is not covered by a line. Subtract 2 from each row without a line.

$$\begin{bmatrix} 1.8 & 0 & 0.5 & 0 \\ 0 & 4.2 & 2 & 2 \\ 3.5 & 0 & 0 & 0.5 \\ 0 & 5.8 & 5 & 5.7 \end{bmatrix} \sim \begin{bmatrix} 1.8 & 0 & 0.5 & 0 \\ -2 & 2.2 & 0 & 0 \\ 3.5 & 0 & 0 & 0.5 \\ -2 & 3.8 & 3 & 3.7 \end{bmatrix}$$

Then add 2 to each covered column.

$$\begin{bmatrix} 1.8 & 0 & 0.5 & 0 \\ -2 & 2.2 & 0 & 0 \\ 3.5 & 0 & 0 & 0.5 \\ -2 & 3.8 & 3 & 3.7 \end{bmatrix} \sim \begin{bmatrix} 3.8 & 0 & 0.5 & 0 \\ 0 & 2.2 & 0 & 0 \\ 5.5 & 0 & 0 & 0.5 \\ 0 & 3.8 & 3 & 3.7 \end{bmatrix}$$

Now return to step 3. Draw all the zeros of the matrix with the minimum number of horizontal or vertical lines.

$$\begin{bmatrix} 3.8 & 0 & 0.5 & 0 \\ 0 & 2.2 & 0 & 0 \\ 5.5 & 0 & 0 & 0.5 \\ 0 & 3.8 & 3 & 3.7 \end{bmatrix}$$

Step 4: since the minimal number of lines is 4, an optimal assignment of zeros is possible; the assignment is then finished. Since the total cost for this assignment is 0, it must be an optimal assignment.

$$\begin{bmatrix} 3.8 & 0 & 0.5 & [0] \\ 0 & 2.2 & [0] & 0 \\ 5.5 & [0] & 0 & 0.5 \\ [0] & 3.8 & 3 & 3.7 \end{bmatrix}$$

Here is the same assignment using the original cost matrix.

$$\begin{bmatrix} 8.8 & 7.5 & 7.5 & [8] \\ 3.5 & 8.2 & [5.5] & 6.5 \\ 12.5 & [9.5] & 9 & 10.5 \\ [4.5] & 10.8 & 9.5 & 11.2 \end{bmatrix}$$

Therefore, the customer 1 is assigned to car D, customer 2 to car C, customer 3 to car B, and customer 4 to car A.

3.6.3 The Hungarian assignment method for multiple object tracking

Normally, multiple object tracking uses the prediction model to identify the same objects in successive image frames. In a frame, there are predictions of objects from previous tracking, and detections are from the current frame. It is usual for the shortest distance between a prediction and a detection to be the same. In such an instance, the Hungarian method is used to allocate the predictions and detections with optimal assignment.

The cost matrix for the Hungarian method contains the distances between predictions and detections. Table 3-4 is the example of the cost matrix.

Table 3-4 The cost matrix, the content is the distance (pixels) between predictions and detections.

Distances	Detection 1	Detection 2	… …	Detection n
Prediction 1	c_{11}	c_{12}	… …	c_{1n}
Prediction 2	c_{21}	c_{22}	… …	c_{2n}
… …	… …	… …	… …	… …
Prediction n	c_{n1}	c_{n2}	… …	c_{nn}

The cost matrix is used to find the optimal assignment. However, sometimes the number of predictions is greater than the number of detections, because some objects disappear on the video. Alternatively, the number of detections may be greater than predictions, because some new objects have just appeared. Another situation is that one object disappears at the same time that another object makes its first appearance. Although the number of predictions and detections are the same, the optimal assignment may not identify the situation correctly.

The solution is to set a maximum pixel distance for assigning a prediction to a detection. If all of the distances of a prediction from the detections are greater than the maximum limit, this

prediction cannot be assigned, so it is not put into the cost matrix for the Hungarian method. Similarly, a detection that is more than the maximum distance from all predictions cannot be assigned and this detection is not put into the cost matrix.

The Hungarian method can be used with the cost matrix to assign other predictions and detections to get the optimal assignment. These assignments identify the same object in the neighbouring image frames.

3.7 Blob analysis with image moments on binary image

Image segmentation includes many methods for object detection, but it can be unsuccessful if the object is too small. The final aim of this research is to detect pollen. The pollen sacs is too small to be detected directly on the whole video frame. Therefore, it is necessary to detect pollen on the bee's detection image, because the bee detection image is much smaller than the video frame. The bee detection result produces the bee's blob as a binary image. The smaller binary image is analysed to detect the pollen.

The analysis method is blob analysis using image moments. The image moment is a particular weighted average of the image pixel's intensities. It is used to calculate blob parameters after segmentation. The properties of a blob which are found from image moments include the blob area, centroid and orientation. For a 2-D continuous function $f(x,y)$, the moment is calculated as:

$$M_{i,j} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^i y^j f(x,y)dxdy \qquad 3\text{-}71$$

where x and y are the coordinates of each pixel. For an image, the x and y values are discrete. Adapting this to a greyscale image with pixel intensities $I(x,y)$, the raw image moments are calculated by:

$$M_{pq} = \sum_x \sum_y x^p y^q I(x,y) \qquad 3\text{-}72$$

If $f(x,y)$ is a digital image, the moments are the following properties of the blob:

Area of the blob:

$$M_{00} = \sum_{x,y} f(x,y) \qquad 3\text{-}73$$

Centroid of the blob region:

$$\{\bar{x},\bar{y}\} = \left\{\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}}\right\} \qquad 3\text{-}74$$

The central moments are defined as:

$$\omega_{pq} = \sum_x \sum_y (x-\bar{x})^p (y-\bar{y})^q f(x,y) \qquad 3\text{-}75$$

According to (Dey et al., 2002), the central moments of order up to 2 are:

$$\omega_{00} = M_{00} \qquad 3\text{-}76$$
$$\omega_{01} = 0 \qquad 3\text{-}77$$
$$\omega_{10} = 0 \qquad 3\text{-}78$$
$$\omega_{11} = M_{11} - \frac{M_{10}M_{01}}{M_{00}} \qquad 3\text{-}79$$
$$\omega_{20} = M_{20} - \frac{{M_{10}}^2}{M_{00}} \qquad 3\text{-}80$$

$$\omega_{02} = M_{02} - \frac{{M_{01}}^2}{M_{00}} \quad \textit{3-81}$$

The central moments are translational invariant. The image orientation can be derived from the second order central moments and a covariance matrix.

$$\omega'_{20} = \frac{\omega_{20}}{\omega_{00}} = \frac{M_{20}}{M_{00}} - \bar{x}^2 \quad \textit{3-82}$$

$$\omega'_{02} = \frac{\omega_{02}}{\omega_{00}} = \frac{M_{02}}{M_{00}} - \bar{y}^2 \quad \textit{3-83}$$

$$\omega'_{11} = \frac{\omega_{11}}{\omega_{00}} = \frac{M_{11}}{M_{00}} - \bar{x}\bar{y} \quad \textit{3-84}$$

The covariance matrix of the image $f(x, y)$ is:

$$cov[f(x, y)] = \begin{bmatrix} \omega'_{20} & \omega'_{11} \\ \omega'_{11} & \omega'_{02} \end{bmatrix} \quad \textit{3-85}$$

From the covariance matrix, the eigenvectors of the matrix correspond to the major and minor axes of the image intensity, so the orientation can be extracted from the angle of the eigenvector associated with the largest eigenvalue towards the axis closest to this eigenvector. This angle θ is:

$$\theta = \frac{1}{2} tan^{-1}\left(\frac{2\omega'_{11}}{\omega'_{20} - \omega'_{02}}\right) \quad \textit{3-86}$$

The eigenvalues of the covariance matrix are:

$$\lambda_i = \frac{\omega'_{20} + \omega'_{02}}{2} \pm \frac{\sqrt{4\omega'^2_{11} + (\omega'_{20} - \omega'_{02})^2}}{2} \quad \textit{3-87}$$

The relative difference in magnitude of the eigenvalues are thus an indication of the eccentricity of the image. The eccentricity is:

$$\sqrt{1 - \frac{\lambda_2}{\lambda_1}} \quad \textit{3-88}$$

If the image $f(x, y)$ is a binary image, the properties of major axis, minor axis and orientation are the 2-D blob information. It can be seen that these simple properties produce the standard ellipse. The major and minor axis of the blob are the major and minor axis of the ellipse, respectively. The orientation of the blob is the rotation angle of the ellipse. The eccentricity of the image is the ellipse's eccentricity. The ellipse is used to detect the main body of the blob, which is discussed in the next chapter.

3.8 The Receiver operating characteristic

The principle of the receiver operating characteristic (ROC) is based on a binary classification model, which is introduced by (Fawcett, 2006). Table 3-5 shows the classification model. The binary classifier outcomes are labelled either as positive or negative. The "Predicted condition" is from a test or a system decision. The true decision is positive and the false is negative. The "True condition" is the actual situation of the instance. If the actual situation is true, it is positive, otherwise it is negative. For example, in a diagnostic test, if a person is tested as having a certain

disease, this is a positive result of the test, otherwise it is a negative result (Zhu et al., 2010). In the true condition, if the person actually has the disease, it is positive, otherwise it is negative.

Table 3-5 A binary classification model

		Predicted condition	
		Condition positive	Condition negative
Actual condition	Condition positive	True positive (TP)	False negative (FN)
	Condition negative	False positive (FP)	True negative (TN)

There are four possible outcomes, when the predicted condition and the true condition are combined. If the outcome from a prediction is positive and the true value is also positive, then it is a true positive (TP). However, if the true value is negative, then the outcome is a false positive (FP). Conversely, a true negative (TN) occurs when both the prediction outcome and the true value are negative, and a false negative (FN) is when the prediction outcome is negative while the true value is positive. With these four outcomes, Table 3-5 is also called contingency table or confusion matrix (Fawcett, 2006).

These four outcomes are used to calculate various standard terms that are used in this research:

Sensitivity (SNS) or true positive rate (TPR) is:

$$Sensitivity = \frac{TP}{TP+FN} \quad \textit{3-89}$$

Specificity (SPC) or true negative rate (TNR) is:

$$SPC = \frac{TN}{FP+TN} \quad \textit{3-90}$$

Precision or positive predictive value (PPV) is:

$$Precision = \frac{TP}{TP+FP} \quad \textit{3-91}$$

False positive rate (FPR) or 1-SPC is:

$$(1 - SPC) = \frac{FP}{FP+TN} \quad \textit{3-92}$$

In binary classification, the class prediction for each instance is often made based on a continuous random variable (X). Given a threshold parameter (T), the instance is classified as "positive" if $X > T$, and "negative" otherwise. Suppose X has the probability density function (pdf) $f_1(x)$ if the instance actually belongs to class "positive", and otherwise X has the pdf $f_0(x)$. Therefore, the true positive rate when the threshold is T is given by

$$TPR\ (T) = \int_T^{\infty} f_1(x)dx \quad \textit{3-93}$$

and the false positive rate is given by

$$FPR\ (T) = \int_T^{\infty} f_0(x)dx \quad \textit{3-94}$$

The choice of T requires a tradeoff between TPR and FPR. Ideally, the desired result is TPR = 100% and FPR = 0%, but increasing T decreases both TPR and FPR. The receiver operating characteristic (ROC) provides a way of estimating the optimum T value by making this tradeoff visible.

The ROC graph (space) parametrically plots $\mathrm{TPR}\,(T)$ versus $\mathrm{FPR}\,(T)$ with threshold *T* as the varying parameter. The ROC space plots FPR on the x axis and TPR on the y axis, to create a 2D space as in Figure 3.14. The best possible prediction method would yield a point in the upper left corner or coordinate (0,1) of the ROC space, representing 100% sensitivity (TPR = 100%) and 100% specificity (FPR = 0%). This point corresponds to a perfect classification. For a completely random classifier, changing the threshold *T* from minimum to maximum, would produce a diagonal line (the line of no-discrimination) from the left bottom corner to the top right corner (the blue dashed line in Figure 3.14). The diagonal divides the ROC space into two parts. Points above the diagonal represent good classification results (better than random); points below the line represent poor results (worse than random).

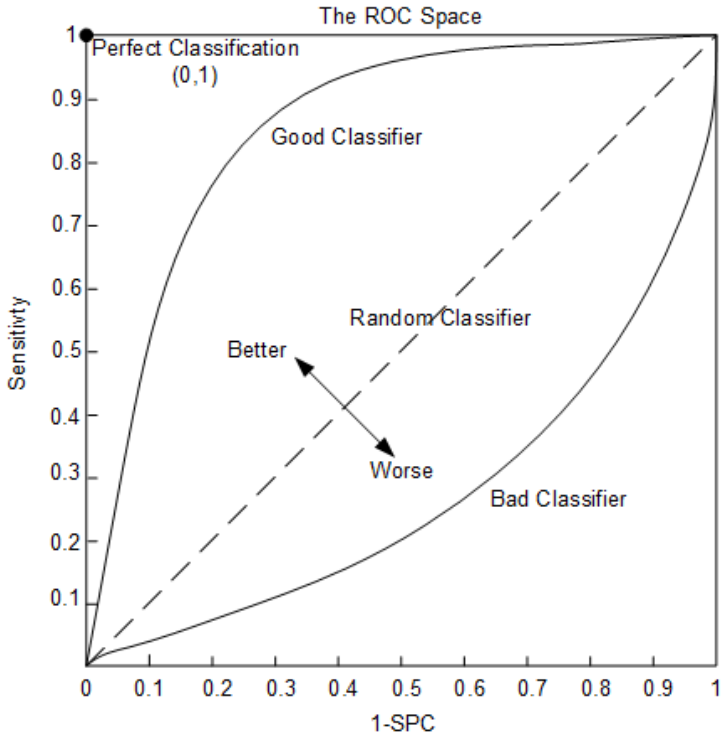


Figure 3.14 The example of ROC space

Changing threshold *T* on a classifier traces a curve on the ROC space. The curve above the diagonal represents a good classifier; the curve under the diagonal represents a bad classifier. For a good classifier, the best threshold value is the point on the curve which is closest to the perfect classification point. This point maximises TPR–FPR which is the same as maximising SNS+SPC.

This binary classification is used in this research to evaluate the results of bee tracking and pollen counting. In addition, the ROC curve is used in this research to calculate optimal thresholds for pollen detection.

3.9 Convolutional deep neural networks

Deep learning is a class of machine learning algorithms that use a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Deep learning architectures are based on an artificial neural network. The convolutional deep neural network (CNN) is one of

the deep learning architectures which is commonly used in computer vision. This network is used in this research for pollen sac detection. This CNN is inspired from the biological structure of a visual cortex, which contains arrangements of simple and complex cells. These cells are found to activate based on the subregions of a visual field. In CNN, the neurons in a convolutional layer connect to the subregions of the layers before that layer instead of being fully-connected as in other types of neural network (van Gerven and Bohte, 2018). The neurons are unresponsive to the areas outside of these subregions in the image. These subregions might overlap, hence the neurons of a CNN produce spatially-correlated outcomes. A CNN consists of multiple layers, such as convolutional layers, max-pooling or average-pooling layers, and fully-connected layers.

3.9.1 The architecture of CNNs

There are a variety of architectures of CNNs. The main layers include: input layer, convolutional layer, ReLU layer, Max and Average-Pooling layer, fully connected layer and output layer.

Image input layer. This layer defines the size of the input images of a convolutional neural network and contains the raw pixel values of the images. The size of an input image corresponds to the height, width, and the number of colour channels of the image. For example, for a grayscale image, the number of channels is 1; and for a colour image, it is 3.

Convolutional (conv.) layer. A convolutional layer consists of neurons that connect to subregions of the input images or the outputs of the layer before it (Murphy, 2012). A convolutional layer learns the features localized by these regions while scanning through an image. For each region, a dot product of the weights and bias are added. A set of weights that are applied to a region in the image is called a filter. The filter moves along the input image vertically and horizontally, repeating the same computation for each region. That is the convolution operation. The convolution operation is defined as follows: given $H = (h_{i,j}^{(k)})_{m\times m}$ at level *k*, and *g*(•) is a nonlinear function, a convolution operation is:

$$h_{i,j}^{(k+1)} = g(a^{(k)} \times h_{i,j}^{(k)} + b^{(k)} \times h_{i+1,j}^{(k)} + c^{(k)} \times h_{i,j+1}^{(k)} + d^{(k)} \times h_{i+1,j+1}^{(k)}) \qquad \textit{3-95}$$

where the a, *b*, *c* and *d* are the weights of a 2×2 filter. In the CNNs, the filter size can be 3×3, 5×5 and 7×7, etc. In deep learning, the convolution operation '$\star$' is also denoted as:

$$s(t) = (x \star \omega)(t) = \sum_{a=-\infty}^{\infty} x(a)\, \omega(t-a) \qquad \textit{3-96}$$

where the function $x(a)$ is input, $\omega(t)$ is a filter, and the output $s(t)$ represents a feature map. For an image I, the convolution operation is:

$$S(i,j) = (I \star K)(i,j) = \sum_m \sum_n I(m,n) K(i-m, j-n) \qquad \textit{3-97}$$

where the K is the 2-dimensional filter. The number of weights used for a filter is $h \times w \times c$, where h is the height, and w is the width of the filter size, and c is the number of channels in the input. The number of filters determines the number of channels in the output of a convolutional layer. As a filter moves along the input, it uses the same set of weights and bias for the convolution, forming a feature map. Hence, the number of feature maps a convolutional layer has is equal to the number of filters (number of channels). Each feature map has a different set of weights and a bias. Therefore, the total number of parameters in a convolutional layer is $((h \times w \times c +$

$1) \times number\ of\ filters)$, where 1 is for the bias. Zero padding can be applied to input image borders vertically and horizontally. The padding is basically adding rows or columns of zeros to the border of an image input, which can help to control the output size of the layer.

A convolutional neural network consists of one or more convolutional layers. The number of convolutional layers depends on the amount and complexity of the data.

ReLU layer. A convolutional layer is usually followed by a nonlinear activation function such as a rectified linear unit (ReLU) (Nair and Hinton, 2010), specified by a ReLU layer. A ReLU layer performs a threshold operation to each element, where any input value less than zero is set to zero, that is:

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \qquad \textit{3-98}$$

The ReLU layer does not change the size of its input. There are extensions of the standard ReLU layer that perform slightly different operations and can improve performance for some applications. A leaky ReLU layer performs a threshold operation, where any input values less than zero is multiplied by a fixed scalar, allowing these values to “leak” into the output (Maas et al., 2013). A clipped ReLU layer sets negative inputs to zero, but also sets input values above a clipping ceiling equal to that clipping ceiling (Hannun et al., 2014).

pooling layer. pooling layer follows the convolutional layers for down-sampling, hence reducing the number of connections to the following layers (usually a fully connected layer). They do not perform any learning themselves, but reduce the number of parameters to be learned in the following layers. They also help reduce overfitting. A pooling layer can be max-pooling layer or average-pooling layer.

A max-pooling layer returns the maximum values of rectangular regions of its input (Nagi et al., 2011). If the rectangle is 2×2, then the layer returns the maximum value in regions of height 2 and width 2. Taking the feature map of $S(i,j)$ from (3-97) as an input example, the output of the max-pooling layer is:

$$S_{max} = max(S(i,j),\ S(i+1,j),\ S(i,j+1),\ S(i+1,j+1)) \qquad \textit{3-99}$$

Similarly, the average-pooling layer outputs the average values of rectangular regions of input. If the rectangular is 2×2, then the layer returns the average value of regions of height 2 and width 2. Using the same input example, the output of the average-pooling layer is:

$$\bar{S} = \frac{1}{4}(S(i,j) + S(i+1,j) + S(i,j+1) + S(i+1,j+1)) \qquad \textit{3-100}$$

If the input to the pooling layer is *n*-by-*n*, and the pooling region size is h-by-h, then the pooling layer down-samples the region by *h*. In the other words, the output of a max- or average-pooling layer for one channel of a convolutional layer is *n*/*h*-by-*n*/*h*.

Fully connected layer (FC). The convolutional and down-sampling layers are followed by one or more fully connected layers. All neurons in a fully connected layer connect to all the neurons in the previous layer. This layer combines all of the features learned by the previous layers across the image to identify larger patterns. For image classification problems, the last fully connected

layer combines the features to classify the images. Therefore, the outputs of the last fully connected layer of the network are the classes of the data set (or images).

Output layer. The output layer includes soft-max and classification layers. For image classification problems, a soft-max layer and then a classification layer must follow the final fully connected layer. The output unit activation function is the soft-max function (Tang, 2013):

$$y_r(x) = \frac{exp(a_r(x))}{\sum_{j=1}^{k} exp(a_j(x))} \qquad \textit{3-101}$$

Where $0 \leq y_r \leq 1$ and $\sum_{j=1}^{k} y_j = 1$.

The soft-max function is the output unit activation function after the last fully connected layer for multi-class classification problems.

A classification output layer must follow the soft-max layer. The classification output layer takes the values from the soft-max function and assigns each input to one of the *m* mutually exclusive classes using the cross entropy function for a 1-of-*m* coding scheme (Bishop, 2012).

In summary, the layers introduced above are the main structures of the CNN. There are other layers which have different aims. For example, a **cross channel normalization layer** following ReLU layer replaces each element with a normalized value it obtains using the elements from a certain number of neighbouring channels (elements in the normalization window). A **dropout layer** randomly sets the layer's input elements to zero with a given probability, which can help prevent overfitting. In next section, an example of VGG network is introduced for more explanation of CNN's architecture.

3.9.2 The VGG networks

The VGG networks are convolutional deep networks, which include from 13 to 19 layers (Simonyan and Zisserman, 2014). These networks are especially designed for image classification, which can recognise 1000 objects (classes). This section introduces the architecture of the VGG network.

The input image to the VGG network is a fixed-size 224×224 RGB image. The image is passed through a stack of convolutional layers, where the VGG network uses filters with small size of 3×3. All these convolutional layers (or hidden layers) are equipped with the ReLU layer. Spatial pooling is carried out by five max-pooling layers, which follow some of the convolutional layers (not all the convolutional layers are followed by max-pooling). Max-pooling is performed over a 2×2 pixel window. The stack of convolutional layers is followed by three fully connected (FC) layers: the first two have 4096 channels each; the third performs 1000-way classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer.

A structure of VGG network is shown in Figure 3.15. This is a VGG-16 network, which includes 13 convolutional (conv.) layers and 3 fully connected (FC) layers. The number of 16 means 16 weight layers.

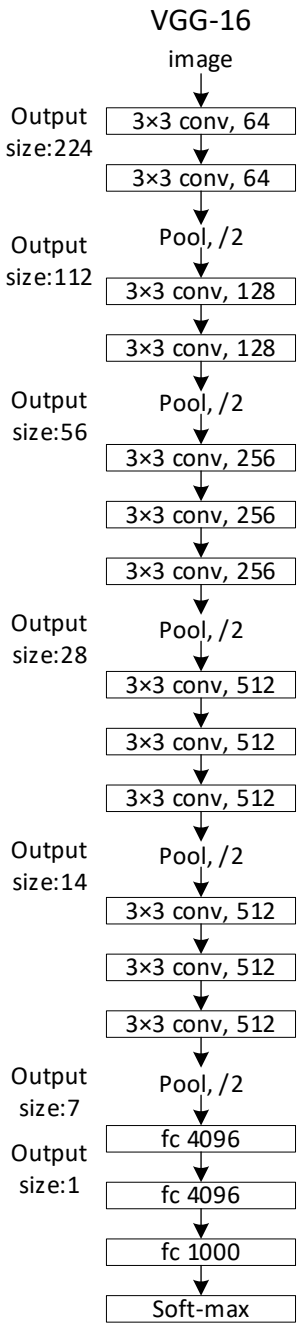


Figure 3.15 The structure of VGG-16 network.

In Figure 3.15, the width of conv. layers (the number of channels) is rather small, starting from 64 in the first conv. layer and then increasing by a factor of 2 after each max-pooling layer, until it reaches 512. The number of parameters in this network is about 138 million.

3.9.3 Network training and learning

Commonly, the neural network training is to find the optimal values of parameters (weights) for prediction of the image classes. In image classification, a large data set of images is collected, each image labelled with its category (or class), such as house, car, person and pet. During training, the network is shown an image and produces an output in the form of a vector of scores, one for each class. The desired class should have the highest score of all classes, but this is unlikely to happen before training. An objective function (or loss function) is computed to measure

the error between the output scores and the desired class scores. The network then modifies its weights to reduce error. In a deep learning system, there may be hundreds of millions of these weights, and hundreds of millions of labelled examples with which to train the network.

To properly adjust these weights (the weight vector), the learning algorithm computes a gradient vector that, for each weight, indicates by what amount the error would increase or decrease if the weight were increased by a tiny amount. The weight vector is then adjusted in the opposite direction to the gradient vector.

The gradient vector (or gradients) in multiple layers of a neural network are propagated by the backpropagation algorithm (LeCun et al., 2015). This algorithm uses the chain rule of derivatives for the gradients propagating. If vector $\boldsymbol{y} = g(\boldsymbol{x})$ (where $\boldsymbol{x}$ is the input vector), and $z = f(\boldsymbol{y})$ then,

$$\frac{\partial z}{\partial x_i} = \frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_i} \qquad \textit{3-102}$$

where x_i (i = 1,$\cdots$, m) and y_j (j = 1,$\cdots$, n) are the elements of vector $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively.

In neural network training, the backpropagation algorithm can be applied repeatedly to propagate gradients through all layers, starting from the output at the top (where the network produces its prediction) all the way to the bottom (input). Once these gradients have been computed, it is straightforward to compute the gradients with respect to the weights of each layer.

The weight adjustment method is gradient descent (GD). This algorithm updates the network weights to minimize the loss function by taking small steps in the direction of the negative gradient of the loss. If the loss function is

$$J(\boldsymbol{\theta}) = L(f_{\boldsymbol{\theta}}(x_i), y_i) \qquad \textit{3-103}$$

where (x_i, y_i) are inputs and outputs (i = 1, $\cdots$, m) of a network, and $\boldsymbol{\theta}$ is the weight vector. The GD is defined as:

$$\boldsymbol{\theta}_{\ell+1} \coloneqq \theta_\ell - \alpha \cdot \frac{\partial}{\partial \boldsymbol{\theta}_\ell} J(\boldsymbol{\theta}_\ell) \qquad \textit{3-104}$$

where ℓ is the iteration number, $\alpha > 0$ is the learning rate. The $\frac{\partial}{\partial \boldsymbol{\theta}_\ell} J(\boldsymbol{\theta}_\ell)$ (or $\nabla J(\boldsymbol{\theta})$) is the gradient of the loss function which is evaluated using he entire training set. The standard gradient descent algorithm uses the entire data set at once. Another weight adjustment approach is called stochastic gradient descent (SGD) (Bottou, 2010). This is the most common method for weight adjustment in CNN network training. This algorithm evaluates the gradient and updates parameters using a subset of the training set. The subset is called a mini-batch. Each evaluation of the gradient using the mini-batch is an iteration. At each iteration, the algorithm takes one step towards minimizing the loss function. The full pass of the training algorithm over the entire training set using mini-batches is an epoch.

The training algorithm and procedure are introduced above. It can be seen that the training is a complicated procedure which uses hundreds of millions of training data to adjust hundreds of millions of weights. Therefore, the training includes different policies and operations. These policies can be controlled by some parameters. Some of the main learning parameters are introduced below.

Mini-batch. As mentioned above, the SGD updates the network weights by taking small steps in the direction of the negative gradient of the loss function. It updates the weights using a subset of the data each step. This subset is called a mini-batch. Each update is called an iteration. The entire data set for deep learning is very large, and if it is all given to a neural network for training at the same time, the training will be very slow. To overcome this problem, the entire data is divided to smaller mini-batches which are given to the network one by one. This procedure updates the weights of the neural network at the end of every step.

Epoch. An iteration is one step taken in the gradient descent algorithm towards minimizing the loss function using a mini-batch. An epoch is the full pass of the training algorithm over the entire training set. As the number of epochs increases, the weights are updated in the neural network and the weights become a better fit to the training data. However too many epochs may cause overfitting to the training data.

Learning rate. Learning rate was mentioned in equation (3-104). This is a hyper-parameter that controls how much the weights are adjusted with respect of the loss gradient. The lower the value, the slower the loss function travels along the downward slope. However, using a lower learning rate can make sure that the loss function dose not miss any local minima, although it takes a long time to converge. If the learning rate is too large, the training may not converge, or even diverge. The weight update can be so big that the optimizer overshoots the minimum and makes the loss function increase. Using different learning rates from large value decreasing to the low value, there will be a point where the loss function stops decreasing and starts to increase (Smith, 2017). This point is the optimal learning rate for training.

The main learning parameters are mentioned above. These parameters affect training speed and convergence. Therefore, the training parameters must be trialled with a variety of values and schedules to find the best way to train the network.

Another training concept is transfer learning (Pan and Yang, 2010). In transfer learning, a pre-trained network is used as a starting point for re-training the network rather than using random initial values of weights as starting points. For example, knowledge gained while learning to recognize cars could apply when trying to recognize trucks. Transfer learning saves a lot of training time and reduces the amount of training data require, because the training starts from patterns that have been learned from solving a related task. In this research, the pre-trained model of VGG-16 network is given additional training to detect pollen sacs on individual bee images. The VGG-16 network in MATLAB was pre-trained by ImageNet (Russakovsky et al., 2015) data set, which consists of 1000 different objects.

3.9.4 The fast region-based convolutional neural network (Fast RCNN)

The convolutional deep neural networks (CNNs) are designed for image classification. The region-based convolutional neural network (RCNN) is designed to detect objects on an entire image (Girshick et al., 2014). Based on the concept of RCNN, a Fast RCNN is built and modified for more accurate object detection.

The Fast RCNN network takes as inputs an entire image and a set of object proposals (Girshick, 2015). The network first processes the whole image with several convolutional (conv.) and max-

pooling layers to produce a conv feature map. Then, for each object proposal a region of interest (ROI) pooling layer extracts a fixed-length feature vector from the feature map. Each feature vector is fed into a sequence of fully connected (fc) layers that finally branch into two different output layers: a soft-max layer and a bounding box (bbox) regressor layer. The soft-max layer produces probability estimates over multiple object classes plus a catch-all “background” class. The bounding box regressor layer outputs four real-valued numbers, which include bounding box position information for each of the multiple object classes. The architecture is shown in Figure 3.16.

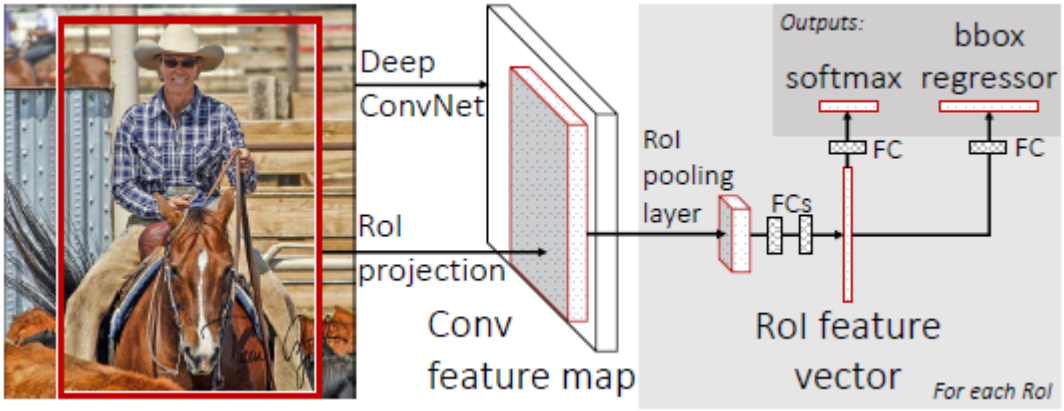


Figure 3.16 The architecture of Fast RCNN (Girshick, 2015)

In Figure 3.16, the deep convolutional network (Deep ConvNet) and the object proposal (ROI projection) are similar to RCNN (Girshick et al., 2014). After the feature map produced by the convolution, there is a ROI pooling layer. This layer uses max pooling to convert the features inside any valid ROI into a small feature map with a fixed spatial extent of $H \times W$ (e.g., 7×7 for VGG-16), where H and W are layer hyper-parameters which are independent of any particular ROI. For instance, a ROI is a rectangular window into a convolutional feature map. Each ROI is defined by a four-tuple (r, c, h, w) that specifies the top-left corner (r, c) and the height and width (h, w).

ROI max pooling works by dividing the $h \times w$ ROI window into a $H \times W$ grid of sub-window of approximate size $\frac{h}{H} \times \frac{w}{W}$ and then max-pooling the values in each sub-window into the corresponding output grid cell. Pooling is applied independently to each feature map channel, as in standard max pooling.

According to the architecture of the Fast RCNN, it has a special training policy. In the Fast RCNN training, stochastic gradient descent (SGD) mini-batches are sampled hierarchically, first by sampling images (for example *N* images) and then by sampling ROIs (for example *R*/*N* ROIs) from each image. Critically, ROIs from the same image share computation and memory in the forward and backward passes. In addition, Fast RCNN uses a streamlined training process with one fine-tuning (Girshick et al., 2014) stage that jointly optimizes a soft-max classifier and bounding box regressors. The components of the detail procedure (such as loss function, mini-batch sampling strategy, back-propagation through ROI pooling layers) are described below.

Multi-task loss. The Fast RCNN network has two sibling output layers. The soft-max layer outputs a discrete probability distribution (per ROI), $p = (p_0, \dots, p_K)$, over $K + 1$ categories. The probability p is computed by a soft-max over the $K + 1$ outputs of a fully connected layer. The bbox regressor layer outputs bounding box regression offsets, $b^k = (b_{\mathrm{x}}^k,\ b_{\mathrm{y}}^k,\ b_{\mathrm{w}}^k,\ b_{\mathrm{h}}^k)$, for each of the K object classes, indexed by k. The b^k specifies a scale-invariant translation and log-space height/width shift relative to an object proposal.

Each training ROI is labelled with a ground-truth class u and a ground-truth bounding box regression target v. The network training uses a multi-task loss L on each labelled ROI to jointly train for classification and bounding box regression:

$$L(p, u, b^u, v) = L_{cls}(p, u) + \delta[u \geq 1]L_{loc}(b^u, v), \qquad \textit{3-105}$$

in which $L_{\mathrm{cls}}(p, u) = -\log p_u$ is log loss for true class u. The second term's loss (L_{loc}) is defined over a tuple of true bounding box regression targets for class u, $v = (v_{\mathrm{x}}, v_{\mathrm{y}}, v_{\mathrm{w}}, v_{\mathrm{h}})$, and a predicted tuple $b^u = (b_{\mathrm{x}}^u,\ b_{\mathrm{y}}^u,\ b_{\mathrm{w}}^u,\ b_{\mathrm{h}}^u)$, for class u. The Iverson bracket indicator function $[u \geq 1]$ evaluates to 1 when $u \geq 1$ and 0 otherwise. By convolution the catch-all "background" class is labelled $u = 0$. For background ROIs there is no concept of a ground-truth bounding box, so the loss L_{loc} is ignored. For bounding box regression, the loss L_{loc} is defined below.

$$L_{loc}(b^u, v) = \sum_{t \in \{x,y,w,h\}} smooth_{L_1}(b_t^u - v_t), \qquad \textit{3-106}$$

in which,

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2 & if\,|x| < 1 \\ |x| - 0.5 & otherwise \end{cases} \qquad \textit{3-107}$$

The hyper-parameter δ in equation (3-105) controls the balance between the two task losses. The ground-truth regression targets v_t are normalized to have zero mean and unit variance. All the training can use $\delta = 1$.

Mini-batch sampling. During the training, the Fast RCNN uses fine-tuning method (Girshick et al., 2014). Each SGD mini-batch is constructed from *N* = 2 images, chosen uniformly at random (commonly, the training can iterate over permutations of the dataset). If using mini-batches of size *R* = 128, the training is sampling 64 ROIs from each image. The training takes 25% of the ROIs from object proposals which have intersection over union (IOU) overlap with a ground-truth bounding box of at least 0.5 (Girshick et al., 2014). These ROIs comprise the examples labelled with a foreground object class, i.e. $u \geq 1$. The remaining ROIs are sampled from object proposals which have a maximum IOU with ground truth in the interval [0.1, 0.5) (Girshick et al., 2014). These are the background examples and are labelled with $u = 0$. The lower threshold of 0.1 appears to act as a heuristic for hard example mining (Felzenszwalb et al., 2010).

Back-propagation through ROI pooling layers. Through the ROI pooling layer, the back propagation routes derivative in the training of Fast RCNN (Girshick, 2015). For example, if there is only one image per mini-batch, then *N* = 1. The extension to *N* > 1 is straightforward because the forward pass treats all images independently.

When the activation inputs go into the ROI pooling layer, the layer max pools outputs on objects' ROIs. A single input may be assigned to several different outputs.

In back-propagation, the partial derivative of the loss function is computed with respect to each input variable through the ROI pooling layer (Girshick, 2015). Then the back-propagation calculates partial derivatives through other layers which are before the ROI pooling layer. The partial derivatives are used by the SGD approach to change the parameters of the network to minimize the loss function. This is the training of the Fast RCNN network.

3.9.5 The Faster RCNN

The Faster RCNN is created from Fast RCNN. Faster RCNN consists of a region proposal network (RPN) (Ren et al., 2015) which shares full-image convolutional features with the detection network (Fast RCNN for example), enabling nearly cost-free region proposals. The RPN is trained end-to-end to generate high-quality region proposals, which are used by Fast RCNN for detection.

The RPN takes an image with any size as input and outputs a set of rectangular object proposals. It is built within a fully-convolutional network. The ultimate goal is to share computation with the Fast RCNN object detection network. Therefore, the RPN and Fast RCNN share a common set of convolutional layers. For example, if the Fast RCNN core network uses VGG-16, then so does the RPN core network. The entire system is a single, unified network for object detection, which is shown in Figure 3.17. The RPN tells the Fast RCNN where to look (Ren et al., 2017).

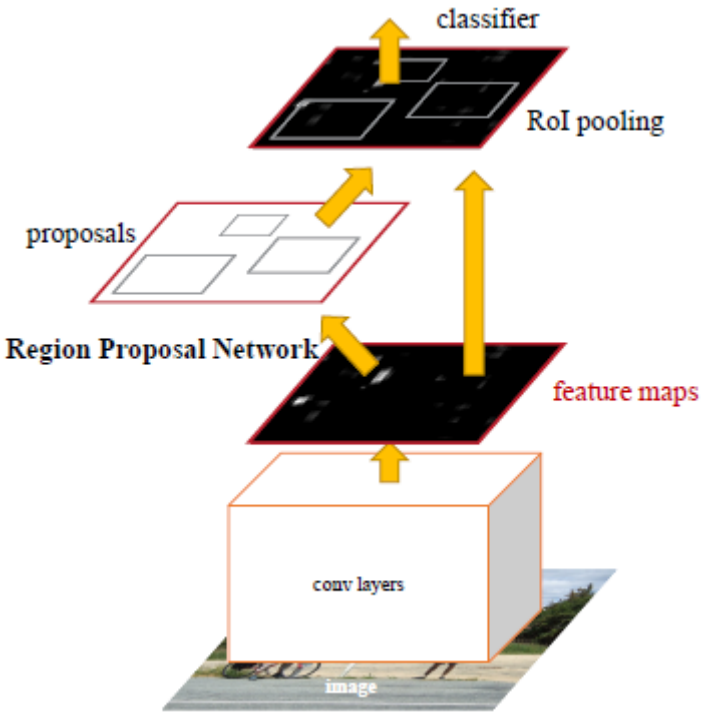


Figure 3.17 Faster RCNN process, which consists of RPN (Ren et al., 2017).

To generate region proposals, a mini-network (VGG-16 for an example) slides over the convolutional feature map output by the last shared convolutional layer. This network is fully connected to an $n \times n$ spatial window of the input convolutional feature map. Each sliding window is mapped to a lower-dimensional vector (512-d for VGG) as an intermediate layer. This layer is

fed into two sibling fully-connected layers: a box regression layer (*reg*) and a box classification layer (*cls*) as shown in Figure 3.18.

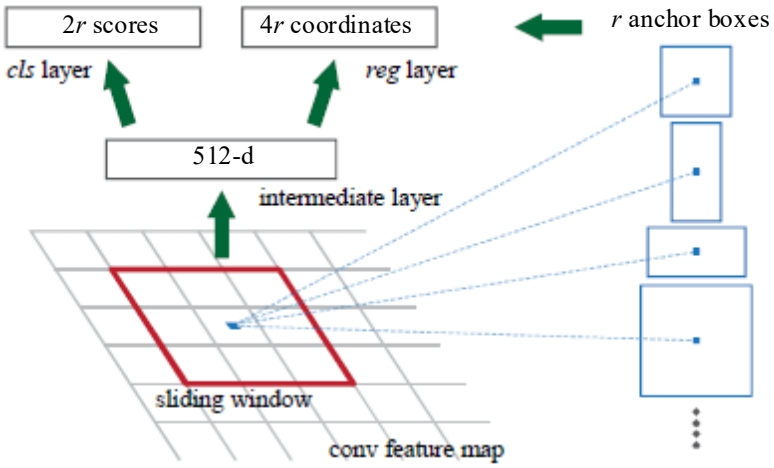


Figure 3.18 The region proposal network (RPN) (Ren et al., 2015).

Using $n = 3$ as an example, the effective receptive field on the input image is large (228 pixels for VGG). This mini-network is illustrated at a single position in Figure 3.18. Because the mini-network operates in a sliding-window, the fully-connected layers are shared across all spatial locations. This architecture is naturally implemented with an $n \times n$ convolutional layer followed by two sliding 1×1 convolutional layers (for *reg* and *cls*, respectively). ReLUs are applied to the output of the $n \times n$ convolutional layer.

At each sliding-window location, if the network simultaneously predicts r region proposals, so the *reg* layer has $4r$ outputs encoding the coordinates of r boxes. The *cls* layer outputs $2r$ scores that estimate the probability of object (or not object) for each proposal. The *cls* layer is implemented as a two-class soft-max layer. Alternatively, one may use logistic regression to produce r scores. The r proposals are parameterized relative to r reference boxes, called anchors. Each anchor is centred at the sliding window in question and is associated with a scale and aspect ratio. The network uses 3 scales and 3 aspect ratios, yielding $r = 9$ anchors at each sliding position. For a convolutional feature map of a size $H \times W$ (typically about 2400), there are $H \times W \times r$ anchors in total. An important characteristic of this model is translation invariant, both in terms of the anchors and the functions which compute proposals relative to the anchors.

In the training of RPN, a binary class label is assigned to each anchor. The positive label is assigned to two kinds of anchors: the anchor (or anchors) with the highest IOU overlap with a ground-truth box; or an anchor that has an IOU overlap higher than 0.7 with any ground-truth box. A negative label is assigned to a non-positive anchor if its IOU ratio is lower than 0.3 for all ground-truth boxes. Anchors that are neither positive nor negative do not contribute to the training objective. With these definitions, the training minimizes an objective function following the multi-task loss in Fast RCNN. The loss function for an image is defined as:

$$L(\{p_\alpha\},\{b_\alpha\}) = \frac{1}{N_{cls}}\sum_\alpha L_{cls}(p_\alpha, p_\alpha^*) + \delta\frac{1}{N_{reg}}\sum_\alpha p_\alpha^* L_{reg}(b_\alpha, b_\alpha^*) \qquad \textit{3-108}$$

In equation (3-108), α is the index of an anchor in a mini-batch and p_α is the predicted probability of anchor α being an object. The ground-truth label p_α^* is 1 if the anchor is positive and is 0 if the anchor is negative. b_α is a vector representing the 4 parameterized coordinates of the predicted bounding box, and b_α^* is the ground-truth box associated with a positive anchor. The classification loss L_{cls} is log loss over two classes (object, or not object). The regression loss is defined as:

$$L_{reg}(b_\alpha, b_\alpha^*) = R(b_\alpha - b_\alpha^*) \qquad \textit{3-109}$$

Where R is the robust loss function (smooth L_1) defined in equation (3-107). The term $p_\alpha^* L_{reg}$ means the regression loss is activated only for positive anchors ($p_\alpha^* = 1$) and is disabled otherwise ($p_\alpha^* = 0$). The outputs of the *cls* and *reg* layers consist of $\{p_\alpha\}$ and $\{b_\alpha\}$ respectively. The two terms are normalized with N_{cls} and N_{reg}, and balancing weight δ. For regression, the parameterizations of the 4 coordinates are:

$$b_x = \frac{(x - x_\beta)}{w_\beta}, b_y = \frac{(y - y_\beta)}{h_\beta}, b_w = log(\frac{w}{w_\beta}), b_h = log(\frac{h}{h_\beta}), \qquad \textit{3-110}$$

$$b_x^* = \frac{(x^* - x_\beta)}{w_\beta}, b_y^* = \frac{(y^* - y_\beta)}{h_\beta}, b_w^* = log(\frac{w^*}{w_\beta}), b_h^* = log(\frac{h^*}{h_\beta}), \qquad \textit{3-111}$$

where x, y, w and h denote the two coordinates of the box centre, width, and height (Girshick et al., 2014). Variables (x, y, w, h) are for the predicted box; $(x_\beta, y_\beta, w_\beta, h_\beta)$ are for anchor box; and (x^*, y^*, w^*, h^*) are for ground-truth box. This is the bounding-box regression from an anchor box to a nearby ground-truth box.

The RPN can be trained end-to-end by back-propagation and stochastic gradient descent (SGD). Each mini-batch arises from a single image that contains many positive and negative anchors. This is the “image centric” sampling strategy (Girshick, 2015).

In the training of the entire Faster RCNN, both RPN and Fast RCNN trained independently. They modify their convolutional layers in different ways. There are 4 steps for training algorithm sharing convolutional features between the two networks. In the first step, the RPN is trained. The network is trained end-to-end for the region proposal task. In the second step, a separate detection network is trained by Fast RCNN using the proposal generated by the last step of RPN. In this case, the two networks do not share convolutional layers. In the third step, the Fast RCNN training from the second step is used to initialize RPN training. Then the training fixes the shared convolution layers and only fine-tune the layers unique to RPN. Now the two networks share convolutional layers. In the final step, the training keeps the shared convolutional layer fixed, and the fc layers of the Fast RCNN are fine-tuned. In this case, both networks share the same convolutional layers and form the unified Faster RCNN network.

Chapter 4

Chapter 4: Methodology of bee detection and tracking

This chapter introduces the equipment and the methods for bee detection and tracking on the honey bee monitoring video. The environment of data collection is first introduced. Then the bee detection approach using image analysis is presented. The main point is the combination of colour thresholding and the foreground (motion) detection. Then the bee tracking method is explained. There are two tracking models: the prediction model and the Kalman filter. In addition, the Hough transform is used to detect bees in a merged situation in the video image.

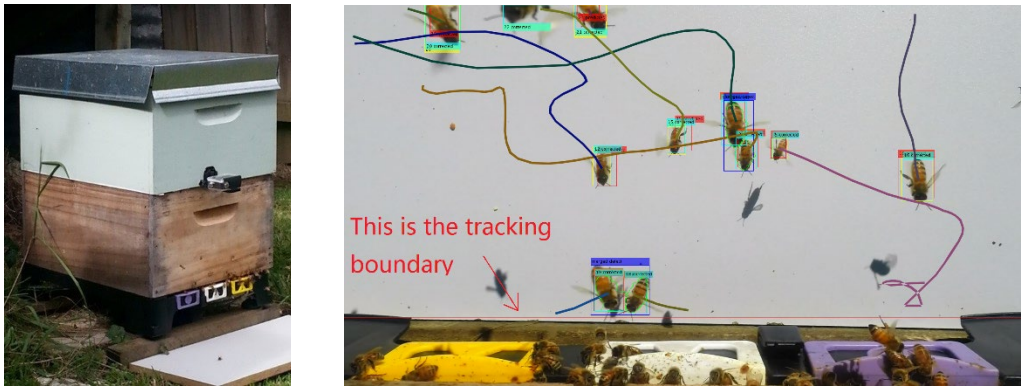
4.1 The research environment and equipment

The species of bee in this experiment is the European bee. These bees' bodies include orange and black stripes. When a bee brings full pollen sacs to the hive, its back legs obviously have sacs. Figure 4.1 (a) shows the beehive and the camera set up situation. The camera is attached to the front wall of the beehive, facing down about 30cm above the entrance. Because the aim of the research is to detect pollen sacs on bee's legs, it needs a high resolution video to record small objects. The video is 1920*1080 resolution at a rate of 50 frames per second (fps). The higher frame rate is better for bee tracking, but the 100 or 200 fps high speed camera is more costly. In addition, a camera with a 30 or 50 frame rate is common, easy to obtain and operate.

The operation of the camera should be convenient, so that the video data is easy to collect. One of the aims of this research is try to detect pollen at most times of the day. Because of the variable climate of the environment, the image of the video includes different situations. Therefore, it has to record the video at different times of the day. One way is to record the video throughout the whole day. However, it is hard to find a big enough battery for the camera to keep operating for more than 10 hours. Even if the camera can be continuously recharged, the video becomes too long to store in the camera memory. It is better to operate the camera on and off at regulated times. Therefore, the camera is programmed to turn on every half hour. Once it is on, the camera records the video for 2 minutes, and then turns itself off automatically. After the next half hour, it repeats the operation. In this way, video is recorded at different times of the day, in different situations.

A single white (or green) colour board was placed on the ground in front of the entrance of beehive, to simplify the image background. This is necessary, because the natural background includes withered grass with a similar colour to the bees and this interferes with the image

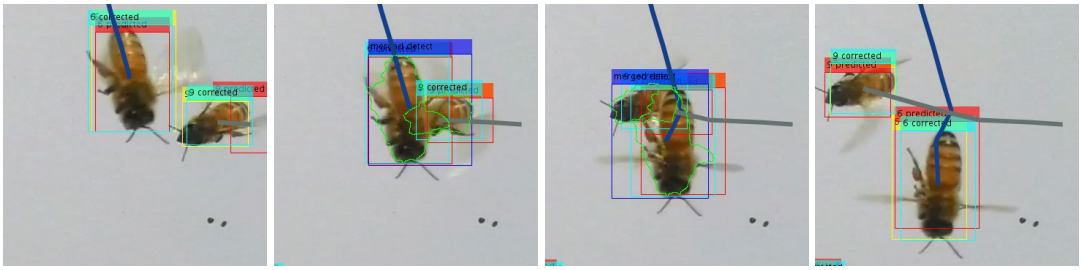
segmentation. The video processing is implemented using MATLAB with the computer vision toolbox.



(a) (b)

Figure 4.1 The camera position (a) and the camera view (b)

A Matlab function performs the blob analysis. It not only calculates the blob information, but also outputs bounding boxes around each bee blob. The bee tracking is shown by adding indexes to each bounding box and curves marking each bee's trajectory with different colours in the video. A tracking boundary is set up just in front of the hive entrance, which is shown by the horizontal red line near the bottom of Figure 4.1 (b), so that the system only tracks bees over the white background area. This is because the area below the boundary line has complex colour information which affects the bee detection. Figure 4.1 (b) shows an example of tracking output over 48 frames (around 1 second later) from beginning of the video. The blue and red bounding boxes mark the detected and predicted bee positions respectively.



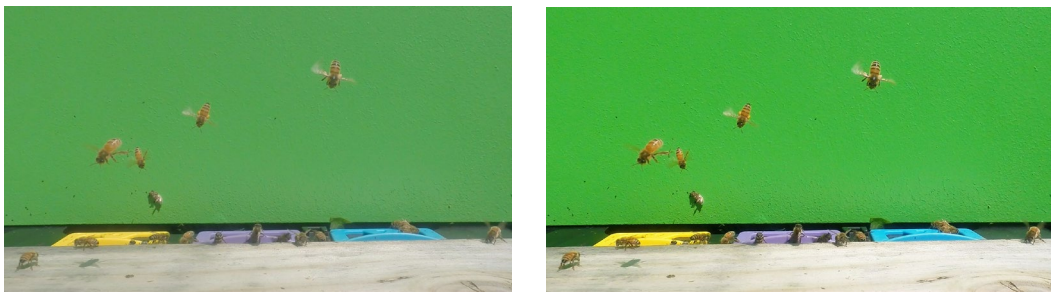
(a) (b) (c) (d)

Figure 4.2 The result of merged situation tracking over successive video frames.

The model implementing tracking in the merged bee situation is shown in Figure 4.2. The green outline shapes show the detections produced by the Hough transform when the bees are merged. The blue bounding boxes show the detection of merged blobs. The images (a) to (d) are successive frames. These examples demonstrate successful tracking.

The camera takes video recordings automatically at different times of the day. When the light changes, the video may not be colourful enough for detection. The camera has a **Protune** operation which can make the video more vivid. Figure 4.3 indicates this operation. This operation can be performed both in hardware and software. The hardware operation charges the camera battery, so that it is unable to record the video throughout the whole day. Therefore, the software

operations are performed offline on a PC, after recording the video. All of the video recordings are enhanced by the Protune operation. This helps to make the colour detection easier.



(a)Video without Portune (b)Video with Portune

Figure 4.3 The Protune operation of the camera

4.2 Bee Detection

The detection of bees is implemented by foreground subtraction and colour base segmentation. Foreground subtraction detects flying bees and ignores the unchanging background. However, this also detects the bees' shadows and the movement of vegetation, which affects the result. The colour segmentation, based on global colour thresholding in the Hue-Saturation-Value (HSV) colour space (Alaux et al., 2010), detects the colours of bees, but it also detects other objects with the same colour as the bees, especially the withered grass. Therefore, combination of the two methods can remove shadows and moving vegetation. Thus, the blob analysis estimates the positions of bees and creates a bounding box around each bee blob.

In addition, after each phase of the object detection, such as colour segmentation and foreground subtraction, the morphological image calculation (or filter) is performed to smooth and de-noise the binary images. These phases are mentioned as a summary after each step of detection. The detail of this method is given in section 3.4.

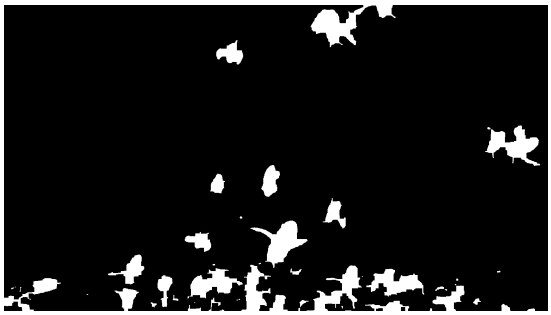
4.2.1 Bee detection with foreground subtraction

The foreground subtraction detects movement objects using the Gaussian Mixture Model (GMM) (Wang et al., 2015b) with the video. In this situation, the Gaussian model has 5 distributions and 40 training frames. Because the model is expected to be a stable Gaussian distribution, a smaller learning rate works well. The initial learning rate is 0.001.

This model is implemented using the Matlab foreground detector function. The colour video has three channels of RGB. The motion is not affected by the colour, so this process converts the image to a grey scale image. After foreground pixel detection in a frame, the model outputs a binary image which displays the foreground pixels as white and the background as black. The foreground (white) pixels display the blobs of moving objects. Most of them are bee blobs. After the morphological opening, closing and filling holes (section 3.4), the binary image is smoothed and most noise is removed.

(a)

(b)



(c)

Figure 4.4 The results of motion detection.

Note: (a) The original video. (b) The result of motion detection. (c) The result after the morphological calculation.

Figure 4.4 displays an example of the foreground segmentation. It can be seen that the shape of the blobs (Figure 4.4b) is not smooth, and it has some dust. After the morphological calculation (image opening, closing and filling holes) which is shown in Figure 4.4(c), the blobs are smoother; and the noise is removed. The white pixels are the blobs of moving bees and the bee shadows. This morphological filter is also applied in colour detection.

From Figure 4.4, it can be seen that this model detects not only bee blobs but also bee shadows. This because the shadows are also moved. These shadows are independent of the bees, so another method is needed to remove the shadow blobs. In Figure 4.4(a), the shadows are dark, but the bees have an orange colour. Therefore, the solution is to use the colour information to remove the shadows.

4.2.2 Colour segmentation for bee detection

The bees have orange and black colours, with the head being black and the abdomen orange and black. The video is RGB colour images. If this colour space is used to detect colour, all three channels have to be used in the calculation. Moreover, the colour is affected by the brightness of the video. However, if it is transformed to the HSV colour space, the calculation is simpler. The Hue and the Saturation channels are calculated in proportion to chroma, and are not affected by the brightness.

Figure 4.5(b) shows the Hue channel of image Figure 4.5(a) with the full resolution of 1080x1920. In Figure 4.5(b), it can be seen that the Hue channel is too noisy to detect bees. However, bee regions in the image are all dark, which means the Hue value is low. In addition, in this hue channel, the background pixels of the image mostly have a high value. Therefore, the Hue value is still useful.

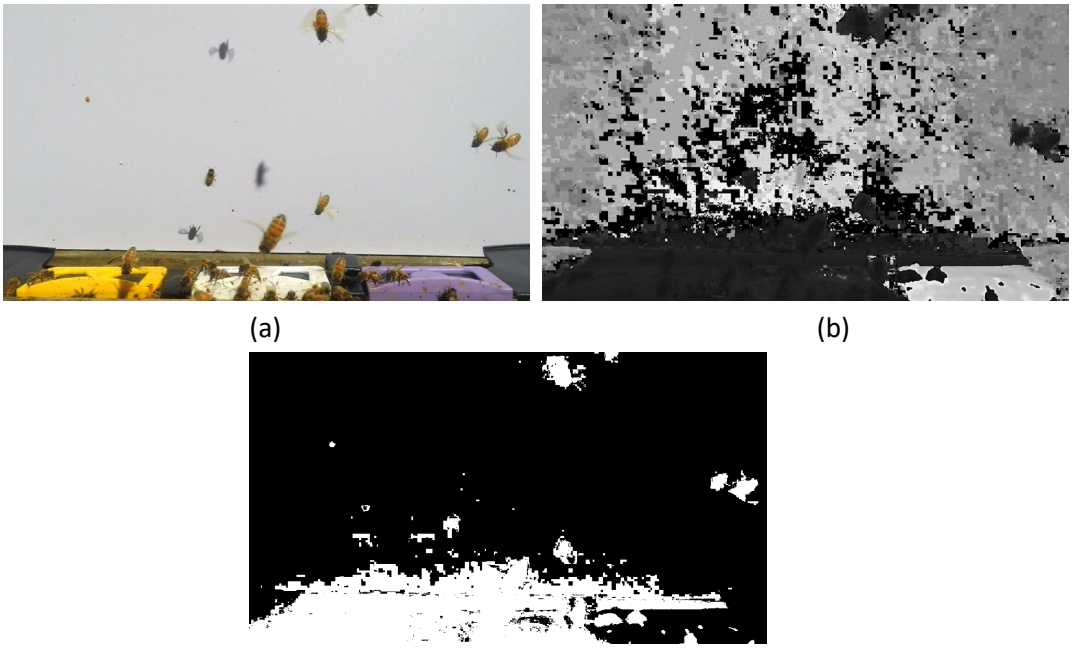
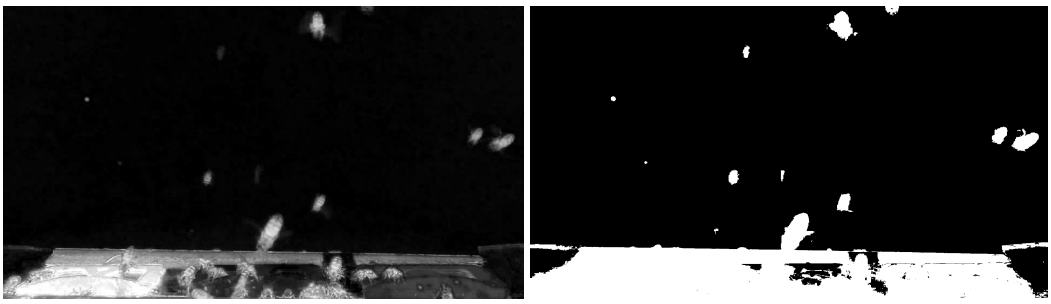


Figure 4.5 The Hue channel example.

Note: (a) The original image. (b) The Hue channel shown as an intensity image. (c) The bee region detection with Hue channel.

The MATLAB function is used to transform from the RGB colour space to the HSV space. It outputs all three channels (H, S and V) with values ranging from 0 to 1. If the Hue channel pixel is $\mathrm{h}(\mathrm{x},\mathrm{y})$, the orange coloured pixel has a Hue value in the range $0.02 < \mathrm{h}(\mathrm{x},\mathrm{y}) < 0.15$. These two threshold values were determined experimentally. Figure 4.5(c) shows the Hue detection of orange coloured pixels. It can be seen that the method did not detect bees correctly. The board is white and the bees have a black colour. These two colours are not stable for the Hue transformation from the RGB colour. Therefore, after the colour detection, the results include some noise, which is indicated by the white dots. In addition, bees sometimes drop their pollen on the board. These pollen drops are also detected. The saturation channel can help to correct the detection.

After the HSV colour conversion, the white and black colours have a lower value of saturation, but the orange colour has a higher value of saturation. Figure 4.6(a) shows the saturation channel as an intensity image. It can be seen that the bee bodies include higher and lower saturation values. The orange part has a higher value, while the head and black parts of the body have a lower value. After testing 20 video images, the threshold for bee detection using the Saturation pixel $\mathrm{s}(\mathrm{x},\mathrm{y})$ was chosen as $\mathrm{s}(\mathrm{x},\mathrm{y}) > 0.12$. Figure 4.6 (b) indicates the blobs from saturation detection. The result includes some shadow regions because their values are the same as the bee's head saturation



(a) (b)

Figure 4.6 Example of the Saturation channel.

Note: (a) The Saturation channel shown as an intensity image. (b) The bee region detection using the Saturation channel.

The combination of the two channels produces the logical "and" calculation on each of the corresponding pixels. If the Hue detected image is $\mathrm{HD}(\mathrm{x},\mathrm{y})$, and the saturation detected image is $\mathrm{SD}(\mathrm{x},\mathrm{y})$, the combination is:

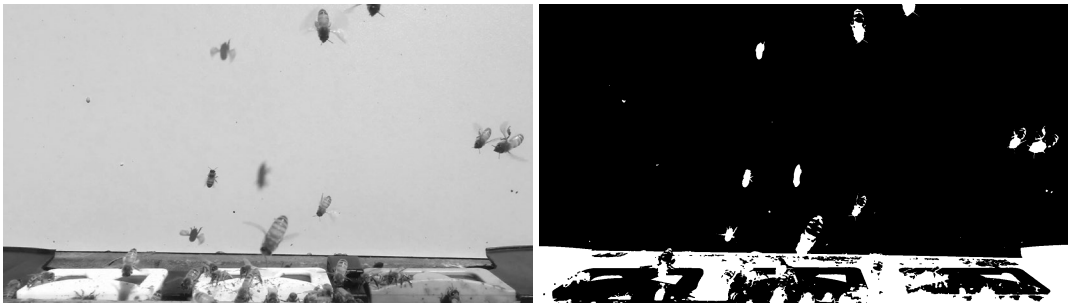
$$MOI = HD(x,y) \wedge SD(x,y) \qquad 4\text{-}1$$

where MOI is the binary image known as the Mask Orange Image. The "$\wedge$" is the logical and operation applied to the binary pixels of two detected images. After morphological filtering by image opening and filling holes for the MOI, Figure 4.7 is the final result of the orange colour detection. The shadows on the saturation detection are removed and the bee shapes are clearer.

Figure 4.7 The orange mask binary image after morphological calculation (MOI).

However, this method still loses some of the bees' heads. The orange dust of the pollen on the board can still be detected. While the bee's body has orange and black colours, only the orange colour is used to detect part of the bees' bodies. The black colour is the other feature of bees' bodies, but this colour is not stable in the Hue and Saturation channels. Therefore, black is only detected from the Value channel. If the Value pixel is $\mathrm{v}(\mathrm{x},\mathrm{y})$, the black pixels have $\mathrm{v}(\mathrm{x},\mathrm{y}) < 0.5$ from the experiment.

Figure 4.8(a) is the value channel which looks similar to a grey scale image. Figure 4.8(b) displays the Value detected result image (Mask of the Black Image, MBI). It can be seen that the black detection detects the bee heads correctly, although it also detects shadows. If the orange colour and the black colour detection is combined, the bee detection is better. The orange and black colour mask image is MOBI, calculated by



(a) (b)

Figure 4.8 Example of the Value channel and black colour detection.
Note: (a) The Value channel displayed as intensity image. (b) The black colour detection binary image (MBI) after the morphological filter.

$$MOBI = MOI \vee MBI \qquad 4\text{-}2$$

where the "$\vee$" is the logical or calculation applied to each binary pixel. The final bee colour detection image (MOBI) is shown in Figure 4.9 which has smoothed edges after image opening and filling holes. The bee blobs are clear and intact. However, the colour segmentation only detects the colour, so the shadows and other orange dust are also detected as bees. Therefore, this method of detection is not sufficient. The colour segmentation should be combined with foreground detection to remove other dust blobs.

Figure 4.9 The combination of the orange and black colour (MOBI)
Note: this is the result after the morphological filter.

All the example images above show the detection with a white colour background. However, the final aim of this research is to detect the pollen sac. Some bees bring white coloured pollen, which cannot be detected when there is a white background. Therefore, a colour background that is different from all of the pollen colours is required. The grass has a green colour, so green will not affect the bee's action and green coloured pollen has not been found in the experiment. The green coloured background with its HSV colour transform is shown in Figure 4.10. The Hue value of background is near constant in Figure 4.10 (b) because of the green background, whereas the Hue in Figure 4.5 (b) was not constant because of the white colour. Bee regions in this Hue channel are still dark, especially the orange coloured parts, but they are more obvious in Figure 4.10 (b) than in Figure 4.5 (b). The Saturation channel in Figure 4.10 (c) is much different from Figure 4.6 (a). The background has a higher saturation value, which is similar to that of bee bodies. Therefore, this channel may not detect a bee's body accurately. In addition, the black colour regions also have a higher level of saturation, such as in the shadows and bee's heads. Conversely, the wings reflection has a lower value because of the white colour. The Value channel

in Figure 4.10(c) looks similar to Figure 4.8(a). This channel is still useful for the black colour detection.

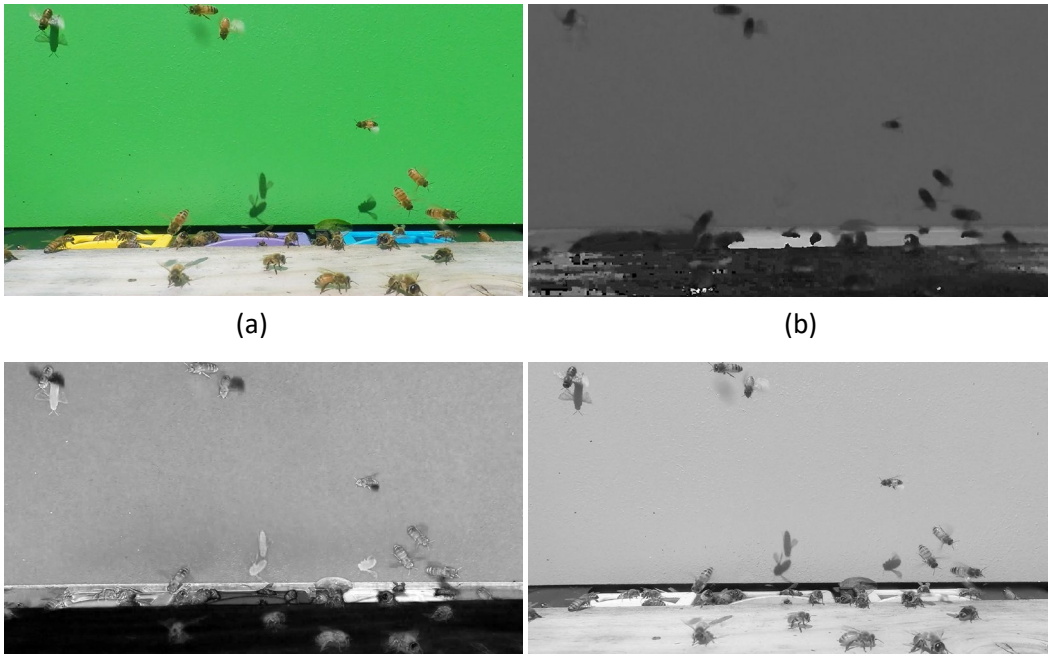


Figure 4.10 The Green background colour transform.

Note: (a) The original image. (b) The Hue channel shown as intensity. (c) The Saturation channel shown as intensity. (d) The Value channel shown as intensity.

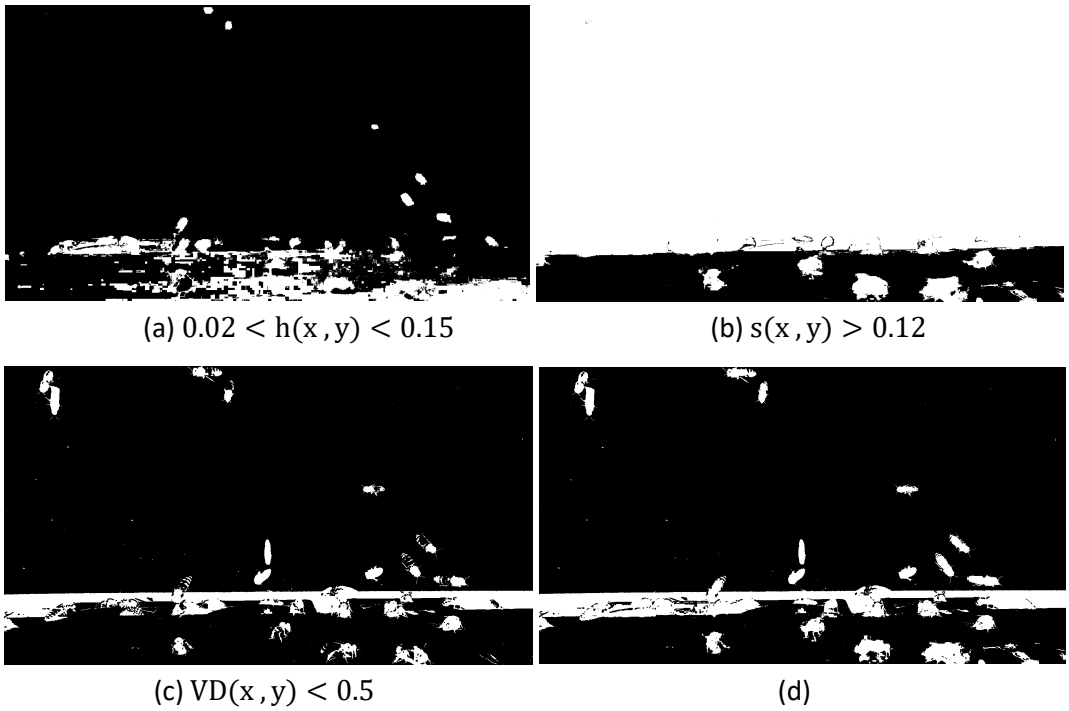


Figure 4.11 An example of the detection with same thresholds.

Note: (a) The Hue thresholding for orange colour. (b) The Saturation thresholding for orange colour. (c) The Value thresholding for black colour. (d) The final colour detection

The thresholds for the range and black colour were initially set the same as for the white background.

- Orange colour: $0.02 < \mathrm{h}(\mathrm{x},\mathrm{y}) < 0.15,\ \mathrm{s}(\mathrm{x},\mathrm{y}) > 0.12.$
- Black colour: $\mathrm{v}(\mathrm{x},\mathrm{y}) < 0.5$

An example result is displayed in Figure 4.11. It can be seen that the saturation threshold almost covers the whole image. It is therefore necessary to judge another threshold. As a result of detection, it can be seen there is no threshold which can separate bee regions from the background, because the main regions of bee bodies are similar to the background in the Saturation channel. In addition, the thresholds need to be chosen to fit most of the different videos. If the saturation difference between the background and the bee body is small, the detection is not reliable. Therefore, the Saturation channel is not employed for orange detection on the green background.

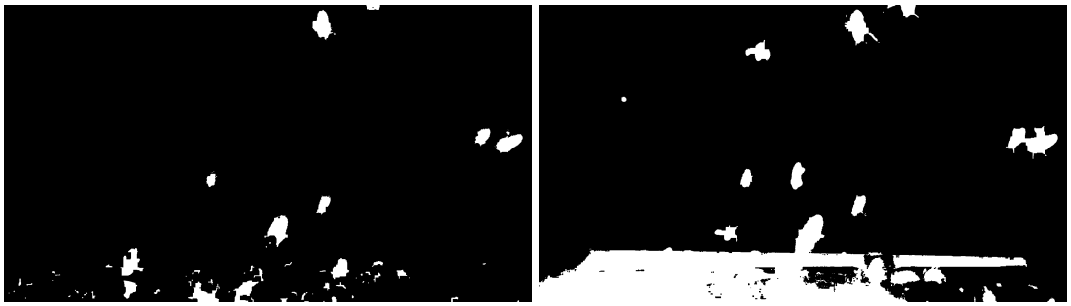
A typical result for the final detection with green background is shown in Figure 4.11(d). For each bee, most of the body is in the binary blob, but some of the tail parts are not. However, all the shadows are still detected as binary blobs. The green background cannot remove the shadows, because the black colour region can be part of a bee or a shadow.

4.2.3 The combination of foreground and colour detection

The foreground subtraction method (from section 4.2.1) outputs a binary image which masks the foreground pixels. This is the mask of the foreground image (MFgI). In addition, the colour segmentation detects objects with an orange colour. It produces another binary image which masks the orange colour (MOI). The black colour includes the shadows which has the same problem as in foreground detection, so initially it is not combined with the orange. At the beginning, the combination method uses the binary images which come from the two methods of foreground and orange colour detection. There are two further methods of combining these two binary images: logical "and" ($\wedge$) and logical "or" ($\vee$).

$$Combine1 = MOI \wedge MFgI \qquad 4\text{-}3$$

$$Combine2 = MOI \vee MFgI \qquad 4\text{-}4$$



(a) Combine1 (b) Combine2

Figure 4.12 The results of the two combined methods.
Note: (a) The result of the logical "and" of mask images of MOI and MFgI. (b) The result of the logical "or" of the mask images of MOI and MFgI.

Example results of these two methods are shown in Figure 4.12. Combine1 removes the shadows and orange noise of the background pollen, but only parts of the bee bodies are detected. Most of the bee heads are lost in the result. Combine2 covers the whole bee bodies, but the shadows

and orange background noise are also detected. The aim of the detection is to detect the whole bee bodies and remove the shadows and noise.

4.2.3.1 Bee detection model A

With this idea, a new method was created to solve the problem. The method is shown below and displayed in Figure 4.13:

$$DilatedImage(DI) = Dilate(MOI \wedge MFgI) \quad 4\text{-}5$$

$$MOjI = MOI \vee MFgI \quad 4\text{-}6$$

$$MBBI = DI \wedge MOjI \quad 4\text{-}7$$

The equations (4-5) – (4-7) above are referred to as detection model **A**, in which the logical calculations of "$\wedge$" and "$\vee$" are applied to corresponding binary pixels of two mask images. The morphological dilation in equation (4-5) produces a dilated image (DI) which is used to find the approximate position and area of the bee. Moreover, the DI can remove the shadows and orange background which are independent of the bee bodies. A "good" dilation makes the DI image cover whole bee bodies. The dilation operation causes the blobs to expand. This depends on the size of the original blobs on Figure 4.12(a). However, the image shows some bee blobs are big enough to be detected, while some blobs are too small, in particular, some blobs do not cover the bee heads. Experimentally, for detection model **A**, the dilation operation was chosen as a disk-shaped structuring element with a radius of 35 pixels.

The MOI and MFgI were combined to obtain the mask object image (MOjI in (4-6)) which is the same as Combine2. The final result was the mask bee blob image (MBBI) obtained from the combination of DI and MOjI. Moreover, all of the results of the equations were smoothed by image opening. The MBBI was smoothed by image opening and filling holes.

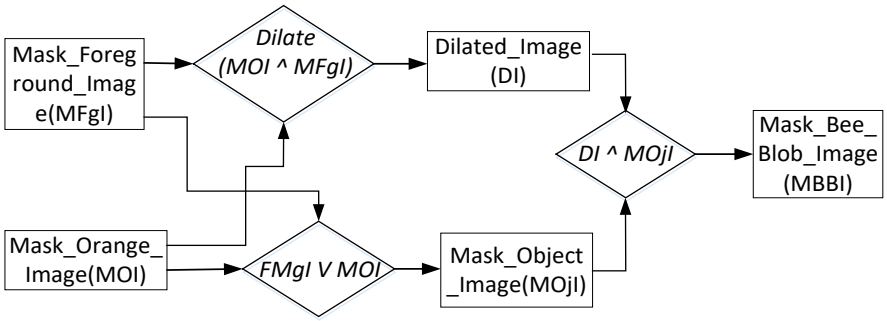


*Figure 4.13 The combination logic (model **A**) with binary images.*
Note: the MBBI is the final binary image.

Using to the above approach, the DI image and the MOjI image determine the region of bee bodies. The dilation radius was determined experimentally. A small dilation did not cover the whole bee body (especially the heads of bees), so the bee heads might be lost. Therefore, a large radius (35 pixels in this research) was chosen to detect more of the bee body regions, although it also detected shadows or orange background near the bee bodies. Figure 4.14 shows examples for two dilation radii (16 and 35 pixels) and the resulting binary bee blob images (MBBI). The advantage of this method is that it detected most of the bee bodies and removed individual shadows and the orange background. If the dilation radius was 16 pixels (in Figure 4.14(a)), the

result of the MBBI as shown in Figure 4.14(b) could be the loss of part of the head and tail. This shows that this method of detection is not reliable. This might have a worse result if employed in different videos. Figure 4.14(c) indicates the dilation image (DI) with a 35 pixel dilation radius. These blobs appear to be big, but this is useful in detecting the whole body and other details of bees (result in Figure 4.14(d)), especially parts of their wings and antennae.

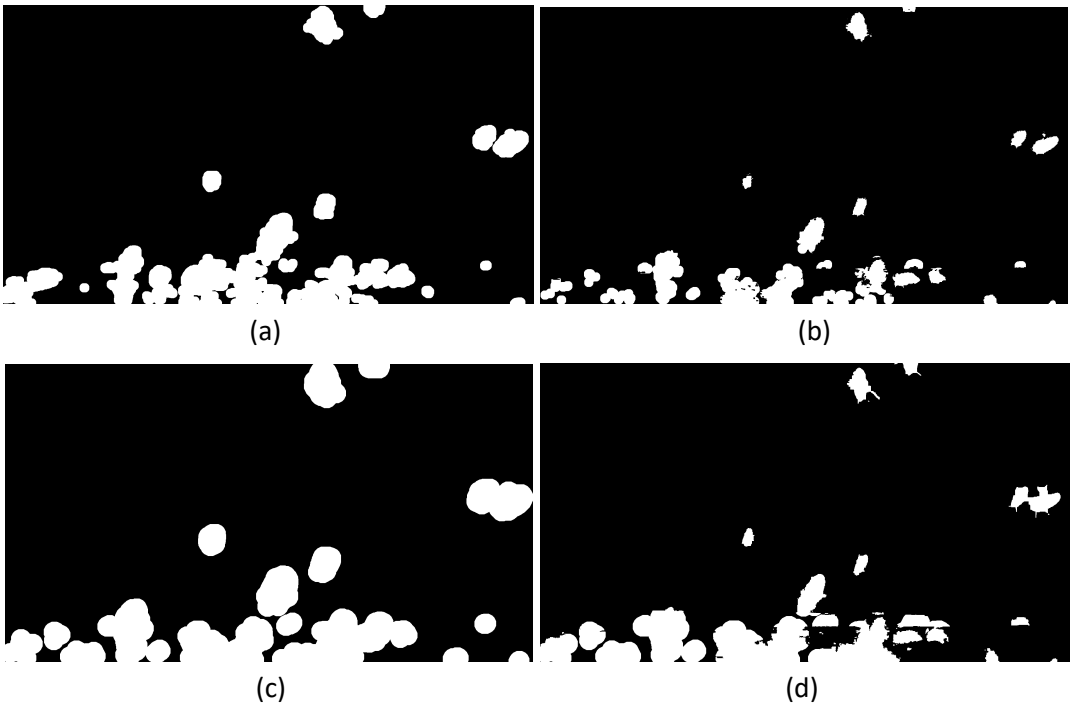


Figure 4.14 The result of different dilation and corresponding bee detection.

Note: (a) The DI image with dilation radius of 16 pixels. (b) The binary image of (MBBI) from (a). (c) The DI image with dilation radius of 35 pixels. (d) The final detection result with DI image of (c).

However, the disadvantage is shown in the two bees on the right side of the image. In Figure 4.14(b), these two bees were detected as two individual bees, but in Figure 4.14 (d) these two bees were detected as one blob (known as a merged blob), because the two regions linked to each other. This can generate a problem for bee tracking, which is discussed in section 4.3.3.

Black colour detection can be used to handle the disadvantage of 35 pixels dilation. According to MOBI in Figure 4.9, the colour detection almost covers the whole of the bee bodies and the two bees on the right side are detected as individual blobs. Therefore, if the equation (4-1) is changed to equation (4-2), the merged blob problem is removed. Moreover, the MBI in equation (4-2) can replace MFgI in equation (4-6). The MFgI is the foreground detection which is not necessary for the calculation of equation (4-6). In addition, the detection of the moving wings of MFgI may generate a similar problem to the disadvantage of the 35 pixels dilation.

4.2.3.2 Bee detection model B

The model has been rebuilt as shown below. This is the model **B**:

$$DilatedImage(DI) = Dilate(MOI \wedge MFgI) \quad \textit{4-8}$$

$$MOBI = MOI \vee MBI \quad \textit{4-9}$$

$$MBBI = DI \wedge MOBI \qquad 4\text{-}10$$

In this model (**B**), equation (4-8) is the same as equation (4-5). Equation (4-9) is derived from equation (4-2). The final result is the mask bee blob image (MBBI) obtained from the combination of DI and MOBI. Black colour detection (MBI) cannot be employed in the DI image in equation (4-8). For example, if the calculation is $\mathrm{Dilate}(\mathrm{MOI} \wedge \mathrm{MFgI} \wedge \mathrm{MBI})$, the orange and black colour does not share the same regions, so that the original blobs for dilation are too small. If the calculation is $\mathrm{Dilate}(\mathrm{MOI} \wedge \mathrm{MFgI} \vee \mathrm{MBI})$, the DI image includes shadows, so that the equation (4-10) may not remove shadows.

Figure 4.15 displays this approach as a diagram. As previously, all of the binary images (DI and MOBI) are smoothed by image opening, and the MBBI is smoothed by image opening and filling holes.

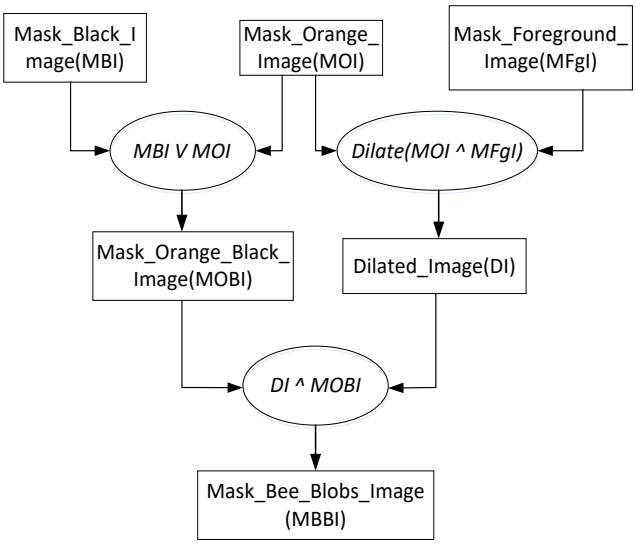
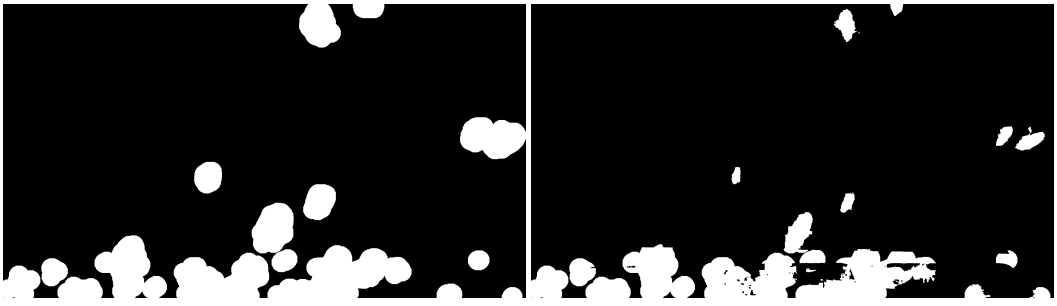


*Figure 4.15 The binary images logical calculation of model **B**.*
Note: the MBBI is the final binary image.

Figure 4.16 displays an example. Figure 4.16(a) shows the dilated image (DI). It can be seen that the dust blobs are removed. Figure 4.16(b) is the final result of bee detection.



(a) (b)

Figure 4.16 The detection result of model B.
Note: (a) The dilated image (DI). (b) The binary image of (MBBI) after morphological filter.

Compared with Figure 4.9, it removes the shadows and orange background. Moreover, compared with Figure 4.14(d), this binary image shows the main body shape more clearly. In particular, the

two bees on the right side of the image are detected individually rather than as one merged blob in Figure 4.14(d). This is useful for the bee tracking.

The blobs on the bottom of the binary image pose a difficulty. As can be seen in the original colour image in Figure 4.5(a), this part is close to the entrance, which includes the beehive wall, bees and coloured boxes. Bees in this area cannot be detected precisely. The blobs in this part of the video are removed, as shown in Figure 4.17. This is the final flying bee detection.

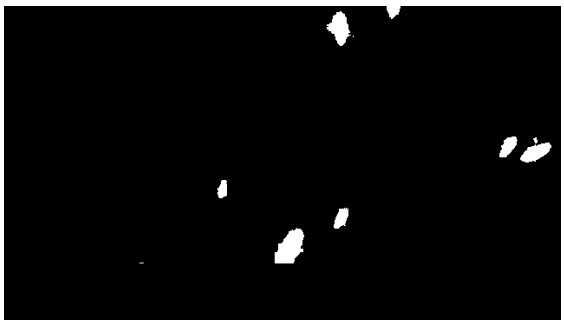


Figure 4.17 The final flying bee detection.

The detection result on the green background is similar to that on the white background. In the model **B** result shown in Figure 4.18, it can be seen that this latter model detects most of the bees and removes shadows. However, one bee is not detected (in the top left corner). In addition, the bee blobs in Figure 4.18(b) lose some detail, especially in their tails. After checking the colour detection of Figure 4.11, the main reason for this is due to the colour detection. The orange colour detection (Figure 4.11(a)) cannot detect the lost bee, because the bee is too dark. As a result, the DI image does not show this bee. Moreover, the two colour detection of MOBI loses some detail of the bee bodies in Figure 4.11(d). To solve this problem, it may be necessary to find new colour thresholds or incorporate the foreground detection of MFgI in the equation (4-8). This will be discussed in the next section dealing with pollen detection problems.

(a) (b)

Figure 4.18 An example of detection on the green background

4.2.4 Introduction to the problem of pollen detection

If the research is limited only to bee detection, the model **B** is good enough. However, the final aim is the detection of the pollen sacs on the bee's legs in flight. The pollen detection method is described in section 4.4, where the pollen detection relies on knowing the main body shapes of the bees. Moreover, the merged tracking investigated in section 4.3.3 also requires a clear main body shape. Therefore, it is important to detect more accurately the main body shape from the

blobs in Figure 4.18(b). Because the bees can have red, yellow or white coloured pollen sacs, a green rather than a white background is used for models **C** and **D**.

4.2.4.1 Bee detection model C

There are two ways to improve on the detection of bee shapes. The first approach to improve on the detection process of model **B** is to incorporate foreground detection into equation (4-8). This was introduced in section 4.2.3.2. Therefore, the equation now becomes:

$$MOBFI = MOI \vee MBI \vee MFgI \quad 4\text{-}11$$

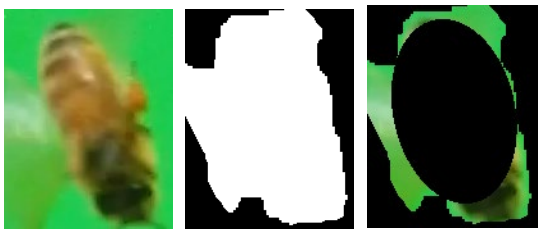
As a result, model **C** is:

$$DilatedImage(DI) = Dilate(MOI \wedge MFgI) \quad 4\text{-}12$$

$$MOBFI = MOI \vee MBI \vee MFgI \quad 4\text{-}13$$

$$MBBI = DI \wedge MOBFI \quad 4\text{-}14$$

With this change, model **C** replaces model **B**. Because the foreground subtraction now covers the movement part of the whole bee body, and the DI image can remove the shadows, the outcome is the detection of a bigger blob. However, this experiment leads to another problem. As the foreground subtraction is sensitive to wing movement, the blob is too fat for the detection of the pollen sac. An example is shown in Figure 4.19, where image (b) shows the detection result, with the blob covering a large part of the wings. Figure 4.19 (c) shows the main body which was masked in blob (b). The main body detection will be presented in section 5.1.1. It can be seen in Figure 4.19(c) that the main body is too fat, and the pollen sac is masked. This problem is solved in section 4.2.4.2.



(a) (b) (c)

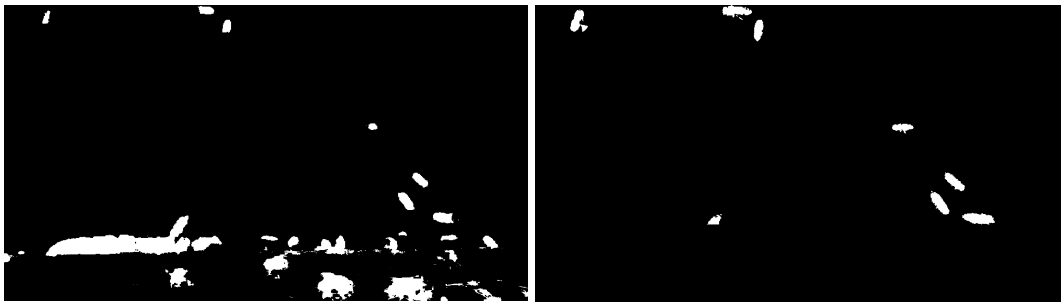
Figure 4.19 Bee main body removal using the detection

4.2.4.2 Bee detection model D

The second approach to improve on the detection process of model **B** is to change to new colour thresholds. The threshold range for orange colour detection on a green background is $0.02 < \mathrm{h}(\mathrm{x},\mathrm{y}) < 0.15$. It can be seen that orange detection loses some regions in Figure 4.11 (a), especially in some bees with a dark body. In addition, some pollen sacs have a yellow or white colour. Therefore, these two colours should be considered alongside orange, so that the detection of the bee shape will include both main body and pollen sacs.

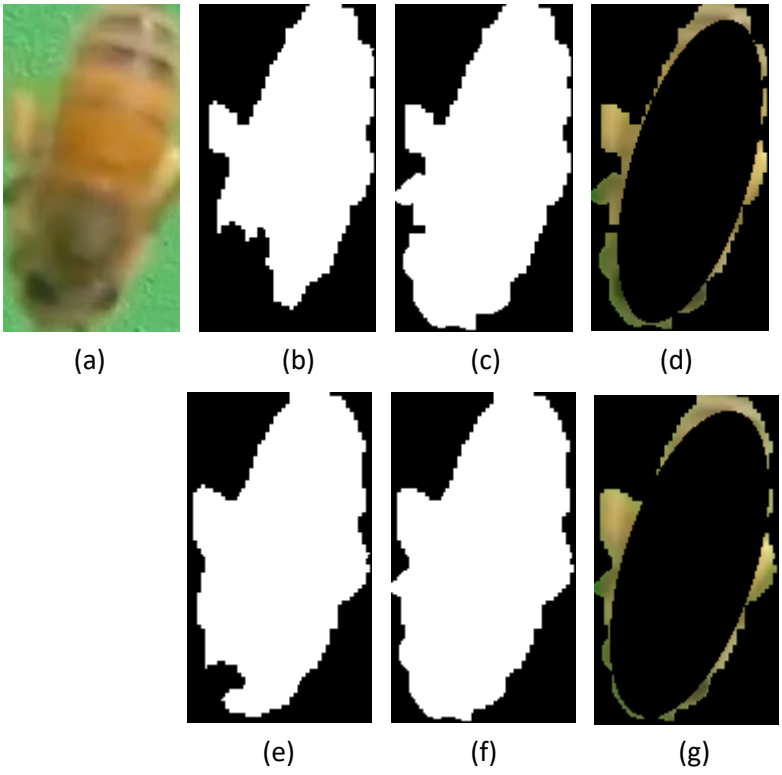
As the yellow colour, with a hue range of $0.15 < \mathrm{h}(\mathrm{x},\mathrm{y}) < 0.19$, is near to the orange colour detection in the Hue channel, these two colours can be combined readily within one threshold range, which now becomes $0.02 < \mathrm{h}(\mathrm{x},\mathrm{y}) < 0.19$.

A bee detection result with these colour thresholds is shown at Figure 4.20. The orange and yellow detection in Figure 4.20(a) covers more of the bee bodies. In particular, in the case of the bee in the top left corner, part of its body is now detected, whereas no detection was possible using model **B**. In the final detection result, as shown in Figure 4.20(b), the whole body is now detected.



(a) (b)

Figure 4.20 The bee detection result with new orange and yellow thresholds.
Note: (a) The orange and yellow colour detection result. (b) The final bee detection result.



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

(e) (f) (g)

Figure 4.21 Comparison of detection using the old and new thresholds.
Note: (a) The original image. (b) The old threshold with only orange colour detection. (c) The final bee detection with old threshold. (d) The main body masking from blob (c). (e) The new threshold with orange and yellow colour detection. (f) The final bee detection with new thresholds. (g) The main body masking from blob (f).

Another result that is related to pollen detection is shown in Figure 4.21. This figure shows one single bee detection with the old thresholds (only orange colour) and new thresholds (orange and yellow). Figure 4.21(b) shows the old colour detection, and image (c) illustrates the final result of bee detection with only orange detection. Both results were obtained using the thresholds of model **B**. Figure 4.21(e) and (f) show the new threshold detection results (with orange and yellow).

Image (e) detects more pollen and more of the bee head, compared with image (b). The blob shape on image (e) is more similar to the bee shape than image (c). Figure 4.21(d) and (g) show the main body mask from blobs (c) and (f) respectively. From the overall result, it can be seen that the image (g) detects the main body closer to its actual shape.

The white colour has a lower value of Saturation and a higher value of the Value channel. Because the background colour is green, white pollen sometimes reflects the green colour. In addition, white pollen can reflect the orange colour from the bee body. Therefore, a range encompassing orange and green is chosen in the Hue channel to detect the white colour of pollen. Experimentally, the white colour thresholds were chosen to be:

$$(\mathrm{v}(\mathrm{x},\mathrm{y}) > 0.5, \mathrm{s}(\mathrm{x},\mathrm{y}) < 0.4, \mathrm{h}(\mathrm{x},\mathrm{y}) < 0.23).$$

The threshold for the Hue channel filters out the transparent part of bee wings. The new model **D** resulting from the addition of the white colour is:

$$MOWI = MOI \vee MWI \quad \textit{4-15}$$

$$DI = Dilate(MOWI \wedge MFgI) \quad \textit{4-16}$$

$$MBCI = MOWI \vee MBI \quad \textit{4-17}$$

$$MBBI = DI \wedge MBCI \quad \textit{4-18}$$

In the equation of (4-15), MWI is the mask white colour image. This image is combined with image MOI to produce the mask orange and white image (MOWI). The dilation image (DI) comes from the orange and white colours and foreground. The mask bee colour image (MBCI) in equation (4-17) includes orange, white and black colours. The final outcome of the model **D** is the MBBI image. The diagram of model **D** is shown in Figure 4.22.

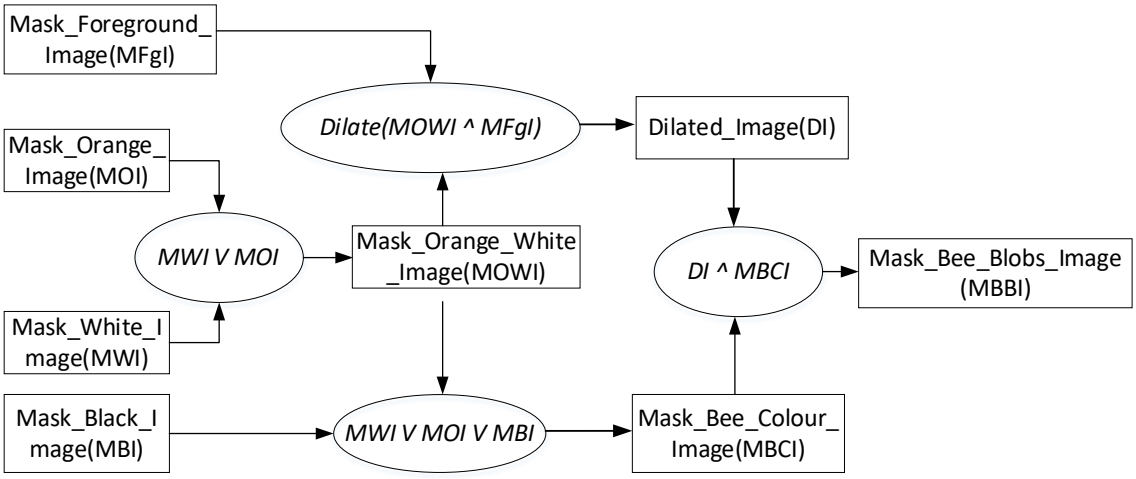


*Figure 4.22 The bee detection model **D** with orange, black, white colour and foreground detection.*

Figure 4.22 shows the final bee detection model. The mask image of MFgI comes from the foreground subtraction. The changing pixels have value "1", the unchanging pixels have value "0". The mask image of MOI is from the orange colour detection, where each pixel ($p_{MOI}(x,y)$) is calculated as follows:

$$p_{MOI}(x,y) = \begin{cases} 1, & 0.02 < h(x,y) < 0.19 \\ 0, & otherwise \end{cases} \quad \textit{4-19}$$

The mask image of MWI comes from the white colour detection, where each pixel ($p_{\mathrm{MWI}}(x,y)$) value is:

$$p_{MWI}(x,y) = \begin{cases} 1, & v(x,y) > 0.5 \cap s(x,y) < 0.4 \cap h(x,y) < 0.23 \\ 0, & otherwise \end{cases} \quad \textit{4-20}$$

The mask image of black colour (MBI) is from the black colour detection, and the pixel ($p_{MBI}(x,y)$) value is calculated using:

$$p_{MBI}(x,y) = \begin{cases} 1, & v(x,y) < 0.5 \\ 0, & otherwise \end{cases} \quad \textit{4-21}$$

All of the four binary images are combined following the model **D** in Figure 4.22. Figure 4.23 illustrates the improvement of the bee detection. Figure 4.23(b) shows the model **B** result. It can be seen that the blob excludes most of the white pollen. Conversely, the model **D** result (shown in Figure 4.23(d)) includes most of the pollen sac. Figure 4.23(b) and (d) also used morphological image closing, morphological image opening and filling holes.

*Figure 4.23 Comparison of the bee detection. Note: (a) Bee image from the old model. (b) The result blob image of model **B**. (c) Bee image from the new model. (d) The result blob image of new model.*

Figure 4.24 displays the detection result from model **D**. Compared with Figure 4.19, Figure 4.24(b) gives a clear main body shape. Figure 4.24 (c) indicates the main body detection fitting the actual shape of the bee, with the pollen being shown next to the bee body.

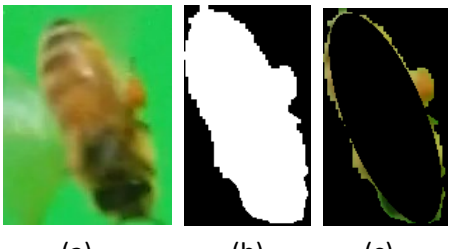


*Figure 4.24 The main body detection after bee detection of model **D***

So far, there are four bee detection models, which are labelled **A**, **B**, **C** and **D**. Table 4-1 summaries the characteristics of these four models. Models **A** and **B** are good enough for bee detection. Model **B** is better for detecting bees' heads than model **A**. However, considering the next step of pollen sac detection requiring bigger bee blob and clearer bee shape detection, models **C** and **D** have been created. Model **C** produces a bigger bee blob, but the blob is too big to help with pollen sac detection. Model **D** is more suitable for improving pollen sac detection, although model **D** can only be applied on the green background. The different of pollen sac

detection between model **B** and model **D** will be shown in section 5.1.2. Bee detection model **D** is finally chosen to be used for bee detection.

Table 4-1 The summary of characteristic of 4 bee detection models

Bee detection models	Summary of characteristic
Model **A**	Combining orange colour detection and foreground detection. The logic is in section 4.2.3.1.
Model **B**	Adding black colour detection into model **A**. This model detects bees' heads clearly. The logic is shown in section 4.2.3.2.
Model **C**	Changing of the logic to put foreground detection with logic "or" into orange and black colour detection. The logic is shown in section 4.2.4.1. This model is built for considering the next step of pollen sac detection requiring bigger bee blob. However, this model produces bee blob too big to help pollen sac detection.
Model **D**	Base on model **B**, adding white and yellow colour for bee detection. The logic is shown in this section. Model **B** lost some pollen sacs which have white and light-yellow colour. This model adds these colours, which can help to improve pollen sac detection, although it is only suitable for the green background, but not white background.

4.2.5 Discussion of the natural background

The natural background is more complicated than the single colour background. The natural background is dependent on the environment of the beehive. In this section, the detection method is attempted on a complex background. One example of the use of a complex background is shown in Figure 4.25. In this figure, the background colour includes green (grass), yellow (withered grass) black (earth) and white (stone). Using the detection model **D** which was introduced in the previous section does not work well in this situation. This is because if a bee flies past the orange, white and black background colours, the detection cannot produce a clear bee shape. Therefore, white and black colours cannot be used for bee detection in this background. The best model for this situation is model **A** which only uses the orange colour and foreground (motion) detection.

Figure 4.25 A natural complex background for the detection model trial.

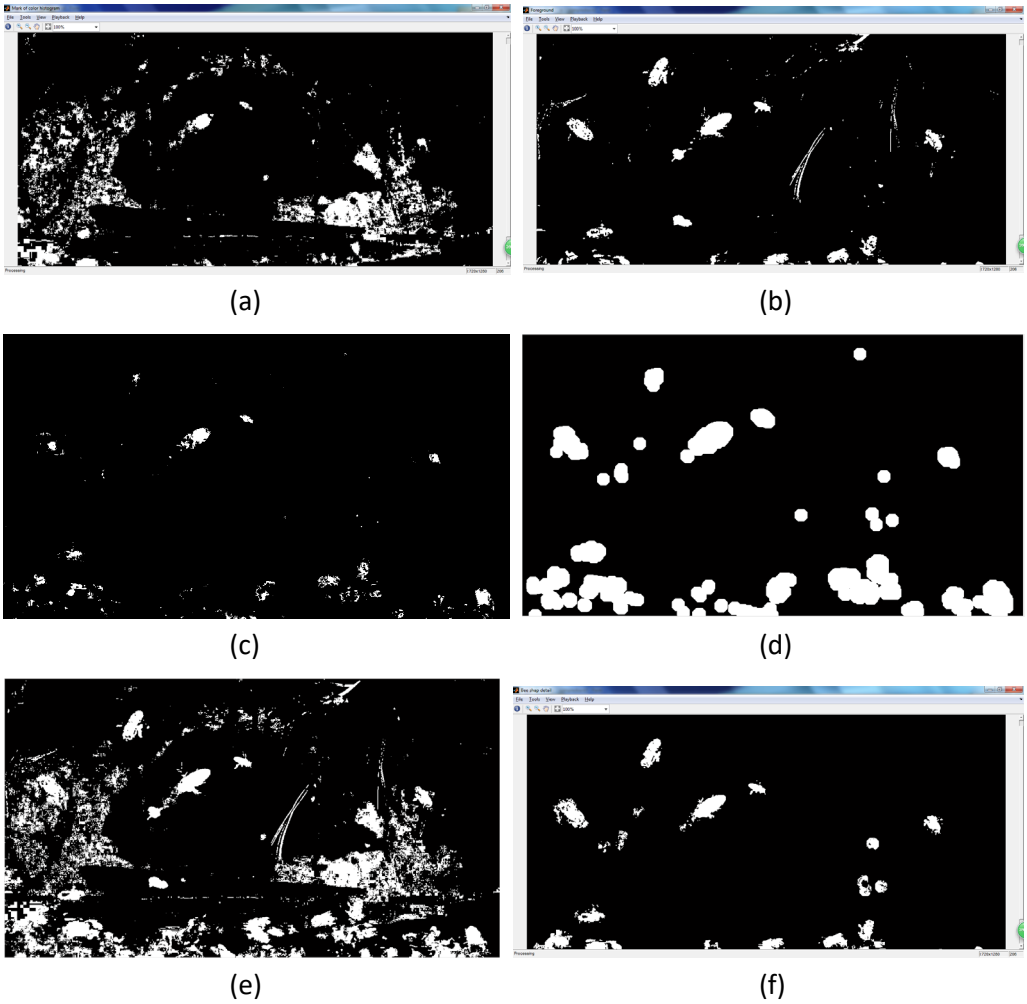


Figure 4.26 The bee detection trial on a natural complex background.

Note: (a) The orange colour detection. (b) The foreground detection. (c) The result from $MOI \wedge MFgI$. (d) The result of equation (4-16). (e) The result of equation (4-17). (f) The result of MBBI which is from equation (4-18).

Figure 4.26 illustrates the detection progress for model **A**. Figure 4.26 (a) shows the orange colour detection. Because the withered grass colour is similar to that of the bee body, most of the bees are not detected clearly. However, the foreground detection (Figure 4.26 (b)) is better than colour detection in the case of withered grass, because while most of the withered grass is still, the bees in contrast are active. However, bee detection is limited in the case of some of the green grass and their shadows moving in the wind, because the bees are also moving. Figure 4.26 (c), being the result of the logic "and" calculation, does not detect many bee bodies. On the other hand, the dilation image (Figure 4.26 (d)) does give correct bee positions and regions, although there is some confusion because it also detects small blobs coming from the moving of some of the withered grass or moving shadows on the yellow background. Figure 4.26 (e) is the result of MOjI image. Although some bee shapes can be found on this image, it is hard to detect the bees flying in the area of the withered grass. The final detection result is shown in Figure 4.26 (f). Here, most of bees are detected, but three out of the six bees lose their shapes. The model also detects an additional six blobs which are not bees. These results show that, when compared with the bee detection on a single colour background which detects all the bees, the natural background outcome detects only half of the bees and some non-bee blobs are also detected.

This natural background investigation indicates that the bee detection model needs a single colour background or a background which does not include the bees' colour. The natural background always changes with the environment of the beehive, which leads to the bee detection being unreliable. Therefore, the single colour background is necessary for the bee detection. In this research, the single green background is chosen as the best method to simplify the background.

After bee detection, the next step is to analyse the binary image bee blobs. The blob analysis uses image moments to calculate blob parameters, such as position, length, width and orientation. In addition, it draws a bounding box around each bee blob to mark the detection. Blob analysis was introduced in section 3.7. The blob position is the candidate position of the bee, which is very close to the actual position of the bee. It can be used to track bees on the video. However, merged blobs caused by bees crossing each other in the video frames is a problem, which will be discussed in the next section. The blob length, width and orientation are used to detect the pollen and to improve the tracking of bees in merged blobs.

4.3 Bee Tracking

The video has a 50Hz frame rate and the camera is positioned to provide a close up view of the bees. There are two difficulties in tracking the bees: (1) the bees can fly anywhere in the video and (2) most of the bees have the same features. Because of this, the only way to track individual bees is from their previous positions. Initially, it was assumed that the shortest distance between blobs in successive frames indicated the same bee, but when bees fly close to each other, this method fails. It is therefore necessary to calculate a predicted position for each bee to correct this.

In addition, the bee detection may detect one merged blob rather than individual blobs. Although the merged blob detection can be reduced by using a more accurate detection method (as outlined in section 4.2.3, merged blobs still occur when the bees are very close to each other or are flying over each other in the video image.

4.3.1 Prediction method

In the prediction tracking model, it is assumed that each bee's velocity is constant over three successive frames. The bee position in the third frame can be predicted from its position in the two previous frames. If *k* is the frame number in the frame flow of the video, the detected bee position in frame *k* is $D_k = [x_k, y_k]$, where (x_k, y_k) is the centre of the bee blob and the bounding box. The tracking model is:

$$P_k = A * [D_{k-1}, D_{k-2}]^T \qquad 4\text{-}22$$

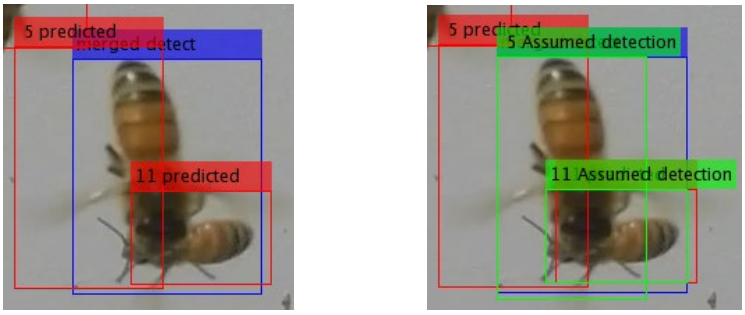
Where:

$$A = \begin{bmatrix} 2 & 0 & -1 & 0 \\ 0 & 2 & 0 & -1 \end{bmatrix} \qquad 4\text{-}23$$

Here, $P_k = [x_{pk}, y_{pk}]^T$ is the predicted position of the bee in frame *k* and $[D_{k-1}, D_{k-2}]^T$ is the detected positions within the two previous frames.

When a bee first appears in the video, there is no previous position to work from, so the predicted position for the next frame is taken to be the current frame detection position. In the following

frame, bees are tracked using the Hungarian assignment method (Aizen and Harder, 2009). This method assigns predictions to detections in an optimal way, to minimize the sum of the distances between the assigned predictions and detections. As a constraint, a prediction will not be assigned to a detection when the distance is more than 80 pixels. This will usually result in some unassigned predictions and detections. When a detection is unassigned, it is probably a bee appearing for the first time. In addition, unassigned predictions can be caused by a bee disappearing or bees moving across each other in the video.



(a) (b)

Figure 4.27 The merged bees situation and solution

Because it is 2D video, when two bees pass across each other in the image, the detection process only finds one big merged blob rather than two separate bee blobs. In this case, there are two predicted positions and one detected position. One prediction is assigned to the detection, but then there is an unassigned prediction as well. If the unassigned prediction bounding box overlaps a detected bounding box, this is taken to be a merged situation. Figure 4.27 (a) displays the merged situation. One prediction is assigned with this merged detection and the other prediction is unassigned, but the unassigned prediction bounding box overlaps the detected merged bounding box.

Once the merging has been identified, the next step is to find the merged bees positions. In Figure 4.27 (a), it can be seen that if the prediction bounding box centre is used to calculate the next prediction, it is not accurate, because the actual positions of the bees are not at the centre of the prediction bounding box. However, in the image, if the prediction bounding boxes are adjusted to align with the boundary of the merged detected bounding box, the adjusted bounding boxes cover the bees more accurately. Then, the centre of the prediction bounding boxes are close to the actual bees' positions. The assumed detections in Figure 4.27 (b) are the result of using the adjusted bounding boxes. The centres of the assumed detection bounding boxes are the positions calculated from this type of detection. Following this, these two assumed detections are used to calculate the next predictions. This method is used until the two bees separate.

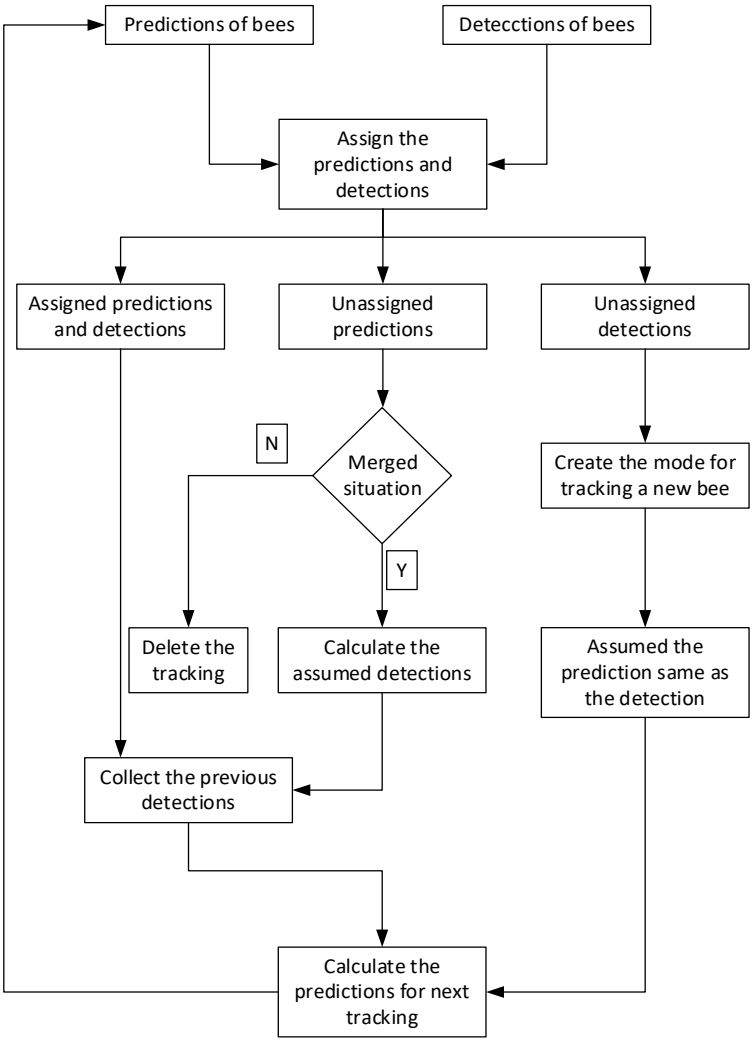


Figure 4.28 The tracking model of prediction method

Figure 4.28 shows the whole tracking model process. When a bee appears for the first time, there is an unassigned detection for it. As a result, a tracking model is built for tracking the bee. It is assumed that the predicted position for the second appearance will be the same as the first detection. Therefore, this prediction may not be accurate, but the next prediction based on the two previous detections should be more accurate. When a bee disappears from the video, there is an unassigned prediction, but it is not a merged situation. In this case, the tracking model is deleted, because this bee cannot be tracked any more.

However, this model has disadvantages. If bees stay merged for a long time, they may change their flight direction, so the predictions become less accurate. Furthermore, there could be more than two bees merging, making it more difficult to track them correctly using this method. Finally, the model assumes that the bee's velocity is constant over three successive frames, but the bee velocity often changes. Therefore, the model may lose the bee when the bee velocity changes quickly.

4.3.2 Kalman filter

4.3.2.1 Principle of Kalman filter

The video has a 50Hz frame rate and its view is very close to the bees. Therefore, a difficulty is that the bees can quickly fly anywhere from one frame to the next. In this case, the Kalman filter for object tracking (Shantaiya et al., 2015) may not work well, because the bees are changing positions and velocities. There is no steady variable (velocity or acceleration) for the Kalman filter. However, taking another view, the Kalman filter can combine the prediction and measurement to produce a correction for the prediction of the next frame. If the position of a bee is the state of the Kalman filter, in this research, the measurement of bee detection is more reliable than the prediction when tracking a single bee, so the actual position is close to the detected position (measurement). The corrected position should be close to the detected position of the bee. Conversely, in the merged bee situation, the detection is not as reliable as single bee tracking, so the corrected position calculation is more complicated. This method requires the estimation of the covariance matrices of the measurement and the prediction.

To apply the Kalman filter formulae, the state of a bee is its position (x, y), and the prediction of the Kalman filter tracking model in frame *k* is:

$$x_k^- = 2x_{k-1}^+ - x_{k-2}^+ \quad \textit{4-24}$$

$$y_k^- = 2y_{k-1}^+ - y_{k-2}^+ \quad \textit{4-25}$$

If the prediction of the Kalman filter is:

$$s_k^- = A[{s_{k-1}^+}^T, {s_{k-2}^+}^T]^T \quad \textit{4-26}$$

then the predicted state vector is:

$$s_k^- = [x_k^-, y_k^-]^T \quad \textit{4-27}$$

The corrected state vector in the frame *k* is:

$$s_k^+ = [x_k^+, y_k^+]^T \quad \textit{4-28}$$

The transition matrix is defined as:

$$A = \begin{bmatrix} 2 & 0 & -1 & 0 \\ 0 & 2 & 0 & -1 \end{bmatrix} \quad \textit{4-29}$$

The measurement model of the Kalman filter is:

$$m_k = Hs_k \quad \textit{4-30}$$

The actual state vector is:

$$s_k = [x_k, y_k]^T \quad \textit{4-31}$$

The measurement vector is:

$$m_k = [x_k, y_k]^T \quad \textit{4-32}$$

The measurement transition matrix is:

$$H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \textit{4-33}$$

Apart from the states, the Kalman filter also includes the Kalman gain K_k for the Kalman correction calculation. The Kalman gain is calculated from the prediction covariance matrix P_k and the measurement covariance matrix R_k. P_k is calculated from the transition matrix *A*, a covariance matrix C_0 and the correction covariance matrices from the two previous frames *k-1* and *k-2*, which are combined to form the matrix $\mathrm{O_{k-1}}$.

$$O_{k-1} = \begin{bmatrix} Q_{k-1} & \mathbf{0} \\ \mathbf{0} & Q_{k-2} \end{bmatrix} \quad \textit{4-34}$$

In the beginning, this $\mathrm{O_{k-1}}$ is the zero matrix. The covariance matrix of the correction $\mathrm{Q_k}$ is calculated from the Kalman gain K_k and prediction covariance matrix P_k. Each element of Q_k is shown below:

$$Q_k = \begin{bmatrix} q_{11k} & 0 \\ 0 & q_{22k} \end{bmatrix} \quad \textit{4-35}$$

In summary, at frame number *k*, the discrete Kalman filter estimation for object tracking is:

Prediction:

$$s_k^- = A[{s_{k-1}^+}^T, {s_{k-2}^+}^T]^T \quad \textit{4-36}$$

$$P_k = AO_{k-1}A^T + C_0 \quad \textit{4-37}$$

Measurement:

$$m_k = Hs_k \quad \textit{4-38}$$

Correction:

$$K_k = P_k(P_k + R_k)^{-1} \quad \textit{4-39}$$

$$s_k^+ = s_k^- + K_k(m_k - s_k^-) \quad \textit{4-40}$$

$$Q_k = (I - K_k)P_k \quad \textit{4-41}$$

The detailed algorithm of the Kalman filter equations was introduced in section 3.5. Equations (4-36) and (4-37) represent the prediction. s_k^- is the predicted state and P_k is the covariance matrix of the prediction s_k^-. Equation (4-38) is the measurement calculation. s_k is the actual state that the detection is measuring (the x and y coordinates of the bee). The matrix H is the measurement model matrix, which is the identity matrix in this case. m_k is the measurement result. Equations (4-39) to (4-41) show the correction calculation. s_k^+ is the corrected state after combining the m_k with s_k^-. R_k is the covariance matrix of the measurement m_k. $\mathrm{Q_k}$ is the covariance matrix of the corrected state $\mathrm{s_k^+}$. The correction is controlled by the Kalman gain K_k. In this paper, for individual bees, the measurement (m_k) (bee detection) is more reliable than the prediction (s_k^-), so the covariance P_k is greater than R_k. Then the estimated state $\mathrm{s_k^+}$ is closer to the measurement than to the prediction.

In this research, C_0 is the variance due to the fact that the bee velocity is not constant. It adds to the noise of the prediction. C_0 is not only added once at the beginning of tracking, it is added in each Kalman filter calculation.

The covariance matrices of C_0 and R_k need to be calculated, but this depends on the Kalman filter application. The matrices are calculated from the tracking noise which comes from two main sources.

First is the bee movement. Bees change velocity between frames, so the prediction is different from the actual position. This error is treated as tracking noise, so that the Kalman filter can allow for changes in the bee velocity.

Second is the detection error. The detection is based on motion detection and colour segmentation. This does not detect the bee positions perfectly. As the position of the bee's wings and body is constantly changing, this will affect the detection. In addition, the merged detection situation can only detect the large blob of the merged bees. The centre of the merged blob is not the actual location of these bees. All of these detection errors are treated as noise, so that the Kalman filter can incorporate them to calculate the covariance matrix of the measurement.

A Kalman filter is used to track each bee appearing in the frame, but the merged situation is different. The detection only produces one bee position, which is the centre of the merged blob, but there is more than one prediction associated with the blob. Taking two bees merging as an example, the detected position is given by two Kalman filters, one for each bee, but this position is not necessarilly as close to the actual positions of the two bees, as for the measured position of a single bee. The two predicted positions of two bees are calculated from two previous frames of the video. In the next (second) merged frame, the predicted positions may be even further from the actual positions of two bees. Therefore, the merged situation has greater error than single bee tracking.

Considering the tracking noise, the covariance values are estimated from the differences (errors) between the system calculation of the bee positions in the model, and the actual bee positions calculated manually. The measurement error is calculated as:

$$e_m = [e_{mx}, e_{my}] = [x_d - x_a, y_d - y_a] \qquad \textit{4-42}$$

where e_m is the measurement error, $[x_d, y_d]$ is the detected position and $[x_a, y_a]$ is the actual position. In a similar way, the prediction error is calculated from the difference between the system prediction and the actual position.

$$e_p = [e_{px}, e_{py}] = \left[x_p - x_a, y_p - y_a\right] \qquad \textit{4-43}$$

where e_p is the prediction error and $[x_p, y_p]$ is the predicted position.

It is assumed the x and y values are independent of each other, so both the covariance matrices are diagonal. If the expected value of the measurement error is **0**, the variance of the measurement is:

$$v_m = [v_{mx}, v_{my}] = E[(e_m - E(e_m))^2] = E[{e_m}^2] = [E[{e_{mx}}^2], E[{e_{my}}^2]] \qquad \textit{4-44}$$

Similarly, the variance of the prediction is:

$$v_p = [v_{px}, v_{py}] = E[(e_p - E(e_p))^2] = E[{e_p}^2] = [E[{e_{px}}^2], E[{e_{py}}^2]] \qquad \textit{4-45}$$

The covariance matrix of measurement is:

$$R_k = \begin{bmatrix} r_{xk} & 0 \\ 0 & r_{yk} \end{bmatrix}, \quad 4\text{-}46$$

where r_{xk} and r_{yk} are the measurement error variances for the x and y coordinates.

The covariance matrix of the prediction is:

$$C_0 = \begin{bmatrix} c_1 & 0 \\ 0 & c_2 \end{bmatrix}, \quad 4\text{-}47$$

where c_1 and c_2 are the prediction error variances for the x and y coordinates.

For single bee tracking, 42 video frames have been used to estimate the measurement errors and calculate the variance. There were a total of 199 position samples from individual bees on these frames. The actual positions of bees were estimated manually. The measurement positions come from the bee detection process. The measurement errors and variances are calculated by using equations (4-42) and (4-44) respectively. The error distribution is shown in Figure 4.29. It can be seen that the errors are close to 0, with most of the errors concentrated near x = -1.2 and y = 0 pixels. The bee position detection is reliable and the single bee images are satisfactory for calculating the measurement variance.

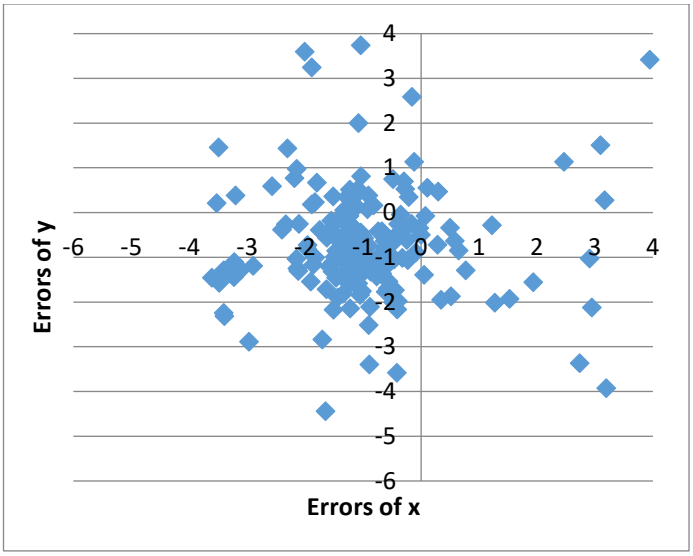


Figure 4.29 The error distribution of measurement

The covariance matrix of the measurement is:

$$R_k = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} \quad 4\text{-}48$$

42 video frames have been used to estimate prediction errors and calculate their variance. There are a total of 142 position samples from individual bees on these frames. In the first and second frames, the tracking for each bee does not have a predicted position. The actual positions of bees are estimated manually. The predicted positions come from the Kalman filter prediction. The prediction errors and variances are calculated using equations (4-43) and (4-45) respectively. The error distribution is shown in Figure 4.30. The sample of distribution is approximately symmetrical around the origin (0, 0).

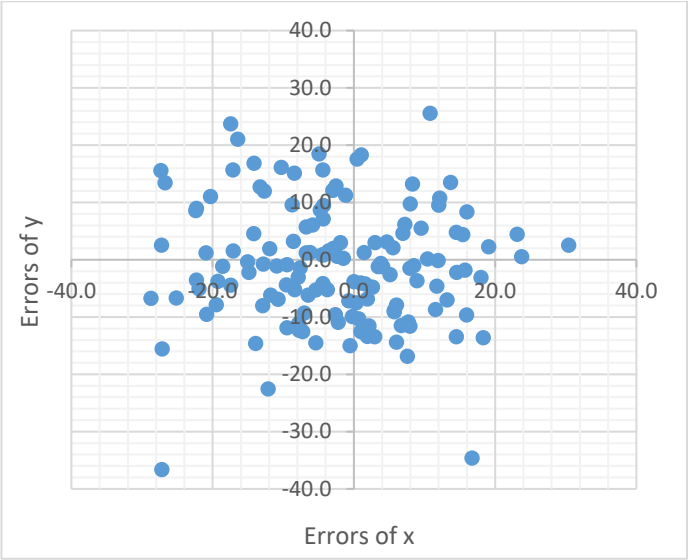


Figure 4.30 The error distribution of prediction

The covariance matrix of the prediction is approximately:

$$C_0 = \begin{bmatrix} 158 & 0 \\ 0 & 151 \end{bmatrix} \qquad 4\text{-}49$$

The off diagonal elements are small enough to ignore.

Figure 4.31 The measurement error distribution of merged detection

The two covariance matrices are expected to be constant during single bee tracking. However, when bees merge in the frame images, the covariance matrices of prediction and measurement in the merged tracking are different from the single tracking case. 48 video frames (which included merged situations) were utilised to estimate the measurement errors and calculate the variance. There were a total of 91 position samples of merged bees in these frames. The actual positions of the bees were calculated manually. The detected (measurement) positions are the merged blobs' positions (centre coordinates). The measurement errors and variance are calculated using equations (4-42) and (4-44) respectively. The distribution of these measurement errors is displayed in Figure 4.31.

The measurement covariance matrix for merged bee tracking is:

$$R_k = \begin{bmatrix} 1009 & 0 \\ 0 & 932 \end{bmatrix} \quad \textit{4-50}$$

48 video frames were used to estimate the prediction errors and calculate the variance for merged bee tracking. There were a total of 91 position samples of merged bees in these frames. The actual positions of bees were estimated manually. The predicted positions are from the Kalman filter prediction of merged bee tracking. The prediction errors are calculated using equation (4-43). The error distribution is shown in Figure 4.32.

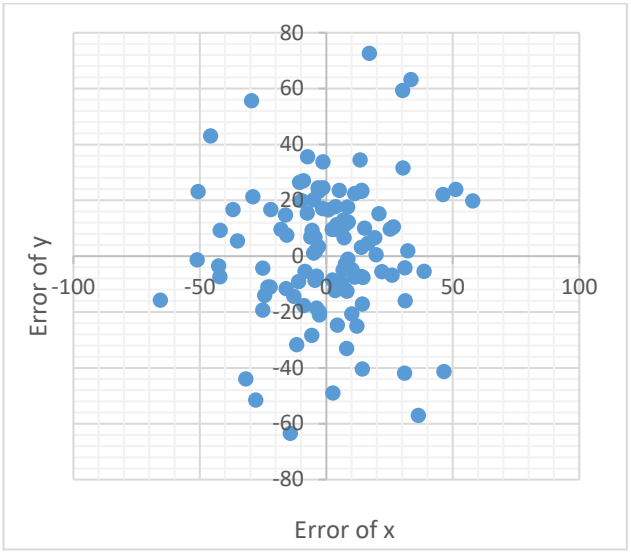


Figure 4.32 The prediction error distribution of merged detection

The prediction covariance matrix for merged bee tracking is:

$$C_0 = \begin{bmatrix} 503 & 0 \\ 0 & 578 \end{bmatrix} \quad \textit{4-51}$$

In the merged tracking, R_k is greater than C_0. From equations (4-37) to (4-41), when merging occurs, the Kalman filter tends towards the prediction rather than the detection. These two covariance matrices are also assumed to be constant for merged bee tracking.

4.3.2.2 The tracking model of the Kalman filter

The Kalman filter tracking model is introduced in this section. In the first frame of a bee appearing, there is no previous Kalman filter tracking, so the predicted position for the next frame is the same as the detected position in the first frame. From the second frame, bees are tracked using the Hungarian assignment method (Nandashri and Smitha, 2015c). This produces some unassigned predictions and detections. When a detection is unassigned, it is usually a bee appearing for the first time. In addition, unassigned predictions can be caused by a bee disappearing or bees merging in the video.

Figure 4.33 shows the whole tracking model of the Kalman filter. When a bee just appears for the first time, a Kalman filter model is created for tracking it. The prediction is the same as the detection. After the creation of the Kalman filter, predictions and detections are assigned for the Kalman filter tracking. The correction produces the next prediction using the prediction equation (4-36). If a prediction is unassigned, it has to be decided whether it is in a merged situation or not.

The merged situation can be identified by the prediction bounding box overlapping the merged detected bounding box. The detail is introduced in Figure 4.27 (a) in section 4.3.1.

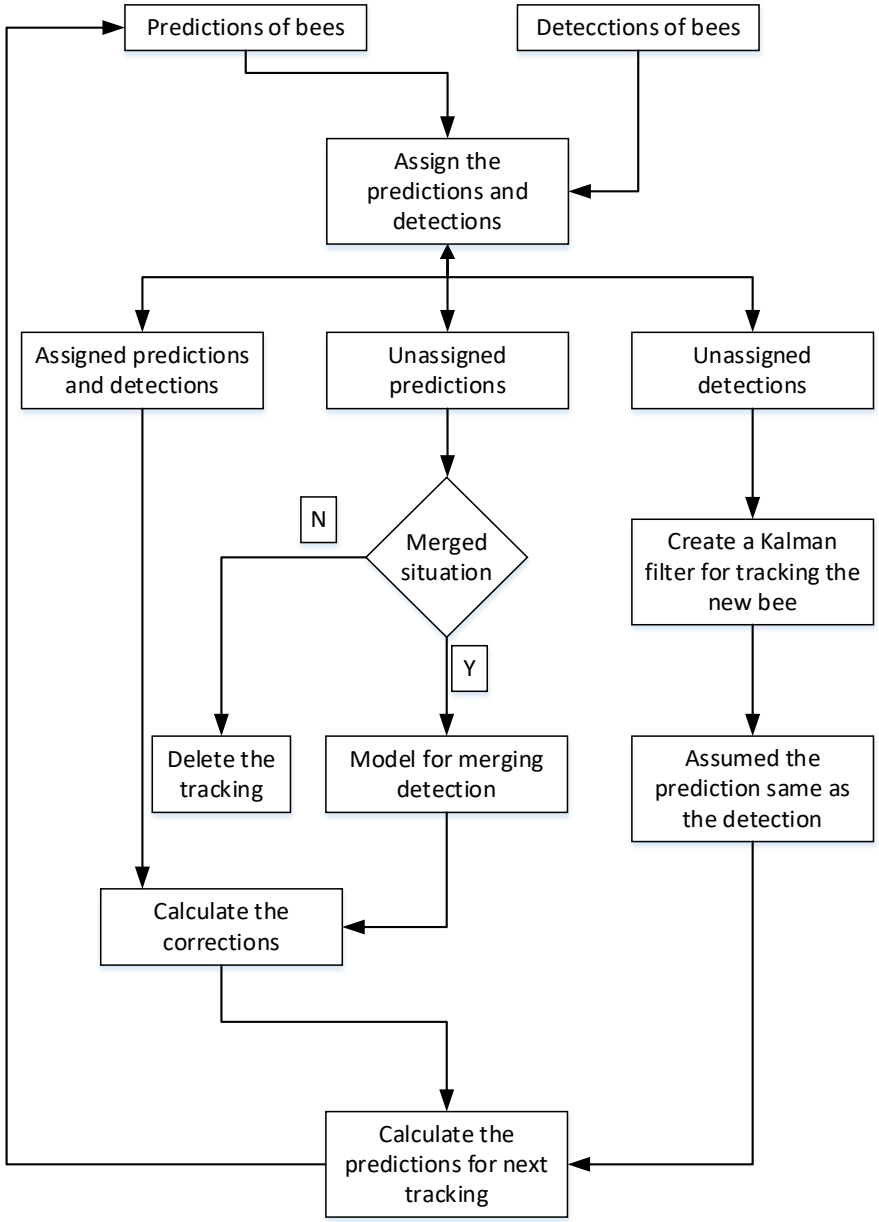


Figure 4.33 The tracking model of Kalman filter

If it is not merged, this tracking of the bee is deleted, because it has probably disappeared. If it is in the merged situation, the model for merging detection in Figure 4.33 produces the measurement of each bee's tracking. The method can:

- Use the merged blob centre as the measurement (section 4.3.2.1).
- Use the assumed detection as the measurement (Figure 4.27 (b) in section 4.3.1).
- Use the Hough transform for bees tracking (section 4.3.3).

From the above results, the covariance matrices of measurement and prediction are different for single bee tracking and merged bee tracking. The result is the best balance for the Kalman gain to calculate correction. In Figure 4.33, the correction is used to calculate the prediction for the next tracking.

4.3.2.3 Kalman filter model with assumed detection

In section 4.3.2.1 for the merged situation, a large error can be occur between the actual position and the measurement, because the measurement comes from the centre coordinates of the merged bee blob. Therefore, the covariance matrix of the measurement in merged bee tracking has a large value. The assumed detection from section 4.3.1 can reduce the error. This detection is based on the predicted position and the bounding boxes for the predictions and the merged detection. When merging takes place, the assumed detection position is used as the measurement for the Kalman filter of all bees in the merged blob, as in the model for merging detection in Figure 4.33. The distribution of the assumed detection error is shown in Figure 4.34 from the analysis of 23 video frames which include merged situations. There were 54 sample positions in these frames. The actual positions are estimated manually.

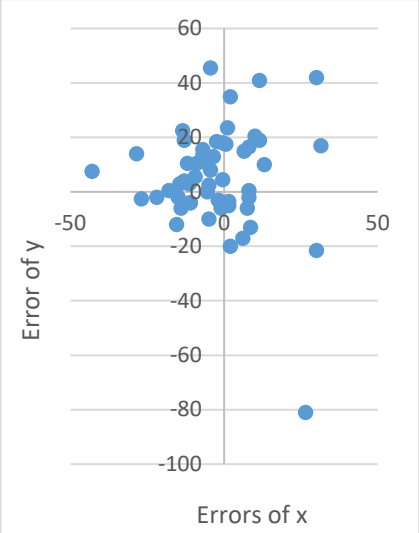


Figure 4.34 The error distribution of assumed detection error in merged situation

In this case, the assumed detection produces the following covariance matrix for the measurement:

$$R_k = \begin{bmatrix} 206 & 0 \\ 0 & 379 \end{bmatrix} \qquad 4\text{-}52$$

23 video frames were used to estimate the prediction errors and calculate the variance for merged bee tracking. There were a total of 54 position samples of merged bees in these frames. The actual positions of the bees were estimated manually. The predicted positions are from the Kalman filter prediction using merged bee tracking. The prediction errors are calculated using equation (4-43). The error distribution is shown in Figure 4.35.

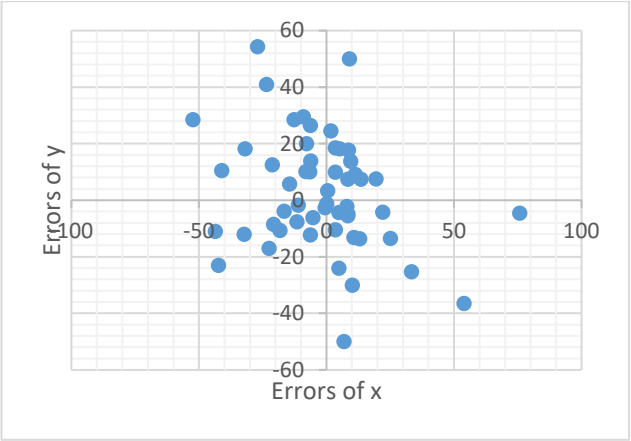


Figure 4.35 The error distribution of prediction in merged situation

The prediction covariance matrix for merged bee tracking is:

$$C_0 = \begin{bmatrix} 502 & 0 \\ 0 & 410 \end{bmatrix} \quad \text{4-53}$$

It can be seen that the variance of the measurement in equation (4-52) is less than in equation (4-50), but the variance of the prediction in equation (4-53) is similar to equation (4-51). The R_k is slightly smaller than C_0. When merging occurs, according to equations (4-37) to (4-41), the Kalman filter correction is closer to the measurement than prediction. However, because the assumed detection relies on the prediction before the merging, if the bee changes direction during the merging, the detection will lose the bee. Therefore, an improved mathematical approach is necessary to calculate the measurement of the Kalman filter in the merged situation.

4.3.3 Hough transform for improvement of merged bee tracking

Because bees have the same colour and texture, it is impossible to use colour or texture to detect bees in the merged situation. The Hough transform is a mathematical model for object detection that depends on the shape of the object being detected. In this research, the shape parameters of the individual bees are produced from the frame just before merging. Then these parameters are utilized to detect individual bees in the merged situation.

Figure 4.36 The segmentation of bees blob.

Note: (a) The first frame of two bees (b) The blob segmentation of (a). (c) The next frame of the two bees. (d) The merged blob segmentation of (c).

After merging has been identified, as discussed for Figure 4.27 (a) in section 4.3.1, the next step is to detect bees in the merged blob. Figure 4.36 displays the situation. Figure 4.36 (b) shows the segmentation of two bee blobs. For the next frame, Figure 4.36 (d) gives the result of

segmentation of the merged blob. The solution is to use the two individual bee shapes in Figure 4.36 (b), to detect each bee in the merged blob in Figure 4.36 (d).

Although the bee shapes appear different in the merged detection, the single bee shapes in the Figure 4.36 (b) frame can be partly seen in the merged shape of the Figure 4.36 (d) frame. Therefore, the shape parameters from the Figure 4.36 (b) frame can be used to detect the individual bees in the merged frame of Figure 4.36 (d). However, this method can only find an approximate area of pixels for each bee's position, because of changes in the shapes.

4.3.3.1 Definition of images

In Figure 4.36 (b), the blob of each single bee can be cropped out. This blob image can be used to produce the edge image using Canny method which is applied by a MATLAB function. Using the bee blob near the bottom right corner in Figure 4.36 (b) as an example, Figure 4.37 shows the edge of this blob which is obtained using Canny method. It is assumed that the edge image (I_s) has width ***I*** and height ***J***. The image pixel at i, j has the value:

$$I_s(i,j) = \begin{cases} 1, & on\ the\ edge \\ 0, & not\ on\ the\ edge \end{cases} \qquad 4\text{-}54$$

where $1 \leq i \leq I$ and $1 \leq j \leq J$. This is the boundary image that shows the edge of the bee blob.

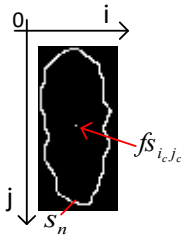


Figure 4.37 The edge image of a single bee blob.

The centre of the image is at (i_c, j_c). The pixels with value "1" define the edge curve on the image I_s. The set of pixels (**S**) on the edge of the single bee blob is:

$$S = \{(i,j) | I_s(i,j) = 1\} \qquad 4\text{-}55$$

It is assumed that the set of edge pixels has *N* members. Each member of the set **S** is denoted by $s_n = (i_n, j_n)$ representing the coordinates of edge pixel number $n \in [1, N]$.

In the edge image, the origin is at the top left corner. To detect a single bee in the merged blob, it is convenient to shift the origin to the centre of the edge image. The transformation is:

$$c_n = [u_n, v_n] = [i_n - i_c, j_n - j_c] \qquad 4\text{-}56$$

so that the transformed edge pixel set (**C**) is:

$$C = \{c_n | n \in [1, N]\} \qquad 4\text{-}57$$

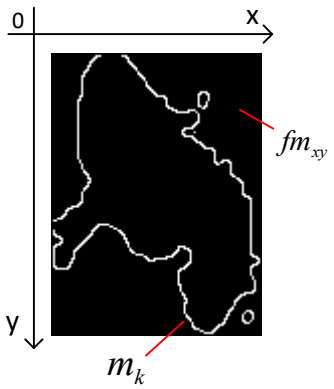


Figure 4.38 The edge image of the merged bee blob.

Figure 4.38 shows the merged blob edge of the two bees in Fig. 2 (c). This blob can also be cropped. The edge image of the merged blob (I_m) has width *X* and height *Y*, and its pixels are:

$$I_m(x,y) = \begin{cases} 1\,, & on\ the\ edge \\ 0\,, & not\ on\ the\ edge \end{cases} \qquad 4\text{-}58$$

where $1 \le x \le X$ and $1 \le y \le Y$. The pixels with value "1" are on the edge of the blob on the image I_m. The set of pixels (M) on the edge of the merged blob is:

$$M = \{(x,y) | I_m(x,y) = 1\} \qquad 4\text{-}59$$

This set of pixels has *K* members. Each member is denoted by $m_k = (x_k, y_k)$ representing the coordinates of pixel number $k \in [1, K]$.

4.3.3.2 The Hough transform

In the Hough transform technique, the edge pixels of the single bee blob are drawn on the merged blob edge image, with the centre of the single bee blob positioned on each pixel on the edge of the merged blob. Therefore, if the merged blob edge has ***K*** pixels, the single blob edge is drawn ***K*** times on the merged blob edge image. The coordinates of the resulting pixels drawn on the image are

$$t_{nk} = [u_n + x_k\,, v_n + y_k] \qquad 4\text{-}60$$

This transform set is known as *T*, where:

$$T = \{t_{nk} | n \in [1, N], k \in [1, K]\} \qquad 4\text{-}61$$

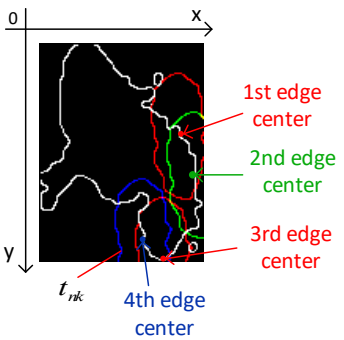


Figure 4.39 Examples of drawing the single bee blob edge on the merged blob edge image

These are the Hough Transform pixels. Some of the pixels t_{nk} are in the same position as each other. Figure 4.39 is an example of the single bee blob edge drawn four times on the merged blob edge image, with each drawing shown as a different colour. The centre of each drawing is at a different position on the edge of the merged blob. All of the pixels on these single bee edge drawings belong to *T*.

4.3.3.3 The voting map

Each pixel in the merged blob edge image (I_m) may match none, one or several of the Hough Transform pixels (*T*). This creates a voting map ($V(\mathrm{x},\mathrm{y})$), whose size is the same as the size of merged blob edge image. Each element of the voting map array corresponds to a pixel of the merged edge image. These elements are bins that record the number of Hough Transform pixels located at position (x,y).

For (x,y) inside the merged blob, $V(\mathrm{x},\mathrm{y})$ is the number of t_{nk} values equal to (x,y). These points are $x \in [1, X],\ y \in [1, Y]$. $V(\mathrm{x},\mathrm{y})$ is also called the voting value.

The voting map shows the possible positions of the single bee on the merged image. If the single bee blob edge fits part of the merged blob edge, there is expected to be a corresponding element of voting map that has a peak value. This peak point is a candidate for the single bee's position in the merged blob.

4.3.3.4 The rotation of the bee

However, it has been found the motion of the bee can change its orientation by up to $\pm 10°$ between successive video frames. If the bee changes orientation, the voting map may produce a lower valued peak point. By using several fitting orientations for each merged blob edge pixel, the voting map can also estimate the chnaged orientation of the bee. The rotation of the bee's blob is described as:

$$\alpha = \{\alpha_l | (l \in 1,2,\dots,L)\} \qquad \textit{4-62}$$

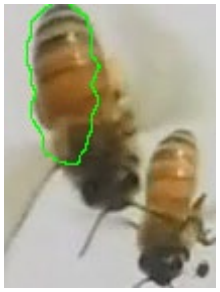
Where the L=21 and the

$$[\alpha_1, \dots \alpha_{21}] = [-10°, -9°, \dots, 0°, \dots, 9°, \dots, 10°] \qquad \textit{4-63}$$

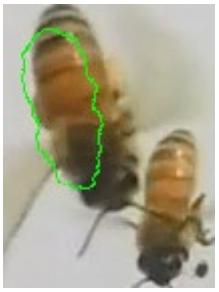
are the angles which are used to rotate the single bee blob edge from its original orientation. The plus sign indicates an anticlockwise rotation, and the minus sign indicates a clockwise rotation. If the single bee blob edge is rotated to α_l, the edge pixel coordinates relative to the centre of the single bee blob are:

$$c_{n\alpha_l} = [u_n \times cos\alpha_l + v_n \times sin\alpha_l,\ v_n \times cos\alpha_l - u_n \times sin\alpha_l] \qquad \textit{4-64}$$

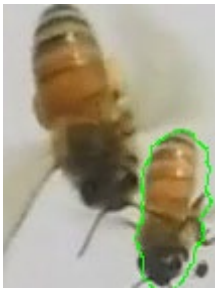
Each rotated single bee blob edge curve can be drawn on the merged edge image to generate the voting map for an α_l rotation. In this case, there are 21 different voting maps corresponding to the 21 different rotations. The different voting maps display different valued peak points. If a rotation makes the single bee edge fit the merged edge, a peak point in the voting map is high and sharp, with other peak points being much lower. If the rotation does not fit the merged edge, there may be two or three peak points with a similar, but lower, value.



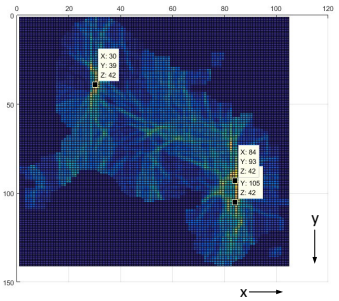
(a) 0° rotation



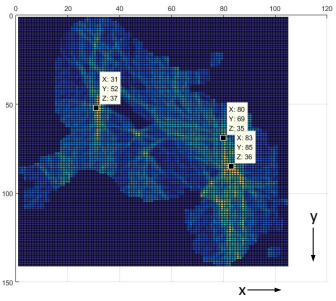
(d) 10° rotation



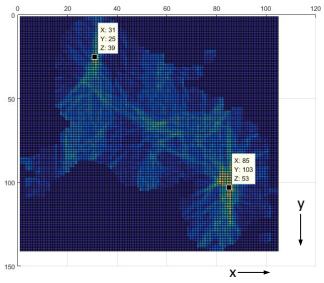
(g)−10° rotation (fitting)



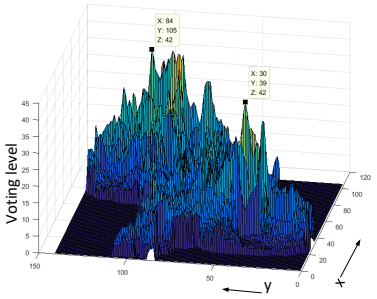
(b) 0° rotation



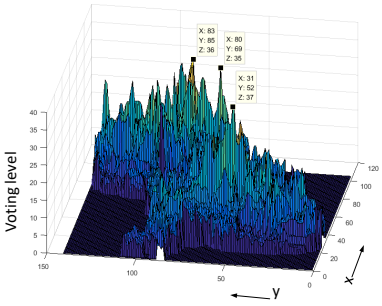
(e) 10° rotation



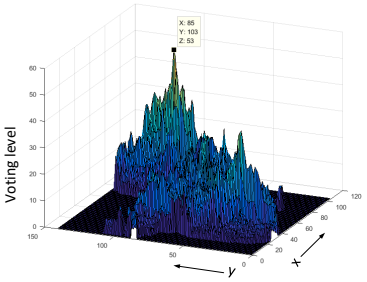
(h)−10° rotation (fitting)



(c) 0° rotation



(f) 10° rotation

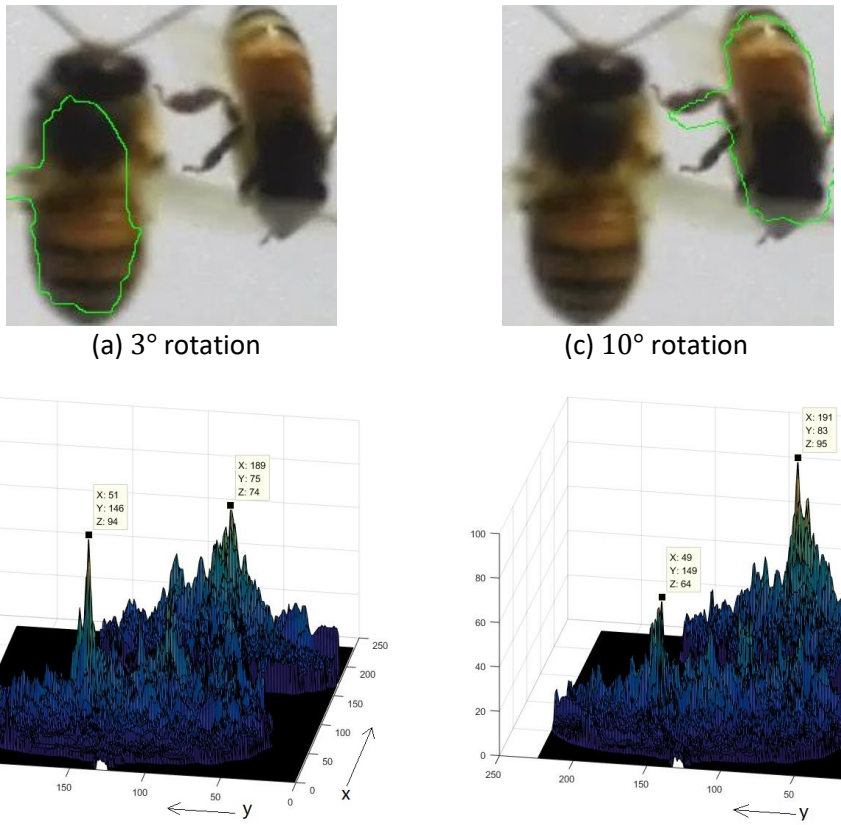


(i)−10° rotation (fitting)

Figure 4.40 The final detection of results with different rotation of the single bee edge. Notes: The −10° rotation fit the detection, and voting map have top peak point.

Figure 4.40 shows an example. In this figure, the **X** and **Y** values are the coordinates of the peak point, and the **Z** value is the voting value of the point. The rotation of 0 degrees means no change in the orientation of the single bee blob edge (Figure 4.40 (a-c)). In this case, there are three peak points having the same value. The rotation of 10 degrees in Figure 4.40 (d-f), does not fit the merged edge. The three peak points have similar values of 35, 36 and 37. Figure 4.40 (g-i) is the fitting rotation in which the voting map has the sharp peak point with a value of 53, while the second highest peak value is only 39. It can be seen that the fitting rotation produces the correct detection and this usually has the highest peak value.

In summary, the 21 voting maps produce 21 candidate positions: since each voting map produces exactly one candidate position, corresponding to its peak value. However, the fitting rotation may not be the highest peak value. This is because the single bee shape sometimes fits another part of the merged shape rather than the correct part, so that the wrong position also includes the high peak point. If the peak point is near the correct position, the region around this point includes many higher voting value points. This is described as the **correct region**. Conversely, the region



(a) 3° rotation (c) 10° rotation

(b) 3° rotation (d) 10° rotation

Figure 4.41 The peak point problem with different rotation. The fitting rotation is 10°

around the wrong position has some low value points, and this is known as the **incorrect region**, so that this peak point is sharp and isolated.

Figure 4.41 shows the situation. It can be seen that the top point of the wrong rotation (Figure 4.41 (a-b)) has a similar value (94) to the top point of the fitting rotation (value 95 in Figure 4.41 (c-d)). However, it is difficult to choose the position of the bee from the similar top points in Figure 4.41. The problem is solved by using the sum of the region values around the top points. In each of the 21 voting maps, all the values at voting points within ±5 pixels of the top point are added together. It is assumed the top point in the voting map for orientation index $\boldsymbol{l}$ is $V_l(x_p, y_p)$, and the total value of the region around this top point is:

$$r_l = \sum_{x=x_p-5}^{x_p+5} \sum_{y=y_p-5}^{y_p+5} V_l(x, y) \qquad 4\text{-}65$$

Where $l \in [1,21]$. The top value of the 21 region total values produces the correct region which includes the actual position of the bee. In addition, l gives the orientation of the bee. Figure 4.41 shows an example. After the calculation, the region around the top point of Figure 4.41 (b) has a value of 3950; but the value of the region around the top point of Figure 4.41 (d) is 5501. Therefore, the correct region is determined from Figure 4.41 (c-d) in which the fitting rotation is 10°.

The top point in the correct region might be expected to be the candidate position. However, this point is only near the actual position, because of the shape noise. The shape noise comes from the segmentation. It is difficult to get accurate shape detail when performing segmentation, but

the detection can be improved by a weighted average calculation near the correct region. The correct region itself is too small to be used to modify the position calculation. The correct region is extended to an area of ±20 pixels from the peak point, and a weighted average of the voting map is calculated as below:

$$w_x = \frac{\sum_{x=x_p-20}^{x_p+20}\sum_{y=y_p-20}^{y_p+20} xV(x,y)}{\sum_{x=x_p-20}^{x_p+20}\sum_{y=y_p-20}^{y_p+20} V(x,y)} \quad 4\text{-}66$$

$$w_y = \frac{\sum_{x=x_p-20}^{x_p+20}\sum_{y=y_p-20}^{y_p+20} yV(x,y)}{\sum_{x=x_p-20}^{x_p+20}\sum_{y=y_p-20}^{y_p+20} V(x,y)} \quad 4\text{-}67$$

In the formula above, the point at (w_x, w_y) replaces the peak point as the candidate position of the bee.

This is the final detection position of the bee on the merged image. The other merged bee is detected in the same way.

In the next merged frame, the fitting rotation still uses the original orientation, and the single bee detection obtained just before the first merged frame is still used to detect the merged bee. This method operates until the two bees separate.

4.3.3.5 The covariance matrix of the Kalman filter

The covariance matrices were estimated in section 4.3.2.1 for single bee tracking. However, because the Hough transform is utilized for merged bee tracking, the prediction and measurement covariance matrices should be re-estimated. 82 video frames (including 184 position samples) were used to calculate the measurement and prediction covariance matrices.

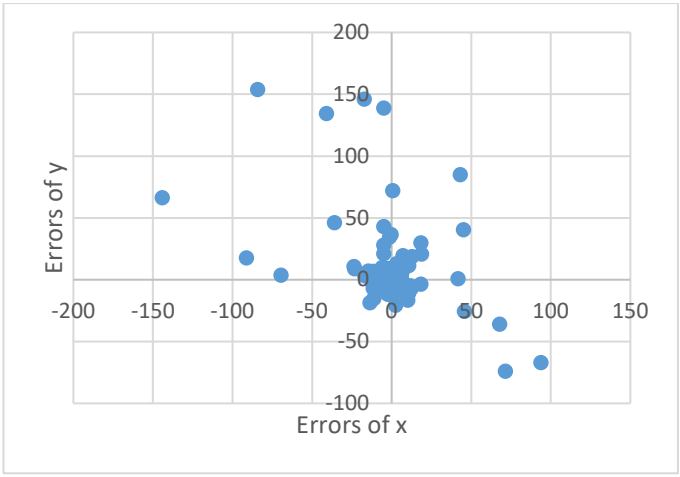


Figure 4.42 The measurement errors distribution from Hough transform

For the measurement error, the actual positions of bees are calculated manually. The measurement positions are calculated using the Hough transform. The measurement errors are calculated by using equation (4-42). The distribution of these measurement errors is displayed in Figure 4.42. Some errors are large, because the Hough transform can be unsuccessful for calculating the bee position.

The measurement covariance matrix from the Hough transform for merged bee detection is:

$$R_k = \begin{bmatrix} 419 & 0 \\ 0 & 717 \end{bmatrix} \quad \textit{4-68}$$

For the prediction error, the actual positions of bees were estimated manually. The predicted positions are from the Kalman filter prediction of merged bee tracking. The prediction errors are calculated by using equation (4-43). The error distribution is shown in Figure 4.43.

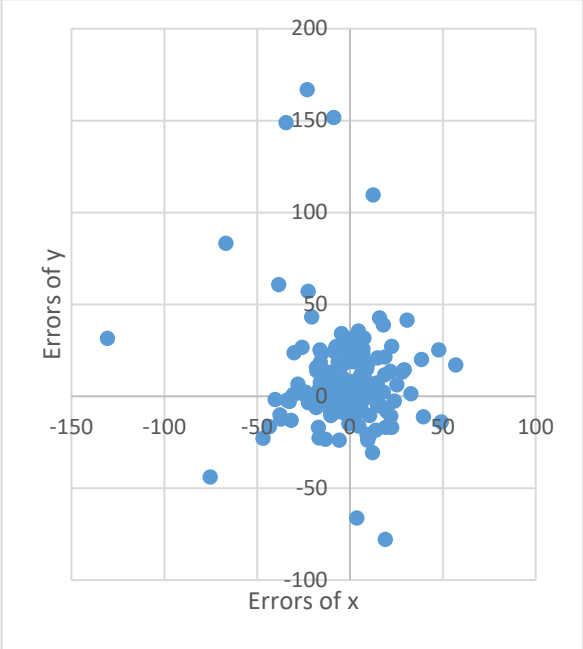


Figure 4.43 The prediction errors distribution with Hough transform

The prediction covariance matrix for merged bee tracking becomes:

$$C_0 = \begin{bmatrix} 435 & 0 \\ 0 & 825 \end{bmatrix}$$

These two covariance matrices above are also assumed to be constant during merged bee tracking. The Hough transform model is the model for merged detection in Figure 4.33. However, the Hough detection can sometimes be wrong. The wrong detection detects another part of the merged blob rather than the actual position of the bee, so the error can be high. This can be seen from Figure 4.42 and can lead to incorrect tracking.

4.3.4 Brief explanation of the Hungarian assignment method for bees tracking

In this research, the Hungarian assignment method is applied by a Matlab function (assignDetectionsToTracks). For bee tracking, the predicted positions (prediction) and detected positions (detection) are the pixel coordinates. A matrix records distances between each prediction and detection. The Hungarian assignment method uses this matrix to assign predictions and detections. The maximum distance is 80 pixels, which means that if a prediction (or detection) is more than 80 pixels from any detection (or prediction), this prediction (or detection) is unassigned.

If the numbers of predictions and detections are the same, they are assigned by the Hungarian assignment method. The optimal assignment is that the sum of all of the distances is the smallest

possible. If the number of predictions (or detections) is greater than the number of detections (or predictions), the remaining detections (or predictions) are unassigned.

Chapter 5

Chapter 5 Methodology of pollen sacs detection and measurement

This chapter describes the pollen detection and measurement using two methods. The first method is to use image processing and statistics. For this, bee blob analysis is used to remove the main body of the bee. The other parts of the body which have different colours from pollen are removed by colour thresholding. The pollen sacs are finally discriminated from noise blobs by using various feature thresholds. The pollen and non-pollen blobs are measured by using an estimation model and a ROC curve. The second method is to utilize a deep learning algorithm. The Faster RCNN and VGG-16 network is used for detecting pollen sacs on individual bee images. Then, individual bee images can be identified as pollen or non-pollen images, depending on whether the pollen detection network finds pollen sacs on the images or not. Both methods are finally applied on the honey bee monitoring video. A concept of pollen carrying ratio is created to identify whether a bee is carrying pollen or not on the video. After that, the number of pollen carrying bees can be measured.

5.1 Pollen sacs detection and measurement with image processing and statistics

Single bee images are cropped from the whole image frames, and the binary image of the single bee blobs are also cropped from the whole binary images. The initial idea is to analyse the bee's body shape by calculating image moments of the blob. Then, the main (centre) body of the bee is detected and separated. After that, possible pollen is distinguished from the other separated parts of the bee, such as legs and wings. Then pollen features are calculated for identification of pollen sacs.

5.1.1 Detect main body of bee

A bee has a main body (including head and tail) which is similar to a standard ellipse. The bee's blob on the binary image is analysed by calculating geometric image moments. As shown in section 3.7, this produces the following bee blob parameters: centre position (x_c, y_c), orientation (θ), height (h), width (w), major axis ($2a$) and minor axis ($2b$). Figure 5.1 shows an example of the parameters. The white colour is the bee blob. The green ellipse (in the middle) is calculated from the moments of the blob. The ellipse formula is:

$$E(x,y) = \left(\frac{(x-x_c)\cos\theta+(y-y_c)\sin\theta}{a}\right)^2 + \left(\frac{(x-x_c)\sin\theta-(y-y_c)\cos\theta}{b}\right)^2 = 1 \qquad 5\text{-}1$$

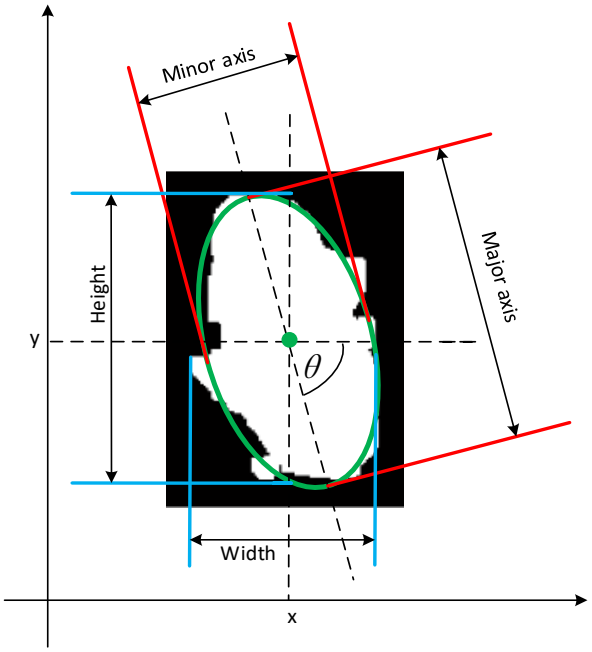


Figure 5.1 An example of blob parameters

The ellipse shown in Figure 5.1 is the candidate main body of a bee.

Figure 5.2 The result of using the ellipse to separate the blob.

Note: (a) The original blob of the honey bee. (b) The ellipse which is calculated from the blob moments. (c) The pixels belonging to the bee's body and inside the boundary of ellipse. (d) The pixels belonging to the bee's body and outside the ellipse boundary. (e) The pixels inside the boundary of ellipse, but not belonging to the body.

If a pixel (x, y) is inside the ellipse, it will satisfy the inequality $E(x, y) < 1$. This is used to display the set of ellipse pixels (blob), as displayed in Figure 5.2 (b). Using Figure 5.2 (a) as an example, the bee's blob image can then be separated into three parts, when compared with the ellipse of $E(x, y)$. The first part is the pixels which belong to the bee blob (body of the bee) and are inside the boundary of the ellipse: ($I(x, y) = 1$ and $E(x, y) \leq 1$) (Figure 5.2 (c)). The second part is the pixels which belong to the body of the bee, but are outside the boundary of the ellipse. This comprises the protruding parts: ($I(x, y) = 1$ and $E(x, y) > 1$) (Figure 5.2 (d)). The last group of pixels is inside the boundary of ellipse, but the pixels are not part of the body: ($I(x, y) = 0$ and $E(x, y) \leq 1$) (Figure 5.2 (e)).

The candidate main body blob is obtained by the following steps, as illustrated in Figure 5.3:

1) Calculate the fitting ellipse for the bee blob as Figure 5.3 (b).

2) The fitting ellipse is used to remove the protruding parts (the second part pixels in Figure 5.3 (d)) and the first part is kept as a new blob (Figure 5.3 (c)). The new blob is smaller than the original blob, but it is also closer to the main body of the actual bee.

3) The image moments and ellipse calculation can be performed again on the new blob, to obtain a new ellipse with parameters.

4) These new parameters are used to give another smaller ellipse to remove protruding parts again.

The procedure is repeated until all protruding parts have been removed. This results in a smaller elliptical blob that is a better estimate of the main body of the bee (Figure 5.3 (f)).

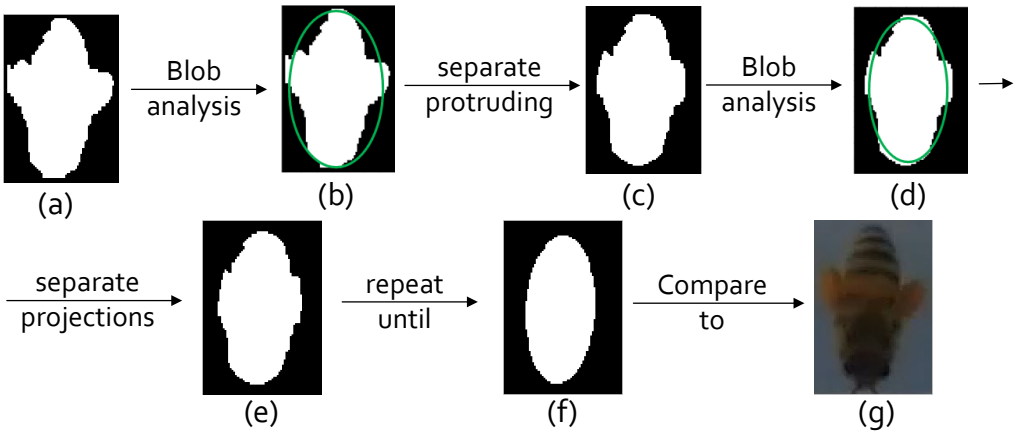


Figure 5.3 The example of a bee's main body blob detection

5.1.2 Detect colour of pollen sacs

Once the main body has been detected, the other parts of the bee can be identified, such as the wings, legs and pollen sacs. The main body is removed from the colour image, and the remaining parts and the background are segmented. Normally, the pollen has orange, yellow and white colours, so the colour histogram thresholding produces the final blobs from the background on the single bee image. The method is shown in Figure 5.4.

Figure 5.4 (a-f) shows the detection of the pollen sacs. The bee image (a) is detected and cropped from the whole frame. The blob image (b) is cropped from the whole binary image. Then this blob is approximated by an ellipse (c) that corresponds to the main body of the bee, and the main body ellipse is then eliminated from the bee image. The result is displayed in (d), which only includes the peripheral parts of the bee, such as parts of its tail and wings, and the pollen sacs. Normally, the pollen sacs have orange, yellow or white colours, with the colour segmentation being utilised to detect the pollen blob shown in image (e). The orange and yellow colour are close to each other in the Hue channel, so they can be detected together. White colour can be detected from the Saturation and Value channels in HSV colour space. The colour detection model generates two binary images which are the mask pollen orange and yellow image (MpOYI) and the mask pollen white image (MpWI). The final pollen colour detection model is:

$$MPCI = MpOYI \vee MpWI \qquad 5\text{-}2$$

where MPCI is the mask pollen colour image which is shown in Figure 5.4 (e). This is a binary image that shows blobs which have pollen's colour. The MpOYI binary image masks the orange and yellow colours. Each pixel ($p_{\mathrm{MpOYI}}(\mathrm{x},\mathrm{y})$) value is calculated from the hue channel using:

$$p_{MpOYI}(x,y) = \begin{cases} 1, & 0.02 < h(x,y) < 0.15 \\ 0, & otherwise \end{cases} \quad \textit{5-3}$$

The MpWI image masks the white colour of pollen sacs. Each pixel ($p_{\mathrm{MpBI}}(\mathrm{x},\mathrm{y})$) value is calculated from the value and saturation channels using:

$$p_{MpWI}(x,y) = \begin{cases} 1, & v(x,y) > 0.7 \cap s(x,y) < 0.4 \\ 0, & otherwise \end{cases} \quad \textit{5-4}$$

Figure 5.4 The pollen detection process.

Note: (a) The original image of the bee cropped from the whole colour frame. (b) The binary blob of image (a). This is cropped from the whole binary image. (c) The main body blob. (d) The image of parts other than the main body. (e) The pollen sacs with other dust blobs. (f) The pollen colour detection shown on a colour image.

A problem is that the pollen may not be visible after the bee's main body detection, so that the pollen colour detection can not detect the pollen. This commonly happens when using the bee detection model **B** that was discussed in section 4.2.4. An example was shown in Figure 4.23. The image in Figure 4.23(b) comes from this bee detection model. The blob does not include the pollen sac, so that the peripheral parts of the bee body shown in Figure 5.5(a) do not include the pollen sac. The pollen detection result (Figure 5.5(b)) only detects a small blob that is hard to identify as a pollen sac. Conversely, Figure 4.23(d) displays the result of bee detection using model **D**, which adds white colour detection to the bee detection. The different characteristics of models **B** and **D** are shown in Table 4-1 in section 4.2.4.2. The advantage of model **D** is shown in Figure 5.5(c), where the bee body's peripheral parts detection includes the pollen sac. Thus the final pollen detection (Figure 5.5(d) detects a blob that can be identified as a pollen sac. In this way, the bee detection model **D** helps to improve pollen detection.

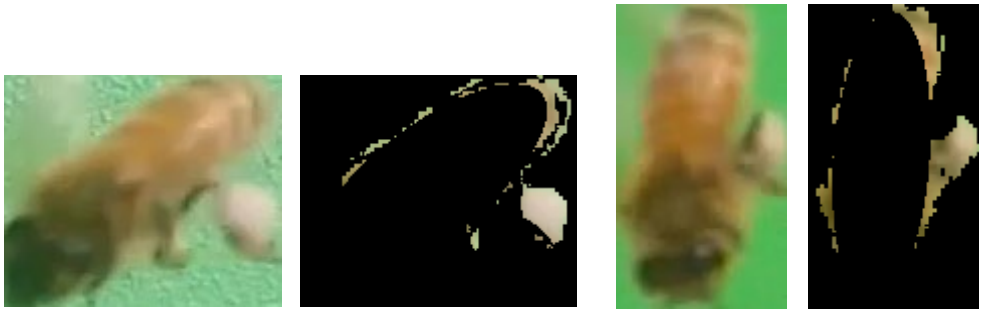
Figure 5.5 The comparison of the result with different bee detection.

*Note: (a) The result of bee detection from model **B**. (b) The result of pollen detection from (a). (c) The result of bee detection from model **D**. (d) The pollen detection result from (c).*

Another problem is shown in Figure 5.4 (e), in which the pollen image includes not only the pollen blob but also other noise blobs. This is because the bee detection is sometimes not precise

enough. Figure 5.4 shows a particular example, in that there may be some colour misdetection. The noise blobs affect the pollen detection. If this method is applied to a bee without pollen, the noise blobs may be misinterpreted as pollen. Therefore, to make sure all of the detected blobs include any pollen blobs, the blob features need to be analysed.

Other noisy examples are shown in Figure 5.6. In image (b) and (d), it can be seen that the pollen detection model also detects other parts of the bee as pollen. It is necessary to identify pollen more accurately using other features of the pollen sacs.



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

Figure 5.6 Other examples of pollen detection result.

5.1.3 Pollen blob features

It is necessary to identify the features of pollen blobs that are different from other blobs, so that the pollen sacs can be recognised by the system. There are five features automatically produced by the blob detection process that could be useful. These are the blob area, orientation, size ratio, extent and position.

The first feature is the **area** of the pollen region, which is the number of pixels inside the blob. However, a larger bee blob leads to the pollen blob also being larger. Therefore, simply using the pollen blob area will not be accurate. The solution is to calculate the **area ratio** (R_A) between pollen blob and the bee main body (ellipse) blob:

$$R_A = \frac{A_p}{A_m} \qquad 5\text{-}5$$

A_p is the pollen blob area, and A_m is the single bee main body blob area.

The second feature is the **orientation** of the pollen blob. Because the pollen sac has an elliptical shape, its orientation is similar to the orientation of the flying bee. The difference between the orientations of the pollen blob and the bee blob (D_o) is:

$$D_o = |o_p - o_b| \qquad 5\text{-}6$$

o_p is the orientation of pollen blob, and the o_b is the orientation of **main body** (the main part of the body) of bee. However, the results later show that the orientation is not reliable for pollen detection.

The third feature is the ratio of the minor and major axes (**shape ratio**) of the pollen blob ellipse. If b_p is the minor axis of the pollen blob, and a_p is the major axis, the size ratio is:

$$R_S = \frac{b_p}{a_p} \quad 5\text{-}7$$

As the pollen sac has an elliptical shape the size ratio is different from some noise blobs which look more like straight lines.

The fourth feature is the **blob extent**. The blob analysis draws a bounding box around the pollen blob. The blob extent (E) is defined as the area of the pollen blob relative to its bounding box area:

$$E = \frac{A_p}{A_b} \quad 5\text{-}8$$

A_b is the area of the bounding box, calculated in pixels. This feature can be used to eliminate arc-shaped noise. The pollen blob has a higher value of blob extent than these noise blobs.

The last feature is the location of the pollen blob (**position distance ratio**). The pollen is beside the bee's body. The positions of the pollen sacs should be to the left and/or the right side of the bee's main body. The ends of the minor axis of the main body ellipse represent the left and right side of the main body.

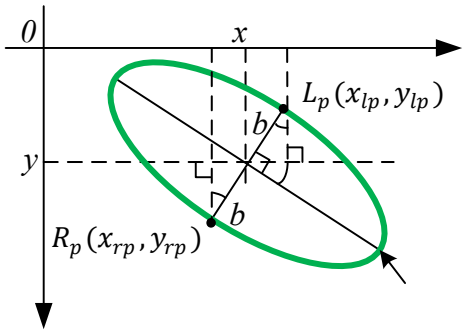


Figure 5.7 The main body ellipse showing the left and right points of the minor axis.

Figure 5.7 shows the main body ellipse. It is assumed the head of the bee points in the direction of the positive x-axis. The orientation of the ellipse ($\theta \in (-90^o, 90^o)$) is the angle between major axis of the ellipse and the positive x-axis. θ is less than zero in the situation shown in Figure 5.7. The point L_p is on the left side of the bee, which is at the top end of minor axis of the ellipse in the figure, and the point R_p is on the right side of the bee, which is at the bottom end of the minor axis of the ellipse in the figure. In Figure 5.7, the point positions are:

$$R_p(x_{rp}, y_{rp}) = (x + b\,sin\,\theta\,, y + b\,cos\,\theta) \quad 5\text{-}9$$

$$L_p(x_{lp}, y_{lp}) = (x - b\,sin\,\theta\,, y - b\,cos\,\theta) \quad 5\text{-}10$$

x and y are the centre position coordinates of the main body, b is the half minor axis. The pollen sac blobs should be near the left point or right point. The distances between the left/right points and the pollen blob centre position (P_b) are calculated as:

$$D_{pr} = \sqrt{|R_p - P_b|^2} = \sqrt{(x_{rp} - p_{bx})^2 + (y_{rp} - p_{by})^2} \quad 5\text{-}11$$

$$D_{pl} = \sqrt{|L_p - P_b|^2} = \sqrt{(x_{lp} - p_{bx})^2 + (y_{lp} - p_{by})^2} \quad 5\text{-}12$$

D_{pr} and D_{pl} are the distances to the blob position. p_{bx} and p_{by} are the x and y coordinates of the blob position. However, the distances will depend on the size of the bee's main body. Therefore, they need to be calculated as q_{Rp} and q_{Lp}, relative to the minor axis of the main body ellipse.

$$q_{Rp} = \frac{D_{pr}}{2b} \quad \textit{5-13}$$

$$q_{Lp} = \frac{D_{pl}}{2b} \quad \textit{5-14}$$

b is the half minor axis of the main body of the bee. The minimum value of q_{Rp} and q_{Lp} ($Q_p = \min\{q_{Rp}, q_{Lp}\}$) is the final value that can be used for pollen detection. This is called the pollen **position distance ratio**.

5.1.4 The pollen feature detection model for pollen detection

The previous section defined five features of a pollen sac blob that may be useful to distinguish pollen blobs from other similar blobs (non-pollen blobs). However, some features may not be useful. For example, the orientation feature is not useful. Some pollen blobs have an orientation that is different from the main body of the bee, as can be seen in Figure 5.6 (b), for example. In addition, some pollen blobs are detected as a circle by the pollen colour detection. Figure 5.4 (f) shows this type of pollen blob. Therefore, the feature of **orientation** of the pollen blob is not useful. The other four features cannot be checked as simply as the orientation. They need further analysis to calculate suitable thresholds for pollen blob detection.

5.1.4.1 Statistics analysis for the blob features

To choose the useful features and their corresponding thresholds, it is necessary to collect samples of pollen blobs and non-pollen blobs. However, pollen blob sample collection is more difficult than for non-pollen blobs. On the videos, only a small number of bees bring pollen back, with the maximum ratio being only about one-tenth of the bees. Only individual bee images can be used to detect pollen. The merged bee image is too complicated to be useful. A clear video was chosen to collect individual bee images, which included images that have pollen (pollen images) and images that have no pollen (non-pollen images). This video was recorded in the morning of a day. It clearly displays the pollen sacs.

The individual bee images that include pollen sacs (pollen bee images), were collected from the first 400 frames of the video. In these frames, 276 pollen bee images were cropped from the colour image and the corresponding 276 binary image blobs were cropped from the binary frames after bee detection. These images included 11 individual bees that were carrying pollen. The non-pollen bee images were also collected from the first 400 frames of the video. 1320 non-pollen bee images were cropped from these colour frames. In addition, the 1320 corresponding binary image blobs were cropped from the 400 binary frames. These images included 99 different bees that were not carrying pollen.

In the 276 pollen bee images, the pollen colour detection method detected 273 blobs that were pollen sacs (pollen blobs), and 732 blobs that were not pollen sacs (non-pollen blobs). For the pollen blobs, the mean, standard deviation, minimum and maximum values of the four features (not including orientation) were calculated. These values are shown in Table 5-1 below.

Table 5-1 The statistical parameters of features of pollen blobs

Features	Mean	Standard deviation	Minimum	Maximum
Area ratio	0.0358	0.0234	0.0012	0.1228
Shape ratio	0.4312	0.1937	0.1039	0.9426
Extent	0.5136	0.1639	0.1424	1.0000
Position Distance	0.2452	0.1908	0.0327	1.9482

For the Area ratio, the mean value is 0.0358 and the standard deviation is 0.0234. This shows the pollen blobs have a wide Area ratio distribution around the mean value. Sometimes, the pollen detection method only detects parts of the pollen sacs. Therefore, some of these detected blobs are very small which causes the minimum value of Area ratio to be only 0.0012. The maximum value of the Area ratio is 0.1228, which is almost 4 standard deviations from the mean. In some cases, these pollen blobs include other parts of the bee as well as the pollen sac itself.

The Shape ratio of the pollen blobs has a mean of 0.4312, a minimum of 0.1039, and a maximum of 0.9426. It can be seen that some pollen sacs are detected as being very thin, and some of them closely resemble a circle.

The blob Extent shows that the area of the pollen blobs relative to the corresponding bounding box has a mean value of 0.5136. It has a minimum value of 0.1424. When the blob Extent is small, the blob is very thin or arc-shaped. Some pollen blobs have the maximum value of 1.0000, because they are very small (about 12~18 pixels). Also, sometimes, the pollen sac is partly hidden behind the bee's body.

The Position Distance mean value is 0.2452, with a minimum of 0.0327 and a maximum of 1.9482. When the Position Distance value is large, the pollen blob is far away from the main body.

The pollen sac detection method is not perfect. 47 pollen sacs were not detected from the pollen images. These pollen sacs are not in the statistics, because it was not possible to obtain feature measurements from these pollen blobs.

In addition, 3250 non-pollen blobs were detected from the 1320 non-pollen images. These blobs are combined together with the 732 non-pollen blobs that were detected from the pollen images. In total, 3982 non-pollen blobs were detected and collected for the statistics. The statistics calculation of these blobs are displayed in Table 5-2 below.

Table 5-2 The statistical parameters of features from non-pollen blobs in images

Features	Mean	Standard deviation	Minimum	Maximum
Area ratio	0.0158	0.0325	0.0002	1.0887
Shape ratio	0.3060	0.1605	0.0417	1.0000
Extent	0.4437	0.1990	0.0828	1.0000
Position Distance	0.7077	0.3837	0.0103	1.9888

From Table 5-2, the mean value of the Area ratio is 0.0158, which is smaller than the pollen blob Area ratio. The standard deviation of the Area ratio is 0.0325, where the pollen blob has a

standard deviation 0.0234. It can be seen that the Area ratio of the non-pollen blob is generally smaller than the pollen blob.

The mean value of the Shape ratio is 0.3060, and the standard deviation is 0.1605. The minimum and maximum values are 0.0417 and 1.0000 respectively. The Shape ratio of the non-pollen blobs is usually slightly smaller than for the pollen blobs.

The blob Extent for non-pollen blobs has a mean value of 0.4437. The standard deviation is 0.1990. The blob Extent has minimum value of 0.0828 and maximum value of 1.0000. This feature of non-pollen blobs is similar to the pollen blobs, so this feature may not be useful.

The Position Distance feature has a mean value 0.7077 and a standard deviation of 0.3837. The minimum value is 0.0103 and the maximum value is 1.9888. The Position Distance of the non-pollen blob is generally larger than for the pollen blob.

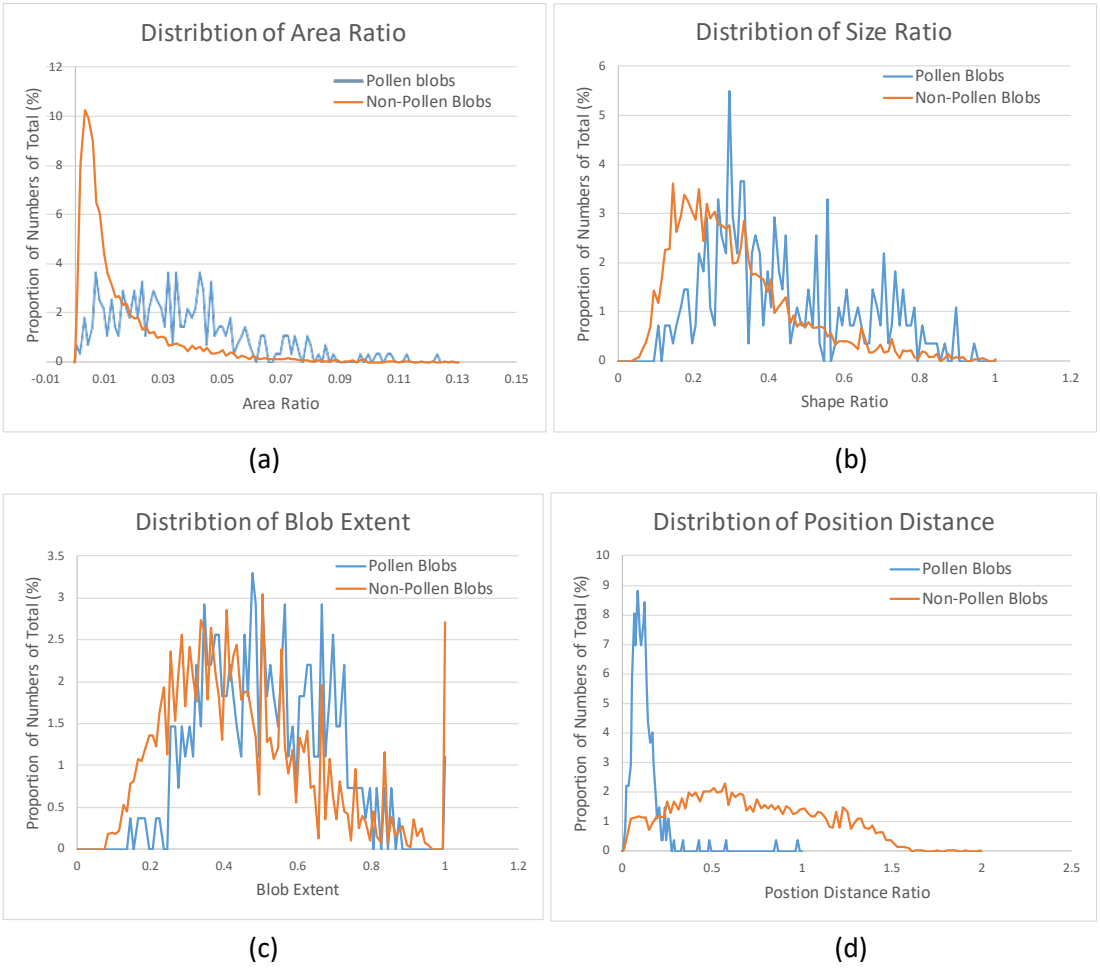


Figure 5.8 The distribution of the 4 features

The distributions of the four features are shown in Figure 5.8. It can be seen that the pollen and non-pollen blobs have different distributions for the Area ratio and Position Distance. In Image (a), most of the non-pollen blobs have a very small value of the Area ratio with the value ranging from 0 to 0.05. The Area ratio range from 0.02 to 0.06 covers most of the pollen blobs.

In Image (d), most of the pollen blobs have a value of Position Distance from 0.1 to 0.25. The non-pollen blobs are distributed over a wider range than pollen blobs, with values up to 1.5 for the Position Distance ratio.

Image (b) shows that the distribution curves for pollen and non-pollen blobs cross at about 0.3 for the Shape ratio. They share about two-thirds of their distributions. This feature cannot be used to separate pollen and non-pollen blobs.

The blob Extent distribution is displayed in image (c). The distribution curves coincide, which means the pollen and non-pollen blobs have similar values of blob Extent. This feature may not be suitable for distinguishing the blobs.

The statistics analysis shows the distribution of the four features for the pollen and non-pollen blobs. In addition, the analysis indicates the character of the pollen and non-pollen blobs. The pollen blobs generally have a bigger Area ratio than the non-pollen blobs, but most of the pollen blobs have a smaller Position Distance ratio than the non-pollen blobs. Therefore, the features of Area ratio and Position Distance can be used to distinguish pollen blobs from non-pollen blobs. However, these distributions are not suitable for finding the optimal threshold for this process. In Figure 5.8 (a), the threshold for the Area ratio appears to be value of about 0.015, but this threshold value categorises some pollen blobs as non-pollen blobs. The threshold for Position Distance appears to be a value of about 0.25, which is shown in Figure 5.8 (d). However, many non-pollen blobs also have a value less than 0.25. To find the optimal threshold, these blob statistics need further analysis.

5.1.4.2 The receiver operating characteristic (ROC) analysis

The statistics distributions do not clearly show suitable thresholds for pollen detection using these features. The optimal thresholds must not only detect more pollen sacs, but also reduce incorrect detections from bees not carrying pollen. The receiver operating characteristic (ROC) method can be used to choose the threshold values. The theory of the ROC method is explained in section 3.8. In different situations, the four outcomes from the ROC have different meanings. For this research, the meanings are listed in Table 5-3.

Table 5-3 Statistical titles explanation

		System condition	
		Pollen	Non-pollen
Actual condition	Pollen blob	True positive (TP)	False negative (FN)
	Non-pollen blob	False positive (FP)	True negative (TN)

The thresholds for each feature can be set up with different values to show the four outcomes in Table 5-3. In Table 5-3, there are two actual conditions: bee with pollen and bee without pollen. There are also two system conditions: detecting a pollen blob and detecting a non-pollen blob. Among these four conditions, the four outcomes are shown below.

- True positive (TP): system detects pollen for a pollen blob, using the thresholds.
- False positive (FP): system detects pollen for a non-pollen blob, using the thresholds.
- False negative (FN): system detects non-pollen for a pollen blob, using the thresholds.
- True negative (TN): system detects non-pollen for a non-pollen blob, using the thresholds.

To produce the ROC curves, for each feature, 100 threshold points were chosen from a value of zero to the maximum value of the samples. Each threshold point is used to perform pollen detection to get the four outcomes from Table 5-3. As a result, for each feature, the sensitivity and specificity were calculated for 100 different threshold values. Then these sensitivities and specificities were plotted in the ROC space to construct the ROC curve of each feature. (Note that the ROC x-axis is 1 – specificity, not specificity).

For the Area ratio distribution, the threshold (T_A) for the Area ratio is the lower limit for pollen detection. The positive result is $\mathrm{R_A} > T_A$, otherwise the result is negative. The Shape ratio also has a threshold that is the lower limit for a positive result. The positive is $\mathrm{R_S} > T_S$. For the blob Extent, the positive result is also $\mathrm{E} > T_E$. However, the Position Distance ratio has a threshold (T_Q), where the positive is defined as $Q_p < T_Q$.

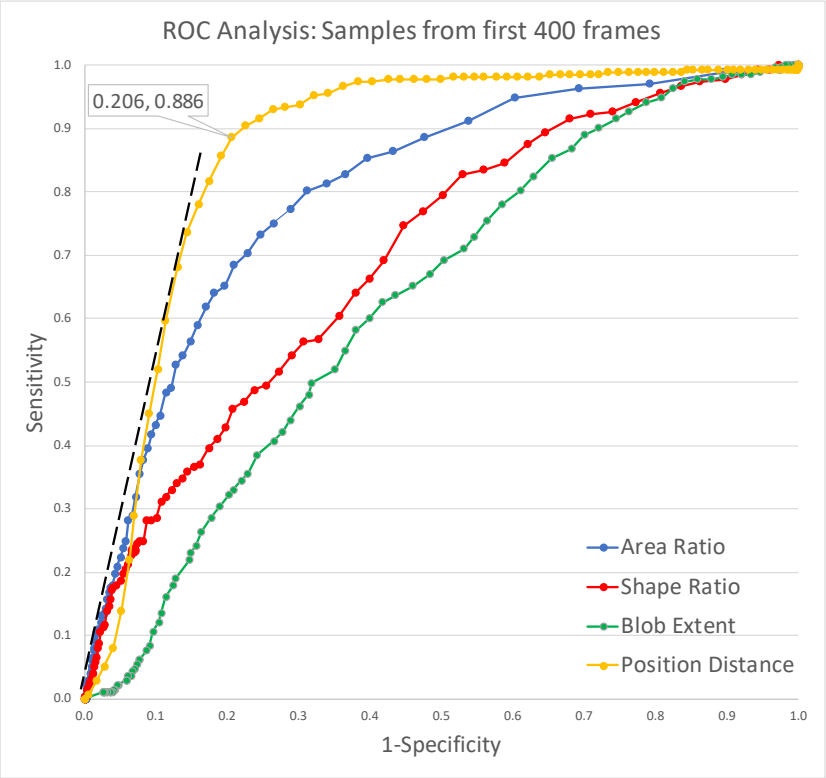


Figure 5.9 The ROC curves analysis with all of the samples.

The 3982 non-pollen blobs and 273 pollen blobs were used to calculate the ROC curves. Figure 5.9 shows the ROC analysis of all of the blobs. From this figure, the blob Extent is the worst ROC curve, so this feature is not useful, which is the result expected from the earlier examination of the statistics. The Area ratio, Shape ratio and Position Distance all have some points that are maximum sensitivity values for a given specificity value. They are the ROC convex hull (Fawcett, 2006). The convex hull can be used to generate a classifier that can help find the optimal threshold. This classifier is a straight line (shown as a dashed line in Figure 5.9) linking the two peak points in the curves of Area ratio and Position Distance. Considering this classifier (dashed line) and the Position Distance curve, the ROC point that is closest to the perfect classification (1, 0) is on the curve of Position Distance.

In the theory of (Fawcett, 2006), the point (0.206,0.886) is the closest point to the best classification of (0,1). This is the optimal threshold for the Position Distance value. The threshold value is 0.36, and positive results are obtained for position distances below this threshold. The x coordinate is one minus specificity, so the specificity is $1 - 0.206 = 0.794$. The y coordinate of this point is the sensitivity, which is 0.886.

Based on this ROC model analysis, the feature of Position Distance with a threshold of 0.36 should be used to detect pollen blobs. This method produces the four outcomes that are shown in Table 5-4.

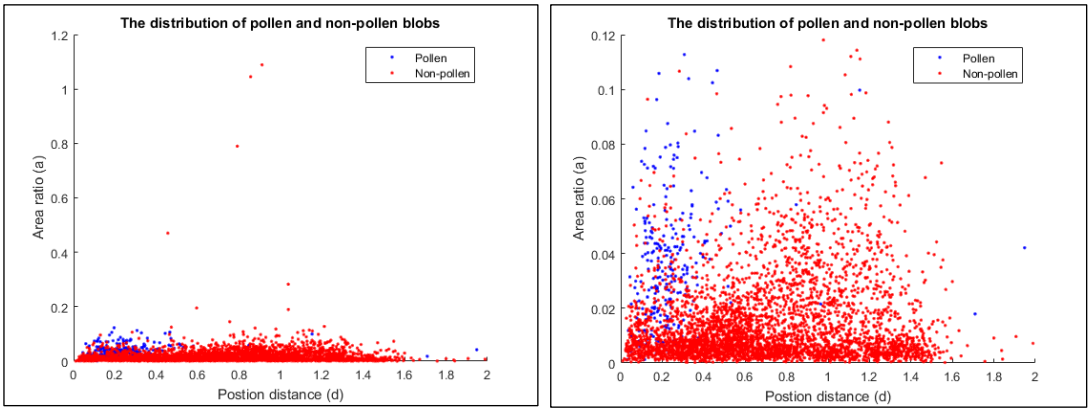
Table 5-4 The four outcomes of the ROC model

Pollen blobs	Non-pollen blobs	Threshold	TP	FN	FP	TN
273	3982	0.36	242	31	821	3161

Using this threshold, the number of true positives (TP) is 242, and the number of false negatives (FN) is 31. The number of false positives (FP) is 821, which is much smaller than the number of true negatives of 3161. However, the detection has 242 true positives (TP), and 821 false positive (FP). The proportion of true positives in the total number of positive outcomes is only 22.8%. The number of positive outcomes is the estimate of the number of pollen blobs in all of the blobs. This low precision (precision is defined in section 3.8) indicates the measurement is not accurate enough to be useful. It is necessary to improve the estimation.

5.1.4.3 Blob feature combination

The ROC method shows each feature's ROC curve on the ROC space independently. This shows Position Distance is the best single feature for pollen detection. However, in Figure 5.9, the ROC curve of the Area ratio shows that this feature could also be useful. The two features of Position distance and Area ratio can be combined to detect pollen blobs. The result of the combination cannot be shown with a single ROC curve, but the ROC space can still be used to evaluate the combined result. The two features can be combined, using a range of thresholds for each feature, to calculate the number of pollen and non-pollen blobs from the blob samples. Then the sensitivity and specificity can be calculated for each combination of threshold values.



(a) (b)

Figure 5.10 The distribution of pollen and non-pollen blobs on two features.
Notes: (a) The whole distribution. (b) The zoomed distribution.

Before using the combination procedure, the pollen and non-pollen bobs were plotted on a 2-D distribution with axes being “Position distance” and “Area ratio”. The distribution is shown in Figure 5.10. The x-axis is the “Position distance” (d-axis), and the y-axis is the “Area ratio” (a-axis). In this figure, image (a) displays all the points, but the detail is mostly in the range [0, 0.12] of the a-axis. Several non-pollen blobs have a large value of “Area ratio”, so image (a) does not focus on the main part of the distribution. Image (b) zooms the y-axis to [0, 0.12]. The distribution of the two types of blobs is shown more clearly than in image (a). From image (b), pollen blobs are mainly distributed in range 0 to 0.4 for “Position distance" and 0.01 to 0.08 for “Area ratio”. The non-pollen blobs are concentrated in the range 0 to 0.02 for “Area ratio” and 0 to 1.5 for “Position distance”. The pollen and non-pollen blobs overlap on part of the distribution, but most of the non-pollen blobs can be separated from the pollen blobs.

A straight line can be drawn on the sample map of Figure 5.10 to distinguish pollen blobs from most of the non-pollen blobs. The two variables of this line are a (“Area ratio”) and d” (“Position distance”). The straight line is defined below.

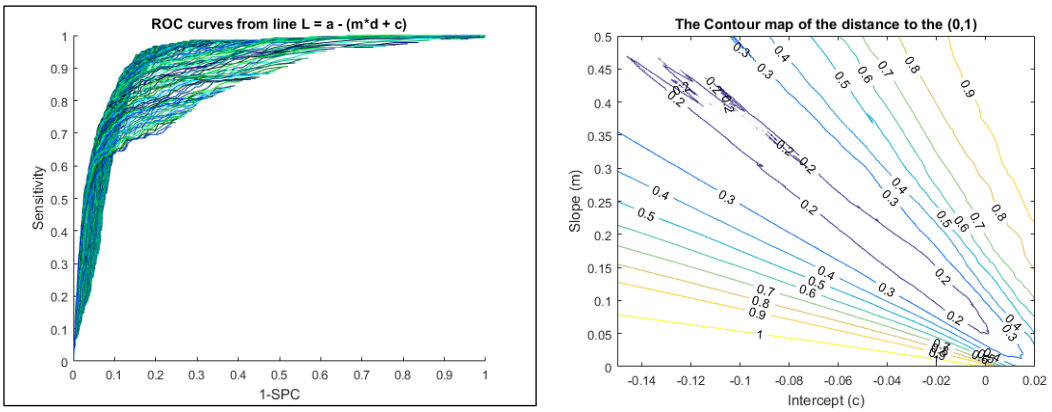
$$a = m \times d + c \qquad \textit{5-15}$$

Where m is the slope of the line and the c is the intercept on the a-axis. For each blob sample $s_n(d_n,\ a_n)$ ($n \in [1,4255]$, for 273 pollen blobs plus 3982 non-pollen blobs) in Figure 5.10, it is possible to calculate:

$$L = a_n - (m \times d_n + c) \qquad \textit{5-16}$$

For each sample, L indicates the side of the line where the sample is located. Using the line as a classifier, L indicates whether the sample is positive or negative, using the concept of ROC. If $L \geq 0$, the sample point is positive, and if $L < 0$, it is negative. The pollen blobs should be in the positive region and the non-pollen blobs should be in the negative region. This should produce higher sensitivity and specificity values than using the single threshold in the previous section. For the straight line (equation 5-15), slope values m from 0.0 to 0.5 and intercept values c from -0.15 to 0.02 were tested. An optimal line was searched for in the range of $m \in [0.0, 0.5]$ and $c \in [-0.15, 0.02]$. Each test line produces a point in the ROC space. ROC curves were produced for each different c value, with each point on the curve being a different m value. The optimal values of these two parameters produce a point that is the nearest to the perfect classifier (0, 1) on the ROC space. The experiment uses 501 different values for both m and c to get 251001 (501×501) ROC points. The ROC curves are shown in Figure 5.11 (a).

In Figure 5.11 (a), there are 501 ROC curves. Each curve corresponds to a different value of intercept c. Each curve includes 501 ROC points which correspond to 501 different slope m values. These curves had one point in the ROC space which was the shortest distance to the perfect classifier (0,1). There are 251001 points in the ROC space corresponding to 251001 different lines that have different values of slopes m and integers c. These points in the ROC space have different distances to the perfect classifier (0, 1). These distances are displayed in Figure 5.11 (b) as a contour map. The axes on this map are the intercept c and slope m. From the contour map, it can be seen that the shortest distance is less than 0.2.



(a) (b)

Figure 5.11 The process of finding optimal straight line.
Notes: (a) The ROC curves from different slopes and intercepts. (b) The contour map of distance between sample point and perfect classifier (0, 1) in ROC space.

Table 5-5 The result of one straight-line method

SES	1-SPC	Distance	Intercept (*c*)	Slope (*m*)	TP	FN	FP	TN
0.9011	0.1253	0.1596	-0.0205	0.1310	246	27	499	3483

Table 5-5 displays the result of this **one straight-line method**. The ROC point which has the shortest distance is the (0.1253, 0.9011). The distance to the perfect classifier (0, 1) is 0.1596. The optimal straight line is $L_{op} = \mathrm{a} - (0.1310 \times \mathrm{d} - 0.0205)$. This line is drawn on the (a-d) space in Figure 5.12. The blue line in this figure is the optimal line. Points to the top-left of the line are positive results and points to the bottom-right of the line are negative results. This line separates most of the pollen blobs from many of the non-pollen blobs. In addition, comparing to Figure 5.9, this straight line increases sensitivity from 0.886 to 0.9011 and improves specificity from 0.794 to 0.8747. It can be seen that this method is better than using only distance as a classifier.

Table 5-5 also shows the four ROC outcomes of the one straight-line method. The FP number is 499. Compared to Table 5-4 with FP of 821, this method almost halves the number of false positives (FP), while slightly increasing the number of true positives. This result shows that the straight-line method is better than using only position distance as a single classifier.

However, Figure 5.12 shows a limitation of the one straight-line method. The positive region still includes many non-pollen blobs that are concentrated near the bottom-left corner of the figure, close to the origin. These non-pollen blobs can be excluded using another straight-line that has a slope less than zero. This line can cross the first line on the d-axis. This crossing point on the d-axis will be denoted as d_c. Both lines cross this point. Therefore, these two lines can be defined using the intercept d_c and the reciprocals of their slopes.

$$L_1 = a_n - \frac{d_n - d_c}{m_1} \ (m_1 > 0, d_n \geq d_c) \qquad 5\text{-}17$$

$$L_2 = a_n - \frac{d_n - d_c}{m_2} \ (m_2 < 0, d_n < d_c) \qquad 5\text{-}18$$

Here $1/m_1$ and $1/m_2$ are the slopes of the two lines. The positive region has $L_1 \geq 0$ and $L_2 \geq 0$ on the plane of the 2-D distribution. The rest of the plane is the negative region. This is known as the "**two lines method**" that is used to calculate ROC outcomes for pollen and non-pollen blobs.

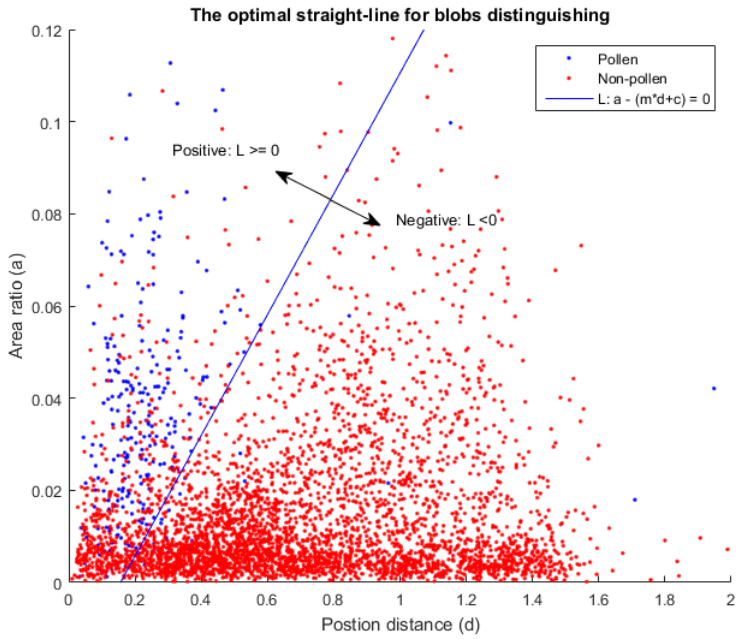


Figure 5.12 The optimal straight line for classifying blobs.

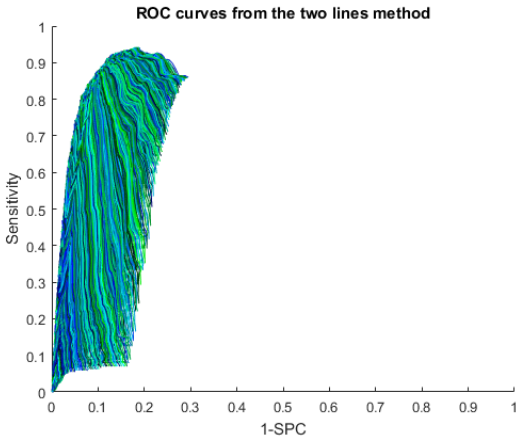


*Figure 5.13 The ROC curves of **two lines method***

From Figure 5.12, the ranges of the three parameters can be: $\mathrm{d_c} \in [0, 0.4]$, $\mathrm{m_1} \in [0.01, 11]$ and $\mathrm{m_2} \in [-20, -0.01]$. For each parameter, 51 thresholds were used, ranging from between these minimum and maximum values. The ROC curves are shown in Figure 5.13. There are 2601 curves in the ROC space corresponding to different values of $\mathrm{m_1}$ and $\mathrm{m_2}$. Each curve includes 51 ROC points corresponding to the 51 different values of $\mathrm{d_c}$. It was found that one point had the shortest distance to the perfect classifier (0, 1).

Because the **two lines method** used three parameters, it is not possible to show the contour plot of the distance from the perfect classifier. The experiment calculated the shortest distance point in the ROC space as (0.1185, 0.9121). The results are shown in Table 5-6.

*Table 5-6 The result of **two lines method***

SES	1-SPC	Distance	m_1	m_2	d_c	TP	FN	FP	TN
0.9121	0.1185	0.1476	6.8238	-16.002	0.208	249	24	472	3510

From Table 5-6, the shortest distance between the optimum point and the perfect classifier (0, 1) is 0.1476, which is shorter than the result in Table 5-5. In addition, the sensitivity and specificity have slightly improved. The two optimal line formulas are shown below.

$$L_{1op} = a_n - \frac{d_n - 0.208}{6.8238} \quad (d_n \geq 0.208) \qquad 5\text{-}19$$

$$L_{2op} = a_n - \frac{d_n - 0.208}{-16.002} \quad (d_n < 0.208) \qquad 5\text{-}20$$

The two lines are drawn on the a-d distribution plot in Figure 5.14 using blue and green colours respectively. The two lines cross the d-axis at 0.208. The positive region is shaded in the figure. It can be seen that the positive region covers most of the pollen blobs, and the negative region includes most non-pollen blobs and few pollen blobs.

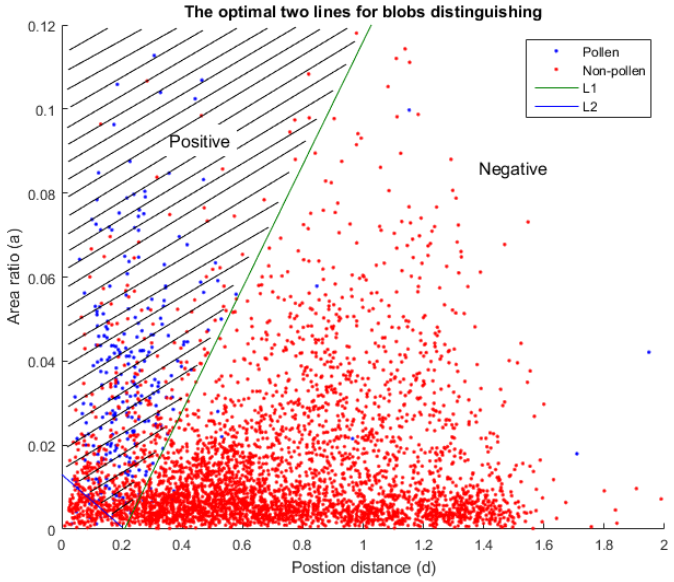


Figure 5.14 The optimal two lines for pollen detection from non-pollen blobs.

Table 5-6 also displays the four outcomes using this classifier. The number of true positives (TP) is 249, which slightly increases from Table 5-5. In addition, the false positives are 472, which is less than in Table 5-5. The **two lines method** is better than the **one straight-line method**, although the improvement is not significant.

5.1.4.4 The estimation for decreasing false positive (FP)

From the last section, the **two lines method** uses the "Area ratio" and "Position distance" to detect pollen blobs. This method produces the four outcomes as shown in Table 5-6. According to the table, the sensitivity and specificity are both high. However, the detection has 249 true positives (TP), but also has 472 false positives (FP). The TP in the total number of positive outcomes is only 35%. The sum of positive outcomes is the measurement of pollen blobs in all of the blobs. This low precision (precision is defined in section 3.8) indicates the measurement is not accurate enough. It is necessary to improve this estimation.

After applying the **two lines method**, the two results are the measured positive (M_P) and measured negative (M_N) values.

$$M_P = T_P + F_P \quad \textit{5-21}$$

$$M_N = T_N + F_N \quad \textit{5-22}$$

Where T_P, F_P , T_N and F_N are the four outcomes of true positive (TP), false positive (FP), true negative (TN) and false negative (FN) respectively. These four values are still unknown after the measurement. However, if the sensitivity (S_N) and the specificity (S_P) were known, then TP, FP, TN and FN could be calculated.

Sensitivity and specificity are defined as:

$$S_N = \frac{T_P}{T_P + F_N} \quad \textit{5-23}$$

$$S_P = \frac{T_N}{T_N + F_P} \quad \textit{5-24}$$

The four outcomes can be calculated back by using the four formulas above. From equation (5-23):

$$S_N F_N = (1 - S_N)\, T_P \quad \textit{5-25}$$

And from equation (5-24):

$$S_P F_P = (1 - S_P)\, T_N \quad \textit{5-26}$$

Multiplying equation (5-21) by S_P and equation (5-22) by S_N these become:

$$M_P S_P = T_P S_P + F_P S_P \quad \textit{5-27}$$

$$M_N S_N = T_N S_N + F_N S_N \quad \textit{5-28}$$

Substituting equation (5-26) into equation (5-27):

$$M_P S_P = T_P S_P + (1 - S_P)\, T_N \quad \textit{5-29}$$

Substituting equation (5-25) into equation (5-28):

$$M_N S_N = T_N S_N + (1 - S_N)\, T_P \quad \textit{5-30}$$

After solving equations (5-29) and (5-30), the four outcomes can be estimated as:

$$T_P = \frac{S_N S_P M_P - (1 - S_P) M_N S_N}{S_N S_P - (1 - S_P)(1 - S_N)} \quad \textit{5-31}$$

$$T_N = \frac{S_N S_P M_N - (1 - S_N) M_P S_P}{S_N S_P - (1 - S_P)(1 - S_N)} \quad \textit{5-32}$$

$$F_P = \frac{(1 - S_P)\, T_N}{S_P} \quad \textit{5-33}$$

$$F_N = \frac{(1 - S_N)\, T_P}{S_N} \quad \textit{5-34}$$

The above equations from (5-31) to (5-34) are the estimated four outcomes. The final estimated number is equal to:

$$P_E = T_P + F_N \quad \textit{5-35}$$

$$N_E = T_P + F_N \quad \textit{5-36}$$

where $\mathrm{P_E}$ and N_E are the estimated numbers of positives and negatives respectively. This method could give a better estimate of the number of positives. For the **two lines method**, the sensitivity and specificity are obtained from the ROC analysis of the samples. To calculate the number of pollen blobs in a new sample of blobs, firstly, the **two lines method** is used to measure the positive and negative values. After that, these measured positive and negative values are used with the sensitivity and specificity from the original sample to calculate better estimates of the actual numbers of positives and negatives. If the sensitivity and specificity are similar in both samples, the estimated positive and negative numbers should be similar to the actual numbers in the second sample.

5.1.5 Pollen measurement on images.

The whole pollen measurement method is shown in Figure 5.15. The individual bee images are cropped from the video frames, and then they are analysed using image moments to detect the main bodies of the bees. The main body is the elliptical shaped central body of the bee. It mainly includes the head and abdomen. The main body ellipse is then removed from the bee image. The background is removed from the individual bee image by using the individual bee blob from the binary image. Next, the pollen colour (orange and white) is detected. As a result, many small blobs are detected on the image. These blobs include pollen and non-pollen blobs. The two lines method uses the features of "Area ratio" and "Position distance" to measure the number of pollen and non-pollen blobs. The measured result and the expected sensitivity and specificity are used to estimate a more accurate number of pollen blobs.

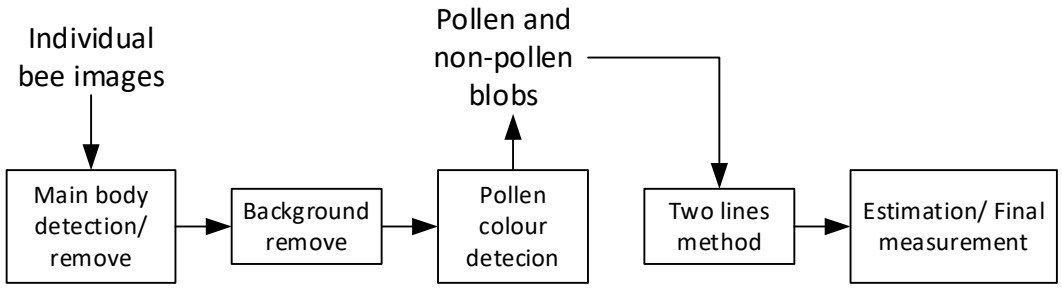


Figure 5.15 The pollen measurement progress

The estimation step is based on the sensitivity (0.9121) and specificity (0.8815) produced by the two lines method. This sensitivity and specificity were calculated from a sample of individual bee images from 400 video frames.

Another 100 frames were used to test the estimation. These are from the same video as the sample collection. There are 64 pollen images from 5 bees and 350 non-pollen images from 28 bees in these 100 frames. The main bodies of these bees are detected and removed from these individual bee images. Thus, the other parts of the bees' bodies are left. The pollen colour detection then detects many blobs by using the colour of pollen sacs. As a result of this method, 1293 blobs are detected. There are actually 73 pollen blobs and 1220 non-pollen blobs. The **two lines method** is used to distinguish the pollen and non-pollen blobs and to measure the number of them. Finally, the estimation method uses the sensitivity of 0.9121 and specificity of 0.8815 to estimate the final value of the number of pollen blobs. This result is shown in Table 5-7.

Table 5-7 The pollen estimation result from the test

Actual condition		Measured condition		Final result	
Pollen blobs	Non-pollen blobs	Measured positive	Measured negative	Estimated positive	Estimated negative
73	1220	201	1092	60	1233

In Table 5-7, the actual number of pollen and non-pollen blobs are 73 and 1220 respectively. The "Measured condition" is the result after applying the **two lines method**. The "Measured positive" number is 201 which is much greater than the actual number. The "Measured negative" number is 1092, which is smaller than the actual number of non-pollen blobs. The estimation produces the final result. The "Estimated positive" is the final estimated number of pollen blobs. The number is 60, which is closed to number of actual pollen blobs (73). The "Estimated negative" is the final estimated result of non-pollen blobs, which is 1233.

The estimated positive and negative numbers are both similar to the "Actual condition" pollen and non-pollen blob numbers. This is the final result of the test. This result shows the whole method for pollen measurement on the pollen and non-pollen blobs is reliable on the same video of different continuous frames.

However, the estimation has a limitation. Therefore, this method relays on the sensitivity and specificity keeping similar between analysis and test. The sensitivity (0.9121) and specificity (0.8815) are calculated from the sample of blobs in section 5.1.4.3. To apply this on the video is a problem, because the situation of measuring pollen on the video is different from the sample of blobs. It is to count number of pollen carrying bees rather than number of pollen and non-pollen blobs. The sensitivity and specificity of pollen measurement on the video may be different from the sample blobs. In addition, in different videos, the sensitivity and specificity may also differ, in which case the estimation will be incorrect. Therefore, this estimation is not used for the next step. With the possible exception of the "Estimation" step, the pollen measurement process from "Main body detection" to "Two lines method" can be applied for pollen carrying bee counting on bee monitoring video.

5.1.6 Pollen measurement on the bee monitoring video

The previous section showed the model for pollen measurement. Except for the "Estimation" step, other processes in the model can be applied on the video to count pollen. On the video, the purpose of the pollen counting is to estimate the number of pollen carrying bees.

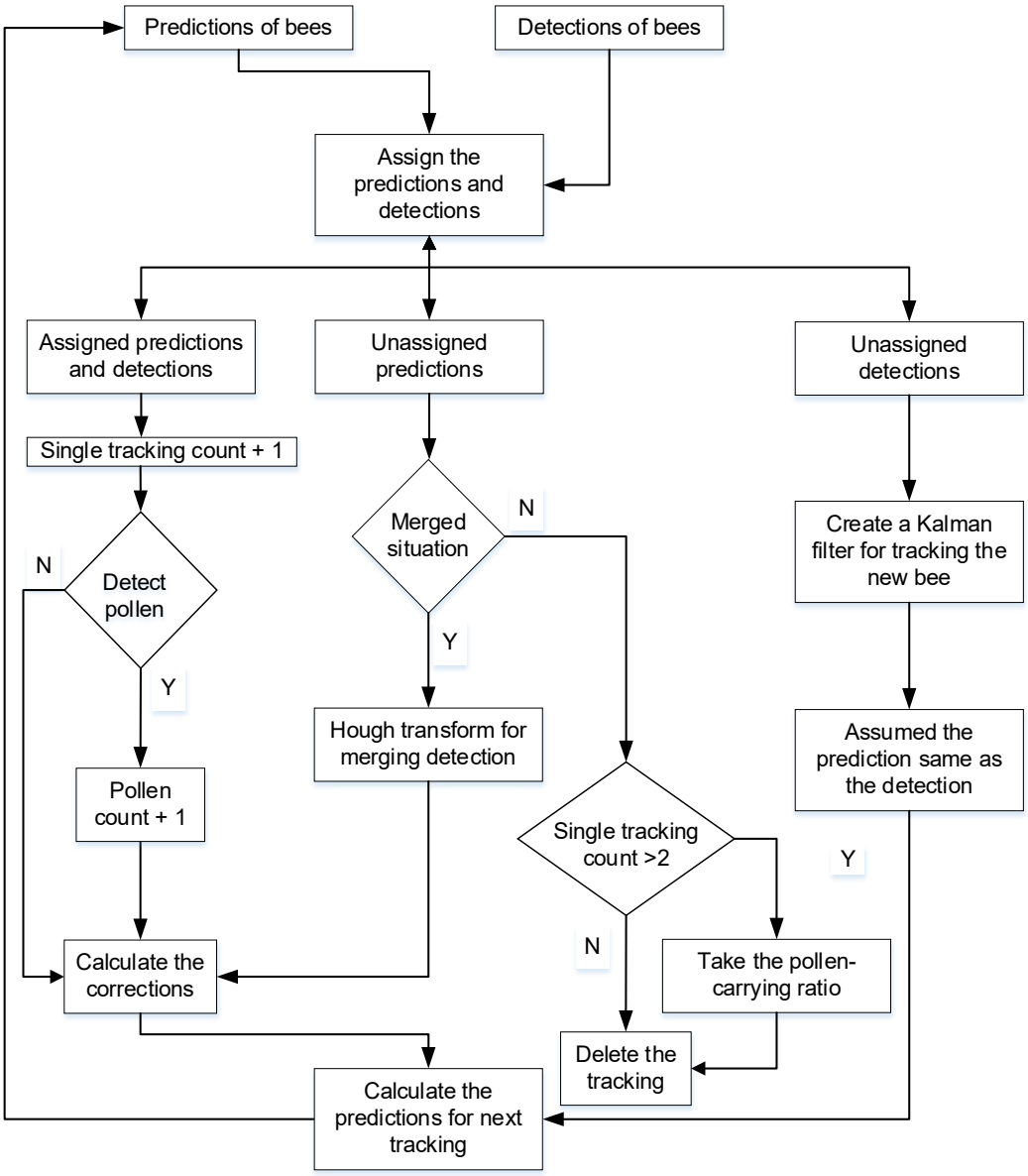


Figure 5.16 The whole model of the pollen measurement on the video

Figure 5.16 shows the model for pollen counting on the video. The main structure is based on the bee-tracking model. If a bee is tracked, the model records the number of frames in which the bee individually appears on the video. This is the **single tracking count**. The bee tracking avoids multiple counting of pollen sacs, because the pollen sacs are detected on the individual bee images on each frame of the video. During tracking, the model identifies individual bee images and detects the pollen they are carrying. As the pollen detection model cannot find the pollen in the merged situation, these images are not processed for pollen counting. In each frame of the video, once a bee's individual image is detected including pollen, the model increases the count of pollen detection on that individual bee. Because the bee is tracked on the video, the **pollen count** can be incremented each time pollen is detected on that bee in a frame. When the bee disappears from the video, the model calculates the ratio of the **pollen count** to the **single tracking count**. This **pollen carrying ratio** estimates the probability of the bee having pollen. However, if an individual bee fully appears on only one or two frames, the ratio is not meaningful. Therefore, pollen is not counted on a bee appearing in fewer than three frames.

This **pollen carrying ratio** can be used to decide whether the bee has pollen and also helps to measure pollen on the video. The threshold of the ratio is chosen by the ROC curve. In the first 500 frames of the sample video, 16 bees are carrying pollen sacs and 85 bees do not have pollen. All of these bees appear individually on more than two frames. The pollen-carrying ratio from each of these bees is used to draw a ROC curve with 20 thresholds from 0 (0%) to 1 (100%), to find the optimum threshold percentage to identify a bee as pollen-carrying. The ROC curve is shown in Figure 5.17.

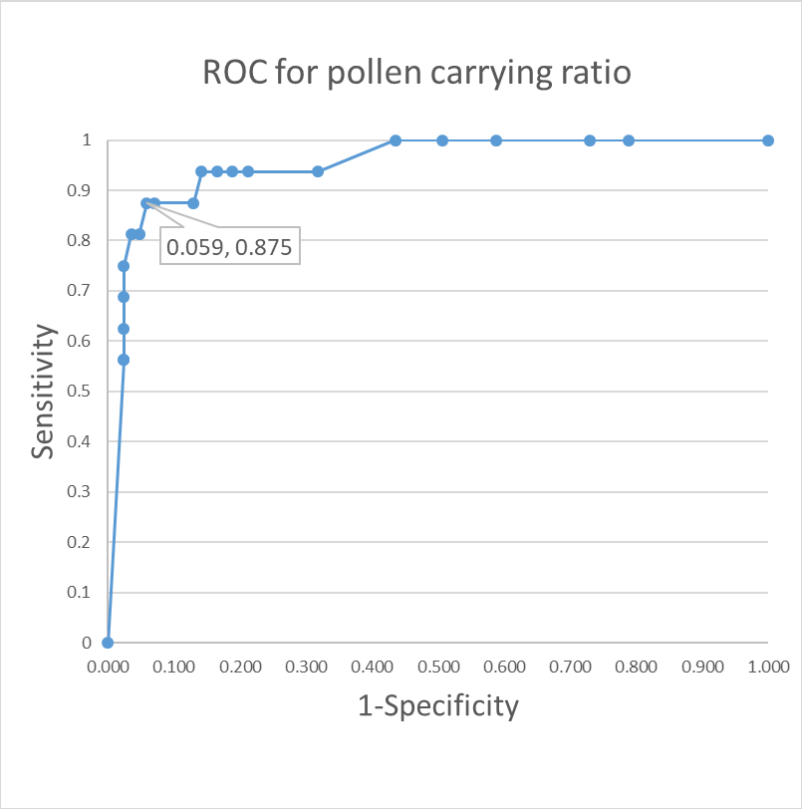


*Figure 5.17 The ROC curve for the bee **pollen carrying ratio***

In Figure 5.17, the point in the ROC space which is closest to the perfect classifier (0,1) has a value of 1-SPC equal to 0.059 and a sensitivity of 0.875. The distance between the point (0.059, 0.875) and perfect classifier (0,1) is 0.1381. The optimal threshold is 0.65. Therefore, a bee that has a **pollen carrying ratio** greater 0.65 is counted as carrying pollen sacs.

The results for the 16 pollen-carrying bees and 85 non-pollen bees are shown in Table 5-8.

*Table 5-8 The ROC result with the **pollen carrying ratio** threshold*

Threshold	Sensitivity	Specificity	Pollen bees	Non-pollen bees	TP	FN	FP	TN
0.65	0.875	0.941	16	85	14	2	5	80

From Table 5-8, the number of positive results is 19 (14+5). This means that the model measured 19 bees carrying pollen sacs during the first 10 seconds (500 frames) of the video. The actual number of pollen carrying bees was 16. The measured result is close to the actual number. Following this measured result, more video frames were used to test the pollen counting model. This will be shown in the next chapter.

In summary, when a bee appears on the video, the pollen-measurement model checks whether the bee appears in more than 2 frames on the video. After that, the model checks the **pollen carrying ratio**. If the ratio is greater than 0.65 (65%), the bee is identified as carrying pollen sacs. The model measures the number of bees carrying pollen on the video with this process. However, the sample simulation above has not considered the tracking problem. The failure of bee tracking may affect the result of pollen measurement on the video. In section 6.1.2, the first minute (first 3000 frames) of the sample video is used to test the pollen measurement model on monitoring video.

Furthermore, bees that are carrying pollen and going into the beehive are counted when they cross the tracking boundary. The tracking boundary was introduced in Figure 4.1 in section 4.1. This approach may not be perfect. Bees may not go into the hive after crossing the boundary. In addition, bees may fly back from the boundary. However, these two situations are probably infrequent, because the tracking boundary is close to the actual entrance of the beehive. If a bee crosses the tracking boundary and disappears from the tracking model, the pollen-measurement model calculates the **pollen-carrying ratio** of this bee. If the ratio is high enough (more than 65%), the pollen is counted as going into the beehive. The test of this is also shown in section 6.1.2.

5.2 Deep learning for pollen sacs detection and measurement

This section will introduce deep learning methods to detect and count pollen sacs. There are several deep networks which can be used for multiple object detection on a single image, which including region-based convolutional neural network (RCNN) (Girshick et al., 2014), fast region-based convolutional neural network (Fast RCNN) (Girshick, 2015), Faster-RCNN (Ren et al., 2015), Mask RCNN (He et al., 2017), you only look once (YOLO) (Redmon et al., 2016) and single shot MultiBox detector (SSD) (Liu et al., 2016). The pollen sacs on the video are very small on the back legs of flying bees. The old version of YOLO is not suitable for this small object detection (Redmon et al., 2016). The new version (YOLOv3) (Redmon and Farhadi, 2018) has improved smaller object detection, and it has been applied in Linux computer, but it has a software conflict in windows. SSD and Mask RCNN have been applied well on CAFFE and TensorFlow, but MATLAB application is not stable. The bee tracking model in this research is built with MATLAB, so it is better to find a method which is easy to apply on MATLAB, so that the pollen detection model can be combined into the bee tracking model for a final test. RCNN and FRCNN and Faster RCNN have been built into MATLAB. Faster RCNN is the latest version among these three methods. It is also easier to apply in MATLAB. In this research, faster region-based convolutional neural network (Faster RCNN) is utilized for pollen detection. The software tool is MATLAB with neural network tool box, using MATLAB version 2018a. MATLAB includes the function of “trainFasterRCNNObjectDetector()” which can train a faster RCNN network. The computer information for training and detection is:

CPU: Intel Core i7-6700, 8 cores, 3.4GHz

RAM: 32.0 GB

GPU: NVIDIA GeForce GT 730

GPU RAM: 2.0GB

Operating system: Windows 10

The computer is slow for current deep learning training. Because of this hardware limitation, the pretrained network is used in this research. VGG-16 pretrained network is chosen as the core network for the Faster RCNN network. This network includes 16 convolutional layers. The experiment shows the GPU RAM of the training computer is not enough for more network layers, such as VGG-19 (19 layers), GoogLeNet (22 layers), ResNet-50 (50 layers) and ResNet-101 (101 layers). Pollen bee images are collected from two recorded videos for transfer learning.

5.2.1 Training and validation data

The honey bee monitoring video has resolution of 1980x1280, but the pollen sacs on bee's back legs have a resolution from 15x15 to 40x40, which is very small on the full video frame. The individual bee images are firstly collected, then pollen sacs are detected from the smaller bee images. An individual bee image has a resolution from 100x100 to 200x200. Therefore, it is easier to detect pollen sacs from the individual bee images. The bees are detected from video frame with colour and motion, which is introduced from section 4.2. The individual bee images are cropped from full video frames for data collection.

The Faster RCNN and VGG-16 networks are used to detect pollen sacs on the individual bee images. Therefore, the individual bee images which include pollen sacs are collected for training. Two videos are used for data collection. One video (training video 1) is recorded from 11 am in the morning, the other video (training video 2) is recorded from 1pm on another day. This enables network training for image under different light condition. (Training video 1 is the same video which provided the first 400 frames used for collecting data for the **two lines method**.) In training video 1, 700 individual bee images are collected from 2000 frames. All these images include pollen sacs. These are the "pollen bee images". These 700 images are randomly separated to two groups: 600 images for training, and 100 for validation. There are also many individual bee images which do not have pollen sacs in the 2000 frames. These are the "non-pollen bee images". 100 non-pollen bee images are randomly collected from the 2000 frames, which are also used as validation data. In training video 2, 500 pollen bee images are collected from 2000 frames. These 500 images are also randomly separated to two groups: 400 images for training, and 100 for validation. Another 100 non-pollen bee images are randomly collected from the 2000 frames in this video for validation. The overview of all data is shown in Table 5-9.

Table 5-9 The detail of data collection

Training Videos No.	Frames	Training data	Validation data	
		Pollen bee images	Pollen bee images	Non-pollen bee images
1	2000	600	100	100
2	2000	400	100	100
Total		1000	200	200

All the training data from the two videos are combined for Faster RCNN network training. These 1000 pollen bee images are labelled pollen sacs using the MATLAB “Image Labeller” APP. Figure 5.18 shows an example of pollen sac labelling. The pollen sac region-of-interest (RoI) is labelled on pollen bee images with a rectangle. Then, the non-RoI part on the images is the background in a Faster RCNN network.

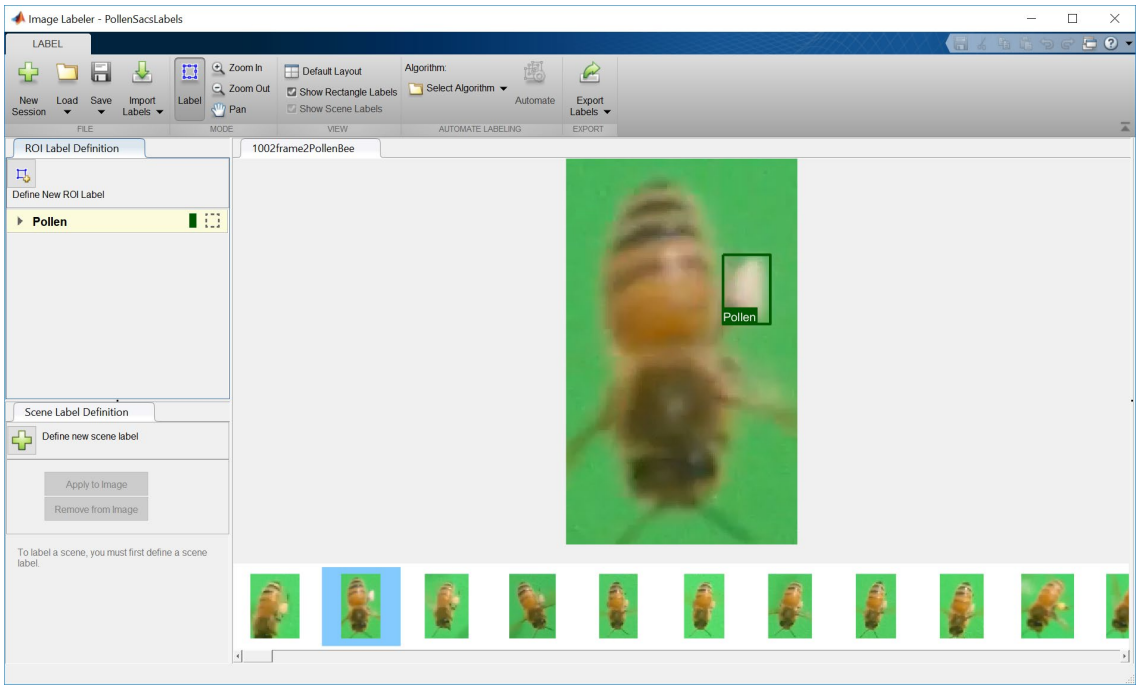


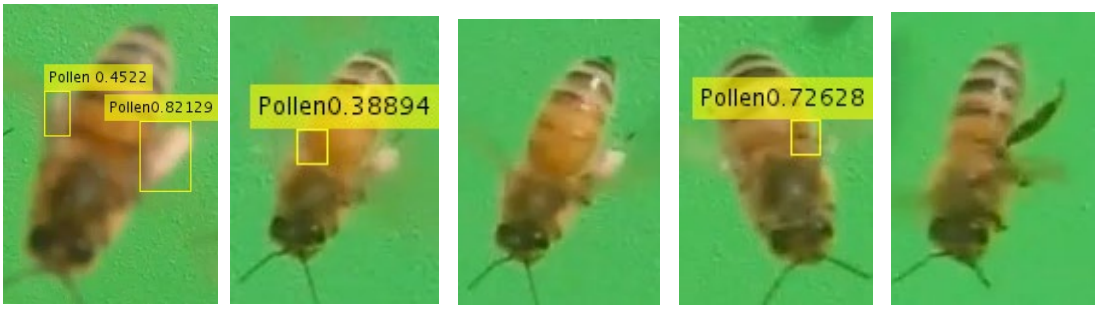
Figure 5.18 An example of pollen sacs labelling in pollen bee images.

After these 1000 training images are labelled manually, these images and labels are also flipped horizontally for data augmentation. This is because there are many images showing pollen sacs on the left side of bees, but few images have pollen sacs on the right side of bees. More data augmentation can also be used, such as rotation, resizing and modified brightness and contrast. However, because of the hardware limitation, more data augmentation will require a very long time to train the network. In this research, most bees fly face down to the beehive entrance on the video. Therefore, only flipped horizontal images are used for data augmentation. As a result, there are 2000 pollen bee images with labels for training.

5.2.2 The training, pollen detection and pollen and non-pollen bee image identification

The VGG-16 pretrained network is used as the core network for Faster RCNN network training. The Faster RCNN network is trained using the 2000 labelled pollen bee images. These training uses 50 epochs. A MATLAB function randomly shuffled the images. The detection overlap with ground truth is greater than 0.5. The training time of each network is about 24 hours on the computer. An important training operation parameter is learning rate. Different learning rates give different detection results. The learning rate choice depends on the detection results. This will be introduced in section 5.2.3.

The training produces a network for pollen detection on individual bee images. The network is applied for pollen detection on the 400 validation images (200 pollen bee images and 200 non-pollen bee images). The network detected pollen on the validation images. Some examples of results are shown in Figure 5.19.



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

Figure 5.19 Examples of pollen detection on images.
Notes: these five images are detected pollen with the same trained network. (a)-(c) The pollen bee images detection results. (d) and (e) The non-pollen bee images detection results.

In Figure 5.19, there are five images showing different pollen detection results. The network produces bounding boxes to mark the detection of pollen sacs on each image. Each bounding box has label of "Pollen" and a number. The number represents a probability (or confidence score) of it actually being a pollen sac. For example, in image (a), the 0.82129 of the bigger bounding box means the network identifies that this RoI bounding box contains a pollen sac with about 82.1% probability. A threshold of this confidence score needs to be chosen to filter out low score detections. If the threshold is too small, many wrong detections will be shown on the image.

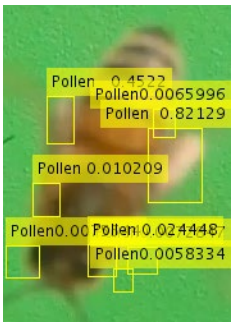


Figure 5.20 An example of detection with 0.005 threshold of detect score.

Figure 5.20 is an example with the threshold score of 0.005. It can be seen that the two pollen sac bounding boxes have scores 0.4522 and 0.82129 respectively. Other bounding boxes have very small scores. They are not pollen sacs. Therefore, the threshold should not be too low. However, if the threshold is too high, only few pollen sacs will be detected on the images. The results in Figure 5.19 are obtained with threshold of 0.36.

After this pollen detection, a bee image can be identified as to whether it has pollen or not. A bee carrying pollen can show one or two pollen sacs on the image. Therefore, if an image is detected having at least one pollen sac, this image is a positive result, which is identified as a pollen bee image. In Figure 5.19, images (a), (b) and (d) are positive results. Image (a) is a true positive result. Image (b) is also a true positive result, although the detection does not detect the pollen

location correctly. Image (d) is a false positive result, because this image does not have any pollen sac. Images (c) and (e) are negative results, which are identified as non-pollen bee images. The network does not detect anything on both images, but image (c) has a pollen sac. Therefore, image (c) is false negative result. Image (e) shows a true negative result, because this image does not have pollen sac.

The threshold of the bounding box confidence score relates to the pollen and non-pollen bee image identification. A lower threshold will cause more positive results, which increases true positives among the pollen bee images, but it will also increases the false positive results among the non-pollen bee images. A higher threshold will cause more negative results, which increases true negative results among non-pollen bee images, but it will produce more false negative results among the pollen bee images. A confidence threshold should be chosen which can produce a good balance of sensitivity and specificity for the identification of pollen and non-pollen images.

A ROC curve can be applied to see the effect of choosing different thresholds. 100 thresholds (from 0 to 1) are chosen to produce the ROC curve. The trained network is applied on the 400 validation images with the different thresholds. This produces different results for sensitivity and specificity. The detail of ROC curve and the learning rates chosen will be shown in the next section.

5.2.3 Training repetition and learning rate choosing

Section 5.2.2 shows the process of training, pollen detection and pollen and non-pollen images identification. Training Faster RCNN with different learning rates produces different results. When the learning rate is changed, the process of section 5.2.2 should be repeated. The results of the ROC curves can help choose a good learning rate. The learning rate trialled were 0.01, 0.001, 10^{-4} and 10^{-5}. Each learning rate is used for Faster RCNN network training with 2000 training images, then a network is produced for pollen detection. Then pollen and non-pollen bee identification is applied on the 400 validation images. With 100 different score thresholds from 0 to 1, a ROC curve can be produced to show the performance of the network. The whole learning rate and network model choice process is shown in Figure 5.21 below:

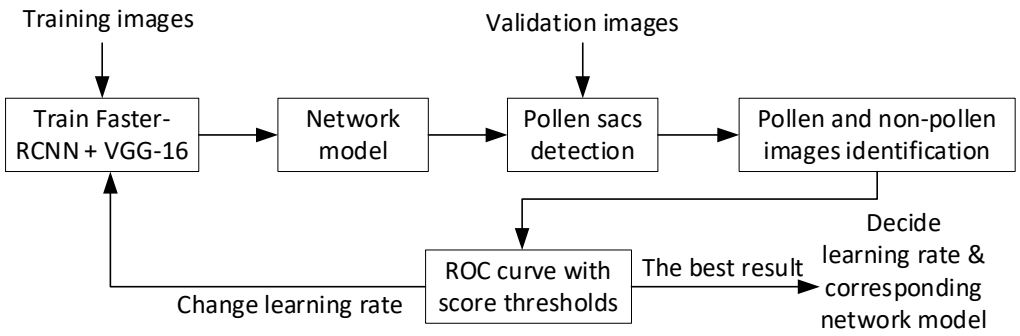


Figure 5.21 The training process for finding a good learning rate and good network model

In Figure 5.21, a learning rate and other training operations are decided initially, the training images are input to the Faster RCNN + VGG-16 network for training. After the training, a network model for pollen sac detection are produced. The network model is utilized to detection pollen sacs on the validation images. The results are used to identify pollen and non-pollen bee images.

Then the ROC curve is built with a range of confidence score thresholds. Different learning rates produce different ROC curves. ROC curves can be compared using the method from section 3.8. The best curve corresponds to the best learning rate among all the trialled learning rates. Then, the corresponding network of the best learning rate is chosen as the pollen sac detection model.

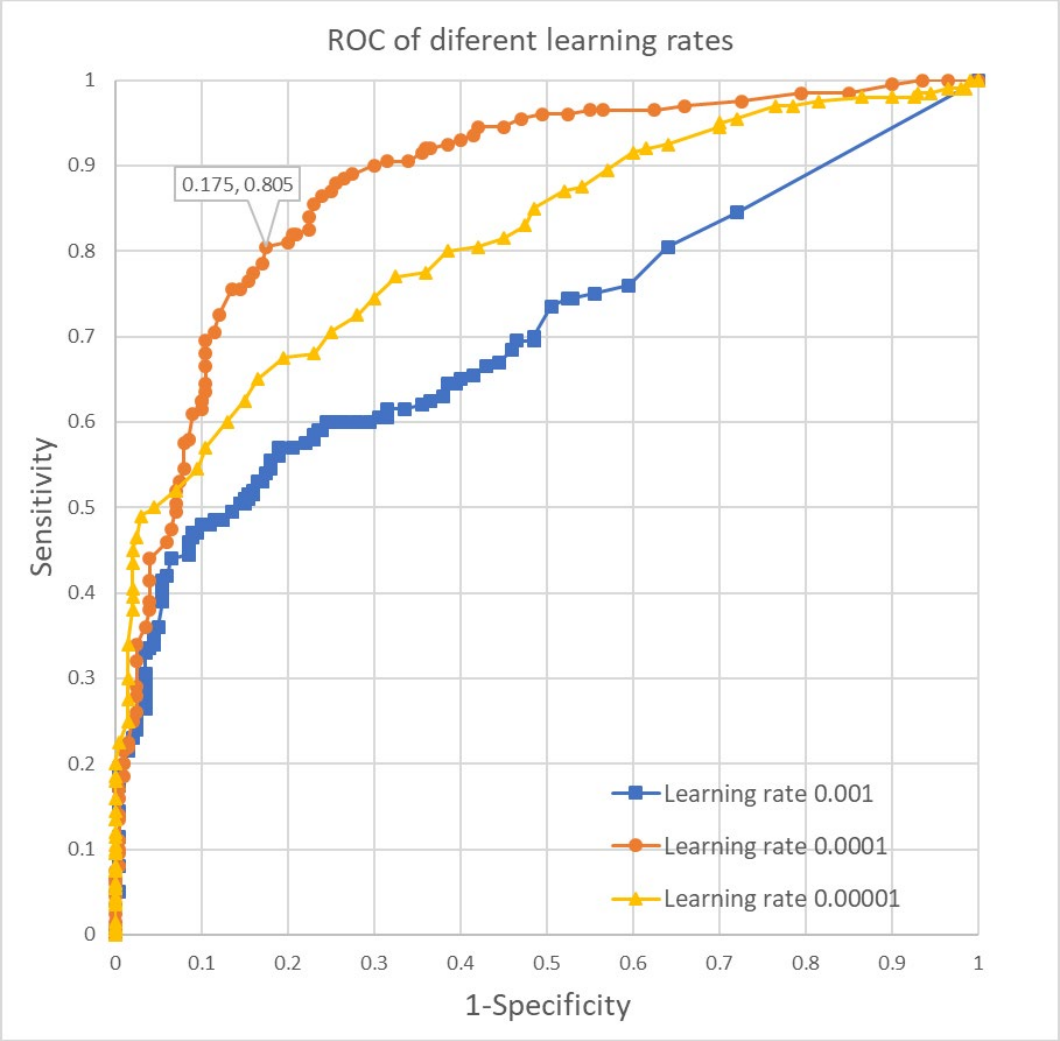


Figure 5.22 ROC curve of three learning rate result.

In the experiment, the learning rate of 0.01 cannot detect pollen sacs on any validation images. However, the other three learning rates have better results. The ROC curve results for these three networks (from the three learning rates) are shown in Figure 5.22.

From Figure 5.22, the learning rate of 10^{-4} produces a better ROC curve than other two learning rates. The best learning rate in this situation is 10^{-4} . However, there may be a better learning rate between 0.001 and 10^{-5}. Therefore, two more learning rates which are close to 10^{-4} are chosen to check whether a better result can be found. They are $1.2*10^{-4}$ and $0.8*10^{-4}$. ROC curves from these two learning rates are compared to the ROC curve from 10^{-4} . The result is shown in Figure 5.23. The figure shows that a learning rate 10^{-4} still give the best result. The other two learning rates produce very similar results, but they are worse than the learning rate 10^{-4}. As a result, the learning rate of 10^{-4} is chosen. The corresponding network will be used for pollen detection model.

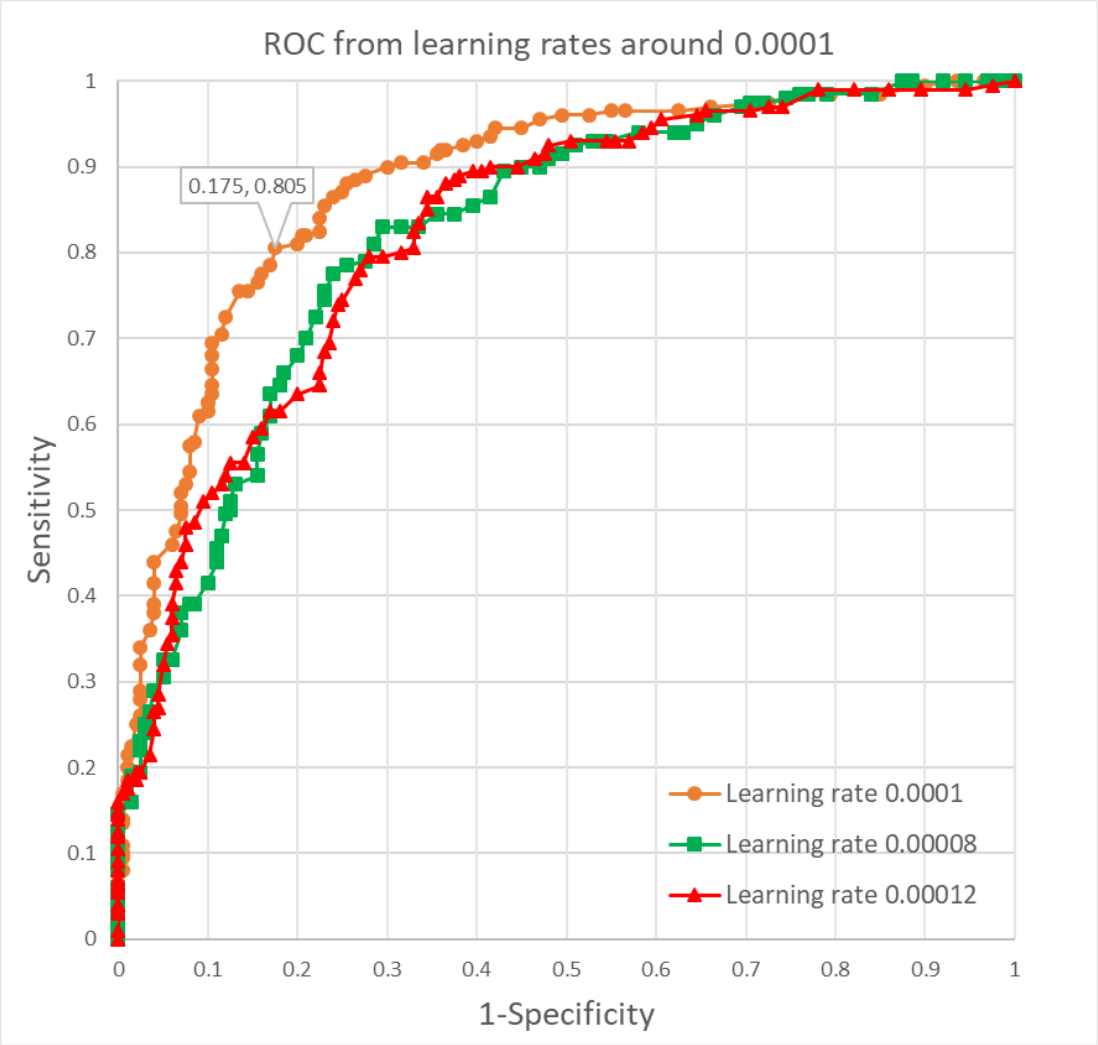


Figure 5.23 ROC curves of learning rate 10^{-4}, and two other learning rates beside.

In addition, the ROC curve of learning rate 10^{-4} is smooth. The point on the curve which is closest to the point (0, 1) is (0.175, 0.805). Table 5-10 displays this point's information. This point leads to a threshold of confidence score. In addition, this point also satisfies the Youden index, which is the maximum value of senstivity+sepecificity-1 (Subtil and Rabilloud, 2015). Therefore, a threshold for detection is chosen as 0.36.

Table 5-10 The confidence threshold information

LR	Thr	TP	FN	FP	TN	Sens	SPC	Precision	Acc	Shortest distance	Youden index
10^{-4}	0.36	161	39	35	165	0.805	0.825	0.821	0.815	0.262	0.63

LR: Learning Rate
Thr: Threshold
Acc: Accuracy

5.2.4 Pollen measurement on video frame sequences

Section 5.2.3 introduced the training process of the pollen detection model. Then, an automatic model can be used on each video frame to identify pollen and non-pollen bee images. This model is shown in Figure 5.24.

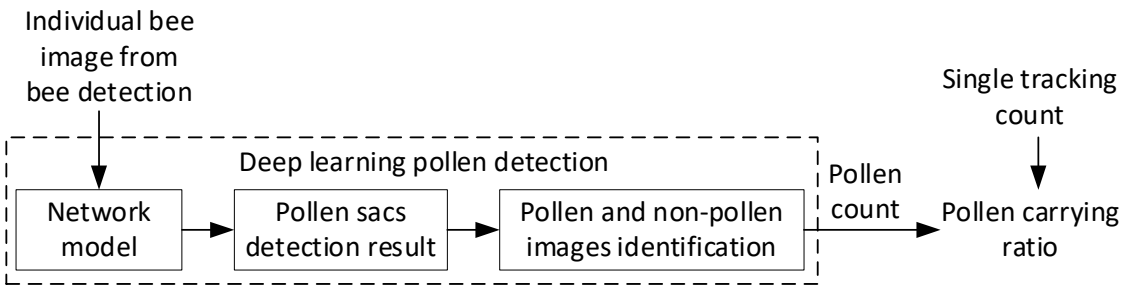


Figure 5.24 The deep learning model for pollen detection in bee tracking model.

In Figure 5.24, an individual bee image is input to the deep learning network model for pollen sac detection. The network model is the one chosen in section 5.2.3. It produces a pollen sac detection result. Then the result is used to identify whether the bee image has pollen or not. The output of the model using deep learning pollen detection is the pollen count which is combined with the single tracking count from bee tracking model to calculate the pollen carrying ratio for a bee appearing on the video. The deep learning pollen detection model is combined with the bee tracking model which as shown in Figure 5.25.

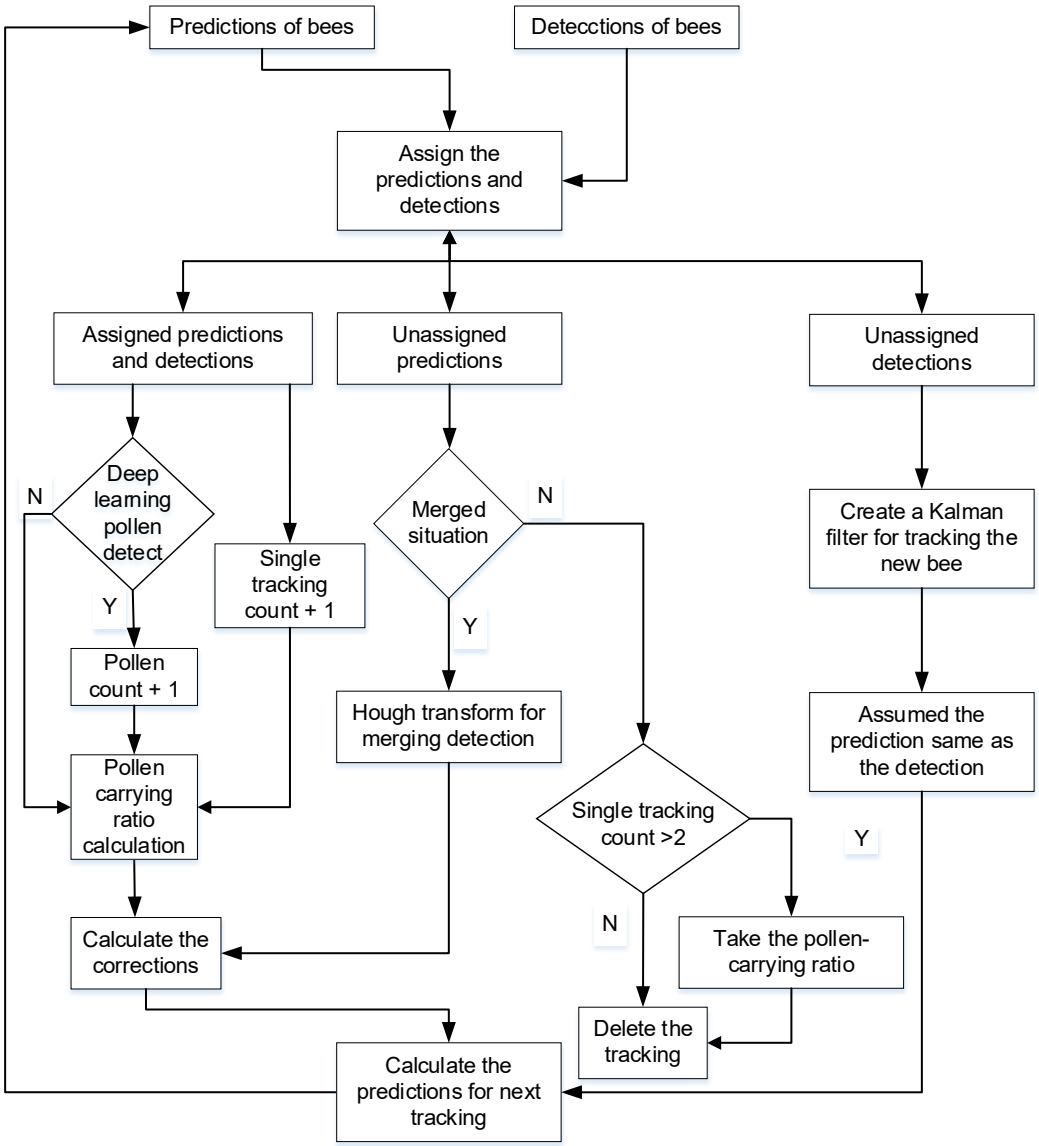


Figure 5.25 The whole pollen count model with deep learning pollen detection.

In Figure 5.25, the bee detection and tracking are similar to Figure 5.16 in section 5.1.6. The difference is that the pollen detection uses the deep learning model. Once a bee is detected flying individually (single bee), this bee image is input to the deep learning model to detect whether the bee image has pollen or not. If the individual bee image is detected having pollen, the **pollen count** is incremented. Otherwise, the pollen count is not changed from previous video frame. At the end of the bee tracking sequence, the **pollen carrying ratio** is calculated as the **pollen count** divided by a **single tracking count** from the bee tracking model.

For each bee flying on the video, the ratio calculated for the **pollen count** and the **single tracking count** is the bee's **pollen carrying ratio**. However, in this section, the pollen detection is using deep learning method, therefore the threshold of pollen carrying ratio should be recalculated.

The ROC curve is used to identify the threshold of the **pollen carrying ratio**. Because two videos are used for training the network, the ROC method is used to analyse these two videos. From training video 1, another 1000 frames were collected for this analysis. In these 1000 frames, 31 bees are carrying pollen sacs and 194 bees do not have pollen. In training video 2, another 2000 continuous frames were collected. In the 2000 frames, 58 bees are carrying pollen and 210 bees do not have pollen sacs. All of the bees from these two videos appear individually on more than two frames. The video information is shown in Table 5-11.

*Table 5-11 Two videos' new frames for **pollen carrying ratio***

Training videos No.	Frames	Pollen bees	Non-pollen bees	Flying frames
1	1000	31	194	>2
2	2000	58	210	>2

The **pollen-carrying ratio** for each of bee is used to draw a ROC curve with 100 thresholds from 0 (0%) to 1 (100%) for the two videos separately, to find a good threshold which can identify a bee as pollen-carrying. If the ratio is greater than the threshold, the bee is identified as carrying pollen, otherwise it is not. Figure 5.26 displays the ROC curve of training video 1. On this curve, the point in the ROC space which is closest to the perfect classifier (0,1) has a value of 1-SPC equal to 0.13 and a sensitivity of 0.84. The distance between the point (0.13, 0.84) and perfect classifier (0,1) is 0.2069.

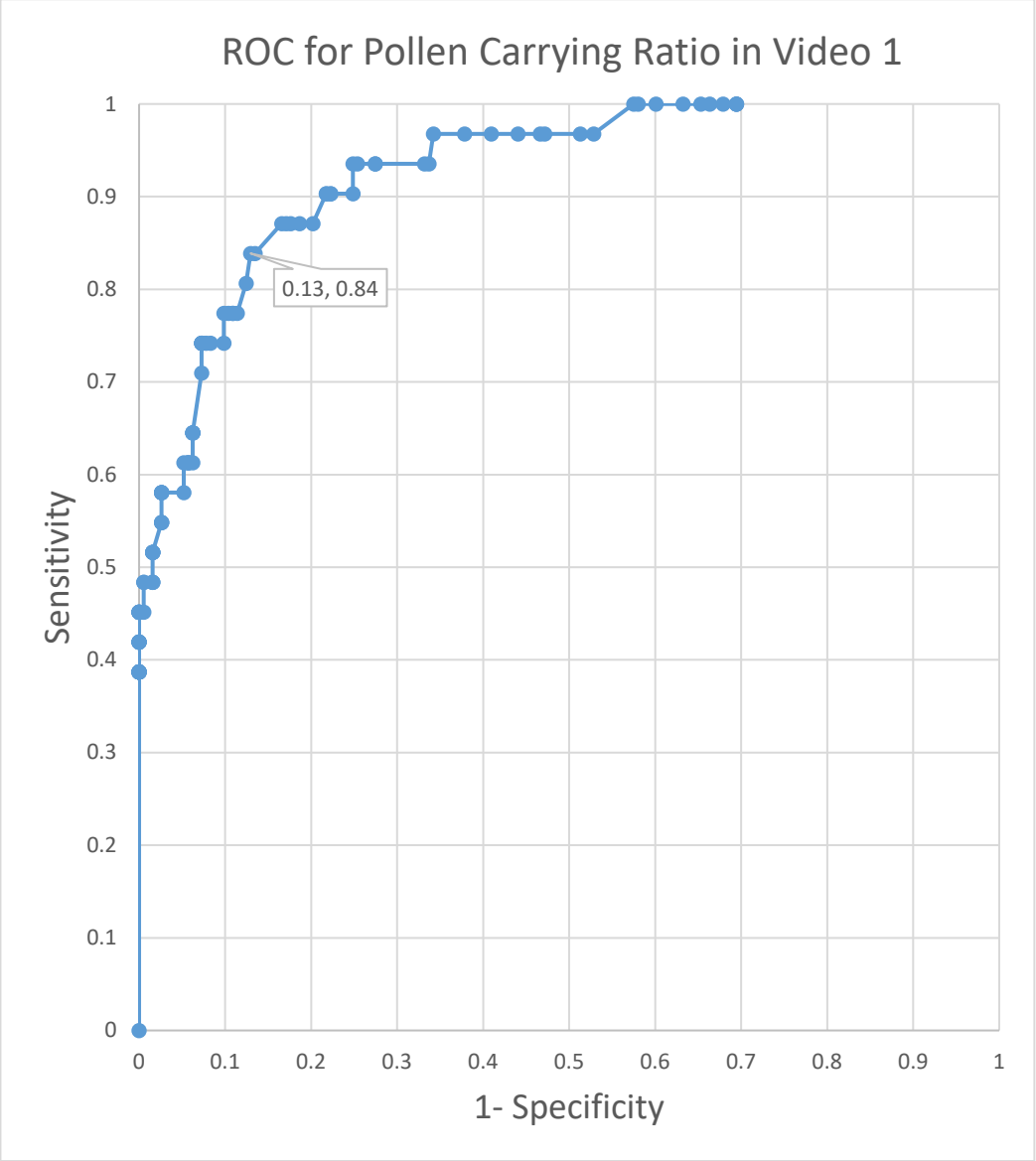


Figure 5.26 The ROC curve for the bees' pollen carrying ratio from video 1.

The corresponding threshold is 0.35 which is shown on Table 5-12. However, the threshold produces 25 false positives (FP), whereas the number of false negatives (FN) is only 5. This may not be a good threshold, because the number of false positives (FP) affects the measurement accuracy significantly.

Table 5-12 The ROC point closest to perfect classifier

Threshold	Distance	TP	FN	FP	TN	Sensitivity	Specificity
0.35	0.2069	26	5	25	168	0.84	0.87

Figure 5.27 shows the ROC curve of training video 2. On this curve, the point which is closest to the point (0, 1) is (0.17, 0.88). The shorted distance is 0.2097. The corresponding threshold is 0.29.

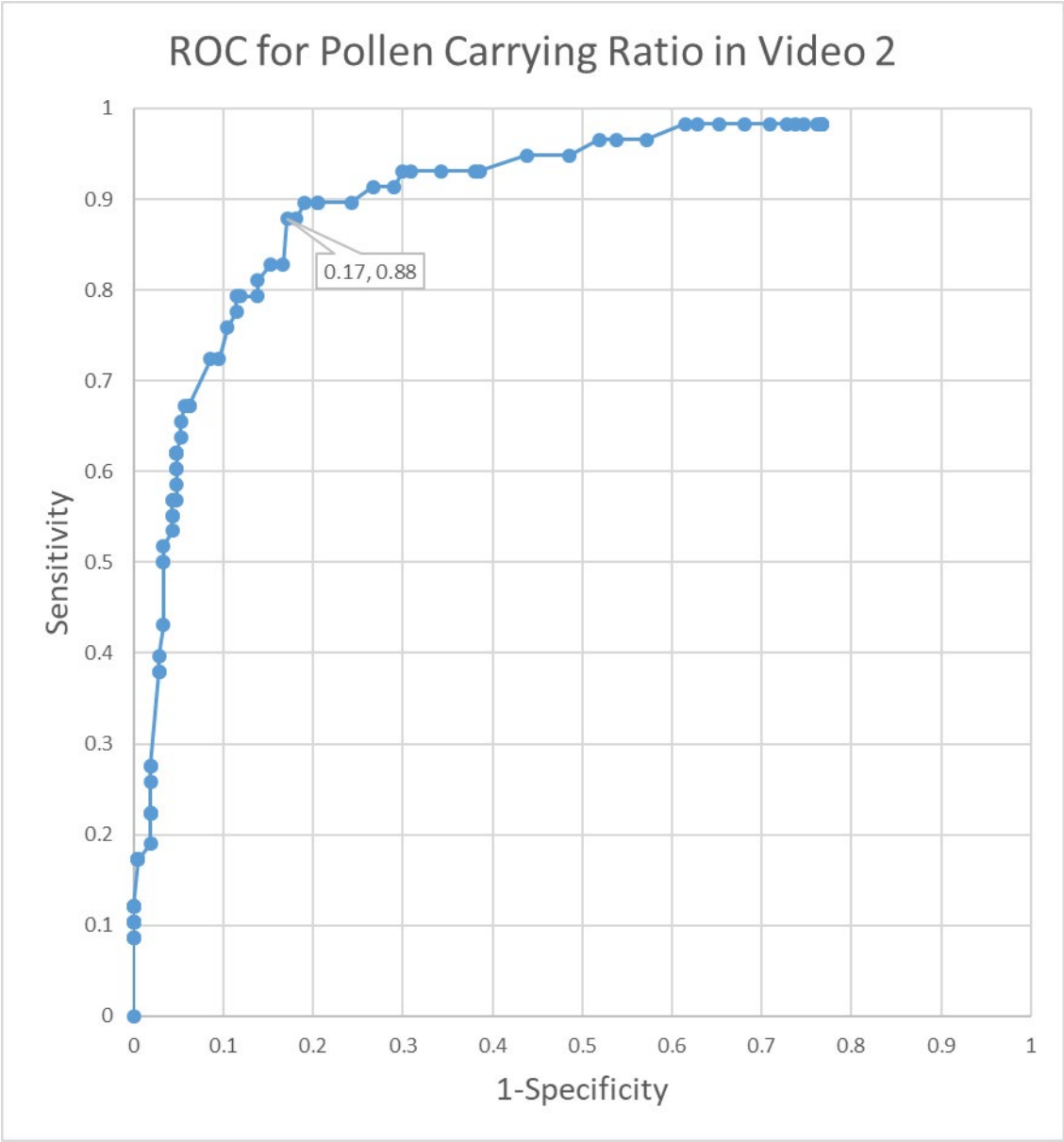


Figure 5.27 The ROC curve for the bees' pollen carrying ratio from video 2.

The point in the ROC space which is closest to the point (0, 1) is shown in Table 5-13. From this table, the threshold of 0.29 produces 7 false negatives (FN) and 36 false positives (FP). This value is also not suitable as a threshold for identifying whether a bee is carrying pollen or not, because the number of false positives (FP) is too high to measure the number of pollen bees accurately.

Table 5-13 The ROC point on ROC curve which is closest to point (0, 1)

Threshold	Distance	TP	FN	FP	TN	Sensitivity	Specificity
0.29	0.2097	51	7	36	174	0.88	0.83

Considering the two videos' pollen carrying ratio thresholds analysis, the points which are closest to the perfect classifier (0,1) are not suitable thresholds for the **pollen carrying ratio**. However, the aim of this research is to measure the number of pollen bees on the video. Therefore, a good threshold is one that produces the same number of false negatives and false positives. Then, the

resulting measurement is similar to the actual number of pollen bees. For the two videos above, the thresholds that can satisfy this, are show in Table 5-14:

Table 5-14 The thresholds in two videos having similar FN and FP.

Training videos No.	Thresholds	TP	FN	FP	TN	Sensitivity	Specificity
1	0.54	19	12	12	181	0.61	0.94
2	0.39	42	16	18	192	0.72	0.91

In Table 5-14, the two videos have two different thresholds which produce the same number of false negatives (FN) and false positives (FP). These thresholds have sensitivities greater than 60% and specificities greater than 90%. The threshold of **pollen carrying ratio** can be chosen as the average value of these two thresholds, which is **0.46**. In section 6.2, more videos are used to test the pollen measurement model.

Chapter 6

Chapter 6: Results and discussion

The pollen detection and measurement model was detailed in the previous chapter. The research includes two steps: bee detection and tracking, and pollen detection and measurement. These are both evaluated in this chapter. This chapter begins with the results of the bee tracking using the different methods outlined in previous chapters. After that, the model of measuring number of pollen carrying bees are tested on 5 bee monitoring videos. This will test **two lines method** and deep learning. The results are used to discuss the comparing of these two methods.

6.1 Results of bee tracking

6.1.1 The preparation

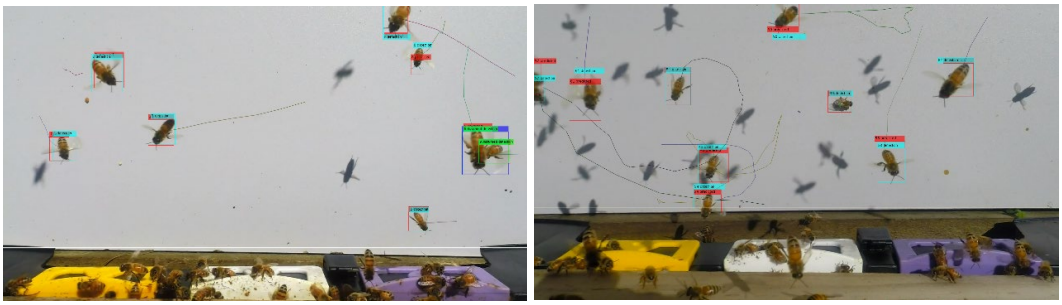
Bee tracking has been evaluated by viewing recorded video, frame by frame. Choosing the video to analyse is the first task. There are some conditions that may affect the result.

- Lack of Colour Information. If there is not enough sunlight to show the bee colour clearly, the system cannot detect them. Bee tracking will not operate until the bee colour reappears. When detection restarts, the system will treat these bees as new objects. If this occurs during a merged situation, the result will be unpredictable.
- Background problem. Although the single colour (white) board simplifies the background, the board can be made dirty by bees. Sometimes, bees drop their pollen sacs on the board. In addition, bees can stand or crawl on the board. Normally, these bees are detected as foreground. The flying bees also merge with them. This may reduce the tracking accuracy.
- Bee Shadows. The shadows do not have an orange colour, but they may be detected as moving objects. If the shadows pass standing bees or pollen sacs on the board, the system detects them as bees. Therefore, detection may be affected by shadows. If there are many shadows on the image, tracking may not work well.
- Bee Activity. If bees are quiet, there are not so many merged situations. Then tracking is more accurate. If bees are active, many bees are flying in front of the entrance. Then the video records more merging, so there will be more merged tracking.

In the conditions above, colour information is the most important factor. If the video is unclear, the bee detection can fail, and then the tracking will not be accurate. The chosen videos are colourful

and vivid, and then enhanced by a **Protune** operation which was mentioned in section 4.1. The background and shadow issues depend on whether bees are active or not. If many bees are flying and crawling at the entrance, there are many shadows on the background and the background becomes complicated.

Considering these conditions, two videos were chosen to demonstrate the results. Both of them were taken in a sunny environment, so that few detections were lost. The first video is an “Active” video (Figure 6.1 (a)). There were about eight bees in a frame view. This video was taken from the middle of the day, when the sunlight was clear and bees were flying actively. In addition, this video captured many merged situations, so that it can be used to estimate results for the merged situation tracking. The second video also shows bees flying busily, but there are many shadows on the board (Figure 6.1 (b)). Therefore, this is called a “Shadow” video. This video also captures many merged situations. In addition, it can test the method when there are many shadows in the background. Both videos have a white background, because these results were obtained very early in this research when the green background had not been created. The green background did not significantly improve the tracking results, so these white background videos are satisfactory for this evaluation.



(a) (b)

Figure 6.1 Example frames from three videos.
Note: (a) The “Active” video. (b) The “Shadow” video.

In this research, there are two main tracking situations that need to be evaluated: single bee tracking and merged bee tracking. Single bee tracking is the tracking model which tracks individual flying bees. Merged bee tracking occurs when the model tracks bees that appear to cross over each other. This tracking is complex, because, as shown in section 4.3.1, the bees cannot be detected individually.

Three tracking models are used to implement bee tracking: the prediction model, the Kalman filter model and the model using the Kalman filter + Hough transform *(K+H)*. They are all successful in single bee tracking, but experience difficulty in merged bee tracking. The prediction model uses the assumed detection (outlined in section 4.3.1) to solve the merged bee problem. The Kalman filter method performs merged tracking using the merged blob as the position measurement for both merged bees. However, this method produces a large error between the actual position and the measurement. The Kalman filter + Hough transform improves the merged bee tracking compared with using only the Kalman filter. The Hough transform estimates merged bee positions

for the Kalman filter, which reduces the error. In considering the tracking methods, several factors may affect the result:

- Boundary Mistakes. Most bees come from beyond the boundary. Some of them are only partly visible on their first appearance. In single bee tracking, tracking starts with this partly visible body. Merged bees on the boundary cannot be detected as a merged situation, but are treated as single bee tracking by the tracking model. In addition, bees may stay merged as they disappear beyond the frame boundary, so they are not seen separately. Then it is hard to know whether the merged bee tracking is correct or not. Furthermore, when two merged bees are close to the boundary, if one of the bees goes beyond the boundary of the video, the tracking cannot detect this bee separating from the merged situation. Thus, this bee is still tracked as merged, but it has actually disappeared. In summary, the video boundary causes incorrect tracking.
- Merged tracking limitation. Apart from the boundary problem, tracking models can also make other mistakes. In particular, merged bee tracking is not perfect. In the prediction model, bee detection during the merged situation is the assumed detection. This detection is based on the velocity just before merging occurs. During the merging, if the bee velocities change by a small amount, tracking may still be correct when the bees separate. However, if bees significantly change direction and/or speed, the model may lose track of some bees. In the Kalman filter model, the measurements of bee positions are the merged blob centre. This may result in a large error between the measurements and the predictions, and may cause tracking failure. In the *K+H*, model, most of the measurements of bee positions are close to the actual positions. However, this can be unsuccessful when a smaller bee is obscured by a larger bee. Although the Hough transform tracks some changes of direction, it cannot track the bees which change direction too quickly. Moreover, if two bees of the same size merge in the same direction, the Hough transform may not calculate the bee positions correctly. Sometimes the merging is complicated. Merging can involve more than two bees. Sometimes as many as five or six bees merge together. All three models may fail to track bees in this situation.
- Problem of shadows. The videos are taken under sunlight, so the shadows of bees are also moving with bees. Some shadows are beside bee bodies, when these bees crawl or stand on the background board. When a shadow is detected in the same blob as the corresponding bee, the shadow does not affect the bee tracking. However, there are many individual shadows on the video. The bee detection process removes most of them, but some shadows can be detected as bees in some situations, such as when the shadow is close to an orange colour or close to a bee body. These shadows can be mistakenly detected as bees. In addition, the shadows may also merge with bees, so that the tracking model will track an individual bee as a merged situation. Therefore, these shadows can affect the bee tracking. The shadows creating problems are recorded in the evaluation results to determine whether they significantly affect the bee tracking.

The evaluation results record the above error factors for further investigation. This can show which factor is the main cause of errors, so that it can indicate possible future improvements in bee tracking.

6.1.2 The results and evaluation

Two results are displayed: single bee tracking and merged bee tracking. The single bee tracking includes the evaluation of the prediction model and *K+H* model. The Kalman filter model for single bee tracking is similar to the *K+H* model, so the Kalman filter model is not displayed. In the merged bee tracking, all three models are evaluated, because they use different methods to track merged bees.

6.1.2.1 Single bee tracking

The first evaluation is for single bee tracking. The single bee tracking is evaluated frame by frame. Each bee is given an index on the video. If a bee is assigned the same index as in the previous frame, the tracking is correct. If the index is different from the previous frame on the same bee, the tracking is incorrect. Two models are evaluated in the single bee tracking: the prediction model and the *K+H* (Kalman filter + Hough transform) model. The Kalman filter model and *K+H* model use the same approach to track single flying bees. This section does not show the result of Kalman filter model.

The evaluation of single bee tracking using the prediction model is displayed in Table 6-1. Both the “Active” and “Shadow” videos were evaluated over about 300 frames to capture single tracking situations. These numbers do not include the tracking of standing and crawling bees, because this research focuses on tracking flying bees. In the “Active” video, single bee tracking produced 1741 correct tracking results, and 3 incorrect tracking results. The “Correct rate” is 99.8%. The “Shadow” video produced 1622 correct tracking results and 6 incorrect tracking results. The “Correct rate” in this video is 99.6%. Both videos have high correct rates. The single bee tracking is close to perfect.

Table 6-1 The single bee tracking results with prediction model

Videos	Frames	Correct	Incorrect	Incorrect reasons			Correct rate (%)
				Boundary	Tracking limitation	Shadow	
Active	300	1741	3	2	0	1	99.83
Shadow	300	1622	6	1	2	3	99.63

The reasons for incorrect tracking are also reported in Table 6-1. The “Active” video has 2 “Boundary” mistakes and the “Shadow” video has one such mistake. The boundary mistake in the single bee tracking happens on a bee disappearing on the boundary of the monitoring video. At the same time, another bee appears at the same place, the tracking uses the same index for the new coming bee. The mistakes because of “Tracking limitation” are also a small problem. Some bees fly too fast to be tracked. The model always gives these bees new indexes among frames. This mistake only happens for two bees in the “Shadow” video. The “Active” video does not have this problem. The “Shadow” mistake is that the shadow is tracked as a bee by the model. The “Active” video has one “Shadow” mistake; the “Shadow” video has three of these mistakes.

The “Shadow” video has more shadow mistakes than the “Active” video. However, the shadow does not affect single tracking significantly in the prediction model.

Table 6-2 reports the single bee tracking evaluation with the *K+H* (Kalman filter + Hough transform) model. To make sure this can be compared to the prediction model, both videos are also evaluated over 300 frames. The definition of “Correct” and “Incorrect” is similar to Table 6-1. Table 6-2 shows that the *K+H* model achieves a 99.83% “Correct rate” in the “Active” video and 99.63% in “Shadow” video. The “Correct rate” is similar to Table 6-1. The *K+H* model has a correct rate the as same as prediction model.

Table 6-2 The single bees tracking result with K+H model

Videos	Frames	Correct	Incorrect	Incorrect reasons			Correct rate (%)
				Boundary	Tracking limitation	Shadow	
Active	300	1744	3	2	0	1	99.83
Shadow	300	1627	6	1	2	3	99.63

In Table 6-2, the reasons of incorrect tracking are also shown. The boundary problem causes the same number of mistakes as in the prediction model in Table 6-1. There are two “Boundary” mistakes in the “Active” video and one mistake in the “Shadow” video. The number of “Tracking limitation” problems is similar to the prediction model in this single bee tracking evaluation. The “Active” video does not have this problem, but the “Shadow” video has two tracking limitation problems. The “Shadow” problem also has a similar number to Table 6-1. One “Shadow” problem occurs in the “Active” video, and the “Shadow” problem occur three times in the “Shadow” video. Although the “Shadow” video includes more “Shadow” problems than the “Active” video, the total number of this problem is very small. Therefore, the “Shadow” is not a significant problem in the *K+H* model in single bee tracking.

The single bee tracking has been evaluated above. The tracking is close to perfect with a correct rate of more than 99%. The prediction model and *K+H* model have similar results. All of the three reasons for errors (boundary, tracking limitation and shadows) are similar between the two models and they do not affect the result significantly. Especially, both the prediction and Kalman filter models deal with the shadow problem very well. The Kalman filter without Hough transform has similar results to the *K+H* model, so the results are not shown in this section.

6.1.2.2 The merged bee tracking

Because the bee monitoring video is 2D, bees can merge on the view of the video. Usually, merging consists of two bees merged together on the video. Sometimes, more than two bees merge, and merges of about six bees have been observed. It is difficult to evaluate the merged bee tracking frame by frame, because the individual bees cannot be detected during the merged situation. However, a bee can merge with other bees and then separate at the end of the merging. Once a bee separates from the merging, the evaluation is to check whether each separate bee has been tracked correctly from before the merging until after the merging ends. For example, if three bees merge and then separate, and one of them is correctly identified with itself before merging, but the other two are incorrectly identified, then one result is “Correct” and other two results are “Incorrect”. Incorrect tracking includes cases where a single bee is tracked as a

merged bee. In addition, if merged bees (bees merged to a single blob) are tracked as one single bee, these bees are counted as incorrect tracking. Two videos of “Active” video and “Shadow” video are used to perform the evaluation. The merged situation is less common than the single bee situation on the video, so the first 1000 frames from both videos are used for the evaluation.

The result of merged bee tracking with the prediction model is shown in Table 6-3. In the “Active” video, 173 merged bees are tracked correctly, 82 are incorrect. The “Correct rate” is 67.8%. In the “Shadow” video, 167 merged bees are tracked correctly and 125 merged bees are tracked incorrectly. The “Shadow” video has a 57.2% “Correct rate”.

Table 6-3 The evaluation of merged bee tracking with prediction model

Videos	Frames	Correct	Incorrect	Incorrect reasons			Correct rate (%)
				Boundary	Tracking limitation	Shadow	
Active	1000	173	82	55	23	4	67.8
Shadow	1000	167	125	103	18	4	57.2

Table 6-3 shows the factors causing mistakes. From the above table, the video boundary is the main factor. The “Active” video has 55 mistakes due to the “Boundary” problem. The “Shadow” video results in 103 incorrect trackings on the “Boundary”. “Tracking limitation” is the second main problem in producing mistakes in the videos. The “Active” video has 23 mistakes because of “Tracking limitation” and the “Shadow” video has 18 mistakes due to this problem. The tracking limitation of the prediction model is caused by bees changing velocity during their merging. This method for merged bee tracking (in section 4.3.1) only uses the direction and speed before their merging. Bees that change velocity quickly may be lost in the tracking. The “Shadows” in the video recording create another problem in the tracking. There are 4 “Shadow” mistakes in the “Active” video and a further 4 in the “Shadow” video. An interesting result is that the two videos have the same number of “Shadow” problems.

Merged tracking can also be performed by the Kalman filter model. Table 6-4 displays the results of merged tracking using only the Kalman filter. The same as the evaluation of the prediction model, over the 1000 frames, merged tracking using Kalman filter has “Correct rates” of 54.6% and 55.8% in the “Active” and “Shadow” videos respectively. Compared with the prediction method, the Kalman filter has advantages and disadvantages. The “Active” video has a lower “Correct rate” with Kalman filter model compared with the prediction model, but the “Correct rate” in “Shadow” video is similar to the prediction model.

Table 6-4 The merged tracking evaluation with only Kalman filter

Videos	Frames	Correct	Incorrect	Incorrect reasons			Correct rate (%)
				Boundary	Tracking limitation	Shadow	
Active	1000	136	113	46	63	4	54.6
Shadow	1000	154	122	87	31	4	55.8

Table 6-4 displays the reasons for the incorrect tracking. In the “Active video”, “Tracking limitation” is the main problem. There are about 63 mistakes under the “Tracking limitation” problem. The “Boundary” problem generates 46 mistakes which is the second main reason for the “Incorrect” tracking. In the “Shadow video”, the biggest problem is “Boundary”, which produces 87 mistakes.

The second biggest problem is the “Tracking limitation”, with 31 mistakes come due to this problem. An interesting result is that the Kalman filter reduces the number of “Boundary” problems when compared with the prediction model. However, the Kalman filter has more tracking limitation problems than the prediction model. The tracking method of the Kalman filter also uses the velocity before merging, but the difference is that the measurement is the centre coordinate of the merged blob (section 4.3.2.1). This causes a big error between the measurement and the actual bee position during bee merging. In addition, if more than two bees merge together (complicated merging), these methods can only match one measurement, but the bees are at different positions in the merged blob. Therefore, the Kalman filter has more “Tracking limitation” problems. The “Shadow” problem is also a small problem in the Kalman filter model. The number of these mistakes is the same as in the prediction model in Table 6-3. Both “Active” and “Shadow” videos have 4 “Incorrect” tracking errors because of the “Shadow” problem.

The results for the *K+H* model on merged bee tracking are shown below in Table 6-5. The “Active” video has a 67.9% “Correct rate”. In the “Shadow video”, the “Correct rate” is 60.7%. It can be seen that the Hough transform improves the tracking of the Kalman filter model, although it is not perfect. The “Correct rate” in both two videos is increased from Table 6-4. Moreover, compared to the Table 6-3, the “Active” video has a similar “Correct rate”, and the “Shadow” video has a higher “Correct rate”, than the prediction model.

Table 6-5 The merged bee tracking evaluation with K+H model

Videos	Frames	Correct	Incorrect	Incorrect reasons			Correct rate (%)
				Boundary	Tracking limitation	Shadow	
Active	1000	169	80	49	27	4	67.9
Shadow	1000	167	108	80	24	4	60.7

Table 6-5 displays the incorrect tracking reasons. The “Boundary” factor is the major problem in both videos. This problem produces 49 incorrect tracking in the “Active” video and 80 incorrect tracking in the “Shadow” video. This problem is similar to the Kalman filter model and the prediction model. Comparing the *K+H* model to the prediction model, the “Active” video has a similar number of “Boundary” mistakes, and the “Boundary” problem reduces from 103 to 80 in the “Shadow” video. The “Tracking limitation” problem produces 27 mistakes in the “Active” video and 24 in the “Shadow” video. The method of merged bee tracking in the *K+H* model is different from other two models above. The Hough transform uses the bees’ shapes before their merging to track the bees in the merged blob. This method decreases the “Tracking limitation” problem. However, the problem of “Tracking limitation” is similar to the prediction model. The “Shadow” problem is similar to the other two models, as both videos also have 4 cases of incorrect tracking because of “Shadow”. The “Shadow” is not a big problem in the *K+H* model.

Considering Table 6-3, Table 6-4 and Table 6-5, the worst model is the Kalman filter which has a correct rate of about 55%. The prediction model is better for the “Active” video, which has a correct rate of about 67%. The *K+H model* has the correct rate in the “Active” video is similar to the prediction model. The “Shadow” video with the K+H model has a correct rate around 60%.

For the merged bee tracking results above, the “Correct rate” is about 54%-68%. The main cause of errors is the boundary problem. Although the *K+H* model reduces the boundary errors in the “Shadow” video, none of the three models solve this problem completely. In addition, the “Active” video does not show the same phenomenon. However, the boundary problem only happens on the boundary of the video. When bees fly inside the frame of the video, the tracking still works. For example, if two merged bees appear on the boundary, when they separate in the video, they can still be tracked. The boundary can interfere with the tracking, but the tracking inside the video is reliable. The tracking models have been evaluated while ignoring the boundary problems. Table 6-6 shows the results without including boundary errors.

Table 6-6 The merged bee tracking result without considering boundary errors

Model	Video	Frames	Correct	Incorrect	Rate
Prediction	Active	1000	169	27	86.2%
	Shadow	1000	154	22	87.5%
Kalman filter	Active	1000	128	67	65.6%
	Shadow	1000	143	35	80.3%
Kalman filter and Hough transform	Active	1000	164	31	84.1%
	Shadow	1000	149	28	84.1%

In Table 6-6, most of the correct rates are more than 80%, if the evaluation does not include the boundary mistakes. Only the “Active” video with the Kalman filter model has a lower 65.6% correct rate. This table shows the prediction model performs best. The “Active” video has an 86.2% correct rate and the “Shadow” video has 87.5%. The *K+H* model has a slightly smaller correct rate. The “Active” video is 84.1% and the “Shadow” video is 84.1%. The worst model is Kalman filter. The “Active” video has a very low correct rate, but the “Shadow” video has a correct rate of 80.3%.

6.1.3 Summary of tracking results

The bee tracking model includes single bee tracking and merged bee tracking. In single bee tracking, the prediction and the *K+H* models produce similar results. This can be seen from Table 6-1 and Table 6-2. Both prediction and *K+H* models have a more than 99% correct rate. The single tracking is near perfect.

The merged bee tracking includes three models: prediction model, Kalman filter and K+H (Kalman filter + Hough transform). The full evaluation includes boundary, shadow and tracking limitation problems. The three tracking methods produce similar results. The tracking correct rate is about 55% to 68%. These are shown from Table 6-3 to Table 6-5. The boundary problem causes the highest number of tracking errors. However, this error only occurs on the boundary of videos. The three tracking methods are not designed to solve the boundary problem. The methods are only designed to operate inside the video. If the boundary problem is not included in the evaluation, the three methods all have similar results. The tracking rate is from 80% to 87% most of the time, although one video had only 66% accuracy.

The prediction model uses the assumed detection to solve the merging problem. It calculates the position of a merged bee using the predicted position and velocity from the video frame before merging. However, the disadvantage of the prediction model is the bee’s flying direction. The

model cannot respond when a bee changes direction during merging. The Kalman filter uses the merged blob position (blob centre) as the measurement to calculate the final tracking result. However, the error between the measurement and the prediction is very large. In addition, if bees merge for a long time, the prediction and measurement converge to the same position. Thus, the Kalman filter tracking can be unsuccessful. The Kalman filter + Hough transform (K+H) model finds the merged bees using the individual bee's shapes from the frame before merging. The merged bee tracking with Kalman filter and Hough transform reduce the error between prediction and measurement. In addition, the Hough transform can respond the bee's flying direction during merging. However, the tracking can still fail, because the Hough transform can sometimes be inaccurate in finding the position of the bees, and this affects the Kalman filter calculation of the flying direction of the bees. Consequently, the merged bee tracking can be a failure due to such Hough transform inaccuracies.

6.1.4 Mistakes analysis

According to section 6.1.2, the bee tracking model has a 99% correct rate in single bee tracking. The merged bee tracking has a maximum 87.5% correct rate, when the boundary problem is ignored. However, there are still mistakes in the bee tracking model, showing that the tracking model has limitations. This section will describe the mistakes and limitations of the bee tracking model on the video. If these limitations and problems can be solved, the bee tracking result will be improved.

6.1.4.1 Merged bee tracking limitation

The merged tracking limitations were explained briefly in section 6.1.1. This section will describe more detail of the advantages and disadvantages of the three tracking models (prediction, Kalman filter and K+H models).

Figure 6.2 shows the tracking results of the three models in the same merged situation. The red, cyan and yellow bounding boxes display prediction, correction and single bee detection respectively. The blue bounding box indicates the merged blob detection. The positions (including prediction, correction and single and merged bee detections) are the centres of the bounding boxes. The prediction in each image is for the current frame. It is calculated from the two previous frames (see section 4.3). Images (a) to (d) show the merged bee tracking of the prediction model. The green bounding box shows the assumed detection of the prediction model. The smaller bee's tracking in image (c) is not precise, but the final result of the tracking in image (d) is correct. The Kalman filter model result is displayed in images (e) to (h). The Kalman filter uses the centre of the merged blob detection (blue bounding box) as the measurement to calculate the correction with the prediction for the tracking. The correct position is also not precise in image (g) which is the same merged situation as in image (c). However, the result is correct, because the prediction still has the correct direction for the bee's flight path. The images (i) – (l) of Figure 6.2 show the K+H (Kalman filter + Hough transform) model for merged bee tracking. The measurement is different from the Kalman filter model. This comes from the Hough transform. The result of the Hough transform detection is shown by the bee's shape (shown as a green curve on the images)

which is from the shape before the merging. The position for the Hough transform detection is the centre of the shape. Most of the time, the Hough transform is accurate, as shown in images (j) and (k). The tracking result is also correct as image (l) shows.

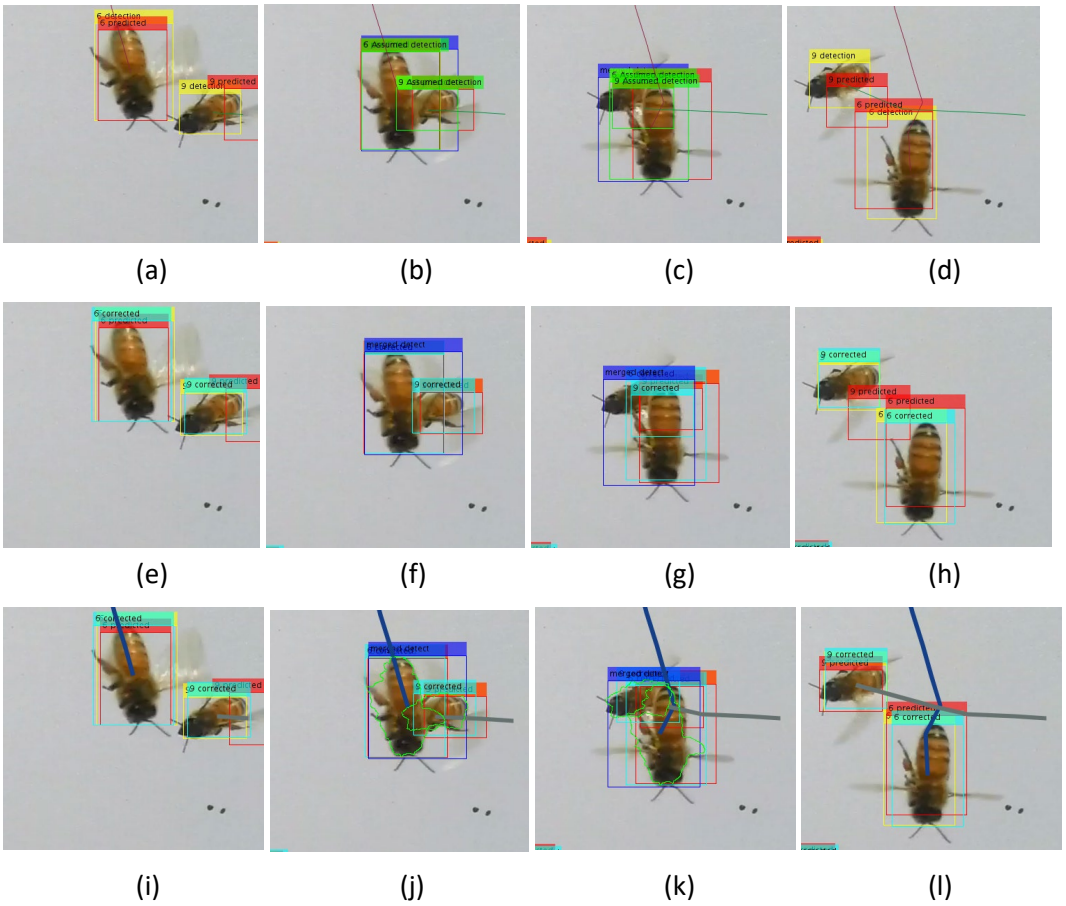


Figure 6.2 The results of correct tracking of three models.

Note: Consecutive frames from (a) to (d) show the result of prediction model. Consecutive frames from (e) to (f) show the result of Kalman filter model. Consecutive frames from (i) to (l) show the result of K+H model.

The problem with the prediction model is that it cannot follow the bees' changing velocities during merging, because the prediction of position and velocity comes from the frame before the merging. Figure 6.3 shows an example of incorrect tracking using the prediction model. In this merged situation, the Kalman filter model and K+H model are both correct, as indicated in Figure 6.4 and Figure 6.5 respectively. In Figure 6.3, these two bees fly and touch each other. Then, they both stop and fly back away from each other. However, the prediction model follows the flying direction before their merging. In image (c), the indexes of the two bees are swapped. The assumed detection does not follow the change in flying direction. Therefore, the tracking is incorrect. After merging, the two bees finish with different indexes in image (f) compared with image (a) before merging.

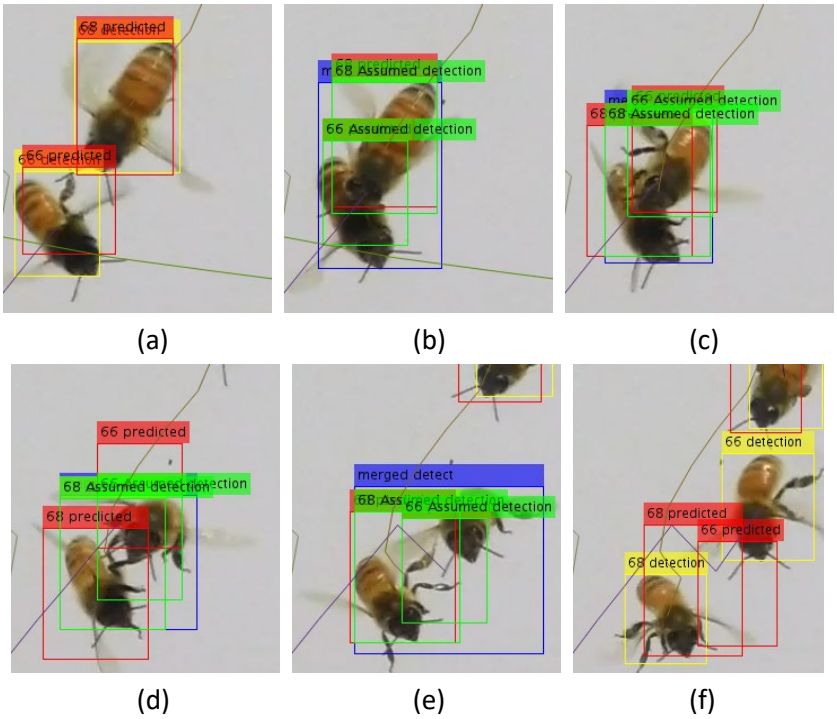


Figure 6.3 The incorrect tracking of prediction model.
Note: These are consecutive frames.

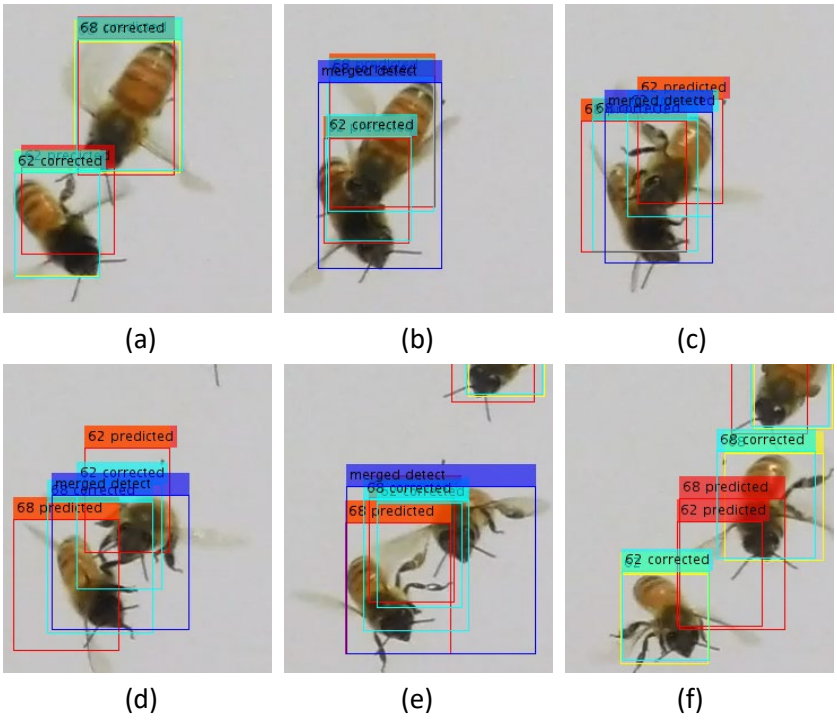


Figure 6.4 The Kalman filter model tracking in the same situation as Figure 6.3.
Note: These are consecutive frames.

Figure 6.4 shows the Kalman filter model tracking the bees correctly in this situation. In image (c), the prediction and correction of the Kalman filter show the same problem as the prediction model, but the prediction and correction are then made good by the measurements in images (d) and (e), because the Kalman filter uses the centre of the merged blob as the measurement. However,

the two predictions and corrections for the tracking of the two bees are close to the centre of the merged blob (the centre of the blue bounding box). Although the tracking is correct in image (f), the two predictions are very close to each other. Therefore, this tracking is not perfect.

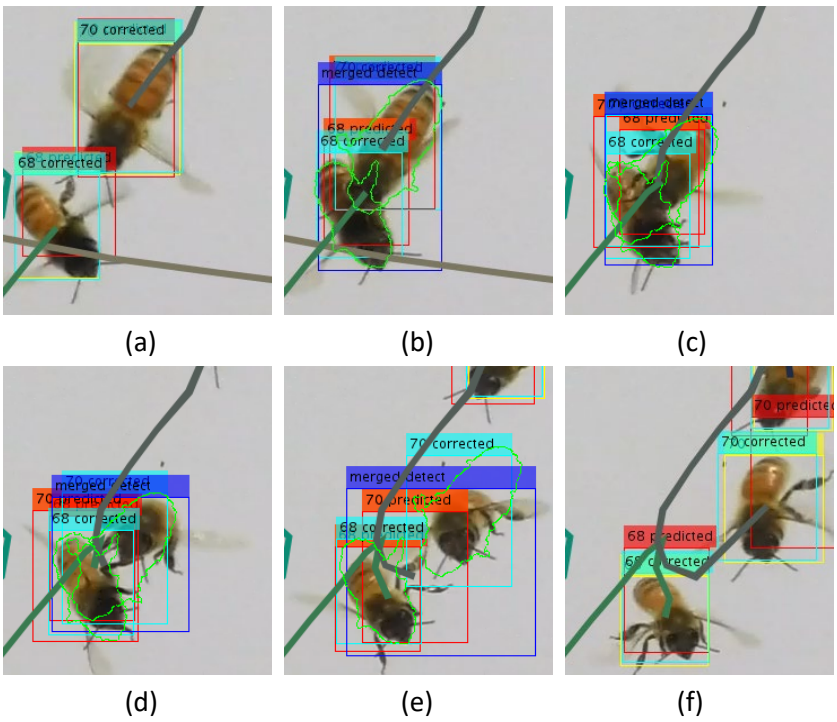


Figure 6.5 The K+H model tracking in the same situation as Figure 6.3. Note: These are consecutive frames.

In Figure 6.5, the K+H model tracks these two bees correctly. Although the two bees change their flying directions, their shapes' orientations are not changed significantly. The Hough transform detection uses the shapes and the orientations to produce perfect measurements, which are close to the actual positions of the bees in images (b) - (e). These measurements are used by the Kalman filter for tracking. In image (f), the two predictions of the bees are close to their actual positions, which is better than the Kalman filter model.

Figure 6.6, Figure 6.7 and Figure 6.8 show a merged situation in which the Kalman filter model is a failure, but the other two models are correct. In Figure 6.6, the prediction model calculation of the bees' flying directions is approximately correct. The assumed detection position is far from the actual smaller bee's position in image (e), but the prediction for image (f) is inside the smaller bee image. Therefore, the overall tracking is correct.

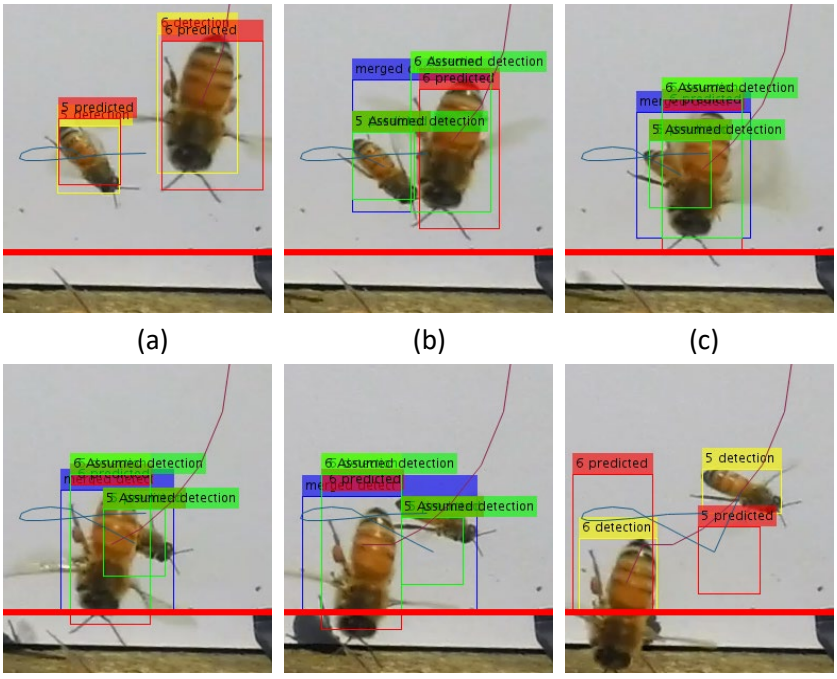


Figure 6.6 The correct tracking of prediction model.
Note: These are consecutive frames.

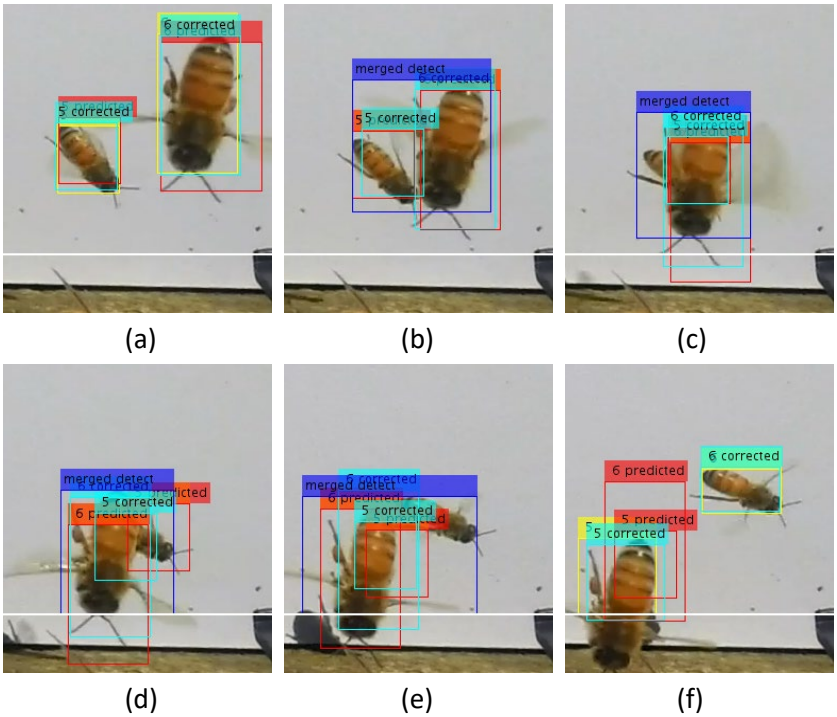


Figure 6.7 The incorrect tracking of the Kalman filter model.
Note: These are consecutive frames. The situation is similar to Figure 6.6.

The result of the Kalman filter model is shown in Figure 6.7. It can be seen that in image (d) and (e), the Kalman prediction and correction are close to the centre of the merged blob, because the prediction covariance matrix values get larger during the merging, so the Kalman gain adjusts the correction to be close to the measurement. The two bee trackings have only one measurement,

which is the centre of the merged blob. The predictions of the two bees relate to the corrections, and therefore the predicted positions of the bees are close to each other. This causes the tracking to be inaccurate. When the two bees separate, there is an incorrect assignment of the predictions to the detections of the bees (section 4.3.4). In image (f), the two bees are assigned different indexes from image (a). This is the main problem of the Kalman filter which causes the tracking result to be worse than the prediction model in section 6.1.2.2.

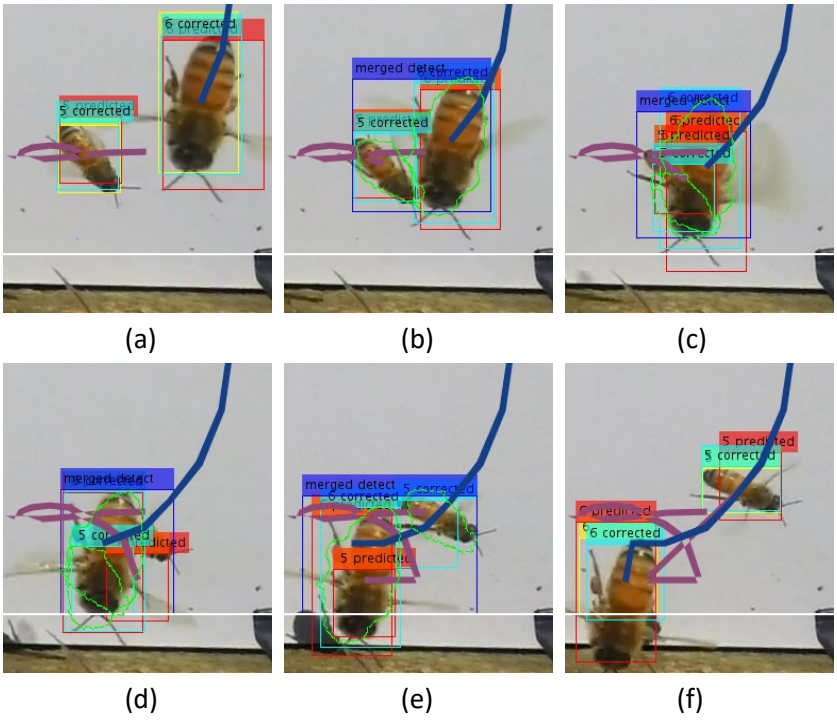


Figure 6.8 The correct traking of K+H model.
Note: There are consecutive frames. The situation is similar to Figure 6.6.

In Figure 6.8, the K+H model tracks these two bees correctly, even though the Hough transform detection loses the smaller bee in images (c) and (d). The mistake arises because as the smaller bee is hidden under the big bee, the shape information is not enough to use the Hough transform successfully. However, in the later image (e), the Hough transform detection finds the smaller bee perfectly. This leads to the correct tracking in the final image (f), when the two bees separate from the merging.

Figure 6.9, Figure 6.10 and Figure 6.11 indicate another merged situation. In this case, the prediction model and Kalman filter are correct, but the K+H model fails. Figure 6.9 shows the prediction model performance in this situation. It starts to lose the smaller bee from image (d), because the bee changes its flying direction during the merging. The tracking of the bee is correct, even though the prediction is far away from the detection in image (f), because the bigger bee's predicted position is close to the actual position.

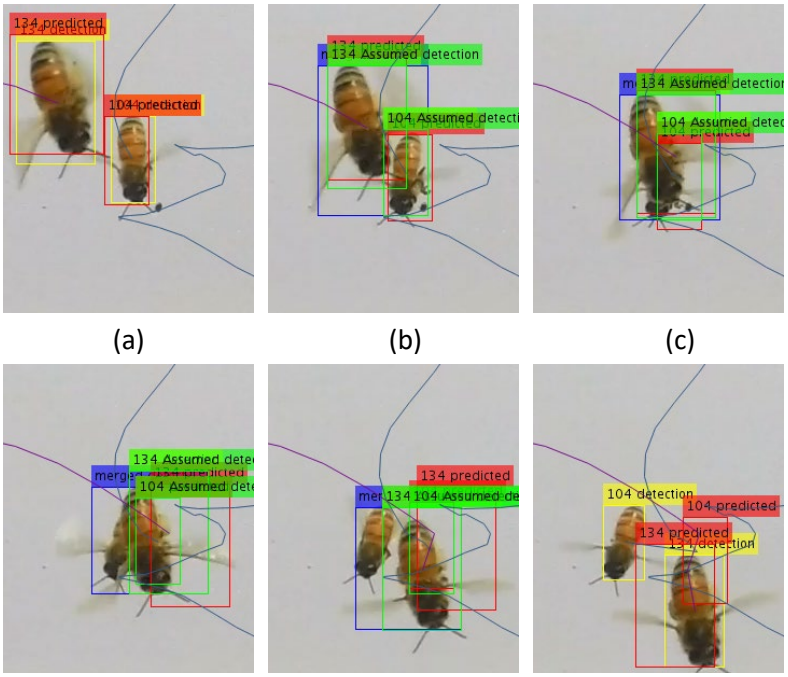


Figure 6.9 The correct tracking of the prediction model.
Note: These are consecutive frames.

In Figure 6.10, the Kalman filter tracks these two bees correctly. However, it can be seen that, the two bees trackings are close to each other and near the centre of the merged blob in image (d) and (e). This problem is similar to the situation of Figure 6.7. In image (f), the two predicted positions of the bees are close to each other and far from the actual positions. In this case the tracking is correct, but this is not always true.

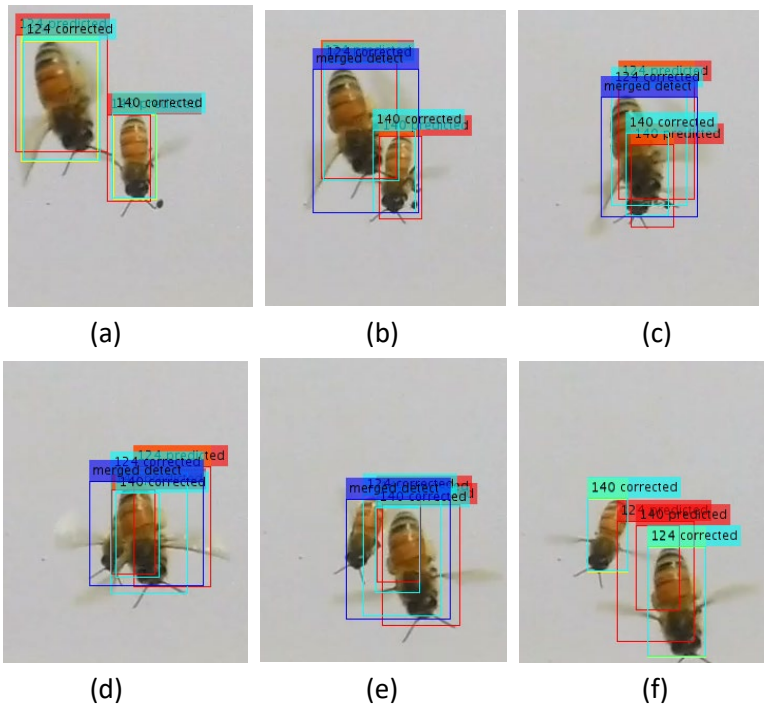


Figure 6.10 The correct tracking of Kalman filter model.
Note: there are consecutive frames from (a) to (f). The situation is similar to Figure 6.9.

In Figure 6.11, the K+H model wrongly tracks the bees, because the Hough transform detection is incorrect. In image (e), the Hough transform makes an error in the small bee detection. Although the Hough transform has a tolerance for the change in body orientation, the small bee changes its orientation more than the tolerance. Once the Hough transform detection is wrong, the correction of the Kalman filter is also wrong, and the tracking prediction for the next frame in image (f) is also different from the actual direction of the bee flight. In this case, the tracking fails. The indexes of the two bees in image (f) are different from image (a). This is the limitation of the K+H model.

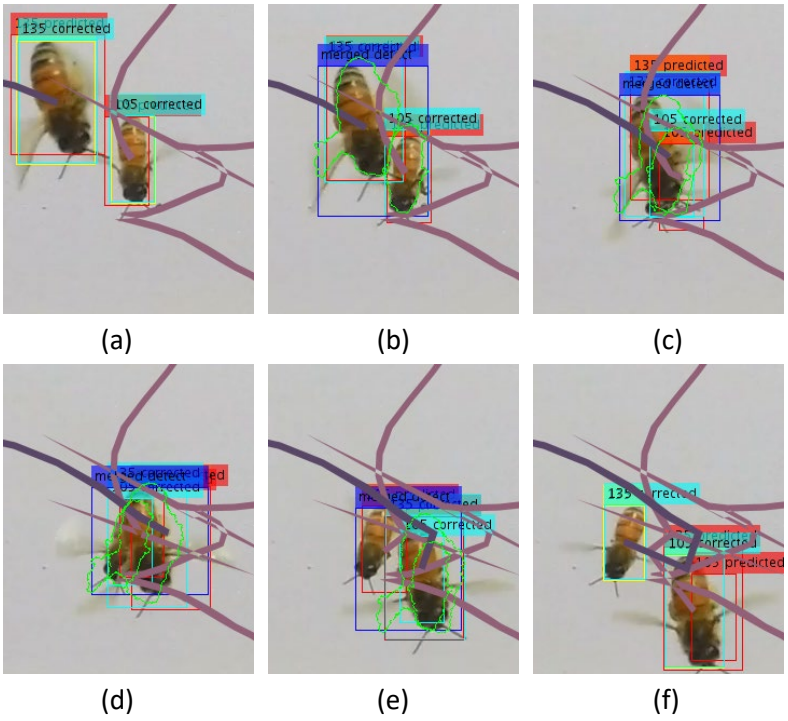


Figure 6.11 The incorrect tracking of the K+H model.
Note: These are consecutive frames. The situation is similar to Figure 6.9.

In Figure 6.3 to Figure 6.11, the three tracking models are compared. The disadvantages of each model are also explained. It can be seen that the K+H performs perfect tracking in the merged situation of Figure 6.5 and Figure 6.8. However, if the Hough transform detection is incorrect during the merging (image (e) in Figure 6.11), then tracking is unsuccessful. The prediction model can follow the flying direction and speed of the bees for the merged bee tracking, but it is not suitable for changes of flying directions. The Kalman filter always shifts the correction and prediction towards the centre of the merged blob, so it is not suitable for long merging times.

The next three figures (Figure 6.12, Figure 6.13 and Figure 6.14) show an example of a complex merged situation. There are 5 bees merging in 7 consecutive frames. In Figure 6.12 image (a), there are two bees merging, with a large bee coming towards them. Using north, south, east and west to describe the positions, from image (a) to (b), the large bee from the west and another bee from the north merge with the two bees in the south, appearing as 4 bees in the merged blob. In image(c), the south-west bee separates from the merged blob. Another bee from the north-east (in image (c)) merges with the large bee and a smaller bee in image (d). The north bee (in image (d) and (c)) moves to the north-west and separates from the merging. The north-east bee stays

merged in image (e) and separates in image (f). On the east of image (f) and (g), it can be seen that the large and the smaller bee are still merged. These two bees separate in the frame after image (g), which is not shown here.

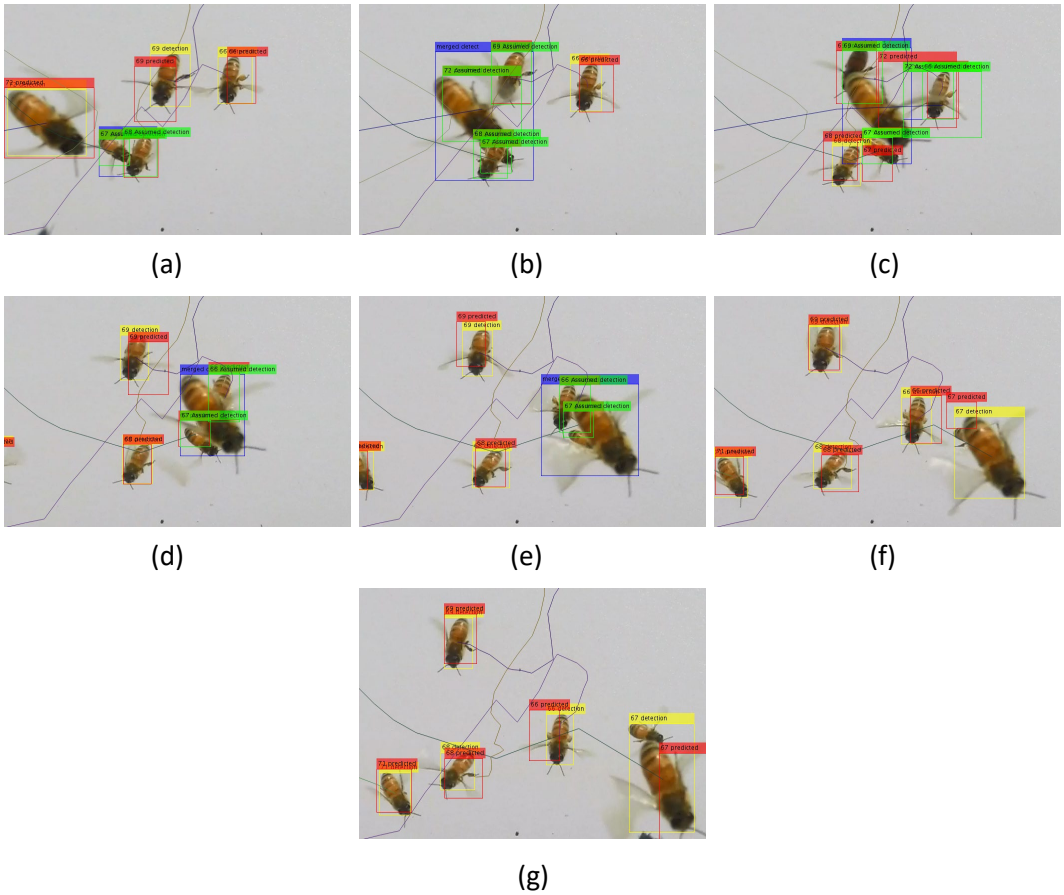


Figure 6.12 More complex merged tracking with the prediction model.
Note: There are consecutive frames.

This complicated merging sometimes happens during busy times on the bee monitoring videos. This is used to examine the performance of the three tracking models in a complicated merged situation.

Figure 6.12 shows the prediction model tracking process. In image (c), the south-west bee is tracked well. The index is the same as at the beginning in image (a). In image (d), the north bee is tracked correctly. It is recognised the same as in image (a). The north-east bee in image (a) is tracked well, because it is indexed the same in image (f) when it separates from merging. However, the larger bee in the west and a smaller bee (bee 67) in image (a) are not tracked correctly. In images (f) and (g), they are mistakenly tracked as a single bee flying. It can be seen that the larger bee is lost in image (c). In image (c), the smaller bee in the west is flying individually, but it is tracked as a merged bee. This is because the predicted position of the larger bee is much further forward than its actual position. The prediction model in this complex merged situation tracks 3 bees correctly and 2 bees incorrectly.

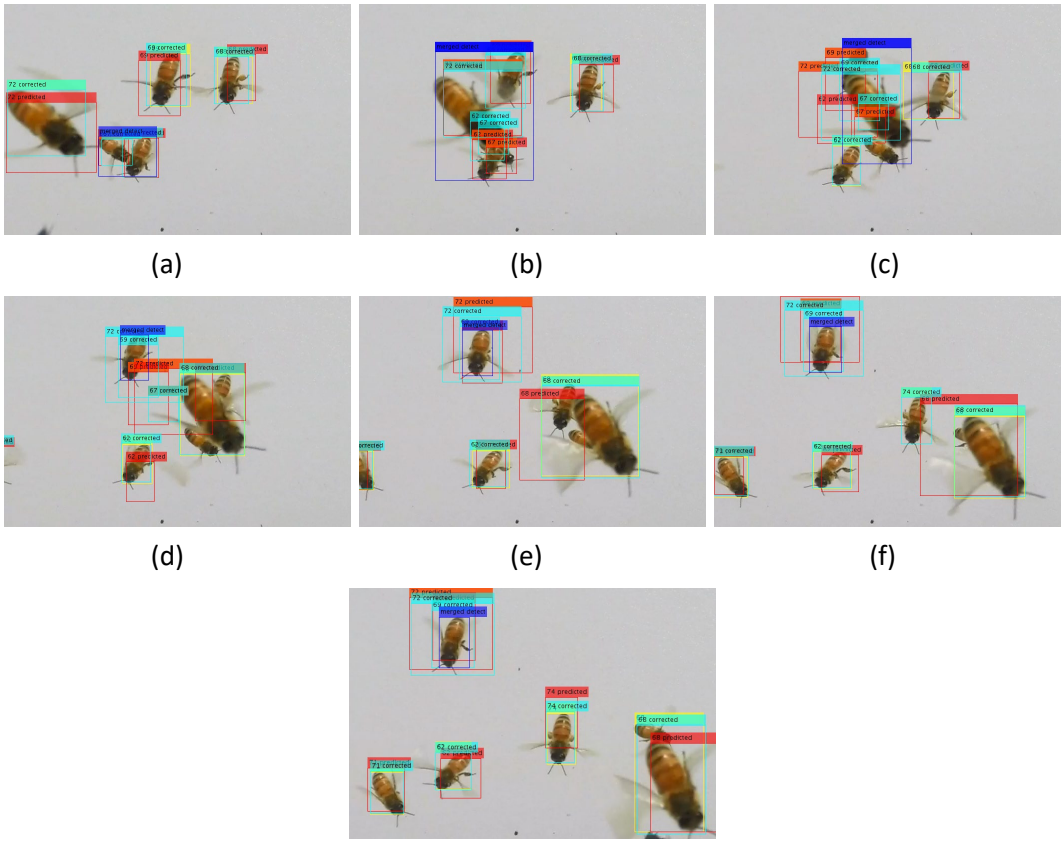


Figure 6.13 The more complex merged tracking with Kalman filter.
Note: These are consecutive frames. The situation is similar to Figure 6.12.

In Figure 6.13, the Kalman filter is used to analyse the same complex merged situation. The south-west bee, in image (c), is tracked well. The index is the same as at the beginning in image (a). In image (d), the north bee separates from the merging, but is mistakenly tracked as a merged bee. In the same image, the biggest bee is lost. This bee is still merged with other bees, but the model incorrectly identifies it as merged with the north bee. In image (e), the north-east bee, the biggest bee and a smaller bee (bee 67 in image (a)) are tracked as a single flying bee by the Kalman filter model. However, they are actually merged together. These three bees are tracked incorrectly by the model. The Kalman filter model in this complex merged situation tracks only one bee correctly. The other four bees are tracked incorrectly.

Figure 6.14 shows bees in the same complicated merged situation tracked by the K+H model. The north bee is detected incorrectly by the Hough transform from image (b), but the other three bees are still detected well. Following image (b), the north bee is lost by the tracking model in image (c). In the same image, the south-west bee is separated from the merged blob, but is mistakenly tracked as merged. These are two incorrect merged bee trackings. Another bee from the north-east merges with the biggest bee and a smaller bee in image (d). The north bee (in images (d) and (c)) moves to the north-west and separates from the merging. However, this bee is given the index which belongs to the north-east bee in image (c). Later, the north-east bee stays merged in image (e) and separates in the image (f). Because it is not detected as merged in image (d), it is given the index of the smaller bee which is hidden behind the biggest bee in

image (f). The tracking of the north-east bee (which is now in the centre in image (f)) is an incorrect merged bee tracking, because the bee is not recognised before it merges. On the east of images (f) and (g), it can be seen that the biggest and the smaller bee are still merged, but they are tracked as a single bee tracking. These are two incorrect merged bee trackings. It can be seen that all 5 bees are tracked incorrectly in the complex merged situation.

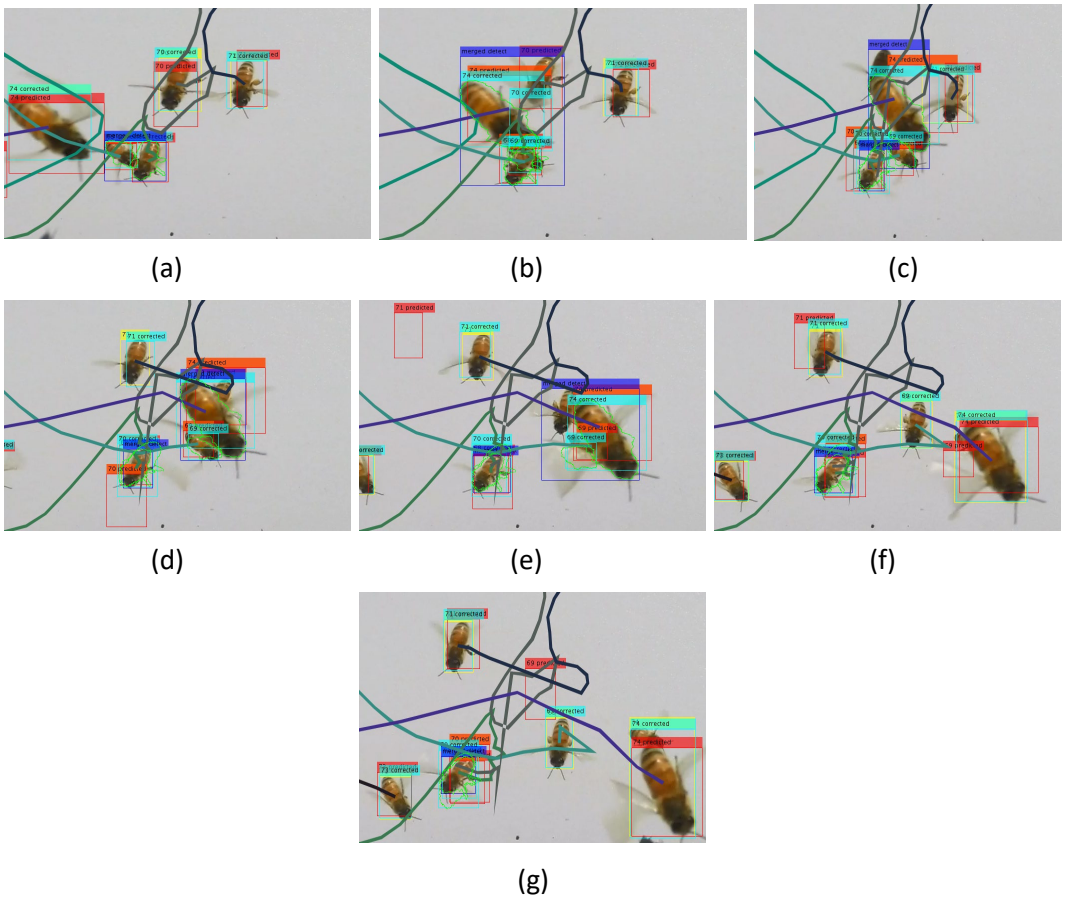


Figure 6.14 The more complex merged tracking with K+H model.
Note: there are consecutive frames from (a) to (g). The situation is similar to Figure 6.12.

In the example of the complex merged situation in the above three figures, there are 5 bees participating in the merging. The prediction model has the best performance, producing 3 correct trackings and 2 incorrect trackings. The Kalman filter tracks only 1 bee correctly, with 4 bees tracked incorrectly. The K+H model has the worst result with none of the bees being tracked correctly. This indicates the prediction model is more suitable for the complex merged situation. However, this example does not represent all complex merged situations. In addition, this type of complex merging is not common on the video. Therefore, this does not mean that the prediction model is the best tracking algorithm overall. The whole video shows all three models cannot always track bees accurately. The final evaluations of the three models have been shown in section 6.1.2. Moreover, all of the figures in this section indicate that the worst model is the Kalman filter, whereas the prediction model and K+H model have different advantages and disadvantages.

Considering the improvement of these three models, the prediction model cannot be improved any more. The Kalman filter model may be improved by calculating more optimal covariance values, which might improve the Kalman filter performance.

The K+H model can be improved by two approaches. One method is to improve the Hough transform detection in the merged situation. The Hough transform uses a bee's shape before its merging to detect the bee in the merge situation. This was introduced in section 4.3.3.2. In addition, the Hough transform considers 21 orientations that bees can change to, as explained in section 4.3.3.4. There are 21 voting maps from the 21 orientations. The 21 detected positions may include the actual position of the bee. These are the candidate positions. The final detected position is chosen as the candidate position with the highest vote value. However, the actual position may not be at the point with the highest vote value. Although the highest value is chosen by the region values around the 21 candidate positions (section 4.3.3.4), the experiment indicates this is not enough to track merged bees precisely. If there is a better method to select the bee position in the future, the Hough transform detection may be improved.

Another improvement for the K+H model concerns the Kalman filter. The Kalman filter combines measurement and prediction to calculate a corrected position. However, the corrected position can still be lost. In this model, the measurement is defined by the Hough transform, which usually improves the tracking. However, the Hough transform detection can still make mistakes. Figure 6.14 (b) displays an example of a mistake for the north bee. This bee is detected incorrectly and the detected position is far from the actual position (This bee's shape curve is near the south west bee). In this case the prediction model is working well. Figure 6.12 (b) shows the assumed detection of the bee (the green bounding box with the index 69) as being close to the actual position. It may be possible to combine the assumed detection of the prediction model with the K+H model, to improve tracking in the future.

6.1.4.2 Boundary errors

The problems on the boundary of videos was mentioned in section 6.1.1. Taking Figure 4.1 (b) in section 4.1 as an example, the boundary includes two types: video boundary and tracking boundary. The video boundary is at the left, right and top sides of the image, and the tracking boundary is on the bottom of the video (the red line). When a bee flies across the video boundary, the bee has disappeared completely. If a bee crosses the tracking boundary, the bee has not disappeared on the video, but the system treats it as having disappeared. The boundary problem on these two types of boundaries is similar. The examples below only focus on the tracking boundary.

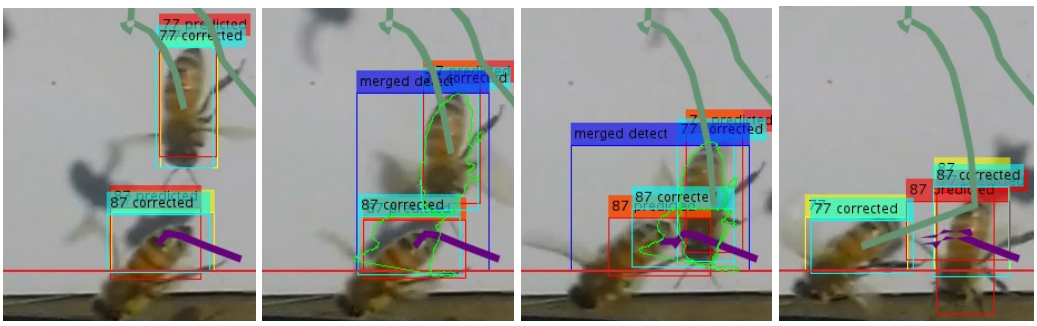
Figure 6.15 displays the first example of a boundary error. The upper bee is flying close to the boundary, while the lower bee is flying towards the boundary from the beehive. Because the system only detects one blob on image (b), the merged situation is identified as single bee tracking until the two bees separate in the image (d). It can be seen that the new incoming bee is only detected when it is seen as an individual. In this case, there are also two incorrect merged bee trackings. This problem occurs frequently in the video. Most of the bees appear and disappear

on the boundary of the video. It is common to see them merged on the boundary. However, they can be tracked well only after they are no longer merged.

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

Figure 6.15 The first example of a boundary problem.
Note: These are consecutive frames.

Figure 6.16 shows the second example of an error on the video boundary. In image (a), a bee has just appeared on the boundary with part of its body visible, and then it merges with another bee which is heading towards the boundary. In image (b), the newly appeared bee has only part of its body shape available for use by the Hough transform in the merged tracking, because the body of the bee in image (a) only shows half the bee. This affects the merged detection in the next merged frame, shown in image (c). The Hough transform detects the bee in a wrong position because of the lack of edge information. When the two bees separate in image (d), the tracking is a failure, which is shown by the bee indexes switching from one bee to the other. Merged tracking does exist in this situation, but the tracking is wrong, so there are two incorrect trackings. This problem is common in video bee tracking. Bees normally appear at the beginning with only part of their body showing. If the merging happens at this time, the problem cannot be avoided. However, the tracking may be successful, because bees can be tracked well if the direction of motion is detected correctly.



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

Figure 6.16 The second example of boundary problem.
Note: These are consecutive frames.

Figure 6.17 shows the third example of a boundary mistake. From image (a) to image (h), two bees merge near the boundary, while the smaller bee is flying towards the tracking boundary.

When their bodies separate in image (d), they are still detected as merged. If they separate inside the boundary, the merged tracking is correct. However, the smaller bee crosses the boundary and disappears. The tracking model still detects the bigger bee in the merged situation, because the bee detection model only detects one blob during the period from image (b) to (h). These are two incorrect merged trackings, because the merged trackings still exist, but the small bee has disappeared. This problem occurs when two bees are merged near the boundary and one bee crosses the boundary without having been detected separate from the merging.

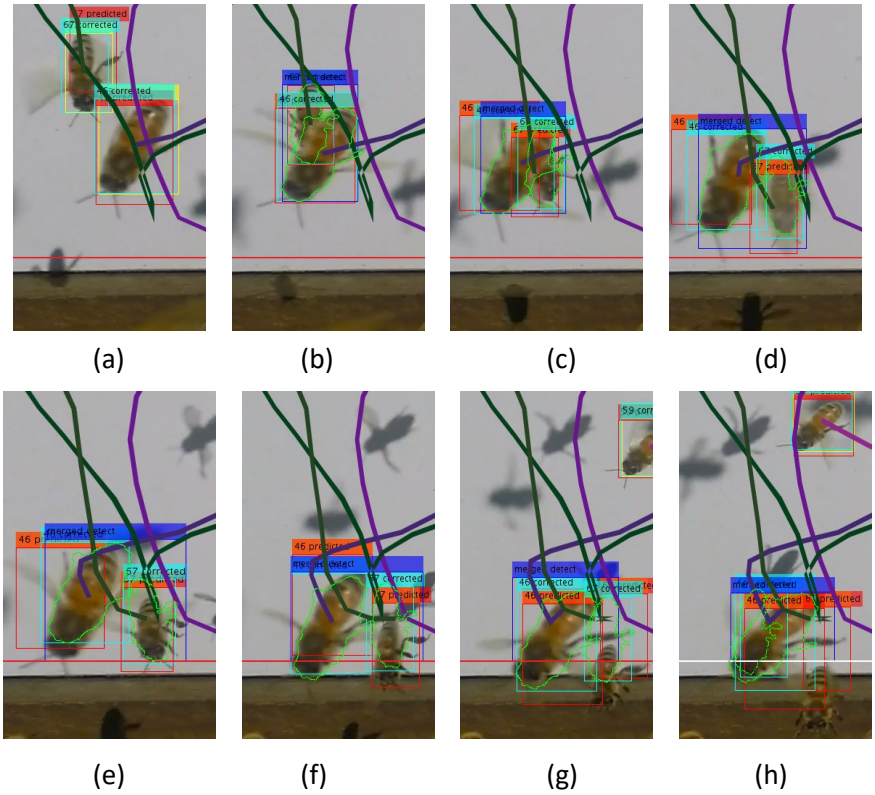


Figure 6.17 The third example of a boundary problem.
Note: These are consecutive frames.

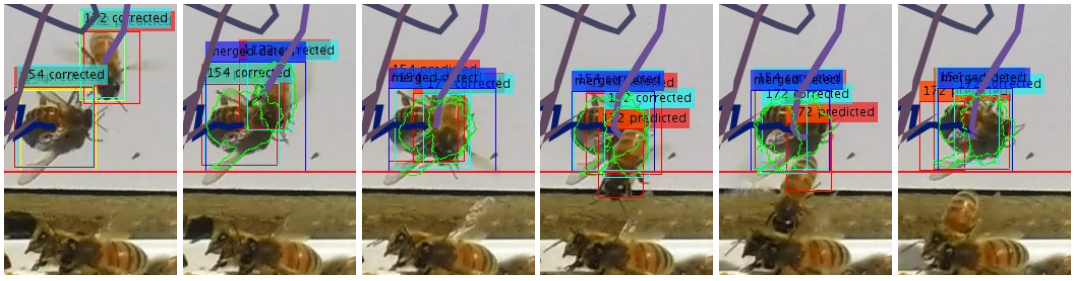


Figure 6.18 The fourth example of a boundary problem.
Note: These are consecutive frames.

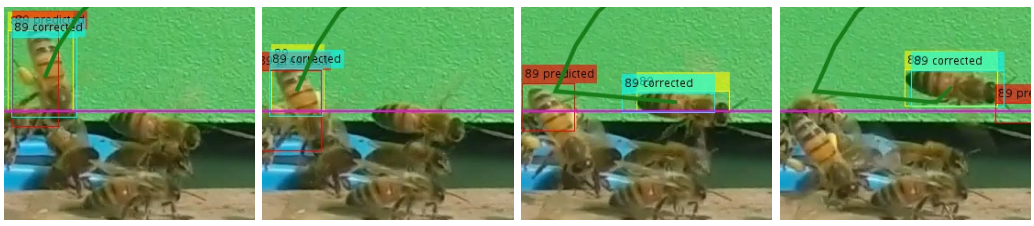
Figure 6.18 displays the fourth example of a boundary problem. A bee merges with a crawling bee, while it is moving over the tracking boundary. The consecutive frames from image (a) to (f) show this situation. These two bees stay merged, while the flying bee is crossing the boundary during the frames from image (d) to image (f). Then the flying bee disappears across the tracking boundary, but the system still detects a merged situation. This problem produces two incorrect

merged bee trackings. If there are bees standing or crawling in front of the beehive entrance, close to the tracking boundary, this problem can easily occur. The frequency of this problem depends on the time of day when the video is recorded.

Figure 6.19 displays the fifth example of a boundary problem. In image (a), a bee has just appeared near the boundary, and it is assigned index 260 and a bounding box. As described in section 4.3.1, when the bee first appears in a video frame, the prediction position and bounding box for the next frame is the same as for the detection in the current frame. Section 4.3.1 described the prediction model, and the Kalman filter treats the first appeared bee in the same way. When this bee merges with the smaller bee in image (b), the merged blob is far away from the large bee's prediction position (The prediction position is deleted in the image, so it is not shown here). Therefore, the tracking model does not detect this merged situation, but it tracks these two bees as a single bee. When the two bees separate in image (c), the smaller bee is tracked correctly, but the bigger bee is given a new index which means it is a new tracking. These are two incorrect merged bee trackings. This problem does not occur frequently. Most of the merged situations are detected by the bee tracking model, if there are two blobs before the merging. The bee 260 in this example is moving too fast to be tracked initially.

(a) (b) (c)

Figure 6.19 The fifth example of a boundary problem.
Note: These are consecutive frames.



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

Figure 6.20 The sixth example of a boundary problem.
Note: These are consecutive frames.

Figure 6.20 shows the sixth example of a boundary problem. Because this problem is not common, it was not found in the white background videos which were used to test the tracking model. Here is an example that occured on a green background video. On the left, a bee with pollen is crossing the tracking boundary, while another bee appears just below the middle of the boundary. Image (c) shows the index shifting from the bee with pollen to the new non-pollen bee. The bee with pollen has disappeared, but its single tracking keeps going with the other bee. This

is an incorrect single bee tracking result. The new bee in image (d) is another incorrect tracking, because the model tracks the middle bee as the bee with pollen. This problem rarely occurs. It is unusual for a disappearing bee and an appearing bee to be confused in this way. However, this problem may occur more frequently when bees are flying quickly, at a busy time.

The above six examples the video on boundaries exhibit simple problems. The first three problems are common on the video tracking. The last three problems depend on the time of day of the video recording. In the afternoon, it is common for many bees to crawl in front of the entrance of the beehive. At a busy time, bees fly faster, and therefore are more difficult to track.

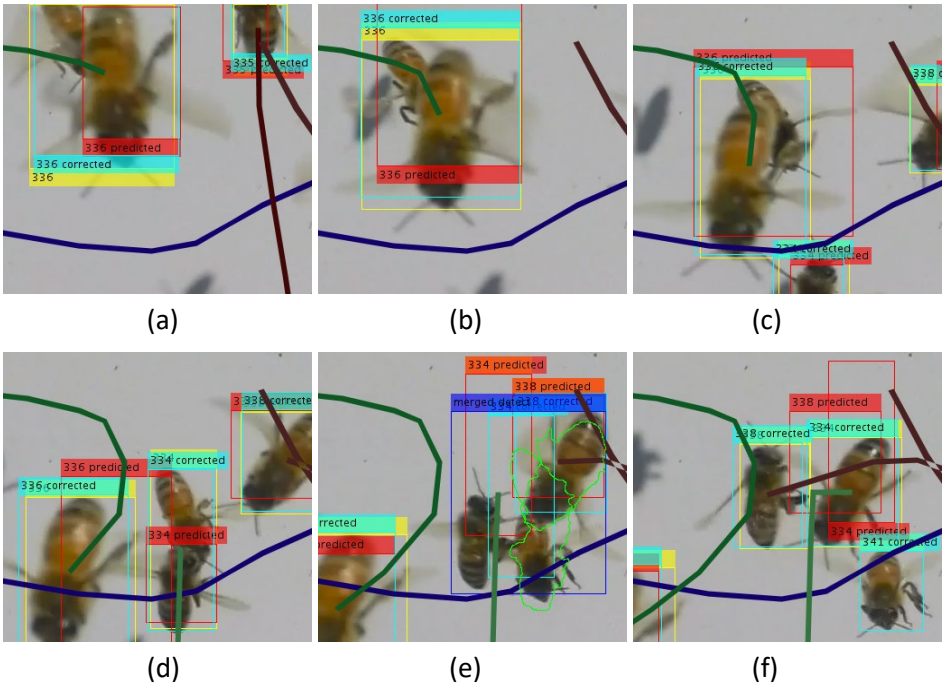
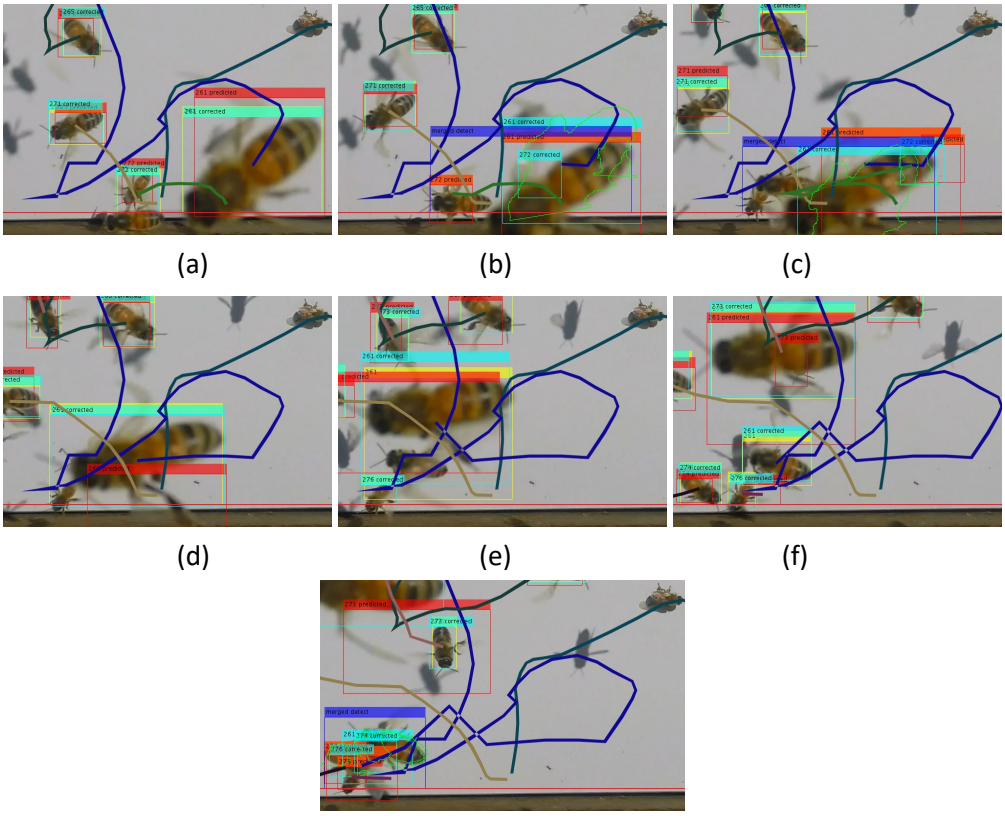


Figure 6.21 An example of complex merged situation on a boundary. Note: These are consecutive frames.

Moreover, the boundary merged situation can be more complicated. Figure 6.21 and Figure 6.22 show two examples of this. In Figure 6.21, two bees appear from the upper boundary in a merged situation. The smaller bee separates from the bigger bee, and then merges with another bee in image (d). After that, the two smaller bees merge with another bee in image (e). The three bees finally separate in image (f). None of these bees are tracked correctly.

Figure 6.22 displays a more complex situation than Figure 6.21. One bee appears much bigger than the other bees, because this bee is flying closer to the camera. In addition, other smaller bees also merge with the biggest bee during images (b) to (e). In image (f), another smaller bee is hidden under the biggest bee, but it reappears in the next frame in image (g). The two complex merged problems (Figure 6.21 and Figure 6.22) from the boundary do not occur frequently, but they do indicate a disadvantage of the bee tracking model. Consequently, this model is not very suitable for complex merging.



(a) (b) (c)

(d) (e) (f)

(g)

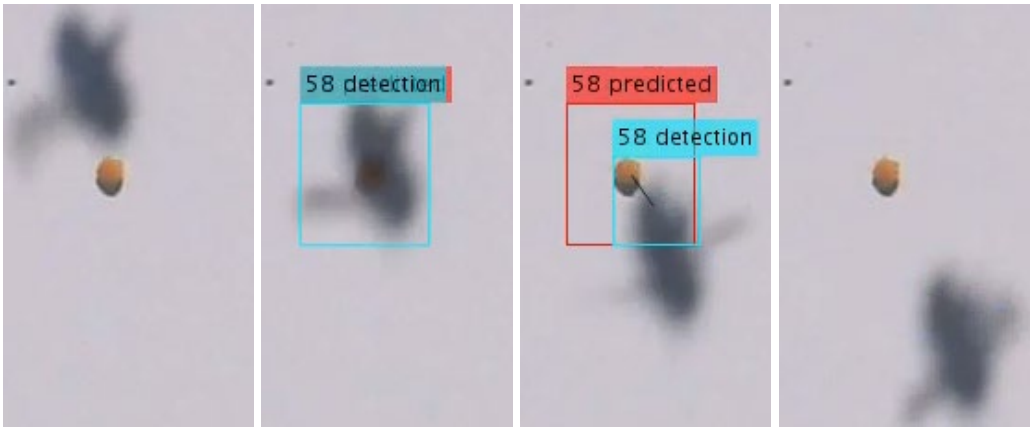
Figure 6.22 Another example of a complicated problem on boundary.
Note: There are consecutive frames.

Boundary errors are an obvious problem. This can be seen from Table 6-3, Table 6-4 and Table 6-5, in which this problem accounts for almost half of the number of 'Incorrect reasons'. No solution has been found for this problem, because the video cannot get more information from outside the boundary.

6.1.4.3 Problem of bee shadow

A single colour board is used to simplify the video background. Whether the background colour is white or green, the shadows of bees cannot be removed. The background subtraction with GMM is sensitive to the shadows, because they move with the bees. The colour detection reduces the shadow detection with bees. However, orange, black and white colours (on the green background) are detected on the video, so if a bee's shadow moves across these colours, they can still be detected. Sometimes, these moving shadows affect the bee tracking model.

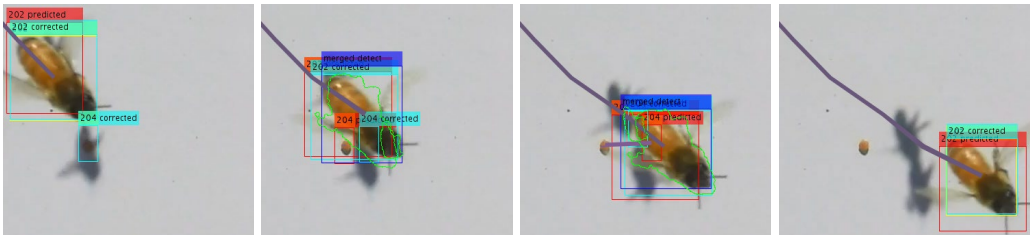
Figure 6.23 shows an example of the shadow problem. This affects single bee tracking, because the bee tracking model does not track bees, but the shadow's movement. The mistake is shown from image (a) to image (d). There is a pollen on the board, and it is not moving. When a shadow passes over the pollen, the bee tracking model detects this shadow with the pollen as an individual bee.



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

Figure 6.23 An example of the shadow problem which affects single bee tracking. Note: Images from (a) to (d) are consecutive frames.

The shadow problem also affects merged bee tracking. Figure 6.24 shows an example of the shadow problem. In image (a), a shadow is detected on the orange pollen. Then it merges with an incoming bee in image (b). The merged tracking continues in image (c) and ends at image (d). Because there is only a single bee flying, the merged bee tracking in this case is not correct.



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

Figure 6.24 An example of shadow problem which affect merged bee tracking. Note: These are consecutive frames.

The problem of bee shadows does not greatly affect the bee tracking model. This can be seen in section 6.1.2 in Table 6-1 and Table 6-2. The individual shadows are not frequently detected and tracked. Most of them can be removed by the bee detection method which is shown in section 4.2.3. In addition, there are different shadows which are detected and tracked with bees' bodies, but they do not affect the tracking. However, they may affect the pollen detection.

6.2 Evaluation of pollen measurement

6.2.1 Test videos

In this section, new videos are required to test the pollen counting models of the **two lines method** and deep learning model. The difficulty of video collection is that only a small group of bees brings pollen back to the hive. In this thesis, they are called "pollen bees". The best situation is during good weather, because the temperature is higher and bees are more active. In addition, the camera cannot record pollen sacs clearly on cloudy or rainy days. Therefore, all of the video data was recorded on sunny day. The training video 1 and 2 in section 5.2.1 have the highest ratio of pollen bees among all collected videos. They were used for training because they were more convenient for pollen bee image collection. Most of the monitoring videos include only a few

pollen bees. Apart from the two training videos, five other videos are chosen from all of the recorded videos for the test. These five videos all have some pollen bees. They are recorded from different days and different times of a day. Each video contains 3000 frames (one minute of 50 frame rate video). Because the pollen detection is on flying bees, these videos have also been chosen for having more individual flying bees. The background is green colour in all the videos. The number of bees on these videos are shown in Table 6-7 below.

Table 6-7 Test videos' information

Test videos No.	Frames	Recording time	Pollen bees	Non-pollen bees
1	3000	10:00 am	43	458
2	3000	10:46 am	41	453
3	3000	11:26 am	29	408
4	3000	2:00 pm	38	509
5	3000	4:00 pm	46	1107

For these five videos, the number of pollen bees has been counted manually, but the number of non-pollen bees is too high to be counted manually. However, the bee tracking model adds an index to each bee appearing on the video. The number of bee indexes can be used to obtain an approximate number of non-pollen bees. Although the bee tracking can include errors, the number of bees from tracking is very close to the actual number of bees. Therefore, the number of non-pollen bees is calculated from the number of tracked bees minus the number of pollen bees.

From Table 7, it is a challenge to identify pollen bees on the honey bee monitoring video. This is because the number of pollen bees is from 30 to 50, whereas the number of non-pollen bees is greater than 400. Especially, the test video 5 has 1107 non-pollen bees, which is about 24 times the number of pollen bees. This video was taken in the late spring, when the temperature is higher than earlier spring. This may cause the total number of bees to be twice that on other videos.

The evaluation parameters for the test include sensitivity and measurement error. The sensitivity indicates the ratio of true positive (TP) in the actual positives (actual number of pollen carrying bees). The measurement error is defined as below:

$$error = \frac{(TP+FP)-(TP+FN)}{TP+FN} \qquad 6\text{-}1$$

This measurement error indicates the difference between measured positive number (TP+FP) and the actual positive number (TP+FN). The measured number is the result of the pollen counting model; the actual positive number is the true number of pollen carrying bees appearing on the video. The measurement error shows how close the measured positive number is to the actual number of pollen bees. The test of the two methods is shown in next section.

6.2.2 Experimental results and comparison

The first test is the **two lines method** which applies the image processing technology. In this model, the **pollen carrying ratio** threshold is 0.65. This was discussed in section 5.1.6. Results from videos are shown in Table 6-8.

*Table 6-8 The pollen bees counting results with **two lines method***

Test videos No.	TP	FN	FP	TN	Sensitivity	Measurement error
1	30	13	10	448	0.70	-0.07
2	25	16	22	431	0.61	0.15
3	14	15	3	405	0.48	-0.41
4	15	23	3	506	0.39	-0.53
5	9	37	1	1106	0.20	-0.78
Total	93	104	39	2896	0.47	-0.33

From Table 6-8, test video 1 has the best result. The sensitivity is 0.70; and the measurement error is -0.07. This video includes large pollen sacs than other videos, so the sensitivity is the highest. The second-best result is test video 2, which has 0.61 sensitivity and 0.15 measurement error. The results of this video is closer to test video 1 than other 3 videos. Test videos 3, 4 and 5 have sensitivity lower than 50%. The results are worse when the recording time is closer to the end of day time. Sensitivities are 0.48, 0.39 and 0.2 respectively. In addition, there are much higher false negative (FN) errors than the number of false positive (FP) errors in test video 3, 4 and 5. This causes the measurement errors to be much higher than videos 1 and 2. The errors are -0.41, -0.53 and -0.78 respectively. The total number of true positives (TP) and false negatives (FN) are the totals over all five videos. They are 93 and 104 respectively. The sensitivity of the total result is 0.47. The total number of false positives (FP) is 39 which is less than the false negatives (FN). The measurement error for the totals is -0.33.

The second test is the deep learning model. In this model, the threshold of the pollen carrying ratio is 0.46. This was discussed in section 5.2.4. Table 6-9 displays the pollen counting test.

Table 6-9 The pollen bees counting results with the deep learning model

Test Videos No.	TP	FN	FP	TN	Sensitivity	Measurement error
1	30	13	13	445	0.70	0.00
2	30	11	19	434	0.73	0.20
3	18	11	12	396	0.62	0.03
4	12	26	6	503	0.32	-0.53
5	21	25	50	1057	0.46	0.54
Total	111	86	100	2835	0.56	0.07

From Table 6-9, test video 1 has a dramatically similar sensitivity (0.70) to the **two lines method** in test video 1. Both models have good results on this video. In addition, test video 1 has the best measurement error (0.00). The best sensitivity for deep learning is on test video 2, it is about 0.73. However, the measurement error is 0.2, because the number of false positives (FP) is more than the false negatives (FN).Test video 3 has a result close to test videos 1 and 2. It has a sensitivity of 0.62 and a measurement error 0.03.

Test videos 4 and 5 have different results from test videos 1, 2 and 3. In test video 4, the sensitivity is 0.32; and measurement error is -0.53. The main problem in test video 4 is the falsie negative (FN) which is 26, while the false positive (FP) is 6. The result for test video 5 has a large number of false positives (FP), which is 50. The number of false negatives (FN) is 25, which is half number

of false positives (FP). This causes the higher measurement error of 0.54. The sensitivity for test video 5 is about 46%. The number of true positives (TP) is little less than the false negatives (FN).

Comparing the two tests, the deep learning method has 3 videos (1, 3 and 5) which include higher sensitivities and lower measurement errors than **two lines method**. In test video 2, the deep learning model has higher sensitivity, but higher measurement error. In test video 4, the **two lines method** has better sensitivity than deep learning. In addition, in this video, the **two lines method** has a similar measurement error to the deep learning method. In **two lines method** test, the total measurement number of pollen bees is 132 (TP+FP in total), while the actual number of the pollen bees is 197 (TP+FN in total). The measurement error is about 33%. In the deep learning method test, the total pollen bee count is totally 211 (TP+FP in total), while the actual number of pollen bees is 197 (TP+FN in total). The measurement error is about 7%. In this case, the deep learning produces better results than **two lines method**.

6.2.3 Discussion of two methods

In section 6.2.2, the total measurement error shows the deep learning is much better than the **two lines method**. The **two lines method** produces a measurement error of 33%, but deep learning has 7% error. Apart from the tests, this section will explain more advantages and disadvantages of these two methods. In addition, comparing the two methods is discussed.

6.2.3.1 Data collection and analysis

In the **two lines method,** 276 pollen bee images and 1320 non-pollen bee images are collected from 400 frames. In total, 1596 individual bee images are used for analysis. The images are collected from bee detection bounding boxes. Then the bee main body detection and colour thresholding are utilized to analyse the images for pollen sacs detection. This outputs blobs on binary images. These blobs need to be identified manually as pollen or non-pollen blobs by observing the binary images, which takes a long time to complete. All these blobs are analysed by the blob detection software to produce values for four features. Then, the features are analysed find a method which can distinguish pollen and non-pollen blobs automatically in the future test. Therefore, the **two lines method** requires a lot of manual work.

In deep learning method, training replaces the data analysis. 1000 pollen bee images are collected for training and 400 other images (200 pollen bee images and 200 non-pollen bee images) for validation. They are collected from 4000 frames from two videos (2000 frames from each video). The images are also collected from bee detection bounding boxes on the full 1920*1080 resolution video frames. The 1000 pollen bee images are used for data augmentation which produces another 1000 images. Totally about 2400 individua bee images are used. The data collection also includes pollen sac labelling which needs to be done manually on the individual bee images. However, MATLAB has an APP named “Image Labeller” which makes the labelling much easier. Although, the labelling is manual job, it will not cost so much time. 1000 pollen images were labelled manually. However, the data augmentation of flipping images and labels is not a manual job. It is done by programming code. The data training can be done by the computer, not manual analysis. This research required 24 hours for each training when trying different learning rates.

Comparing the two methods, the **two lines method** uses all the pollen and non-pollen images included in 400 frames of one video for analysis. This is because the statistical method requires random and objective data collection. However, the deep learning method only needs pollen images for data training. This is because the Faster RCNN is only trained by the desired object (pollen sacs in this research) and background. The data collection for the two methods are different. Therefore, this can not show which method is better.

However, the **two lines method** data analysis costs more manual work than the deep learning method. The four features of pollen and non-pollen blobs are distinguished manually. It requires the researcher looking at pollen bee images and finding the corresponding blob features, especially the pollen blob features. In addition, the feature analysis requires manual work. The deep learning method also require manual work for labelling pollen sacs before its training, but it is simplified by software. Then, the data is trained automatically by a computer which saves manual work. Although the training costs time, it could be faster, if a faster computer could be used.

6.2.3.2 Error analysis

The **two lines method** partly uses image processing techniques for pollen detection, such as the bee's main body detection and colour thresholding. The main reasons for errors can be found by observing the image again. From the test, the false negative (FN) errors are the first problem. The main reasons for false negatives (FN) include different light condition, small pollen sacs, and some situations unsuitable for this method.

The different light conditions on the videos cause the pollen sacs to not appear clearly on the video, so the pollen colour detection cannot detect the whole of the pollen sac. This problem causes false negative (FN) errors. Figure 6.25 shows some examples. In image (a), the pollen sac on the right of the bee is partly dark, because the bee body blocks most of the sunlight. The pollen sac in image (b) has lost more colour than the sac in image (a) because the sunlight is blocked and there is motion blur. Images (a) and (b) mainly show the sunlight position problem. These pollen sacs can not be detected correctly using the colour thresholding method, because some of them only display part of the pollen, and some are too dark to be detected. The probability of these pollen sacs being detected is therefore lower than the clearly coloured pollen sacs.

Figure 6.25 The example of unclear colour pollen sacs.

Images (c) and (d) in Figure 6.25 show another light condition problem. On image (c) the pollen sac includes green colour. The pollen sac on image (d) is not clearly displayed. All the test videos have this problem to some extent, because their brightness is different from the training video.

The training video is recorded at 11 am, so the morning videos have mainly better results than the afternoon videos. Particularly, test video 5 was recorded at 4pm in the afternoon, when the sunlight is very soft. Most of the false negative (FN) mistakes in test video 5 are caused by the unclear colour of the pollen sacs.

Small pollen sacs cause false negative (FN) errors. The result of pollen colour detection includes pollen and non-pollen blobs. The next step is using pollen features to detect pollen sacs, as introduced in section 5.1.4. One of the main features is the area of the pollen. Most of the non-pollen blobs have a small area, so pollen blobs which also have a small area are difficult to distinguish from the non-pollen blobs. Figure 6.26 shows some examples of small pollen sacs. Comparing Figure 5.6 in section 5.1.2, these pollen sacs are much smaller. These pollen sacs appear small for two reasons. One is that the pollen sacs are actually small as shown on images (a) and (b), and the other is that the pollen is hidden under the abdomen of the bee, as shown on images (c) – (f). On images (c) and (d), part of pollen sac is hidden. The pollen sacs can still be seen, but they are not clear enough for the method. Pollen sacs on images (e) and (f) can be seen with difficultly. They are almost hidden under the bee body. Test videos 3, 4 and 5 have not only different light conditions, but also many bees carrying small pollen sacs, especially some bees with frequently hidden pollen sacs. Therefore, there are higher ratios of false negatives (FN) in these two videos with the **two lines method**.

Figure 6.26 Examples of small pollen sacs.

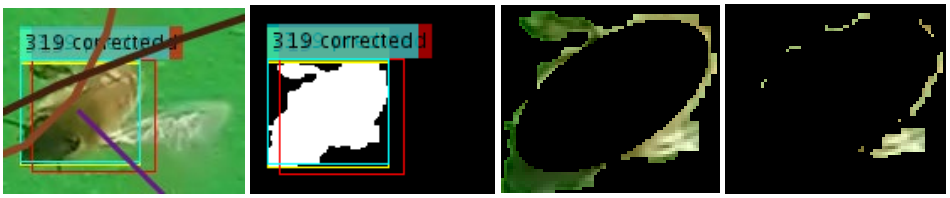
Some false negatives (FN) are caused when the image situation is not suitable for the method. For example, a pollen sac can be on top of the bee body. In this case, the pollen sac is shown clearly, but it is not beside the abdomen of the bee, so the bee blob does not show the pollen sac as a projection (or convex) feature. Figure 6.27 shows an example. Image (a) is the original single bee image; where the main part of the pollen sac is on the abdomen of the bee. The bee detection produces the bee blob in image (b), which shows the bee shape very well. However, the main body detection and removal in image (c) removes not only the main body, but also most of the pollen sac. Image (d) shows the pollen colour detection result. The pollen sac is only partly detected as a small blob. This problem occurs because of the unsuitable angle of the bee's body. The current **two lines method** is not suitable for this flying angle.

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

Figure 6.27 An example of a pollen sac on top of the body.
Note: (a) The original image. (b) The bee blob from bee detection. (c) The main body removal. (d) The pollen detection result.

False positives (FP) are another error which occurs on the non-pollen bee's identification. The main reasons of this error include wing reflection, bee body detection effects and shadows beside bees.

The wings reflect light and appear as a white colour, and wings are also beside the bee body. Therefore, wing reflection can be confused with the pollen sacs. The pollen colour detection can mistakenly detect this as a pollen blob. This problem produces false positive (FP) mistake. An example is shown in Figure 6.28. Image (a) shows the whole wing of the bee. The bee detection detects the bee blob in image (b), and it removes most of the wings, leaving only a small part. Image (c) shows the main body removed from the bee blob. The south east part of the image includes an area with a white colour. The blob features analysis will identify this white blob in image (d) as a pollen sac.



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

Figure 6.28 An example of wing reflection.
Note: (a) The original image. (b) The bee blob from bee detection. (c) The main body removal. (d) The pollen detection result.

Bee body detection effects are the second reason for false positives (FP). The **two lines method** sometimes relies on the quality of bee body detection. The bee detection works well on bee position identification, but it cannot always clearly detect the bee's shape. This comes from the disadvantages of colour thresholds. The colour of the bee's body can be affected by the green background, so that the shape or dark area includes a slightly green colour. In addition, the brightness changes at different times while recording the videos, but the colour thresholds are constant values in these different situations. Therefore, the bee detection can lose the bee shape, subsequently affecting all pollen detection methods. Figure 6.29 shows an example of this. Image (a) is the original bee image. The bee blob detection result in the image (b) does not show the bee's shape correctly. It includes a convex projection of the bee's thorax (on the right of the image) and a concave part (which is a small black hole on the image.). The main body detection in image

(c) does not include the thorax correctly. In addition, the concave hole causes the main body to be detected as smaller than the actual body, so that part of the abdomen on the right of the bee is also not included in the main body in image (c). Image (d) shows the result of pollen colour detection. There are two blobs on the left and right of image (d). They are similar to pollen blobs.

Figure 6.29 An example of bee blob detection problem effecting.

Note: The (a) The original image. (b) The bee blob from bee detection. (c) The main body removal. (d) The pollen detection result.

Another example of the bee blob detection problem is displayed in Figure 6.30. A bee can curl up its body while flying. Then the shape is too complex to remove as its main body. Image (a) shows a bee slightly curling up its body, so that the bee blob (in image (b)) is too bent to detect the main body. Therefore, the main body detection in image (c) does not remove the whole body. Image (d) shows the pollen colour detection detecting a blob which looks like a pollen blob.

Figure 6.30 An example of problem due to body curling.

Note: (a) The original image. (b) The bee blob from bee detection. (c) The main body removal. (d) The pollen detection result.

The bee shadow is another reason for false positives (FP). The bee detection method can remove individual bee shadows, but shadows beside bee body can still be detected. Figure 6.31 shows the detail of the problem. This is a disadvantage of bee detection model **D**. Model **D** is described in section 4.2.4.2. Image (a) is the single bee's original image. Image (b) is from the orange and white colour detection. It detects most of the bee body except the bee head. The foreground detection is shown in image (c). It includes body, wings, legs and shadow. The dilated (DI) image is shown in image (d). It includes the bee body and part of the shadow. Image (e) shows the colour detection of orange, white and black. It detects the whole bee body and the shadow. Because the dilated image and the colour detection both include the shadow, the final result in image (f) shows the bee body with the shadow. Image (g) is obtained after morphological filtering of image (f). In summary, because the shadow is near the bee, model **D** detects the shadow as part of the bee body.

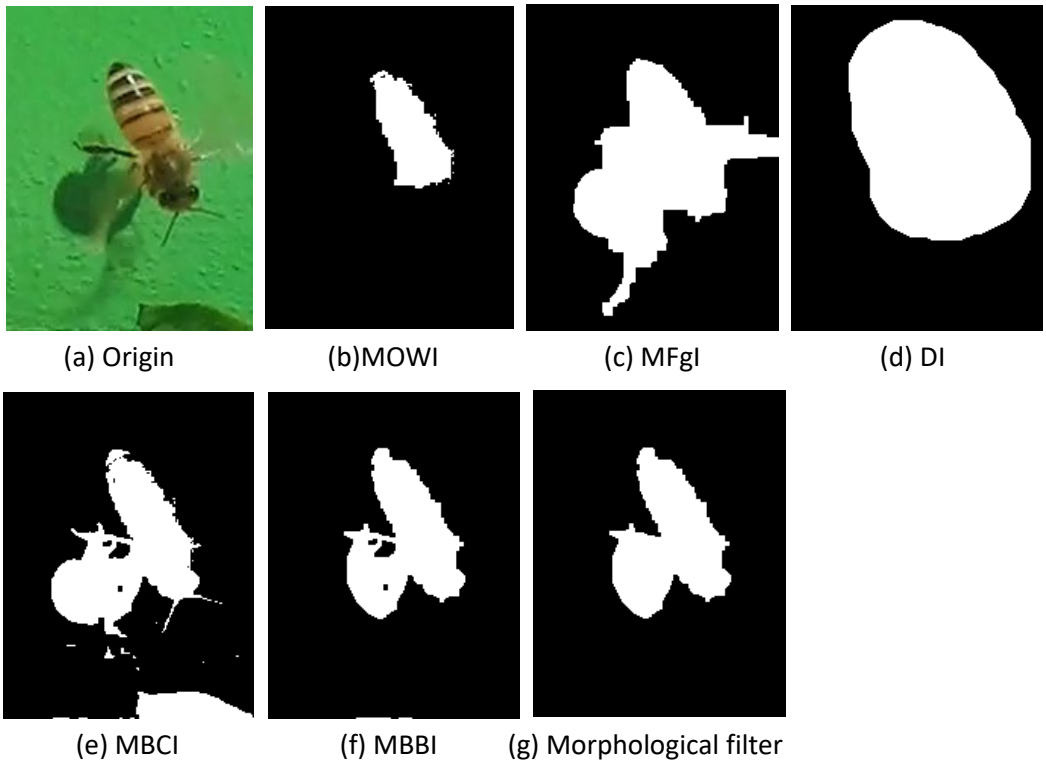


Figure 6.31 An example of the bee detection process with a shadow problem.

Note: (a) The original single bee image. (b) The orange and white colour detection. (c) The foreground detection. (d) The dilation image with orange, white colour and foreground. (e) The orange, white and black colour. (f) The combination detection. (g) The final result after morphological filtering.

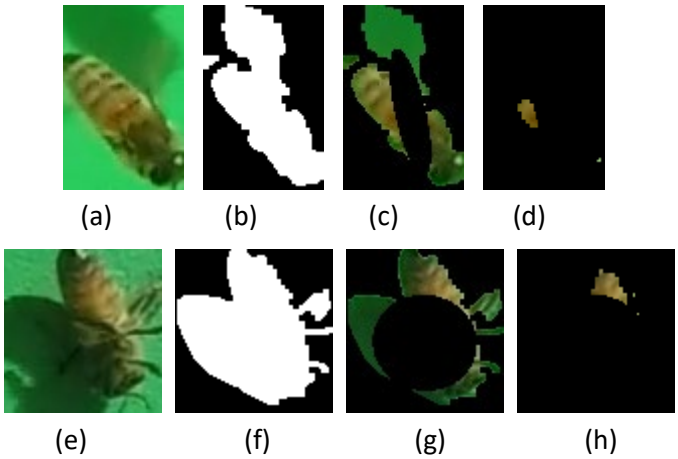


Figure 6.32 Two examples of shadow effects.

Note: (a)&(e) Original images. (b)&(f) The bee blob. (c)&(g) The main body removal. (d)&(h) The pollen detection.

This shadow problem which affects pollen detection is displayed in Figure 6.32. One example is shown from image (a) to image (d). The bee flies across a shadow in image (a), and the blob includes part of the shadow in image (b). Image (c) shows the main body detection and removal. The black ellipse is the main body calculated by the system, and it is different from the actual body of the bee. The pollen detection in image (d) shows a blob similar to a pollen sac. The second example is displayed from image (e) to image (h). This bee may be dead on the board, but the shadow detection in this case may be the same as for a standing bee. The image (e)

shows its body and shadow together. Image (f) shows a bee blob including both the shadow and body. The main body removal in image (g) is incorrect because the bee blob does not show the bee body shape correctly. The pollen colour detection in image (h) shows a blob which is similar to a pollen sac. This problem produces a significant error. The bee's blob on the binary image includes the bee's body and shadow. Then the main body detection method (in section 5.1.1) may not work correctly. The experiment shows that the shadow problem commonly happens with standing bees. If a bee stands on the green board, its shadow is just beside its body. However, this rarely happens with flying bees which typically cross the shadows of other bees.

The last problem which can produce mistakes is failure of bee tracking. Bee tracking failure can produce both false negative (FN) and false positive (FP) mistakes. The bee tracking model and pollen detection model are combined for counting pollen carrying bees. It calculates the **pollen carrying ratio** which is used to decide whether a bee is carrying pollen or not. If the bee has pollen, the **pollen carrying ratio** is updated from each frame of the video. However, if the bee is tracked incorrectly, it is given another bee's index, and the **pollen carrying ratio** are assigned to the wrong bee (section 5.1.6). Because the pollen carrying bee counting is based on the **pollen carrying ratio**, the count will not be correct.

In summary, the false negatives (FN) and false positives (FP) in **two lines method** are found for the reasons above. Solving these problems can help to improve the method. This can give the future work some direction.

The deep learning method uses neural networks to detect pollen, which is different from the **two lines method**. The reason for errors is difficult to analyse, because the detection process is in the hidden convolutional layers. The false negative (FN) errors may also occur because of the light conditions and small pollen sacs. It is hard to find the evidence from the detection process, but there are some inferences. The smallest pollen image resolution is 15*15. The CNN network performs convolution between weight filters of features and images. It focuses on the intensity pattern from the three colour channels on the image. The small pollen sacs may not show enough features (or pattern) for deep learning to detect them.

However, the deep learning method detects more true positive (TP) results than the **two lines method** in video with different light conditions. In the deep learning test, the test videos 3 and 5 have better sensitivity than the **two lines method**. One reason is that although many bees on these videos do not show clear colour, the convolution can still find the shape feature of the pollen sacs. This may help the deep learning method to detect more pollen carrying bees, when pollen sacs are not shown clearly. In addition, another reason may be that two videos were used for training. These videos have different light conditions. This may cause deep learning to handle more different light conditions. Thus, deep learning can detect more pollen carrying bees on test videos 3 and 5.

The reason for false positive (FP) errors is also difficult to find. Perhaps, the CNN finds something including similar features to pollen sacs. Figure 6.33 displays some examples of false positive (FP) errors. In image (a), one bounding box marks green background to be a pollen sac, another bounding box marks part of the abdomen as a pollen sac. Image (b) shows that the bounding box

also marks the bee's abdomen to be pollen sac. In image (c), part of the bee's wing is detected as a pollen sac. Image (d) indicates that the thorax of the bee is detected as a pollen sac. In image (e), the result shows the deep learning model detected part of the shadow as a pollen sac. The deep learning finds some things have similar features to pollen sacs, which people cannot see. In this case, it is difficult to find the reason of these errors.

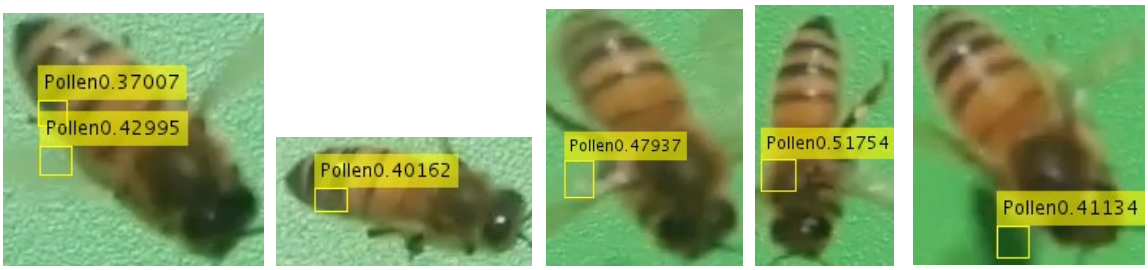


Figure 6.33 Examples of false positives (FP) from deep learning method pollen detection

However, the testing of the deep learning method shows the test video 5 has more than twice the rate of false positives (FP) of other videos. In addition, the non-pollen bee number in test video 5 is also more than twice that of other videos. Therefore, the number of non-pollen bees may cause deep learning to produce producing more false positive (FP) errors.

Because the deep learning method includes hidden layers, the reason for mistakes is hard to analyse. However, the deep learning method can be improved by using more layers than the VGG-16 network or more data augmentation, if the hardware is fast enough.

In this error analysis, the problem of the **two lines method** can be found by analysing the detection results on the bee images. The advantage of this is that it can give clear direction for method improvement. The reason for errors in the deep learning method is unclear. However, the model can be improved by designing a better network. The deep learning method is developing quickly at present. There may be better networks especially for small objects in the near future.

6.2.3.3 Comparison of research methodologies

In this research, the **two lines method** was built first, and this cost time to collect and analyse data. After that, the deep learning method was used near the end of the research. Some research methodologies which were changed for deep learning, were not modified in the **two lines method**. The result is that the two methods have been analysed in different ways.

First, although the number of images are similar for the two methods, the **two lines method** uses one video's data for the training, while the deep learning method uses two videos' data for its training. This may benefit the deep learning method results. If another video's data could be collected for training of **two lines method**, it might have different light condition and different sizes of pollen sacs. The blob features distribution may be different. The results of the **two lines method** may be different from Table 6-8.

Second, the threshold of the **pollen carrying ratio** selection is different for the two methods. In the **two lines method,** it was chosen at the point on the ROC space which is closest to the perfect classification point (0,1), while the deep learning model used a similar number of false negatives

(FN) and false positives (FP) to choose the threshold. In addition, the deep learning used the two training videos as data for the threshold selection, while the **two lines method** had only one video.

The different research methodologies which are mentioned above may cause the five video test results for comparing the two methods to not produce a very strong conclusion. It is better to use one more video data for the **two lines method**. However, collecting and analysing more data for the **two lines method** was not possible at the end of the research. In addition, training data for the deep learning method cannot be shared with the **two lines method**. This is because the data collection of the **two lines method** includes non-pollen images, while deep learning data does not include so many non-pollen images (only 200). Therefore, the **two lines method** still used the same methodology as before. This is a weakness of this research, but the result of this research is strong enough to give a conclusion that the deep learning is much better than the **two lines method**.

Chapter 7

Chapter 7: Conclusions and future work

This chapter will present conclusions concerning the bee detection model, bee tracking performance and the pollen detection and measurement. It presents both the advantages and disadvantages of these models. After that, future work from this research is presented. This includes the application of pollen blob measurement to the video, improvement of the methodology and further areas for research such as detection of other insects and the use of stereo video.

7.1 The bee detection model

The bee detection model detects all movements of bees utilizing foreground (motion) detection. The colour of bees is easily detected by using a single colour board as a background. The green colour background is better than the white colour for bee detection. The detected positions are close to the actual positions of bees. The bounding boxes around the bee bodies are all correct. The individual shadows on the background are removed. Small orange regions, such as pollen sacs which are dropped down on the board and dead bees' bodies on the board, can also be removed. When bees cross over each other in the video, the model detects them together as a merged blob rather than as individual bee blobs.

Conversely, there are some disadvantages. Firstly, bee shapes are not detected clearly and in the process some important parts of the body may be lost, such as pollen sacs, thorax and tail. The shape detection can be affected by the reflections off wings, orange colour on other body parts and the bee's angle relative to the camera. Secondly, shadows which are near or connected to bees' bodies, are also detected as part of the bees. This changes the bee blobs significantly, so that the main body detection is incorrect, and this affects the pollen detection later on.

7.2 The bee tracking performance

The flight tracking for individual single bees is successful in more than 99% cases. Flight tracking for the merged situation is more than 80% correct, when the boundary problem is ignored. The tracking produces bee flight curves that help predict the future positions of bees. The indexes assigned to each bee can be used to count the number of bees appearing in the video. Although there are some failures, faulty tracking does not significantly affect the pollen counting. Therefore, the tracking is suitable for pollen measurement.

The tracking can be unsuccessful. One of the main problems concerns bees crossing each other in the video frames (the merged situation). Sometimes, a bee cannot be detected in the merged situation, so its position is predicted (prediction model) or assumed (using the Hough transform). If the detection position is incorrect, the tracking may fail. Another problem is the boundary of the video. Bees appear and disappear at the boundary of videos. Sometimes, a bee appears at the same time as another bee disappears at the boundary. In this case, the tracking information (index and curve) is assigned to the new bee, which means the model tracks the new bee as the previous bee. This affects single bee tracking. Moreover, bees can merge when they are near the boundary. In this research, there is no method to check for merged bees on the boundary. In addition, if the merged bees separate from the merged blob and disappear on the boundary, they cannot be recognised as having separated. The boundary problem affects the merged tracking accuracy significantly. When the boundary problem is included, the merged bee tracking has a success rate of about 60%.

7.3 The pollen detection and measurement

Two pollen detection and measurement methods are used in this research: the **two lines method** and the deep learning method. The five videos test shows that the deep learning method has better performance than the **two lines method**. Especially in the total number of the pollen carrying bees counting, deep learning has 7% measurement error, while the **two lines method** has 33% error. In data collection and analysis, the **two lines method** requires more manual work and costs more time than deep learning. Deep learning can save manual work in its training, although it requires a faster computer. Regarding future improvement, the **two lines method** can be improved by analysing reasons for the problems described in section 6.2.3.2, but it is difficult to analyse reasons for errors in the deep learning method. However, the deep learning method could be improved by designing a better network, such as a deeper network or using more training data. In the case of overfitting, deep learning handles the overfitting better than the **two lines method**. This may be because of the data augmentation and training data from two videos. In summary, the deep learning method has more advantages than the **two lines method**. In addition, the technology is developing very quickly, which means a good future for the deep learning method. Therefore, applying the deep learning method for pollen measurement is very powerful.

However, the pollen detection and measurement has disadvantages. In the process of the **two lines method**, the main body detection relies on the bee blob from the bee detection. If the bee blob and main body detection do not work well, the pollen colour detection cannot detect the whole pollen sac. In addition, the pollen colour detection can detect some parts of the bee body as blobs which have a similar colour to the pollen sac. In the deep learning process, the application did not use the state-of-art methods (ResNet, Mask RCNN and YOLOv3), because of hard ware limitations. The deep neural network is a complicated model. It calculates millions of parameters (weights and biases), which requires more RAM of GPU. In addition, the training costs time, if the training computer is slow, although the training saves manual work. Deep learning requires the latest technology computer.

7.4 Future work

This research has made good progress in bee tracking and detection. The pollen detection and measurement also have a good performance in the test results. There several other ideas given in this section which may improve the results. Moreover, some ideas can extend the research to detect and count other insects on the video.

7.4.1 Bee detection improvement

The colour thresholding method is used in this research for bee and pollen detection. The disadvantage of using colour is the brightness of video. The brightness always changes in different videos, especially at different recording times. The research uses constant thresholds to perform the detection. Although the thresholds are suitable for most of the videos, they sometimes lose important parts of bee bodies, such as the thorax, tail and pollen sacs. It is necessary to use **adaptive thresholds** for the detection. The thresholds can be changed for different light situations. This may solve the problem of dark pollen and shadows. It may also detect clearer bee shapes and pollen sacs. It is complicated to find a method for the adaptive colour thresholds. The initial idea is to calculate the mean value of brightness level in the whole frame. The threshold can be the mean value or a value related to the mean. Another idea is the histogram of the frame. This relates to the colour difference. If the colour difference is detected, it may be able to be used for detecting other insects apart from bees.

The colour difference can also be used to detect the edges between different colour regions. The edges can be combined together to find objects' contour (shape). **Contour detection** is a common method used in computer vision technology. There are two examples of using contour detection. First, contour detection is used to create a hierarchical approach to detect many objects such as human beings, animals and plants (Arbelaez et al., 2011). Second, a shape-based (contour) detection algorithm is used to perform boundary segmentation (Toshev et al., 2012). In this example, some objects can be segmented from a complex background. In this research, most of the time, bees appear as a clear shape on the simple green background. The contour of pollen sacs is also clear on the individual bee image. Therefore, it may be reasonable to use contour detection for bee and pollen sac detection. With contour detection, bees may be detected even in some complex (natural) backgrounds. If the time required is not a problem, contour detection may be a good trial for this research.

7.4.2 Bee tracking improvement

There are two tracking models in this research: prediction method and Kalman filter with Hough transform. The second method can still be improved. If a bee disappears (under another bee) in a merged situation, the Hough transform cannot detect it correctly, causing the tracking to fail. These disappeared bees can still be tracked with prediction. **The assumed detection** from the prediction method and **Hough transform** can be combined for merged bees tracking. In this case, the Hough transform detects a merged bee directly if it still appears during the merging. In addition, if it has disappeared in the merged situation, the assumed detection can calculate the position of the merged bee from the prediction bbox. With the advantages of both methods, the tracking will be improved.

Another approach to improve the tracking is to optimize the **Kalman gain**. The tracking model uses different noise covariance matrices for the individual bee tracking and the merged situation. This modifies the Kalman gain to change the Kalman filter. In the individual bee tracking, the Kalman filter is close to the measurement (detection). In the merged tracking, the Kalman filter should be close to the prediction, because the detection has a large error concerning the actual position. However, when the variance value is a bigger value, the Kalman gain automatically becomes smaller. The Kalman filter is close to the prediction at the beginning, and then it goes towards the detection during the merged situation. Conversely, if the Kalman filter only follows the prediction, when bees stay merged over a long period of time, the prediction may go more wrong, because it is calculated using the velocity from the situation before the merging. Then the tracking may fail. Therefore, the Kalman gain may be not suitable for this situation. The Hough transform detection gives a measurement which can fix this problem, but it can be incorrect. If the Kalman gain can be calculated from the error between the prediction and the detection (measurement), this adaptive gain will improve the tracking.

7.4.3 Improvement of pollen detection and measurement

In **two lines method**, problems are mentioned in section 6.2.3.2. Solving those problems can improve the model of pollen detection. Particularly, pollen sacs are detected by colour detection. The colour detection may be improved by using adaptive thresholds or contour detection, as discussed above in section 7.4.1. In addition, shadows can affect the detection. This has been outlined in section 6.2.3.2. It is necessary to find a theory to detect and remove the shadow without removing the head of the bee. Thus, the main body detection and removal can be more accurate. Then the number of non-pollen blobs can be reduced.

For the deep learning method, a better computer is required. Then a deeper network can be used for pollen detection. Current technology can build very deep networks even to 101 layers (ResNet). The pollen sac is small, so the deeper networks will perform with higher accuracy for pollen detection. In addition, because of the time limitation, this research does not apply the latest network, such as Mask RCNN and YOLOv3. MATLAB (2018a) does not include functions to apply these two methods. Creating custom functions for these methods requires more time. However, both methods have been shown to be more accurate than Faster RCNN (He et al., 2017, Redmon, 2018 and Farhadi, 2018). If these two latest methods can be applied, the pollen detection will be improved.

The whole model of bee tracking and pollen detection is running very slowly, because the programme is not optimized. If using the computer mentioned in section 5.2 with CPU programming, the tracking model (K+H) with the **two lines method** required 0.8s ~ 4.4s per frame; the tracking model (K+H) with deep learning model required 1.4s ~ 5.5s per frame. Application of this model in real life requires a fast speed, especially applying this model in an embedded system. One of the problems which slows down the model is the Hough Transform. The Hough Transform that transforms the edges of a bee includes many iterations. Simplifying the Hough Transform can reduce the calculation. Another problem is that the deep neural network model also costs time. Each individual bee image is analysed with millions of parameters in the network model. In the computer, GPU programming may reduce the running speed. For an

embedded system application, a mobile network (Howard et al., 2017) model or SqueezeNet (Iandola et al., 2016) may be used to replace the VGG network in this research. This can increase the running speed, but it may be not accurate enough. This requires more research.

In addition, this research uses 2D video in the detection of bees and pollen sacs. This causes problems when the bees merge or pollen sacs are hidden. If stereo vision with two cameras could be used for this research, the situation and outcome would be different. If bees merge in one camera's view, they might still be separate for the other camera. In addition, pollen sacs might be hidden behind a bee body for one camera, but might still be visible by the other camera. Stereo vision needs more precise setup and the cameras need to be synchronized. Space would be synchronized by a manual measurement. The cameras would be set up using precise measurement to create an axis space, so that bees shown on the two cameras could be correctly matched. One trial has already taken place using a stereo video camera (Chiron et al., 2013a, Chiron et al., 2013b). This has only been used to track bees, and has not been utilized for pollen detection. In addition, the stereo video creates a depth map which can calculate the distance between the camera and the detected object. This can be used to remove bees which are too far from or too close to the camera without considering the bee size. Then the system can be assumed to only count pollen on bees which are close to the entrance of the hive in both the horizontal and vertical positions.

References

ADAMS, R. & BISCHOF, L. 1994. Seeded region growing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* 16**,** 641-647.

AIZEN, M. A. & HARDER, L. D. 2009. The Global Stock of Domesticated Honey Bees Is Growing Slower Than Agricultural Demand for Pollination. *Current Biology,* 19**,** 915-918.

ALAUX, C., BRUNET, J. L., DUSSAUBAT, C., MONDET, F., TCHAMITCHAN, S., COUSIN, M., BRILLARD, J., BALDY, A., BELZUNCES, L. P. & LE CONTE, Y. 2010. Interactions between Nosema microspores and a neonicotinoid weaken honeybees (Apis mellifera). *Environmental microbiology,* 12**,** 774-782.

ANANTH, C., SENTHILKANI, A., GOMATHY, S. K., RENILDA, J. A. & JEBITHA, G. B. 2014. Color Image Segmentation using IMOWT with 2D Histogram Grouping.

ANEJA, K., LAGUZET, F., LACASSAGNE, L. & MERIGOT, A. 2009. Video-rate image segmentation by means of region splitting and merging. *Proc. Signal and Image Processing Applications***,** 437-442.

ARBELAEZ, P., MAIRE, M., FOWLKES, C. & MALIK, J. 2011. Contour Detection and Hierarchical Image Segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* 33**,** 898-916.

ARULAMPALAM, M. S., MASKELL, S., GORDON, N. & CLAPP, T. 2002. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing,* 50**,** 174-188.

AYCARD, O., SPALANZANI, A., BURLET, J., FULGENZI, C., VU, T. D., RAULO, D. & YGUEL, M. Pedestrians Tracking Using Offboard Cameras. 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, 9-15 Oct. 2006 2006. 513-518.

BABIC, Z., PILIPOVIC, R., RISOJEVIC, V. & MIRJANIC, G. 2016. POLLEN BEARING HONEY BEE DETECTION IN HIVE ENTRANCE VIDEO RECORDED BY REMOTE EMBEDDED SYSTEM FOR POLLINATION MONITORING. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences,* 3.

BALLARD, D. H. 1981. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition,* 13**,** 111-122.

BAOJUN, Q., TAO, W., BIN, D. & HANGEN, H. Fast detection of small infrared objects in maritime scenes using local minimum patterns. Image Processing (ICIP), 2011 18th IEEE International Conference on, 11-14 Sept. 2011 2011. 3553-3556.

BARRON, J. L., FLEET, D. J. & BEAUCHEMIN, S. S. 1994. Performance of optical flow techniques. *International Journal of Computer Vision,* 12**,** 43-77.

BEAUCHEMIN, S. S. & BARRON, J. L. 1995. The computation of optical flow. *ACM Comput. Surv.,* 27**,** 433-466.

BENGIO, Y. 2009. Learning Deep Architectures for AI. *Foundations and Trends® in Machine Learning,* 2**,** 1-127.

BHATTACHARYA, P., BISWAS, A. & MAITY, S. 2014. Wavelets-Based Clustering Techniques for Efficient Color Image Segmentation. *In:* KUMAR KUNDU, M., MOHAPATRA, D. P., KONAR, A. & CHAKRABORTY, A. (eds.) *Advanced Computing, Networking and Informatics-Volume 1.* Springer International Publishing.

BIESMEIJER, J., ROBERTS, S., REEMER, M., OHLEMÜLLER, R., EDWARDS, M., PEETERS, T., SCHAFFERS, A., POTTS, S., KLEUKERS, R. & THOMAS, C. 2006. Parallel declines in pollinators and insect-pollinated plants in Britain and the Netherlands. *Science,* 313**,** 351-354.

BISHOP, C. M. 2012. Pattern recognition and machine learning, 2006. *대한토목학회지,* 60**,** 78-78.

BOTTOU, L. Large-Scale Machine Learning with Stochastic Gradient Descent. 2010 Heidelberg. Physica-Verlag HD, 177-186.

BOUWMANS, T. & EL BAF, F. 2009. Modeling of dynamic backgrounds by type-2 fuzzy Gaussians mixture models. *MASAUM Journal of of Basic and Applied Sciences,* 1**,** 265-276.

BOUWMANS, T., EL BAF, F. & VACHON, B. 2008. Background modeling using mixture of gaussians for foreground detection-a survey. *Recent Patents on Computer Science,* 1**,** 219-237.

BRODSCHNEIDER, R. & CRAILSHEIM, K. 2010. Nutrition and health in honey bees. *Apidologie,* 41**,** 278-294.

CAMPBELL, J., MUMMERT, L. & SUKTHANKAR, R. 2008. Video monitoring of honey bee colonies at the hive entrance. *Visual observation & analysis of animal & insect behavior, ICPR,* 8**,** 1-4.

CARDANI, D. 2001. Adventures in hsv space. *Laboratorio de Robótica, Instituto Tecnológico Autónomo de México*.

CELENK, M. 1990. A color clustering technique for image segmentation. *Computer Vision, Graphics, and Image Processing,* 52**,** 145-170.

CHANGJIANG, Y., DURAISWAMI, R. & DAVIS, L. Fast multiple object tracking via a hierarchical particle filter. Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on, 17-21 Oct. 2005 2005. 212-219 Vol. 1.

CHEN, C., LIANG, J., ZHAO, H., HU, H. & TIAN, J. 2009. Frame difference energy image for gait recognition with incomplete silhouettes. *Pattern Recognition Letters,* 30**,** 977-984.

CHENG, H. D., JIANG, X. H., SUN, Y. & WANG, J. 2001. Color image segmentation: advances and prospects. *Pattern Recognition,* 34**,** 2259-2281.

CHIRON, G., GOMEZ-KRÄMER, P. & MÉNARD, M. Outdoor 3d acquisition system for small and fast targets. application to honeybee monitoring at the beehive entrance. Workshop Proceedings of GEODIFF 2013, In conjuction with VISIGRAPP 2013, 2013a. 10-19.

CHIRON, G., GOMEZ-KRÄMER, P., MÉNARD, M. & REQUIER, F. 2013b. 3D Tracking of Honeybees Enhanced by Environmental Context. *In:* PETROSINO, A. (ed.) *Image Analysis and Processing – ICIAP 2013.* Springer Berlin Heidelberg.

CHITADE, A. Z. & KATIYAR, S. 2010. Colour based image segmentation using k-means clustering. *International Journal of Engineering Science and Technology,* 2**,** 5319-5325.

CHIU, S.-H. & LIAW, J.-J. 2005. An effective voting method for circle detection. *Pattern Recognition Letters,* 26**,** 121-133.

CHOI, Y. S., ZAIJUN, P., KIM, S. W., KIM, T. H. & PARK, C. B. Salient Motion Information Detection Technique Using Weighted Subtraction Image and Motion Vector. Hybrid Information Technology, 2006. ICHIT '06. International Conference on, 9-11 Nov. 2006 2006. 263-269.

CIREŞAN, D., MEIER, U., MASCI, J. & SCHMIDHUBER, J. 2012. Multi-column deep neural network for traffic sign classification. *Neural Networks,* 32**,** 333-338.

CRANDALL, D., FELZENSZWALB, P. & HUTTENLOCHER, D. Spatial priors for part-based recognition using statistical models. Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, 20-25 June 2005 2005. 10-17 vol. 1.

CUCCHIARA, R., GRANA, C., PICCARDI, M. & PRATI, A. 2003. Detecting moving objects, ghosts, and shadows in video streams. *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* 25**,** 1337-1342.

DAVIS, J. W. & SHARMA, V. 2004. Robust background-subtraction for person detection in thermal imagery. *IEEE Int. Wkshp. on Object Tracking and Classification Beyond the Visible Spectrum*.

DEARDEN, A., DEMIRIS, Y. & GRAU, O. Tracking football player movement from a single moving camera using particle filters. Proceedings of the 3rd European Conference on Visual Media Production (CVMP-2006), 2006. 29-37.

DESAI, U. B., MERCHANT, S. N., ZAVERI, M., AJISHNA, G., PUROHIT, M. & PHANISH, H. S. 2005. Small Object Detection and Tracking: Algorithm, Analysis and Application. *In:* PAL, S., BANDYOPADHYAY, S. & BISWAS, S. (eds.) *Pattern Recognition and Machine Intelligence.* Springer Berlin Heidelberg.

DEY, S., BHATTACHARYA, B. B., KUNDU, M. K. & ACHARYA, T. 2002. A simple architecture for computing moments and orientation of an image. *Fundam. Inf.,* 52**,** 285-295.

DUDA, R. O. & HART, P. E. 1972. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM,* 15**,** 11-15.

DUTTA, J. & PAL, S. 2015. A note on Hungarian method for solving assignment problem. *Journal of Information and Optimization Sciences,* 36**,** 451-459.

EILERS, E. J., KREMEN, C., GREENLEAF, S. S., GARBER, A. K. & KLEIN, A.-M. 2011. Contribution of pollinator-mediated crops to nutrients in the human food supply. *PLoS one,* 6**,** e21363.

EL BAF, F., BOUWMANS, T. & VACHON, B. Fuzzy statistical modeling of dynamic backgrounds for moving object detection in infrared videos. Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on, 20-25 June 2009 2009. 60-65.

ELGAMMAL, A., HARWOOD, D. & DAVIS, L. 2000. Non-parametric model for background subtraction. *Computer Vision—ECCV 2000.* Springer.

ERHAN, D., SZEGEDY, C., TOSHEV, A. & ANGUELOV, D. Scalable object detection using deep neural networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014. 2147-2154.

FAWCETT, T. 2006. An introduction to ROC analysis. *Pattern recognition letters,* 27**,** 861-874.

FELZENSZWALB, P. F., GIRSHICK, R. B., MCALLESTER, D. & RAMANAN, D. 2010. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 32**,** 1627-1645.

FENG, N., DELHOMME, D., LECUN, Y., PIANO, F., BOTTOU, L. & BARBANO, P. E. 2005. Toward automatic phenotyping of developing embryos from videos. *IEEE Transactions on Image Processing,* 14**,** 1360-1371.

FLEYEH, H. Shadow and highlight invariant colour segmentation algorithm for traffic signs. Cybernetics and Intelligent Systems, 2006 IEEE Conference on, 2006. IEEE, 1-7.

FRIEDMAN, N. & RUSSELL, S. Image segmentation in video sequences: A probabilistic approach. Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence, 1997. Morgan Kaufmann Publishers Inc., 175-181.

FULKERSON, B., VEDALDI, A. & SOATTO, S. Class segmentation and object localization with superpixel neighborhoods. Computer Vision, 2009 IEEE 12th International Conference on, Sept. 29 2009-Oct. 2 2009 2009. 670-677.

GABORATORY, B. B. 1978. Chemical composition and nutritive value of bee-collected and bee-stored pollen.

GALLAI, N., SALLES, J.-M., SETTELE, J. & VAISSIÈRE, B. E. 2009. Economic valuation of the vulnerability of world agriculture confronted with pollinator decline. *Ecological Economics,* 68**,** 810-821.

GANESAN, P. & RAJINI, V. YIQ color space based satellite image segmentation using modified FCM clustering and histogram equalization. Advances in Electrical Engineering (ICAEE), 2014 International Conference on, 9-11 Jan. 2014 2014. 1-5.

GAO, J., KOSAKA, A. & KAK, A. C. 2005. A multi-Kalman filtering approach for video tracking of human-delineated objects in cluttered environments. *Computer Vision and Image Understanding,* 99**,** 1-57.

GARCIA, C. & DELAKIS, M. 2004. Convolutional face finder: a neural architecture for fast and robust face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 26**,** 1408-1423.

GIBSON, J. J. 1950. The perception of the visual world.

GIRSHICK, R. Fast r-cnn. Proceedings of the IEEE international conference on computer vision, 2015. 1440-1448.

GIRSHICK, R., DONAHUE, J., DARRELL, T. & MALIK, J. Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition, 2014. 580-587.

GIRSHICK, R. B., FELZENSZWALB, P. F. & MCALLESTER, D. 2012. Discriminatively trained deformable part models, release 5.

GONZALEZ, R. C. & WOODS, R. E. 2011. *Digital Image Processing Third Edition,* India, Dorling Kindersley Pvt. Ltd.

GOULD, S., GAO, T. & KOLLER, D. Region-based Segmentation and Object Detection. NIPS, 2009. 2.

GRITZMAN, A., RUBIN, D. & PANTANOWITZ, A. 2014. Comparison of colour transforms used in lip segmentation algorithms. *Signal, Image and Video Processing***,** 1-11.

HA, J.-E. & LEE, W.-H. 2010. Foreground objects detection using multiple difference images. *Optical Engineering,* 49**,** 047201-047201-5.

HAN, B., COMANICIU, D. & DAVIS, L. Sequential kernel density approximation through mode propagation: applications to background modeling. proc. ACCV, 2004. 818-823.

HANBURY, A. 2007. Image Segmentation by Region Based and Watershed Algorithms. *Wiley Encyclopedia of Computer Science and Engineering.* John Wiley & Sons, Inc.

HANNUN, A., CASE, C., CASPER, J., CATANZARO, B., DIAMOS, G., ELSEN, E., PRENGER, R., SATHEESH, S., SENGUPTA, S. & COATES, A. 2014. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

HARVILLE, M. 2002. A framework for high-level feedback to adaptive, per-pixel, mixture-of-gaussian background models. *Computer Vision—ECCV 2002.* Springer.

HASSANAT, A. B., ALKASASSBEH, M., AL-AWADI, M. & ESRA'A, A. 2016. Color-based object segmentation method using artificial neural network. *Simulation Modelling Practice and Theory*.

HE, K., GKIOXARI, G., DOLLÁR, P. & GIRSHICK, R. Mask r-cnn. Computer Vision (ICCV), 2017 IEEE International Conference on, 2017. IEEE, 2980-2988.

HE, K. & SUN, J. Convolutional neural networks at constrained time cost. Proceedings of the IEEE conference on computer vision and pattern recognition, 2015. 5353-5360.

HE, K., ZHANG, X., REN, S. & SUN, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. 2014 Cham. Springer International Publishing, 346-361.

HE, K., ZHANG, X., REN, S. & SUN, J. Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016. 770-778.

HENDRIKS, C., YU, Z., LECOCQ, A., BAKKER, T., LOCKE, B. & TERENIUS, O. Identifying all individuals in a honeybee hive-progress towards mapping all social interactions. Workshop Vis. Observation Anal. Anim. Insect Behav. ICPR, 2012.

HORPRASERT, T., HARWOOD, D. & DAVIS, L. S. A statistical approach for real-time robust background subtraction and shadow detection. IEEE ICCV, 1999. 1-19.

HOUGH, P. V. 1962. Method and means for recognizing complex patterns.

HOWARD, A. G., ZHU, M., CHEN, B., KALENICHENKO, D., WANG, W., WEYAND, T., ANDREETTO, M. & ADAM, H. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

HSIEH, F.-Y., HAN, C.-C., WU, N.-S., CHUANG, T. C. & FAN, K.-C. 2006. A novel approach to the detection of small objects with low contrast. *Signal Processing,* 86**,** 71-83.

HUANG, Z. 2010. *Honey Bee Nutrition* [Online]. Available: http://www.beeccdcap.uga.edu/documents/CAPArticle10.html [Accessed June 23 2014].

IANDOLA, F. N., HAN, S., MOSKEWICZ, M. W., ASHRAF, K., DALLY, W. J. & KEUTZER, K. 2016. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*.

IKOMA, N., HARAGUCHI, Y. & HASEGAWA, H. On an evaluation of tracking performance improvement by SMC-PHD filter with intensity image of pedestrians detection over on-board camera using neural network. 2014 World Automation Congress (WAC), 3-7 Aug. 2014 2014. 273-278.

ILEA, D. E. & WHELAN, P. F. An adaptive unsupervised segmentation algorithm based on color-texture coherence. IEEE Trans. Image Process, 2005. Citeseer.

ILLINGWORTH, J. & KITTLER, J. 1988. A survey of the hough transform. *Computer Vision, Graphics, and Image Processing,* 44**,** 87-116.

INDUSTRIES, M. F. P. 2013. *2013 apiculture monitoring programme* [Online]. Available: http://nba.org.nz/uploads/MPI%202013%20Apiculture%20Report.pdf [Accessed June 23 2014].

IQBAL, A., SHAH, S. W. & KHAN, S. 2014. Non-Linear Moving Target Tracking: A Particle Filter Approach. *International Journal of Computer and Communication System Engineering (IJCCSE),* 1.

JACQUES, J. C. S., JUNG, C. R. & RAUPP MUSSE, S. Background subtraction and shadow detection in grayscale video sequences. Computer Graphics and Image Processing, 2005. SIBGRAPI 2005. 18th Brazilian Symposium on, 2005. IEEE, 189-196.

JAVED, O., SHAFIQUE, K. & SHAH, M. A hierarchical approach to robust background subtraction using color and gradient information. Motion and Video Computing, 2002. Proceedings. Workshop on, 5-6 Dec. 2002 2002. 22-27.

JOHNSON, R. & CORN, M. L. 2014. Bee Health: Background and Issues for Congress.

KAVITHA, K. & TEJASWINI, A. 2012. VIBE: Background Detection and Subtraction for Image Sequences in Video. *International Journal of Computer Science and Information Technologies,* 3**,** 5223-5226.

KIMURA, T., OHASHI, M., CRAILSHEIM, K., SCHMICKL, T., ODAKA, R. & IKENO, H. Tracking of Multiple Honey Bees on a Flat Surface. Emerging Trends in Engineering and Technology (ICETET), 2012 Fifth International Conference on, 2012. IEEE, 36-39.

KIMURA, T., OHASHI, M., OKADA, R. & IKENO, H. 2011. A new approach for the simultaneous tracking of multiple honeybees for analysis of hive behavior. *Apidologie,* 42**,** 607-617.

KLATT, B. K., HOLZSCHUH, A., WESTPHAL, C., CLOUGH, Y., SMIT, I., PAWELZIK, E. & TSCHARNTKE, T. 2014. Bee pollination improves crop quality, shelf life and commercial value. *Proceedings of the Royal Society B: Biological Sciences,* 281**,** 20132440.

KLEIN, A.-M., VAISSIERE, B. E., CANE, J. H., STEFFAN-DEWENTER, I., CUNNINGHAM, S. A., KREMEN, C. & TSCHARNTKE, T. 2007. Importance of pollinators in changing landscapes for world crops. *Proceedings of the Royal Society B: Biological Sciences,* 274**,** 303-313.

KNAUER, U., HIMMELSBACH, M., WINKLER, F., ZAUTKE, F., BIENEFELD, K. & MEFFERT, B. 2005. Application of an adaptive background model for monitoring honeybees.

KNAUER, U. & MEFFERT, B. Evaluation based combining of classifiers for monitoring honeybees. Applications of Computer Vision (WACV), 2009 Workshop on, 7-8 Dec. 2009 2009. 1-6.

KNAUER, U. & MEFFERT, B. Fast computation of region homogeneity with application in a surveillance task. Proceedings of ISPRS Commission, 2010. 337-342.

KOTOULAS, L. & ANDREADIS, I. Image analysis using moments. 5th Int. Conf. on Technology and Automation, Thessaloniki, Greece, 2005. 360-364.

KRIZHEVSKY, A., SUTSKEVER, I. & HINTON, G. E. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 2012. 1097-1105.

KUANG, X., SANG, N. & GAO, C. 2016. Object localization with mid-level part detectors. *Neurocomputing*.

KUHN, H. 2005. The Hungarian method for the assignment problem. *Naval Research Logistics (NRL),* 52**,** 7-21.

KUMAR, A. An empirical study of selection of the appropriate color space for skin detection: A case of face detection in color images. Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014 International Conference on, 7-8 Feb. 2014 2014. 725-730.

LAN, Y., JI, Z., GAO, J. & YAOWEI, W. Robot fish detection based on a combination method of three-frame-difference and background subtraction. Control and Decision Conference (2014 CCDC), The 26th Chinese, May 31 2014-June 2 2014 2014. 3905-3909.

LAWRENCE, S., GILES, C. L., AH CHUNG, T. & BACK, A. D. 1997. Face recognition: a convolutional neural-network approach. *IEEE Transactions on Neural Networks,* 8**,** 98-113.

LECUN, Y., BENGIO, Y. & HINTON, G. 2015. Deep learning. *Nature,* 521**,** 436.

LECUN, Y., BOTTOU, L., BENGIO, Y. & HAFFNER, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE,* 86**,** 2278-2324.

LI, P., ZHANG, T. & MA, B. 2004. Unscented Kalman filter for visual curve tracking. *Image and Vision Computing,* 22**,** 157-164.

LIEW, L. H., LEE, B. Y. & CHAN, M. Cell detection for bee comb images using circular Hough transformation. Science and Social Research (CSSR), 2010 International Conference on, 2010. IEEE, 191-195.

LIN, H.-H., CHUANG, J.-H. & LIU, T.-L. 2011. Regularized background adaptation: a novel learning rate control scheme for Gaussian mixture modeling. *Image Processing, IEEE Transactions on,* 20**,** 822-836.

LIN, T.-Y., MAIRE, M., BELONGIE, S., HAYS, J., PERONA, P., RAMANAN, D., DOLLÁR, P. & ZITNICK, C. L. Microsoft COCO: Common Objects in Context. 2014 Cham. Springer International Publishing, 740-755.

LIPTON, A. J., FUJIYOSHI, H. & PATIL, R. S. Moving target classification and tracking from real-time video. Applications of Computer Vision, 1998. WACV'98. Proceedings., Fourth IEEE Workshop on, 1998. IEEE, 8-14.

LITTMANN, E. & RITTER, H. 1997. Adaptive color segmentation-a comparison of neural and statistical methods. *Neural Networks, IEEE Transactions on,* 8**,** 175-185.

LIU, W., ANGUELOV, D., ERHAN, D., SZEGEDY, C., REED, S., FU, C.-Y. & BERG, A. C. SSD: Single Shot MultiBox Detector. 2016 Cham. Springer International Publishing, 21-37.

LO, B. & VELASTIN, S. Automatic congestion detection system for underground platforms. Intelligent Multimedia, Video and Speech Processing, 2001. Proceedings of 2001 International Symposium on, 2001. IEEE, 158-161.

LONG, J., SHELHAMER, E. & DARRELL, T. Fully convolutional networks for semantic segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition, 2015. 3431-3440.

LU, N., WANG, J., WU, Q. & YANG, L. 2008. An improved motion detection method for real-time surveillance. *IAENG International Journal of Computer Science,* 35**,** 1-10.

LU, S., TSECHPENAKIS, G., METAXAS, D. N., JENSEN, M. L. & KRUSE, J. Blob Analysis of the Head and Hands: A Method for Deception Detection. HICSS, 2005. 20.3.

LU, W. & TAN, J. 2008. Detection of incomplete ellipse in images with strong noise by iterative randomized Hough transform (IRHT). *Pattern Recognition,* 41**,** 1268-1279.

MA, C. & LIU, C. 2015. Two dimensional hashing for visual tracking. *Computer Vision and Image Understanding,* 135**,** 83-94.

MA, Y., ZHANG, Z. & LI, Y. 2015. Moving Vehicles Detection Based on Improved Gaussian Mixture Model.

MAAS, A. L., HANNUN, A. Y. & NG, A. Y. Rectifier nonlinearities improve neural network acoustic models. Proc. icml, 2013. 3.

MIGLIORE, D. A., MATTEUCCI, M. & NACCARI, M. 2006. A revaluation of frame difference in fast and robust motion detection. *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks.* Santa Barbara, California, USA: ACM.

MIN, H., JIA, W., WANG, X.-F., ZHAO, Y., HU, R.-X., LUO, Y.-T., XUE, F. & LU, J.-T. 2015. An Intensity-Texture model based level set method for image segmentation. *Pattern Recognition,* 48**,** 1547-1562.

MOHAPATRA, S., KAR, A., DASH, S., MOHANTY, S. & SWAIN, P. 2014. A Novel Approach to Face Detection Using Advanced Support Vector Machine. *Intelligent Computing, Networking, and Informatics.* Springer.

MURPHY, K. P. 2012. *Machine learning, a probabilistic perspective,* Cambridge, Massachusetts, The MIT Press.

NAGI, J., DUCATELLE, F., CARO, G. A. D., CIRE, D., X015F, AN, MEIER, U., GIUSTI, A., NAGI, F., SCHMIDHUBER, J. & GAMBARDELLA, L. M. Max-pooling convolutional neural networks for vision-based hand gesture recognition. 2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA), 16-18 Nov. 2011 2011. 342-347.

NAIR, P. & SAUNDERS, A. 1996. Hough transform based ellipse detection algorithm. *Pattern Recognition Letters,* 17**,** 777-784.

NAIR, V. & HINTON, G. E. Rectified linear units improve restricted boltzmann machines. Proceedings of the 27th international conference on machine learning (ICML-10), 2010. 807-814.

NANDASHRI, D. & SMITHA, P. 2015a. An Efficient Tracking of Multi Object Visual Motion using Hungarian Method. *International Journal of Engineering Research & Technology (IJERT),* 4.

NANDASHRI, D. & SMITHA, P. 2015b. An Visual Motion Tracking of Multi-Object Using Hungarian Method. *International Journal of Innovative Research and Development|| ISSN 2278–0211,* 4.

NANDASHRI, D. & SMITHA, P. 2015c. An Visual Motion Tracking of Multi-Object Using Hungarian Method. *International Journal of Innovative Research and Development,* 4.

NAUG, D. 2009. Nutritional stress due to habitat loss may explain recent honeybee colony collapses. *Biological Conservation,* 142**,** 2369-2372.

NIRGUDE, R. & JAIN, S. 2014. COLOR IMAGE SEGMENTATION WITH K MEANS CLUSTERING AND DYNAMIC REGION MERGING. *Sai Om Journal of Science, Engineering & Technology: A Peer Reviewed National Journal (Online ISSN 2347-7547),* 1**,** 1-10.

OHLANDER, R., PRICE, K. & REDDY, D. R. 1978. Picture segmentation using a recursive region splitting method. *Computer Graphics and Image Processing,* 8**,** 313-333.

OHTA, Y.-I., KANADE, T. & SAKAI, T. 1980. Color information for region segmentation. *Computer Graphics and Image Processing,* 13**,** 222-241.

OSADCHY, M., CUN, Y. L. & MILLER, M. L. 2007. Synergistic face detection and pose estimation with energy-based models. *Journal of Machine Learning Research,* 8**,** 1197-1215.

OTSU, N. 1975. A threshold selection method from gray-level histograms. *Automatica,* 11**,** 23-27.

PAN, S. J. & YANG, Q. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering,* 22**,** 1345-1359.

PAVLIDIS, T. & LIOW, Y. T. 1990. Integrating region growing and edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* 12**,** 225-233.

PENG, X. 2015. Combine color and shape in real-time detection of texture-less objects. *Computer Vision and Image Understanding,* 135**,** 31-48.

QI, B., WU, T., HE, H. & HU, T. 2011. Real-Time Detection of Small Surface Objects Using Weather Effects. *In:* KIMMEL, R., KLETTE, R. & SUGIMOTO, A. (eds.) *Computer Vision – ACCV 2010.* Springer Berlin Heidelberg.

QI, Y., ZHANG, G. & LI, Y. Fast Segmentation Algorithm Based on Texture. Proceedings of the Second International Conference on Mechatronics and Automatic Control, 2015. Springer, 541-547.

RAKIBE, R. S. & PATIL, B. D. 2013. Background subtraction algorithm based human motion detection. *International Journal of scientific and research publications,* 3.

REDMON, J., DIVVALA, S., GIRSHICK, R. & FARHADI, A. You only look once: Unified, real-time object detection. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016. 779-788.

REDMON, J. & FARHADI, A. 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

REN, S., HE, K., GIRSHICK, R. & SUN, J. Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 2015. 91-99.

REN, S., HE, K., GIRSHICK, R. & SUN, J. 2017. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis & Machine Intelligence***,** 1137-1149.

REZENDE, D. J., MOHAMED, S. & WIERSTRA, D. 2014. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*.

RUSSAKOVSKY, O., DENG, J., SU, H., KRAUSE, J., SATHEESH, S., MA, S., HUANG, Z., KARPATHY, A., KHOSLA, A., BERNSTEIN, M., BERG, A. C. & FEI-FEI, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision,* 115**,** 211-252.

SANCHEZ-GARCIA, A. J., RIOS-FIGUEROA, H. V., MARIN-HERNANDEZ, A. & ACOSTA-MESA, H. G. Tracking and prediction of motion of segmented regions using the Kalman filter. Electronics, Communications and Computers (CONIELECOMP), 2014 International Conference on, 26-28 Feb. 2014 2014. 88-93.

SCHMIDHUBER, J. 2015. Deep learning in neural networks: An overview. *Neural Networks,* 61**,** 85-117.

SEERAM, N. P. 2008. Berry fruits: compositional elements, biochemical activities, and the impact of their intake on human health, performance, and disease. *Journal of agricultural and food chemistry,* 56**,** 627-629.

SERMANET, P., EIGEN, D., ZHANG, X., MATHIEU, M., FERGUS, R. & LECUN, Y. 2013a. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*.

SERMANET, P., KAVUKCUOGLU, K., CHINTALA, S. & LECUN, Y. Pedestrian detection with unsupervised multi-stage feature learning. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013b. 3626-3633.

SHADEED, W., ABU-AL-NADI, D. I. & MISMAR, M. J. Road traffic sign detection in color images. Electronics, Circuits and Systems, 2003. ICECS 2003. Proceedings of the 2003 10th IEEE International Conference on, 2003. IEEE, 890-893.

SHAFIE, A. A., HAFIZ, F. & ALI, M. 2009. Motion detection techniques using optical flow. *World Academy of Science, Engineering and Technology,* 56**,** 559-561.

SHANTAIYA, S., VERMA, K. & MEHTA, K. 2015. Multiple Object Tracking using Kalman Filter and Optical Flow. *European Journal of Advances in Engineering and Technology,* 2**,** 34-39.

SHUIGEN, W., ZHEN, C. & HUA, D. Motion detection based on temporal difference method and optical flow field. 2009 Second International Symposium on Electronic Commerce and Security, 2009. IEEE, 85-88.

SIANG TAN, K. & MAT ISA, N. A. 2011. Color image segmentation using histogram thresholding – Fuzzy C-means hybrid approach. *Pattern Recognition,* 44**,** 1-15.

SIMONYAN, K. & ZISSERMAN, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

SINGLA, N. 2014. Motion Detection Based on Frame Difference Method. *International Journal of Information & Computation Technology,* 4**,** 1559-1565.

SMITH, L. N. Cyclical learning rates for training neural networks. Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on, 2017. IEEE, 464-472.

SOBRAL, A. & VACAVANT, A. 2014. A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. *Computer Vision and Image Understanding,* 122**,** 4-21.

SRIVASTAVA, R. K., GREFF, K. & SCHMIDHUBER, J. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

STAUFFER, C. & GRIMSON, W. E. L. Adaptive background mixture models for real-time tracking. Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on., 1999 1999. 252 Vol. 2.

STAVELEY, J. P., LAW, S. A., FAIRBROTHER, A. & MENZIE, C. A. 2014. A causal analysis of observed declines in managed honey bees (Apis mellifera). *Human and Ecological Risk Assessment: An International Journal,* 20**,** 566-591.

STENGER, B., MENDONÇA, P. R. & CIPOLLA, R. Model-Based Hand Tracking Using an Unscented Kalman Filter. BMVC, 2001. 63-72.

SUBTIL, F. & RABILLOUD, M. 2015. An enhancement of ROC curves made them clinically relevant for diagnostic-test comparison and optimal-threshold determination. *Journal of Clinical Epidemiology,* 68**,** 752-759.

SZEGEDY, C., LIU, W., JIA, Y., SERMANET, P., REED, S., ANGUELOV, D., ERHAN, D., VANHOUCKE, V. & RABINOVICH, A. Going deeper with convolutions. Proceedings of the IEEE conference on computer vision and pattern recognition, 2015. 1-9.

TAN, R., HUO, H., QIAN, J. & FANG, T. 2006. Traffic video segmentation using adaptive-k gaussian mixture model. *Advances in Machine Vision, Image Processing, and Pattern Analysis.* Springer.

TANG, Y. 2013. Deep learning using linear support vector machines. *arXiv preprint arXiv:1306.0239*.

TASHITA, A., KOBASHI, S., MORI, Y., MORIMOTO, M., AIKAWA, S., YOSHIOKA, Y. & HATA, Y. Macrophage Tracking Using the Hungarian Algorithm in Time Lapse MR Images. 2015 7th International Conference on Emerging Trends in Engineering & Technology (ICETET), 18-20 Nov. 2015 2015. 169-173.

TELAGARAPU, P., RAO, M. V. N. & SURESH, G. A novel traffic-tracking system using morphological and Blob analysis. 2012 International Conference on Computing, Communication and Applications, 22-24 Feb. 2012 2012. 1-4.

THOU-HO, C., YU-FENG, L. & TSONG-YI, C. Intelligent Vehicle Counting Method Based on Blob Analysis in Traffic Surveillance. Innovative Computing, Information and Control, 2007. ICICIC '07. Second International Conference on, 5-7 Sept. 2007 2007. 238-238.

TIAN, J., QI, X., QU, L. & TANG, Y. 2016. New spectrum ratio properties and features for shadow detection. *Pattern Recognition,* 51**,** 85-96.

TOMPSON, J., GOROSHIN, R., JAIN, A., LECUN, Y. & BREGLER, C. Efficient object localization using convolutional networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015. 648-656.

TOSHEV, A., TASKAR, B. & DANIILIDIS, K. 2012. Shape-Based Object Detection via Boundary Structure Segmentation. *International Journal of Computer Vision,* 99**,** 123-146.

TOYAMA, K., KRUMM, J., BRUMITT, B. & MEYERS, B. Wallflower: Principles and practice of background maintenance. Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on, 1999. IEEE, 255-261.

TREMEAU, A. & BOREL, N. 1997. A region growing and merging algorithm to color segmentation. *Pattern Recognition,* 30**,** 1191-1203.

TSE-WEI, C., YI-LING, C. & SHAO-YI, C. Fast image segmentation based on K-Means clustering with histograms in HSV color space. Multimedia Signal Processing, 2008 IEEE 10th Workshop on, 8-10 Oct. 2008 2008. 322-325.

VAILLANT, R., MONROCQ, C. & CUN, Y. L. 1994. Original approach for the localisation of objects in images. *IEE Proceedings - Vision, Image and Signal Processing* [Online], 141. Available: http://digital-library.theiet.org/content/journals/10.1049/ip-vis_19941301.

VAN GERVEN, M. & BOHTE, S. 2018. *Artificial neural networks as models of neural information processing*, Frontiers Media SA.

VANENGELSDORP, D. & MEIXNER, M. D. 2010. A historical review of managed honey bee populations in Europe and the United States and the factors that may affect them. *Journal of Invertebrate Pathology,* 103, Supplement**,** S80-S95.

VARD, A., JAMSHIDI, K. & MOVAHHEDINIA, N. 2012. Small object detection in cluttered image using a correlation based active contour model. *Pattern Recognition Letters,* 33**,** 543-553.

VEERARAGHAVAN, A., CHELLAPPA, R. & SRINIVASAN, M. 2008. Shape-and-Behavior Encoded Tracking of Bee Dances. *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* 30**,** 463-476.

VERMAAK, J., DOUCET, A. & PEREZ, P. Maintaining multimodality through mixture tracking. Proceedings Ninth IEEE International Conference on Computer Vision, 13-16 Oct. 2003 2003. 1110-1116 vol.2.

VIOLA, P. & JONES, M. J. 2004. Robust Real-Time Face Detection. *International Journal of Computer Vision,* 57**,** 137-154.

VITABILE, S., POLLACCIA, G., PILATO, G. & SORBELLO, F. Road signs recognition using a dynamic pixel aggregation technique in the HSV color space. Image Analysis and Processing, 2001. Proceedings. 11th International Conference on, 26-28 Sept. 2001 2001. 572-577.

WANG, H., ONEATA, D., VERBEEK, J. & SCHMID, C. 2015a. A robust and efficient video representation for action recognition. *International Journal of Computer Vision*, 1-20.

WANG, K., LIANG, Y., XING, X. & ZHANG, R. 2015b. Target Detection Algorithm Based on Gaussian Mixture Background Subtraction Model. *In:* DENG, Z. & LI, H. (eds.) *Proceedings of the 2015 Chinese Intelligent Automation Conference.* Springer Berlin Heidelberg.

WANG, P., SHEN, X., LIN, Z., COHEN, S., PRICE, B. & YUILLE, A. L. Joint object and part segmentation using deep learned potentials. Proceedings of the IEEE International Conference on Computer Vision, 2015c. 1573-1581.

WEI, J. 2013. Small moving object detection from Infra-Red sequences. *International Journal of Image and Graphics,* 13.

WELCH, G. & BISHOP, G. 2006. An introduction to the kalman filter. Department of Computer Science, University of North Carolina. Chapel Hill, NC, unpublished manuscript.

WENG, S.-K., KUO, C.-M. & TU, S.-K. 2006. Video object tracking using adaptive Kalman filter. *Journal of Visual Communication and Image Representation,* 17**,** 1190-1208.

WILLIAMS, I. 1994. The dependence of crop production within the European Union on pollination by honey bees. *Agricultural Zoology Reviews (United Kingdom)*.

WU, B. & NEVATIA, R. 2007. Detection and Tracking of Multiple, Partially Occluded Humans by Bayesian Combination of Edgelet based Part Detectors. *International Journal of Computer Vision,* 75**,** 247-266.

XANGDONG, X., SUDHAKAR, R. & HANQI, Z. 1995. Real-time eye feature tracking from a video image sequence using Kalman filter. *Systems, Man and Cybernetics, IEEE Transactions on,* 25**,** 1568-1577.

YONG-MEI, Z., SHENG-YI, J. & MEI-LIN, Y. A Region-Based Image Segmentation Method with Mean-Shift Clustering Algorithm. Fuzzy Systems and Knowledge Discovery, 2008. FSKD '08. Fifth International Conference on, 18-20 Oct. 2008 2008. 366-370.

YU, P. & POH, C. L. 2015. Region-based snake with edge constraint for segmentation of lymph nodes on CT images. *Computers in Biology and Medicine,* 60**,** 86-91.

YUAN, J., WANG, D. & CHERIYADAT, A. M. 2015. Factorization-Based Texture Segmentation. *IEEE Transactions on Image Processing,* 24**,** 3488-3497.

ZEALAND, T. N. B. A. O. N. 2013. *Honey bees have been kept in New Zealand for over 150 years* [Online]. Available: http://nba.org.nz/about-bees/interesting-facts [Accessed June 23 2014].

ZENG, J., XIE, L. & LIU, Z.-Q. 2008. Type-2 fuzzy Gaussian mixture models. *Pattern Recognition,* 41**,** 3636-3643.

ZHANG, B., LI, Z., PERINA, A., BUE, A. D. & MURINO, V. Adaptive Local Movement Modelling for Object Tracking. 2015 IEEE Winter Conference on Applications of Computer Vision, 5-9 Jan. 2015 2015. 25-32.

ZHANG, S., CHAN, S. C., LIAO, B. & TSUI, K. M. A new visual object tracking algorithm using Bayesian Kalman filter. 2014 IEEE International Symposium on Circuits and Systems (ISCAS), 1-5 June 2014 2014. 522-525.

ZHEN, H., YEWEI, L. & JINJIANG, L. Image Segmentation Algorithm Based on Improved Visual Attention Model and Region Growing. Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference on, 23-25 Sept. 2010 2010. 1-4.

ZHU, W., ZENG, N. & WANG, N. 2010. Sensitivity, specificity, accuracy, associated confidence interval and ROC analysis with practical SAS implementations. *NESUG proceedings: health care and life sciences, Baltimore, Maryland,* 19.

ZHU, Y.-F. 2011. Moving Objects Detection and Segmentation Based on Background Subtraction and Image Over-Segmentation. *JSW,* 6**,** 1361-1367.

ZIA, K., BALCH, T. & DELLAERT, F. A Rao-Blackwellized particle filter for EigenTracking. Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on, 27 June-2 July 2004 2004. II-980-II-986 Vol.2.