FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Behaviour Monitoring from Honey Bees based on Video Analysis

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Gernot Winkler, BSc
Matrikelnummer 0929255

an der Fakultät für Informatik
der Technischen Universität Wien
Betreuung: PD Dipl.-Ing. Dr.techn. Martin Kampel

Wien, 19. November 2018

Gernot Winkler

Martin Kampel

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Behaviour Monitoring from Honey Bees based on Video Analysis

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Gernot Winkler, BSc
Registration Number 0929255

to the Faculty of Informatics
at the TU Wien
Advisor: PD Dipl.-Ing. Dr.techn. Martin Kampel

Vienna, 19th November, 2018

Gernot Winkler

Martin Kampel

Erklärung zur Verfassung der Arbeit

Gernot Winkler, BSc
Neunkirchnerstraße 17, 2732 Willendorf

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 19. November 2018

Gernot Winkler

Kurzfassung

Bienen sind ein wichtiger Bestandteil von Wirtschaft und Umwelt. In den letzten Jahren wurden Bienenstöcke durch Parasiten und anderen Umweltfaktoren bedroht. Aus diesem Grund steigt das Interesse Bienenstöcke automatisch zu überwachen. Mit Hilfe von Videodaten kann so eine Überwachung durchgeführt werden.

Diese Arbeit stellt eine Lösung vor, um alle Bienen am Eingang des Stocks gleichzeitig zu tracken und zu zählen, während sie durch eine künstliche Eingangsstruktur gehen. Unter der Verwendung der HSV- und HSL-Farbmodelle kann mit Hilfe von Mathematischer Morphologie eine genaue binäre Segmentierung erreicht werden. Durch Anwendung des K-Means Algorithmus können Cluster von Bienen getrennt werden und das Problem auf Tracking einzelner Bienen vereinfacht werden. Als Ergebnis der Zählung wird eine Fehlerrate von 6.86% auf Extremdaten mit einer Bildrate von rund 67 Bildern pro Sekunde auf einem Desktop-PC erreicht. Vier Datensätze, die mit drei verschiedenen Kameramodellen aufgenommen wurden, werden evaluiert. Ein Langzeittest, welcher Aufnahmen über einen Zeitraum von einer Woche (insgesamt 73 Stunden Videodaten) durchläuft, demonstriert die praktische Anwendbarkeit der Methode.

Abstract

Honey bees are an important part of the environment and economy. Recently bees are threatened by parasites and other environmental factors. This raises the need to automatically monitor bee hives. With the use of video data such a monitoring can be performed.

This thesis presents a way to use computer vision to automatically monitor a bee hive. The presented technique is able to simultaneously track and count bees at the hive entrance while they are walking through an artificial entrance structure. By using HSV and HSL color models together with mathematical morphology an accurate binary segmentation can be achieved. With the help of the k-means algorithm bee clusters can be resolved and broken down to single bee tracking. This allows to get counting results with less than 6.86% average error rate on extreme data while running at a framerate of around 67 frames per second on a desktop PC. Four datasets which where recorded with three different camera models have been evaluated. A long-time test which evaluates the algorithm with recordings of a whole week (73 hours in total) demonstrates the practical applicability of the presented method.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Beekeeping goes back far in the history of mankind, the first use of bee hives was probably between the years 5000 to 3000 BC[Cra13]. There are hints of hunting honey in the history of different cultures all around the world[Cra13]. Honey bees are used for production of honey and wax but also needed as pollinator for many plants[M+10]. Pollination by bees is an important part of our environment and has a huge influence on agriculture. Honey bees are the economically most important pollinator for monocultures and are used for pollination with managed bee hives when not enough wild bees are available[KVC+07]. Crops that contribute to around 35% of the worlds food production require animal pollination or are increased in productivity by it[KVC+07]. The amount of crops dependent on animal pollution has increased by over 300% within the last half century[AH09]. Also the global amount of managed honey bee hives has increased by only around 45% in this time span. This means the demand is growing faster than the supply.

In recent years a phenomenon called *colony collapse disorder* has appeared causing a rapid loss in adult worker bees[ESM+09][RC10][RBH13]. One of the reasons is believed to be caused by parasitic infections but other factors like pesticides can also have influences. The most dangerous threat is a mite called 'varroa destructor'[SGN00][RC10]. This mite transmits diseases that are capable of killing a bee hive. There is need for more research in this area and tools to help in analyzing the health of a bee hive can provide valuable insights to beekeepers or researchers.

Computer Vision has been used in different areas to provide automatic solutions based on visual data. Tasks that have been performed by humans can be entirely performed by automated systems. This allows to achieve lower costs and permanent observations. With modern computing power the complexity of data that can be handled in real-time

increases each year. With the help of Computer Vision a bee hive can be monitored and data can be automatically processed with minimal or no human interaction. Also using automated electronic systems to observe the hives does not disrupt the natural behavior of bees opposed to manual measurements.

1.2 Problem Statement

The threat of bees leads to the question on how observation and monitoring can be improved by computer vision. Measuring and counting the flight activity of bees is tedious to manually perform and can be automated, also measuring movement paths (trajectories) can be performed by a computer. To be able to use computer vision a way to acquire visual data is obviously required. This will be done by using a camera that records video data at the hive entrance but leads to the question on how exactly should a recording setup be built?

1.2.1 Data Acquisition

Using a hardware setup with a video camera to record the entrance of a bee hive is not a novelty and has already been done by multiple research projects. This means those existing hardware setups should be analyzed and evaluated on what advantages and disadvantages they posses.

Existing hardware setups can be categorized in two major groups.

- 3-Dimensional setups where recording is done with a camera mounted near the entrance and there is no restriction to the bees movement and bees are able to flying around. A landing pad may be placed in front of the hive. A possible camera placement is directly above the landing pad. Another option is to place the camera anywhere near the entrance and record the bees while flying in and out.

- 2-Dimensional setups where recording is performed with the help of an entrance structure where bees have to walk trough. The camera is placed in a way that the bees cannot move in and out without being recorded by it. Movement is restricted so bees cannot fly around freely limiting the movement of bees to 2D.

3-Dimensional setups

Many setups place the camera above the entrance of the bee hive. This has the advantage that the hive entrance is unchanged and the behavior of bees should not be disrupted. Such a setup has for example been used by Chiron[CGKMR13], Campbell[CMS08] and Yang[YC15]. Two of this setups are shown in figure1.1.

(a) The setup used by Campbell[CMS08] (b) The setup used by Yang[YC15]

Figure 1.1: Existing hardware solutions that record bees with a setup where bees can move in 3D

2-Dimensional setups

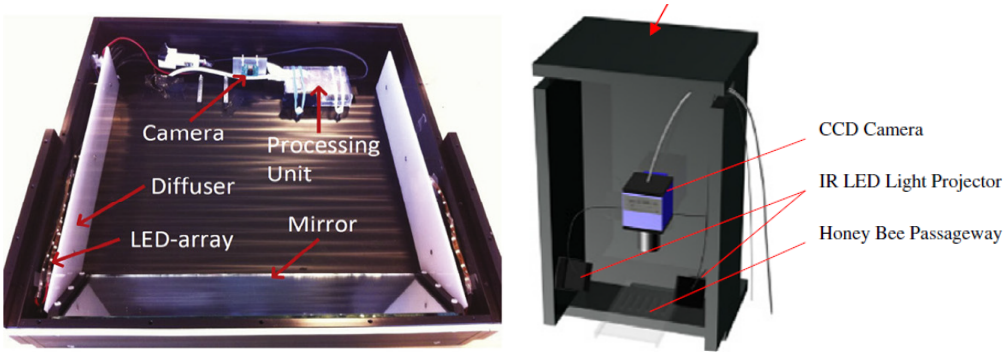
Since bees being able to freely move in 3D offer some challenges to algorithms like shadows or occlusion a way to counter this is by using a setup that restricts the bees movement to 2D while being recorded. For example Chen[CYJL12] and Tu[THKA16] use such setups, shown in figure 1.2.

The setup used by Tu et. al.[THKA16] uses a mirror and a glass pane to record bees from below while walking through the entrance tunnel. Chen et. al.[CYJL12] also use acrylic walls to separate the entrance tunnel into tracks to separate bees.

Comparisons

Using no entrance structure and recording the bees with a camera placed above requires little additional structure and can easily be applied to bee hives. On the other hand using an entrance structure requires more space and is expected to be more difficult to apply to an existing hive.

From an algorithmic point of view the two kind of setups have to deal with different problems. When bees can move in 3D phenomenons like occlusion have to be considered. When a background plate is used flying bees will cast shadows on it. When bees are



(a) The setup used by Tu[THKA16] (b) The setup used by Chen[CYJL12]

Figure 1.2: Existing hardware solutions that record bees with a setup where bees can move in 2D

limited to walking on the ground shadows are always directly below them and cannot be cast farther away. Segmenting the bees from the background will be more difficult in a 3D setup. In a 2D setup a homogenous background that has a high contrast to the bees can be chosen. If illumination is from above shadows will also be cast directly below the bees. When a transparent plate is placed in a height so that bees cannot move over each other, it is also possible to avoid occlusions. This means occlusions can be completely ignored by the algorithm in a 2D setup.

Another point that has to be considered is what data can be retrieved from the recorded videos. Counting the bees will be possible with both kind of setups, but the quality of bee images is likely better in a 2D setup as the camera records a closed off area and the camera can be precisely aligned to this. This means that bees will appear bigger on the images and therefore can be better used for further analysation like detecting parasite infections. Also with a custom built entrance structure it is possible to use artificial lights to create a homogenous lighting situation while a 3D setups has to deal with illumination changes over different times of day and weather situations.

Out of this reasons a 2D setup seems to be the superior choice when considering computer vision algorithms as difficulties like occlusion and a changing lighting situation can be avoided.

As part of the MIC-Cam[1] project a 2D setup[Sch18][SZKL16] is already present at the institute. The thesis from Schurischuster[Sch18] handles the problem of classifying images from bees if they contain varroa mites or not. A deep learning based approach is compared with a traditional image processing pipeline. This thesis solves the problem of real time monitoring of bees at the entrance and in the process providing input data for the classification.

[1] https://cvl.tuwien.ac.at/project/mic-cam/ (Accessed: 18. 10. 2018)

Setup specification

With the decision made to use this existing setup the next step is to define the constraints that can be made by the setup.

The camera is positioned with a top down view above the entrance area. The camera is aligned with the borders so that the left and right borders of the recorded images are aligned with the side walls. This makes it impossible for bees to leave or enter the recorded area on the sides. The camera height is also constant during operation. Exact height positioning and zooming factors are not defined but are not changed during operation. This means that bees have constant size during operation and parameters that are affected by the size of bees or camera height do not need to be adaptive during operation but need to be defined before starting the processing.

A transparent glass plate is placed as ceiling of the entrance area and the recording is performed through this plate. The plate is placed in a height that bees can not move over each other. This allows to define the condition that bees can not occlude each other. This further implies the algorithm can work in 2D as no 3 dimensional movement is possible. The glass plane is clear at the beginning but it is important to consider that accumulation of dirt happens during operation.

The bottom of the entrance area, and background of the video, is a gray monotonous plate. Like the glass plate this background will also accumulate dirt over time and can not be expected to be clean at all times.

The lighting situation is constant as the entrance area is illuminated by two rows of LED lights placed at the upper and lower borders. This means that only minor changes in illumination happen during the day. Also this causes reflections of the lights to be visible at the top and bottom borders of the video.

The camera records RGB videos at a framerate between 20 and 30 fps. The available test data has been recorded with different types of cameras. The resolution of the recordings is 1416x540 or 1920x1080 pixel. This means no depth or infrared data is available. Exact details of the used test data are presented in chapter5.

As for the behavior of the bees it can not be expected that bees will run in and out in a straight path. Bees can run around freely including running a loop or standing still for some time and forming clusters.

Figure 1.3 shows an example of the recorded video data. The position of the LED light artifacts and the dirt in the scene can be observed.

To conclude this the following conditions can be expected:

- the camera location is constant during operation
- the average size of a bee is constant during operation
- the lighting situation will only change subtle during operation

Figure 1.3: An example image of the recorded data

- bees can only enter at the top or bottom borders of the recorded area
- bees can not overlap and occlude each other

The following criteria can not be expected:

- The amount of bees that are visible at the same time is absolutely unknown. The area can be empty with zero visible bees or also be very crowded with less than 20% of the background being visible. This means no ratio between foreground and background area can be assumed.
- The glass plane can not be expected to be free of dirt at all times.
- The background can not be expected to be free of dirt at all times.
- Bees do not need to be moving. They can stand still.
- Bees do not need to move in a straight path from inside to outside or vice versa. They can roam freely and run in loops.

1.3 Aim of the Work

The goal of this work is to analyze existing approaches and based on that present a solution that can automatically monitor a bee hive with the aid of computer vision. Further it should be evaluated on accuracy, runtime and practical applicability.

The resulting software should count the bees and also retrieve the location and trajectories. Also this program should be able to perform the calculations with a runtime performance of at least 5 frames per second on a desktop PC. The testing system uses an Intel i7 4770k CPU. The program should also be oriented at long time operation and should be able to process recorded video data of a bee hive the whole day. As goal for the accuracy of this program a target of less than 10% average error rate is defined. This means that overall counting results of a day should be below 10% error rate.

To quantify the error rate it is required to perform an evaluation which requires ground truth data. While there are many hours of video data available no ground truth information is known. This means that ground truth data has to be acquired for evaluation. Evaluation is mandatory as it can not be verified if the defined accuracy requirements are met otherwise. To acquire the ground truth data a manual annotation is required as no automatic process to detect and count the bees is known.

The evaluation also consists of measuring the runtimes and comparing them with the target framerate of 5 fps or higher. The average frames per second over test video sequences used for evaluating the accuracy is used as measurement for the runtime.

The counting of bees consists of the amount of bees that entered and exited during the recorded video sequence that is processed. These are two separate counting values. With this information a difference can be observed and also the general activity of a hive can be measured during a specific time interval.

Retrieving the location and trajectories of the bees is not directly required to measure the activity as there are algorithms that can count the bees without tracking but these information provide additional data that can be used for further research. Detecting single bees allows to extract images of them and pass them to further processing like checking parasite infections.

1.4 Contributions of this thesis

This thesis presents a combination of algorithms that can monitor bee activity in real time and performs intensive evaluation on accuracy and runtimes with test data from different cameras and different resolutions.

- Six video sequences of the bee hive entrance have been fully annotated with the movement of each individual bee. This allows to get accurate ground truth data about counting and trajectories.

- It is shown that the HSV and HSL color models can be effective to segment bees from an artificial background. While this idea is not novel the evaluation is extensive and compares with other color models.

- Morphological opening was used to extract the bee shapes from an initial segmentation. The evaluation shows that this operation can be performed in real time with OpenCV on current hardware.
- While the usage of the k-means algorithm to split clusters of bees has already been used in a regression based solution for bee counting, this thesis demonstrates that it can also be used in a tracking based real-time solution for bee monitoring.
- The counting algorithm proposes a measurement to self diagnose the accuracy of the algorithm by differentiating between sure and unsure counting.
- A longtime test with a duration of 73 hours in total was performed. The results of this test show the robustness against illumination changes of the algorithm combined with the hardware setup and the practically applicability.

1.5 Methodological Approach

The methodological approach of this thesis can be broken down into the following steps

- Hardware setup and test data evaluation
- Ground truth data acquisition
- Algorithmic evaluation
- Implementation
- Evaluation
- Longtime test

Hardware Setup and Test Data

The first step is to evaluate existing hardware solutions and applicability. A recording setup already exists at the institute prior to this thesis. This setup and already recorded test data was analyzed on their applicability for this thesis. The decision to use the existing hardware solution was made, as the video sequences meet all requirements for this thesis and seemed of high quality.

Ground Truth Data Acquisition

While over thousand hours of bee recordings are available no information of ground truth data is present. Out of this reason different situations and supposedly difficult video segments were chosen to be manually annotated with ground truth data.

Algorithmic Evaluation

With test data available related publications where analyzed on their applicability. Further state of the art methods in image segmentation, tracking and counting where analyzed.

Implementation

A software solution using C++ and OpenCV was implemented that counts the bees moving in and out of the hive in test videos.

Evaluation

The resulting program was evaluated with the ground truth data on accuracy and runtime. The algorithmic evaluation, the implementation and this step where repeated in order until a solution was found that seemed optimal and met all requirements.

Longtime Test

Finally a longtime test was performed where video sequences of 7 days with recording time windows between 06:00 and 21:00 where processed. This results in 73 hours of recordings being processed. This demonstrates the practical applicability and how the resulting algorithm together with the hardware setup work in a real world scenario.

1.6 Structure of the Work

- Chapter 2 discusses the state of the art and evaluates which existing solutions can be applied or not. After a concise description of each of these publications a comparison to the requirements of this work is made. Also these approaches are grouped into different categories.
- Chapter 3 presents and explains methods that have been used or considered to be used and it is explained how they solve the problems of this thesis.
- Chapter 4 explains the concrete implementation and algorithms used in the program. This includes the algorithms used for segmentation, matching, tracking and counting. Also it is explained how bee clusters are handled.
- In Chapter 5 the used datasets are introduced first, then extensive testing on them with different resolutions is presented. Both the accuracy of the counting and the runtime is evaluated including runtime of specific algorithms in the program.
- The final Chapter (6) draws a conclusion and discusses possible future work.

CHAPTER 2

Related Work

To solve the problem of monitoring honey bees with video analysis a study about existing work and methods is required. This includes state of the art methods in video based monitoring in general and specific algorithms directly aimed at honey bee monitoring from video data.

2.1 State of the Art

While monitoring honey bees is a specific application, methods that can be applied are not restricted to bees. When the problem of monitoring bees is looked at in a more generic approach parallels to observing humans or other objects can be drawn. Out of this reason the state of the art in crowd analysis and counting of humans or other objects is also interesting to consider for this work. Further segmentation and tracking of other objects from image or video data is considered.

Counting Objects in Images and Videos

Counting people in a crowd has been done using Bayesian Poisson Regression [CV09]. This approach estimates the count of people on a walkway with two separate classes: away and towards. This method is based on statistics and does not need any tracking at all.

An approach by Spampinato et. al[SCBNF08] aims to track and count fish in underwater videos. To segment fish an adaptive Gaussian Mixture Model is used. A video is automatically annotated with the following values: environmental condition (based on brightness or smoothness of the video), the amount of fish present per frame, the total number of fish in a video and the quality of a video. Tracking fish between frames is based on shape features and histograms.

Zhang et. al[ZLWY15] propose a method for cross-scene crowd counting based on deep convolutional neural networks. The CNN uses both crowd counts and crowd density maps as learning objective. This algorithm estimates the number of people in a crowd from an input image. Further a method for single image crowd counting using multi-column CNNs[ZZC+16] was proposed by the same author. This algorithm aims to count the number of people in a crowd with arbitrary camera perspective and crowd density.

A novel approach to interactively count objects in a scene was proposed by Arteta et. al[ALNZ14]. The goal is to count a type of object in an image that is interactively defined by the user. The user annotates a reference object and the algorithm counts all similar objects in the scene. This algorithm is based on object density estimation and ridge regression.

Segmentation

Lin and Davis[LD10] propose a method to detect and segment humans with hierarchical part template matching. This algorithm uses local part-based and global shape-template-based schemes. The idea is to detect humans and estimate their pose with a hierarchical part-template tree. The algorithm is a learned human detector and uses a kernel-SVM as classifier.

Morar et. al.[MMG12] proposed new segmentation technique based on contours without edges. This approach aims to segment bones in CT images. To remove noise a Gaussian filter is used to smooth the initial image and an anisotropic diffusion filter is applied. Further steps include adaptive thresholding, island extraction and hole filling. This technique aims to counter image characteristics like low contrast between foreground and background or inhomogeneities within objects.

Superpixels are another technique to segment images. The SLIC algorithm (simple linear iterative clustering)[ASS+12] is an adaption of k-means for superpixel generation with two distinctions: by limiting the search space of a region proportional to superpixel size the amount of distance calculations for optimization is reduced. Also a weighted distance measure uses color and spatial proximity and provides control over size and compactness of the superpixels.

Yao et. al.[YBFU15] proposed a coarse to fine topological approach for superpixel generation. Compared to SLIC and other approaches a better snapping to image boundaries is achieved. In a recent state of the art survey[SHL18] the algorithm from Yao et. al.[YBFU15] ranked first for superpixel generation.

Deep learning has also been used for image segmentation. Kaiming et. al[HGDG18] proposed a method called Mask R-CNN that works as a framework for general object instance segmentation. This algorithm was published in 2018 and is an extension to faster R-CNN([RHGS15]). Mask R-CNN adds a branch for predicting segmentation masks for each Region of Interest in addition to the branches for classification and bounding box regression. Also a pixel to pixel alignment between network inputs and outputs is added.

The paper states a runtime of 5 frames per second but which hardware was used is not stated.

DeepLab[CPK+18] is another deep learning based approach for image segmentation. DeepLab aims to solve the problem of semantic image segmentation using deep convolutional neuronal networks (DCNN). This approach used convolution with upsampled filters, called atrous convolution and conditional random fields (CRF).

In the area of biomedical image segmentation a deep learning based approach called U-net[RFB15] was published. The U-net architecture uses data augmentation with elastic deformation which counters the problem of limited available training data in biomedical segmentation tasks.

2.1.1 Tracking

Choi[Cho15] proposes a near-online algorithm for multi-target tracking using aggregated local flow descriptors (ALFD). The ALFD encodes relative motion patterns between temporally distant detections using long term interest point trajectories. As runtime a framerate of 10 frames per second on a 2.5GHz CPU with 16 cores is achieved by using parallel computing.

A method aimed at multi person tracking was published by Tang et. al.[TAAS16]. This method consists of 2 major steps. First a minimum cost graph multicut problem is solved. The second step is using DeepMatching[WRHS13] to calculate pairwise costs between two bounding boxes.

A neural network based approach was published by Sadeghian et. al.[SAS17]. This approach uses recurrent neural networks and encodes long-term temporal dependencies across multiple cues to overcome difficulties like occlusion or similar appearance properties with surrounding objects.

Kieritz et. al.[KBHA16] proposed a method for multi-person tracking that uses online multiple instance learning to incrementally train an appearance model. Integral channel features are used for pedestrian detection.

An approach from Kim et. al.[KLCR15] propose a method based on multiple hypotheses tracking and introduce a method for online appearance model training for each track hypothesis. Appearance models are learned with a regularized least squares framework.

Fagot-Bouquet[FBADL16] proposed an approach that formulates the multi-frame data association step as energy minimization problem with an energy based on sparse representations. Further a structured sparsity-inducing norm is proposed. As runtime measurement 7.5fps could be reached on a 2.7GHz 8-core CPU.

2.1.2 Honey Bee Monitoring

Commercial solutions to observe bee hives already exists. One of this solutions is Arnia [1] which offers a commercial closed source hardware and software solution to monitor bee hives. This includes environmental data and also measures acoustic data. Exact information about internal algorithms is not published.

Beeandmee[2] offers a product to monitor a bee hive with environmental data like temperature, humidity and wind. Hive data can be viewed via internet and also a mobile app is offered.

The HOBOS[3] project offers live stream data of bee hive entrances, further processing is not performed. The camera is mounted vis-à-vis of a bee hive entrance and records bees flying to or from the entrance.

Melixa[4] offers a sensor that can be mounted on a bee hive to monitor the activity an environmental data. An electronic scale to monitor production is utilized. Also a patented flight sensor to analyze the flights throughout the day is used, how this sensor works exactly is not available to public. Further temperature is measured and rain can be detected. A video stream of the hive entrance is not available.

The BeeCam[5] project is an interdisciplinary education and research project from Georgia Tech with the focus on impact of urban habitats on honey bees.

The MIC-Cam[6] (Mite Invasion Control Camera) project[SZKL16][SRRK18] has the goal to monitor bee hives for varroa mite infections based on video data. The classification of mite infected bees is solved with a deep-learning based approach and compared with a traditional image processing pipeline[Sch18].

Other than commercial solutions or research projects there are also research papers that handle the monitoring of honey bees. This monitoring has been done with the following types of camera setups:

- **Laboratory** - Recording of the bees is done in a closed of laboratory area and not in or at a real bee hive.
- **Entrance 2D** - The recording of the bees happens at the entrance to the hive while the bees are on the ground and have to walk trough some kind of tunnel.
- **Entrance 3D** - The bees are recorded at the entrance while flying to or from the hive. Typically the camera is located above a landing platform. Bees can move 3-dimensional in this category.

[1] https://www.arnia.co.uk/ (Accessed: 7. 9. 2018)
[2] http://beeandmegmbh.com/home (Accessed: 18. 10. 2018)
[3] https://www.hobos.de/ (Accessed: 14. 9. 2018)
[4] http://melixa.eu/en/ (Accessed: 14. 9. 2018)
[5] https://bees.gatech.edu/home (Accessed: 18. 10. 2018)
[6] https://cvl.tuwien.ac.at/project/mic-cam/ (Accessed: 18. 10. 2018)

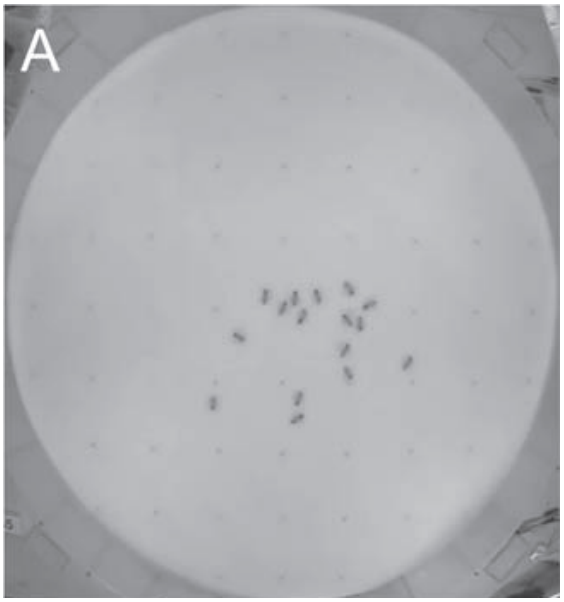


Figure 2.1: An image recorded by the laboratory based setup used by [KOC+14].

- **Inside hive** - The bees are recorded inside the hive itself. No artificial background is placed that can offer a high contrast to the bees.

- **Non Video based** - Using environmental data is also a possibility to monitor a bee hive. Using video data is not the only option.

Laboratory Based Monitoring

Kimura [KOC+14] proposed a method in 2014 to track bees on a flat laboratory area. The setup is a camera recording from the top and the bees are at a small scale. The segmentation is done with background images. An example image taken by this setup is shown in figure 2.1. The produced program has been given the name 'K-Track'. This algorithm is directly compared to the publication Ctrax[BRB+09] and gives better results on bees. Ctrax[BRB+09] was published in 2009 and is an open-source tracking program. The goal is tracking of fruit flies (Drosophila) in a laboratory area. Ctrax is also called The Caltech Multiple Walking Fly Tracker. While it is not directly aimed at tracking honey bees the problem is similar. The program's source code is available and is written in python. The tracking was performed with the camera placed at the top and viewing down on a round laboratory area. The area is also closed off which means it is not possible for flies to leave or enter the area during tracking. Ctrax uses a background model for segmentation. This background model is calculated as the mean of all frames of the input video sequence.

Figure 2.2: The honey bee pool that is used by [THKA16] for counting the bees. This is an example of a 2D setup with recording in an entrance tunnel.

2D Monitoring at the Entrance

A setup used in multiple publications is that the camera is placed above an entrance way to the hive in way that the bees are walking on the ground and the tracking problem can be reduced to a 2D. A method that uses an entrance structure has been proposed by Chen et. al. [CYJL12]. The hardware setup has entry tracks split with acrylic walls to separate bees while walking in or out and uses an infrared camera. The goal of this algorithm is to track bees that have a marker glued to their back.

A regression based method was proposed by Tu et. al. [THKA16] to measure in and out activity of bee. The hardware setup consists of an entry pool where bees have to pass trough to enter or leave the hive. A glass plate is at the bottom of this area and a mirror and a camera are located below. The background appears bright and the bees are dark in the recorded images. The images are converted to grayscale and segmentation is performed using background subtraction. To perform the counting the area is split into a fly-in and a fly-out zone and the activity of bees in each area is used to estimate the count. This method works with a raspberry pi as processing unit, but is not a continuous real-time process. Every 10 minutes a 30 second video clip is recorded with a framerate of 5 frames per second and the processing takes place during the non-recording time. An example of this recording setup is shown in figure 2.2.

Another approach[KR16] counts bees on a landing pad using computer vision. A camera records bees on a landing pad in front of the hive. This landing pad is only a few centimeters long (exact measurement is not given) and bees are segmented in 2D. The goal is to count the bees that are currently located on the pad without distinguishing between moving in and out. The hardware setup has a strict performance restriction as a solar powered raspberry pi and pi camera are used. Because of the limited processing power a permanent observation is not possible and recording is only done in time intervals.

To segment the bees from the background the HSV color model is used. To count the number of bees foreground areas larger than a bee are divided by the average area of a bee to estimate the counting. As evaluation the paper states an accuracy of 85.5% compared to manual bee counting.

3D Monitoring at the Entrance

Opposed to using an entrance structure some publications use a landing pad and the bees are recorded while flying to or from the hive. This kind of setup has to deal with problems like 3D occlusion. A stereo vision based method that tracks bees in 3d at the hive entrance was proposed by Chiron et. al. [CGKMR13] in 2013. This method uses a hybrid segmentation using disparity and intensity images from the stereo camera. Another method that uses a setup with a camera recording the bees while flying was proposed by Campbell et. al. [CMS08] in 2008. The camera is placed above the hive entrance and the bees are recorded during flying away and arriving at a landing platform in front of the hive. As segmentation technique background subtraction is used. An example of an image taken by this setup is shown in 2.3. Another method that uses a setup where bees are recorded while approaching or leaving the hive through the air has been proposed by Yang and Collins [YC15] in 2015. The camera in their setup is located outside of a bee hive above the entrance. Below the camera a white ground is placed to have a high contrast background. 3D setups have to deal with problems that occur because of the 3-dimensional freedom of the bees, including occlusions and shadows.

Monitoring inside the Hive

Another commonality shared by multiple publications is monitoring bees inside the hive. Khan et. al[KBD04] propose a Rao-Blackwellized particle filter for eigentracking that has experimental results on tracking a bee in a hive. The setup consists of a camera recording directly inside a bee hive. In this setup bees have similar colors as the hive behind them. In the experiment a single bee inside a hive is tracked. Landgraf et. at[LR07] propose an optical flow based approach of tracking honey bee movements. They specialize in tracking a bee's "waggle dance" which is a movement that scientists believe is used to communicate. Therefore this work's goal is an accurate trajectory for tracking. The data for this publication was recorded with framerates of 90 and 100 frames per second. The videos were recorded through a glass plate at the hive and artificial lighting is used. Bees in this setup are tracked with either manual selection and or with markers. Long time tracking is also a goal here. This paper states a performance of up to 1.5fps on their hardware. An example of an image recorded by this setup is shown in figure 2.4. Another publication that uses inside hive tracking has bees proposed by Kimura [KOOI11] in 2011. This method tracks multiple honeybees using vector quantization to analyze hive behavior. This method is used on images taken inside the hive and with a smaller scale compared to other publications. The videos where recorded with 720x480 resolution with 30fps. The recorded bee hive has a size of 44x19.6cm with around 700 bees being visible at the same time. This means that the size of a single bee is only a few pixels. This

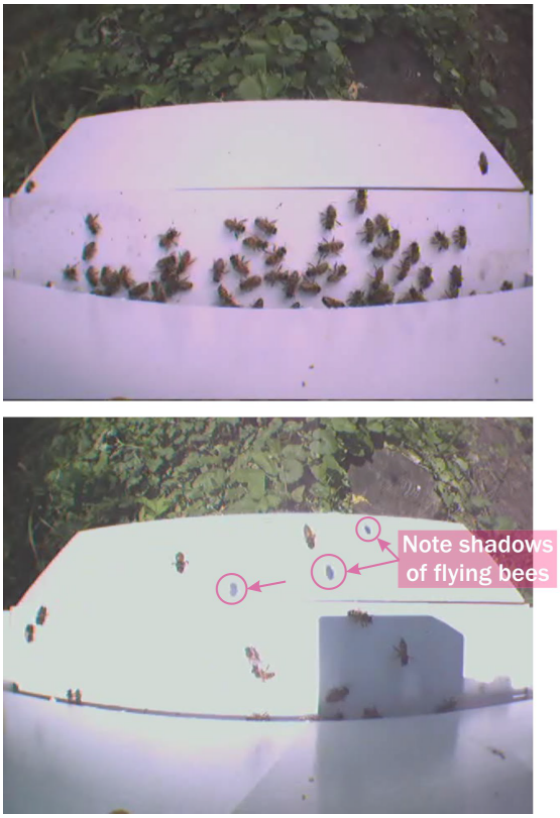


Figure 2.3: An example of a 3D tracking setup at the hive entrance. This specific setup shown in the image is used by [CMS08].

method uses the red color channel of the RGB video as this channel gives the best results. The bees are unmarked and can walk freely in the hive. The runtime of this algorithm is far from realtime with a given example of an hour for a ten second clip.

Non Video based Monitoring

Monitoring a bee hive was also done without using video data. Edwards [EMMW+16] proposed a decision tree based algorithm based on environmental data to decided the health of a bee hive. Measured data includes carbon dioxide, oxygen, pollutant gases, temperature, relative humidity and acceleration. Weather data is also taken into account and includes sunshine, rain and temperature.

2.2 Comparison

Additionally to the hardware setup types a categorization can also made regarding goals and tracking types.

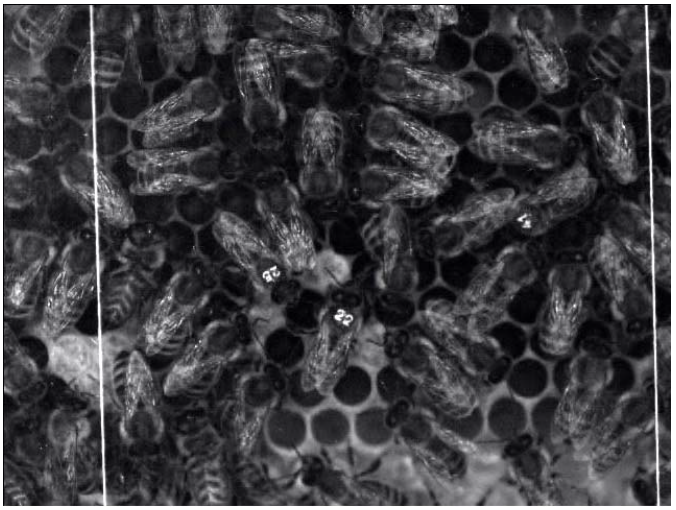


Figure 2.4: An example of a recording inside a bee hive from the setup used by [LR07]. Markers are also used in this image.

The goal can be categorized the following way and are not mutually exclusive:

- **Longtime tracking** - The tracking uses markers to identify bees across different recorded sequences. A tracked bee can be associated to a global ID and it can be detected if two tracked bees in different recordings are the same bee or not. This allows to analyze behaviour like the time how long a bee was outside the hive or how often a specific bee leaves the hive.
- **Counting** - The goal is to retrieve a counting result on how many bees have entered or left. Trajectories do not matter for this goal.
- **Trajectories** - The goal is to retrieve the trajectories of the tracked objects, for example to study their movement behavior.

Also the type of tracking can be grouped into **single** target and **multi** target tracking. Single target tracking only tracks one bee at a time, while with multi-target tracking every bee that is currently visible is tracked simultaneously. Also it is possible no tracking at all is performed as tracking is not mandatory to retrieve a counting result.

In table 2.1 a categorization of different methods is made. Opposed to the solution wanted for this thesis which is a 2D multi-target tracking at the entrance with the goal of retrieving the counting results and optionally the trajectories, none of the publications compared in the table fits exactly.

Further the adaptability of these publications related to this thesis is discussed:

3Dimensional setups record the bees while flying to and from the hive. This means the bees are able to move freely in 3D while they are being tracked, therefore the algorithm

Publication	Hardware setup	Goals	Tracking
Khan 2004[KBD04]	inside hive	trajectories	single
Landgraf 2007[LR07]	inside hive	trajectories, longtime tracking	single
Campbell 2008[CMS08]	entrance 3d	counting, (trajectories)	multi
Branson 2009[BRB+09] (Ctrax)	laboratory	trajectories	multi
Kimura 2011[KOOI11]	inside hive	trajectories	multi
Chen 2012[CYJL12]	entrance 2d	longtime tracking	multi
Chiron 2013[CGKMR13]	entrance 3d	trajectories	multi
Kimura 2014[KOC+14]	laboratory	trajectories	multi
Yang 2015[YC15]	entrance 3d	trajectories	multi
Tu 2016[THKA16]	entrance 2d	counting	-
Kulyukin 2016[KR16]	entrance 2d	counting	-

Table 2.1: Grouping of related publications into categories depending on setup, tracking type and goals

has to deal with problems like perspective, occlusion and shadows being cast far away from the bees. Also a problem is merging and splitting of tracked paths. All these problems do not occur in the setup used for this thesis, therefore it can be concluded that such algorithms cannot be directly adapted. [CMS08], [CGKMR13] and [YC15] belong to this category.

Ctrax[BRB+09] and Kimura's algorithm[KOC+14] operate with a camera farther away from the bees and with a closed off laboratory area, which means that no objects will leave or enter the tracking area in their setups, this is a common event in an actual bee hive though.

Landgraf[LR07] and Khan[KBD04] use recordings from inside a bee hive which means the background is non homogenous and has a worse contrast than when using an entrance tunnel where a specific background can be chosen. Also the tracking focuses on accurate trajectories of a single bee with computationally expensive methods opposed to simultaneous tracking of all bees in real-time.

Chen [CYJL12] uses a setup with track separation which was already ruled out for this thesis. Prior to starting this thesis such a setup has been used and high bee frequency and dead bees can cause the area to get stuffed when using restricting tracks.

The algorithm from Tu[THKA16] has a similar setup and goal as this thesis but uses a very strict computational constraint and does not offer a permanently running real time solution but only offers counting with regression based methods running in time

intervals.

After analyzing the state of the art and existing solutions the decision was made to not use a deep learning based approach but a traditional image and video processing pipeline instead. To the best of my knowledge at the time of writing this thesis there is no scientific publication that handles segmentation, tracking or counting of bees with a deep learning based approach. Other than lacking reference material there are two other reasons to not use a deep learning based approach:

- Runtime - A practical application scenario for bee monitoring is that the hardware is located directly at the hive. This means the processing power is limited. A deep learning based approach is expected to cause too high hardware costs. A performance of 5fps for a deep learning based segmentation[HGDG18] was stated without which hardware was used.

- Data - Deep learning requires a lot of annotated input data to train against. Acquiring ground truth annotated data for bee segmentation or counting is too time consuming or costly in the scope of this thesis.

CHAPTER 3

Methodology

To monitor bees with video analysis in the setup used multiple steps are required. First bees need to be separated from the background. The problem of bee clustering needs to be countered as well. When bees are detected they need to be matched to detections in past frames to reconstruct the motion of the individual bees. From the resulting trajectories the bees that enter and exit the hive can be counted.

3.1 Image and Video Segmentation

Image segmentation can be seen as the process to subdivide an image in constituent regions or objects[GW16]. In the specific context of this thesis the term segmentation refers to the problem of separating bees from the background, the bees are considered as foreground. This section provides an overview of options and reasoning on why to use them or not.

3.1.1 Background Subtraction

An approach to segment objects from background is to have an explicit background image and calculate the difference to this background to segment all foreground objects [PP12]. This is a basic way to model a background in a static scene[BJE+08].

The foreground is calculated with the following equation: I is the intensity of a pixel, B is the intensity of the background at the same image location, *d(x,y)* is a distance measurement, for example the absolute difference between the two intensity values. T is a predefined threshold value that defines how big the difference to the background has to be to be considered as foreground.

$$Foreground = \begin{cases} 1 & d(I,B) > T \\ 0 & otherwise \end{cases} \tag{3.1}$$

The downside of this method is that a background image must be taken while no bee is in the recording area. This has to be done manually. Also it only works for short time periods because over time the background will change with either dirt or small illumination changes. One goal is to have a long time solution that can observe a bee hive over multiple weeks or longer. Because of this reason background subtraction was not chosen as segmentation technique. To overcome the problems of a static background there are also methods that model a dynamic background.

3.1.2 Background Models

Background models[Pic04][CGPP03] [SG99] are an approach to background subtraction where instead of a background image a dynamically updated mathematical model is used to describe the background.

A possible way to model a dynamic background is to use the median of the last N frames as background. This has the disadvantage that a buffer with the N recent pixel values is required, also this does not include a precise statistical description and does not provide a deviation measure for adapting the subtraction threshold[Pic04].

Another method to model the background are Gaussian Mixture Models (GMM)[SG99] where the probability of observing a pixel is described with a mixture of K Gaussian distributions. The formula to calculate the probability of observing the current pixel value is the following[SG99]:

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t}) \tag{3.2}$$

$\{X_1, ..., X_t\}$ is the recent history of each pixel, $\omega_{i,t}$ is a weight estimation of the i^{th} Gaussian in the mixture at time t, $\mu_{i,t}$ is the mean value of the i^{th} Gaussian at time t and $\sigma_{i,t}$ is the covariance matrix of the i^{th} Gaussian at time t. η is a Gaussian probability density function[SG99]:

$$\eta(X_t, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} * e^{-\frac{1}{2}(X_t - \mu_t)^T \Sigma^{-1} (X_t - \mu_t)} \tag{3.3}$$

A problem with background models is ghosting ([CGPP03]), which happens when something that is considered background suddenly moves away, leaving a hole in the background model where the real background is currently not known. If bees stand still or move very little for longer periods of time they can get merged into the background creating ghosts the moment they move again. Also it is possible that there is more foreground than background visible for an hour or longer which provides a challenge to build a background model. A concrete example for ghosting can be given for this thesis. Early experiments with the test data used the program Ctrax[BRB+09] to process test videos. Ctrax uses the median of a defined amount of frames as background. In one of the videos

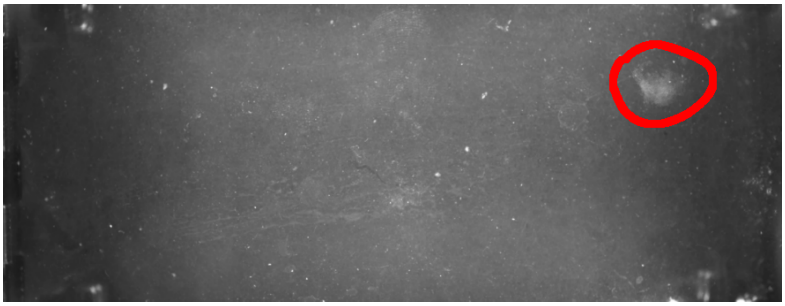


Figure 3.1: A ghost (red circle) in a background generated with Ctrax

a bee is moving slowly at the same location resulting in being longer exposed than the background at this location. Therefore this bee gets merged into the background which causes the background subtraction to be faulty at that location. While Ctrax offers a tool to manually fix ghosts this is not an option for automated processing. In figure3.1 the background with the ghost is shown.

3.1.3 Image Thresholding

Image thresholding[GW16] is a technique where a single color channel or grayscale image gets converted to a binary image by comparing the intensity of each pixel with the threshold. This technique does not require information from previous frames thus resulting in low memory usage. Also this means that image thresholding does not need a buildup time interval like background models. While background models are a generic approach that can work with many different scenes, image thresholding needs to be tailored to a specific case. This means that there needs to be a color channel where a threshold parameter can be found that is able to distinguish between foreground and background. This parameter is not adaptive and vulnerable to lighting condition changes over time. This can be described with a simple equation where I is the intensity of a pixel and T is a predefined threshold:

$$Foreground = \begin{cases} 1 & if I > T \\ 0 & otherwise \end{cases} \tag{3.4}$$

The advantages of thresholding are that it is simple to calculate and the ghosting problem does not happen. The downsides are the limited applicability as there needs to be a color channel where the background and foreground are different enough to be separable by a threshold value. This is dependent on the scene and in usual outdoor scenes thresholding is not able to segment most objects. In the specific approach for this thesis these downsides can all be dealt with, as there is a very specific setup and the background is artificially placed and can be optimized for this use case. Also the setup has internal LED lights to provide a constant lighting situation. Image thresholding allows to directly detected objects and is not depending on motion at all.

3.1.4 Comparison

Background subtraction and image thresholding are methods that require only one other image or none to perform the processing and are therefore fast to start processing with a buildup time of only one frame or they can even start on the first frame. A background model with a temporal median filter or GMM will need a longer time frame to build up a model because some bees are moving little over longer periods of time especially when they are clustered together and stuck. This also means that some parts of the background are not visible for longer time periods. There is no known exact number how long this time period is, which means this would require longtime test sequences just to evaluate the time window.

Background subtraction is dependent on a background image that must be taken manually. Considering one goal is a long time operation such a background image is vulnerable to changes in the scene like dirt accumulating over time and slight changes in illumination. Thus it is required to update the background image over time which can not be automated because it is possible the background is not clear from bees for multiple hours. Therefore a background subtraction with a static background image is not a viable solution.

With background subtraction already ruled out the choice was between background models and image thresholding. Background models have difficulties with ghosting which is a problem that occurs with the test video sequences. There are situations where a dead bee lies around for multiple minutes without any motion and then suddenly gets carried away by another bee. Such a situation will very likely cause the dead bee to be merged with the background and creating a ghost after getting removed. The difficulty with image thresholding is that a segmentation criteria must be found, specifically a parameter value that differentiates foreground from background, which is not available in general and is specific to the scene. With the artificial background it is possible to chose the background to provide a high contrast to the bees. The scene is very specific: the object that should be tracked are bees and the background is a plate with homogenous color. Thus it is possible to find a color channel where a segmentation can be performed with image thresholding. In conclusion image thresholding is the optimal choice to segment the bees from the background. The color models chosen for the segmentation were HSV and HSL.

3.2 Color Models

There are different color models a digital image can be described with. RGB is a common format for images and stands for red, green and blue and an image is represented by three channels that each contains the information of one of theses colors. These color models all use three different values to describe the color. An exception are grayscale images where the color channels get converted into a single channel containing only the grayscale information which is a reduction of information.

Since image thresholding is performed on a single channel it is important to chose the

color channel that gives the best segmentation results. Limiting this choice to the RGB color model or grayscale images neglects possible better options.

The following color models have been evaluated:

- RGB
- Grayscale
- HSV
- HSL
- XYZ
- YCrCb
- CIE L*a*b*
- CIE L*u*v*

Following the results shown in chapter 5 the decision was made to use HSV and HSL color models for segmenting the bees from the background. The HSV color model has also been used by Kulyukin et. al.[KR16] to segment bees from the background.

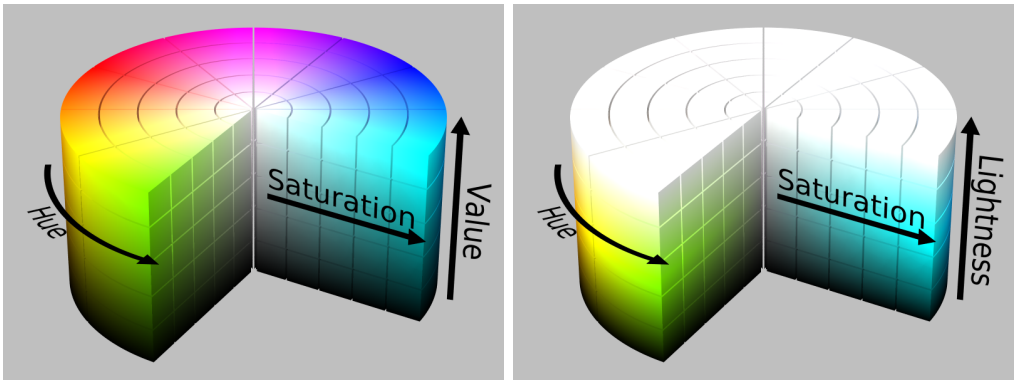
3.2.1 HSV and HSL

HSV and HSL [FP02] are two color models that share a lot of similarities. The abbreviations stand for Hue, Saturation, Value and Hue, Saturation, Lightness respectively. The Hue component represents the primary color component and is oriented at the human color spectrum, mapping the visible colors to a circle starting with red at 0°, passing the green primary color at 120° and reaching blue at 240°. From 240° to 360° the loop is closed by passing the purple color tones. Saturation can be defined as the colorfulness in proportion to the own brightness [Fai13]. A minimal saturation denotes a color without any primary color wavelength, therefore white, gray or black. A high saturation stands for high colorfulness. The lighting and value components can be seen as brightness relative to white[Fai13], with the difference that HSV maps a pure color to maximum saturation and value, while in HSL the same color is represented as maximum saturation and half lightness. A maximal lightness stands for white in HSL.

The hue is identical in both models, while the saturation has differences due different interaction with value or lightness. This color models can be visually represented as cylinders which show the mathematical interactions, as seen in figure 3.2.

Converting from RGB to HSV and HSL is done using the following equations[GW16]:

R, G and B are the channels from the RGB model and MAX and MIN are the highest and lowest value out of R, G and B.



(a) HSV (b) HSL

Figure 3.2: The HSV and HSL color models visualized as cylinder[1]
[1] Image source: https://en.wikipedia.org/wiki/HSL_and_HSV

$$Hue = \begin{cases} 0, & if MAX = MIN \\ 60° * (0 + \frac{G-B}{MAX-MIN}), & if R = MAX \\ 60° * (2 + \frac{B-R}{MAX-MIN}), & if G = MAX \\ 60° * (4 + \frac{R-G}{MAX-MIN}), & if B = MAX \end{cases} \tag{3.5}$$

The hue is an angle between 0° and 360° and is identical in both HSV and HSL.

$$S_{HSV} = \begin{cases} 0, & if MAX = 0 \\ \frac{MAX-MIN}{MAX}, & if MAX \neq 0 \end{cases} \tag{3.6}$$

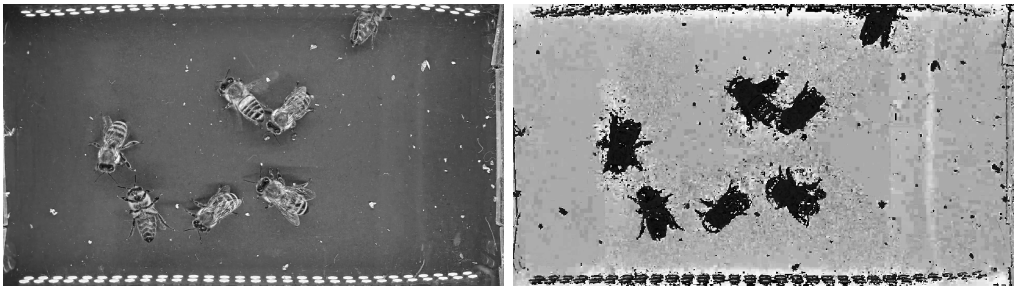
$$S_{HSL} = \begin{cases} 0, & if MAX = 0 \\ 0, & if MIN = 1 \\ \frac{MAX-MIN}{1-|MAX+MIN-1|}, & else \end{cases} \tag{3.7}$$

$$V = MAX \tag{3.8}$$

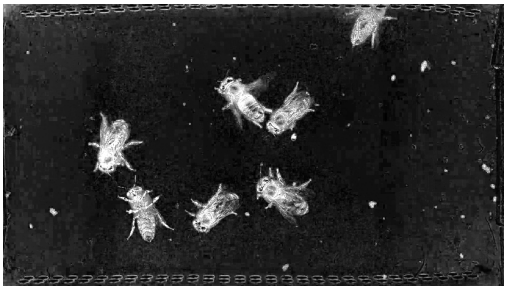
$$L = \frac{MAX + MIN}{2} \tag{3.9}$$

A specialty in these color models is that hue and saturation offer a robustness to subtle lighting changes in a scene because this will mainly affect lighting or value and leaves the hue and saturation intact. This is an important advantage to RGB where a brightness increase or decrease is reflected in all color channels.

In figure 3.3 the red channel is compared with Hue and HSV saturation to demonstrate how the bees are easier to distinguish from the background when using HSV.



(a) Red color channel (b) Hue Channel



(c) HSV Saturation Channel

Figure 3.3: Comparing the red color channel with Hue and HSV Saturation channels

3.3 Image Processing

While thresholding provides a segmentation between foreground and background the segmentation can still be enhanced by further processing. After thresholding a big problem is the remaining noise visible as single white pixels or small white blobs consisting of a few white pixels.

Also the segmented bees have holes because of the black parts of the bee body not exceeding the threshold value. These holes should be filled to closely match the shape of a bee body, which is roughly an ellipse.

By further processing the initial segmentation after thresholding the segmentation from bees from the background is improved, this is shown in figure3.4

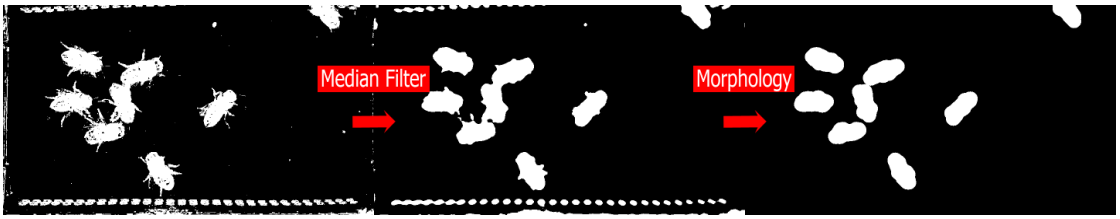


Figure 3.4: Image processing applied to initial segmentation

3.3.1 Median filter

A median filter[GW16] calculates the median value of the neighborhood of a pixel and returns this value as result for it. The neighborhood is defined as the kernel size which is usually a square of odd size. In the case of a binary image this means the median of a pixel is only white when more than half of the other pixels in the neighborhood are also white. This removes all single white pixels and small white blobs that are not connected to one of the bigger blobs. A lot of dirt in the background image that is still present after thresholding can be removed by median filtering if the kernel is large enough.

3.3.2 Morphology

Mathematical morphology[Ser83] can be used to retrieve shape characteristics out of image data. Opening is a morphological erosion followed by dilation.

Erosion($\ominus$) can be defined as[Ser83]:

$$A \ominus B = \{x \in E^N | x + b \in A \text{ for every } b \in B \tag{3.10}$$

Dilation($\oplus$) can be defined the following[Ser83]:

$$A \oplus B = \{c \in E^N | c = a + b \text{ for some } a \in A \text{ and } b \in B \tag{3.11}$$

A and B are sets in N-space (E^N) and subsets of E^N, a and b are N-tuples of element coordinates.

In the case of image segmentation the image is A and B is a structure element. In this thesis the image is a binary image of the initial bee segmentation. The structure element is a disc to retrieve the elliptical shape of bee bodies.

After filtering the image can still contain some erroneously segmented blobs caused by dirt and the segmented bee bodies might still be fractured into a few parts or contain holes. This situation can be improved by applying morphological opening to the image. The erosion step removes most blobs that were caused by dirt and only leaves the centers of the bee bodies. The dilation step fills remaining white pixels which are the bee bodies to elliptical shapes resulting in blobs that accurately resemble the bee bodies. The size of the structure element must be appropriately chosen. If it is too big the bee bodies get removed. On the other hand a too small structure element does not remove big noise blobs and might leave holes in the bee bodies. Also the bee shape is not elliptical in that case but consists of multiple smaller circles. The correct size of the structure element depends on the size of a bee in the processed frame. In figure 3.5 the erosion and dilation steps are visualized on a segmented bee.

3.4 Tracking

In the task of simultaneously counting all honey bees that enter or leave an image there needs to be a way to retrieve their motion. Tracking the bees while they are visible allows

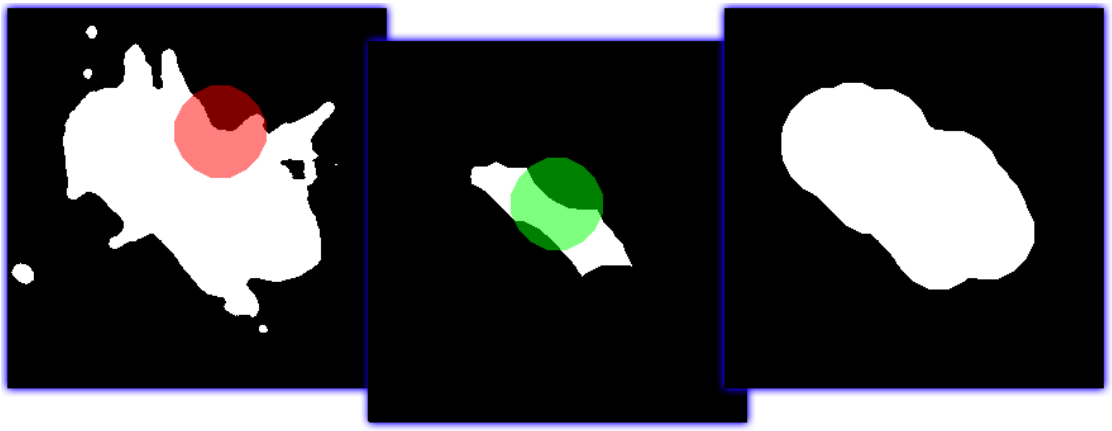


Figure 3.5: A segmented bee (left), the result after erosion(middle) and the result after dilation(right). The green and red discs are the structuring element that has been used.

to reconstruct their trajectories and in conclusion count if they have entered or left the hive.

Tracking is the process of estimating the trajectory of an object on the image plane while it moves around the scene[YJS06]. This means the known location of an object in a frame is updated to the position of the same object in a future frame. Tracking is used in many different computer vision tasks[CRM03][HHD00].

The main difference in tracking algorithms is the way in which an object is represented. This includes which image features can be used and how the shape or appearance of an object can be modeled[YJS06].

3.4.1 Detection vs Tracking

Tracking methods need an objected detection to either initially detect an object and then follow it or a detection in every frame with a correspondence calculation between detections in different frames[YJS06].

Detection and tracking can be differentiated into two distinct processes. Detection can be defined as process of obtaining the location of an object in the scene without further information about previous or future locations being available. This is required for this thesis since no information about location or the amount of bees is known. Tracking can be seen as the specific process of updating the location of a detected bee to the position of the same bee in the next frame. This means tracking refers to algorithms that update an already known position of an object to a newer position in the future without having to newly detect the object. The point of this kind of algorithms is that detecting is often computationally more complex or has to be done manually. So only in a few frames objects are detected while in frames in between tracking is used to follow the motion of the object over time.

For tracking bees at the hive entrance detection has to happen very frequently as new objects may enter or exit the area every frame. This means it will be required to run a detection every frame. This concludes that detection has to be focused on and tracking can only work when a detection is already done. Also it is required to simultaneously track all bees currently visible in the video.

3.4.2 Position Matching

With a frequent detection of every frame a way to track the bees is to match the detected positions of two adjacent video frames to pairs. While this seems like a trivial task there are special cases that have to be considered. In an easy example both frames detected three bees and every position gets matched with the position closest in Euclidean distance in the other frame. This principle is visualized in figure3.6

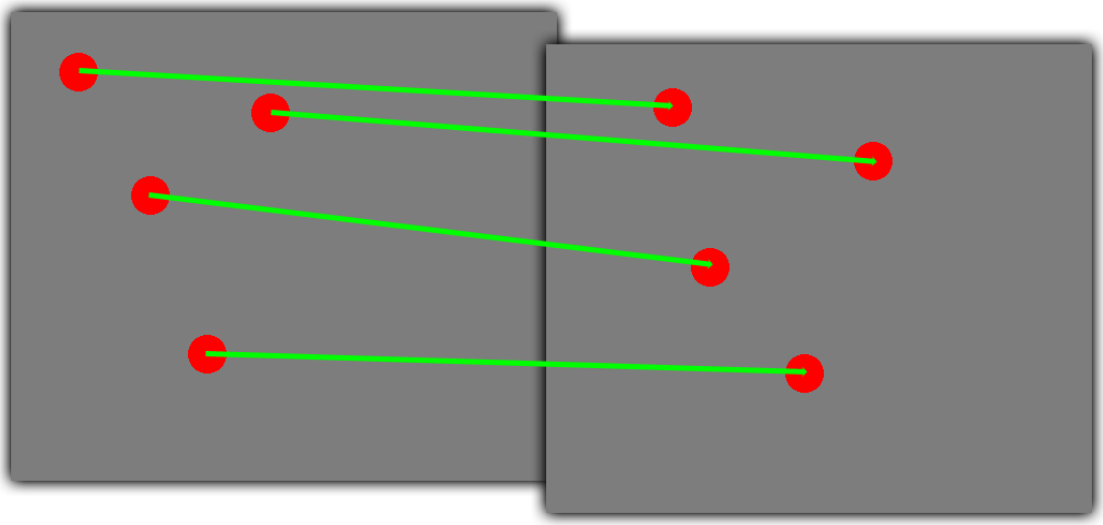


Figure 3.6: The bee detections from the previous frame get assigned to the nearest detection in the next frame.

Ideally matching the nearest neighbor is not ambiguous and the matching is finished, but this is not always the case since it is possible that two bees in one frame are closest to the same bee in the other frame. This also assumes that both frames have detected the same number of bees and these are the same bees. It is a common occurrence that bees enter or leave the area so a bee that was detected in the previous frame can be absent or a new bee can appears in the detection. To solve this problem a maximum tracking distance is defined. This is dependent on the frame rate and the size of the bees and should be the maximum distance a bee will move between two frames. In the test videos which are all between 20 and 30 frames per seconds a value of 2/3 of the size of a bee showed to be a value that not causes loss of tracks for the maximum tracking distance. This allows to filter out most new detections or lost bees without losing track when a bee moves too fast. So every bee that has no other bee closer than this maximum

tracking distance will not be matched and is considered as new or lost. Bees in the older frame that cannot be paired are considered as lost, which means either they could not be correctly segmented or they left the visible area. Unmatchable Bees in the newer frame are considered as new bees that entered the area.

An example of an ambiguous situation is visualized in figure 3.7. The yellow arrows show the motion from frame N (red) to frame N+1 (green), but when trying to assign the nearest neighbor both green spheres in the upper left corner are nearest to the upper left red sphere.

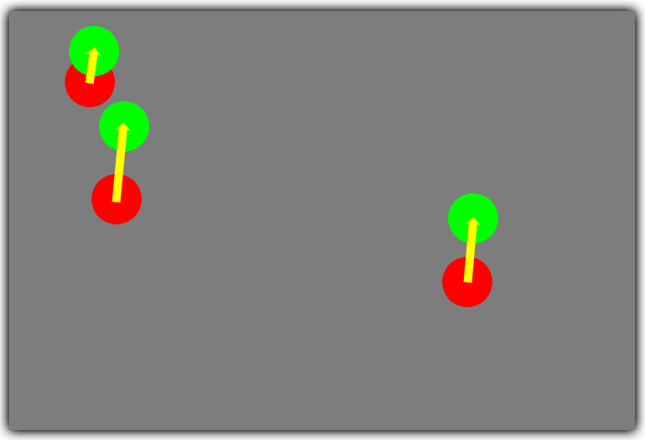


Figure 3.7: Detections in frame N (red) and frame N+1(green) with the create an ambiguous situation when trying to assign the nearest neighbor.

Since not always a non ambiguous matching is produced a more sophisticated approach is needed. The state of the art approach to this problem is the munkres algorithm.

3.4.3 Munkres Algorithm

The Munkres algorithm [Mun57][Kuh55], also called Hungarian algorithm, is an algorithm that produces an optimal matching for a bipartite graph. The problem of matching the detected objects in two frames can be seen as a bipartite graph matching problem. Every object should get matched to an object in the other frame and each matching has a weight. As weight between two objects in different frames the Euclidean distance is used.

While this algorithm produces an optimal matching on a graph, there are still some flaws when used to match the bees of two adjacent frames, as an optimally matched graph alone is not the needed solution. The algorithm always tries to match nodes thus even two points far apart will get matched if there is no other partner because the algorithm treats not matched nodes as worse. In this project every frame new bees may appear or leave therefore even when two adjacent video frames contain the same number of bees the wanted solution is only to match the points which are close to each other. To overcome this weakness of the algorithm every bee that has no possible matching partner within

the defined maximum tracking distance is filtered out prior to finding the matching. This bees are considered as new or lost bees respectively. This does not fully solve the problem but avoids many problematic cases. Even though when all bees have a potential partner within the maximum distance it still is possible for the algorithm to produce a result where two bees get matched that are farther apart than the maximum distance. Such a matching will be broken up and both bees are considered to have no partner. With these tricks the weaknesses of the algorithm for this application can be countered.

An example of a problematic situation are shown in figure3.8. The detections from frame N (red) got assigned to the new detections from frame N+1 (green). While in the left upper corner a case that would be ambiguous with nearest neighbor matching gets resolved by the munkres algorithm on the right side two detections got assigned that are far apart. Such a situation happens when a bee leaves the visible area in frame N and a new bee enters somewhere else in frame N+1. If such detections get filtered out this assignment does not happen.

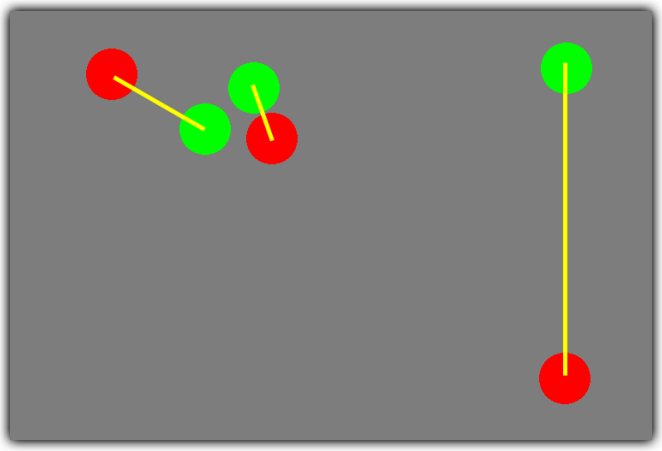


Figure 3.8: Detections in frame N (red) and frame N+1(green) get assigned with munkres algorithm. Detections far apart also get matched

3.4.4 OpenCV Tracking

Another idea to improve the matching is to not just use the position of the last frame but to predict the new position in the next frame and match them with the new detections. A possible way to predict the new position is by using a tracking algorithm provided by OpenCV. Out of the previous and current frame a tracking algorithm can track the position of a region to where it moved in the next frame. This is performed by using the bounding box of a detected bee as region to be tracked. The tracking algorithm then updates this bounding box to a new position where the same object is supposed to be located in the next frame. In theory this allows a more accurate matching as the new detected position and the predicted new position should be very close.

The practical application did not provide any improvements though. Two tracking algorithms available in OpenCV (Median Flow[KMM10] and Kernelized Correlation Filter[HCMB12]) have been tested and but there was no definite improvement in the counting results, only minor changes occurred that were not always improvements. The runtime of the whole program got a lot worse by using OpenCV trackers and increased by over 200%. This lead to the conclusion that trying to improve the program by using sophisticated tracking algorithms is pointless as the improvements in accuracy are minor or non existent and the runtime suffers severely.

3.5 Clusters

When using thresholding and applying filtering it is possible to reliably segment bees when they are not located in close proximity. This means as long as no segmented bee blob in the binary image touches another blob each single bee can be detected. But when a lot of bees are visible at the same time, the area gets crowded and bees begin to touch each other and form clusters. The solution to this problem is to apply the k-means algorithm. The idea of using the k-means algorithm to split bee clusters was from Tu et. al 2016[THKA16].

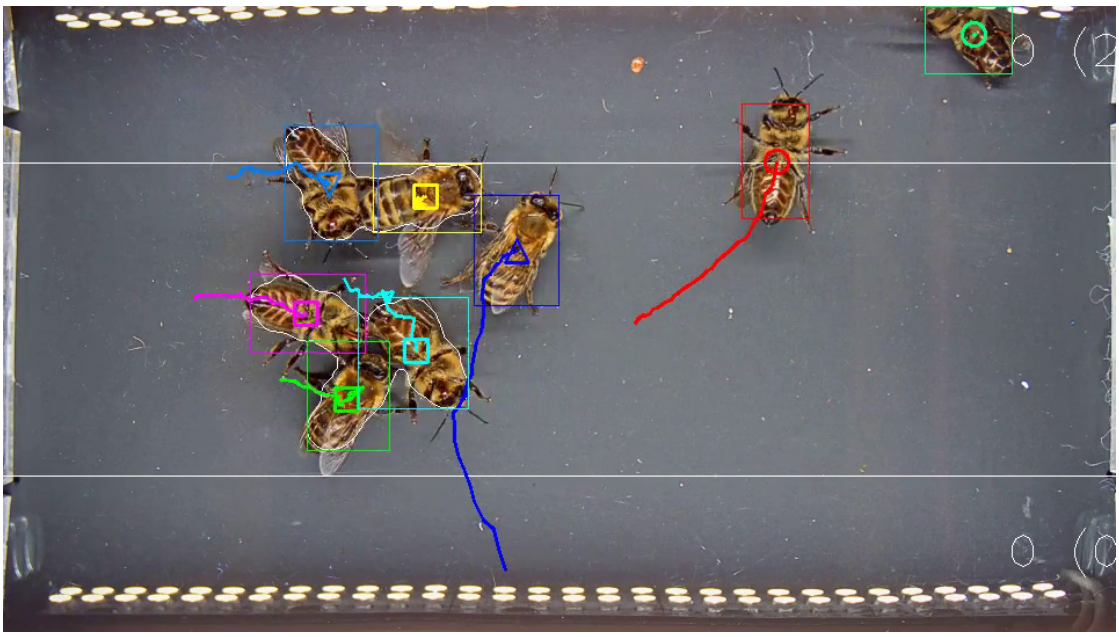


Figure 3.9: Example of a frame were clustering needed to be countered

Figure 3.9 shows an example of a frame that needed clustering to detect the bee centers. The white contours show areas that were one region after segmentation but with an area that exceeds the threshold of belonging to a single bee. After applying the k-means

algorithm the different bee centers could be detected and the bounding boxes of each bee are estimated from the results of the k-means clustering.

3.5.1 K-Means

The k-means algorithm [Llo82][AV07] is a cluster algorithm that splits an area into k cluster. K is a parameter which stands for the number of clusters and has to be known previously. This is the biggest weakness of the k-means algorithm.

The k-means algorithm is a simple but effective solution to a clustering problem when the amount of clusters is known. The algorithm works the following:

1. Choose k initial cluster centers
2. Assign each point to the closest cluster center
3. For each cluster calculate a new cluster center that is the center of mass for all points in this cluster.
4. Repeat steps 2 and 3 until there is no more change, or the change is below a defined threshold.

The segmented bees are roughly the shape of an ellipse and the setup restricts the bees from overlapping. This means that the number of bees in a cluster can be estimated by dividing the cluster area through the average area of a bee. The average area of a bee must be defined and the shape of a bee is expected to be an ellipse with 2:1 ratio of the primal axes. This allows to calculate the number of bees in a cluster. A precise segmentation that matches the bee body contours results in the correct number of bees. K-Means optimizes the individual cluster centers returning center points for the bees and also each pixel is assigned to one of the k clusters allowing to estimate the bee positions by fitting an ellipse around the pixels.

3.6 Ground Truth data

To evaluate the accuracy of the used algorithms it is mandatory to have ground truth data for test video sequences as the accuracy of the algorithm can not be quantified otherwise. Since this data is not available prior to this work, it must be created. To label such a video a tool that assists the user in the process is absolutely necessary. Such a program was found in Vatic.

3.6.1 Vatic

Vatic [VPR13] stands for Video Annotation Tool from Irvine, California and was released as a program that allows to publish the annotation tasks to online platforms where they can be performed by other people compensated with money. While this will become too

expensive and the quality of outsourced annotations is expected to be lacking, Vatic still can be used as offline tool to annotate videos and is quite helpful in this process. The user has to place bounding boxes around the objects that should be tracked and then update them after a few frames and the tool handles the interpolation.

Annotating videos with Vatic is still time consuming, a short video sequence of around 13 seconds with 143 different bee tracks (which can be considered as an extreme case) required over 8 hours of time to annotate. Because of the high effort required the amount of videos that were annotated that way are limited. In total 197 seconds of video data have been manually annotated with vatic. The advantage is that these annotations are of high quality and expected to have close to zero errors. An example of a very difficult annotation task is shown in figure 3.10.

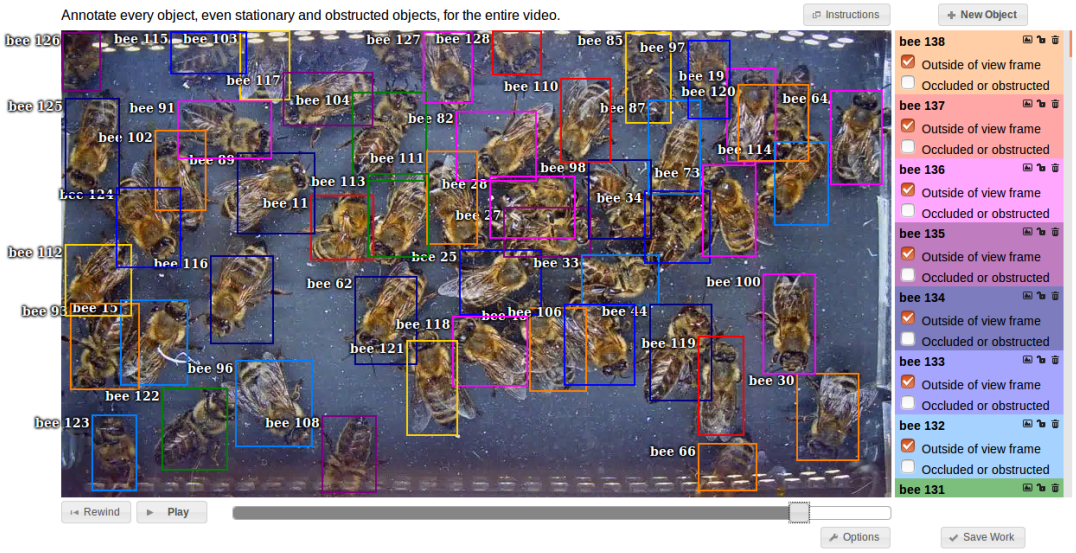


Figure 3.10: Vatic

3.6.2 Manual counting

Because of the high effort of annotating videos and since evaluating only the count does not require ground truth of the trajectories a few 1-2 minute videos where counted manually to provide a rough estimation to compare to. Counting the bees that way is a very difficult task for humans thus this data is expected to be less accurate than the labeled videos. An error rate is estimated of up to 10%. Because of the lower accuracy it is still important to have the fully annotated videos available whose error is estimated to be less than 1%.

3.6.3 Possible Error Cases

While all the algorithms presented it also is necessary to discuss what kind of errors can happen during the tracking and counting and how to deal with them. This includes methodological errors for the tracking, matching or counting.

Incorrect matching

It is possible that the matching stage, even when using the munkres algorithm, does not assign the bees correctly. A case where this can happens is when two bees pass each other very closely and the two bees touch in the binary segmented image on their sides. When this happens it is possible the merged blob appears similar to a single elliptic shape instead of two touching ellipses. When this happens k-means might separate these two bees incorrectly which causes the detected positions to be wrong which in conclusion increases the chance the matching will be incorrect. As long as the framerate is high enough to not cause mismatches by fast moving bees the assignment will be correct. But errors occur because of incorrectly detected bee centers. This is dependent on the image quality of the recorded data. The cases where two bees touch each other and then cause the tracker to swap the bees is rare. From manual observations of the test data the influence of this problem is estimated to be smaller than 1% on the overall counting accuracy.

Losing Track

Another possible problem that can occur is that a bee gets lost while being in the transition zone for a few frames because of just barely not matching the criteria to be segmented. This also depends a lot on the image quality and parameters. If a bee gets lost this means that it will be a new detection shortly afterward which causes that it is no longer known where the bee entered the inner zone. This decreases the accuracy of the counting algorithm.

To counter this problem a newly detected bee in the inner zone of the image is checked with positions of recently lost bees. If a lost bee exists in close proximity the probability is high that it is the same bee. In such a case the newly detected bee is considered to be the same bee that recently got lost instead of a new detection. Since bees can only enter or exit the tracking area at the top or bottom a bee that gets newly detected in the inner zone has to either been lost or not detected at all. Bees not getting detected does occur very rarely, in the test sequences it does not happen at all when the parameters are suitable. So it is safe to assume that when a newly detected bee is close to a recently lost bee that it is the same bee. The time window for how many frames a bee will be recovered is specified by a parameter and a value between 3 and 5 seems reasonable as most of the time bees only get lost for 1 or 2 frames during the test sequences.

Recovering track losses over a few frames does improve the counting accuracy as bees that can be recovered do not lose the information on which side they entered and the counting can be sure when the bee exited or entered.

Bee walking back in

When a bee walks in the transition zone and walks out on the other side it will be counted. If the bee then decides to not walk out of the camera view but back into the transition zone and exits it on the same side it entered it at first it will be counted as both a bee walking in and a bee walking out.

When the track is not lost this error could be corrected by reducing the count when an already counted be walks back in, but it only does make a difference if the bee passes both lines two times. If a bee that got counted walks back across one line but does not pass the second line the count is correct. Manually observing the test data for this kind of error lead to the conclusion that it is not worth the extra effort as the occurrence of this problem is rare and happens in less than 1% of all counted bees.

Another reason not to resolve such cases is that it could interfere with the next problem, the wrong U-turns, decreasing counting accuracy of the algorithm instead of improving it.

Wrong U-turn

When a bee leaves the tracking area and a new bee gets detected next to it in the next frame the matching algorithm might assign this to the same bees resulting in a bee track walking a U-turn and moving back in. While in reality this are two different bees they will incorrectly get merged together.

The counting algorithm does not get influenced by this error since a bee at the top or bottom border has already been counted and there is no difference to a new bee walking in or an already counted bee walking in again.

The best solution to this error and also the previous error with bees walking back in is to not handle them as the influence on the counting accuracy is insignificant.

With the different options evaluated the decision for the actual implementation was made. The tracking is done by matching the center points with the help of the munkres algorithm. The center point of each bee is detected by image thresholding in the HSV or HSL color model and further processing by applying a median filter and morphological opening. Clustering is countered by using the k-means algorithm.

CHAPTER 4

Implementation

The software solution was implemented in C++[1] 11 and with OpenCV[2] version 3.2. For testing purposes a GUI was created with the QT Framework[3], but the processing part is not dependent on QT.

4.1 Segmentation

Segmentation was implemented by segmenting each frame of a video with binary thresholding and further enhance the image with median filtering and morphological opening. For each step in the segmentation stage of the algorithm the result is shown on the frame in figure 4.1 as example.

4.1.1 Thresholding

Thresholding transforms a single input frame into a binary image with black as background and white what is supposed to be a bee. But before this thresholding can be performed the first task is to convert the RGB color image to a different color model where the segmentation should be performed. The evaluation 5 showed that HSV and HSL provide the best results. A dataset where the glass plane already accumulated a lot of dirt, resulting in less saturated images gave better results with the hue than the saturation.

After the image is converted into the chosen color model and the color channel is extracted the initial step of the segmentation is performed by converting it into a binary image by applying a threshold value. This threshold value is a parameter that must be defined depending on the setup or dataset. The result of this stage is a binary image that roughly

[1] http://www.cplusplus.com/ (Accessed: 24. 09. 2018)
[2] https://opencv.org/ (Accessed: 24. 09. 2018)
[3] https://www.qt.io/ (Accessed: 24. 09. 2018)

Figure 4.1: Frame that will be segmented

segments the bees from the background but still contains noise and holes. Also smaller white pixel blobs that are not bees might appear in the image. Those are mostly caused by dirt in the scene. LED light reflections at the top and bottom of the image also get segmented, but those areas are unimportant for the tracking and counting as at least a strip of the length of a bee has to be ignored at both the top and bottom to provide enough area for the bees to fully enter or leave the image. The left and right image borders are aligned with the sides of the area where the bees can move and therefore do not allow bees to be partially occluded.

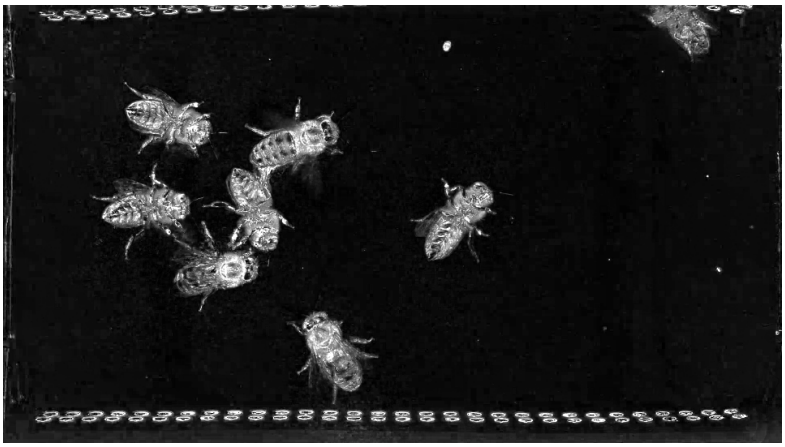


Figure 4.2: The saturation channel (HSV) of the frame that will be segmented

In the example image (4.2) the bees are well distinguishable from the background in the saturation channel for a human. This suggests a segmentation will be accurate which is shown in figure (4.3).

After thresholding like seen in figure 4.3 the bees got segmented from the background

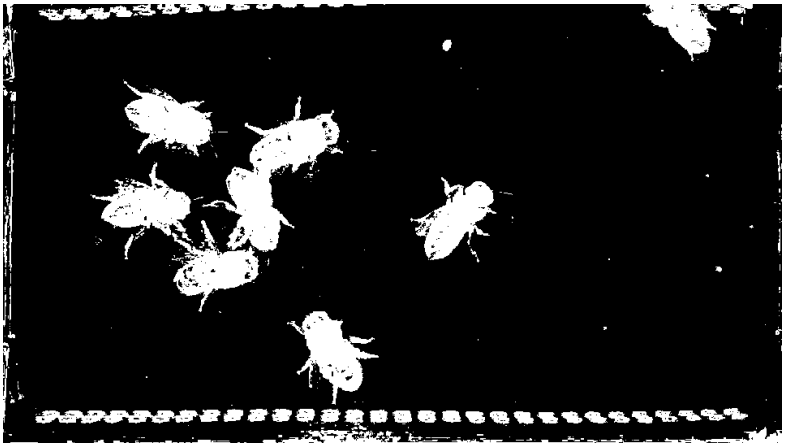


Figure 4.3: The binary image of the example frame after thresholding

but some minor noise and small blobs from the LED lighting are still present. Also there are small holes present in the bee bodies in the binary image. These artifacts will be removed in the further steps.

4.1.2 Image Filtering

The result of the first segmentation step contains holes and the bee blobs are fractured and therefore the segmentation of the bees contains inaccuracies. The first filtering step which is the application of a median filter tries to counter the noise but also fills small holes. The median filter calculates the median of each pixel's neighborhood and if this value is 1 the resulting pixel is white, otherwise it will be black. Small noise of single white pixels or blobs with a few connected white pixels get completely eliminated by the the median filtering. The size of the blobs that can be removed is dependent on the filter kernel size. This filter kernel size is a parameter that must be defined. The choice for this parameter is both dependent on the dataset but also on the size of the bees in the image. The goal for the kernel size is that noise and the legs of the bees get removed but the main body parts are kept intact. The size of the median filter kernel has to be lower than the size of the structure element for the morphology step. The median filtering has a significant influence on the runtime (shown in chapter 5) depending on the filter kernel size.

The application of the median filter (see figure 4.4) removes noise from the image but also removes the legs from the bees which is an improvement since the tracking focuses on the bee bodies. The remaining artifacts in the image are blobs caused by the LED lights and sometimes also small holes in the bee bodies. In the example image (4.4) there are no holes in the bee bodies after the median filtering but it might occur in datasets with lower image quality.

After the median filter has been applied the image should contain close to zero noise and

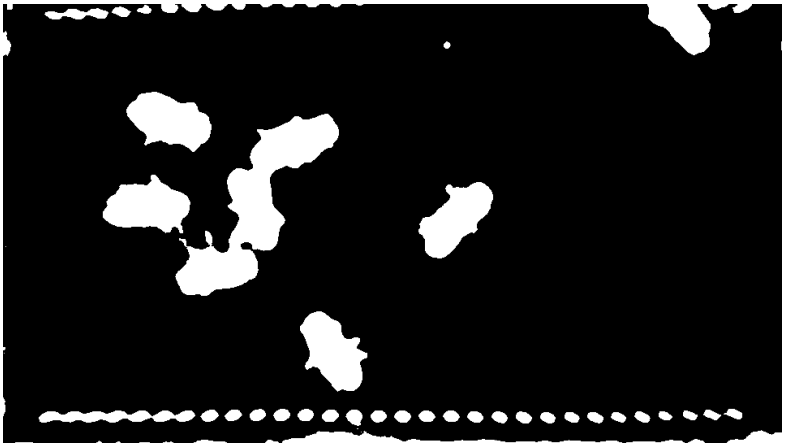


Figure 4.4: The segmented image after median filtering

only fractured parts of the bee bodies should be visible. To finish the segmentation it is required to fill all holes between close blobs and merge them together to resemble the roughly elliptical shape of a bee body. The morphological opening operation achieves this. Morphological opening consists of an erosion step followed by dilation. The erosion step removes the outer parts of the remaining blobs only leaving the inner central parts. Remaining noise gets eliminated by this process. The parts that are still present after erosion are expected to be bee body parts. As structuring element for this task a disc is used as the bees are elliptically shaped and this will result in round shapes after dilation. The size of the disc is mostly dependent on the size of a bee and evaluation showed that around 60% to 70% of the width is a value range that allows to get the described segmentation consisting only of the bee bodies. When the remaining blobs do not resemble an ellipse but clumps of circles are visible the structuring element is too small. If bees get completely removed it is too large.

In the dilation step the remaining parts get filled with the disc shape resulting in roughly bee body sized ellipses. This completes the morphological opening operation. After this step the remaining white blobs are the bodies of the bees in the scene.

As shown in figure 4.5 the morphological opening removed the remaining artifacts that persist after median filtering and only the bee bodies are left. All small blobs got eliminated and only the bigger blobs remain which are the bee bodies.

To compare all the steps the intermediate images have been put together to one image in figure 4.6. The threshold image is the red channel, the median filter result is the green channel and the result of the morphological opening is the blue channel. Every color containing blue is part of the final result which includes white, cyan and purple.

Figure 4.5: The segmented image after morphological opening

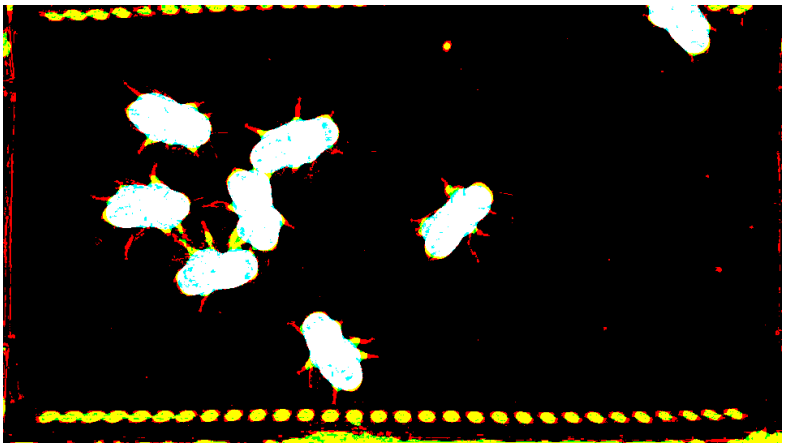


Figure 4.6: The debug image with the three segmentation steps put together as color channels. Thresholding (red), Median filter (green) and Morphological Opening (blue)

4.1.3 Bee Extraction

After the segmentation step the locations of the bees need to be extracted from the binary image and transferred into data structures for the counting and tracking algorithm to work on. The tracking and counting part does not need image data anymore and only requires the center points of each bee in the scene in the 2-dimensional Euclidean plane. The pixel positions are used as coordinates.

The first step to retrieve the locations is to find all the blobs in the image. This is done by using the `findContours` method from OpenCV. From the contours the center points of the bees can be calculated but the special case of multiple bees forming a cluster must be considered. Each contour with an area below a certain threshold will be considered as single bee detection. The threshold is calculated as 1.25 times the average bee area,

which is calculated as the ellipse area from two parameters of this program: the bee width and length in pixels. The value of 1.25 is used to leave room for bees being slightly larger than average. Each contour with an area lower than a third of the average area will be filtered out as incorrect detection.

All the single bee detections will get added to the list of detected bees in this specific frame and will passed to the matching stage.

Bee Clusters

Unfortunately not all bees are easily extractable and some bee blobs may touch and form bigger clusters. These clusters need special treatment to extract the single bee positions. The algorithm used to split these clusters is k-means. But to use k-means the k-parameter must be found first. This parameter is calculated by dividing the area of the blob through the average bee area and will be rounded up. After k is calculated the k-means algorithm is run to retrieve the center positions of the bees in the cluster. These resulting positions are then treated equally to the single detections.

4.2 Bee Matching

After the detection of the new bee locations they are passed to the matching stage. These positions have to be matched with the ones from the previous frame to assign them to the correct bees. First all bees that have no possible matching within a maximum tracking range parameter are filtered out as they will end as either new bee or lost bee respectively. All other bees now get assigned to the bee with the closest distance in the other frame. If this assignment is unique the matching is finished.

4.2.1 Munkres Algorithm

In some cases the simple logic of assigning bees to the closest bee in the other frame produces ambiguous results. To solve such situations the munkres algorithm is used. This algorithm produces an optimal bipartite graph matching for the bee assignment. Since the algorithm always wants to match no matter how high the weight is, all assignments that have a larger weight than the maximum distance are discarded. These bees are considered as not having a matching partner in the other frame. After this step the matching stage is always finished.

4.2.2 Lost bees

When a bee that was present in a previous frame can not be matched to a new detection it is considered as lost bee. This means it either left the visible area or the segmentation failed to capture it. If a bee gets lost at the upper or lower part of the image which is outside of the counting area it does not matter and it likely just left but when a bee gets lost in the middle of the image there has to be a segmentation error. These bees are kept

in a separate lost bees list for a predefined low number of frames. After the maximum amount of lost frames is reached it will be discarded. When a new bee is detected by the algorithm all lost bees are checked if they are close to the position of the new detection and if this is the case the closest lost bee gets assigned to the new detection and the bee is put back into the active list. The positions from the missing frames are interpolated. The radius for the assignment to a lost bee is the same as the maximum tracking distance parameter used in the matching stage. The goal of the lost bee recovery is to deal with short detection losses of around 1 to 3 frames which sometimes occur. If a bee that gets lost for a short time is not recovered the track is lost and the counting algorithm is no longer able to determine from which side the bee entered.

4.3 Counting

The first and simplest idea to counting is by putting an imaginary horizontal line in the middle of the image and check every frame if a bee passed this line. The line is passed when the last two positions lie on different sides of the line. While this works in theory it is prone to errors as the detection is not perfectly accurate and the bee center might move slightly between frames even if the bee is standing still. Also bees are not motionless and still move their body slightly even if they hold their position. This means that a bee standing at the line causes a lot of erroneous counts. Also another problem that can not be handled that way is when a bee walks in and passes the line then turns around and walks back out. Such a case should not cause any counting but with one line counting the result will be 1 in and 1 out.

To overcome those difficulties the solution is to put two horizontal lines in the image that split the recorded area into three different zones:

- Upper zone
- Transition zone
- Lower zone

The two lines, the upper line and the lower line, separate these zones and a bee should only be counted when both lines are passed in succession. To achieve this it is required to store a state for each bee where it entered the transition zone. The following states are used by counting algorithm:

- Outside
- Entered upper
- Entered lower
- Entered unknown

Every bee that is not in the transition zone and therefore located in either the upper or lower zone is labeled with the state **Outside**. When a bee passes the upper line from the upper zone to the transition zone it will get the state **Entered upper** and when a bee passes the lower line from the lower zone to the transition zone it gets the state **Entered lower**.

When a bee with the state **Entered lower** or **Entered upper** passes the same line as it has entered the state is set to **Outside** without increasing any counters. This happens when a bee runs a loop and exits the same side it entered.

Every bee that is located in the transition zone in the first frame or that gets detected as new bee while in the transition zone will be labeled with the state **Entered unknown**. The program has no knowledge about where these bees have entered the zone.

When a bee passes the lower line and has the state **Entered upper** the counter at the lower line is incremented and the bee's state gets updated to **Outside**. When the bee passes the upper line and the state is **Entered lower** the upper line counter is increased and the state is updated.

The only case that is left is when a bee with the state **Entered unknown** passes one of the counting lines. Since it is not possible to know where these bees entered they will be counted but a different counter is used for them on both lines. This means the counting will be separated into three different types per line:

- Sure count
- Unsure count
- Total count (Sure count + Unsure count)

Bees where it is unknown on which side they entered will be added to the **unsure count**. Other counting events will increment the **sure count**. The reason to split the counting is because the **unsure count** value is more likely to contain erroneous counts. This is because these **unsure counts** happen when the bee is lost in the transition zone which happens when incorrect number of bee centers get detected by the segmentation stage. Having two counting values allows to use that information to self diagnose the quality of the counting algorithm which is explained in section4.3.2.

To conclude this the program has to decide between the following counting events:

- `COUNT_NOTHING`
  causes no counting
- `COUNT_DOWN`
  increment the **sure count** of the lower line

- `COUNT_UP`
  increment the **sure count** of the upper line

- `COUNT_DOWN_UNSURE`
  increment the **unsure count** of the lower line

- `COUNT_UP_UNSURE`
  increment the **unsure count** of the upper line

The whole bee state and counting can be described with pseudocode 4.1.

Algorithm 4.1: Updating bee states and counting

```
1  if Bee passes upper line from inside to outside then
2  |  if bee state is Entered_Down then
3  |  |  set bee state to Outside
4  |  |  return COUNT_UP
5  |  else if bee state is Entered_Unknown then
6  |  |  set bee state to Outside
7  |  |  return COUNT_UP_UNSURE
8  |  else
9  |  |  set bee state to Outside
10 |  |  return COUNT_NOTHING
11 else if Bee passes upper line from outside to inside then
12 |  set bee state to Entered_upper
13 |  return COUNT_NOTHING
14 else if Bee passes lower line from inside to outside then
15 |  if bee state is EnteredUp then
16 |  |  set bee state to Outside
17 |  |  return COUNT_DOWN
18 |  else if bee state is Entered_Unknown then
19 |  |  set bee state to Outside
20 |  |  return COUNT_DOWN_UNSURE
21 |  else
22 |  |  set bee state to Outside
23 |  |  return COUNT_NOTHING
24 else if Bee passes lower line from outside to inside then
25 |  set bee state to Entered_lower
26 |  return COUNT_NOTHING
27 else
28 |  return COUNT_NOTHING
```

4.3.1 Multiple Counters

Another weakness that can be observed by using two lines is that they are arbitrary placed into the scene. There is need for enough space to the borders so the bees are fully visible and reliably detected and the transition zone should be as large as possible to avoid counting of bees that only run a loop. But then there is no reason why to put the line at which exact location. A lot of errors in counting are dependent on the location of the counting line. For example a bee close at the line gets incorrectly assigned to a position on the other side of the line or when a bee runs a loop and returns the size of the transition zone determines if it gets counted or not. When the zone is small it is more likely a bee runs a loop while passing both lines. Two overcome this problem a solution is to use multiple counters with different lines at the same time. This is possible without noticeable performance impact because all of the performance heavy segmentation stage is shared between all counters. The biggest difference is that for each bee now an array of states with one value for each counter is stored instead of only a single state. The state is not memory intensive since there are only 4 different values.

But how many counters should be used? There are still the two criteria points that are needed to be fulfilled for the counting line placement. First there must be enough space to the upper or lower border so the bees are fully visible, also in the test videos it can be observed that in some cases the image quality is locally different. At the top and bottom borders there are LED lights for illumination which causes the segmentation to erroneously detect reflections as bees. Second the inner zone should be as big as possible to avoid errors by bees running loops. Placing the lines at 25% and 75% of the image height is a compromise between enough distance to the border and a big transition zone. When using multiple counters their lines will now be placed in similar locations, specifically 5 counters in total are placed at 23%, 24%, 25%, 26% and 27% of the image size and 73% to 77% for the second line. Using many more counters does not improve the counting result. If the lines are too close to each other there is a high chance that errors that happen at one counter will occur on the other too. Also putting them too close to the border or making the transition zone too small increases the likelihood for errors.

4.3.2 Analyzing the Results

As explained in the previous subsection (4.3.1) there are two different counters: the **sure count** and the **unsure count**. From these two values it is possible to calculate a **sure count ratio**.

$$sure\ count\ ratio = \frac{sure\_count}{sure\_count + unsure\_count} \tag{4.1}$$

This **sure count ratio** allows to get information about how well the algorithm performed in a specific time interval or on short test sequence. This can be used to self diagnose the accuracy.

When multiple counters are used there also needs to be away to calculate the final counting result. There are two options that have been tried and evaluated:

- Median
- Weighted Average

The first solution is to use the median of all counters for each line. This suppresses extreme values if at one of the line a lot of erroneous counts happened. A case where this can be relevant is when a big cluster is located at the counting line as clusters make bees harder to correctly segment and track.

The second solution is to use a weighted average where each counter gets weighted by it's **sure count ratio**. This factors in an accuracy estimation for each single counter but still let everything have an influence opposed to the median which would remove a single high or low value.

As shown in the evaluation (chapter 5) the difference between those two results is 0.32% or less in the error rates of datasets 1,3 and 4. On the overall error rate the weighted average always performed equal or better than the median but the difference is small.

4.4 Parameters

There a multiple parameters that are required by the segmentation and matching algorithm to work on a specific dataset or recording setup.

- `VIDEO_SCALE`
  This parameter is used to downsize the video prior to all processing to increase the performance at the cost of image quality. The number of pixels scales quadratically with resolution scaling. This heavily reduces the required memory of the segmentation stage and all image filters are faster with fewer pixels to calculate. The evaluation in chapter 5 shows some detailed results on how the resolution affects quality and runtime.
- `BEE_SIZE_X` and `BEE_SIZE_Y`
  These two values define how big a bee is on average in the video. The x size is expected to be the width and the y size the height of a bee. These two values are used to calculate the average bee area by using x and y as the two primary axes of an ellipse.
- `COLOR_SPACE` and `COLOR_CHANNEL`
  These two parameters are used to define which color model and channel are used for the segmentation. The used values for this are HSV Saturation, Hue and HSL Saturation. This has to be chosen depending on the dataset or setup. Chapter 5 contains information on what color model works best with each dataset.
- `BINARY_THRESHOLD`
  This defines the threshold value used for the binary image conversion in the

segmentation stage. The value is an integer between 0 and 255 and has to be chosen depending on the dataset or recording setup. A different color channel also requires a different threshold value. This parameter should be chosen so that most parts of the bees get segmented but the background does not.

- `MEDIAN_FILTER_KERNEL`
  This parameter defines the size for the median filter kernel, it has to be an odd number larger than 1. For example 3 will create a filter kernel for a 3x3 neighborhood. The median filter removes noise from the image and it has shown that a large filter kernel removes more noise, but it should not bee too large and not exceed the size of the morphology structure element. A too large filter kernel can remove bees. Also increasing the filter kernel size has an impact on the runtime. The exact size is dependent on how big a bee is and on the quality of the images. When there is a lot of dirt in the image a larger filter kernel is better at removing it.

- `MORPHOLOGY_DISC_SIZE`
  This parameter defines how big the structure element disc for the morphological opening is. The size of the disc is dependent on the size of a bee in the video and it showed that ideally the size is between 60% to 70% of the width of a bee in pixels. The exact value for optimal results depends on the dataset. The goal of applying morphological opening is to remove remaining noise blobs and to fill out the bee blobs to retrieve the elliptical shapes of the bee bodies. If the disc is too big then bees might get fully removed, if it is too low then a bee might be fractured into smaller unconnected blobs or it is not a single ellipse but a clump of circles, which causes the detection to work less accurate.

- `MAXIMUM_TRACKING_DISTANCE`
  This value is the maximum distance in pixels how far a bee can move between two frame. This value is needed to break up erroneous assignments in the matching stage and also defines the maximum distance for lost bee recovery. This value is dependent on the size of a bee and the framerate. In all test videos the framerate is between 20 and 30fps and around 2/3 of the height of a bee is a reasonable value to avoid errors but not lose tracking when a bee moves fast. If this value is set too low then a fast moving bee can no longer be tracked if it moves further than this value between two frames. If the value is too high incorrect assignments are more likely to happen.

4.4.1 Finding parameters

While some parameters like the bee sizes have to be chosen depending on the video setup and the maximum tracking distance has to be chosen depending on framerate and bee size, the segmentation parameters are more difficult to chose. Especially the color model, color channel and segmentation threshold can not be decided by just analyzing the sizes and framerate.

The idea is to define what the segmentation should look like and then try all possible values to obtain the best parameters. At the beginning one frame or a few frames have to be taken from a dataset and manually segmented to a binary image using an arbitrary image manipulation program. This defines the target segmentation. Now all different color channels are tested with all possible threshold values. The threshold is limited to the 0-255 range.

Another problem is how to evaluate how close a segmentation is to the target image. To obtain a comparable result the F1-score is used. The whole comparison is treated as a two class discrimination problem per pixel. The target frame is the ground truth and each pixel of the segmentation is then evaluated as **True Positive (TP)**, **False Positive (FP)**, **True Negative (TP)** or **False Negative (FN)**. True Positives are pixels that get segmented as white and also are a white pixel in the target frame. False Positives are white pixels that are black in the target frame. True Negatives are pixels that are black in both images and False Negatives are black pixels that are white in the target frame. While True Positives and True Negatives occur when the segmentation is correct, False Positives mean that background gets segmented as bee while False Negatives mean that a part of a bee gets erroneous segmented as background.

From the amount of TP, TN, FP and FN for the whole image the F1 score is calculated. The F1 score is the harmonic average of precision and recall [Sas07]. This measurement takes both cases of wrong segmentation (FN and FP) into account which is important since both segmenting too much and segmenting too little reduce the accuracy of the algorithm. This test only evaluates the initial segmentation threshold but this step is important because further processing depends on it and also the parameter is independent of bee size and framerate. Evaluating median filter and morphology the same way would increase computation time by a lot. Also thresholding is a faster operation than median filtering and morphological opening especially with larger filter kernels or structure elements.

CHAPTER 5

Evaluation

Evaluation is the process to quantify the results of the algorithm. This includes the accuracy of the counting results and the runtime. The different datasets used for the evaluation are presented at the beginning of this chapter.

5.1 Datasets

All datasets have been recorded with the hardware setup present at the institute. The setup was adjusted multiple times and and the algorithm requires different parameters for each dataset. Also all videos from the same dataset were processed with identical parameters. This is needed for long time operation because fine-tuning parameters to a single video does not provide any insight on how stable parameters are over longer time periods.

In total the annotated test data is grouped into 4 different datasets. Dataset 1 has been recorded with a Toshiba BU238MC camera. Dataset 2 has been recorded with a Raspberry Pi camera module. Datasets 3 and 4 have both been recorded with a Axis M1125 camera. These two datasets have been split because of clearly visible differences in camera settings. For a practical application such a change in settings should not be used.

5.1.1 Dataset 1

This is the first dataset that was tested. Compared to datasets 2 and 4 the bee activity and amount of clusters is lower.

The videos are recorded in a resolution of 1416x540 at a framerate of 24 frames per second. The camera model used for this recordings is a Toshiba BU238MC camera. At this size the videos can be processed without downsizing and sill result in a processing

Figure 5.1: An example image of dataset 1

speed that meets the requirements of 5 frames per seconds. The example image of this dataset (5.1) shows one of the moments with the highest bee activity. Compared to the others, this dataset has low bee activity and therefore is expected to give accurate results as bee clusters are close to non existent. In this dataset the background is dark and unsaturated and the bees appear bright. The image appears blurry, but the low bee activity and high contrast of bees to the background allow the algorithm to perform with an average error rate of 2.3% which will be shown in section 5.3 in detail.

Evaluating the color models on this dataset resulted in both HSV and HSL saturation to be the best choices, which can be explained with the low saturation of the background and the high saturation of the bees.

Three specific sequences of this dataset have been ground truth annotated with vatic and evaluated. The second sequence is the supposedly most difficult sequence that was found in this dataset.

- Sequence 1
  - Duration: 30s
  - Average situation
  - Ground Truth: annotated with vatic
- Sequence 2
  - Duration: 25s
  - Highest bee activity in this dataset
  - Ground Truth: annotated with vatic
- Sequence 3

- Duration: 1m33s
- Contains empty parts without bees, longer than other sequences
- Ground Truth: annotated with vatic

5.1.2 Dataset 2

This dataset is recorded with 1920x1080 resolution at 30 fps but the image is not sharp and the bees appear blurred. The recording was performed with a Raspberry Pi camera module which explains the lower quality. Also the lighting situation is hindering for the program as a lot of light comes in from the sides. The background also appears brighter compared to the other datasets. This causes a lot of problems for this algorithm as saturation thresholding is not expected to allow the same distinguishability compared to the other datasets since the background has a low contrast to the bees in terms of saturation. Another observable problem is the inconsistency in the background brightness where the outer parts appear brighter than the inner part of the image which is caused by the light coming in from the sides.

Figure 5.2: An example image of dataset 2

Out of the 4 evaluated datasets, this dataset performs worst with the proposed algorithm. With the used technique of segmentation by binary thresholding no parameters can be found that results in a satisfying segmentation. Single bees can be tracked but splitting bee clusters with k-means is inaccurate because the bee contours are not precise after segmentation. This causes inaccuracy in the area of a bee and the calculation of how many bees are in a cluster does not give accurate results, also this number fluctuates a

lot between frames. Because of this inconsistency between frames a lot of bees are lost during tracking and this further implies that it is unknown on which side a bee entered and therefore the counting has more errors than when the side a bee entered is known.

One specific sequence was ground truth annotated with vatic for this dataset. The sequence is a difficult situation with a cluster located at the upper counting lines.

- Sequence 1
  - Duration: 19s
  - Difficult situation, big cluster
  - Ground Truth: annotated with vatic

As shown in the next section (5.3) this dataset perform worst. This also shows the dependence of the algorithm to the recording setup. The conclusion is that this camera setup should not be used together with the proposed algorithm.

5.1.3 Dataset 3

Dataset number 3 got recorded with 1920x1080 resolution at 30 frames per second. The camera model used for this recordings is an Axis M1125 camera. The camera quality is high resulting in sharp images. The colors are also strong causing the yellow parts of bees to have a highly saturated color.

Figure 5.3: An example image of dataset 3

The only visual problem that occurs in this dataset are artifacts in the video compression caused by streaming. This means that there are image errors where a bee is visible twice or color errors occur where a part of the image is green. Not all sequences contain errors and this also allows to evaluate the algorithm against this kind of errors. Streaming the images from a bee hive is not an unrealistic scenario and could be a possible solution for practical applications therefore it is useful to have test data containing such errors. An example of such artifacts is shown in figure 5.4.

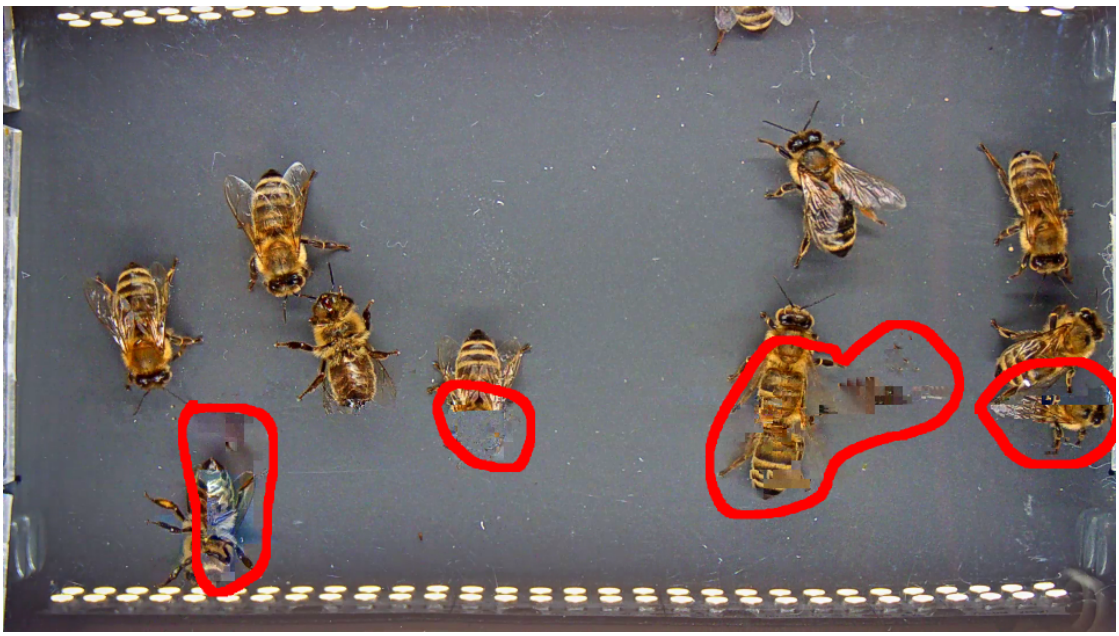


Figure 5.4: An example of video compression artifacts in dataset 3

The recordings in this dataset contain lower bee activity compared to datasets 2 and 4 but two sequences containing bee clusters have specifically be chosen to test the algorithm against. Because of the high image quality the splitting of clusters with k-means is accurate enough to keep track of every single bee in the test sequences even though when a cluster of 5 bees was formed.

- Sequence 1
  - Duration: 18s
  - Average bee activity, contains slight video compression artifacts
  - Ground Truth: annotated with vatic
- Sequence 2
  - Duration: 8s
  - Bees form a cluster and then walk apart, splitting the cluster

  - Ground Truth: manually counted

- Sequence 3
  - Duration: 9s
  - Bees form a cluster and then walk apart, splitting the cluster
  - Ground Truth: manually counted

The figure (5.4) about image artifacts is taken from sequence 1 of dataset 3. A few more short clips have been tested and analyzed manually but since most clips have low bee activity and no clusters occur it is not worth the effort to annotate more simple videos as in such sequences the algorithm has an accuracy of greater than 95% as will be shown in the next section and the evaluation should focus on the most difficult situations that occur while recording.

5.1.4 Dataset 4

Like dataset 3 this dataset has been recorded with an Axis M1125 camera but the conditions are worse with a lot more dirt in the recording area. Also the camera seems to have automatically adapted it's settings which is visible by the blueish appearance of the background. Such a change in settings should be disabled for long time monitoring. These color changes require different parameters than dataset 3 since the background is more saturated compared to the previous recordings in dataset 3. The shift from gray background to a blueish color tone causes the saturation to no longer accurately segment the bees because the saturation between bees and background is now more similar compared to the other datasets. But this change in color tone allows the hue channel to better distinguish the bees now since blue and yellow are complementary and 180 degree apart when visualizing the hue channel on a circle.

The sequences from this dataset have been specifically chosen to be hard situations to evaluate the algorithm in those conditions. One video was fully annotated with vatic but this required a huge time effort of more than 8 hours for a 13s sequence. Therefore only one video was annotated and 3 others were counted manually to estimate the ground truth count. Manually counting these videos is difficult and an error of up to 10% has to be expected. The videos that are annotated with vatic are accurate and the annotation error can be expected to be close to zero, therefore counting results from such videos can be considered as ground truth. While manually counted videos should only be considered as estimation.

- Sequence 1
  - Duration: 12s
  - Extreme situation, a lot of dirt and bee clustering

Figure 5.5: An example image of dataset 4

  - Ground Truth: annotated with vatic

- Sequence 2
  - Duration: 1m16s
  - Hard situation, a lot of dirt and bee clustering
  - Ground Truth: manual counting
- Sequence 3
  - Duration: 1m17s
  - Hard situation, a lot of dirt and bee clustering
  - Ground Truth: manual counting
- Sequence 4
  - Duration: 1m23s
  - Hard situation, a lot of dirt and bee clustering
  - Ground Truth: manual counting

5.2 Color Model Parameters

The optimal color model and channel parameters for each dataset are found by comparing a segmentation with a manual ground truth segmentation and calculating the F1-score on a per pixel basis. This gives an estimation which parameters are ideal for the whole dataset. This step does not evaluate the final segmentation after filtering but the initial segmentation which is the first step of the whole algorithm. As there is filtering in later steps the borders are not of bigger importance than inner pixels. Each pixel is of equal importance in this step as the thresholding and then median filtering needs to create a basis for the morphology stage and calculating the F1 score gives a measurement that both considers false positives and false negatives and weighs each pixel equally.

Dataset 1

Three different frames from the dataset were taken and manually segmented with an image manipulation program. The parameters were calculated from the average result of the three frames.

Table 5.1 shows the color model evaluation for this dataset, the presented values are the best found threshold value and the F1 score of the segmentation that gets produced by it. As there are three images the F1 score is averaged for the same parameter value across all three images. The value range for the threshold is from 0 to 255 as the images are stored as one byte per color channel. The F1 score values are rounded to 3 digits after zero. The color channels are tested normally and inverted. Inverted means the binary image is negated after thresholding.

The results from table 5.1 show that the following color channels should be further investigated: HSV Saturation, HSL Saturation, u, v, Cb inverted and Cr. The overall winner is the inverted segmentation of the Cb channel, but the other mentioned channels are not far behind. In further practical tests L*u*v*, YCrCb and L*a*b* color models have shown to not allow segmentation with constant parameters over longer time periods. The ideal parameter range shifts over the duration of only minutes so a constant threshold does not allow to segment the bees in test sequences shorter than 5 minutes. On the other hand HSV and HSL Saturation are constant over time and even sequences taken hours apart work with the same parameter. This suggests that HSV and HSL Saturation are the best color channels to solve this segmentation task.

Dataset 2

The testing procedure is obviously the same as with dataset 1. The results from this dataset are expected to be worse as the image quality and situations are seem less suitable for the chosen methodology. This recording setup is already superseded but testing against it still can provide valuable insights on different conditions when and when not the algorithm performs well.

Color Channel	Threshold	F1 Score
Blue	137	0.384
Blue inverted	254	0.109
Green	131	0.559
Green inverted	254	0.110
Red	140	0.668
Red inverted	254	0.102
Hue	0	0.111
Hue inverted	29	0.582
HSV Saturation	41	0.795
HSV Saturation inverted	183	0.110
HSV Value	142	0.660
HSV Value inverted	254	0.102
HSL Lighting	136	0.576
HSL Lighting inverted	254	0.110
HSL Saturation	**25**	**0.818**
HSL Saturation inverted	251	0.103
YCrCb Y	135	0.588
YCrCb Y inverted	254	0.110
YCrCb Cr	134	0.802
YCrCb Cr inverted	175	0.110
YCrCb Cb	0	0.110
YCrCb Cb inverted	**123**	**0.822**
CIE L*a*b* L	141	0.587
CIE L*a*b* L inverted	254	0.110
CIE L*a*b* a	134	0.629
CIE L*a*b* a inverted	163	0.110
CIE L*a*b* b	132	0.817
CIE L*a*b* b inverted	179	0.110
CIE L*u*v* L	143	0.587
CIE L*u*v* L inverted	254	0.110
CIE L*u*v* u	102	0.795
CIE L*u*v* u inverted	144	0.110
CIE L*u*v* v	140	0.808
CIE L*u*v* v inverted	199	0.110
XYZ X	130	0.596
XYZ X inverted	242	0.110
XYZ Y	132	0.583
XYZ Y inverted	254	0.110
XYZ Z	148	0.414
XYZ Z inverted	254	0.109
Greyscale	135	0.588
Greyscale inverted	254	0.110

Table 5.1: Color model testing results for dataset 1

Due known problems or difficulties with L*u*v*, YCrCb and L*a*b* these color channels have been already ruled out and will be excluded in the results of this test and further tests. Also the XYZ color space performs worse than RGB in these tests and the datasets are similar enough to expect no changes to this. Therefore these channels are also left out.

Color Channel	Threshold	F1 Score
Blue	0	0.338
Blue inverted	92	0.607
Green	15	0.340
Green inverted	102	0.442
Red	175	0.498
Red inverted	254	0.340
Hue	0	0.342
Hue inverted	15	0.615
HSV Saturation	**98**	**0.793**
HSV Saturation inverted	254	0.338
HSV Value	175	0.495
HSV Value inverted	254	0.340
HSL Lighting	32	0.340
HSL Lighting inverted	217	0.349
HSL Saturation	85	0.762
HSL Saturation inverted	253	0.338
Greyscale	28	0.340
Greyscale inverted	212	0.344

Table 5.2: Color model testing results for dataset 2

The best F1 score of 0.79 (table 5.2) is close to the best value of dataset 1, although applying this parameters does not allow to accurately segment the bee contours. While single bees can be tracked fine, the lighting conditions and low image quality cause clusters to not be able to be splitted correctly as the amount of bees in a cluster can not be accurately estimated. The problems are shadowed areas between bees or holes in a bee cluster where the background is visible. Due the light background the shadows of the bees cause this areas to be segmented as bees. Because of this the estimation of the K parameter for k-means does not work reliably by using the area of the bees.

Image 5.6 shows an example image of the dataset where this problems are visible. In the untransformed image it is already noticeable that the bees are quite blurry and the lighting conditions are not equal across the image as there is a lot of light coming in from the sides and casting shadows.

Figure 5.6: An example image of dataset 2 to show the difficulties in this dataset

The HSV saturation color channel (shown in image 5.7) which resulted in being the best option concluded by the color model test, does not show clear borders between the bees and the background.

After thresholding, applying median filtering and morphological opening the segmentation of the clusters does not resemble the actual bee bodies. The main problem is that spaces between bees that belong to the background get segmented as bee. This causes the area of the clusters to be unstable in context to the number of bees. The different processing steps on the example image are shown in image 5.8.

Dataset 3

This dataset provides clean images with saturated colors and homogeneous lighting. The only difficulty are video compression artifacts that happened during recording.

As expected the HSV and HSL Saturation score best in this test as seen in table 5.3. The conditions of this dataset are ideal for using the saturation to segment the image as the

Figure 5.7: The color channel used for segmentation of 5.6

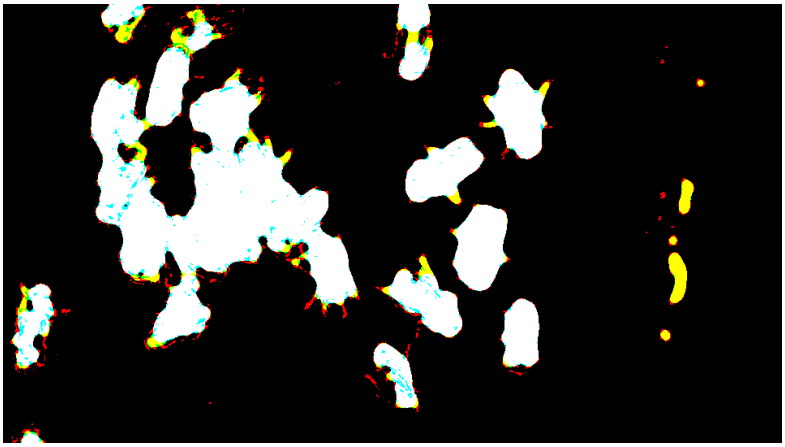


Figure 5.8: The segmentation steps visualized for image 5.6. Threshold is the red channel, median filtered result is the green channel and the result of the morpholohical opening is the blue channel.

background is unsaturated greyish and the bees are highly saturated. The F1 score is not as high as in the datasets before but this does not conclude a worse overall segmentation quality since applying a median filter and morphological opening are applied afterward and improve the quality differently on each dataset. The lighting conditions are also close to perfect in this setup as the brightness appears homogeneous across the whole image. The shadows also blend well with the background and do not interfere in the segmentation.

Dataset 4

The best segmentation for this dataset is found in the hue channel as shown in table 5.4. Opposed to other datasets the saturation does not work well because the background

Color Channel	Threshold	F1 Score
Blue	0	0.108
Blue inverted	54	0.572
Green	133	0.230
Green inverted	63	0.346
Red	131	0.409
Red inverted	73	0.169
Hue	0	0.115
Hue inverted	22	0.556
HSV Saturation	**98**	**0.763**
HSV Saturation inverted	254	0.108
HSV Value	140	0.375
HSV Value inverted	78	0.190
HSL Lighting	133	0.197
HSL Lighting inverted	69	0.358
HSL Saturation	64	0.719
HSL Saturation inverted	253	0.107
Greyscale	132	0.259
Greyscale inverted	68	0.315

Table 5.3: Color model testing results for dataset 3

Color Channel	Threshold	F1 Score
Blue	0	0.470
Blue inverted	85	0.572
Green	0	0.472
Green inverted	248	0.473
Red	104	0.580
Red inverted	254	0.472
Hue	0	0.469
Hue inverted	**39**	**0.750**
HSV Saturation	93	0.538
HSV Saturation inverted	253	0.470
HSV Value	0	0.473
HSV Value inverted	254	0.472
HSL Lighting	0	0.473
HSL Lighting inverted	246	0.473
HSL Saturation	59	0.570
HSL Saturation inverted	252	0.469
Greyscale	0	0.473
Greyscale inverted	247	0.473

Table 5.4: Color model testing results for dataset 4

appears blueish and is more saturated than the background in other datasets. Also the plate below the camera is dirty and the overall image quality suffers from this. The bees appear less saturated than they do in the other datasets. Combined with the more saturated background this does not allow a satisfying segmentation with the saturation channels.

The Hue channel is the clear winner in this test and the result can easily be explained with the blueish background having a high contrast to the bees whose colors are mainly

in the yellow-red area. In the Hue channel red color is at 0° and yellow at 60° while blue is at 240°. When the hue is put on a circle those two color tones are nearly 180° apart and therefore complimentary.

5.3 Quality Results

The result of the counting are two integer numbers standing for the amount of bees that entered and exited the hive respectively. Since it can be ambiguous in which rotation the camera is mounted the resulting values are named **count up** and **count down**. The testing is done with five different counting windows with slight offsets. The results table of each video sequence presents the median count and the weighted average count (denoted as *W. Average* in the tables) at both sides. The weighted average is the average of each counter weighted by it's sure count ratio.

5.3.1 Dataset 1

As presented in subsection 5.1.1 for this dataset three video sequences have been fully annotated with vatic.

Counter	Sure Count	Unsure Count	Total Count	Sure count ratio
Counter 1 up	41	3	**44**	0.932
Counter 1 down	62	2	**64**	0.969
Counter 2 up	41	3	**44**	0.932
Counter 2 down	62	2	**64**	0.969
Counter 3 up	41	3	**44**	0.932
Counter 3 down	61	3	**64**	0.953
Counter 4 up	41	3	**44**	0.932
Counter 4 down	63	1	**64**	0.984
Counter 5 up	41	3	**44**	0.932
Counter 5 down	63	1	**64**	0.984
Median up			**44**	
Median down			**64**	
W. Average up			**44**	0.932
W. Average down			**64**	0.972
Ground truth up			**44**	
Ground truth down			**61**	

Table 5.5: Results for sequence 1 of dataset 1

The results for the first sequence of this dataset, as seen in table 5.5, results in an exact match on the count at the upper side and 3 bees too much at the lower counting line.

The second sequence performs similar to the first one. The results shown in table 5.5 show an exact match at the upper counting line and a difference of 2 bees at the lower

Counter	Sure Count	Unsure Count	Total Count	Sure count ratio
Counter 1 up	41	8	**49**	0.837
Counter 1 down	42	5	**47**	0.894
Counter 2 up	41	7	**48**	0.854
Counter 2 down	42	5	**47**	0.894
Counter 3 up	41	7	**48**	0.854
Counter 3 down	41	6	**47**	0.872
Counter 4 up	42	7	**49**	0.857
Counter 4 down	42	6	**48**	0.875
Counter 5 up	43	5	**48**	0.896
Counter 5 down	42	6	**48**	0.875
Median up			**48**	
Median down			**47**	
W. Average up			**48.394**	0.860
W. Average down			**47.397**	0.882
Ground truth up			**48**	
Ground truth down			**49**	

Table 5.6: Results for sequence 2 of dataset 1

one compared to the ground truth. Both median and weighted average after rounding provide the same results.

Counter	Sure Count	Unsure Count	Total Count	Sure count ratio
Counter 1 up	49	1	**50**	0.980
Counter 1 down	47	4	**51**	0.922
Counter 2 up	49	2	**51**	0.961
Counter 2 down	47	4	**51**	0.922
Counter 3 up	49	1	**50**	0.980
Counter 3 down	44	7	**51**	0.863
Counter 4 up	50	1	**51**	0.980
Counter 4 down	50	3	**53**	0.943
Counter 5 up	50	1	**51**	0.980
Counter 5 down	50	4	**54**	0.926
Median up			**51**	
Median down			**51**	
W. Average up			**50.599**	0.976
W. Average down			**52.020**	0.915
Ground truth up			**50**	
Ground truth down			**53**	

Table 5.7: Results for sequence 3 of dataset 1

In the third sequence (table 5.7) the difference is 1 bee on the upper side for both median

and weighted average after rounding. On the lower side the weighted average is only 1 bee off, while the median counted 2 bees too less.

Sequence	count type	Count	Ground Truth	Error rate
Sequence 1 up	median	44	44	0%
Sequence 1 down	median	64	61	4.91%
Sequence 1 up	weighted average	44	44	0%
Sequence 1 down	weighted average	64	61	4.91%
Sequence 2 up	median	48	48	0%
Sequence 2 down	median	47	49	4.08%
Sequence 2 up	weighted average	48	48	0%
Sequence 2 down	weighted average	47	49	4.08%
Sequence 3 up	median	51	50	2%
Sequence 3 down	median	51	53	3.77%
Sequence 3 up	weighted average	51	50	2%
Sequence 3 down	weighted average	52	53	1.89%

Table 5.8: Overview of results and error rates for dataset 1.

In table 5.8 the results from all three sequences can be compared and also the error rates are given. All sequences have less than 5% error compared to the ground truth values, some counters even give the exact result. The error rate has been calculated as the absolute difference between the counting result and the ground truth count divided by the ground truth count.

When comparing the median count with the weighted average count both give similar results with weighted average being more accurate in case of sequence 3.

Count type	Errors	Total counting events	Error rate
Median	8	305	**2.62%**
Weighted Average	7	305	**2.3%**

Table 5.9: Overall error rates for dataset 1.

Table 5.9 shows the combined error rates of this dataset by dividing the sum of errors through the total amount of ground truth counting events resulting in the error rates of 2.62% and 2.3%.

5.3.2 Dataset 2

In this dataset only one sequence was annotated with ground truth. As there are difficulties with the conditions in this dataset and the annotation is time consuming only this one sequence is evaluated.

As expected the results shown in 5.10 to not meet the targeted accuracy requirement of 10% error rate and exceeds it clearly with 68%. While this is only one sequence with a hard situation, such a situation can occur in this dataset and the results show that this recording setup is unsuitable for the proposed algorithm.

Counter	Sure Count	Unsure Count	Total Count	Sure count ratio
Counter 1 up	5	22	**27**	0.185
Counter 1 down	2	5	**7**	0.286
Counter 2 up	4	26	**30**	0.133
Counter 2 down	2	4	**6**	0.333
Counter 3 up	4	28	**32**	0.125
Counter 3 down	1	5	**6**	0.167
Counter 4 up	5	22	**27**	0.185
Counter 4 down	3	4	**7**	0.429
Counter 5 up	5	26	**31**	0.161
Counter 5 down	4	5	**9**	0.444
Median up			**30**	
Median down			**7**	
W. Average up			**29.114**	0.158
W. Average down			**7.234**	0.332
Ground truth up			**15**	
Ground truth down			**10**	

Table 5.10: Results for test sequence 1 of dataset 2

Count type	Errors	Total counting events	Error rate
Median	18	25	**72%**
Weighted Average	17	25	**68%**

Table 5.11: Overall error rates for dataset 2.

The overall error rates for this datasets are 72% and 68% (as shown in table 5.11). The reason for this is that clusters can not be resolved properly in this dataset and a cluster occurs at the top counting line in the test sequence. This causes a lot of erroneous counts at this position. It is important to evaluate the difficult cases and there this dataset is unsuitable for the proposed algorithm.

5.3.3 Dataset 3

Sequence 2 as shown in table 5.13 is a short video with only 15 counting events. In this sequence some bees form a cluster and then walk apart splitting the cluster. This is interesting as it can be used to evaluate how well clusters can be handled. In this case the algorithm performed perfect and gave the same result as the ground truth. It is also worth to note that this situation demonstrates how using multiple counters improves the counting accuracy as two of the five counters give an incorrect result on the lower line.

The third sequence is a similar situation as sequence 2 and the results are shown in table 5.14. As with sequence 2 the counting result is an exact match with the ground truth. The sure count ratio can not be used as general quality indicator for accuracy when the test sequence is short or the total number of counting events is low as in this case because

Counter	Sure Count	Unsure Count	Total Count	Sure count ratio
Counter 1 up	21	2	**23**	0.913
Counter 1 down	24	1	**25**	0.960
Counter 2 up	21	2	**23**	0.913
Counter 2 down	24	1	**25**	0.960
Counter 3 up	21	2	**23**	0.913
Counter 3 down	24	1	**25**	0.960
Counter 4 up	21	2	**23**	0.913
Counter 4 down	24	1	**25**	0.960
Counter 5 up	21	2	**23**	0.913
Counter 5 down	24	1	**25**	0.960
Median up			**23**	
Median down			**25**	
W. Average up			**23**	0.913
W. Average down			**25**	0.960
Ground truth up			**24**	
Ground truth down			**25**	

Table 5.12: Results for sequence 1 of dataset 3

Counter	Sure Count	Unsure Count	Total Count	Sure count ratio
Counter 1 up	6	4	**10**	0.600
Counter 1 down	4	1	**5**	0.800
Counter 2 up	6	4	**10**	0.600
Counter 2 down	4	1	**5**	0.800
Counter 3 up	6	4	**10**	0.600
Counter 3 down	4	1	**5**	0.800
Counter 4 up	7	3	**10**	0.700
Counter 4 down	4	2	**6**	0.667
Counter 5 up	7	3	**10**	0.700
Counter 5 down	4	2	**6**	0.667
Median up			**10**	
Median down			**5**	
W. Average up			**10**	0.640
W. Average down			**5.357**	0.747
Ground truth up			**10**	
Ground truth down			**5**	

Table 5.13: Results for sequence 2 of dataset 3

unsure counts are also generated by bees that are located inside the counting area at the beginning of the video sequence. With longer sequences or higher number of bees this is negligible but in the case of sequence 2 and 3 it has an impact on the sure count ratio.

Tables 5.15 and 5.16 show the individual and combined error rates for the test sequences

Counter	Sure Count	Unsure Count	Total Count	Sure count ratio
Counter 1 up	4	3	**7**	0.571
Counter 1 down	1	1	**2**	0.500
Counter 2 up	4	3	**7**	0.571
Counter 2 down	1	1	**2**	0.500
Counter 3 up	4	3	**7**	0.571
Counter 3 down	1	1	**2**	0.500
Counter 4 up	4	3	**7**	0.571
Counter 4 down	1	1	**2**	0.500
Counter 5 up	4	3	**7**	0.571
Counter 5 down	1	1	**2**	0.500
Median up			**7**	
Median down			**2**	
W. Average up			**7**	0.571
W. Average down			**2**	0.500
Ground truth up			**7**	
Ground truth down			**2**	

Table 5.14: Results for sequence 3 of dataset 3

Sequence	count type	Count	Ground Truth	Error rate
Sequence 1 up	median	23	24	4.17%
Sequence 1 down	median	25	25	0%
Sequence 1 up	weighted average	23	24	4.17%
Sequence 1 down	weighted average	25	25	0%
Sequence 2 up	median	10	10	0%
Sequence 2 down	median	5	5	0%
Sequence 2 up	weighted average	10	10	0%
Sequence 2 down	weighted average	5	5	0%
Sequence 3 up	median	7	7	0%
Sequence 3 down	median	2	2	0%
Sequence 3 up	weighted average	7	7	0%
Sequence 3 down	weighted average	2	2	0%

Table 5.15: Overview of results and error rates for dataset 3.

Count type	Errors	Total counting events	Error rate
Median	1	73	**1.37%**
Weighted Average	1	73	**1.37%**

Table 5.16: Overall error rates for dataset 3.

for this dataset. With only 1 counting error in 73 counting events according to the ground truth the error rate is low with 1.37%. Overall the proposed algorithm performs well on this dataset. Although it is worth to mention that even when there are clusters

no crowded situations where bees take more space than the background occurs. So this dataset can only be classified as of medium difficulty.

5.3.4 Dataset 4

This dataset contains the most difficult situations in context of number of bees and clusters. The amount of dirt on the glass pane causes additional difficulty and degradation of image quality.

Counter	Sure Count	Unsure Count	Total Count	Sure count ratio
Counter 1 up	17	29	**46**	0.370
Counter 1 down	19	24	**43**	0.442
Counter 2 up	16	31	**47**	0.340
Counter 2 down	18	24	**42**	0.429
Counter 3 up	15	31	**46**	0.326
Counter 3 down	16	24	**40**	0.400
Counter 4 up	19	28	**47**	0.404
Counter 4 down	20	25	**45**	0.444
Counter 5 up	19	28	**47**	0.404
Counter 5 down	23	22	**45**	0.511
Median up			**47**	
Median down			**43**	
W. Average up			**46.623**	0.369
W. Average down			**43.127**	0.445
Ground truth up			**39**	
Ground truth down			**42**	

Table 5.17: Results for sequence 1 of dataset 4

This sequence is a difficult one and was explicitly chosen for that reason. It is the supposedly most difficult time window found in the dataset. Therefore it is expected that the results are worse than easier videos, although the lower count was quite accurate with only 1 bee off. The upper count is 20% off which is worse than the goal of <10% error but it is only a short sequence and the average error can be lower across the whole dataset and such an inaccuracy spike in extreme data will not necessarily have a noticeable impact on longtime usage. The image quality is also lower than dataset 3 due to more dirt on the glass pane.

Sequence 2, as shown in table 5.18, results in only 2 bees off compared to the ground truth at the upper counting line when using the weighted average count. The lower line counted 6 bees wrong after rounding and is 3% different to the ground truth.

In table 5.19 the results for sequence 3 are shown. This sequence is more troublesome than sequence 2 and has some errors at the upper counting line. The lower line was quite accurate though with a difference of 4 which is only around 1.7% error. The upper line has difficulties because a cluster forms at that position. While the result at that

Counter	Sure Count	Unsure Count	Total Count	Sure count ratio
Counter 1 up	125	73	**198**	0.631
Counter 1 down	140	64	**204**	0.686
Counter 2 up	118	83	**201**	0.587
Counter 2 down	133	65	**198**	0.672
Counter 3 up	109	91	**200**	0.545
Counter 3 down	127	67	**194**	0.655
Counter 4 up	130	77	**207**	0.628
Counter 4 down	144	62	**206**	0.699
Counter 5 up	131	74	**205**	0.639
Counter 5 down	147	58	**205**	0.717
Median up			**201**	
Median down			**204**	
W. Average up			**202.282**	0.606
W. Average down			**201.532**	0.686
Ground truth up			**204**	
Ground truth down			**196**	

Table 5.18: Results for sequence 2 of dataset 4

Counter	Sure Count	Unsure Count	Total Count	Sure count ratio
Counter 1 up	95	171	**266**	0.357
Counter 1 down	89	154	**243**	0.366
Counter 2 up	92	175	**267**	0.345
Counter 2 down	83	154	**237**	0.350
Counter 3 up	79	184	**263**	0.300
Counter 3 down	77	158	**235**	0.328
Counter 4 up	101	169	**270**	0.374
Counter 4 down	99	149	**248**	0.399
Counter 5 up	107	173	**280**	0.382
Counter 5 down	106	150	**256**	0.414
Median up			**267**	
Median down			**243**	
W. Average up			**269.577**	0.352
W. Average down			**244.430**	0.371
Ground truth up			**214**	
Ground truth down			**240**	

Table 5.19: Results for sequence 3 of dataset 4

line has an 26% error rate it can still be evened out to a lower total error over longer time periods. Also it is important to note that the ground truth for this sequence was manually counted and this is a difficult task for a human therefore it is likely that this manual counted ground truth is inaccurate. An estimated error of up to 10% can be

expected.

Counter	Sure Count	Unsure Count	Total Count	Sure count ratio
Counter 1 up	106	110	**216**	0.491
Counter 1 down	120	90	**210**	0.571
Counter 2 up	99	122	**221**	0.448
Counter 2 down	112	93	**205**	0.546
Counter 3 up	94	116	**210**	0.448
Counter 3 down	107	91	**198**	0.540
Counter 4 up	110	107	**217**	0.507
Counter 4 down	129	85	**214**	0.603
Counter 5 up	113	104	**217**	0.521
Counter 5 down	132	81	**213**	0.620
Median up			**217**	
Median down			**210**	
W. Average up			**216.241**	0.483
W. Average down			**208.283**	0.576
Ground truth up			**203**	
Ground truth down			**189**	

Table 5.20: Results for sequence 4 of dataset 4

Table 5.20 shows the results for the fourth sequence of dataset 4. In this sequence the error is around 6.5% at the upper line and around 10% on the lower line. The ground truth for this sequence is manually counted and can be inaccurate.

Sequence	count type	Count	Ground Truth	Error rate
Sequence 1 up	median	47	39	20.51%
Sequence 1 down	median	43	42	2.38%
Sequence 1 up	weighted average	47	39	20.51%
Sequence 1 down	weighted average	43	42	2.38%
Sequence 2 up	median	201	204	1.47%
Sequence 2 down	median	204	196	4.08%
Sequence 2 up	weighted average	202	204	0.98%
Sequence 2 down	weighted average	202	196	3.1%
Sequence 3 up	median	267	214	24.42%
Sequence 3 down	median	243	240	1.25%
Sequence 3 up	weighted average	270	214	26.17%
Sequence 3 down	weighted average	244	240	1.67%
Sequence 4 up	median	217	203	6.89%
Sequence 4 down	median	210	189	11.11%
Sequence 4 up	weighted average	216	203	6.4%
Sequence 4 down	weighted average	208	189	10.05%

Table 5.21: Overview of results and error rates for dataset 4.

Table 5.21 shows the summary of all error rates on all sequences on both counting lines

with both ways of combining the multiple counters. Also it is important to note that sequences 2,3 and 4 where manually annotated and the ground truth can be inaccurate, with an estimation of up to 10% error.

Count type	Errors	Total counting events	Error rate
Median	111	1327	**8.36%**
Weighted Average	109	1327	**8.21%**

Table 5.22: Overall error rates for dataset 4.

The overall error rate of the whole dataset (shown in table 5.22) is around 8% error on both lines despite being a difficult dataset. This is below the goal of 10% average error rate.

5.4 Observations

From the results a few important conclusions can be drawn. First the results show that recording conditions have a huge influence on the results as dataset 2 is unusable with the proposed algorithm. The color of the background has a huge impact on the segmentation quality and should be complimentary to the bees. Depending on color of the background this complimentary coloring can be achieved with a different color model. Bees have a high saturation and when using a camera that produces colorful and saturated images the saturation in both HSV and HSL color models performs well to segment the bees from an unsaturated background as shown in datasets 1 and 3. A blueish background color can best be segmented in the hue color channel.

5.5 Runtime

After analyzing the accuracy of the algorithm the next step is to evaluate the runtime performance. The runtime testing will be done on the same datasets. Dataset 1 is used with full resolution as it has a smaller resolution. The other datasets are treated as 50% scale baseline. The results from section 5.3 have been evaluated with these video resolutions. Dataset 1 has a resolution of 1416x540 and will be presented in an own table (5.23) as the image resolution is different.

Sequence	Average Frametime	FPS
1	96.3ms	10.38
2	96.01ms	10.42
3	95.37ms	10.49

Table 5.23: Runtime of dataset 1

Datasets 2,3 and 4 are combined in a single table (5.24) as they have the same resolution with 1920x1080 at 50% scaling resulting in a resolution of 960x540 pixel.

Dataset	Sequence	Average Frametime	FPS
2	1	89.43ms	11.18
3	1	57.11ms	17.51
3	2	56.89ms	17.58
3	3	58.69ms	17.04
4	1	78.26ms	12.78
4	2	73.62ms	13.58
4	3	65.35ms	15.30
4	4	66.08ms	15.13

Table 5.24: Runtime of dataset 2,3 and 4

Dataset 3 has less bees active at the same time and provides better framerates. In dataset 4 and 2 the number of bees is constantly high resulting in the higher runtime. The ratio of the worst case compared to the best case is 57.1% longer runtime.

Resolution influence

While the tests have been done on a fixed resolution it is interesting how much influence on runtime the resolutions has and how this affects the quality results.

This tests will focus on dataset 4 as it is the most difficult situation and the amount of counting events in the ground truth data is the highest.

Seq	Error	FPS	Err 50% Res	FPS 50% Res	Ratio of Err	Ratio of RT
1	12	1.88	9	12.78	133%	15%
2	12	2.02	8	13.58	150%	15%
3	35	2.06	60	15.3	58%	13%
4	24	1.93	32	15.13	75%	13%

Table 5.25: Influence of resolution on runtime and accuracy on Dataset 4. Columns: sequence (Seq), error in counting at full resolution (Error), frames per second at full resolution (FPS), errors at 50% resolution (Err 50% Res), FPS with 50% resolution (FPS 50% Res), ratio of errors between 100% and 50% (Ratio of Err), ratio of run times between 100% and 50%. (Ratio of RT)

Table 5.25 shows the differences and ratios of runtime and counting errors when comparing full resolution with half resolution. What can be seen is that in the case of sequences 1 and 2 using half resolution actually has better results. In case of sequence 2 this is negligible as the ground truth has 400 counting events and both an error of 12 and 8 is quite low. This difference can be caused by the randomness of the k-means initialization or the minor parameter differences. Since the median filter kernel size has to be an odd

number it cannot be perfectly doubled or divided by 2 and has to be either rounded up or down.

Sequence 1 is shorter and only has 81 counted bees according to the ground truth. The relative error influence is higher here but with an error difference of only 3 this is also expected to be caused by the randomness of the k-means initialization especially when considering that this sequence contains a lot of clustering.

Sequences 3 and 4 performed better with full resolution. 35 errors versus 60 is a clear improvement in the case of sequence 3 and not only caused by the k-means randomness. Sequence 3 performed worst out of dataset 4 which shows that using the full resolution does improve the accuracy in difficult situations. Although in situation where less counting errors happen the higher resolution does not improve the accuracy at all.

As for the runtime using the full resolution only allows around 2 frames per second to be processed on a desktop PC which is on average only 14% of the framerate that can be achieved when using half resolution.

To finish the resolution influence testing it is necessary to compare to quarter resolution videos. The testing is also performed on dataset 4.

Seq	Err 25%	FPS 25%	Err 50%	FPS 50%	Ratio of Err	Ratio of RT
1	4	60.35	9	12.78	44%	472%
2	4	65.88	8	13.58	50%	485%
3	60	72.46	60	15.3	100%	474%
4	23	71.17	32	15.13	72%	470%

Table 5.26: Influence of resolution on runtime and accuracy with quarter resolution. Columns: sequence (Seq), error in counting at 25% resolution (Err 25%), frames per second at 25% resolution (FPS 25%), errors at 50% resolution (Err 50%), FPS with 50% resolution (FPS 50%), ratio of errors between 25% and 50% (Ratio of Err), ratio of run times between 25% and 50%. (Ratio of RT)

In table 5.26 the comparison of quarter resolution with half resolution is shown. Surprisingly quarter resolution actually performed better than half resolution accuracy wise. All videos have less counting errors or equal compared to half resolution. In the case of sequences 1,2 and 4 this difference can again be caused by the randomness of the k-means initialization. These differences can also be cause by parameter differences like median filter kernel size which has to be rounded. But it seems that decreasing the resolution to 480x270 which is a quarter of the original resolution does not decrease the accuracy of the counting result compared to using 50% resolution.

As for the runtime the lower resolution allows more than 60 fps which is over double real-time for the test sequences. This is interesting for practical use as actual realm-time performance could be achieved with lower processing power than a current desktop PC.

To better verify this conclusion datasets 1 and 3 will be tested too. Dataset 2 is left out as it was declared unsuitable for this algorithm.

Seq	Err 25%	FPS 25%	Err 50%	FPS 50%	Ratio of Err	Ratio of RT
1	3	121.36	3	10.38	100%	1169%
2	2	123.15	2	10.42	100%	1182%
3	1	128.04	2	10.49	50%	1221%

Table 5.27: Influence of resolution on runtime and accuracy with quarter resolution for datasets 1. Columns: sequence (Seq), error in counting at 25% resolution (Err 25%), frames per second at 25% resolution (FPS 25%), errors at 50% resolution (Err 50%), FPS with 50% resolution (FPS 50%), ratio of errors between 25% and 50% (Ratio of Err), ratio of run times between 25% and 50%. (Ratio of RT)

Seq	Err 25%	FPS 25%	Err 50%	FPS 50%	Ratio of Err	Ratio of RT
1	2	95.9	1	17.51	200%	548%
2	0	99.5	0	17.58	100%	566%
3	0	97.75	0	17.04	100%	574%

Table 5.28: Influence of resolution on runtime and accuracy with quarter resolution for datasets 3. Columns: sequence (Seq), error in counting at 25% resolution (Err 25%), frames per second at 25% resolution (FPS 25%), errors at 50% resolution (Err 50%), FPS with 50% resolution (FPS 50%), ratio of errors between 25% and 50% (Ratio of Err), ratio of run times between 25% and 50%. (Ratio of RT)

In tables 5.28 and 5.28 the results for quarter resolution for datasets 1 and 3 is shown. With one more error in dataset 1 and one less error in dataset 3 the overall error count stayed about equal. The runtime improved significantly with over 500% in case of dataset 3 and even above 1000% in the case of dataset 1. The difference of the improvement through downscaling is explainable by the difference of the parameters for the kernel sizes. Dataset 1 contains nearly no clusters and therefore the median filtering stage is less important and a lower kernel size is sufficient. The influence of the different parameters and algorithms on the runtime will be analyzed in section 5.6.1.

What can be concluded from the testing with different resolutions is that in cases with little clustering or a high image quality the resolution can be reduced a lot and the counting accuracy does not get worse. At quarter resolution the runtime that can be achieved is already high and exceeds realtime framerate which is between 20 and 30 fps for the test videos.

Reducing the resolution has other downsides though. If the bee images should be extracted for a further processing which is something that is planned to do with this setup, it is required to scale up the positions and orientations which is likely less accurate

when done with a lower resolution. But when only counting numbers are considered quarter resolution does work well.

Since quarter resolution does work quite well it is interesting to see what happens if the resolution gets reduced even further to 1/8 of full resolution.

Seq	Err 12.5%	FPS 12.5%	Err 50%	FPS 50%	Ratio of Err	Ratio of RT
1	4	108.19	9	12.78	44%	847%
2	40	138.08	8	13.58	500%	1017%
3	83	142.11	60	15.3	138%	929%
4	45	137.05	32	15.13	141%	906%

Table 5.29: Influence of resolution on runtime and accuracy with 1/8 resolution. Columns: sequence (Seq), error in counting at 12.5% resolution (Err 12.5%), frames per second at 12.5% resolution (FPS 12.5%), errors at 50% resolution (Err 50%), FPS with 50% resolution (FPS 50%), ratio of errors between 12.5% and 50% (Ratio of Err), ratio of run times between 12.5% and 50%. (Ratio of RT)

The results in table 5.29 show the results of 1/8 resolution (12.5%) compared to the baseline scaling used for testing of half resolution. While the results of quarter resolution showed significant improvements in runtimes without increasing the error, reducing the resolution even further does not seem to be an overall improvement. Error rates increased in most cases while the runtime is only around twice as fast compared to quarter resolution. Considering the amount of pixels reduces quadratically when halving the resolution the runtime increase is a bad tradeoff. Only in the case of sequence 1 the errors got less but this can be explained by a lucky situation. Overall the conclusion is that using a smaller resolution than 25% of 1920x1080 does not seem an improvement when both considering the runtime and accuracy.

Overall 25% resolution of 1920x1080 seems to be the best combination of accuracy and runtime. On extreme situations the low resolution might get difficulties, but even there it did not give worse results than 50% resolution and using full resolution is a significant runtime increase. This applies to counting results only, if the trajectories should be retrieved lowering the resolution does have a negative impact. The ratio of sure counts to total counting events decreases on average when lowering the resolution and unsure counting events are caused by lost tracks concluding the trajectory quality will be worse. But when only the counting result is needed using 25% resolution on full HD videos is fine in the used hardware setup.

5.6 Longtime Test

To better verify the quality of the algorithm a long time test was performed by processing the recordings of multiple days. The data is split in 3 minute video sequences. The whole dataset contains 1460 files. The following recording time windows are available:

- 29.9.2017 15:12-19:39
- 30.9.2017 06:00-21:00 (Missing: 6:03-6:48)
- 1.10.2017 06:00-21:00
- 2.10.2017 06:00-21:00
- 3.10.2017 06:00-09:24
- 4.10.2017 09:18-21:00
- 5.10.2017 06:00-15:15

The processing was done in 25% resolution of full-HD (480x270) to reduce the required processing time which is still around 2-3 days for this amount of data. Overall the sure count ratio of each video averages to over 94% in both directions.

Date	Start time	End time	In	Out	°C 7h	°C 14h	°C 19h
2017-09-29	15:15	19:39	4859	3276	9	19	15
2017-09-30	06:00	21:00	17943	17904	9	18	14
2017-10-01	06:00	21:00	18655	18821	10	18	13
2017-10-02	06:00	21:00	21600	21669	7	19	14
2017-10-03	06:00	09:24	572	585	11	15	12
2017-10-04	09:18	21:00	18059	16734	12	19	13
2017-10-05	06:00	15:12	28252	28448	14	23	20

Table 5.30: Testing results for long time operation

Across the whole testing only 1 time window of 4 clips which are around 12 minutes together seem to caused problems. Especially 2 clips had a sure count ratio of below 10% concluding the results are inaccurate. After further investigation this seemed to be caused by an automatic parameter change of the camera. This videos had the highest amount of bees visible at the same time and caused the background to appear blueish instead of gray. This can also be observed by comparing dataset 3 and 4 which where both recorded with the same camera. While other videos in this longtime test are visually similar to dataset 3 the problematic videos are visually more similar to dataset 4. In this problematic situation the background shifts to blue which causes that the saturation does not work as segmentation channel since the blueish background becomes more saturated. To solve such a situation a segmentation in the hue channel can be used which was also done with dataset 4.

After reprocessing this 4 clips with a Hue segmentation the sure count ratio was higher. The conditions in these videos is comparable to dataset 4. This situation happened on 5.10 between 06:00 and 06:12. The table 5.30 contains the results of the fixed processing.

The difference are around 200 more bees in and 300 less out compared to the original results.

The important question is now how this can be solved for practical application. The ideal solution is to avoid such a situation. This could be done by configuring the camera to not adjust settings. Another possibility would be to automatically detected the problem by analyzing the sure count ratio and use different processing parameters when it drops low, which was effectively done in this test by rerunning the specific clips. The sure count ratio is an indicator to detect segmentation problems and a high sure count ratio is likely a more accurate counting result.

Credibility of the test

Also important is to verify the counting results of this test on how reasonable they are. Since acquiring a ground truth data for this amount of data is out of the question a bee researcher was asked for feedback who claimed the results seem plausible.

To better visualize the results of the 3 days with the full recording time from 06:00 to 21:00 are shown as plots: 5.9 for 30.9, 5.10 for 1.10 and 5.10 for 2.10.

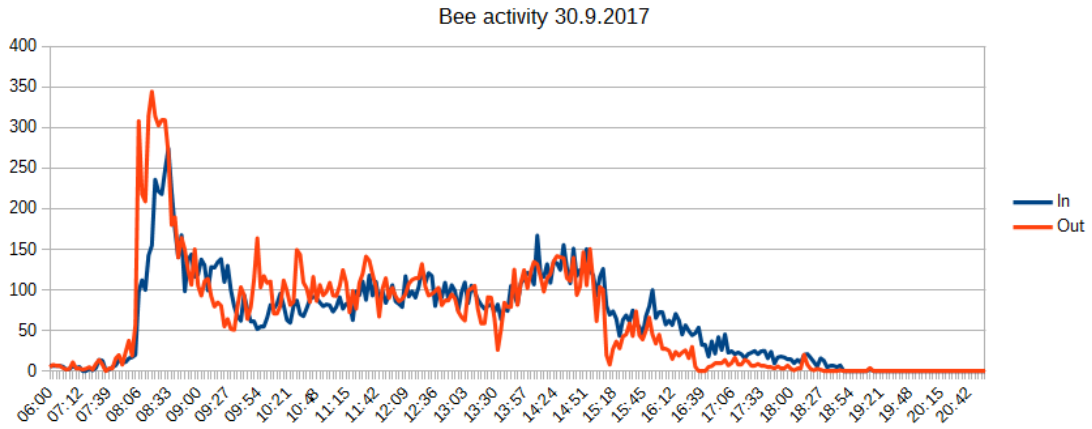


Figure 5.9: Bee activity on 30.9.2017 with 3 minute counting intervals.

Plot 5.9 has a recording hole between around 6:03 and 6:48, but the activity seems low in that time period so it can be neglected. Overall the counting results seem reasonable with similar patterns on all days and more bees are flying out in the morning while in the late afternoon and evening more bees come back. The overall daily count of bees exiting and entering is also quite similar when the whole day was recorded.

5.6.1 Runtime Breakdown

The runtime of the following program stages is measured:

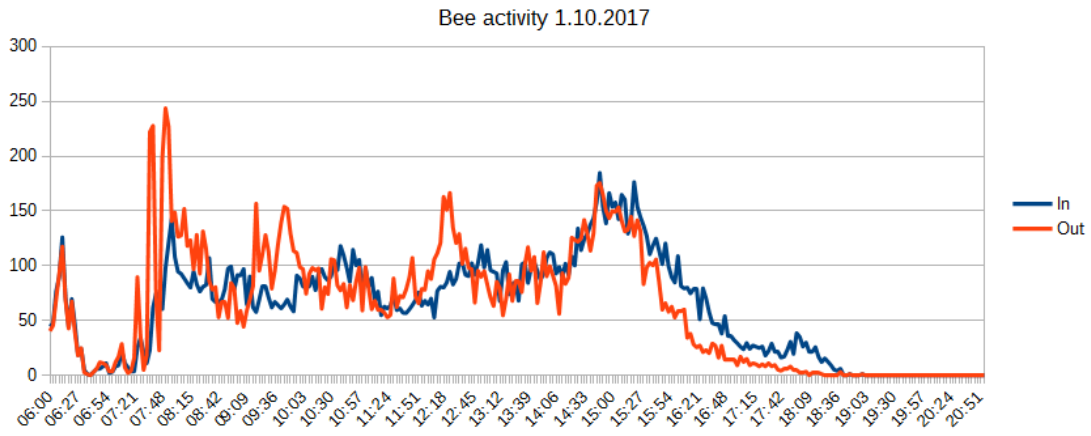


Figure 5.10: Bee activity on 1.10.2017 with 3 minute counting intervals

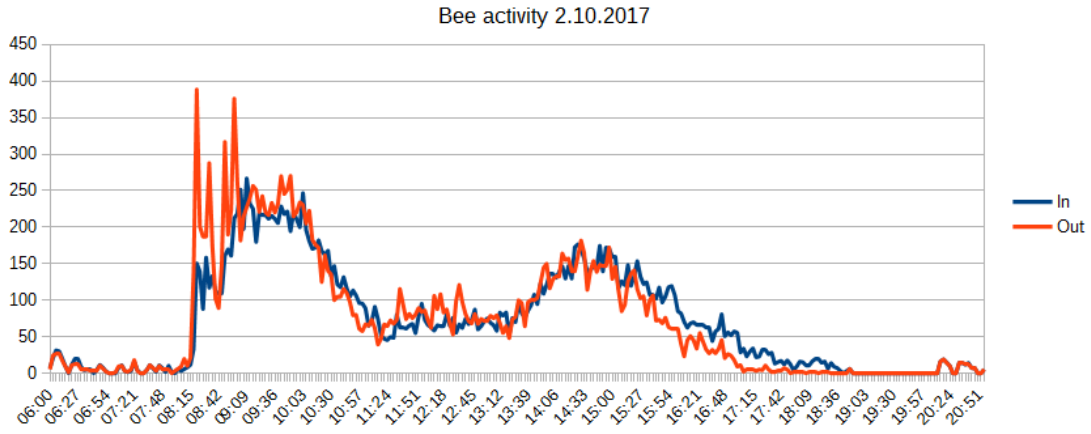


Figure 5.11: Bee activity on 2.10.2017 with 3 minute counting intervals

- Binary Threshold
- Median Filtering
- Morphological Opening
- findContours()
- K-Means
- Matching Stage

Binary thresholding, median filtering and morphological opening are the necessary image detection steps to segment the image. *findContours()* is an OpenCV method that can be

used to detect pixel groups of same color in the binary image. In the case of this program each separate cluster of white pixels is detected. The k-means algorithm is only needed when a detected blob is larger than the defined maximum area of a bee and therefore the blob needs to be split into multiple bees. On video sequences where there is no bee clustering the k-means algorithm is not needed and therefore the runtime of this stage will be faster. The matching stage time is measured as the time that is needed to match the detected bee centers in a new frame with the old detections. This stage optionally uses the munkres algorithm if a nearest neighbor matching is ambiguous.

The runtime testing will be performed on datasets 1, 3 and 4 with different resolutions.

Datasets 1 and 3 have a far lower bee activity than dataset 4 and are expected to have a different spread of runtimes percentages between the different stages and algorithms. The overall percentage does not equal to 100% as only the specific algorithms or stages are measured. All timings are measured in milliseconds (ms).

Stage	Time 100%	Percentage 100%	Time 50%	Percentage 50%
BinaryThreshold	0.177	0.170%	0.026	0.320%
Median Filter	25.587	25.230%	0.568	6.930%
Morphology	71.228	70.190%	5.225	63.570%
Find Contours	0.391	0.380%	0.106	1.280%
K-Means	0.118	0.120%	0.100	1.170%
Matching	0.005	0.010%	0.005	0.060%
Total	108.516	-	8.228	-

Table 5.31: Runtimes of different program stages in absolute values and percentages of dataset 1 with 100% and 50% resolution of 1416x540 and results averaged over all test sequences of the dataset

Table 5.31 shows the runtimes of different program stages on dataset 1. The results are averaged over all test sequences and both full resolution and half resolution have been tested. In the case of this dataset with both resolutions the morphology needs the most processing time with around 70%. The biggest difference that can be observed between changing the resolutions is that the median filtering goes down from 25% of the runtime to around 7%. Dataset 1 uses the smallest filter kernel for median filtering out of the datasets as there is not a lot noise and close to zero bee clustering. This means a smaller filter kernel is sufficient and explains the low runtime of the median filtering stage compared to other datasets which will be shown in the text tables.

The runtime breakdown of dataset 3 which is shown in table 5.32, shows that morphology takes the highest processing time of all stages with 68% but decreases to 46% and 17% of the runtime when reducing the resolution. The median filters relative amount of the runtime increases when reducing resolution concluding it's runtime scales less with resolution than morphology. On 25% resolution the median filter is the most performance heavy operation. The k-means algorithm does only take between 1.5% and 5% of the

Stage	Time 100%	Perc. 100%	Time 50%	Perc. 50%	Time 25%	Perc. 25%
BinaryThreshold	0.689	0.210%	0.153	0.260%	0.024	0.230%
Median Filter	81.448	25.200%	23.134	39.570%	3.171	30.020%
Morphology	220.788	68.318%	27.062	46.280%	1.884	17.840%
Find Contours	1.532	0.470%	0.271	0.460%	0.087	0.820%
K-Means	4.853	1.490%	1.875	3.220%	0.551	5.230%
Matching	0.009	0.000%	0.012	0.020%	0.004	0.040%
Total	323.196	-	58.474	-	10.563	-

Table 5.32: Runtimes of different program stages in absolute values and percentages of dataset 3 with 100%, 50% and 50% resolution of 1920x1080 and results averaged over all test sequences of the dataset

processing time with this dataset which is expected as the bee activity is lower than other datasets and clusters appear less often.

Stage	Time 100%	Perc. 100%	Time 50%	Perc. 50%	Time 25%	Perc. 25%
BinaryThreshold	0.501	0.100%	0.071	0.100%	0.014	0.100%
Median Filter	41.714	8.000%	25.256	36.160%	3.513	23.430%
Morphology	405.948	77.840%	23.494	33.640%	2.003	13.390%
Find Contours	1.444	0.280%	0.318	0.460%	0.129	0.860%
K-Means	51.429	9.770%	13.214	18.490%	3.449	22.430%
Matching	0.049	0.010%	0.047	0.070%	0.049	0.330%
Total	522.049	-	70.155	-	14.602	-

Table 5.33: Runtimes of different program stages in absolute values and percentages of dataset 4 with 100%, 50% and 25% resolution of 1920x1080 and results averaged over all test sequences of the dataset

In table 5.33 the runtime breakdown of dataset 4 is shown. As with the other datasets the morphology is the most expensive operation when using full resolution. On 50% resolution median filtering and morphology are about equal, while when using 25% resolution the median filter is more expensive. Dataset 4 contains the highest amount of bees and clusters of the 4 datasets which causes the k-means algorithm to need more runtime than in the other datasets. Dataset 4 can be considered as extreme data and is supposedly the worst case scenario that happened during recording. This evaluation is important as it showns that even with heavy clustering the performance does not take a serious hit as the k-means algorithm still only takes around 20% of the runtime even in such difficult scenario.

The most expensive operation with higher resolutions is the morphology which scales the most with resolution. Reducing the structure element is not a viable option as the size of the structure element is tied to the bee size. Reducing the size of the structure element will create less accurate shapes of the bee bodies which further decreases the accuracy of

the area estimation and makes it harder to separate clusters. The only way to counter the high runtime of the morphology is to generally work with a lower resolution. The median filter is second most expensive operations. Reducing the resolution does also improve the runtime but the difference is smaller than with the morphology. The size of the median filter kernel is not tied to the size of the bees but is needed to create a clearer image for the morphology stage by filling up holes so the bee shapes can be better extracted by the morphology. If bees have holes or are split in parts after thresholding they can get removed during the morphology stage. The median filtering counters this problem and is therefore needed, but reducing the size of the median filter kernel is an option as it is not tied to the size of bees but the quality of the initial segmentation with a binary threshold. The k-means algorithm is not always needed and only called when there are clusters in the image. This means the needed runtime for k-means scales with the amount and size of clusters that appear after the segmentation stage. Also in extreme situations as with dataset 4 the runtime of k-means is below 25% of the total runtime of the algorithm. Binary Thresholding, *findContours()* and the matching stage are all below 2% of the runtime and thus are not worth to be considered when trying to improve the runtime.

5.7 Summary

The overall error rates of all datasets are summed up in table 5.34. Dataset 1 has been recorded with 1416x540 resolution, the others were recorded at 1920x1080 (full-HD) resolution. As counting method the weighted average is used as in all cases it has equal or better accuracy than the median.

Dataset	Resolution	Errors	Total counting events	Error rate	FPS
Dataset 4	100%	83	1327	**6.25%**	2
Dataset 1	100%	7	305	**2.3%**	10
Dataset 3	50%	1	73	**1.37%**	17
Dataset 4	50%	109	1327	**8.21%**	14
Dataset 1	50%	6	305	**1.97%**	124
Dataset 3	25%	2	73	**2.74%**	98
Dataset 4	25%	91	1327	**6.86%**	67

Table 5.34: Overall error rates for datasets 1,2 and 4 with different resolutions and the average frames per second needed for processing

On datasets 1,3 and 4 the overall counting accuracy is 8.21% or below. Dataset 2 has to be considered as failure with an error rate of 68% concluding the quality of a Raspberry Pi camera module is not good enough for this application. Dataset 4 can be considered as extreme data and is composed of the supposedly most difficult situations found in all of the available recordings, still an error rate of only 8.21% in total could be reached. The phenomenon hat dataset 4 has the worst results with 50% resolution and better results with 25% and 100% resolution can be explained with the randomness of the k-means

initialization. Dataset 4 contains the most clustering of all tested datasets and therefore this has the most influence there.

Comparison with State of the Art

As other publication use different recording setups and test data it is not possible to make an unbiased objective comparison of results. A few numbers for comparison are given to show the competitiveness of the results from this thesis:

Kulyukin et. al[KR16] count the bees currently at the landing pad without making a distinction between in and out. Their evaluation shows an accuracy of 85.5% with a white landing pad. This equals to an error rate of 14.5%.

Tu et. al.[THKA16] state a regression statistic (R^2) of 0.953 for measuring in-activity and 0.888 for out-activity of honey bees.

CHAPTER 6

Conclusion

The problem of counting bees at the hive entrance has been solved with different setups. A differentiation that can be made is between 3D tracking where the bees are recorded while flying to or from the hive and the use of an entrance tunnel with 2D tracking. This has a lot of influence on the algorithms that can be used for video analysis as there are different problems that have to be handled. The 2D setup used with this thesis does simplify the problem further by ensuring some constraints:

First the whole tracking problem is purely 2D-dimensional as bees cannot move over each other as the available space beneath the glass plate is not big enough. This avoids occlusions in all cases and a bee is always visible when it is inside the recorded area. Also the artificial tunnel at the entrance allows to chose a background that distinguishes well from the bees. This is not the case when bees should be tracked inside the hive where a specific background cannot easily be chosen. The LED lights together with closed off walls create a very constant lighting environment which allows to not require parameter changes over long time periods.

The proposed tracking method is specifically tailored to the used hardware setup but provides an average error rate of 6.86% in difficult situations while maintaining a framerate of around 67 fps. Using the HSV and HSL color models proved to be very effective at segmenting the bees from the background while being robust against illumination changes. Using morphological opening together with median filtering provides a segmentation that extracts the bee body shapes and still can work in real time. The morphology only works reliably because the bee movement is restricted to 2D. Morphological opening extracts the shapes of the bee bodies which then allows to estimate the number of a bees in a cluster by diving the area through the average bee size. With this estimation of bee number the k-means algorithm is able to separate bee clusters. With occlusions of bees this would not be reliable.

The used methods support each other. K-means would not give accurate results without morphology because of inaccurate bee areas. Morphology works better with applying a median filter before. The k-means algorithm optimizes cluster centers which are accurate to the center of a bee because of the elliptical body shapes.

As for tracking a position matching showed to be sufficient to achieve a simultaneous tracking of bees in the tracking area. The used algorithms create a program that is instant responsive and can start tracking at the second frame as there is no warm up time needed for building background models as other techniques would require. This means that the memory usage is low because only the current frame needs to be stored. After the bee positions have been extracted only this positions are needed when processing the next frame. Another advantage from this instant responsiveness is that video interruptions can be handled well. Of course information from lost frames cannot be recovered but short frame drops or image errors did have less impact on the tracking and counting algorithm that one would expect.

6.1 Future Work

The next step would be to do more practical tests and to create a prototype that can be tested in practice. This would require testing of the processing power of embedded devices if they can handle realtime performance with this algorithm and at what resolution. Other options could include streaming the video data and processing on a server. Further improvements to the hardware setup could also be done, this includes trying to optimize the background color and camera settings. For example trying a blue background and using the hue channel for segmentation. Also a darker background could improve the segmentation when using the saturation.

Another possible idea for a practical application is to use the ratio of sure and unsure counting events as a way to self-diagnose the quality of the algorithm and apply a automatic warning system that can detected bad counting accuracy which is likely caused by bad parameters or a very dirty scene.

As for the algorithm a possible improvement could be statistical analysis if there is a bias on which side more errors happen. This would require a lot of additional ground truth data though, which is very difficult and time consuming to obtain.

Another future option is to build an integration with another project that can classify if a bee is infected by a parasite. This algorithm estimates the location and can calculate the bounding ellipse of every bee currently visible. This allows to extract images of bees for further processing.

A lot of recent research in computer vision is based on deep learning. This includes publications for image segmentation and tracking. While deep learning has proved to be a solution for a classification problem on bees[Sch18], at the time of writing a deep learning based approach for segmentation or counting of bees does not seem to be a good solution especially when considering runtime and embedded hardware. In a few years

better algorithms or hardware could allow the use of a deep learning based approach that monitors the activity of bees in realtime on hardware directly mounted at the hive.

List of Figures

List of Tables

List of Algorithms

Bibliography

[AH09] Marcelo A Aizen and Lawrence D Harder. The global stock of domesticated honey bees is growing slower than agricultural demand for pollination. *Current Biology*, 2009.

[ALNZ14] Carlos Arteta, Victor Lempitsky, J Alison Noble, and Andrew Zisserman. Interactive object counting. In *European Conference on Computer Vision*. Springer, 2014.

[ASS+12] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, Sabine Süsstrunk, et al. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[AV07] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2007.

[BJE+08] Yannick Benezeth, Pierre-Marc Jodoin, Bruno Emile, Hélène Laurent, and Christophe Rosenberger. Review and evaluation of commonly-implemented background subtraction algorithms. In *International Conference on Pattern Recognition*. IEEE, 2008.

[BRB+09] Kristin Branson, Alice A Robie, John Bender, Pietro Perona, and Michael H Dickinson. High-throughput ethomics in large groups of drosophila. *Nature methods*, 2009.

[CGKMR13] Guillaume Chiron, Petra Gomez-Krämer, Michel Ménard, and Fabrice Requier. 3d tracking of honeybees enhanced by environmental context. In *International Conference on Image Analysis and Processing*. Springer, 2013.

[CGPP03] Rita Cucchiara, Costantino Grana, Massimo Piccardi, and Andrea Prati. Detecting moving objects, ghosts, and shadows in video streams. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 2003.

[Cho15] Wongun Choi. Near-online multi-target tracking with aggregated local flow descriptor. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[CMS08] Jason Campbell, Lily Mummert, and Rahul Sukthankar. Video monitoring of honey bee colonies at the hive entrance. *Visual observation & Analysis of Animal & Insect Behavior, ICPR*, 2008.

[CPK+18] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 2018.

[Cra13] Ethel Eva Crane. *The World History of Beekeeping and Honey Hunting*. Routledge, 2013.

[CRM03] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2003.

[CV09] Antoni B Chan and Nuno Vasconcelos. Bayesian poisson regression for crowd counting. In *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009.

[CYJL12] Chiu Chen, En-Cheng Yang, Joe-Air Jiang, and Ta-Te Lin. An imaging system for monitoring the in-and-out activity of honey bees. *Computers and Electronics in Agriculture*, 2012.

[EMMW+16] Fiona Edwards-Murphy, Michele Magno, Pádraig M Whelan, John O'Halloran, and Emanuel M Popovici. b+ wsn: Smart beehive with preliminary decision tree analysis for agriculture and honey bee health monitoring. *Computers and Electronics in Agriculture*, 2016.

[ESM+09] Jay D Evans, Claude Saegerman, Chris Mullin, Eric Haubruge, Bach Kim Nguyen, Maryann Frazier, Jim Frazier, Diana Cox-Foster, Yanping Chen, Robyn Underwood, et al. Colony collapse disorder: a descriptive study. *PloS one*, 2009.

[Fai13] Mark D. Fairchild. *Color Appearance Models*. John Wiley & Sons, 2013.

[FBADL16] Loïc Fagot-Bouquet, Romaric Audigier, Yoann Dhome, and Frédéric Lerasle. Improving multi-frame data association with sparse representations for robust near-online multi-object tracking. In *European Conference on Computer Vision*. Springer, 2016.

[FP02] David A Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[GW16] Rafael C Gonzalez and Richard E Woods. *Digital Image Processing*. Prentice hall, 2016.

[HCMB12] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *European Conference on Computer Vision*. Springer, 2012.

[HGDG18] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[HHD00] Ismail Haritaoglu, David Harwood, and Larry S Davis. W4: Real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.

[KBD04] Zia Khan, Tucker Balch, and Frank Dellaert. A rao-blackwellized particle filter for eigentracking. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2004*. IEEE, 2004.

[KBHA16] Hilke Kieritz, Stefan Becker, Wolfgang Hübner, and Michael Arens. Online multi-person tracking using integral channel features. In *2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2016.

[KLCR15] Chanho Kim, Fuxin Li, Arridhana Ciptadi, and James M. Rehg. Multiple hypothesis tracking revisited. ICCV, 2015.

[KMM10] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Forward-backward error: Automatic detection of tracking failures. In *20th International Conference on Pattern Recognition (ICPR)*. IEEE, 2010.

[KOC+14] Toshifumi Kimura, Mizue Ohashi, Karl Crailsheim, Thomas Schmickl, Ryuichi Okada, Gerald Radspieler, and Hidetoshi Ikeno. Development of a new method to track multiple honey bees with complex behaviors on a flat laboratory arena. *PloS one*, 2014.

[KOOI11] Toshifumi Kimura, Mizue Ohashi, Ryuichi Okada, and Hidetoshi Ikeno. A new approach for the simultaneous tracking of multiple honeybees for analysis of hive behavior. *Apidologie*, 2011.

[KR16] Vladimir Kulyukin and Sai Kiran Reka. A computer vision algorithm for omnidirectional bee counting at langstroth beehive entrances. In *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*, 2016.

[Kuh55] Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 1955.

[KVC+07] Alexandra-Maria Klein, Bernard E Vaissiere, James H Cane, Ingolf Steffan-Dewenter, Saul A Cunningham, Claire Kremen, and Teja Tscharntke. Importance of pollinators in changing landscapes for world crops. *Proceedings of the Royal Society of London B: Biological Sciences*, 2007.

[LD10] Zhe Lin and Larry S Davis. Shape-based human detection and segmentation via hierarchical part-template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[Llo82] Stuart Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 1982.

[LR07] Tim Landgraf and Raúl Rojas. Tracking honey bee dances from sparse optical flow fields. 2007.

[M+10] Marina Doris Meixner et al. 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*, 2010.

[MMG12] Anca Morar, Florica Moldoveanu, and Eduard Gröller. Image segmentation based on active contours without edges. In *2012 IEEE 8th International Conference on Intelligent Computer Communication and Processing*. IEEE, 2012.

[Mun57] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 1957.

[Pic04] Massimo Piccardi. Background subtraction techniques: a review. In *2004 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 2004.

[PP12] Brajesh Patel and Neelam Patel. Motion detection based on multi frame video under surveillance system. *International Journal of Computer Science and Network Security (IJCSNS)*, 2012.

[RBH13] Stephen Russell, Andrew B Barron, and David Harris. Dynamic modelling of honey bee (apis mellifera) colony growth and failure. 2013.

[RC10] Francis LW Ratnieks and Norman L Carreck. Clarity on honey bee collapse? *Science*, 2010.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-assisted Intervention*. Springer, 2015.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015.

[Sas07] Yutaka Sasaki. The truth of the f-measure. 2007.

[SAS17] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. *arXiv preprint arXiv:1701.01909*, 2017.

[SCBNF08] Concetto Spampinato, Yun-Heh Chen-Burger, Gayathri Nadarajan, and Robert B Fisher. Detecting, tracking and counting fish in low quality unconstrained underwater videos. *VISAPP*, 2008.

[Sch18] Stefan Schurischuster. Image analysis approaches for parasite detection on honeybees. Master's thesis, TU Wien, 2018.

[Ser83] Jean Serra. *Image analysis and mathematical morphology*. Academic Press, Inc., 1983.

[SG99] Chris Stauffer and W Eric L Grimson. Adaptive background mixture models for real-time tracking. In *cvpr*. IEEE, 1999.

[SGN00] Diana Sammataro, Uri Gerson, and Glen Needham. Parasitic mites of honey bees: life history, implications, and impact. *Annual Review of Entomology*, 2000.

[SHL18] David Stutz, Alexander Hermans, and Bastian Leibe. Superpixels: an evaluation of the state-of-the-art. *Computer Vision and Image Understanding*, 2018.

[SRRK18] Stefan Schurischuster, Beatriz Remeseiro, Petia Radeva, and Martin Kampel. A preliminary study of image analysis for parasite detection on honey bees. In *International Conference Image Analysis and Recognition*. Springer, 2018.

[SZKL16] Stefan Schurischuster, Sebastian Zambanini, Martin Kampel, and Benjamin Lamp. Sensor study for monitoring varroa mites on honey bees (apis mellifera). In *Visual Observation and Analysis of Vertebrate and Insect Behavior Workshop*, 2016.

[TAAS16] Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. Multi-person tracking by multicut and deep matching. In *European Conference on Computer Vision*. Springer, 2016.

[THKA16] Gang Jun Tu, Mikkel Kragh Hansen, Per Kryger, and Peter Ahrendt. Automatic behaviour analysis system for honeybees using computer vision. *Computers and Electronics in Agriculture*, 2016.

[VPR13] Carl Vondrick, Donald Patterson, and Deva Ramanan. Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision*, 2013.

[WRHS13] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proceedings of the IEEE International Conference on Computer Vision*, 2013.

[YBFU15] Jian Yao, Marko Boben, Sanja Fidler, and Raquel Urtasun. Real-time coarse-to-fine topologically preserving segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[YC15] Cheng Yang and John Collins. A model for honey bee tracking on 2d video. In *International Conference on Image and Vision Computing New Zealand (IVCNZ)*. IEEE, 2015.

[YJS06] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *Acm Computing Surveys (CSUR)*, 2006.

[ZLWY15] Cong Zhang, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang. Cross-scene crowd counting via deep convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[ZZC+16] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016.